

DATA ANALYTICS ON IT SECTOR IN INDIA USING BIG DATA

**Minor Project report submitted in partial fulfillment of the requirement
for the award of the Degree of**

Bachelor of Technology

In

Information Technology

UNDER SUPERVISION OF

Dr. Bhoomi Gupta

BY

Archit Gupta

Roll No.:04114803114

Tanya Gupta

Roll No.:04214803114



**Maharaja Agrasen Institute of Technology
Affiliated to Guru Gobind Singh Indraprastha University
Sector 22, Rohini, Delhi-110086
(2014-2018)**

DATA ANALYTICS ON IT SECTOR IN INDIA USING BIG DATA

**Minor Project report submitted in partial fulfillment of the requirement
for the award of the Degree of**

Bachelor of Technology

In

Information Technology

UNDER SUPERVISION OF

Dr. Bhoomi Gupta

BY

Archit Gupta

Roll No.:04114803114

Tanya Gupta

Roll No.:04214803114



**Maharaja Agrasen Institute of Technology
Affiliated to Guru Gobind Singh Indraprastha University
Sector 22, Rohini, Delhi-110086
(2014-2018)**

Certificate

This is to certify that the project report entitled Data Analytics on IT sector in India using Big Data being submitted by Mr. Archit Gupta in partial fulfillment for the award of the Degree of Bachelor of Technology in IT to the Maharaja Agrasen Institute of Technology is a record of bonafide work carried out by him under my guidance and supervision. The results embodied in this project report have not been submitted to any other University or Institute for the award of any Degree or Diploma.

(Head of the Department)

Guide Name: Dr. Bhoomi Gupta

Designation: Assistant Professor

ABSTRACT

This is an era of Big Data. Big Data is driving radical changes in traditional data analysis platforms. To perform any kind of analysis on such voluminous and complex data, scaling up the hardware platforms becomes imminent and choosing the right hardware/software platforms becomes a crucial decision if the user's requirements are to be satisfied in a reasonable amount of time.

Apache's Hadoop framework is essentially a mechanism for analyzing huge datasets, which do not necessarily need to be housed in a datastore. Hadoop abstracts MapReduce's massive data-analysis engine, making it more accessible to developers. Hadoop scales out to myriad nodes and can handle all of the activity and coordination related to data sorting.

As a conceptual framework for processing huge data sets, MapReduce is highly optimized for distributed problem-solving using a large number of computers. The framework consists of two functions, as its name implies. The map function is designed to take a large data input and divide it into smaller pieces, which it then hands off to other processes that can do something with it. The reduce function digests the individual answers collected by map and renders them to a final output.

This project utilizes the results of MapReduce & HDFS architecture to analyze the IT sector in India. Processing of data of registered companies being a heavy task for normal processors is being processed using Hadoop technology in this project which has the capability to process huge datasets using **Cloudera and PigScripts**.

Results fetched after analyzation are being utilized to plot the structure of IT sector which would have been costly and difficult using normal processors and memory architecture.

ACKNOWLEDGEMENT

With due respect, We express my profound gratitude and indebtedness to **Dr. Bhoomi Gupta**, Assistant Professor, Maharaja Agrasen Institute of Technology, Delhi for her ample guidance and assistance in building this project and for her inspiring intellectual guidance, constructive criticism and valuable suggestion throughout the project work.

We are also very grateful to faculty of the Information Technology Department, **Maharaja Agrasen Institute of Technology, Delhi** for their ready support.

Date – 23/10/2017

New Delhi

Submitted By:-

Archit Gupta

04114803114

Tanya Gupta

04214803114

CONTENTS

Abstract	
Acknowledgement	
Contents	
Introduction	8
Types of Data	11
Hadoop – Big Data Tool	12
Hadoop Ecosystem	17
BigData Analysis	19
Code snippets	23
Conclusion	33
Future Enhancements	34
References	35

LIST OF FIGURES

S.No.	Figure No.	Screenshot Description	Page Number
01	Figure 1	Big Data Characteristics	09
02	Figure 2	Big Data Customer Scenario	10
03	Figure 3	Hadoop	13
04	Figure 4	Architecture of Hadoop	16
05	Figure 5	HADOOP ECO SYSTEM	18
06	Figure 6	Sqoop	20
07	Figure 7	Graph depicting the results	33

1. Introduction

1.1 Big Data

Big data is a popular term used to describe the exponential growth and availability of data, both structured, unstructured and Semi Structured. And big data may be as important to business – and society – as the Internet has become.

- Lots of Data (Terabytes or Gigabytes)
- Big data is the term for a collection of data sets so large and complex that it becomes difficult to process using on-hand database management tools or traditional data processing applications. The challenges include capture, storage, search, sharing, transfer, analysis, and visualization.
- Systems / Enterprises, Internet users, generate huge amount of data from Gigabytes to and even Terabytes of information.

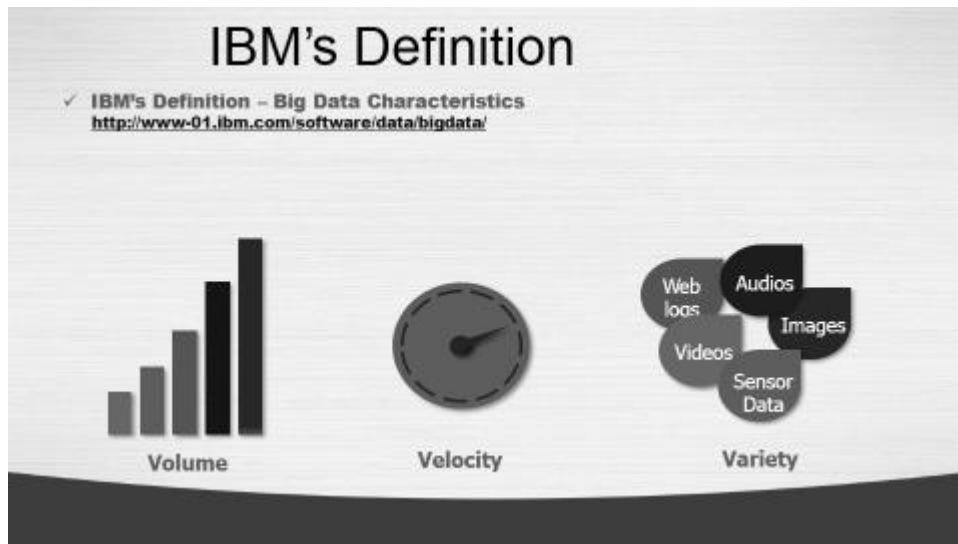




Fig. 1

- **Volume.** Many factors contribute to the increase in data volume. Transaction-based data stored through the years. Unstructured data streaming in from social media. Increasing amounts of sensor and machine-to-machine data being collected. In the

past, excessive data volume was a storage issue. But with decreasing storage costs, other issues emerge, including how to determine relevance within large data volumes and how to use analytics to create value from relevant data.

- **Velocity.** Data is streaming in at unprecedented speed and must be dealt with in a timely manner. RFID tags, sensors and smart metering are driving the need to deal with torrents of data in near-real time. Reacting quickly enough to deal with data velocity is a challenge for most organizations.
- **Variety.** Data today comes in all types of formats. Structured, numeric data in traditional databases. Information created from line-of-business applications. Unstructured text documents, email, video, audio, stock ticker data and financial transactions. Managing, merging and governing different varieties of data is something many organizations still grapple with.

Common Big Data Customer Scenarios

- ✓ **Web and e-tailing**
 - ✓ Recommendation Engines
 - ✓ Ad Targeting
 - ✓ Search Quality
 - ✓ Abuse and Click Fraud Detection
- ✓ **Telecommunications**
 - ✓ Customer Churn Prevention
 - ✓ Network Performance Optimization
 - ✓ Calling Data Record (CDR) Analysis
 - ✓ Analyzing Network to Predict Failure

<http://wiki.apache.org/hadoop/PoweredBy>

Slide

Fig. 2

1.2 Big Data Challenges

- **Need for speed** - Now This Time hypercompetitive business environment, companies not only have to find and analyze the relevant data they need, they must find it quickly. Visualization helps organizations perform analyses and make decisions much more rapidly, but the challenge is going through the sheer volumes of data and accessing the level of detail needed, all at a high speed.
- **Data quality** - Analyze data quickly and put it in the proper context for the audience that will be consuming the information, the value of data for decision-making purposes if the data is not accurate or timely. This is a challenge with any data analysis, but when considering the volumes of information involved in big data projects, it becomes even more pronounced.
- **Understanding the data** - It takes a lot of understanding to get data in the right shape so that you can use visualization as part of data analysis. For example, if the data comes from social media content, you need to know who the user is in a general sense – such as a customer using a particular set of products – and understand what it is you're trying to visualize out of the data. Without some sort of context, visualization tools are likely to be of less value to the user.
- **Handling the volume** - The most obvious challenge associated with big data is simply storing and analyzing all that information. In its Digital Universe report, IDC estimates that the amount of information stored in the world's IT systems is doubling about every two years. By 2020, the total amount will be enough to fill a stack of tablets that reaches from the earth to the moon 6.6 times. And enterprises have responsibility or liability for about 85 percent of that information.

