SRI KRISHNA COLLEGE OF TECHNOLOGY

(An Autonomous Institution)

Affiliated to Anna University | Approved by AICTE

Accredited by NAAC with 'A' Grade

KOVAIPUDUR, COIMBATORE – 641 042.

DEPARTMENT OF CSE (IoT)

22CSE05 – BIG DATA ANALYTICS

(REGULATION 2022)

LAB MANUAL

Laboratory In-charge

Ms.SUGITHA.A

Assistant Professor

PREFACE

In the ever-evolving landscape of technology, data has emerged as one of the most valuable resources of the 21st century. The term "Big Data" encapsulates the enormous volumes of structured and unstructured data generated at unprecedented rates, coming from various sources such as social media, sensors, transactions, and more. The real challenge lies not only in capturing and storing this data but in analyzing it to extract meaningful insights that can drive decision-making and innovation.

The field of Big Data Analytics is at the forefront of this revolution, offering powerful tools and techniques to process, analyze, and visualize large datasets. This lab manual is designed to guide students through the practical aspects of Big Data Analytics, providing hands-on experience with industry-standard tools and methodologies.

The primary goal of this lab is to equip students with the skills necessary to handle real-world data analytics challenges. Through a series of structured exercises and projects, students will learn Understand the Fundamentals, Utilize Analytical Tools, Apply Machine Learning Techniques, Implement Data Visualization, Develop Real-world Applications.

The exercises in this lab manual are designed to be both challenging and rewarding, encouraging students to think critically and creatively. By the end of this course, students will not only be proficient in the technical aspects of Big Data Analytics but also be prepared to leverage these skills in their future careers.

FACULTY OF COMPUTER SCIENCE AND ENGINEERING

SRI KRISHNA COLLEGE OF TECHNOLOGY

COIMBATORE – 641 042

Prepared by	**Verified by**	**Approved by**
Dr. E. Praveen Kumar	Dr. Suma Sira Jacob	Dr. T. Senthilnathan
Assistant Professor	Associate Professor	Dean- SoC
CSE	PC/ CSE (AI & ML, CYS, IoT)	

PROFILE OF THE INSTITUTION

Nestled at the foothills of the Western Ghats, located in a sprawling 52-acre campus in Kovaipudur, Coimbatore, Sri Krishna College of Technology (SKCT) is a vibrant institute of higher education established in 1985 promoted by Sri Krishna Institutions. An extraordinary freedom of opportunity—to explore, to collaborate and to challenge oneself is the hallmark of the Institute. Being an autonomous institute, affiliated to Anna University, Chennai, and approved by AICTE, New Delhi, SKCT lays strong emphasis on collaborative research and stands apart from other institutes by its participatory work culture, student care Programmes and high industry interaction.

In a span of 38 years, it has emerged as one of the premier engineering colleges for learning, discovery and innovation due to the dynamic leadership of the Chairperson and Managing Trustee Smt. S. Malarvizhi. Being an acclaimed educationalist, she continues to contribute profusely for the glory and happiness of advancing generations. The college is accredited with A Grade by NAAC and eligible undergraduate programs are accredited by the National Board of Accreditation (NBA), New Delhi. The college offers 11 undergraduate Programmes, 6 Postgraduate Programmes and 5 Doctorial Programmes in Engineering, Technology, and Management Studies.

VISION:

Sri Krishna College of Technology aspires to be recognized as one of the pioneers in imparting world class technical education through technology enabled innovative teaching learning processes with a focus on research activities to cater, to the societal needs.

MISSION:

To be recognized as centre of excellence in science, engineering and technology through effective teaching and learning processes by providing a conducive learning environment.

To foster research and development with creative and entrepreneurial skills by means of innovative applications of technology. Accomplish expectations of the society and industry by nurturing the students to be competent professionals with integrity.

COURSES OFFERED

UNDER GRADUATE PROGRAMMES (Four Years B.E / B.Tech)

- B.E - Civil Engineering
- B.E - Computer Science and Engineering
- B.E - Computer Science and Engineering (Cyber Security)
- B.E - Computer Science and Engineering (Internet of Things)
- B.E - Computer Science and Engineering (Artificial Intelligence and Machine Learning)
- B.E - Electronics and Communication Engineering
- B.E - Electrical and Electronics Engineering
- B.E - Instrumentation and Control Engineering
- B.E - Mechanical Engineering
- B.Tech - Artificial Intelligence and Data Science
- B.Tech - Information Technology

POST GRADUATE PROGRAMMES (Two Years)

- Master of Business Administration
- M.E - Applied Electronics
- M.E – Computer Science Engineering
- M.E –Engineering Design
- M.E - Power System Engineering
- M.E - Structural Engineering

DOCTORAL PROGRAMMES (Ph.D.)

