

Vidyavardhini’s

**College of Engineering & Technology**

Vasai Road (W)

**Department of Computer Engineering**

**Laboratory Manual**

|  |  |  |  |
| --- | --- | --- | --- |
| Semester | VII | **Class** | B.E |
| Course Code | CSL702 | | |
| Course Name | Big Data Analytics Lab | | |



**Vidyavardhini’s College of Engineering & Technology**

**Vision**

To be a premier institution of technical education; always aiming at becoming a valuable resource for industry and society.

**Mission**

* To provide technologically inspiring environment for learning.
* To promote creativity, innovation and professional activities.
* To inculcate ethical and moral values.
* To cater personal, professional and societal needs through quality education.

# Department Vision:

To evolve as a center of excellence in the field of Computer Engineering to cater to industrial and societal needs.

# Department Mission:

* To provide quality technical education with the aid of modern resources.
* Inculcate creative thinking through innovative ideas and project development.
* To encourage life-long learning, leadership skills, entrepreneurship skills with ethical & moral values.

# Program Education Objectives (PEOs):

PEO1: To facilitate learners with a sound foundation in the mathematical, scientific and engineering fundamentals to accomplish professional excellence and succeed in higher studies in Computer Engineering domain

PEO2: To enable learners to use modern tools effectively to solve real-life problems in the field of Computer Engineering.

PEO3: To equip learners with extensive education necessary to understand the

impact of computer technology in a global and social context.

PEO4: To inculcate professional and ethical attitude, leadership qualities, commitment to societal responsibilities and prepare the learners for life-long learning to build up a successful career in Computer Engineering.

# Program Specific Outcomes (PSOs):

PSO1: Analyze problems and design applications of database, networking, security, web technology, cloud computing, machine learning using mathematical skills, and computational tools.

PSO2: Develop computer-based systems to provide solutions for organizational, societal problems by working in multidisciplinary teams and pursue a career in the IT industry.

# Program Outcomes (POs):

Engineering Graduates will be able to:

* **PO1. Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
* **PO2. Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
* **PO3. Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
* **PO4. Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
* **PO5. Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
* **PO6. The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
* **PO7. Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
* **PO8. Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
* **PO9. Individual and teamwork:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
* **PO10. Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
* **PO11. Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one’s own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
* **PO12. Life-long learning:** Recognize the need for and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

**Course Objectives**

|  |  |
| --- | --- |
| 1 | Solve Big Data problems using Map Reduce Technique and apply to various algorithms. |
| 2 | Identify various types of NoSQL databases and execute NOSQL commands |
| 3 | Understand implementation of various analytic techniques using Hive/PIG/R/Tableau, etc. |
| 4 | Apply streaming analytics to real time applications. |

**Course Outcomes**

|  |  |  |  |
| --- | --- | --- | --- |
| At the end of the course student will be able to: | | Action verb | Bloom Level |
| CSL702.1 | Use Sqoop tool in Hadoop ecosystem for big data analytics. | Identify | Understand (Level 2) |
| CSL702.2 | Implement Map Reduce algorithm on structured and unstructured data | Apply | Apply (Level 3) |
| CSL702.3 | Perform NoSQL commands on Cassandra, Hadoop HBase and MongoDB | Identify | Use (Level 4) |
| CSL702.4 | Implement filtering, counting distinct element and counting ones in window algorithms on data stream. | Apply | Apply (Level 3) |
| CSL702.5 | Implement data visualization techniques on social network graphs using R | Analyze | Analyze (Level 4) |
| CSL702.6 | Built real life application on big data analytics | Apply | Apply(Level 3) |

**Mapping of Experiments with Course Outcomes**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Course Modules | Course Outcomes | | | | | |
| CSL702.1 | CSL702.2 | CSL702.3 | CSL702.4 | CSL702.5 | CSL702.6 |
| Hadoop HDFS Practical | 3 | -- | -- | -- | -- | -- |
| Use of Sqoop tool | 3 | -- | -- | -- | -- | -- |
| To install and configure MongoDB | -- | -- | 3 | -- | -- | -- |
| Experiment on Hadoop Map-Reduce | -- | 3 | -- | -- | -- | -- |
| Create HIVE Database and Descriptive analytics-  basic statistics. | -- | -- | -- | 3 | -- | -- |
| Data Stream Algorithms | -- | -- | -- | 3 | -- | -- |
| Social Network Analysis using R | -- | -- | -- | -- | 3 | -- |
| Data Visualization using Hive/PIG/R/Tableau/. | -- | -- | -- | -- | -- | 3 |
| Mini-Project | 2 | 2 | 2 | 2 | 2 | 3 |

Enter correlation level 1, 2 or 3 as defined below

1: Slight (Low) 2: Moderate (Medium) 3: Substatial (High) If there is no correlation put “—“.

**INDEX**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Sr.**  **No.** | **Name of Experiment** | **D.O.P.** | **D.O.C.** | **Page No.** | **Remark** |
| 1 | **Hadoop HDFS Practical:** -HDFS Basics, Hadoop Ecosystem Tools Overview. -Installing Hadoop. - Copying File to Hadoop. -Copy from Hadoop File system and deleting file.  -Moving and displaying files in HDFS. -Programming exercises on  Hadoop |  |  |  |  |
| 2 | **Use of Sqoop tool** to transfer data between Hadoop and relational database servers.   1. Sqoop - Installation. 2. To execute basic commands of   Hadoop eco system componentSqoop. |  |  |  |  |
| 3 | To install and configure MongoDB/  Cassandra/ HBase/ Hypertable to execute NoSQL commands |  |  |  |  |
| 4 | Experiment on Hadoop Map-Reduce:  -Write a program to implement a  word count program using MapReduce. |  |  |  |  |
| 5 | Create HIVE Database and  Descriptive analytics-basic statistics. |  |  |  |  |
| 6 | Data Stream Algorithms:  Implement Flajolet Martin algorithm  using any programming language |  |  |  |  |
| 7 | Social Network Analysis using R (for example: Community Detection  Algorithm) |  |  |  |  |
| 8 | Data Visualization using Hive/PIG/R/Tableau/. |  |  |  |  |
| 9 | Mini Project |  |  |  |  |

D.O.P: Date of performance

D.O.C : Date of correction

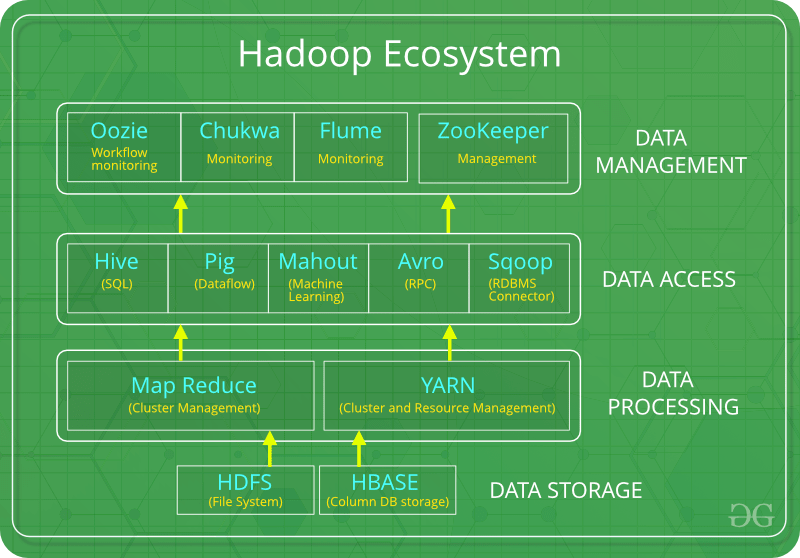
|  |
| --- |
| Experiment No.1 |
| Hadoop HDFS Practical |
| Date of Performance: |
| Date of Submission: |

**AIM** : Installation, Configuration of hadoop and performing basic file management operations in hadoop.

## THEORY :

What is the Hadoop Ecosystem?