2. Types of Data

2.1 Unstructured Data

Unstructured data files often include text and multimedia content. Examples include e-mail messages, word processing documents, videos, photos, audio files, presentations, webpages and many other kinds of business documents. While these sorts of files may have an internal structure, they are still considered "unstructured" because the data they contain doesn't fit neatly in a database. Experts estimate that 80 to 90 percent of the data in any organization is unstructured. And the amount of unstructured data in enterprises is growing significantly often many times faster than structured databases are growing.

Unstructured data is all those things that can't be so readily classified and fit into a neat box like photos and graphic images, videos, streaming instrument data, webpages, PDF files, PowerPoint presentations, emails, blog entries, wikis and word processing documents.

2.2 Structured data

Data that resides in a fixed field within a record or file is called structured data. This includes data contained in relational databases and spreadsheets.

Structured data first depends on creating a data model – a model of the types of business data that will be recorded and how they will be stored, processed and accessed. This includes defining what fields of data will be stored and how that data will be stored: data type (numeric, currency, alphabetic, name, date, address).

2.3 Semi-Structured Data

Semi-structured data is a cross between the two. It is a type of structured data, but lacks the strict data model structure. For eg, XML.

3. HADOOP –BIGDATA TOOL

3.1 Introduction to Hadoop

Apache Hadoop is an open-source software framework written in Java for distributed storage and distributed processing of very large data sets on computer clusters built from commodity hardware. All the modules in Hadoop are designed with a fundamental assumption that hardware failures (of individual machines, or racks of machines) are commonplace and thus should be automatically handled in software by the framework.

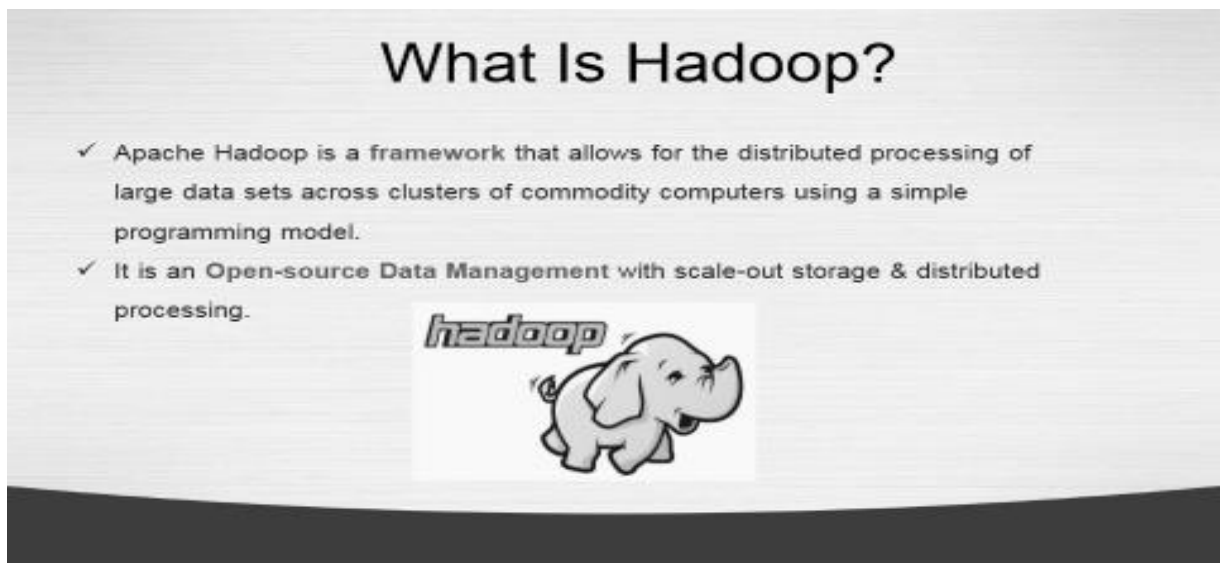


Fig. 3

- Hadoop provides a reliable shared storage (HDFS) and analysis system (MapReduce).
- Hadoop is highly scalable and unlike the relational databases, Hadoop scales linearly. Due to linear scale, a Hadoop Cluster can contain tens, hundreds, or even thousands of servers.
- Hadoop is very cost effective as it can work with commodity hardware and does not require expensive high-end hardware.

3.2 History of Hadoop

Google invented the basic frameworks that constitute what is today popularly called as Hadoop. They faced the future first with the problem of handling billions of searches and indexing millions of web pages. When they could not find any large scale, distributed, scalable computing platforms for their needs, they just went ahead and created their own.

Doug Cutting was inspired by Google's white papers and decided to create an open source project called "Hadoop".

Doug Cutting, Cloudera's Chief Architect, helped create Hadoop out of necessity as data from the web exploded, and grew far beyond the ability of traditional systems to handle it.

Yahoo further contributed to this project and played a key role in developing Hadoop for enterprise applications. Since then many companies such as Facebook, LinkedIn, ebay, Horton works, Cloudera etc have contributed to the Hadoop project.

3.3 Advantages of Hadoop

- ✓ **Data Size and Data Diversity** - When you are dealing with huge volumes of data coming from various sources and in a variety of formats then you can say that you are dealing with Big Data. In this case, Hadoop is the right technology for you.
- ✓ **Future Planning** - It is all about getting ready for challenges you may face in future. If you anticipate Hadoop as a future need then you should plan accordingly. To implement Hadoop on your data you should first understand the level of complexity of data and the rate with which it is going to grow. So, you need a cluster planning. It may begin with building a small or medium cluster in your industry as per data (in GBs or few TBs) available at present and scale up your cluster in future depending on the growth of your data.

- ✓ **Multiple Frameworks for Big Data** - There are various tools for various purposes. Hadoop can be integrated with multiple analytic tools to get the best out of it, like Mahout for Machine-Learning, R and Python for Analytics and visualization, Python, Spark for real time processing, MongoDB and Hbase for No sql database, Pentaho for BI etc.
- ✓ **Lifetime Data Availability** - When you want your data to be live and running forever, it can be achieved using Hadoop's scalability. There is no limit to the size of cluster that you can have. You can increase the size anytime as per your need by adding data nodes to it with minimal cost.
- ✓ **Cost Effective** - Hadoop also offers a cost effective storage solution for businesses' exploding data sets. The problem with traditional relational database management systems is that it is extremely cost prohibitive to scale to such a degree in order to process such massive volumes of data. In an effort to reduce costs, many companies in the past would have had to down-sample data and classify it based on certain assumptions as to which data was the most valuable. The raw data would be deleted, as it would be too cost-prohibitive to keep. While this approach may have worked in the short term, this meant that when business priorities changed, the complete raw data set was not available, as it was too expensive to store.
- ✓ **Scalable** - Hadoop is a highly scalable storage platform, because it can store and distribute very large data sets across hundreds of inexpensive servers that operate in parallel. Unlike traditional relational database systems (RDBMS) that can't scale to process large amounts of data, Hadoop enables businesses to run applications on thousands of nodes involving many thousands of terabytes of data.
- ✓ **Some Other Uses**
 - Log file processing
 - Analysis of Text, Image, Audio, & Video content
 - Recommendation systems like in E-Commerce Websites

3.4 Architecture of Hadoop

Hadoop Server Roles

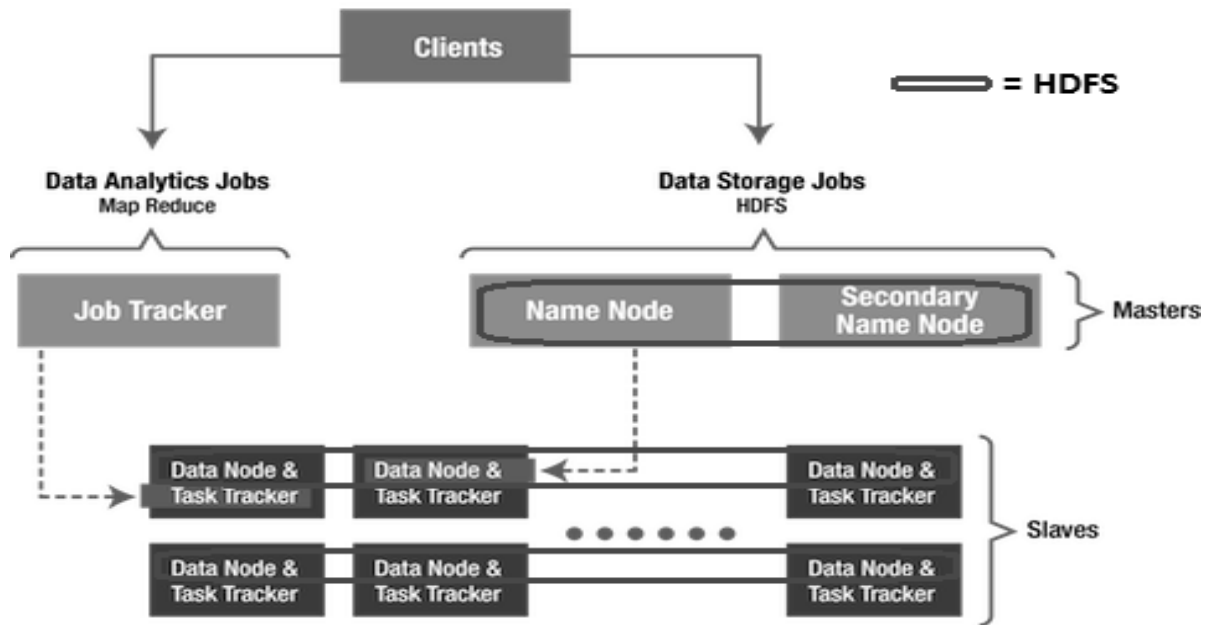


Fig. 4

- Hadoop works in a master-worker / master-slave fashion.
- Hadoop has two core components: HDFS and MapReduce.
- **HDFS (Hadoop Distributed File System)** offers a highly reliable and distributed storage, and ensures reliability, even on a commodity hardware, by replicating the data across multiple nodes. Unlike a regular file system, when data is pushed to HDFS, it will automatically split into multiple blocks (configurable parameter) and stores/replicates the data across various datanodes. This ensures high availability and fault tolerance. HDFS can be deployed on a broad spectrum of machines that support Java. Though one can run several DataNodes on a single machine, but in the practical world, these DataNodes are spread across various machines.