- Civil Engineering
- Computer Science and Engineering
- Electronics and Communication Engineering
- Electrical and Electronics Engineering
- Mechanical Engineering

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING (INTERNET OF THINGS)

The Department of Computer Science and Engineering (Internet of Things) at the esteemed institution, SKCT, which was established in 2022. At our institution, we are committed to providing a 4-year Bachelor of Engineering (B.E.) degree with a specific focus on Internet of Things. The eminent team of faculty is dedicated in delivering a high-quality education to equip students with the necessary skills to navigate the dynamic and ever-changing domains of AI & ML, Cyber Security, and IoT. The program has been strategically developed to cultivate creativity, critical thinking, and problem-solving aptitudes, equipping our graduates with the necessary capabilities to contribute sustainable solutions to industrial and society problems.

VISION:

The department of CSE fosters a conducive ambience to meet the global standards by equipping the students with modern techniques in the area of Computer Science and relevant research to address the societal needs.

MISSION:

- To provide positive working environment that would help the students perform to their highest abilities in various fields of computer science.
- To enable students and faculty with the best of technologies and knowledge emerging in the domain of Computer Science and Engineering.
- To establish nationally and internationally recognized research centers and expose the students to broad research experience.

PROGRAM EDUCTIONAL OBJECTIVES:

PEO1: Apply the acquired engineering knowledge to solve economic, social, ethical, and environmental issues related to Internet of Things.

PEO2: Adapt the emerging Information and Communication Technologies to innovate and to cater the Industrial and Societal needs.

PEO3: Contribute Internet of Things expertise to research and development and create novel products that benefit society.

PROGRAM OUTCOMES:

PO 1: Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

PO 2: 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.

PO 3: 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.

PO 4: 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.

PO 5: 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.

PO 6: 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.

PO 7: 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.

PO 8: Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO 9: Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO 10: 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.

PO 11: 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.

PO 12: 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.

PROGRAM SPECIFIC OUTCOMES

PSO1: Acquire and apply standard Software Engineering practices and strategies in IoT project development to deliver a quality product for industry success.

PSO2: Analyze connected sensors, devices and equipment for transferring data over a network.

22CSE05	BIG DATA ANALYTICS	2/0/2/3
Nature of the Course: Theory With Practical - (External Mark: 50/ Internal Mark: 50) End Semester Mark Splitup: **(End Semester Theory Maximum Marks- 100(Weightage- 25%), End Semester Practical Maximum Marks- 100(Weightage- 25%))**		
Pre-requisite(s):Nil		
Course Objectives:		
1	To understand the need of Big Data, challenges, and different analytical architectures	
2	To install and understanding of Hadoop Architecture and its ecosystems.	
3	To Process of Big Data with Advanced architectures like Spark.	
4	Describe graphs and streaming data in Spark	
Course Outcomes: **Upon completion of the course, students shall have ability to**		
CO1	Explore the evolution of Big data with its characteristics	U
CO2	Demonstrate NOSQL distributed database storage and processing	AP
CO3	Make use of appropriate components for processing, scheduling and knowledge extraction from large volumes in distributed Hadoop Ecosystem.	AP
CO4	Develop a Map Reduce application for optimizing the jobs.	AP
CO5	Explore the importance of big data framework HIVE and its built-in functions, datatypes and services like DDL.	AP
CO6	Make use of Spark MLib in various applications for big data processing	AP