Hadoop Ecosystem is a platform or a suite which provides various services to solve the big data problems. It includes Apache projects and various commercial tools and solutions. There are four major elements of Hadoop i.e. HDFS, MapReduce, YARN, and Hadoop Common.



Following are the components that collectively form a Hadoop ecosystem:

* HDFS: Hadoop Distributed File System
* YARN: Yet Another Resource Negotiator
* MapReduce: Programming based Data Processing
* Spark: In-Memory data processing
* PIG, HIVE: Query based processing of data services
* HBase: NoSQL Database
* Mahout, Spark MLLib: Machine Learning algorithm libraries
* Solar, Lucene: Searching and Indexing
* Zookeeper: Managing cluster
* Oozie: Job Scheduling HDFS:

HDFS is the primary or major component of Hadoop ecosystem and is responsible for storing large data sets of structured or unstructured data across various nodes and thereby maintaining the metadata in the form of log files.

HDFS consists of two core components i.e.

* Name node
* Data Node

Name Node is the prime node which contains metadata (data about data) requiring comparatively fewer resources than the data nodes that stores the actual data. These data nodes are commodity hardware in the distributed environment.

HDFS maintains all the coordination between the clusters and hardware. YARN:

Yet Another Resource Negotiator, as the name implies, YARN is the one who helps to manage the resources across the clusters. In short, it performs scheduling and resource allocation for the Hadoop System.

Resource manager has the privilege of allocating resources for the applications in a system whereas Node managers work on the allocation of resources such as CPU, memory, bandwidth per machine and later on acknowledges the resource manager. Application manager works as an interface between the resource manager and node manager and performs negotiations as per the requirement of the two.

MapReduce:

MapReduce makes the use of two functions i.e. Map() and Reduce() whose task is:

Map() performs sorting and filtering of data and thereby organizing them in the form of group. Map generates a key-value pair based result which is later on processed by the Reduce() method.

Reduce(), as the name suggests does the summarization by aggregating the mapped data. In simple, Reduce() takes the output generated by Map() as input and combines those tuples into smaller set of tuples.

HIVE:

Hive is an ETL and Data warehousing tool used to query or analyze large datasets stored within the Hadoop ecosystem. Hive has three main functions: data summarization, query, and analysis of unstructured and semi-structured data in Hadoop. It features a SQL-like interface, HQL language that works similar to SQL and automatically translates queries into MapReduce jobs.

PIG:

Pig was basically developed by Yahoo which works on a pig Latin language, which is Query based language similar to SQL. It is a platform for structuring the data flow, processing and analyzing huge data sets. Pig does the work of executing commands and in the background, all the activities of MapReduce are taken care of. After the processing, pig stores the result in HDFS.

Apache Spark:

It’s a platform that handles all the process consumptive tasks like batch processing, interactive or iterative real-time processing, graph conversions, and visualization, etc.

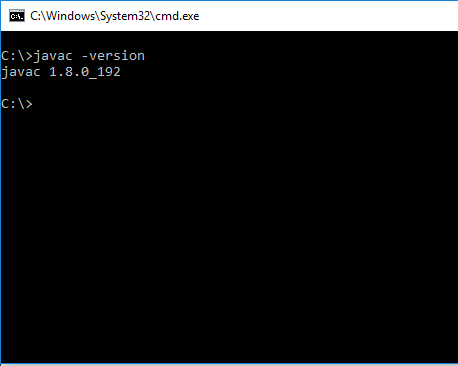
It consumes in memory resources hence, thus being faster than the prior in terms of optimization.

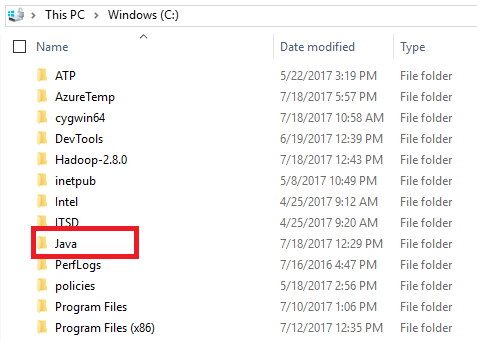
Installation of Hadoop

Download Hadoop 2.8.0 (Link: [http://www-eu.apache.org/dist/hadoop/common/hadoop-](http://www-eu.apache.org/dist/hadoop/common/hadoop-2.8.0/hadoop-2.8.0.tar.gz) [2.8.0/hadoop-2.8.0.tar.gz](http://www-eu.apache.org/dist/hadoop/common/hadoop-2.8.0/hadoop-2.8.0.tar.gz) OR [http://archive.apache.org/dist/hadoop/core//hadoop-](http://archive.apache.org/dist/hadoop/core/hadoop-2.8.0/hadoop-2.8.0.tar.gz) [2.8.0/hadoop-2.8.0.tar.gz](http://archive.apache.org/dist/hadoop/core/hadoop-2.8.0/hadoop-2.8.0.tar.gz))

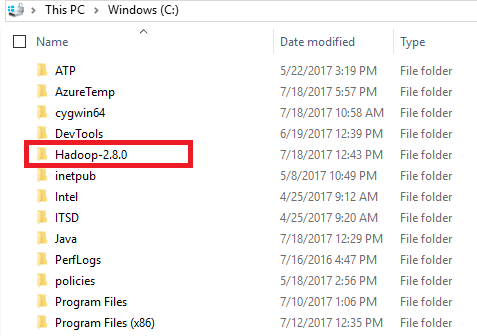
Java JDK 1.8.0.zip (Link: [http://www.oracle.com/technetwork/java/javase/downloads/jdk8-](http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html) [downloads-2133151.html](http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html))

Check either Java 1.8.0 is already installed on your system or not, use "Javac -version" to check.

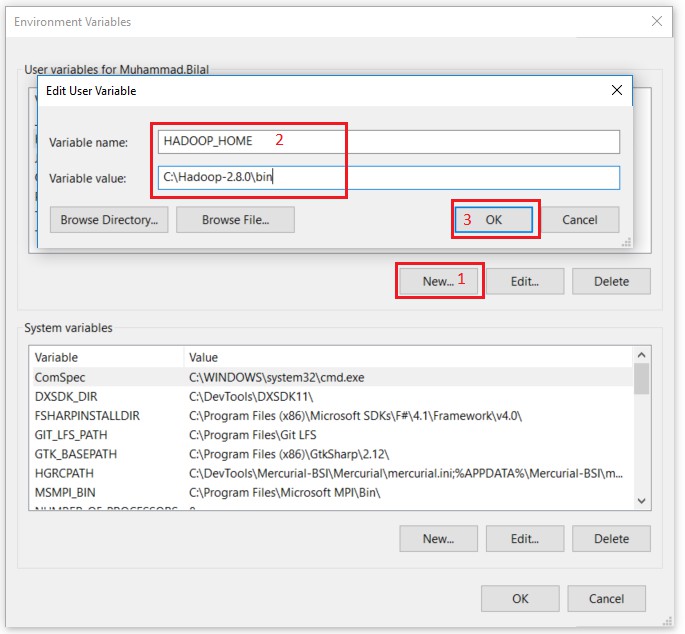


If Java is not installed on your system then first install java under "C:\JAVA"