- **MapReduce** offers an analysis system which can perform complex computations on large datasets. This component is responsible for performing all the computations and works by breaking down a large complex computation into multiple tasks and assigns those to individual worker/slave nodes and takes care of coordination and consolidation of results. For processing large sets of data MR comes into the picture. The programmers will write MR applications that could be suitable for their business scenarios. Programmers have to understand the MR working flow and according to the flow, applications will be developed and deployed across Hadoop clusters. Hadoop built on Java APIs and it provides some MR APIs that is going to deal with parallel computing across nodes.
- The master contains the Namenode and Job Tracker components.
 - **Namenode** holds the information about all the other nodes in the Hadoop Cluster, files present in the cluster, constituent blocks of files and their locations in the cluster, and other information useful for the operation of the Hadoop Cluster.
 - **Job Tracker** keeps track of the individual tasks/jobs assigned to each of the nodes and coordinates the exchange of information and results.
- Each Worker / Slave contains the Task Tracker and a Datanode components.
 - **Task Tracker** is responsible for running the task / computation assigned to it.
 - **Datanode** is responsible for holding the data.
- The computers present in the cluster can be present in any location and there is no dependency on the location of the physical server.

4. HADOOP ECO SYSTEM

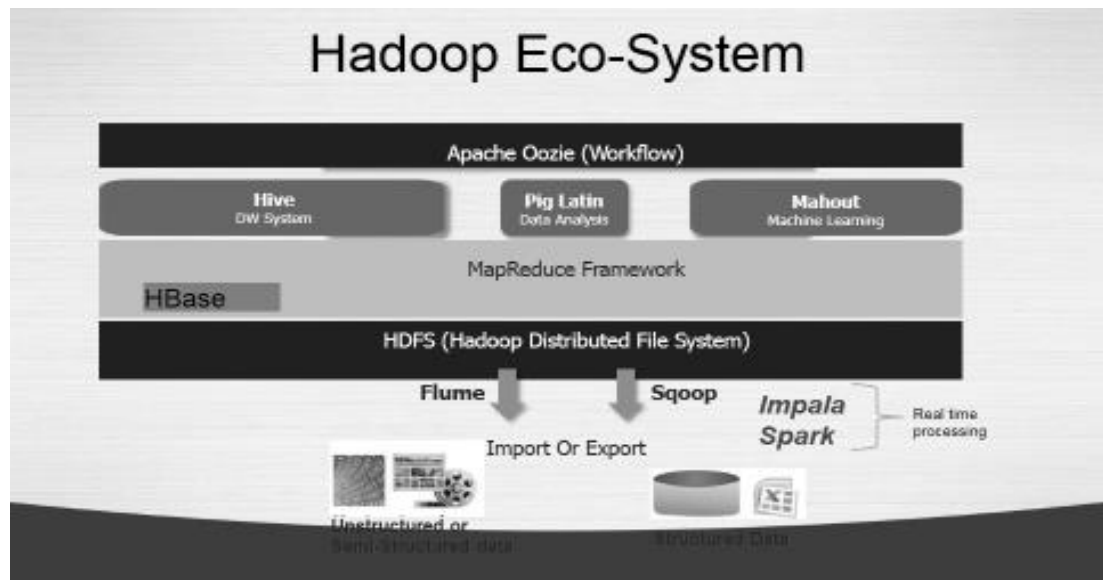


Fig 5

4.1 HIVE

Hive provides a warehouse structure and SQL-like access for data in HDFS and other Hadoop input sources (e.g. Amazon S3). Hive's query language, HiveQL, compiles to MapReduce. It also allows user defined functions (UDFs). Hive is widely used, and has itself become a "sub-platform" in the Hadoop ecosystem.

4.2 PIG

Pig is a framework consisting of a high-level scripting language (Pig Latin) and a run-time environment that allows users to execute MapReduce on a Hadoop cluster. Like HiveQL in Hive, Pig Latin is a higher-level language that compiles to MapReduce.

4.3 MAHOUT

Mahout is a scalable machine-learning and data mining library. There are currently four main groups of algorithms in Mahout:

- recommendations, also known as collective filtering

- classification, also known as categorization
- clustering
- frequent item set mining, also known as parallel frequent pattern mining

4.4 MapReduce

- The MapReduce paradigm for parallel processing comprises two sequential steps: map and reduce.
- In the map phase, the input is a set of key-value pairs and the desired function is executed over each key/value pair in order to generate a set of intermediate key/value pairs.

4.5 HBASE

Based on Google's Bigtable, HBase "is an open-source, distributed, versioned, column-oriented store" that sits on top of HDFS. HBase is column-based rather than row-based, which enables high-speed execution of operations performed over similar values across massive data sets, e.g. read/write operations that involve all rows but only a small subset of all columns. HBase does not provide its own query or scripting language, but is accessible through Java, Thrift, and REST APIs.

4.6 SGOOP

Sqoop ("SQL-to-Hadoop") is a tool which transfers data in both directions between relational systems and HDFS or other Hadoop data stores, e.g. Hive or HBase.

According to the Sqoop blog, "You can use Sqoop to import data from external structured datastores into Hadoop Distributed File System or related systems like Hive and HBase. Conversely, Sqoop can be used to extract data from Hadoop and export it to external structured datastores such as relational databases and enterprise data warehouses."

5. BIG DATA ANALYSIS

Using Sqoop and MySql and HDFS we can handle the data and Transfer the data MySql to HDFS and analyse the Data.

5.1 Working of Sqoop

Sqoop is a tool designed to transfer data between Hadoop and relational database servers. 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. It is provided by the Apache Software Foundation. SQL to Hadoop and Hadoop to SQL. Through Sqoop we can Import full table, part of table and selected value into Hadoop.

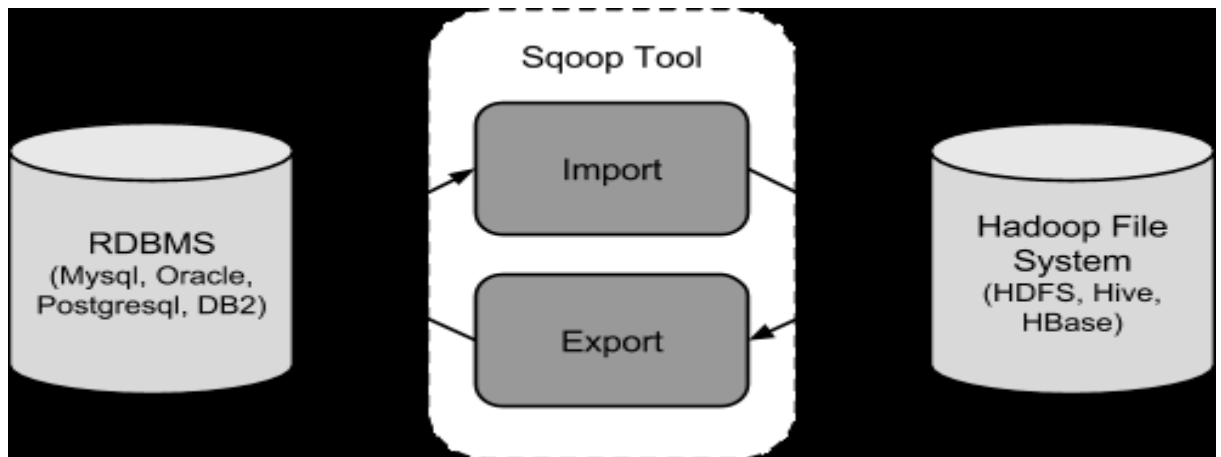


Fig. 6

- **Sqoop Import** - The import tool imports individual tables from RDBMS to HDFS. Each row in a table is treated as a record in HDFS. All records are stored as text data in text files or as binary data in Avro and Sequence files.
- **Sqoop Export** - The export tool exports a set of files from HDFS back to an RDBMS. The files given as input to Sqoop contain records, which are called as

rows in table. Those are read and parsed into a set of records and delimited with user-specified delimiter

5.2 Step To Install and Create Table on Operating system

- ✓ On Centos or Ubuntu, install MySQL.
- ✓ Create Database.
- ✓ Create Table schema into MYSQL.
- ✓ Insert the values into table one by one or Dump full data into MYSQL Table.
- ✓ Now your full data into your table.

Let us take an example of 1 tables named as **H_info** which are in a database called **HadoopGyan** in a MySQL database server.

H_info

ID	Name	Country
101	Map reduce	USA
102	Pig	UK
103	Hive	India
104	HBase	Africa

5.3 Importing Full Table to HDFS

This string will connect to a MySQL database named Hadoopgyan. The connect string will be used on TaskTracker nodes throughout MapReduce cluster; if specify the literal name localhost, each node will connect to a different database (or more likely, no database at all). When full hostname or IP address of the database host were used they were seen by all remote nodes.

