D.R. B.R AMBEDKAR NATIONAL INSTITUTE OF TECHNOLOGY
JALANDHAR-144011, PUNJAB (INDIA)

PROJECT REPORT

[Summer Training (June 2018-July 2018)]

BIG DATA ANALYSIS OF TREE CENSUS DATA

Submitted by :
Name: Anmol Manik Singh
Roll No: 16102012
Department: ECE(3rd year)

ABSTRACT

This project work entitled **'Tree Census Data Analysis'** has been designed to keep record of trees planted in a state or city. The dataset of tree census data in **New York (2015-2016)** is used in the project. The dataset is available online at **www.kaggle.com.** The dataset consists of various fields like Tree Id, Scientific Name, Common Name, Location, City, Disease, etc. The project will give the clear picture of what are the different trees species found in New York with the number of dead, living trees and which species is being affected i.e. the species suffering from various diseases, the various diseases spreading in New York affecting the life of trees.

The project aims in developing a computer application through which the information regarding trees as per requirement can be fetched easily and in a very less time. The concerned authority only has to select their requirement and the related data is presented before them in seconds, the obtained graphs and pictorial representations will be a great help to analyse the situation. Otherwise analysing the files line by line manually or through traditional computer software would have been really a cumbersome and time consuming task.

This project has been developed through big data analysis techniques. The dataset containing 600000 rows approximately was stored in the form of a table in Hive metastore which is basically a database. Using Hive, various queries were solved and employed to fetch the required information from the table. Python was used to plot the results in a graphical manner to make visualisation of results easy and attractive.

ACKNOWLEDGMENTS

The training opportunity I had with **ITRONIX SOLUTION** was a great chance for me to learn and develop logical thinking. Therefore, I consider myself as a very lucky individual as I was provided with an opportunity to be a part of it. I am also grateful for having a chance to meet so many wonderful people and professionals who led me though this training period.

I take this opportunity to express a deep sense of gratitude to **Er. Karan Arora**, **(Founder & CEO) , Itronix Solution,Jalandhar** for his cordial support, valuable information and guidance, which helped me in completing this task through various stages, his unfailing cooperation and sparing his valuable time to assist me in various training related issues throughout the training period.

Bearing in mind previous I am using this opportunity to express my deepest gratitude to **Mr.Varun Nayyar (Director), Itronix Solution, Jalandhar** for his unfailing cooperation and guidance throughout the training period.

It is my radiant sentiment to place on record my best regards, deepest sense of gratitude to **Mr. B.S Saini**, Electronics and Communication Department (HOD), and all teachers of the department for their careful and precious guidance which were extremely valuable for my study both theoretically and practically.

Last but not the least I would like to mention here that I am greatly indebted to each and everybody who has been associated with my project at any stage but whose name does not find a place in this acknowledgement.

TABLE OF CONTENTS

1.INTRODUCTION

1.1 OVERVIEW:

Tree Census Data Analysis , refers to the analysis and refining of a huge set of data containing information regarding every tree planted in NewYork (2015-2016) . To fight against the problem of pollution the world has started adopting various policies and planting trees is one among them. Through various means people are motivated to plant more and more trees. But the idea was not limited only upto plantation of trees but also to protect them and fulfilling each and every requirement that a tree demands in future like watering them, proper treatment when a particular problem, disease affects the trees. But here most of the nations or organisations promoting tree plantation have failed .They failed in taking care of trees ,protecting them from diseases .So this project is designed to aid such organisations by providing them the information/ statistics of various factors affecting particular species over the years. From this data analysis the organisations can develop plans and strategies to fight against the problems the trees are suffering. This will help them to easily visualise the area which requires the most efforts.

KEY FEATURES:

- Quick access to required information .
- User friendly interface, easy to operate the application.
- Accurate analysis of a large dataset without much botheration.
- Pictorial representations, graphs make it really easy to analyze the various factors, affecting the life of trees in the recent years.
- Gives deep knowledge of the trees related issues so that the concerned authority can easily develop the plan, strategy and precautions to prevent tree degradation.

➢ Why Tree census dataset?

- Diseases are common plant problems in the landscape. Often, it can be difficult to identify diseases because the causing organisms are rarely seen with the naked eye, and they spread by microscopic growth and spores. However, the signs and symptoms on the tree become obvious as the disease takes hold.
- Tree diseases can affect any part of the tree, or the entire tree. Named for the type of damage they cause, we have leaf spot, leaf blotch, scab and blister, defoliation, needle cast, and yellowing or chlorosis as named symptoms. Stem canker and galls, trunk and root rot are also very prominent tree disorders.
- Before you take corrective action, you have to determine what is causing the problem. A correct diagnosis is the first and most important step in developing and applying a correct treatment, which allows you to address the problem rather than simply treat the symptom. This is when a diagnosis from a professional arborist may prove critical.
- Not only diseases but there are many other factors that can affect the tree's life.
- To know whether there is proper fencing around every tree to ensure safety, whether every tree is given unique id so that whenever required its data can be accessed easily.
- So to make such analysis easy this dataset has been selected and at the end the results obtained will surely help the authorities to choose the right step to cure each and every tree as per its requirement.

1.2 BIG DATA:

1.2.1 What is Big Data?

'Big Data' is a term used to describe collection of data that is huge in size and yet growing exponentially with time. In short, such a data is so large and complex that none of the traditional data management tools are able to store it or process it efficiently. Big Data challenges include capturing data, data storage, data analysis, search, sharing, transfer, visualisation, querying, updating, information, privacy, and data source.

Examples of Big Data:

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

1.2.1.1 Categories of Big Data:

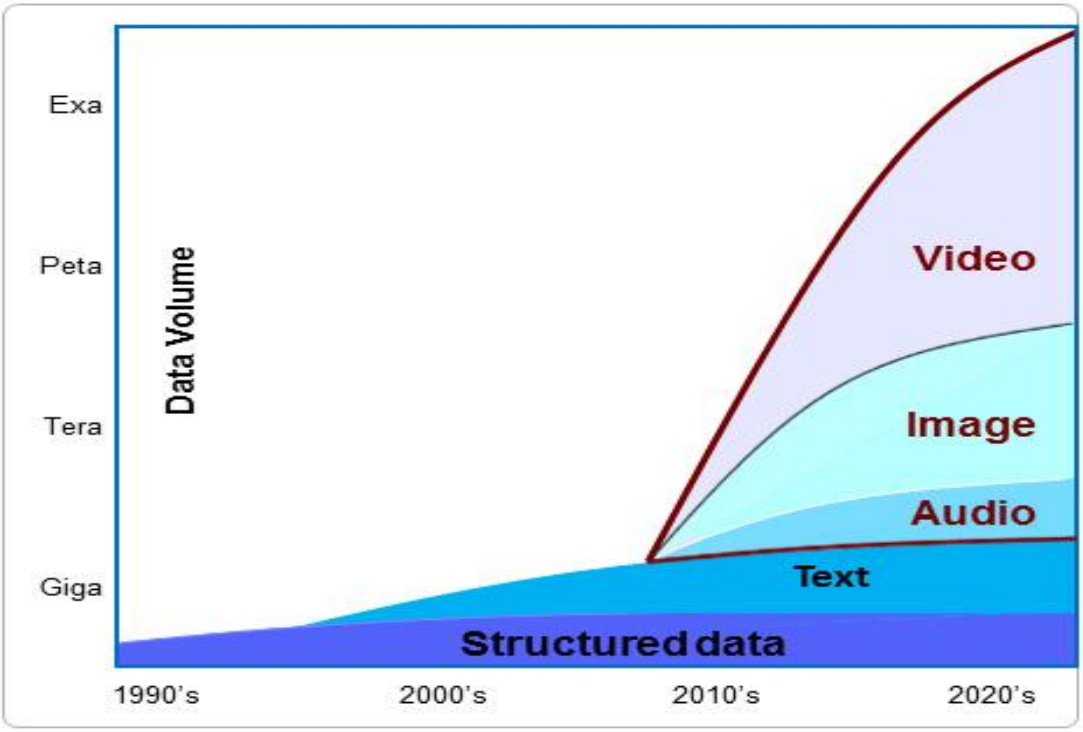
Big data' could be found in three forms:

1. **Structured:**
   Any data that can be stored, accessed and processed in the form of fixed format is termed as a 'structured' data.

2. **Unstructured:**
   Any data with unknown form or the structure is classified as unstructured data. In addition to the size being huge, un-structured data poses multiple challenges in terms of its processing for deriving value out of it.

3. **Semi-structured:**
   Semi-structured data can contain both the forms of data. We can see semi-structured data as a structured in form but it is actually not defined with e.g. a table definition in relational DBMS. Example of semi-structured data is a data represented in XML file.

❖ Data Growth over years:



❖ Characteristics Of Big Data:

(i) **Volume—**The name 'Big Data' itself is related to a size which is enormous. Size of data plays very crucial role in determining value out of data. Also, whether a particular data can actually be considered as a Big Data or not, is dependent upon volume of data.

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

(iii) **Velocity**—The term **'velocity'** refers to the speed of generation of data. How fast the data is generated and processed to meet the demands, determines real potential in the data.
Big Data Velocity deals with the speed at which data flows in from sources like business processes, application logs, networks and social media sites, sensors,Mobile devices, etc. The flow of data is massive and continuous.

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


BIG DATA
with 8 V's
01
VOLUME
Can you find the information you are looking for?
02
VALUE
Can you find it when you most need it?
03
VERACITY
Are you dealing with information or disinformation?
04
VISUALISATION
Can you make sense at a glance? Does it trigger a decision?
05
VARIETY
Is a picture worth a thousand words in 70 languages? Is your information balanced?
06
VELOCITY
Information gains momentum and crises & opportunities evolve in real time. How is outlook for today?
07
VISCOSITY
Does it stick with you? Does it call for action?
08
VIRALITY
AHA to-go? Does it convey a message that can be pasted into a presentation or Instgrammed?

❖ Why is Big Data Important ?

The importance of big data does not revolve around how much data a company has but how a company utilises the collected data. Every company uses data in its own way; the more efficiently a company uses its data, the more potential it has to grow. The company can take data from any source and analyse it to find answers which will enable:

- **Cost Savings :** Some tools of Big Data like Hadoop and Cloud-Based Analytics can bring cost advantages to business when large amounts of data are to be stored and these tools also help in identifying more efficient ways of doing business.

- **Time Reductions :**The high speed of tools like Hadoop and in-memory analytics can easily identify new sources of data which helps businesses analyzing data immediately and make quick decisions based on the learnings.'

- **New Product Development :** By knowing the trends of customer needs and satisfaction through analytics you can create products according to the wants of customers.

- **Understand the market conditions :** By analyzing big data you can get a better understanding of current market conditions. For example, by analyzing customers' purchasing behaviors, a company can find out the products that are sold the most and produce products according to this trend. By this, it can get ahead of its competitors.

- **Control online reputation:** Big data tools can do sentiment analysis. Therefore, you can get feedback about who is saying what about your company. If you want to monitor and improve the online presence of your business, then, big data tools can help in all this.

2- SOFTWARE REQUIREMENTS

This chapter will include the details regarding the software and tools used to develop the project :

2.1 HADOOP:

2.1.1 What is Hadoop?

Hadoop is an open-source framework for storage and large-scale processing of datasets on community Hardware. It was developed by Doug Cutting in 2005 who was working for Yahoo at that time.
Hadoop was inspired by Google's Map Reduce and Google Files System projects. The idea was to develop an open source framework where anyone can write map reduce jobs without worrying about Hardware Failures.

2.1.2 Map Reduce :

It helps to process really large sets of data on a cluster using a parallel distributed algorithm.

2.1.3 HDFS (Hadoop Distributed File System):

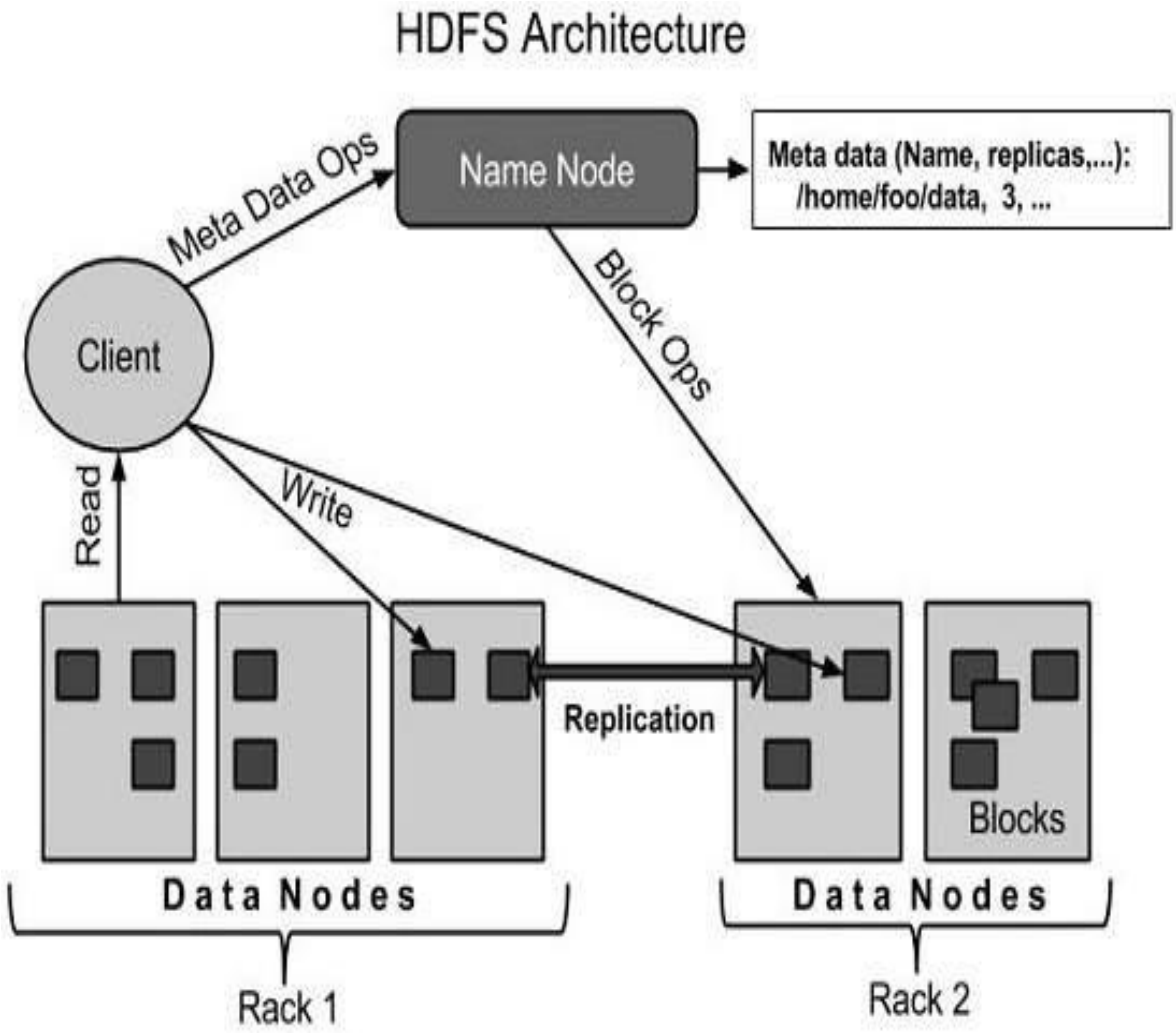
Hadoop File System was developed using distributed file system design. It is run on commodity hardware. Unlike other distributed systems, HDFS is highly faulttolerant and designed using low-cost hardware.
HDFS holds very large amount of data and provides easier access. To store such huge data, the files are stored across multiple machines. These files are stored in redundant fashion to rescue the system from possible data losses in case of failure. HDFS also makes applications available to parallel processing.