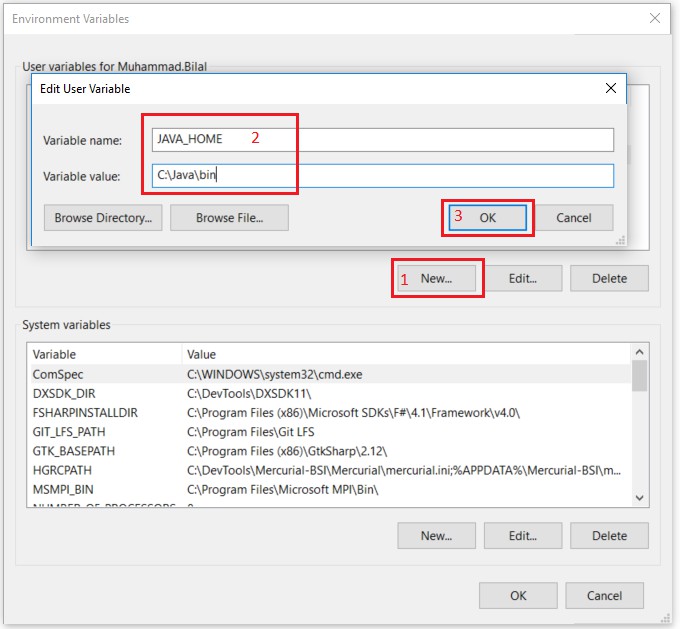
Extract file Hadoop 2.8.0.tar.gz or Hadoop-2.8.0.zip and place under "C:\Hadoop-2.8.0".



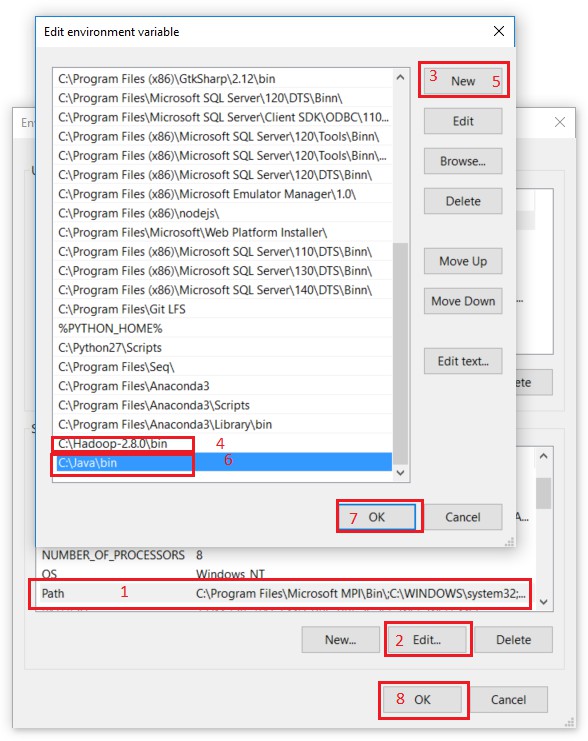
Set the path HADOOP\_HOME Environment variable on windows 10(see Step 1,2,3 and 4 below).



Set the path JAVA\_HOME Environment variable on windows 10(see Step 1,2,3 and 4 below).



Next we set the Hadoop bin directory path and JAVA bin directory path.



CONFIGURATION :

Edit file C:/Hadoop-2.8.0/etc/hadoop/core-site.xml, paste below xml paragraph and save this file.

<configuration>

<property>

<name>fs.defaultFS</name>

<value>hdfs://localhost:9000</value>

</property>

</configuration>

Rename "mapred-site.xml.template" to "mapred-site.xml" and edit this file C:/Hadoop- 2.8.0/etc/hadoop/mapred-site.xml, paste below xml paragraph and save this file.

<configuration>

<property>

<name>mapreduce.framework.name</name>

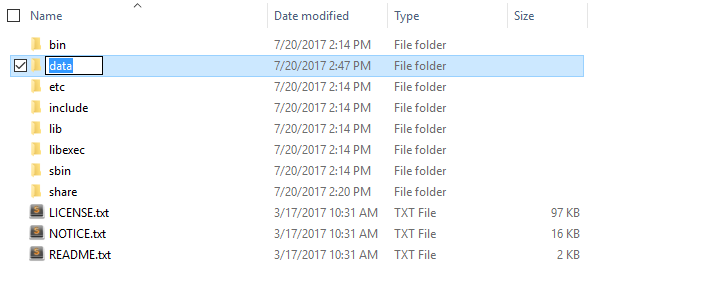
<value>yarn</value>

</property>

</configuration>

Create folder "data" under "C:\Hadoop-2.8.0"

Create folder "datanode" under "C:\Hadoop-2.8.0\data" Create folder "namenode" under "C:\Hadoop-2.8.0\data"



Edit file C:\Hadoop-2.8.0/etc/hadoop/hdfs-site.xml, paste below xml paragraph and save this file.

<configuration>

<property>

<name>dfs.replication</name>

<value>1</value>

</property>

<property>

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

<value>C:\hadoop-2.8.0\data\namenode</value>

</property>

<property>

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

<value>C:\hadoop-2.8.0\data\datanode</value>

</property>

</configuration>

Edit file C:/Hadoop-2.8.0/etc/hadoop/yarn-site.xml, paste below xml paragraph and save this file.

<configuration>

<property>

<name>yarn.nodemanager.aux-services</name>

<value>mapreduce\_shuffle</value>

</property>

<property>

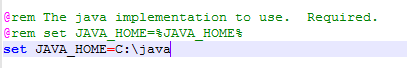
<name>yarn.nodemanager.auxservices.mapreduce.shuffle.class</name>

<value>org.apache.hadoop.mapred.ShuffleHandler</value>

</property>

</configuration>

Edit file C:/Hadoop-2.8.0/etc/hadoop/hadoop-env.cmd by closing the command line "JAVA\_HOME=%JAVA\_HOME%" instead of set JAVA\_HOME=”C:\Java\jdk\bin" (On C:\java this is path to file jdk.18.0)

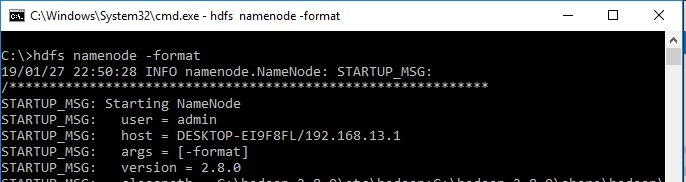


HADOOP CONFIGURATION :

Dowload file Hadoop Configuration.zip (Link: [https://github.com/MuhammadBilalYar/HADOOP-INSTALLATION-ON-WINDOW-](https://github.com/MuhammadBilalYar/HADOOP-INSTALLATION-ON-WINDOW-10/blob/master/Hadoop%20Configuration.zip) [10/blob/master/Hadoop%20Configuration.zip)](https://github.com/MuhammadBilalYar/HADOOP-INSTALLATION-ON-WINDOW-10/blob/master/Hadoop%20Configuration.zip)

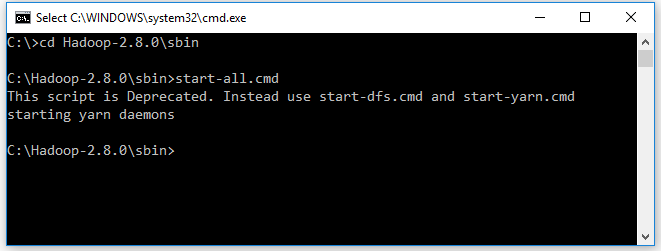
Delete file bin on C:\Hadoop-2.8.0\bin, replaced by file bin on file just download (from Hadoop Configuration.zip).