If not authenticate against the database before you can access it, then can use the --username and --password or -P parameters to supply a username and a password to the database.

Sqoop automatically supports several databases, including MySQL. Connect strings beginning with jdbc:mysql:// are handled automatically in Sqoop.

5.4 Sqoop Import

- `$ Sqoop Import --connect jdbc:mysql://local host/Database name -- username -- table name --target-dir /HDFS -M 1`
- `$ Sqoop Import --connect jdbc:mysql://local host/ Hadoopgyan --username root -- table H_info --Target-dir /user/cloudera/HadoopGyan -- m 1`

Through above command our table H_info import to the hadoop (HDFS)

5.5 Sqoop Export

Command to Export data Into MYSQL From HDFS this is called Export in SQOOP

- `Sqoop export --connect jdbc:mysql://localhost:/portnumber/databasename -- username root -P --table name --export-dir "/path of table "`

Or

- `$ sqoop export --connect jdbc:mysql://localhost/Portnumber/databasename -- username root -Password Password --table name --export-dir "Path of table"`

6. CODE SNIPPETS & SCREENSHOTS

putting data on hdfs architecture

```
hadoop fs -put /home/cloudera/Desktop/Data.csv /user/cloudera/newdata.txt;
```

showing Data file on host

```
http://localhost:50070/explorer.html#/user/cloudera
```

opening pig

```
pig -x local;
```

Loading full data in a variable

```
variable = Load '/home/cloudera/Desktop/Data.csv' using PigStorage(',') as
(CompanyName:chararray,CompanyNumber:int,RegAddressCareOf:chararray,RegAdd
ressPOBox:chararray,RegAddressAddressLine1:chararray,RegAddressAddressLine2:ch
ararray,RegAddressPostTown:chararray,RegAddressCounty:chararray,RegAddressCou
ntry:chararray,RegAddressPostCode:chararray,CompanyCategory:chararray,CompanyS
tatus:chararray,CountryOfOrigin:chararray,DissolutionDate:chararray,IncorporationDat
e:chararray,AccountsAccountRefDay:int,AccountsAccountRefMonth:chararray,Accoun
tsNextDueDate:chararray,AccountsLastMadeUpDate:chararray,AccountsAccountCateg
ory:chararray>ReturnsNextDueDate:chararray>ReturnsLastMadeUpDate:chararray,Mort
gagesNumMortCharges:int,MortgagesNumMortOutstanding:int,MortgagesNumMortPa
rtSatisfied:chararray,MortgagesNumMortSatisfied:chararray,SICCodeSicText_1:charar
ray,SICCodeSicText_2:chararray,SICCodeSicText_3:chararray,SICCodeSicText_4:cha
rarray,LimitedPartnershipsNumGenPartners:int,LimitedPartnershipsNumLimPartners:i
nt,URI:chararray,PreviousName_1CONDATE:chararray,PreviousName_1CompanyName:
chararray,PreviousName_2CONDATE:chararray,PreviousName_2CompanyName:
chararray,PreviousName_3CONDATE:chararray,PreviousName_3CompanyName:char
array,PreviousName_4CONDATE:chararray,PreviousName_4CompanyName:chararra
y,PreviousName_5CONDATE:chararray,PreviousName_5CompanyName:chararray,Pr
eviousName_6CONDATE:chararray,PreviousName_6CompanyName:chararray,Previo
usName_7CONDATE:chararray,PreviousName_7CompanyName:chararray,PreviousN
ame_8CONDATE:chararray,PreviousName_8CompanyName:chararray,PreviousName
_9CONDATE:chararray,PreviousName_9CompanyName:chararray,PreviousName_10
CONDATE:chararray,PreviousName_10CompanyName:chararray,ConfStmtNextDueD
ate:chararray,ConfStmtLastMadeUpDate:chararray);
```

loading csv file into a variable

```
var = Load '/home/cloudera/Desktop/Date.csv' using PigStorage(',') as
(str:chararray,date:chararray);
```

displaying the variable

```
dump var;
```

converting the dates into datetime format

```
convert = foreach var generate ToDate(date, 'dd/MM/yyyy') as  
(IncorporationDate:datetime);
```

displaying the dates

```
dump convert;
```

filtering the dates in a particular range > year 2010

```
fill = filter convert by IncorporationDate >= (datetime)ToDate('2010-01-01', 'yyyy-  
MM-dd');
```

displaying the filtered out dates

```
dump fill;
```

counting the filtered out dates to plot the graph

```
counter = foreach (group fill all) generate COUNT(fill);
```

displaying the count

```
dump counter;
```

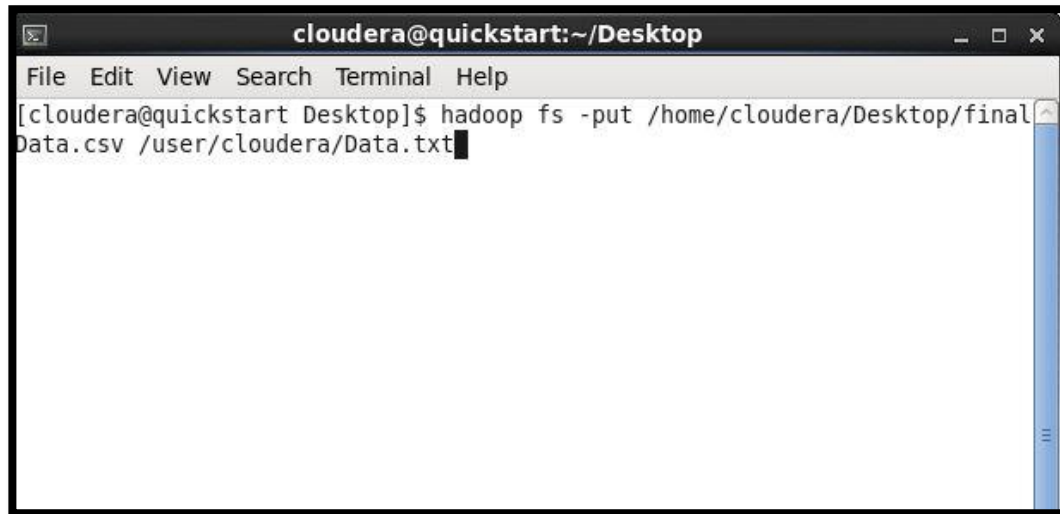
6.1 Open Cloudera CDH for Analyzing Big Data

Cloudera is a commodity software for Hadoop which works on linux for which a software emulator – VMWare is used.



6.2 Copy Input File into HDFS

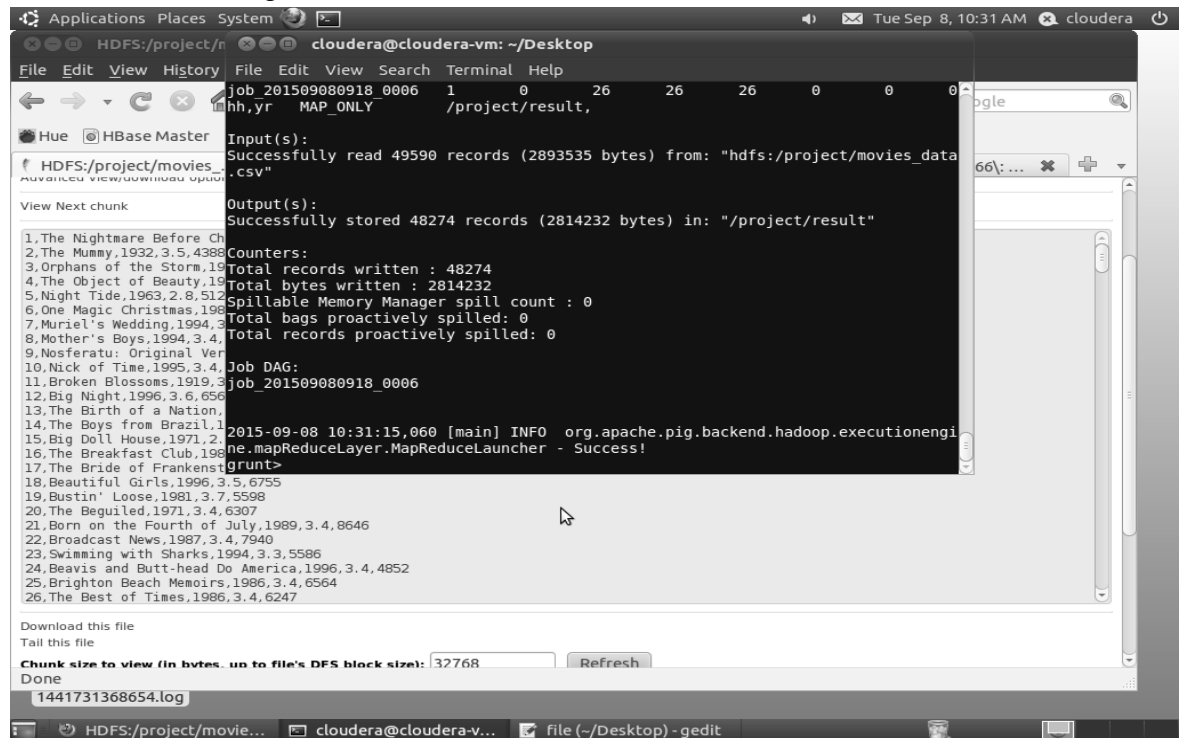
The data file is copied onto HDFS architecture using linux command `-put`.



```
cloudera@quickstart:~/Desktop
File Edit View Search Terminal Help
[cloudera@quickstart Desktop]$ hadoop fs -put /home/cloudera/Desktop/final
Data.csv /user/cloudera/Data.txt
```

6.3 Copy Data into HDFS

The data file is being copied onto HDFS architecture memory whose progress can be seen in the following screenshot.