➢ Features of HDFS

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

2.1.4 HDFS Architecture:

Given below is the architecture of a Hadoop File System.



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

2.1.5 Namenode:

The namenode is the commodity hardware that contains the GNU/Linux operating system and the namenode software. It is a software that can be run on commodity hardware. The system having the namenode acts as the master server and it does the following tasks:

- Manages the file system namespace.
- Regulates client's access to files.
- It also executes file system operations such as renaming, closing, and opening files and directories.

2.1.6 Datanode:

The datanode is a commodity hardware having the GNU/Linux operating system and datanode software. For every node (Commodity hardware/System) in a cluster, there will be a datanode. These nodes manage the data storage of their system.

- Datanodes perform read-write operations on the file systems, as per client request.
- They also perform operations such as block creation, deletion, and replication according to the instructions of the namenode.

2.1.7 Block:

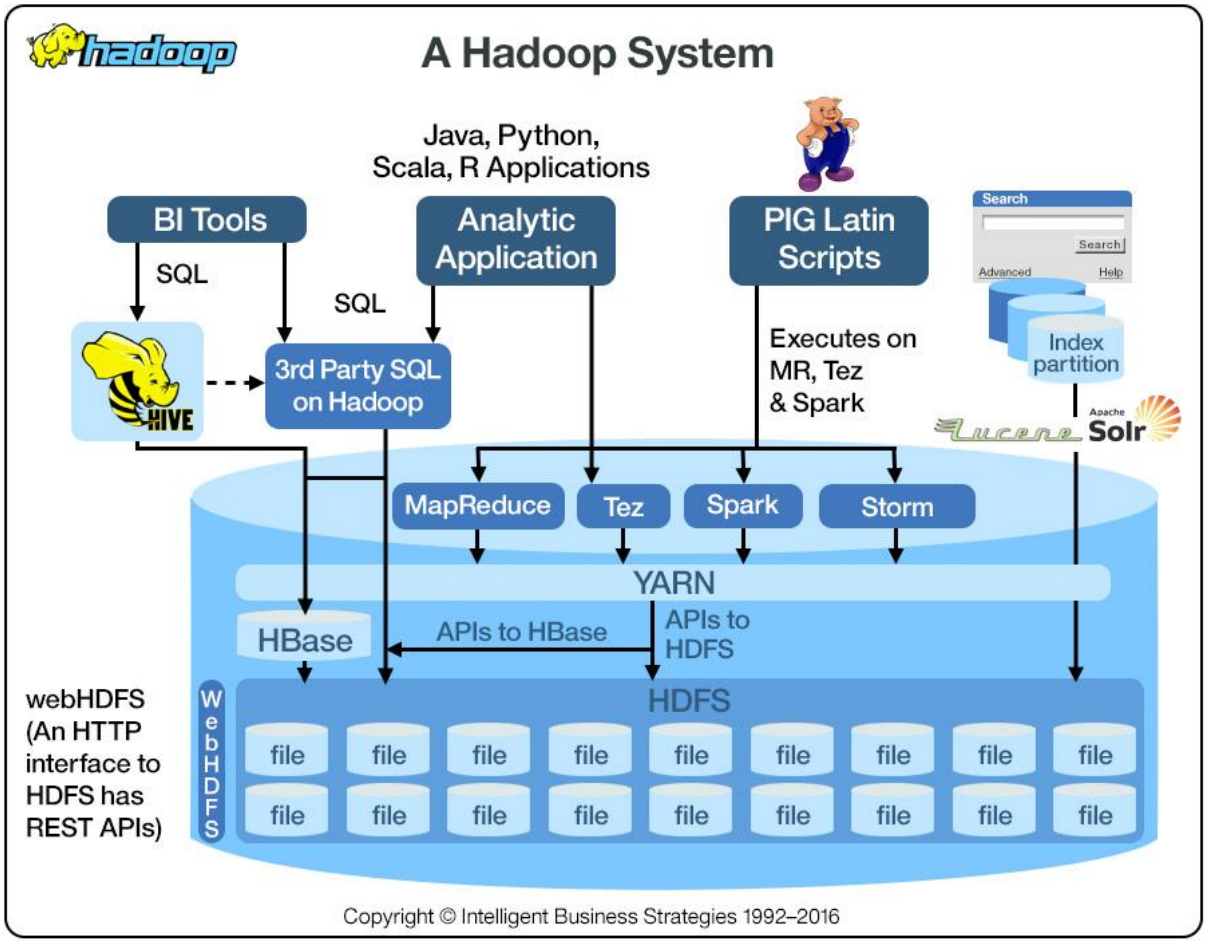
Generally the user data is stored in the files of HDFS. The file in a file system will be divided into one or more segments and/or stored in individual data nodes. These file segments are called as blocks. In other words, the minimum amount of data that HDFS can read or write is called a Block. The default block size is 64MB, but it can be increased as per the need to change in HDFS configuration.

❖ Goals of HDFS :

- **Fault detection and recovery** : Since HDFS includes a large number of commodity hardware, failure of components is frequent. Therefore HDFS should have mechanisms for quick and automatic fault detection and recovery.

- **Huge datasets** : HDFS should have hundreds of nodes per cluster to manage the applications having huge datasets.

- **Hardware at data** : A requested task can be done efficiently, when the computation takes place near the data. Especially where huge datasets are involved, it reduces the network traffic and increases the throughput.



2.2 HIVE:

2.2.1 What is Hive?

Hive is a data warehouse infrastructure tool to process structured data in Hadoop. It resides on top of Hadoop to summarize Big Data, and makes querying and analyzing easy.

Initially Hive was developed by Facebook, later the Apache Software Foundation took it up and developed it further as an open source under the name Apache Hive. It is used by different companies. For example, Amazon uses it in Amazon Elastic MapReduce.

Hive is not

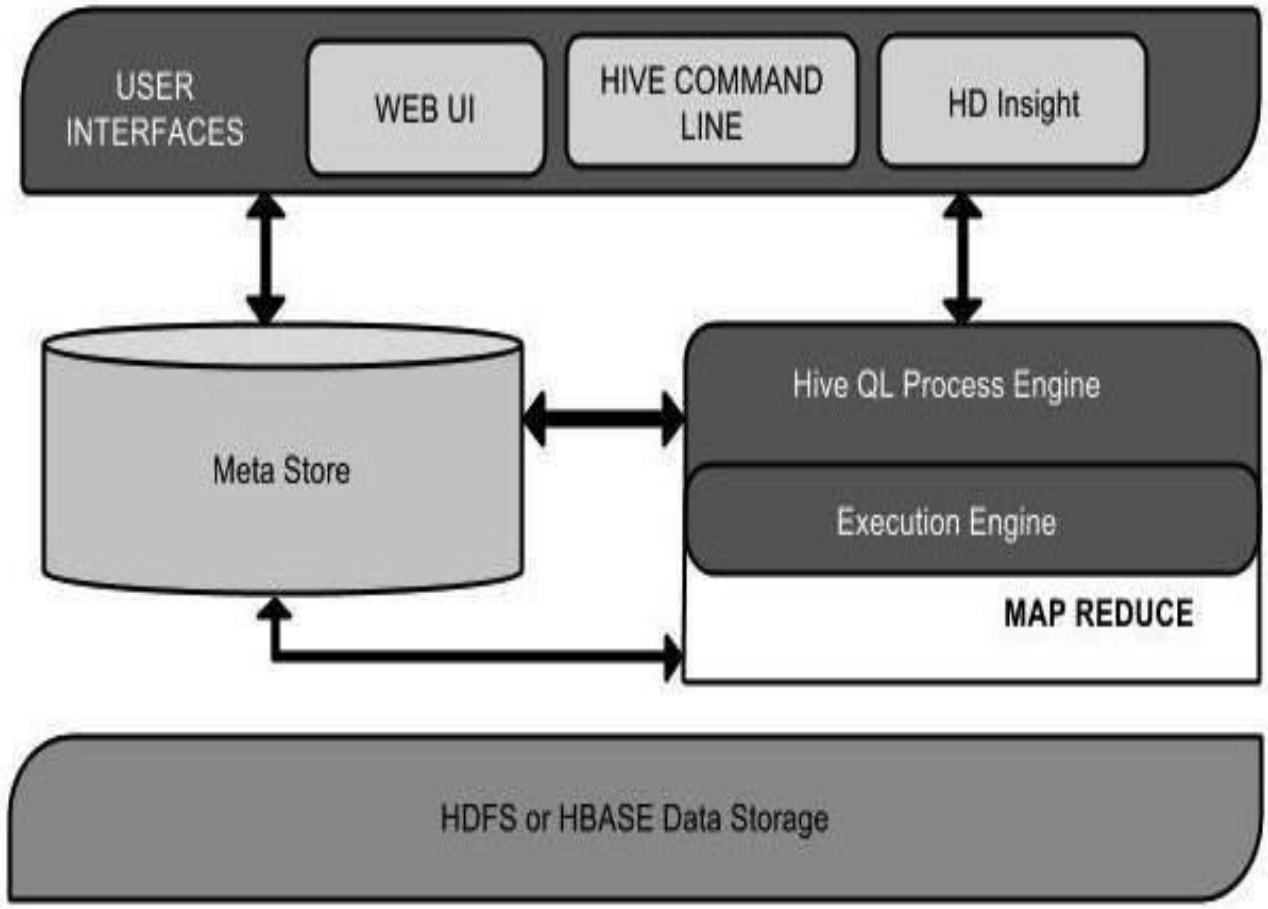
- A relational database
- A design for OnLine Transaction Processing (OLTP)
- A language for real-time queries and row-level updates

❖ Features of Hive :

- It stores schema in a database and processed data into HDFS.
- It is designed for OLAP.
- It provides SQL type language for querying called HiveQL or HQL.
- It is familiar, fast, scalable, and extensible.

2.2.2 Architecture of Hive :

The following component diagram depicts the architecture of Hive:



This component diagram contains different units. The following table describes each unit:

Unit Name	Operation
User Interface	Hive is a data warehouse infrastructure software that can create interaction between user and HDFS. The user interfaces that Hive supports are Hive Web UI, Hive command line, and Hive HD Insight (In Windows server).
Meta Store	Hive chooses respective database servers to store the schema or Metadata of tables, databases, columns in a table, their data types, and HDFS mapping.
HiveQL Process Engine	HiveQL is similar to SQL for querying on schema info on the Metastore. It is one of the replacements of traditional approach for MapReduce program. Instead of writing MapReduce program in Java, we can write a query for MapReduce job and process it.
Execution Engine	The conjunction part of HiveQL process Engine and MapReduce is Hive Execution Engine. Execution engine processes the query and generates results as same as MapReduce results. It uses the flavor of MapReduce.
HDFS or HBASE	Hadoop distributed file system or HBASE are the data storage techniques to store data into file system.

2.3 PIG :

2.3.1 What is Pig?

Apache Pig is an abstraction over MapReduce. It is a tool/platform which is used to analyze larger sets of data representing them as data flows. Pig is generally used with **Hadoop**; we can perform all the data manipulation operations in Hadoop using Apache Pig.

To write data analysis programs, Pig provides a high-level language known as **Pig Latin**. This language provides various operators using which programmers can develop their own functions for reading, writing, and processing data.

To analyze data using **Apache Pig**, programmers need to write scripts using Pig Latin language. All these scripts are internally converted to Map and Reduce tasks. Apache Pig has a component known as **Pig Engine** that accepts the Pig Latin scripts as input and converts those scripts into MapReduce jobs.

2.3.2 Why do we need pig ?

Programmers who are not so good at Java normally used to struggle working with Hadoop, especially while performing any MapReduce tasks. Apache Pig is a boon for all such programmers.

- Using Pig Latin, programmers can perform MapReduce tasks easily without having to type complex codes in Java.
- Apache Pig uses multi-query approach, thereby reducing the length of codes. For example, an operation that would require you to type 200 lines of code (LoC) in Java can be easily done by typing as less as just 10 LoC in Apache Pig. Ultimately Apache Pig reduces the development time by almost 16 times.
- Pig Latin is SQL-like language and it is easy to learn Apache Pig when you are familiar with SQL.
- Apache Pig provides many built-in operators to support data operations like

joins, filters, ordering, etc. In addition, it also provides nested data types like tuples, bags, and maps that are missing from MapReduce.

❖ **Features of Pig :**

Apache Pig comes with the following features −

- **Rich set of operators** − It provides many operators to perform operations like join, sort, filer, etc.
- **Ease of programming** − Pig Latin is similar to SQL and it is easy to write a Pig script if you are good at SQL.
- **Optimization opportunities** − The tasks in Apache Pig optimize their execution automatically, so the programmers need to focus only on semantics of the language.
- **Extensibility** − Using the existing operators, users can develop their own functions to read, process, and write data.
- **UDF's** − Pig provides the facility to create **User-defined Functions** in other programming languages such as Java and invoke or embed them in Pig Scripts.
- **Handles all kinds of data** − Apache Pig analyzes all kinds of data, both structured as well as unstructured. It stores the results in HDFS.