Course Content:	
Module 1: INTRODUCTION TO BIG DATA AND HADOOP FRAMEWORK	**10 Hrs**
Introduction to Big Data: Types of Digital Data-Characteristics of Data – Evolution of Big Data - Definition of Big Data – Challenges with Big Data – 3Vs of Big Data – Non-Definitional traits of Big Data – Business Intelligence vs. Big Data - Data warehouse and Hadoop environment – Coexistence. Big Data Analytics: Classification of analytics – Data Science - Terminologies in Big Data – CAP Theorem – BASE Concept. NoSQL: Types of Databases – Advantages – NewSQL - SQL vs. NOSQL vsNewSQL. Introduction to Hadoop: Features – Advantages - Versions – Overview of Hadoop Eco systems – Hadoop distributions – Hadoop vs. SQL – RDBMS vs. Hadoop- Hadoop Components – Architecture - HDFS – Map Reduce: Mapper – Reducer - Combiner -Partitioner – Searching – Sorting – Compression. Hadoop 2 (YARN): Architecture – Interacting with Hadoop Eco systems.	
Module 2: NO SQL DATABASES	**10 Hrs**
No SQL databases: Mongo DB: Introduction – Features – Data types – Mongo DB Query language – CRUD operations – Arrays – Functions: Count – Sort – Limit – Skip – Aggregate – Map Reduce. Cursors – Indexes – Mongo Import – Mongo Export. Cassandra: Introduction – Features – Data types – CQLSH – Key spaces – CRUD operations – Collections – Counter – TTL – Alter commands – Import and Export – Querying System tables. Memory Model, Shared Memory Matrix Multiplication, Additional CUDA API Features	
Module 3: HADOOP ECO SYSTEMS	**10 Hrs**
Hadoop Eco systems: Hive – Architecture – data type – File format – HQL – SerDe – User defined functions – Pig: Features – Anatomy – Pig on Hadoop - Pig Latin overview – Data types – Running pig – Execution modes of Pig – HDFS commands – Relational operators – Eval Functions – Complex data type – Piggy Bank – User defined Functions – Parameter substitution – Diagnostic operator. Jasper Report: Introduction – Connecting to Mongo DB – Connecting to Cassandra – Introduction of Big data Machine learning with Spark: Introduction to Spark MLib, Linear Regression - Clustering - Collaborative filtering - Association rule mining – Decision tree using Spark. Introduction to Graph - Introduction to Spark GraphX, Introduction to Streams Concepts - Stream Data Model and Architecture Introduction to Spark Streaming - Kafka -Streaming Ecosystem.	
Total Hours(L):	**30**

Lab Components:

S.No.	List of Experiments	CO Mapping	RBT
1	Downloading and installing Hadoop; Understanding different Hadoop modes. Startup scripts, Configuration files.	CO2	AP

2	Hadoop Implementation of file management tasks, such as Adding files and directories, retrieving files and Deleting files	CO2	AP
3	Implement of Matrix Multiplication with Hadoop Map Reduce	CO3	AP
4	Run a basic Word Count Map Reduce program to understand Map Reduce Paradigm.	CO5	AP
5	Installation of Hive along with practice examples.	CO6	AP
6	Installation of HBase, Installing thrift along with Practice examples	CO6	AP
7	Practice importing and exporting data from various databases.	CO6	AP
8	Implement a program using Hive indexes.	CO6	AP
9	Implement a program using Hive views	CO6	AP
10	Implement a program using Hive external table by accessing the external file created by Pig or any other tool.	CO6	AP
11	Program using Hive scripts and aggregate functions	CO6	AP
		Total Hours(P):	**30**

Text Books:	
1	Seema Acharya, SubhashiniChellappan, "Big Data and Analytics", Wiley Publication, 2015.
2	TomWhite,"Hadoop:TheDefinitiveGuide",O'Reilly,4thEdition,2015
Reference Books:	
1	Judith Hurwitz, Alan Nugent, Dr. Fern Halper, Marcia Kaufman, "Big Data for Dummies", John Wiley & Sons, Inc., 2013.
2	Mohammed Guller, Big Data Analytics with Spark, Apress,2015
3	Kyle Banker, "Mongo DB in Action", Manning Publications Company, 2012.
4	Russell Bradberry, Eric Blow, "Practical Cassandra A developers Approach ", Pearson Education, 2014.
Web References:	
1	https://www.amrita.edu/course/big-data-analytics
2	https://www.sas.com/en_in/insights/analytics/big-data-analytics.html
Online References:	
1	https://www.tableau.com/learn/articles/big-data-analytics
2	https://nptel.ac.in/courses/106104189

CO	PO												PSO	
	1	2	3	4	5	6	7	8	9	10	11	12	1	2
CO1	3	3	2	2	3	-	-	-	-	2	-	2	2	2
CO2	3	3	3	3	3	-	-	-	-	2	-	2	2	2
CO3	3	3	2	2	3	-	-	-	-	2	-	2	2	2
CO4	3	3	3	3	3	-	-	-	-	2	-	2	2	2
CO5	3	3	2	2	3	-	-	-	-	2	-	2	2	2
CO6	3	3	2	3	3	-	-	-	-	2	-	2	2	2