The screenshot displays the Hue web interface for HDFS operations. The main panel shows the progress of copying data from a local CSV file to HDFS. The input file is `hdfs://project/movies_data.csv` and the output location is `/project/result`. The progress bar indicates that 49590 records (2893535 bytes) have been successfully read from the input file and 48274 records (2814232 bytes) have been successfully stored in the output location.

Input(s):
Successfully read 49590 records (2893535 bytes) from: "hdfs://project/movies_data.csv"

Output(s):
Successfully stored 48274 records (2814232 bytes) in: "/project/result"

Counters:
Total records written : 48274
Total bytes written : 2814232
Spillable Memory Manager spill count : 0
Total bags proactively spilled: 0
Total records proactively spilled: 0

Job DAG:
job_201509080918_0006

2015-09-08 10:31:15,060 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - Success!

The interface also shows a list of movies and their details, including titles, years, genres, and ratings. The list includes movies like "The Nightmare Before Christmas", "The Mummy", "Orphans of the Storm", "The Object of Beauty", "Night Tide", "One Magic Christmas", "Muriel's Wedding", "Mother's Boys", "Nosferatu: Original Version", "Nick of Time", "Broken Blossoms", "Big Night", "The Birth of a Nation", "The Boys from Brazil", "Big Doll House", "The Breakfast Club", "The Bride of Frankenstein", "Beautiful Girls", "Bustin' Loose", "The Beguiled", "Born on the Fourth of July", "Broadcast News", "Swimming with Sharks", "Beavis and Butt-head Do America", "Brighton Beach Memoirs", and "The Best of Times".

6.4 Resulted Data into HDFS

The copied data file can finally be viewed on HDFS architecture memory browsing through the relevant directory.

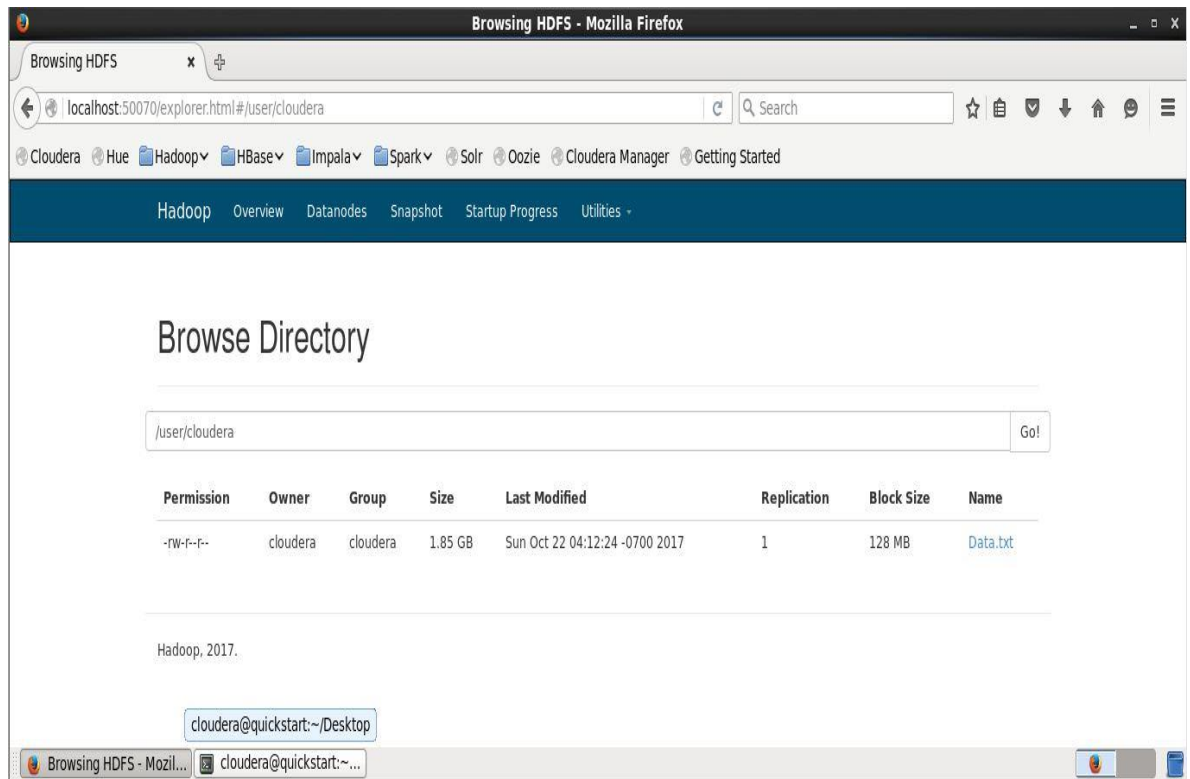
Respective permissions can be seen on the extreme left.

Size of the big data file can be seen under the size tab.

Here we have used a file of 2 GB size.

The path of the respective file can be seen using localhost URL in firefox browser.

The name of the file should be changed or kept same while storing it on HDFS architecture because it takes separate space of memory while being stored in HDFS memory.



6.5 Create Schema in Pig

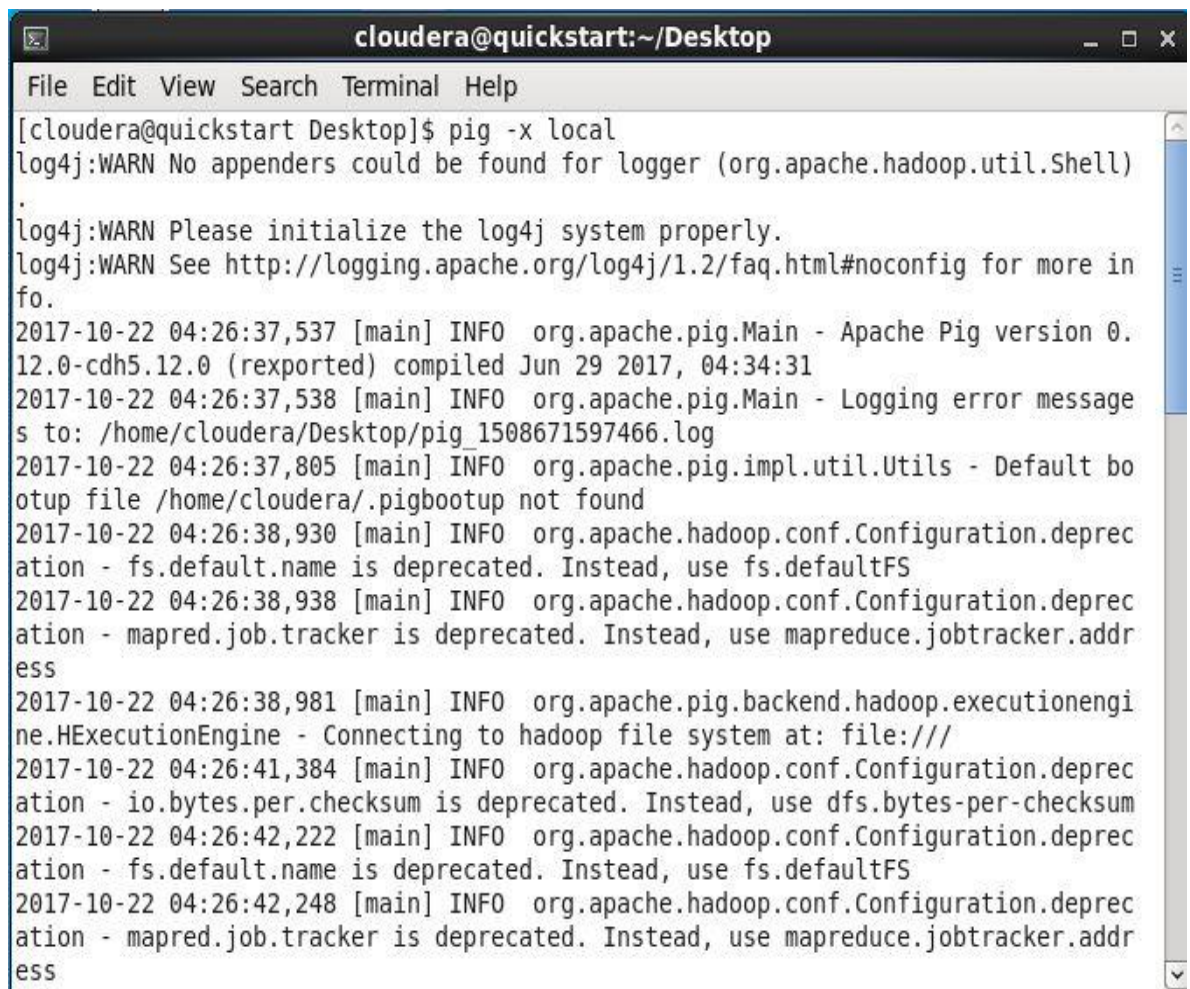
Linux command prompt is used to use a scripting language named PIG

Pig is a framework consisting of a high-level scripting language (Pig Latin) and a run-time environment that allows users to execute MapReduce on a Hadoop cluster.