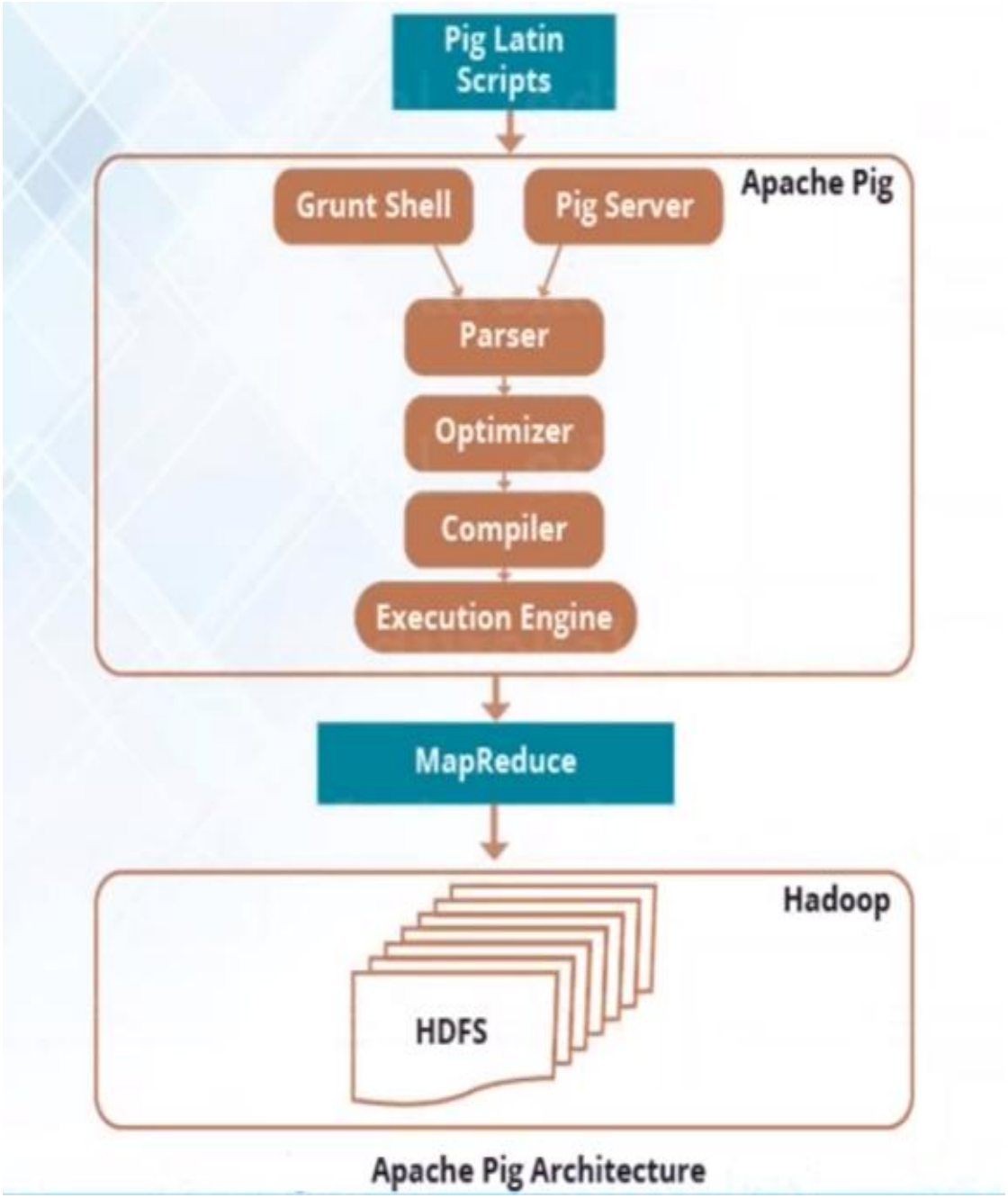


Fig: Apache pig architecture.

2.3.3 Apache Pig Vs Hive :

Both Apache Pig and Hive are used to create MapReduce jobs. And in some cases, Hive operates on HDFS in a similar way Apache Pig does. In the following table, we have listed a few significant points that set Apache Pig apart from Hive.

Apache Pig	Hive
Apache Pig uses a language called **Pig Latin**. It was originally created at **Yahoo**.	Hive uses a language called **HiveQL**. It was originally created at **Facebook**.
Pig Latin is a data flow language.	HiveQL is a query processing language.
Pig Latin is a procedural language and it fits in pipeline paradigm.	HiveQL is a declarative language.
Apache Pig can handle structured, unstructured, and semi-structured data.	Hive is mostly for structured data.

2.4 Python :

Matplotlib is a python library used to create 2D graphs and plots by using python scripts. It has a module named pyplot which makes things easy for plotting by providing feature to control line styles, font properties, formatting axes etc. It supports a very wide variety of graphs and plots namely - histogram, bar charts, power spectra, error charts etc. It is used along with NumPy to provide an environment that is an effective open source alternative for MatLab. It can also be used with graphics toolkits like PyQt and wxPython.

NumPy is often used along with packages like SciPy (Scientific Python) and Mat−plotlib (plotting library). This combination is widely used as a replacement for MatLab, a popular platform for technical computing. However, Python alternative to MatLab is now seen as a more modern and complete programming language.
It is open source, which is an added advantage of NumPy.

2.5 SYSTEM REQUIREMENTS

- Dual Quad-core CPUs or greater that have Hyper-Threading enabled. If you can estimate your computing workload, consider using a more powerful CPU.

- Use High Availability (HA) and dual power supplies for the master node's host machine.

- 4-8 GBs of memory per processor core, with 6% overhead for virtualization.

- Use a 1 Gigabit Ethernet interface or greater to provide adequate network bandwidth.

- Operating system Linux (Ubuntu(16.04))

CHAPTER 3: PROJECT WORK UNDERTAKEN

3.1 INSTALLED THE REQUIRED SOFTWARE.

3.1.1 Installed Ubuntu 16.04 LTS.

Linux is most common operating system of web servers, so ubuntu was installed.

3.1.2 Installed Java.

Java was installed.

3.1.3 Installed Hadoop.

Hadoop was installed to manage the data files and import/export them into hive/pig.

3.1.4 Installed Apache hive.

Hive has three main functions: data summarization, query and analysis. It supports queries expressed in a language called HiveQL, which automatically translates SQL-like queries into MapReduce jobs executed on Hadoop. In addition, HiveQL supports custom MapReduce scripts to be plugged into queries.

3.1.5 Installed Apache Pig.

Pig can execute its Hadoop jobs in MapReduce, Apache Tez, or Apache Spark. Pig Latin abstracts the programming from the Java MapReduce idiom into a notation which makes MapReduce programming high level, similar to that of SQL for RDBMSs. Apache Pig is designed to handle any kind of data.

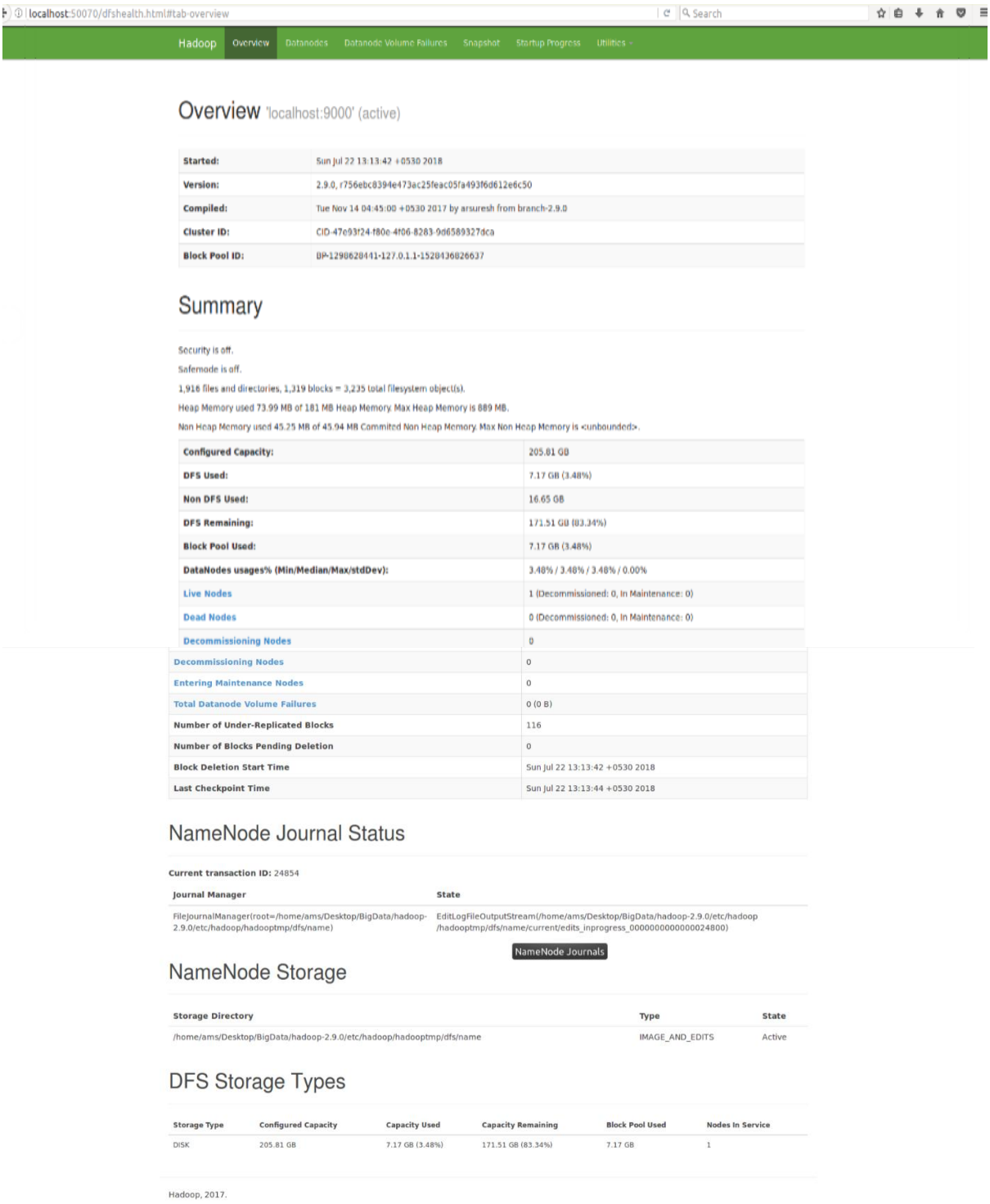
3.1.6 Downloaded the Tree census dataset from kaggle.com.

A sample from the dataset:

```
tree_id,block_id,created_at,tree_dbh,stump_diam,curb_loc,status,health,spc_latin,spc_common,steward,guards,sidewalk,user_type,problems,roo
board,borocode,borough,cncldist,st_assem,st_senate,nta,nta_name,boro_ct,state,latitude,longitude,x_sp,y_sp,council district,census
tract,bin,bbl
180683,348711,2015-08-27T00:00:00,3,0,OnCurb,Alive,Fair,Acer rubrum,red maple,None,None,NoDamage,TreesCount
Staff,None,No,No,No,No,No,No,No,No,No,108-005 70 AVENUE,11375,Forest Hills,406,4,Queens,29,28,16,QN17,Forest Hills,4073900,New
York,40.72309177,-73.84421522,1027431.148,202756.7687,29,739,4052307,4022210001
200540,315986,2015-09-03T00:00:00,21,0,OnCurb,Alive,Fair,Quercus palustris,pin oak,None,None,Damage,TreesCount
Staff,Stones,Yes,No,No,No,No,No,No,No,No,147-074 7 AVENUE,11357,Whitestone,407,4,Queens,19,27,11,QN49,Whitestone,4097300,New
York,40.79411067,-73.81867946,1034455.701,228644.8374,19,973,4101931,4044750045
204026,218365,2015-09-05T00:00:00,3,0,OnCurb,Alive,Good,Gleditsia triacanthos var.
inermis,honeylocust,1or2,None,Damage,Volunteer,None,No,No,No,No,No,No,No,No,390 MORGAN
AVENUE,11211,Brooklyn,301,3,Brooklyn,34,50,18,BK90,East Williamsburg,3044900,New
York,40.71758074,-73.9366077,1001822.831,200716.8913,34,449,3338310,3028870001
204337,217969,2015-09-05T00:00:00,10,0,OnCurb,Alive,Good,Gleditsia triacanthos var.
inermis,honeylocust,None,None,Damage,Volunteer,Stones,Yes,No,No,No,No,No,No,No,1027 GRAND
STREET,11211,Brooklyn,301,3,Brooklyn,34,53,18,BK90,East Williamsburg,3044900,New
York,40.71353749,-73.93445616,1002420.358,199244.2531,34,449,3338342,3029250001
189565,223043,2015-08-30T00:00:00,21,0,OnCurb,Alive,Good,Tilia americana,American
linden,None,None,Damage,Volunteer,Stones,Yes,No,No,No,No,No,No,No,603 6 STREET,11215,Brooklyn,306,3,Brooklyn,39,44,21,BK37,Park Slope-
Gowanus,3016500,New York,40.66677776,-73.97597938,990913.775,182202.426,39,165,3025654,3010850052
190422,106099,2015-08-30T00:00:00,11,0,OnCurb,Alive,Good,Gleditsia triacanthos var.
inermis,honeylocust,1or2,Helpful,NoDamage,Volunteer,None,No,No,No,No,No,No,No,No,8 COLUMBUS AVENUE,10023,New
York,107,1,Manhattan,3,67,27,MN14,Lincoln Square,1014500,New York,40.77004563,-73.98494997,988418.6997,219825.5227,3,145,1076229,10113100
190426,106099,2015-08-30T00:00:00,11,0,OnCurb,Alive,Good,Gleditsia triacanthos var.
inermis,honeylocust,1or2,Helpful,NoDamage,Volunteer,None,No,No,No,No,No,No,No,No,120 WEST 60 STREET,10023,New
York,107,1,Manhattan,3,67,27,MN14,Lincoln Square,1014500,New York,40.77020969,-73.98533807,988311.19,219885.2785,3,145,1076229,1011310031
208649,103940,2015-09-07T00:00:00,9,0,OnCurb,Alive,Good,Tilia americana,American
linden,None,None,NoDamage,Volunteer,MetalGrates,No,Yes,No,No,No,No,No,No,311 WEST 50 STREET,10019,New
York,104,1,Manhattan,3,75,27,MN15,Clinton,1012700,New York,40.76272385,-73.98729652,987769.1163,217157.8561,3,133,1086093,1010410019
209610,407443,2015-09-08T00:00:00,6,0,OnCurb,Alive,Good,Gleditsia triacanthos var. inermis,honeylocust,None,None,NoDamage,TreesCount
Staff,None,No,No,No,No,No,No,No,No,No,65 JEROME AVENUE,10305,Staten Island,502,5,Staten Island,50,64,23,SI14,Grasmere-Arrochar-Ft.
Wadsworth,5006400,New York,40.59657931,-74.07625483,963073.1998,156635.5542,,,,
192755,207508,2015-08-31T00:00:00,21,0,OffsetFromCurb,Alive,Fair,Platanus x acerifolia,London planetree,None,None,NoDamage,TreesCount
Staff,None,No,No,No,No,No,No,No,No,No,638 AVENUE Z,11223,Brooklyn,313,3,Brooklyn,47,45,23,BK26,Gravesend,3037402,New
York,40.58635725,-73.96974394,992653.7253,152903.6306,47,37402,3320727,3072350001
203719,302371,2015-09-05T00:00:00,11,0,OnCurb,Alive,Good,Platanus x acerifolia,London
planetree,None,None,NoDamage,Volunteer,None,No,No,No,No,No,No,No,No,20-025 24
STREET,11105,Astoria,401,4,Queens,22,36,13,QN72,Steinway,4010500,New
York,40.78242823,-73.91117077,1008850.185,224349.0366,22,105,4019061,4008710030
```