Open cmd and typing command "hdfs namenode –format" . You will see



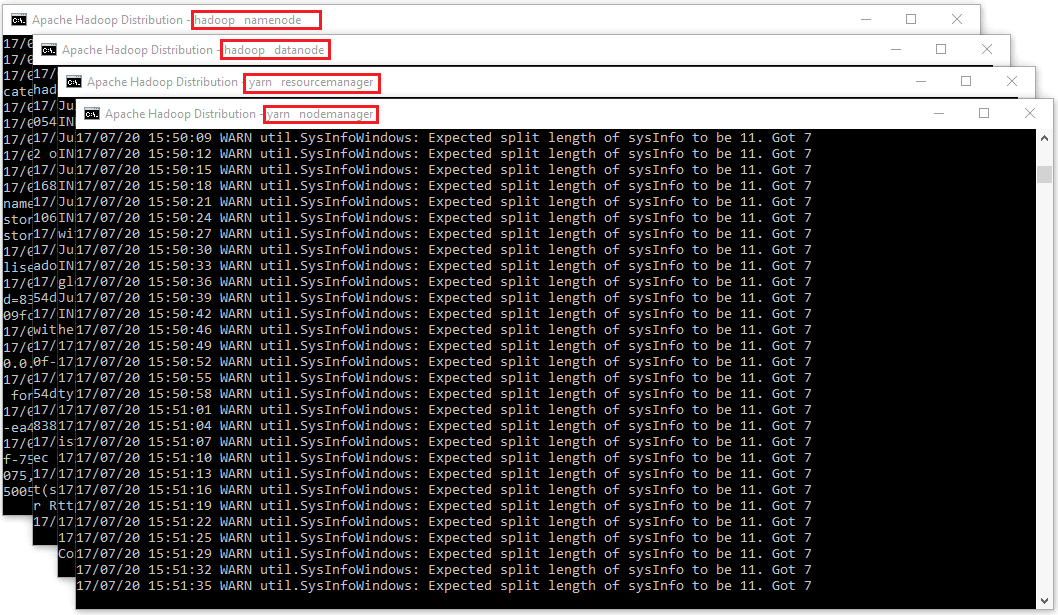
TESTING :

Open cmd and change directory to "C:\Hadoop-2.8.0\sbin" and type "start-all.cmd" to start apache.

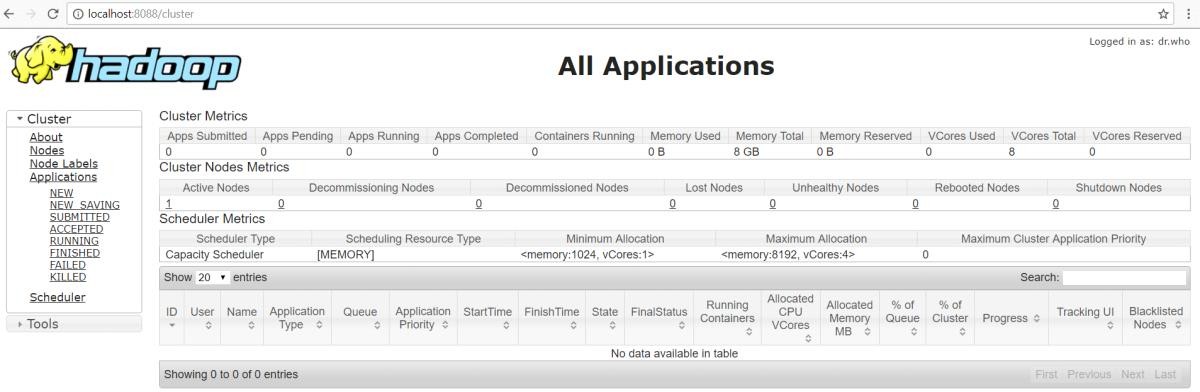


Make sure these apps are running : Hadoop Namenode

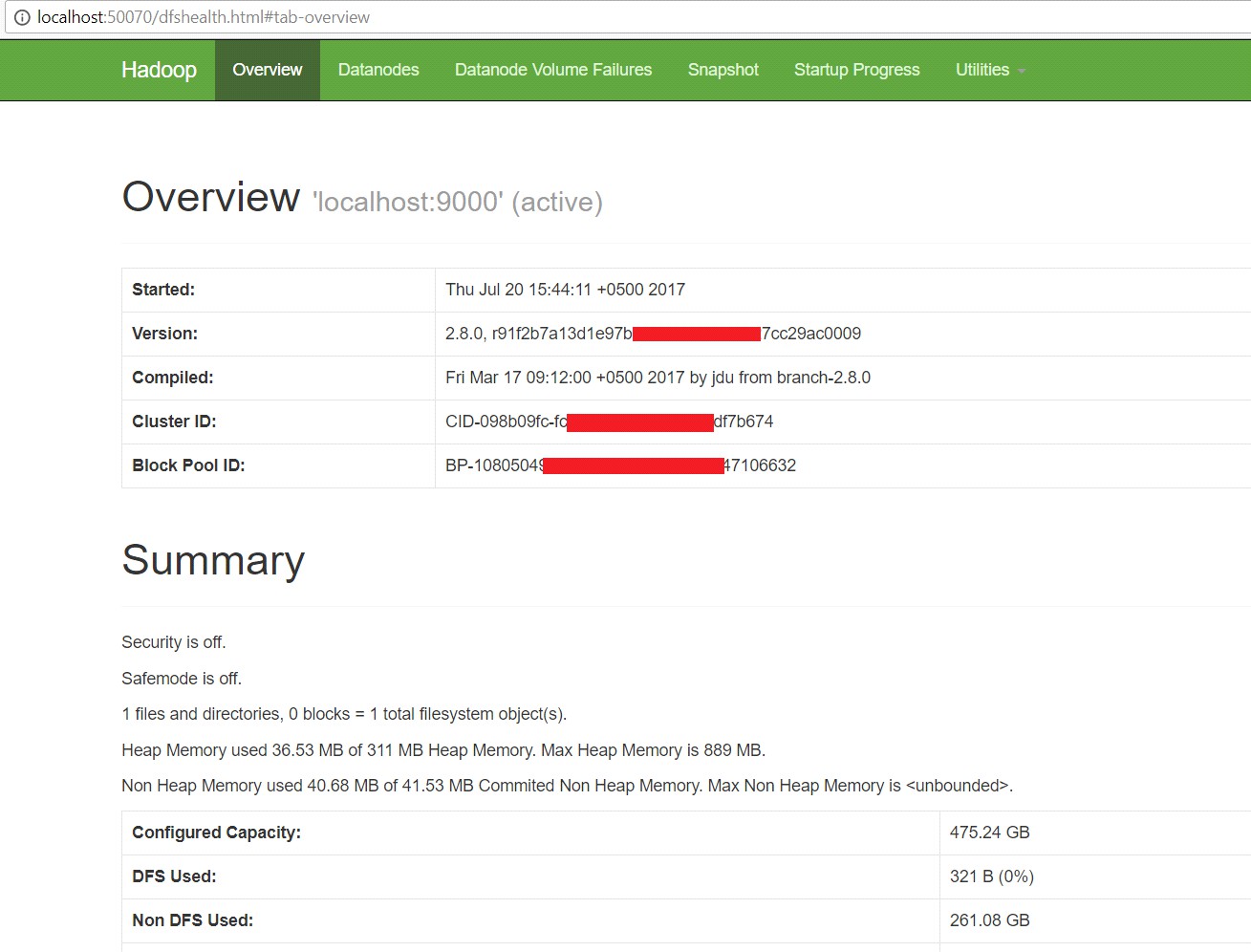
Hadoop datanode YARN Resourc Manager YARN Node Manager



Open: http://localhost:8088



Open: http://localhost:50070



File management tasks in hadoop

In order to perform operations on Hadoop like copy, delete, move etc., following steps can be used:

Basic operations:

1. Create a directory in HDFS at given path(s). Usage:

hadoop fs -mkdir <paths>

1. List the contents of a directory. Usage :

hadoop fs -ls <args>

1. See contents of a file Same as unix cat command:

Usage:

hadoop fs -cat <path[filename]>

1. Copy a file from source to destination

This command allows multiple sources as well in which case the destination must be a directory.