CO - Course Outcome PO - Programme Outcome PSO - Programme Specific Outcomes	1 - Reasonably Agreed 2 - Moderately Agreed 3 - Strongly Agreed

TABLE OF CONTENT

RUBRIC ASSESSMENT FOR : 22CSE05 – BIG DATA ANALYTICS

Items	Excellent	Good	Satisfactory	Needs Improvement
OBJECTIVE & ALGORITHM (20 MARKS)				
	Objective and Algorithm are highly / maximally efficient and effective, demonstrating strong understanding	Objective and Algorithm are efficient andeffective, demonstrating moderate understanding	Objective and Algorithm are somewhat efficient and effective, demonstrating	Objective and Algorithm Inefficient and/or ineffective, demonstrating limited
PROGRAM WITH SYNTAX AND STRUCTURE (30 Marks)	27-30	21-26	15-20	0-14
	The program design uses appropriate Syntax and Structures. The program overall design is	The program design generally usesa ppropriate structures. Program elements	Not all of the selected structures are appropriate. Some of the program elements	Few of the selectedstructures
COMPILATION AND DEBUGGING (30 MARKS)	27-30	21-26	15-20	0-14
	Program compiles and contains no evidence of misunderstanding or misinterpreting the syntax of the language. Program produces correct answers or appropriate results for all inputs tested.	Program compiles andis free from major syntactic misunderstandings, but may contain non-standard usage or superfluous elements.Program producescorrect answers orappropriate results formost inputs.	Program compiles, but contains errors that signal misunderstanding of syntax. Program approaches orrect answers or appropriate results for most inputs, but can contain miscalculationsin some cases.	Program does not compile orcontains typographical errorsleading to undefined names. Program does not producecorrect answers or appropriate resultsfor most inputs.
DOCUMENT ATION (10 MARKS)	9-10	7-8	5-6	0-4
	Clearly and effectively documented including descriptions of all class variables.Specific purpose noted for each function, control structure, input requirements, and output results.	Clearly documented including descriptions of all class variables. Specific purpose is noted for each function and control structure.	Basic documentation has been completed including descriptions of all class variables. Purpose is noted for each function.	Very limited or no documentation included. Documentation does not help the reader understand the code.
VIVA (10 MARKS)	9-10	7-8	5-6	0-4
	Masterfully defends by providing clear and insightful answers to questions	Competently defends by providing very helpful answers	Answers questions, but often with little insight	Very less answers /Does not answer

EXPT.NO.1	Downloading and installing Hadoop; Understanding different Hadoop modes.Startup scripts, Configuration files.

AIM:

To Downloading and installing Hadoop; Understanding different Hadoop modes. Startupscripts, Configuration files.

PROCEDURE:

Prerequisites to Install Hadoop on Ubuntu

Hardware requirement- The machine must have 4GB RAM and minimum 60 GB hard diskfor better performance.

Check java version- It is recommended to install Oracle Java 8.The user can check the version of java with below command.

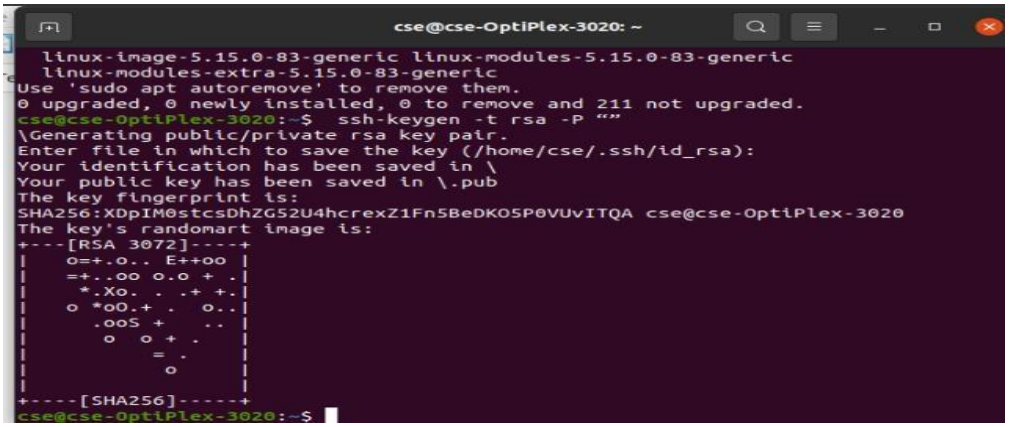
$ java –version

STEP 1: Setup passwordless ssh

a) Install Open SSH Server and Open SSH Client

We will now setup the passwordless ssh client with the following command.