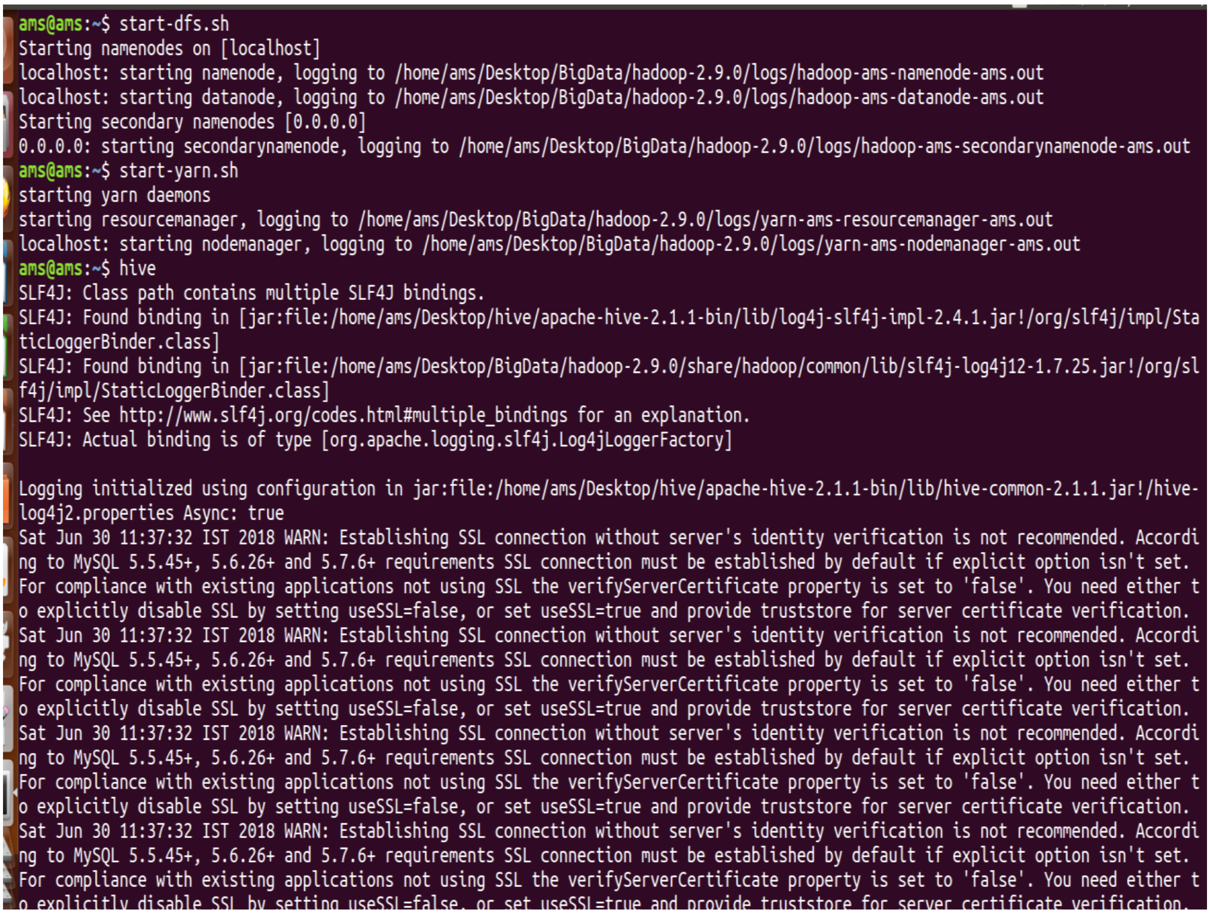
CSV ▾ Tab Width: 8 ▾ Ln 1, Col 1 ▾

3.2 Getting started with Hadoop.



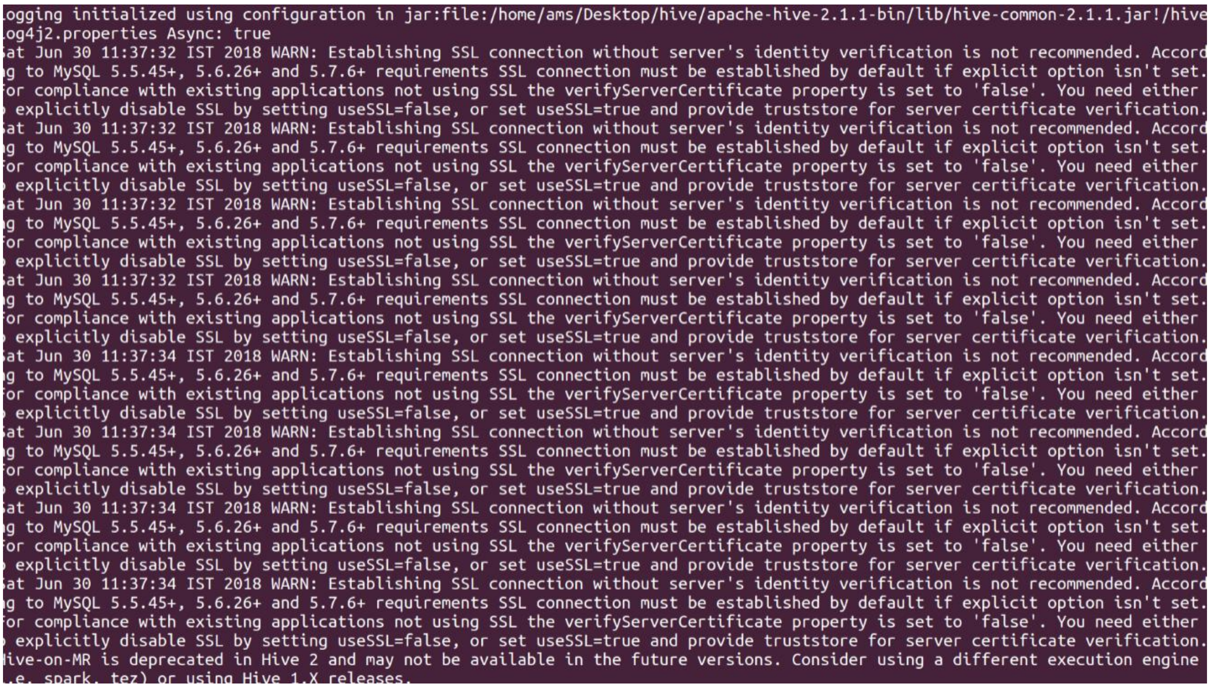
3.3 CREATING TABLE IN HIVE TO STORE THE DATASET IN TABULAR FORM.

➢ Starting HIVE.



```
ams@ams:~$ start-dfs.sh
Starting namenodes on [localhost]
localhost: starting namenode, logging to /home/ams/Desktop/BigData/hadoop-2.9.0/logs/hadoop-ams-namenode-ams.out
localhost: starting datanode, logging to /home/ams/Desktop/BigData/hadoop-2.9.0/logs/hadoop-ams-datanode-ams.out
Starting secondary namenodes [0.0.0.0]
0.0.0.0: starting secondarynamenode, logging to /home/ams/Desktop/BigData/hadoop-2.9.0/logs/hadoop-ams-secondarynamenode-ams.out
ams@ams:~$ start-yarn.sh
starting yarn daemons
starting resourcemanager, logging to /home/ams/Desktop/BigData/hadoop-2.9.0/logs/yarn-ams-resourcemanager-ams.out
localhost: starting nodemanager, logging to /home/ams/Desktop/BigData/hadoop-2.9.0/logs/yarn-ams-nodemanager-ams.out
ams@ams:~$ hive
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/ams/Desktop/hive/apache-hive-2.1.1-bin/lib/log4j-slf4j-impl-2.4.1.jar!/org/slf4j/impl/Sta
ticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/ams/Desktop/BigData/hadoop-2.9.0/share/hadoop/common/lib/slf4j-log4j12-1.7.25.jar!/org/sl
f4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]

Logging initialized using configuration in jar:file:/home/ams/Desktop/hive/apache-hive-2.1.1-bin/lib/hive-common-2.1.1.jar!/hive-
log4j2.properties Async: true
Sat Jun 30 11:37:32 IST 2018 WARN: Establishing SSL connection without server's identity verification is not recommended. Accordi
ng to MySQL 5.5.45+, 5.6.26+ and 5.7.6+ requirements SSL connection must be established by default if explicit option isn't set.
For compliance with existing applications not using SSL the verifyServerCertificate property is set to 'false'. You need either t
o explicitly disable SSL by setting useSSL=false, or set useSSL=true and provide truststore for server certificate verification.
Sat Jun 30 11:37:32 IST 2018 WARN: Establishing SSL connection without server's identity verification is not recommended. Accordi
ng to MySQL 5.5.45+, 5.6.26+ and 5.7.6+ requirements SSL connection must be established by default if explicit option isn't set.
For compliance with existing applications not using SSL the verifyServerCertificate property is set to 'false'. You need either t
o explicitly disable SSL by setting useSSL=false, or set useSSL=true and provide truststore for server certificate verification.
Sat Jun 30 11:37:32 IST 2018 WARN: Establishing SSL connection without server's identity verification is not recommended. Accordi
ng to MySQL 5.5.45+, 5.6.26+ and 5.7.6+ requirements SSL connection must be established by default if explicit option isn't set.
For compliance with existing applications not using SSL the verifyServerCertificate property is set to 'false'. You need either t
o explicitly disable SSL by setting useSSL=false, or set useSSL=true and provide truststore for server certificate verification.
Sat Jun 30 11:37:32 IST 2018 WARN: Establishing SSL connection without server's identity verification is not recommended. Accordi
ng to MySQL 5.5.45+, 5.6.26+ and 5.7.6+ requirements SSL connection must be established by default if explicit option isn't set.
For compliance with existing applications not using SSL the verifyServerCertificate property is set to 'false'. You need either t
o explicitly disable SSL by setting useSSL=false, or set useSSL=true and provide truststore for server certificate verification.
```



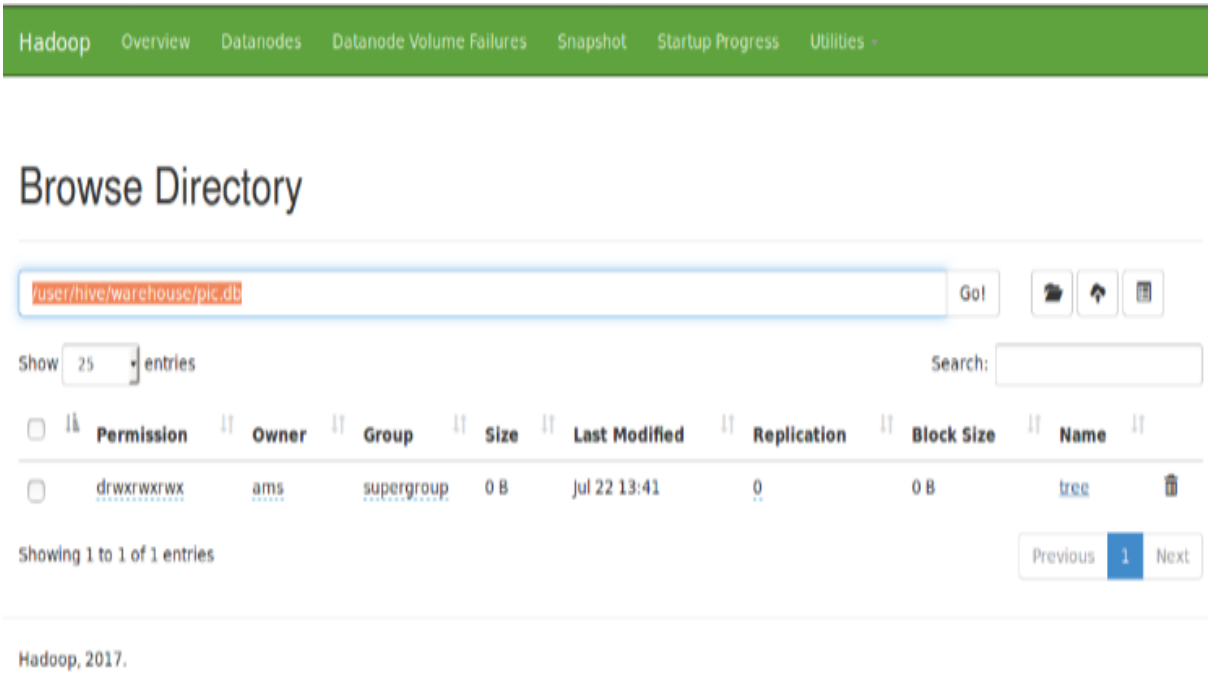
```
ogging initialized using configuration in jar:file:/home/ams/Desktop/hive/apache-hive-2.1.1-bin/lib/hive-common-2.1.1.jar!/hive
og4j2.properties Async: true
at Jun 30 11:37:32 IST 2018 WARN: Establishing SSL connection without server's identity verification is not recommended. Accord
g to MySQL 5.5.45+, 5.6.26+ and 5.7.6+ requirements SSL connection must be established by default if explicit option isn't set.
or compliance with existing applications not using SSL the verifyServerCertificate property is set to 'false'. You need either
 explicitly disable SSL by setting useSSL=false, or set useSSL=true and provide truststore for server certificate verification.
at Jun 30 11:37:32 IST 2018 WARN: Establishing SSL connection without server's identity verification is not recommended. Accord
g to MySQL 5.5.45+, 5.6.26+ and 5.7.6+ requirements SSL connection must be established by default if explicit option isn't set.
or compliance with existing applications not using SSL the verifyServerCertificate property is set to 'false'. You need either
 explicitly disable SSL by setting useSSL=false, or set useSSL=true and provide truststore for server certificate verification.
at Jun 30 11:37:32 IST 2018 WARN: Establishing SSL connection without server's identity verification is not recommended. Accord
g to MySQL 5.5.45+, 5.6.26+ and 5.7.6+ requirements SSL connection must be established by default if explicit option isn't set.
or compliance with existing applications not using SSL the verifyServerCertificate property is set to 'false'. You need either
 explicitly disable SSL by setting useSSL=false, or set useSSL=true and provide truststore for server certificate verification.
at Jun 30 11:37:32 IST 2018 WARN: Establishing SSL connection without server's identity verification is not recommended. Accord
g to MySQL 5.5.45+, 5.6.26+ and 5.7.6+ requirements SSL connection must be established by default if explicit option isn't set.
or compliance with existing applications not using SSL the verifyServerCertificate property is set to 'false'. You need either
 explicitly disable SSL by setting useSSL=false, or set useSSL=true and provide truststore for server certificate verification.
at Jun 30 11:37:34 IST 2018 WARN: Establishing SSL connection without server's identity verification is not recommended. Accord
g to MySQL 5.5.45+, 5.6.26+ and 5.7.6+ requirements SSL connection must be established by default if explicit option isn't set.
or compliance with existing applications not using SSL the verifyServerCertificate property is set to 'false'. You need either
 explicitly disable SSL by setting useSSL=false, or set useSSL=true and provide truststore for server certificate verification.
at Jun 30 11:37:34 IST 2018 WARN: Establishing SSL connection without server's identity verification is not recommended. Accord
g to MySQL 5.5.45+, 5.6.26+ and 5.7.6+ requirements SSL connection must be established by default if explicit option isn't set.
or compliance with existing applications not using SSL the verifyServerCertificate property is set to 'false'. You need either
 explicitly disable SSL by setting useSSL=false, or set useSSL=true and provide truststore for server certificate verification.
at Jun 30 11:37:34 IST 2018 WARN: Establishing SSL connection without server's identity verification is not recommended. Accord
g to MySQL 5.5.45+, 5.6.26+ and 5.7.6+ requirements SSL connection must be established by default if explicit option isn't set.
or compliance with existing applications not using SSL the verifyServerCertificate property is set to 'false'. You need either
 explicitly disable SSL by setting useSSL=false, or set useSSL=true and provide truststore for server certificate verification.
at Jun 30 11:37:34 IST 2018 WARN: Establishing SSL connection without server's identity verification is not recommended. Accord
g to MySQL 5.5.45+, 5.6.26+ and 5.7.6+ requirements SSL connection must be established by default if explicit option isn't set.
or compliance with existing applications not using SSL the verifyServerCertificate property is set to 'false'. You need either
 explicitly disable SSL by setting useSSL=false, or set useSSL=true and provide truststore for server certificate verification.
ive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine
.e. spark, tez) or using Hive 1.X releases.
```