Usage:

hadoop fs -cp <source> <dest>

1. Copy a file from/To Local file system to HDFS copyFromLocal

Usage:

hadoop fs -copyFromLocal <localsrc> URI

Similar to put command, except that the source is restricted to a local file reference. copyToLocal

Usage:

hadoop fs -copyToLocal [-ignorecrc] [-crc] URI <localdst>

Similar to get command, except that the destination is restricted to a local file reference.

1. Move file from source to destination.

Note:- Moving files across filesystem is not permitted. Usage :

hadoop fs -mv <src> <dest>

1. Remove a file or directory in HDFS.

Remove files specified as argument. Deletes directory only when it is empty Usage :

hadoop fs -rm <arg>

Steps for copying file

* 1. Go to Hadoop folder and then to sbin C:\>cd C:\hadoop-2.8.0\sbin
  2. Start namenode and datanode with this command, Two more cmd windows will open C:\hadoop-2.8.0\sbin>start-dfs.cmd
  3. Now start yarn through following command, Two more windows will open, one for yarn resource manager and one for yarn node manager

C:\hadoop-2.8.0\sbin>start-yarn.cmd

* 1. Create a directory named ‘sample’ in the hadoop directory using the following command C:\hadoop-2.8.0\sbin> hdfs dfs -mkdir /sample
  2. To verify if the directory is created C:\hadoop-2.8.0\sbin>hdfs dfs -ls /
  3. Copy text file from D drive to sample

C:\hadoop-2.8.0\sbin>hdfs dfs -copyFromLocal d:\rally.txt /sample

* 1. To verify if the file is copied C:\hadoop-2.8.0\sbin>hdfs dfs -ls /sample

## CONCLUSION :

|  |
| --- |
| Experiment No. 2 |
| Use of Sqoop tool |
| Date of Performance: |
| Date of Submission: |

**AIM**: To install SQOOP and execute basic commands of Hadoop eco system componentSqoop.

## THEORY:

Installation and configuration of SQOOP

1. Download SQOOP from [https://sqoop.apache.org](https://sqoop.apache.org/)
2. Unzip and Install SQOOP

After Downloading the SQOOP, we need to Unzip the sqoop-1.4.7.bin hadoop- 2.6.0.tar.gz file.

1. Create a folder and move the final extracted file in it.
2. Set up the environment variables
   1. Set SQOOP\_HOME
   2. Set up path variable
3. Configure SQOOP

### Basic SQOOP commands:

1. List Table

This command lists the particular table of the [database in MYSQL server.](https://www.educba.com/cheat-sheet-mysql/)

sqoop command list table

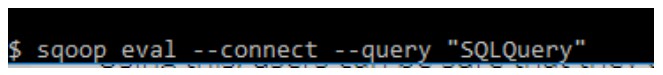
1. Target directory

sqoop command target directoryThis command import table in a specific [directory in HDFS](https://www.educba.com/hdfs-vs-hbase/). -m denotes mapper argument. They have an integer value.

1. sqoop-eval

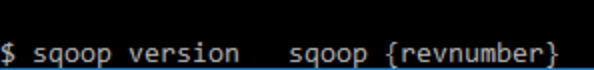
This command runs quickly SQL queries of the respective database.

CSL702: Big Data Analytics Lab



1. sqoop – version

This command displays version of the sqoop.



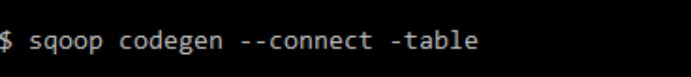
1. sqoop-job

This command allows us to create a job, the parameters that are created can be invoked at any time. They take options like (–create,–delete,–show,–exit).

sqoop command sqoop-job

1. code gen

This Sqoop command [creates java class files](https://www.educba.com/serialization-in-java/) which encapsulate the imported records. All the java files are recreated, and new versions of a class are generated. They generate code to interact with database records. Retrieves a list of all the columns and their datatypes.



1. List Database

This Sqoop command lists have all the available database in the RDBMS server.

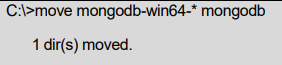
List-database

**CONCLUSION:**

|  |
| --- |
| Experiment No.3 |
| To install and configure MongoDB to execute NoSQL  commands |
| Date of Performance: |
| Date of Submission: |

**AIM**: To install and configure MongoDB/ Cassandra/ HBase/ Hypertable and to execute NoSQL commands.

## THEORY:

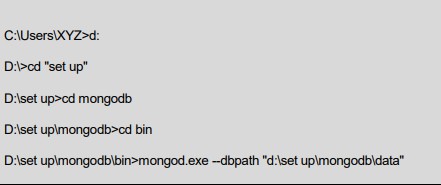
MongoDB can be downloaded from <https://www.mongodb.com/try/download/community2> Now open command prompt and run the following command

MongoDB requires a data folder to store its files. The default location for the MongoDB data directory is c:\data\db. So create the folder using the Command Prompt. Execute the following command sequence.

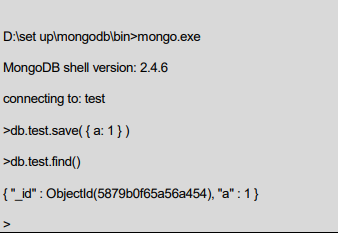


In case mongodb is stored in some other location, navigate to that folder.

In command prompt navigate to the bin directory present into the mongodb installation folder. Suppose the installation folder is D:\set up\mongodb



Now to run the mongodb, open another command prompt and issue the following command:



### The use Command

MongoDB use DATABASE\_NAME is used to create database. The command will create a new database, if it doesn't exist otherwise it will return the existing database

### Syntax:

use DATABASE\_NAME

### The dropDatabase () Method

MongoDB db.dropDatabase () command is used to drop an existing database.

### Syntax:

db.dropDatabase()

### The createCollection() Method

MongoDB db.createCollection(name, options) is used to create collection.

### Syntax:

db.createCollection(name, options)

### Insert Document

To insert data into MongoDB collection, you need to use MongoDB's insert() or save()method

### Syntax

>db.COLLECTION\_NAME.insert(document)

### Example:

>db.post.insert([

{

title: 'MongoDB Overview',

description: 'MongoDB is no sql database', tags: ['mongodb', 'database', 'NoSQL'], likes: 100

},

{

title: 'NoSQL Database',

description: 'NoSQL database doesn't have tables', tags: ['mongodb', 'database', 'NoSQL'],

likes: 20, comments: [

{

user:'user1',

message: 'My first comment', dateCreated: new Date(2022,11,10,2,35), like: 0

}

]

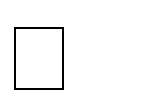
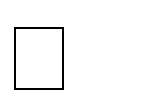
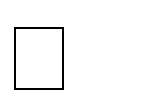
}

])

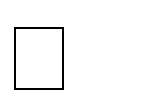
### Creating sample document:

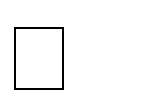
**Example**

Suppose a client needs a database design for his blog website. Website has the following requirements.

Every post has the unique title, description and url. Every post can have one or more tags.

Every post has the name of its publisher and total number of likes.

Every Post have comments given by users along with their name, message, data-time and likes.