Like HiveQL in Hive, Pig Latin is a higher-level language that compiles to MapReduce.

To enable pig scripting, following command is used.

Pig scripting is done in Linux Command Prompt window.



```
cloudera@quickstart:~/Desktop
File Edit View Search Terminal Help
[cloudera@quickstart Desktop]$ pig -x local
log4j:WARN No appenders could be found for logger (org.apache.hadoop.util.Shell)
.
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
2017-10-22 04:26:37,537 [main] INFO org.apache.pig.Main - Apache Pig version 0.12.0-cdh5.12.0 (rexported) compiled Jun 29 2017, 04:34:31
2017-10-22 04:26:37,538 [main] INFO org.apache.pig.Main - Logging error message
s to: /home/cloudera/Desktop/pig_1508671597466.log
2017-10-22 04:26:37,805 [main] INFO org.apache.pig.impl.util.Utils - Default bootu
p file /home/cloudera/.pigbootup not found
2017-10-22 04:26:38,930 [main] INFO org.apache.hadoop.conf.Configuration.deprec
ation - fs.default.name is deprecated. Instead, use fs.defaultFS
2017-10-22 04:26:38,938 [main] INFO org.apache.hadoop.conf.Configuration.deprec
ation - mapred.job.tracker is deprecated. Instead, use mapreduce.jobtracker.addr
ess
2017-10-22 04:26:38,981 [main] INFO org.apache.pig.backend.hadoop.executionengi
ne.HExecutionEngine - Connecting to hadoop file system at: file:///
2017-10-22 04:26:41,384 [main] INFO org.apache.hadoop.conf.Configuration.deprec
ation - io.bytes.per.checksum is deprecated. Instead, use dfs.bytes-per-checksum
2017-10-22 04:26:42,222 [main] INFO org.apache.hadoop.conf.Configuration.deprec
ation - fs.default.name is deprecated. Instead, use fs.defaultFS
2017-10-22 04:26:42,248 [main] INFO org.apache.hadoop.conf.Configuration.deprec
ation - mapred.job.tracker is deprecated. Instead, use mapreduce.jobtracker.addr
ess
```

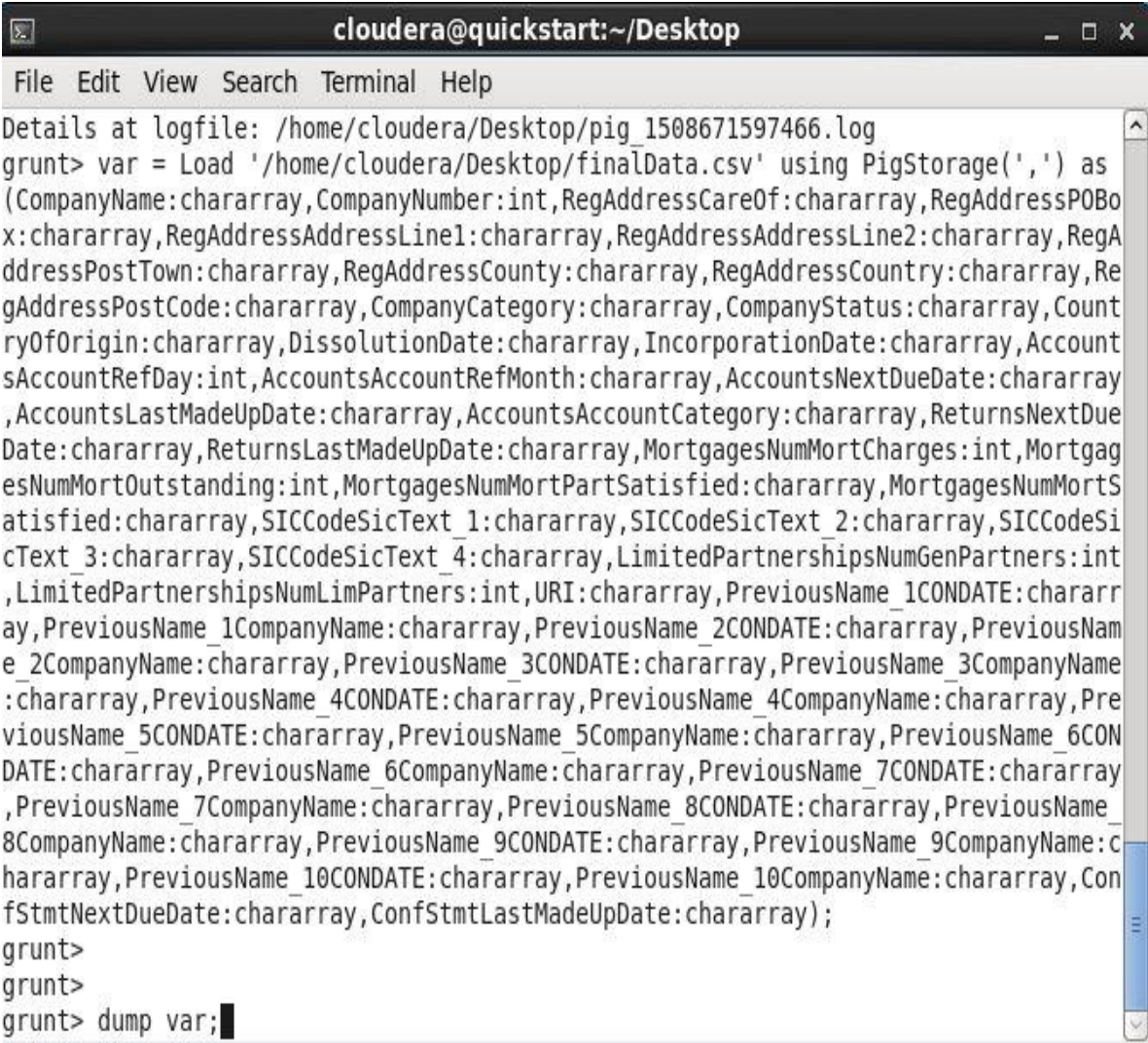
6.6 Loading data in a variable in Pig Schema

To work with the data file, it is first loaded into a variable.

Command “Load” is used to load the data in the temporary storage.

After Load, the path of the file is entered.

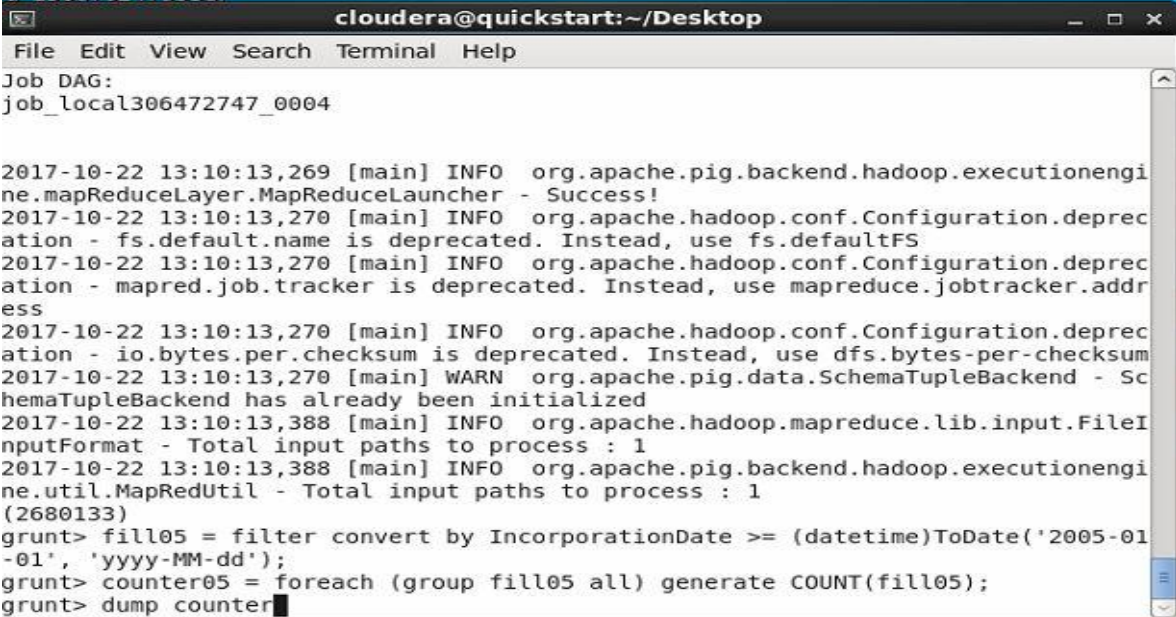
Each column in the csv file is denoted with a column name in the Load command.



```
cloudera@quickstart:~/Desktop
File Edit View Search Terminal Help
Details at logfile: /home/cloudera/Desktop/pig_1508671597466.log
grunt> var = Load '/home/cloudera/Desktop/finalData.csv' using PigStorage(',') as
(CompanyName:chararray,CompanyNumber:int,RegAddressCareOf:chararray,RegAddressPOBo
x:chararray,RegAddressAddressLine1:chararray,RegAddressAddressLine2:chararray,RegA
ddressPostTown:chararray,RegAddressCounty:chararray,RegAddressCountry:chararray,Re
gAddressPostCode:chararray,CompanyCategory:chararray,CompanyStatus:chararray,Count
ryOfOrigin:chararray,DissolutionDate:chararray,IncorporationDate:chararray,Account
sAccountRefDay:int,AccountsAccountRefMonth:chararray,AccountsNextDueDate:chararray
,AccountsLastMadeUpDate:chararray,AccountsAccountCategory:chararray>ReturnsNextDue
Date:chararray>ReturnsLastMadeUpDate:chararray,MortgagesNumMortCharges:int,Mortgag
esNumMortOutstanding:int,MortgagesNumMortPartSatisfied:chararray,MortgagesNumMorts
atisfied:chararray,SICCodeSicText_1:chararray,SICCodeSicText_2:chararray,SICCodeSi
cText_3:chararray,SICCodeSicText_4:chararray,LimitedPartnershipsNumGenPartners:int
,LimitedPartnershipsNumLimPartners:int,URI:chararray,PreviousName_1CONDATE:chararr
ay,PreviousName_1CompanyName:chararray,PreviousName_2CONDATE:chararray,PreviousNam
e_2CompanyName:chararray,PreviousName_3CONDATE:chararray,PreviousName_3CompanyName
:chararray,PreviousName_4CONDATE:chararray,PreviousName_4CompanyName:chararray,Pre
viousName_5CONDATE:chararray,PreviousName_5CompanyName:chararray,PreviousName_6CON
DATE:chararray,PreviousName_6CompanyName:chararray,PreviousName_7CONDATE:chararray
,PreviousName_7CompanyName:chararray,PreviousName_8CONDATE:chararray,PreviousName_
8CompanyName:chararray,PreviousName_9CONDATE:chararray,PreviousName_9CompanyName:c
hararray,PreviousName_10CONDATE:chararray,PreviousName_10CompanyName:chararray,Con
fStmtNextDueDate:chararray,ConfStmtLastMadeUpDate:chararray);
grunt>
grunt>
grunt> dump var;
```