- Creating table.

```
hive> create table tree (tree_id bigint,block_id bigint,created_at string,tree_dbh float,stump_diam float,
curb_loc string,status string,health string,spc_latin string,spc_common string,steward string,guards strin
g,sidewalk string,user_type string,problems string,root_stone string,root_grate string,root_other string,t
runk_wire string,trnk_light string,trnk_other string,brch_light string,brch_shoe string,brch_other string,
address string,postcode string,zip_city string,community_board string,borocode string,borough string,cncld
ist string,st_assem string,st_senate string,nta string,nta_name string,boro_ct string,state string,latitud
e float,longitude float,x_sp float,y_sp float,council_district string,census_tract bigint,bin bigint,bbl b
igint)
    > row format delimited
    >
    > fields terminated by ',';
OK
Time taken: 0.774 seconds
hive> describe tree;
OK
tree_id                 bigint
block_id                bigint
created_at              string
tree_dbh                float
stump_diam              float
curb_loc                string
status                  string
health                  string
spc_latin               string
spc_common              string
steward                 string
guards                  string
sidewalk                string
user_type               string
```

```
root_grate              string
root_other              string
trunk_wire              string
trnk_light              string
trnk_other              string
brch_light              string
brch_shoe               string
brch_other              string
address                 string
postcode                string
zip_city                string
community_board         string
borocode                string
borough                 string
cncldist                string
st_assem                string
st_senate               string
nta                     string
nta_name                string
boro_ct                 string
state                   string
latitude                float
longitude               float
x_sp                    float
y_sp                    float
council_district        string
census_tract            bigint
bin                     bigint
bbl                     bigint
Time taken: 0.161 seconds, Fetched: 45 row(s)
```

- **Created table in Hive:**



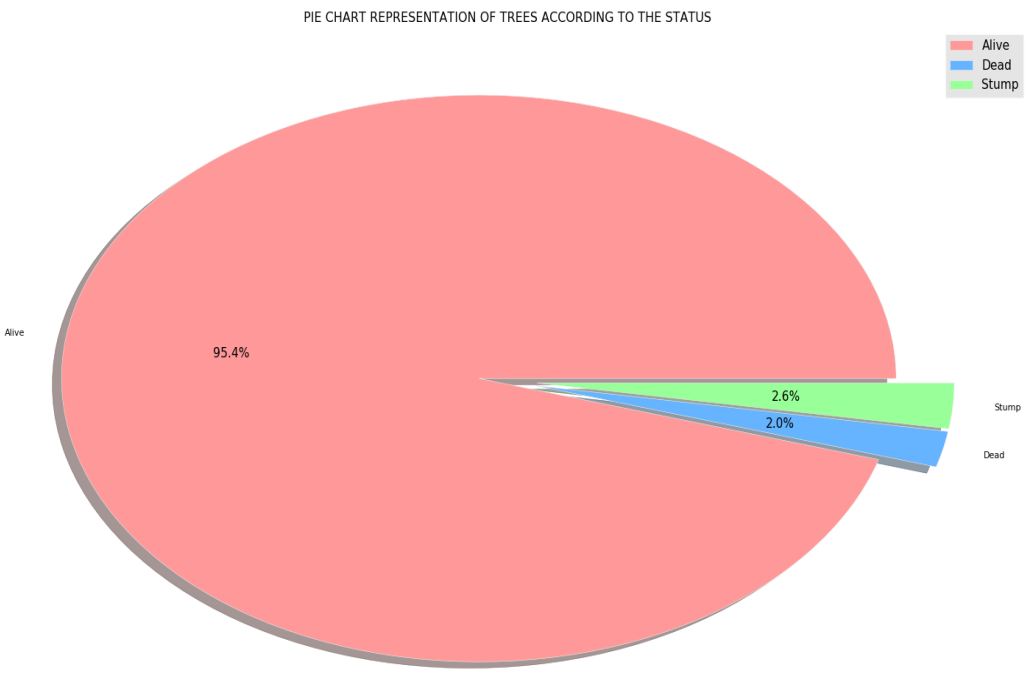
3.4 SOLVING VARIOUS QUERIES TO FETCH THE REQUIRED INFORMATION FROM THE TABLE.

3.4.1 The number of trees as per their condition (good, poor).

```
hive> insert overwrite local directory '/home/ams/Desktop/tree_analysis/status_wise_count'

    > row format delimited
    > fields terminated by ','
    > select status,count(tree_id) from tree  group by status;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider usin
 engine (i.e. spark, tez) or using Hive 1.X releases.
Query ID = ams_20180718003344_d4591370-51c7-4ab9-ab5f-8cb52ec57ded
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Job running in-process (local Hadoop)
2018-07-18 00:33:45,770 Stage-1 map = 0%,  reduce = 0%
2018-07-18 00:33:46,779 Stage-1 map = 100%,  reduce = 100%
Ended Job = job_local2126918952_0014
Moving data to local directory /home/ams/Desktop/tree_analysis/status_wise_count
MapReduce Jobs Launched:
Stage-Stage-1:  HDFS Read: 4528784280 HDFS Write: 0 SUCCESS
Total MapReduce CPU Time Spent: 0 msec
OK
Time taken: 2.484 seconds
hive>
```

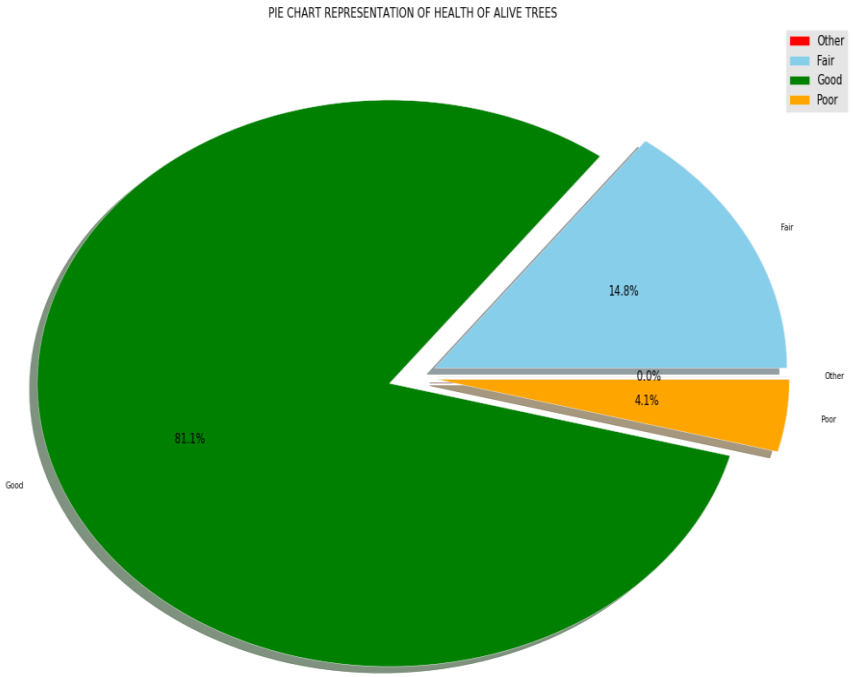
- **RESULT OF THE ABOVE QUERY:**



3.4.2 The number of alive trees as per their health.

```
hive> insert overwrite local directory '/home/ams/Desktop/tree_analysis/status_wise_count_of_alive_trees'
    > row format delimited
    > fields terminated by ','
    > select health,count(tree_id) from tree where status="Alive" group by health;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution
 engine (i.e. spark, tez) or using Hive 1.X releases.
Query ID = ams_20180630121041_80e2a28b-4b1e-4d4c-a27b-5fa62984263d
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Job running in-process (local Hadoop)
2018-06-30 12:10:42,745 Stage-1 map = 0%,  reduce = 0%
2018-06-30 12:10:43,763 Stage-1 map = 100%,  reduce = 100%
Ended Job = job_local1575475962_0006
Moving data to local directory /home/ams/Desktop/tree_analysis/status_wise_count_of_alive_trees
MapReduce Jobs Launched:
Stage-Stage-1:  HDFS Read: 2717270568 HDFS Write: 0 SUCCESS
Total MapReduce CPU Time Spent: 0 msec
OK
Time taken: 2.475 seconds
hive>
```

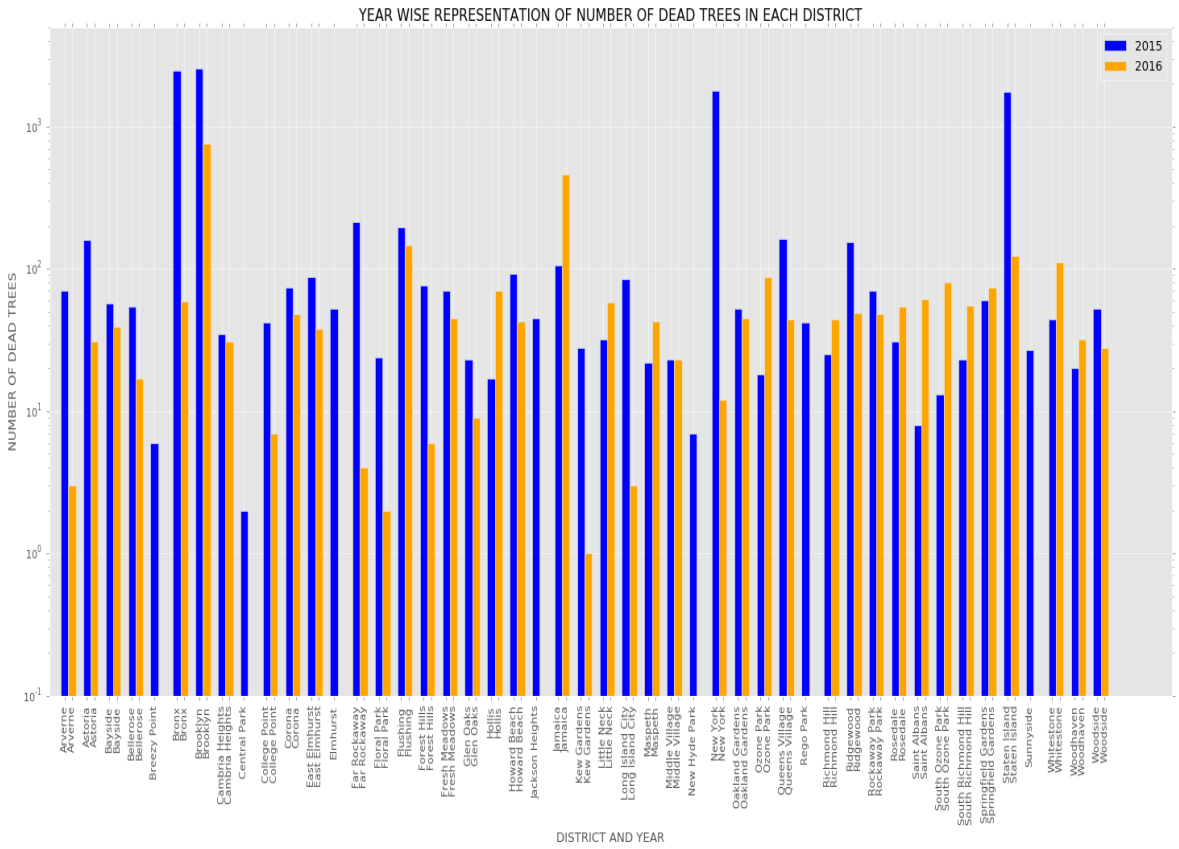
❖ **RESULT OF THE ABOVE QUERY:**



3.4.3 The number of dead trees in each district yearwise.

```
tion specification
hive>
hive> insert overwrite local directory '/home/ams/Desktop/tree_analysis/districtyear'
    > row format delimited
    > fields terminated by ','
    > select zip_city,YEAR(created_at),count(tree_id) from tree where status="Dead"group by zip_city,YEAR(
created_at);
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider usin
g a different execution engine (i.e. spark, tez) or using Hive 1.X releases.
Query ID = ams_20180711121038_73dc1d27-220b-4af6-9b82-1196af60adfa
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Job running in-process (local Hadoop)
2018-07-11 12:10:39,553 Stage-1 map = 0%,  reduce = 0%
2018-07-11 12:10:40,567 Stage-1 map = 100%,  reduce = 100%
Ended Job = job_local1503330837_0002
Moving data to local directory /home/ams/Desktop/tree_analysis/districtyear
MapReduce Jobs Launched:
Stage-Stage-1:  HDFS Read: 905756856 HDFS Write: 0 SUCCESS
Total MapReduce CPU Time Spent: 0 msec
OK
Time taken: 2.54 seconds
hive>
```