On each post there can be zero or more comments.

Document:

{

\_id: POST\_ID

title: TITLE\_OF\_POST,

description: POST\_DESCRIPTION, by: POST\_BY,

url: URL\_OF\_POST,

tags: [TAG1, TAG2, TAG3],

likes: TOTAL\_LIKES, comments: [

{ user:'COMMENT\_BY',

message: TEXT, dateCreated: DATE\_TIME, like: LIKES

},

{ user:'COMMENT\_BY',

message: TEXT, dateCreated: DATE\_TIME, like: LIKES

}

]

}

## CONCLUSION:

|  |
| --- |
| Experiment No.4 |
| Experiment on Hadoop Map-Reduce |
| Date of Performance: |
| Date of Submission: |

**AIM**: -To write a program to implement a word count program using MapReduce.

## THEORY:

WordCount is a simple program which counts the number of occurrences of each word in a given text input data set. WordCount fits very well with the MapReduce programming model making it a great example to understand the Hadoop Map/Reduce programming style. The implementation consists of three main parts:

1. Mapper
2. Reducer
3. Driver

Step-1. Write a Mapper

A Mapper overrides the ―map‖ function from the Class “org.apache.hadoop.mapreduce.Mapper" which provides <key, value> pairs as the input. A Mapper implementation may output <key,value> pairs using the provided Context .

Input value of the WordCount Map task will be a line of text from the input data file and the key would be the line number <line\_number, line\_of\_text> . Map task outputs <word, one> for each word in the line of text.

Pseudo-code

void Map (key, value){ for each word x in value:

output.collect(x,1);

}

Step-2. Write a Reducer

A Reducer collects the intermediate <key,value> output from multiple map tasks and assemble a single result. Here, the WordCount program will sum up the occurrence of each word to pairs as <word, occurrence>.

Pseudo-code

void Reduce (keyword, <list of value>){ for each x in <list of value>:

sum+=x;

final\_output.collect(keyword, sum);

}

Code:

import java.io.IOException; import java.util.StringTokenizer;

import org.apache.hadoop.io.IntWritable; import org.apache.hadoop.io.LongWritable; import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Mapper; import org.apache.hadoop.mapreduce.Reducer; import org.apache.hadoop.conf.Configuration; import org.apache.hadoop.mapreduce.Job;

import org.apache.hadoop.mapreduce.lib.input.TextInputFormat; import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat; import org.apache.hadoop.mapreduce.lib.input.FileInputFormat; import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat; import org.apache.hadoop.fs.Path;

public class WordCount

{

public static class Map extends Mapper<LongWritable,Text,Text,IntWritable> { public void map(LongWritable key, Text value,Context context) throws IOException,InterruptedException{

String line = value.toString();

StringTokenizer tokenizer = new StringTokenizer(line); while (tokenizer.hasMoreTokens()) { value.set(tokenizer.nextToken());

context.write(value, new IntWritable(1));

}

}

}

public static class Reduce extends Reducer<Text,IntWritable,Text,IntWritable> { public void reduce(Text key, Iterable<IntWritable> values,Context context) throws IOException,InterruptedException {

int sum=0; for(IntWritable x: values)

{

sum+=x.get();

}

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

}

}

public static void main(String[] args) throws Exception { Configuration conf= new Configuration();

Job job = new Job(conf,"My Word Count Program"); job.setJarByClass(WordCount.class); job.setMapperClass(Map.class); job.setReducerClass(Reduce.class); job.setOutputKeyClass(Text.class); job.setOutputValueClass(IntWritable.class); job.setInputFormatClass(TextInputFormat.class); job.setOutputFormatClass(TextOutputFormat.class); Path outputPath = new Path(args[1]);

//Configuring the input/output path from the filesystem into the job

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

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

//deleting the output path automatically from hdfs so that we don't have to delete it explicitly

outputPath.getFileSystem(conf).delete(outputPath);

//exiting the job only if the flag value becomes false System.exit(job.waitForCompletion(true) ? 0 : 1);

}

}

## CONCLUSION:

|  |
| --- |
| Experiment No.5 |
| Create HIVE Database and Descriptive analytics-basic  statistics. |
| Date of Performance: |
| Date of Submission: |

**Aim:** Create HIVE Database and Descriptive analytics-basic statistics.

### Theory:

Hive is a database technology that can define databases and tables to analyze structured data. The theme for structured data analysis is to store the data in a tabular manner, and pass queries to analyze it. This chapter explains how to create Hive database. Hive contains a default database named default.

### Create Database Statement

Create Database is a statement used to create a database in Hive. A database in Hive is a namespace or a collection of tables. The syntax for this statement is as follows:

CREATE DATABASE|SCHEMA [IF NOT EXISTS] <database name>

Here, IF NOT EXISTS is an optional clause, which notifies the user that a database with the same name already exists. We can use SCHEMA in place of DATABASE in this command. The following query is executed to create a database named userdb:

hive> CREATE DATABASE [IF NOT EXISTS] userdb;

hive> CREATE SCHEMA userdb;

The following query is used to verify a databases list:

hive> SHOW DATABASES;

default userdb **Program:**

The JDBC program to create a database is given below. import java.sql.SQLException;

import java.sql.Connection; import java.sql.ResultSet;

import java.sql.Statement; import java.sql.DriverManager;

public class HiveCreateDb {

private static String driverName = "org.apache.hadoop.hive.jdbc.HiveDriver";

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

// Register driver and create driver instance

Class.forName(driverName);

// get connection

Connection con = DriverManager.getConnection("jdbc:hive://localhost:10000/default", "", "");

Statement stmt = con.createStatement();

stmt.executeQuery("CREATE DATABASE userdb"); System.out.println(“Database userdb created successfully.”);

con.close();

}

}

Output:

Database userdb created successfully. CONCLUSION:

|  |
| --- |
| Experiment No.6 |
| Data Stream Algorithms:  Implement Flajolet Martin algorithm using any programming  language |
| Date of Performance: |
| Date of Submission: |

**Aim:** Data Stream Algorithms:

Implement Flajolet Martin algorithm using any programming language

### Theory:

Flajolet-Martin algorithm approximates the number of unique objects in a stream or a database in one pass. If the stream contains n elements with m of them unique, this algorithm runs in O(n) time and needs O(log(m)) memory.

Algorithm:

1. Create a bit vector (bit array) of sufficient length L, such that 2L>n, the number of elements in the stream. Usually a 64-bit vector is sufficient since 264 is quite large for most purposes.
2. The i-th bit in this vector/array represents whether we have seen a hash function value whose binary representation ends in 0i. So initialize each bit to 0.
3. The i-th bit in this vector/array represents whether we have seen a hash function value whose binary representation ends in 0i. So initialize each bit to 0.
4. The i-th bit in this vector/array represents whether we have seen a hash function value whose binary representation ends in 0i. So initialize each bit to 0.
5. Once input is exhausted, get the index of the first 0 in the bit array (call this R). By the way, this is just the number of consecutive 1s (i.e. we have seen 0,00,...,0R−1 as the output of the hash function) plus one.
6. Calculate the number of unique words as 2R/ϕ, where ϕ is 0.77351. A proof for this can be found in the original paper listed in the reference section.
7. The standard deviation of R is a constant: σ(R)=1.12. (In other words, R can be off by about 1 for 1 - 0.68 = 32% of the observations, off by 2 for about 1 - 0.95 = 5% of the observations, off by 3 for 1 - 0.997 = 0.3% of the observations using the Empirical rule of statistics). This implies that our count can be off by a factor of 2 for 32% of the observations, off by a factory of 4 for 5% of the observations, off by a factor of 8 for 0.3% of the observations and so on.