6.7 Counting the number of companies

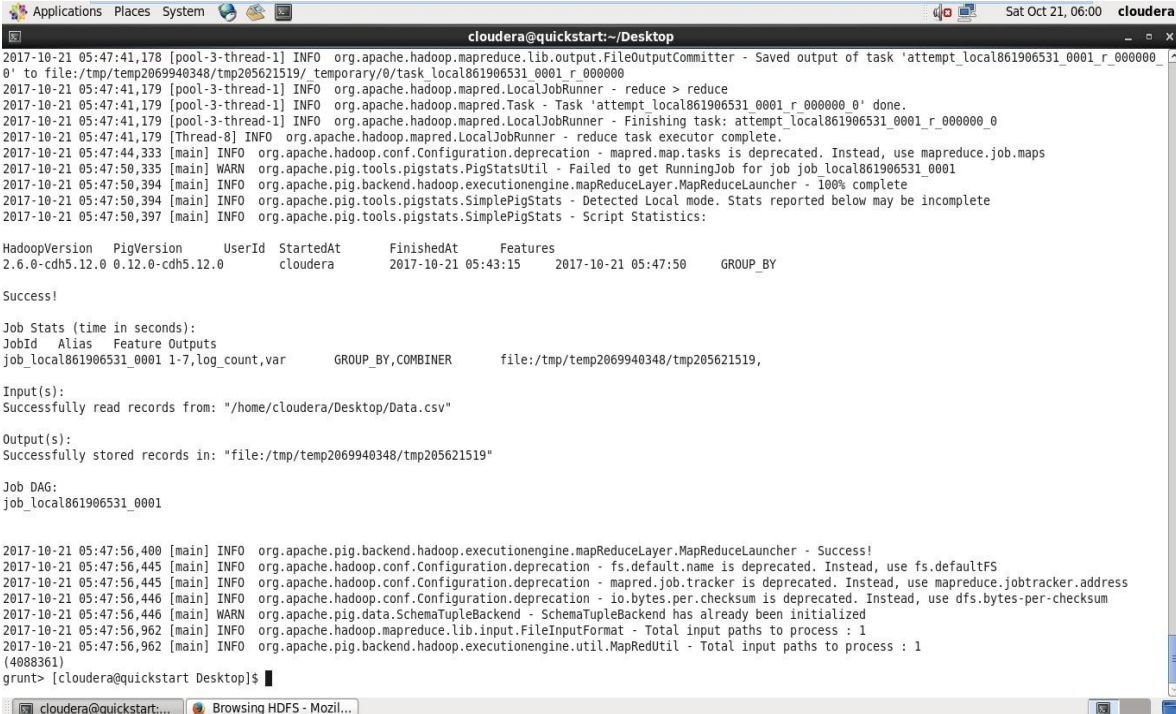
We can count the number of entries in the data file by using a counter function provided in the pig script.



```
cloudera@quickstart:~/Desktop
File Edit View Search Terminal Help
Job DAG:
job_local306472747_0004

2017-10-22 13:10:13,269 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - Success!
2017-10-22 13:10:13,270 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instead, use fs.defaultFS
2017-10-22 13:10:13,270 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - mapred.job.tracker is deprecated. Instead, use mapreduce.jobtracker.address
2017-10-22 13:10:13,270 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - io.bytes.per.checksum is deprecated. Instead, use dfs.bytes-per-checksum
2017-10-22 13:10:13,270 [main] WARN org.apache.pig.data.SchemaTupleBackend - SchemaTupleBackend has already been initialized
2017-10-22 13:10:13,388 [main] INFO org.apache.hadoop.mapreduce.lib.input.FileInputFormat - Total input paths to process : 1
2017-10-22 13:10:13,388 [main] INFO org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input paths to process : 1
(2680133)
grunt> fill05 = filter convert by IncorporationDate >= (datetime)ToDate('2005-01-01', 'yyyy-MM-dd');
grunt> counter05 = foreach (group fill05 all) generate COUNT(fill05);
grunt> dump counter
```

OUTPUT -



```
Applications Places System cloudera@quickstart:~/Desktop Sat Oct 21, 06:00 cloudera
cloudera@quickstart:~/Desktop
2017-10-21 05:47:41,178 [pool-3-thread-1] INFO org.apache.hadoop.mapreduce.lib.output.FileOutputCommitter - Saved output of task 'attempt_local861906531_0001_r_000000_0' to file:/tmp/temp2069940348/tmp205621519/ temporary/0/task_local861906531_0001_r_000000
2017-10-21 05:47:41,179 [pool-3-thread-1] INFO org.apache.hadoop.mapred.LocalJobRunner - reduce > reduce
2017-10-21 05:47:41,179 [pool-3-thread-1] INFO org.apache.hadoop.mapred.Task - Task 'attempt_local861906531_0001_r_000000_0' done.
2017-10-21 05:47:41,179 [pool-3-thread-1] INFO org.apache.hadoop.mapred.LocalJobRunner - Finishing task: attempt_local861906531_0001_r_000000_0
2017-10-21 05:47:41,179 [Thread-8] INFO org.apache.hadoop.mapred.LocalJobRunner - reduce task executor complete.
2017-10-21 05:47:44,333 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - mapred.map.tasks is deprecated. Instead, use mapreduce.job.maps
2017-10-21 05:47:50,335 [main] WARN org.apache.pig.tools.pigstats.PigStatsUtil - Failed to get RunningJob for job job_local861906531_0001
2017-10-21 05:47:50,394 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - 100% complete
2017-10-21 05:47:50,394 [main] INFO org.apache.pig.tools.pigstats.SimplePigStats - Detected Local mode. Stats reported below may be incomplete
2017-10-21 05:47:50,397 [main] INFO org.apache.pig.tools.pigstats.SimplePigStats - Script Statistics:

HadoopVersion PigVersion UserId StartedAt FinishedAt Features
2.6.0-cdh5.12.0 0.12.0-cdh5.12.0 cloudera 2017-10-21 05:43:15 2017-10-21 05:47:50 GROUP_BY

Success!

Job Stats (time in seconds):
JobId Alias Feature Outputs
job_local861906531_0001 1-7,log_count,var GROUP_BY,COMBINER file:/tmp/temp2069940348/tmp205621519,

Input(s):
Successfully read records from: "/home/cloudera/Desktop/Data.csv"

Output(s):
Successfully stored records in: "file:/tmp/temp2069940348/tmp205621519"

Job DAG:
job_local861906531_0001

2017-10-21 05:47:56,400 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - Success!
2017-10-21 05:47:56,445 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instead, use fs.defaultFS
2017-10-21 05:47:56,445 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - mapred.job.tracker is deprecated. Instead, use mapreduce.jobtracker.address
2017-10-21 05:47:56,446 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - io.bytes.per.checksum is deprecated. Instead, use dfs.bytes-per-checksum
2017-10-21 05:47:56,446 [main] WARN org.apache.pig.data.SchemaTupleBackend - SchemaTupleBackend has already been initialized
2017-10-21 05:47:56,962 [main] INFO org.apache.hadoop.mapreduce.lib.input.FileInputFormat - Total input paths to process : 1
2017-10-21 05:47:56,962 [main] INFO org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input paths to process : 1
(4088361)
grunt> [cloudera@quickstart Desktop]$
```