❖ **RESULT OF THE ABOVE QUERY:**

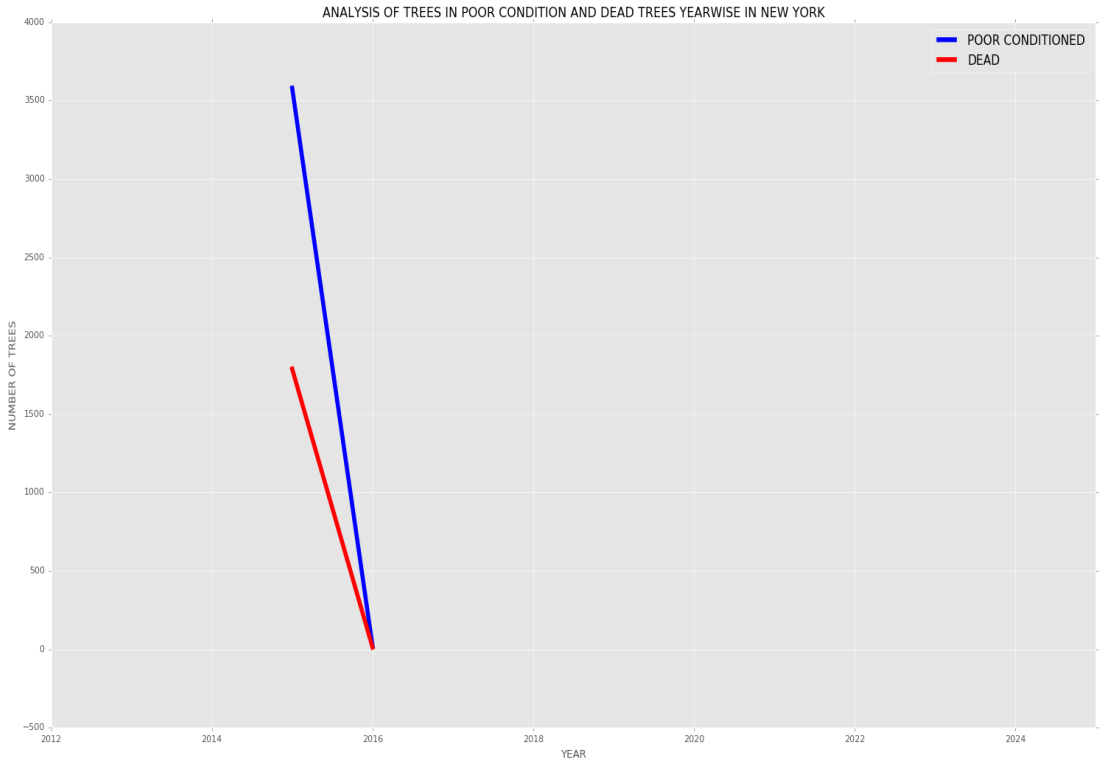


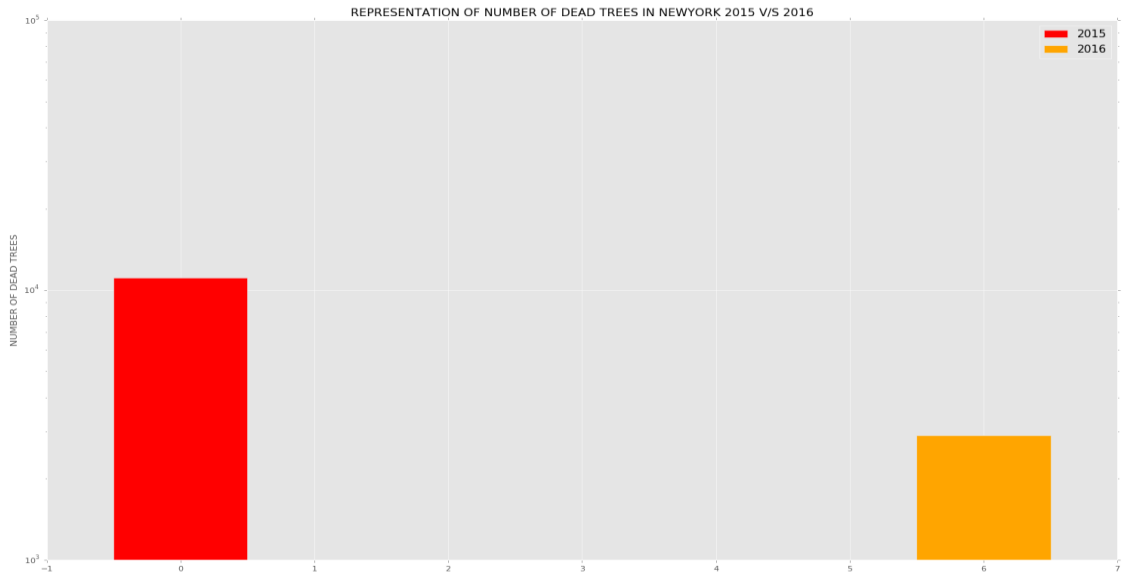
3.4.4 The number of dead trees in New York 2015 v/s 2016.

```
Time taken: 0.052 seconds
hive> insert overwrite local directory '/home/ams/Desktop/tree_analysis/stateyear/'

    > row format delimited
    > fields terminated by ','
    > select state,YEAR(created_at),count(tree_id) from tree where status="Dead"group by state,YEAR(create
d_at);
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider usin
g a different execution engine (i.e. spark, tez) or using Hive 1.X releases.
Query ID = ams_20180711120631_617c1af7-7ce2-4345-aebc-9f9766f518bb
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Job running in-process (local Hadoop)
2018-07-11 12:06:35,846 Stage-1 map = 0%,  reduce = 0%
2018-07-11 12:06:40,861 Stage-1 map = 100%,  reduce = 100%
Ended Job = job_local403663176_0001
Moving data to local directory /home/ams/Desktop/tree_analysis/stateyear
MapReduce Jobs Launched:
Stage-Stage-1:  HDFS Read: 452878428 HDFS Write: 0 SUCCESS
Total MapReduce CPU Time Spent: 0 msec
OK
Time taken: 9.334 seconds
hive>
```

❖ RESULT OF THE ABOVE QUERY:

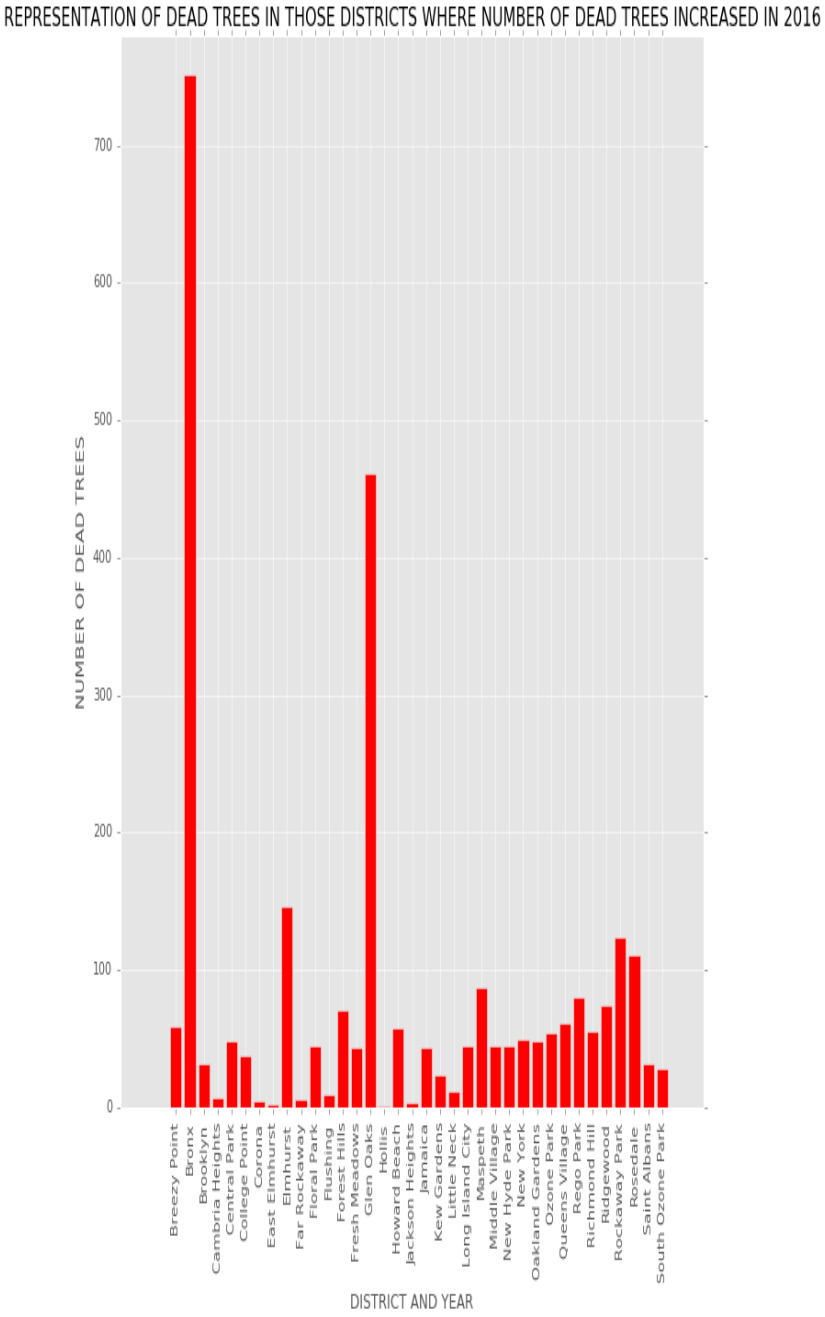




3.4.5 The districts where number of dead trees increased in 2016.

```
hive> insert overwrite local directory '/home/ams/Desktop/tree_analysis/2015'
    > row format delimited
    > fields terminated by ','
    > select zip_city,count(tree_id) from tree where status="Dead" and YEAR(created_at)=2015 group by zip_city;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e. spark, tez) or using Hive 1.X releas
.
Query ID = ams_20180713233311_6be3a5e2-1c2c-4926-bce0-59004fdaff21
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Job running in-process (local Hadoop)
2018-07-13 23:33:12,656 Stage-1 map = 0%,  reduce = 0%
2018-07-13 23:33:13,671 Stage-1 map = 100%,  reduce = 100%
Ended Job = job_local367468758_0002
Moving data to local directory /home/ams/Desktop/tree_analysis/2015
MapReduce Jobs Launched:
Stage-Stage-1:  HDFS Read: 905756856 HDFS Write: 0 SUCCESS
Total MapReduce CPU Time Spent: 0 msec
OK
Time taken: 2.496 seconds
hive> insert overwrite local directory '/home/ams/Desktop/tree_analysis/2015/2016'
    > row format delimited
    > fields terminated by ','
    > select zip_city,count(tree_id) from tree where status="Dead" and YEAR(created_at)=2016 group by zip_city;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e. spark, tez) or using Hive 1.X releas
.
Query ID = ams_20180713233416_98c55026-b82e-4882-988e-61c28173b7a7
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Job running in-process (local Hadoop)
2018-07-13 23:34:17,923 Stage-1 map = 0%,  reduce = 0%
2018-07-13 23:34:18,945 Stage-1 map = 100%,  reduce = 100%
Ended Job = job_local1343070655_0003
Moving data to local directory /home/ams/Desktop/tree_analysis/2015/2016
MapReduce Jobs Launched:
Stage-Stage-1:  HDFS Read: 1358635284 HDFS Write: 0 SUCCESS
Total MapReduce CPU Time Spent: 0 msec
OK
```