## CODE:

stream=[1,2,3,4,5,6,4,2,5,9,1,6,3,7,1,2,2,4,2,1]

print('Using Flajolet Martin Algorithm:') maxnum=0

for i in range(0,len(stream)):

val= bin((1\*stream[i] + 6) % 32)[2:] sum=0

for j in range(len(val)-1,0,-1): if val[j]=='0':

sum+=1 else:

break

if sum>maxnum: maxnum=sum

print('distict elements', 2\*\*maxnum)

Output:

Using Flajolet Martin Algorithm: distict elements 8

## CONCLUSION:

|  |
| --- |
| Experiment No.7 |
| Social Network Analysis using R (for example: Community Detection Algorithm) |
| Date of Performance: |
| Date of Submission: |

**Aim:** Social Network Analysis using R (for example: Community Detection Algorithm)

### Theory:

Online social platforms have enabled people around the world to interact with each other and build relationships with others they share common interests with. This can be observed in real life — naturally, we tend to develop and maintain relationships with others that are similar to us. People with similar interests tend to gravitate towards each other and become associated in communities — clusters or groups of people that share similar traits with each other. Since people tend to cluster with others similar to them, we can use community detection to identify users with a high number of degrees (connections) and see how far their reach can travel in the network.

User Data Extraction — Since we are only interested in user data, we will only extract the following variables:

User\_id — Yelp user ID; this is needed to make nodes and edges Name — user’s first name

Review count — the number of reviews user has written Yelping since — date user joined Yelp

Friends — a list containing all of the user’s friends by user\_id Fans — number of fans user has

Elite — number of years the user has Elite status

Average stars — user’s average rating of all reviews written

## CODE:

#remove users with no friends

sample <- subset(user\_df, friends != "None")

#make a subset; we only need to retain data of users with some social activity sub <- subset(sample, year == 2005 & review\_count >= 2 & no\_of\_friends >= 2) #make links (nodes and edges)

sample\_friends <- sub %>% select(user\_id, friends) sample\_users <- strsplit(sample\_friends$friends, split = ",")

sample\_dat <- data.frame(user\_id = rep(sample\_friends$user\_id, sapply(sample\_users, length)), friends = unlist(sample\_users))

#network is still too big, take a random sample of 100k nodes samp\_net <- sample\_n(sample\_dat, 100000)

#make network

network <- graph.data.frame(samp\_net) network\_s <- simplify(network) net\_deg <- degree(network\_s)

all\_degree <- degree(network, mode = 'all') #graph user with max degrees

sub\_all <- subcomponent(network\_s, which(all\_degree == max(all\_degree)), 'all') g\_sub <- induced\_subgraph(network\_s, sub\_all)

#communities

graph.com <- fastgreedy.community(as.undirected(g\_sub))

V(g\_sub)$color <- graph.com$membership + 1 #create pdf graph for high resolution (try zooming in!) pdf("communities2005.pdf", 10,10)

plot(g\_sub,

vertex.color = V(g\_sub)$color, vertex.size = 1,

vertex.label = NA,

vertex.frame.color = adjustcolor("#41424c", alpha.f = 0.25), edge.arrow.size = 0.1,

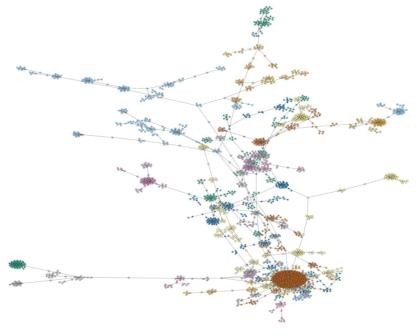
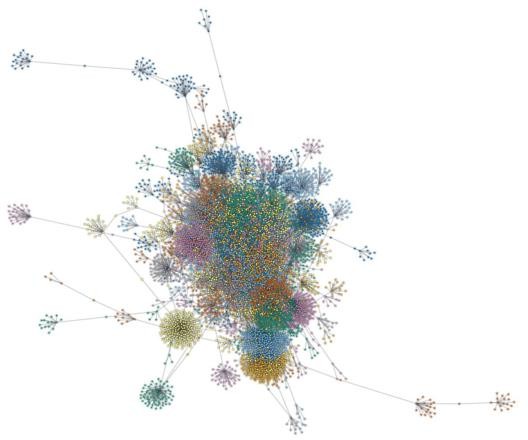
edge.color = adjustcolor("#41424c", alpha.f = 0.20), edge.width = 1.5,

edge.arrow.mode=0, layout=layout\_with\_lgl, asp = 0.9,

dpi=300

)

dev.off()



## CONCLUSION:

|  |
| --- |
| Experiment No.8 |
| Data Visualization using Hive/PIG/R/Tableau/. |
| Date of Performance: |
| Date of Submission: |

**Aim:** Data Visualization using Hive/PIG/R/Tableau/.

### Theory:

**Data visualization** is the technique used to deliver insights in data using visual cues such as graphs, charts, maps, and many others. This is useful as it helps in intuitive and easy understanding of the large quantities of data and thereby make better decisions regarding it. Data Visualization in R Programming Language

The popular data visualization tools that are available are Tableau, Plotly, R, Google Charts, Infogram, and Kibana. The various data visualization platforms have different capabilities, functionality, and use cases. They also require a different skill set . This article discusses the use of R for data visualization.

R is a language that is designed for statistical computing, graphical data analysis, and scientific research. It is usually preferred for data visualization as it offers flexibility and minimum required coding through its packages.

Consider the following *airquality* data set for visualization in R:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Ozone | Solar R. | Wind | Temp | Month | Day |
| 41 | 190 | 7.4 | 67 | 5 | 1 |
| 36 | 118 | 8.0 | 72 | 5 | 2 |
| 12 | 149 | 12.6 | 74 | 5 | 3 |
| 18 | 313 | 11.5 | 62 | 5 | 4 |
| NA | NA | 14.3 | 56 | 5 | 5 |
| 28 | NA | 14.9 | 66 | 5 | 6 |

### Bar Plot

There are two types of bar plots- horizontal and vertical which represent data points as horizontal or vertical bars of certain lengths proportional to the value of the data item. They are generally used for continuous and categorical variable plotting. By setting the horiz parameter to true and false, we can get horizontal and vertical bar plots respectively.

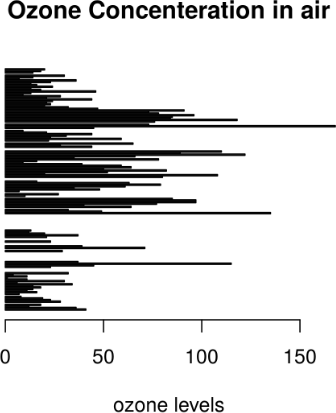
Example 1:

# Horizontal Bar Plot for

# Ozone concentration in air barplot(airquality$Ozone,

main = 'Ozone Concenteration in air', xlab = 'ozone levels', horiz = TRUE)

Output:

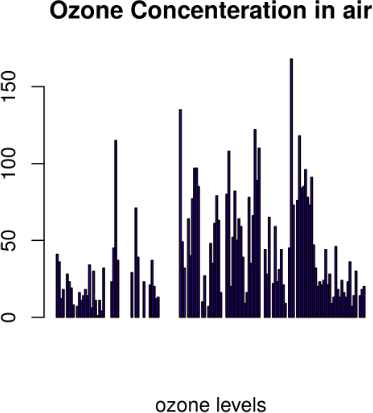


Example 2:

# Vertical Bar Plot for

# Ozone concentration in air

barplot(airquality$Ozone, main = 'Ozone Concenteration in air', xlab = 'ozone levels', col ='blue', horiz = FALSE)



1. Histogram

A histogram is like a bar chart as it uses bars of varying height to represent data distribution. However, in a histogram values are grouped into consecutive intervals called bins. In a Histogram, continuous values are grouped and displayed in these bins whose size can be varied.