6.8 Counting the number of companies under different time range

```
cloudera@quickstart:~/Desktop
File Edit View Search Terminal Help
Output(s):
Successfully stored records in: "file:/tmp/temp-509308496/tmp-1350454844"

Job DAG:
job_local1038937916_0006

2017-10-22 13:15:54,328 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - Success!
2017-10-22 13:15:54,330 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instead, use fs.defaultFS
2017-10-22 13:15:54,330 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - mapred.job.tracker is deprecated. Instead, use mapreduce.jobtracker.address
2017-10-22 13:15:54,332 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - io.bytes.per.checksum is deprecated. Instead, use dfs.bytes-per-checksum
2017-10-22 13:15:54,333 [main] WARN org.apache.pig.data.SchemaTupleBackend - SchemaTupleBackend has already been initialized
2017-10-22 13:15:54,355 [main] INFO org.apache.hadoop.mapreduce.lib.input.FileInputFormat - Total input paths to process : 1
2017-10-22 13:15:54,355 [main] INFO org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input paths to process : 1
(3596533)
grunt>
```

```
cloudera@quickstart:~/Desktop
File Edit View Search Terminal Help
Output(s):
Successfully stored records in: "file:/tmp/temp-509308496/tmp1968580951"

Job DAG:
job_local1761639010_0005

2017-10-22 13:13:40,836 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - Success!
2017-10-22 13:13:40,839 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instead, use fs.defaultFS
2017-10-22 13:13:40,840 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - mapred.job.tracker is deprecated. Instead, use mapreduce.jobtracker.address
2017-10-22 13:13:40,842 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - io.bytes.per.checksum is deprecated. Instead, use dfs.bytes-per-checksum
2017-10-22 13:13:40,843 [main] WARN org.apache.pig.data.SchemaTupleBackend - SchemaTupleBackend has already been initialized
2017-10-22 13:13:40,956 [main] INFO org.apache.hadoop.mapreduce.lib.input.FileInputFormat - Total input paths to process : 1
2017-10-22 13:13:40,956 [main] INFO org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input paths to process : 1
(3231448)
grunt>
```



```
cloudera@quickstart:~/Desktop
File Edit View Search Terminal Help
Output(s):
Successfully stored records in: "file:/tmp/temp-509308496/tmp1114013861"

Job DAG:
job_local1174914231_0008

2017-10-22 13:20:20,044 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - Success!
2017-10-22 13:20:20,045 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instead, use fs.defaultFS
2017-10-22 13:20:20,045 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - mapred.job.tracker is deprecated. Instead, use mapreduce.jobtracker.address
2017-10-22 13:20:20,046 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - io.bytes.per.checksum is deprecated. Instead, use dfs.bytes-per-checksum
2017-10-22 13:20:20,047 [main] WARN org.apache.pig.data.SchemaTupleBackend - SchemaTupleBackend has already been initialized
2017-10-22 13:20:20,168 [main] INFO org.apache.hadoop.mapreduce.lib.input.FileInputFormat - Total input paths to process : 1
2017-10-22 13:20:20,168 [main] INFO org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input paths to process : 1
(1475505)
grunt>
```

```
cloudera@quickstart:~/Desktop
File Edit View Search Terminal Help
Output(s):
Successfully stored records in: "file:/tmp/temp-509308496/tmp-1047401467"

Job DAG:
job_local306472747_0004

2017-10-22 13:10:13,269 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - Success!
2017-10-22 13:10:13,270 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instead, use fs.defaultFS
2017-10-22 13:10:13,270 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - mapred.job.tracker is deprecated. Instead, use mapreduce.jobtracker.address
2017-10-22 13:10:13,270 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - io.bytes.per.checksum is deprecated. Instead, use dfs.bytes-per-checksum
2017-10-22 13:10:13,270 [main] WARN org.apache.pig.data.SchemaTupleBackend - SchemaTupleBackend has already been initialized
2017-10-22 13:10:13,388 [main] INFO org.apache.hadoop.mapreduce.lib.input.FileInputFormat - Total input paths to process : 1
2017-10-22 13:10:13,388 [main] INFO org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input paths to process : 1
(2680133)
grunt>
```

7. Graph depicting the results

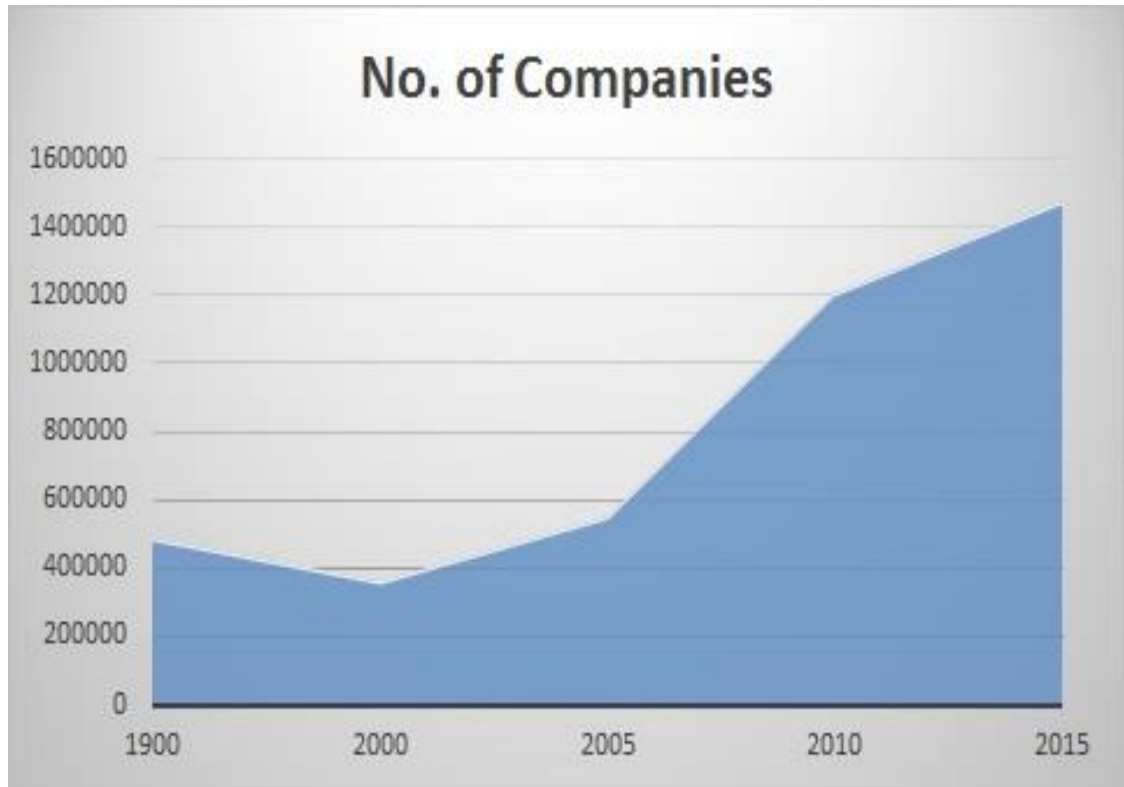


Fig. 7

Number of companies being registered in different time ranges are being plotted on the graph to see the trend of growth of industrial sector over the decades.

To achieve this result, number of companies were counted over each span of 5 years till present.

Number of Companies	Year Span
491860	before 2000
2680133	2010-15
3231448	2005-10
3596533	2000-05
1475505	after 2015

CONCLUSION

The availability of Big Data, low-cost commodity hardware, and new information management and analytic software have produced a unique moment in the history of data analysis.

Report discusses Big Data from its infancy until its current state. It elaborates on the concepts of big data followed by the applications and the challenges faced by it. Finally we have discussed the future opportunities that could be harnessed in this field. Big Data is an evolving field, where much of the research is yet to be done. Big data at present, is handled by the software named Hadoop. However, the proliferating amount of data is making Hadoop insufficient. To harness the potential of Big Data completely in the future, extensive research needs to be carried out and revolutionary technologies need to be developed.

In this project, we utilized the computation power of Hadoop over Big Data to fetch faster results from a data of over 2 GB to analyze some results from the data processing which over normal processors could have taken a lot of time than we took in this project.

FUTURE ENHANCEMENT

- ✓ The world is becoming data driven. Each and every decisions are now taken on data from stock markets to machines that stalk you.
- ✓ The data generated is growing exponentially, with this data we can also develop a great solutions around the data.
- ✓ Data also increases for a better decision making and many problem statements can be designed from the data and also answer can be found from the same data.
- ✓ Also processing power of systems increasing now we have capacity to churn the data to get insights.
- ✓ The power of Hadoop can further be utilized to fetch deeper results from the current data used in the project and even much more heavier data which is almost impossible to do normally.

REFERENCES

- ✓ Apache Hive. Available at <http://hive.apache.org>
- ✓ Oracle and FSN, "Mastering Big Data: CFO Strategies to Transform Insight into Opportunity".
- ✓ *"IBM What is big data? – Bringing big data to the enterprise"*. www.ibm.com.
- ✓ "Big Data for Good" (PDF). ODBMS.org. 5 June 2012.
- ✓ DataSet - <https://data.gov.uk/dataset/basic-company-data>
- ✓ Final Output of Data - http://download.companieshouse.gov.uk/en_output.html
- ✓ Detailed view of dataset -
<http://webarchive.nationalarchives.gov.uk/20140711134125/http://www.companieshouse.gov.uk/toolsToHelp/pdf/freeDataProductDataset.pdf>