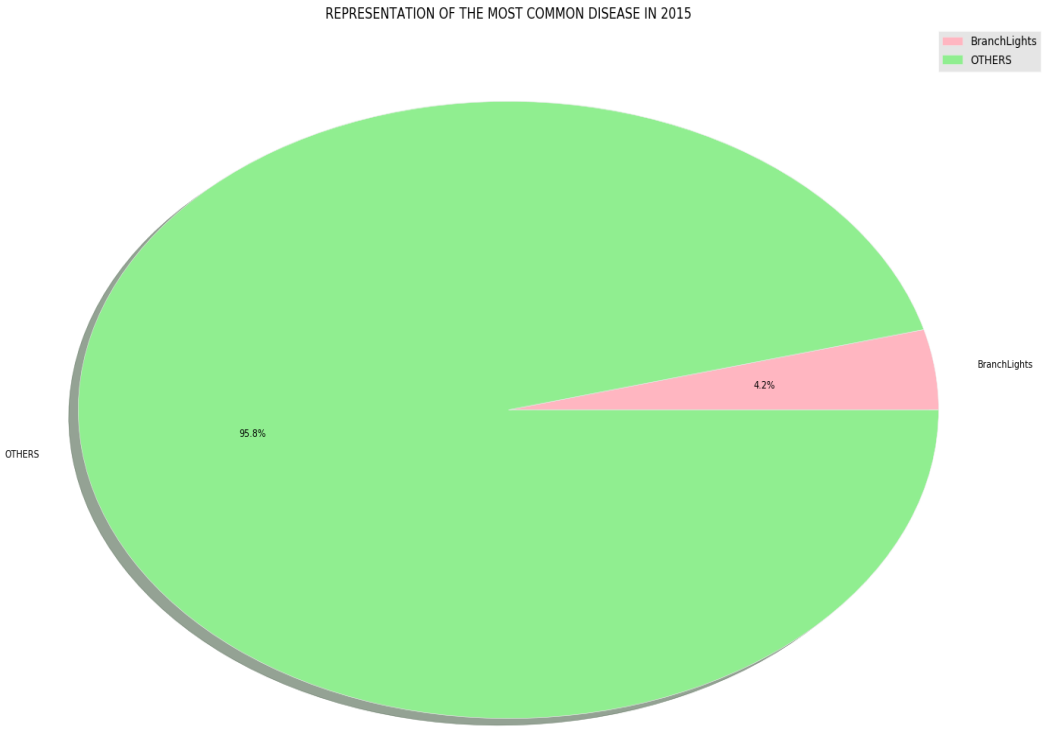
❖ **RESULT OF THE ABOVE QUERY:**

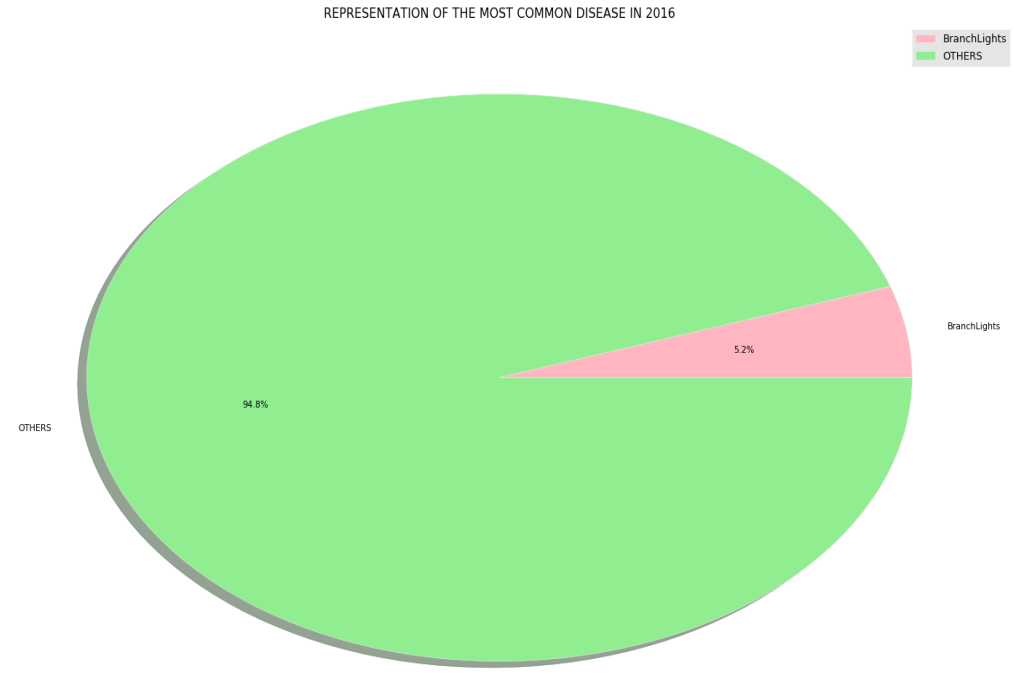


3.4.6 The most common disease among trees found in the year 2015 & 2016:

```
Ended Job = job_local1092852541_0001
Moving data to local directory /home/ams/Desktop/tree_analysis/problem_count
MapReduce Jobs Launched:
Stage-Stage-1:  HDFS Read: 452878428 HDFS Write: 0 SUCCESS
Total MapReduce CPU Time Spent: 0 msec
OK
Time taken: 11.507 seconds
hive>
hive> insert overwrite local directory '/home/ams/Desktop/tree_analysis/problem_wise_trees'
    > row format delimited
    > fields terminated by ','
    > select tree_id,address,postcode,zip_city,problems,YEAR(created_at) from tree group by problems,YEAR(created_at),zip_city,po
stcode,address,tree_id;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution
 engine (i.e. spark, tez) or using Hive 1.X releases.
Query ID = ams_20180717043023_56b783c3-e048-441e-b3b4-83ac18483db4
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Job running in-process (local Hadoop)
2018-07-17 04:30:25,147 Stage-1 map = 0%,  reduce = 0%
2018-07-17 04:30:30,154 Stage-1 map = 100%,  reduce = 0%
2018-07-17 04:30:32,166 Stage-1 map = 100%,  reduce = 100%
Ended Job = job_local1106653978_0002
Moving data to local directory /home/ams/Desktop/tree_analysis/problem_wise_trees
MapReduce Jobs Launched:
Stage-Stage-1:  HDFS Read: 905756856 HDFS Write: 0 SUCCESS
Total MapReduce CPU Time Spent: 0 msec
OK
Time taken: 8.624 seconds
hive>
```

❖ RESULT OF THE ABOVE QUERY:

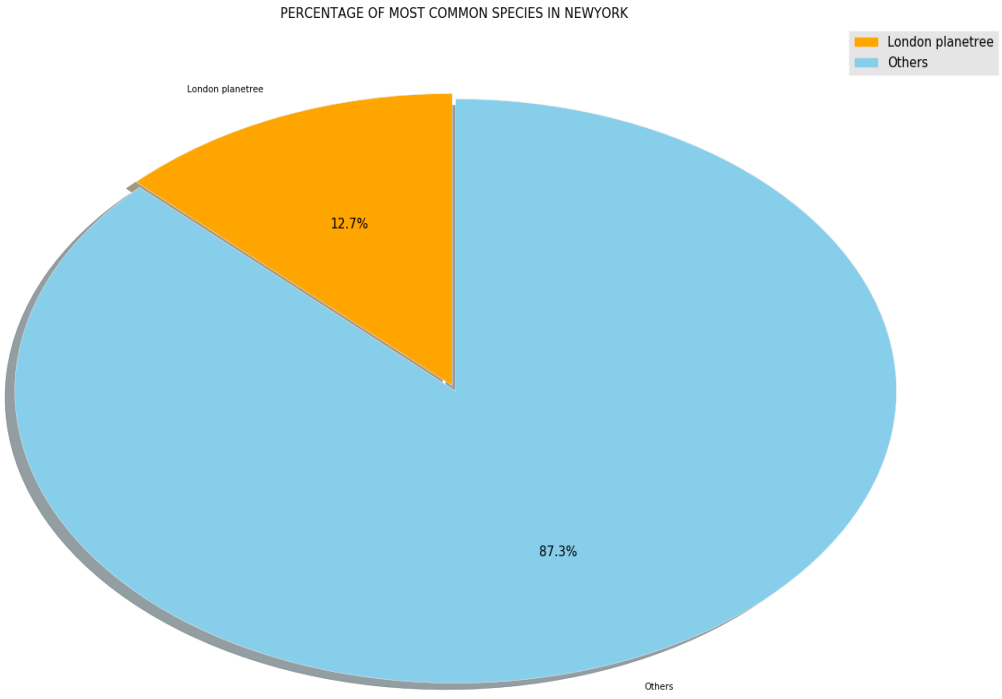




3.4.7 The most common species found in New York :

```
hive> insert overwrite local directory '/home/ams/Desktop/tree_analysis/top5species'
    > row format delimited
    > fields terminated by ','
    > select spc_common,spc_latin,count(tree_id) t from tree group by spc_common,spc_latin order by t  DESC limit 5;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution
 engine (i.e. spark, tez) or using Hive 1.X releases.
Query ID = ams_20180712232933_6359a383-aa63-43b6-92d5-4fc87fffe09c
Total jobs = 2
Launching Job 1 out of 2
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Job running in-process (local Hadoop)
2018-07-12 23:29:34,624 Stage-1 map = 0%,  reduce = 0%
2018-07-12 23:29:35,635 Stage-1 map = 100%,  reduce = 100%
Ended Job = job_local620115973_0008
Launching Job 2 out of 2
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Job running in-process (local Hadoop)
2018-07-12 23:29:36,959 Stage-2 map = 100%,  reduce = 100%
Ended Job = job_local947190331_0009
Moving data to local directory /home/ams/Desktop/tree_analysis/top5species
MapReduce Jobs Launched:
Stage-Stage-1:  HDFS Read: 3170148996 HDFS Write: 0 SUCCESS
Stage-Stage-2:  HDFS Read: 3170148996 HDFS Write: 0 SUCCESS
Total MapReduce CPU Time Spent: 0 msec
OK
```

❖ RESULT OF THE ABOVE QUERY:

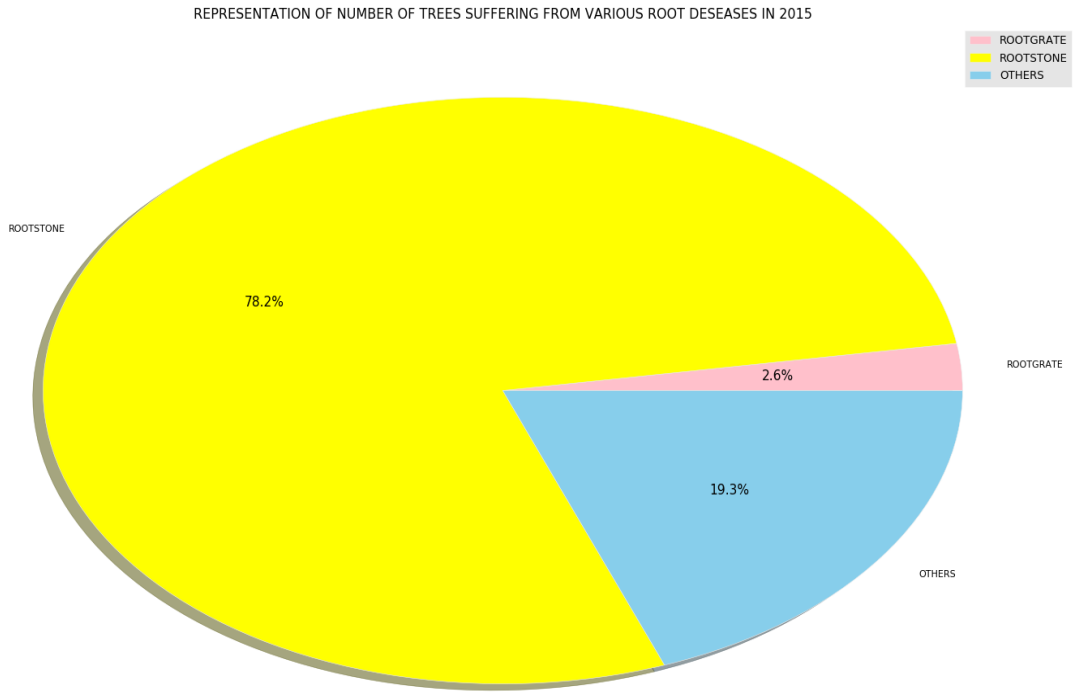


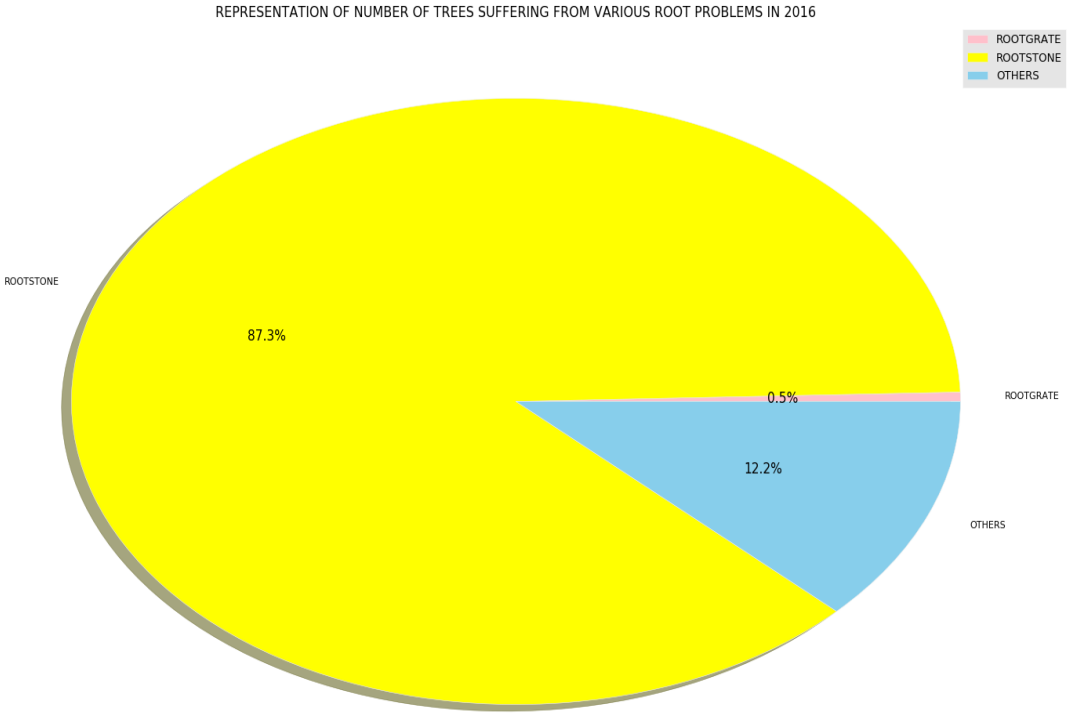
3.4.8 The analysis of various roots related problems..

```
hive> insert overwrite local directory '/home/ams/Desktop/tree_analysis/problem_count/2015/root_grate'
    > row format delimited
    > fields terminated by ','
    > select count(tree_id) from tree where YEAR(created_at)=2015 and root_grate='Yes';
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider usin
g a different execution engine (i.e. spark, tez) or using Hive 1.X releases.
Query ID = ams_20180724162836_0396869b-241d-4139-91ea-6a0883b6445b
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Job running in-process (local Hadoop)
2018-07-24 16:28:38,232 Stage-1 map = 0%,  reduce = 0%
2018-07-24 16:28:40,248 Stage-1 map = 100%,  reduce = 100%
Ended Job = job_local2000394388_0003
Moving data to local directory /home/ams/Desktop/tree_analysis/problem_count/2015/root_grate
MapReduce Jobs Launched:
Stage-Stage-1:  HDFS Read: 1358635284 HDFS Write: 0 SUCCESS
Total MapReduce CPU Time Spent: 0 msec
OK
Time taken: 3.469 seconds
hive>
```