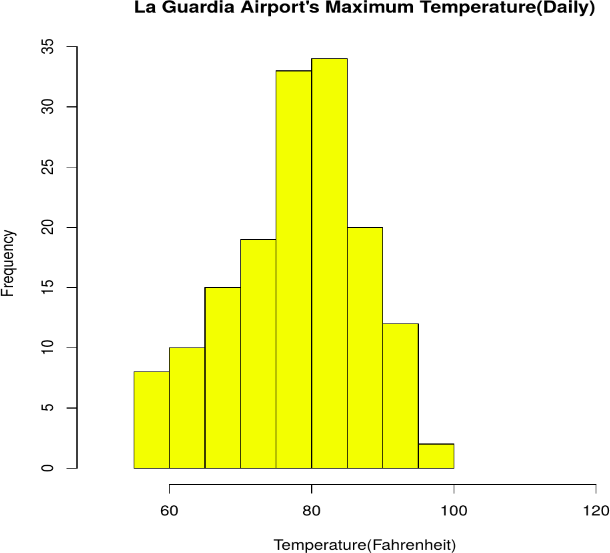
Example:

# Histogram for Maximum Daily Temperature data(airquality)

hist(airquality$Temp, main ="La Guardia Airport's\ Maximum Temperature(Daily)",

xlab ="Temperature(Fahrenheit)", xlim = c(50, 125), col ="yellow", freq = TRUE)

Output:



1. Box Plot

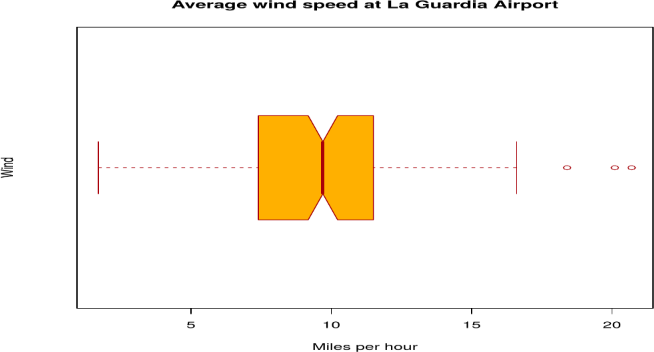
The statistical summary of the given data is presented graphically using a boxplot. A boxplot depicts information like the minimum and maximum data point, the median value, first and third quartile, and interquartile range.

Example:

# Box plot for average wind speed data(airquality)

boxplot(airquality$Wind, main = "Average wind speed\ at La Guardia Airport",

xlab = "Miles per hour", ylab = "Wind", col = "orange", border = "brown", horizontal = TRUE, notch = TRUE)



1. Scatter Plot

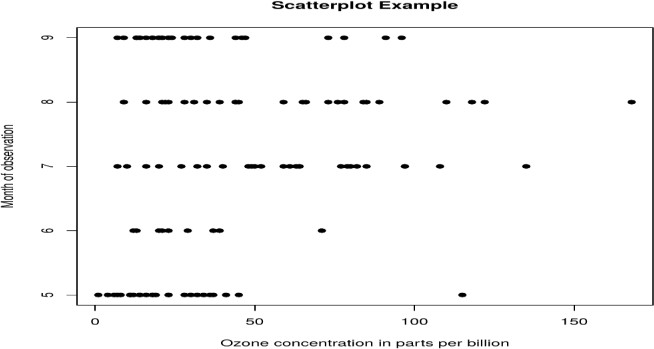
A scatter plot is composed of many points on a Cartesian plane. Each point denotes the value taken by two parameters and helps us easily identify the relationship between them.

Example:.

# Scatter plot for Ozone Concentration per month data(airquality)

plot(airquality$Ozone, airquality$Month, main ="Scatterplot Example",

xlab ="Ozone Concentration in parts per billion", ylab =" Month of observation ", pch = 19)



### Heat Map

Heatmap is defined as a graphical representation of data using colors to visualize the value of the matrix. heatmap() function is used to plot heatmap.

Syntax: heatmap(data)

Parameters: data: It represent matrix data, such as values of rows and columns Return: This function draws a heatmap.

# Set seed for reproducibility # set.seed(110)

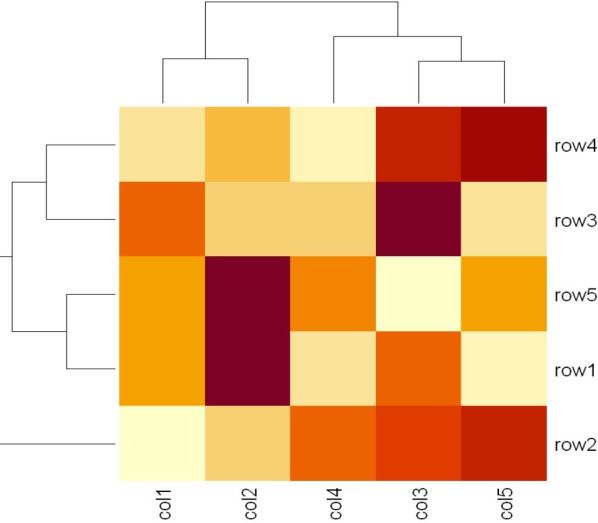
# Create example data

data <- matrix(rnorm(50, 0, 5), nrow = 5, ncol = 5)

# Column names

colnames(data) <- paste0("col", 1:5) rownames(data) <- paste0("row", 1:5) # Draw a heatmap

heatmap(data)



### Map visualization in R

Here we are using maps package to visualize and display geographical maps using an R programming language.

# Read dataset and convert it into # Dataframe

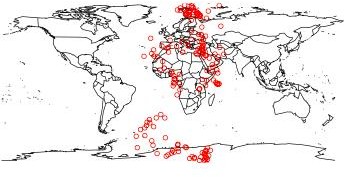
data <- read.csv("worldcities.csv") df <- data.frame(data)

# Load the required libraries library(maps)

map(database = "world")

# marking points on map

points(x = df$lat[1:500], y = df$lng[1:500], col = "Red")install.packages("maps")



1. 3D Graphs in R

Here we will use preps() function, This function is used to create 3D surfaces in perspective view. This function will draw perspective plots of a surface over the x–y plane.

Syntax: persp(x, y, z)

Parameter: This function accepts different parameters i.e. x, y and z where x and y are vectors defining the location along x- and y-axis. z-axis will be the height of the surface in the matrix z.

Return Value: persp() returns the viewing transformation matrix for projecting 3D coordinates (x, y, z) into the 2D plane using homogeneous 4D coordinates (x, y, z, t).

# Adding Titles and Labeling Axes to Plot cone <- function(x, y){

sqrt(x ^ 2 + y ^ 2)

}

# prepare variables.

x <- y <- seq(-1, 1, length = 30) z <- outer(x, y, cone)

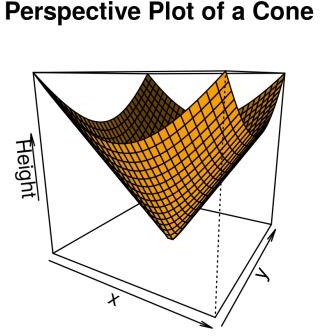
# plot the 3D surface

# Adding Titles and Labeling Axes to Plot persp(x, y, z,

main="Perspective Plot of a Cone", zlab = "Height",

theta = 30, phi = 15,

col = "orange", shade = 0.4)



**CONCLUSION:**