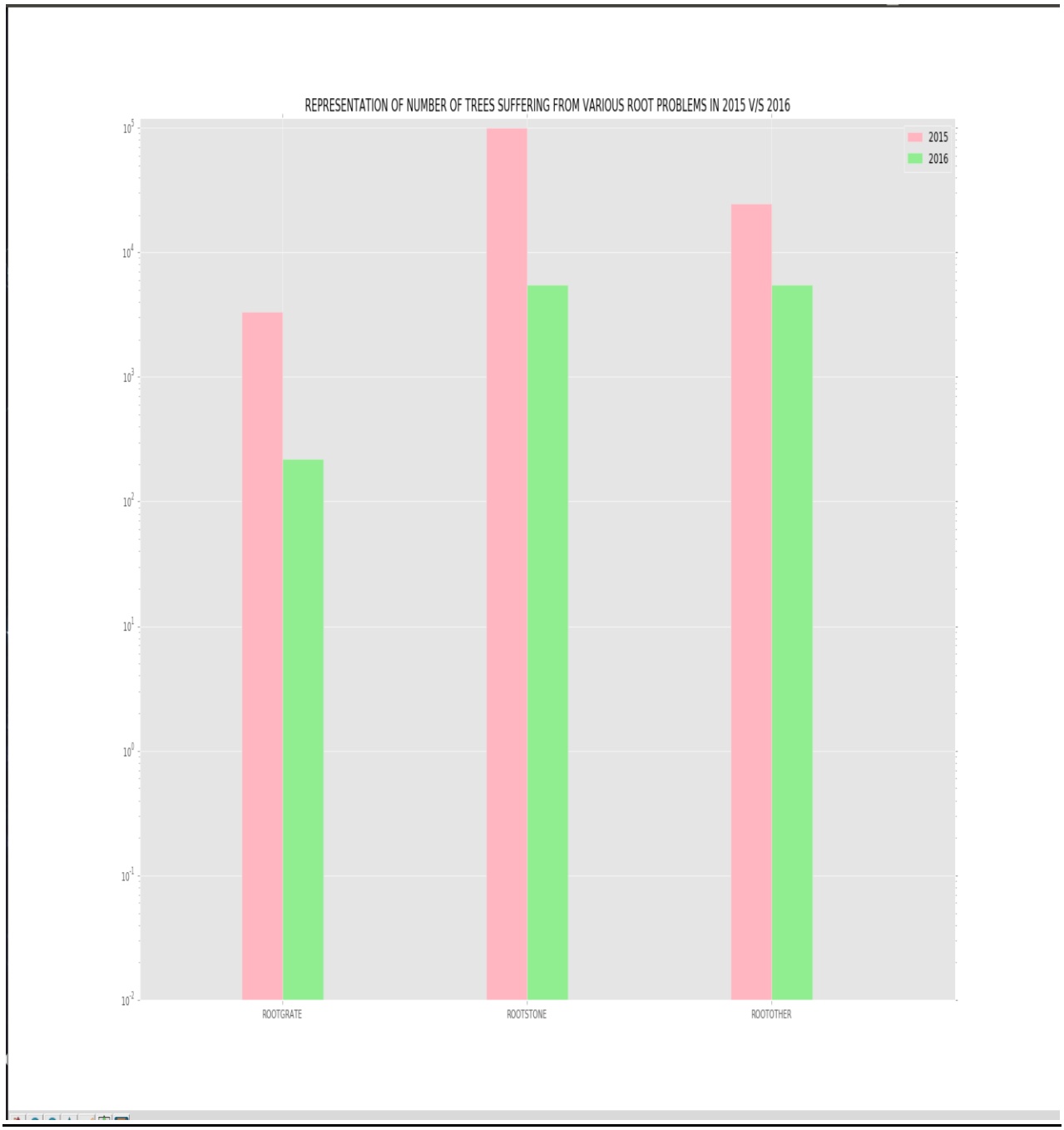
```
hive> insert overwrite local directory '/home/ams/Desktop/tree_analysis/problem_count/2015/root_other'
    > row format delimited
    > fields terminated by ','
    > select count(tree_id) from tree where YEAR(created_at)=2015 and root_other='Yes';
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider usin
g a different execution engine (i.e. spark, tez) or using Hive 1.X releases.
Query ID = ams_20180724162958_81df6077-f456-43a9-b193-3e26cc10d313
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Job running in-process (local Hadoop)
2018-07-24 16:29:59,594 Stage-1 map = 0%,  reduce = 0%
2018-07-24 16:30:00,603 Stage-1 map = 100%,  reduce = 0%
2018-07-24 16:30:01,613 Stage-1 map = 100%,  reduce = 100%
Ended Job = job_local1505111948_0004
Moving data to local directory /home/ams/Desktop/tree_analysis/problem_count/2015/root_other
MapReduce Jobs Launched:
Stage-Stage-1:  HDFS Read: 1811513712 HDFS Write: 0 SUCCESS
Total MapReduce CPU Time Spent: 0 msec
OK
Time taken: 3.496 seconds
hive>
```

❖ RESULT OF THE ABOVE QUERY:





3.4.9 The number of trees affected by root diseases 2015 v/s 2016.



3.4.10 Analysis of the guards provided around the trees to protect them .

- ***Number of trees without guards 2015 v/s 2016.***

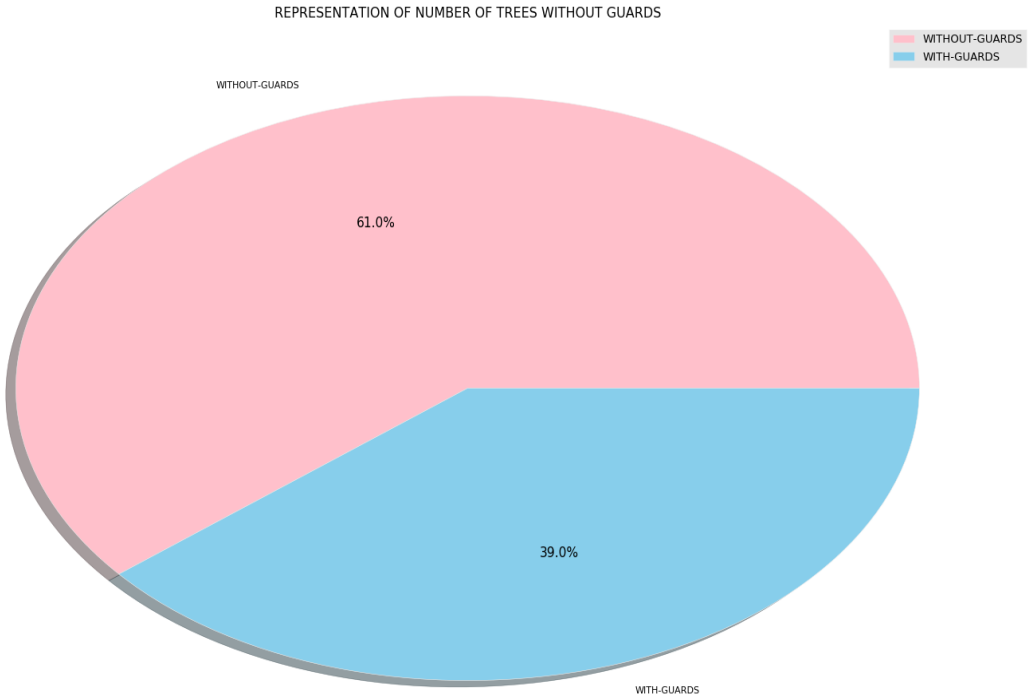
```
hive> insert overwrite local directory '/home/ams/Desktop/tree_analysis/poor_condition'
    > row format delimited
    > fields terminated by ','
    > select tree_id,block_id,status,health,spc_latin,spc_common,problems,address,postcode,zip_city,latitude,longitude,guards,sid
ewalk from tree where health="Poor";
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution
 engine (i.e. spark, tez) or using Hive 1.X releases.
Query ID = ams_20180630120307_1527e5ce-dfad-42ff-bfd9-f11fb8eb2c2c
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks is set to 0 since there's no reduce operator
Job running in-process (local Hadoop)
2018-06-30 12:03:08,763 Stage-1 map = 0%,  reduce = 0%
2018-06-30 12:03:09,774 Stage-1 map = 100%,  reduce = 0%
Ended Job = job_local1553998248_0004
Moving data to local directory /home/ams/Desktop/tree_analysis/poor_condition
MapReduce Jobs Launched:
Stage-Stage-1:  HDFS Read: 905756856 HDFS Write: 0 SUCCESS
Total MapReduce CPU Time Spent: 0 msec
OK
Time taken: 2.471 seconds
hive>
```

- ***Number of trees with harmful guards .***

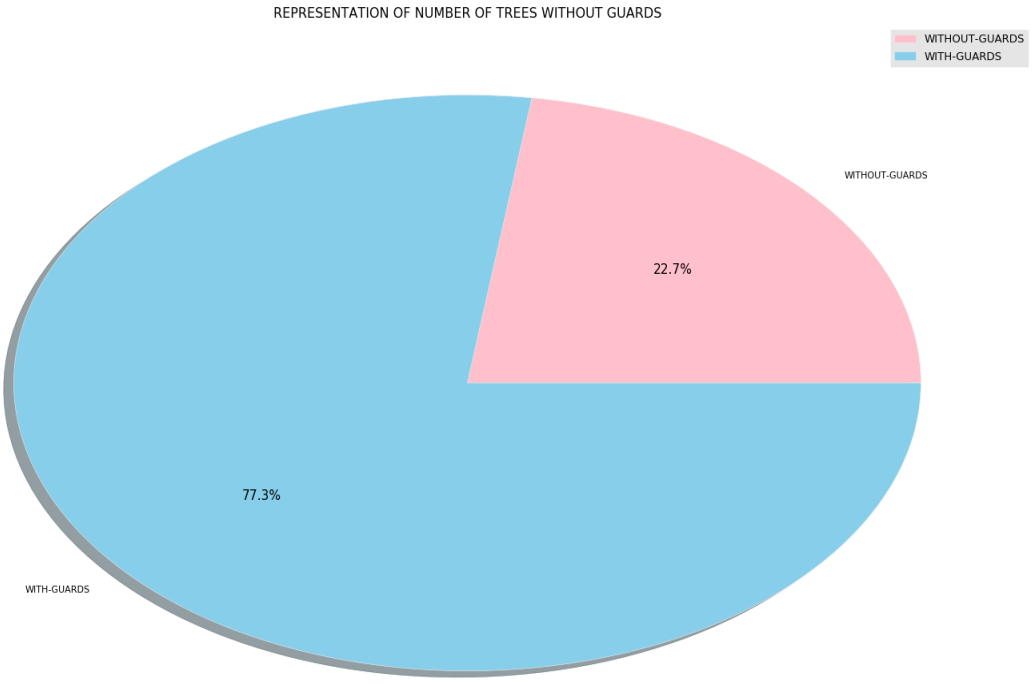
```
hive> insert overwrite local directory '/home/ams/Desktop/tree_analysis/trees_with_harmful_guards'
    > row format delimited
    > fields terminated by ','
    > select tree_id,block_id,tree_dbh,stump_diam,status,address,postcode,zip_city from tree where status="Alive" and guards="Har
mful" group by tree_id,block_id,tree_dbh,stump_diam,status,address,postcode,zip_city ;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution
 engine (i.e. spark, tez) or using Hive 1.X releases.
Query ID = ams_20180630123606_4b09b33b-0c7f-400b-adc0-855a1bfbbedb
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Job running in-process (local Hadoop)
2018-06-30 12:36:07,441 Stage-1 map = 0%,  reduce = 0%
2018-06-30 12:36:08,447 Stage-1 map = 100%,  reduce = 100%
Ended Job = job_local1155361022_0008
Moving data to local directory /home/ams/Desktop/tree_analysis/trees_with_harmful_guards
MapReduce Jobs Launched:
Stage-Stage-1:  HDFS Read: 3623027424 HDFS Write: 0 SUCCESS
Total MapReduce CPU Time Spent: 0 msec
OK
Time taken: 2.45 seconds
hive>
```

❖ **RESULT OF THE ABOVE QUERY:**

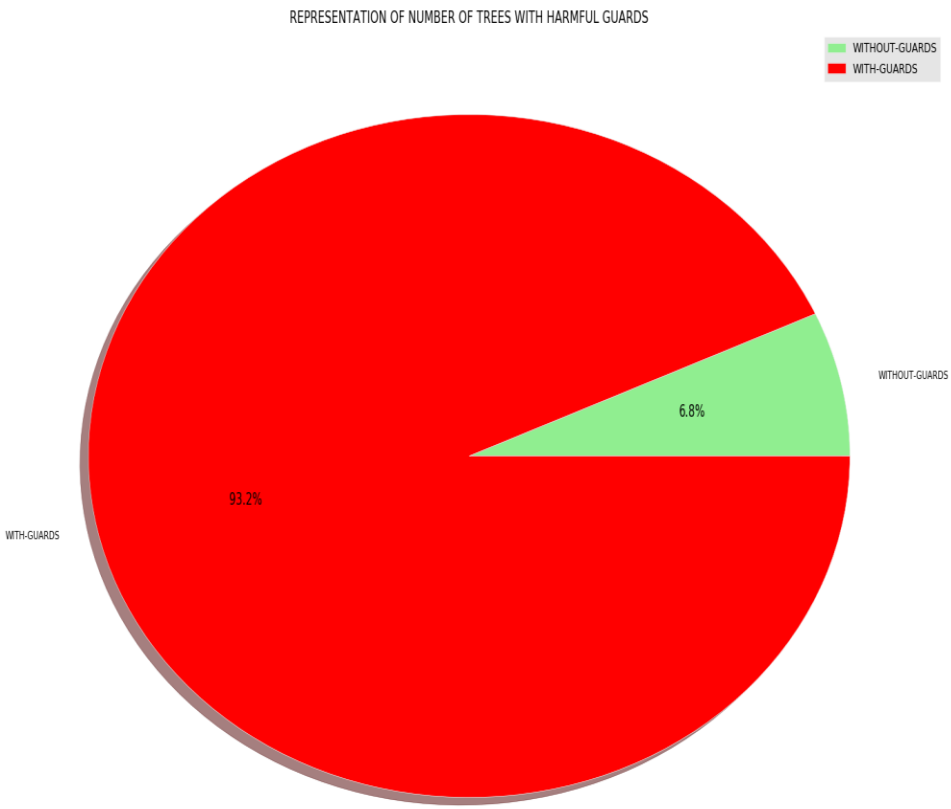
➢ *Year 2015*



➢ *Year 2016*



- ***Number of trees with harmful guards.***



CHAPTER 4 : SUMMARY

So from the results shown it can be summarized that the 'Tree Census Data Analysis' can really assist in taking care of trees. The extraction of useful information from a big dataset in seconds will really help the organizations to make decisions. Through this analysis, the graphs, the statistics, and the results obtained will really help to make policies and plans in an effective manner that is which area requires how much efforts and what type of changes are to be done can be visualized easily. The analysis regarding diseases will force the organizations to adopt the preventive measures to prevent trees from the particular disease.

The key features and benefits of the project:

- Fast response to the user's request.
- Attractive and informative results obtained in the form of pictorial representation from a huge and messy dataset.
- Time consumption really reduced as compared to traditional ways of analyzing.
- The results can be modified with little efforts if there is any change in the dataset or dataset get updated.

CHAPTER 5 : CONCLUSION

The ‘Tree Census Data Analysis’ project concludes that Big Data analysis is capable of transforming a raw data collected over years into fruitful results with very little efforts. As in this particular case the results obtained are really helpful and easy to understand and analyze as compared to the original dataset. The results are drawn from the same dataset and its really awesome to see such informative graphs extracted from raw information.

Hadoop, Hive made it very easy and comfortable to store a table containing more than 600000 rows. The queries were solved in seconds, the speed with which the information was fetched is much faster than that of the traditional tools.

The matplotlib and numpy (libraries of python) helped a lot to plot and convert the information fetched from the dataset into informative and attractive pictorial representations.

At the end we have all the required information which an organization would definitely require to design any policy and plan to prevent the trees from degradation. There is graphical representation of various diseases the districts affected most which helps to know which areas require the earliest action. In this way the project has come up with great results which are of much importance in the terms of analysis.

CHAPTER 6 : REFERENCES

WEBSITES :

- www.tutorialpoint.com
- www.pythonprogramming.net
- www.stackoverflow.com
- www.kaggle.com
- www.edureka.com