1.$sudo apt-get install openssh-server openssh-client

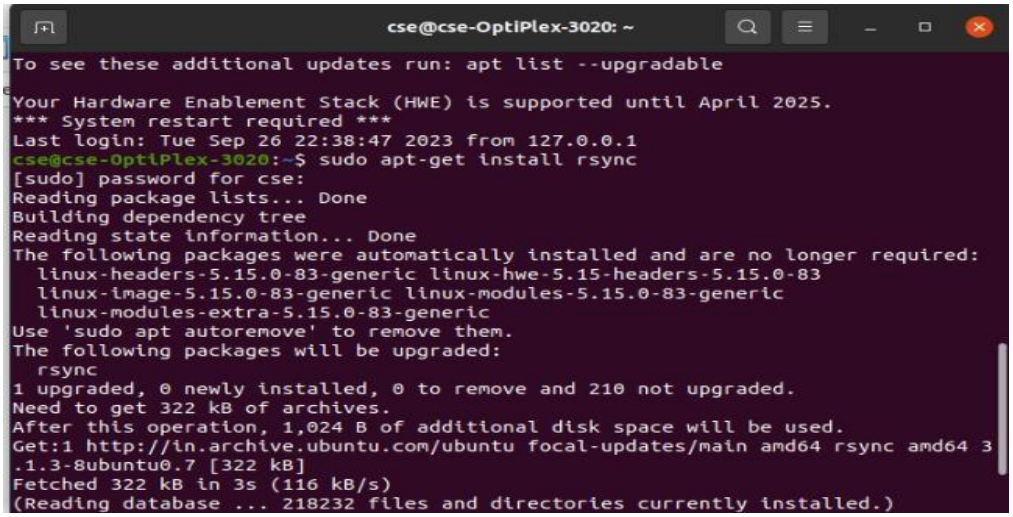


b) Generate Public & Private Key Pairs

2. ssh-keygen -t rsa -P “”

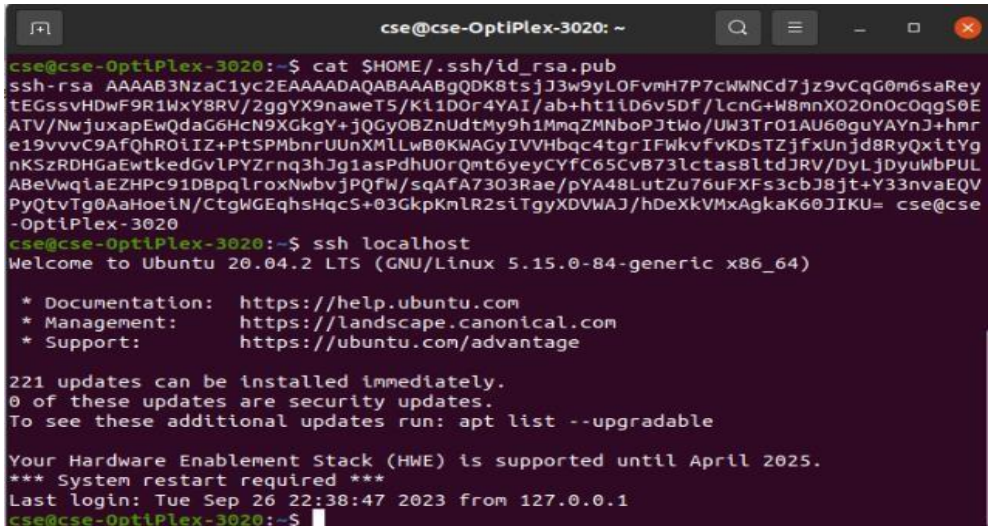
c) Configure password-less SSH

3. cat $HOME/.ssh/id_rsa.pub >> $HOME/.ssh/authorized_keys



d) Now verify the working of password-less ssh

$ ssh localhost



e) Now install rsync with command

$ sudo apt-get install rsync

STEP 1:Setup Configuration

a) Setting Up the environment variables

Edit .bashrc- Edit the bashrc and therefore add hadoop in a path:

$nano bash.bashrc

export HADOOP_HOME=/home/cse/hadoop-3.3.6

export HADOOP_INSTALL=$HADOOP_HOME

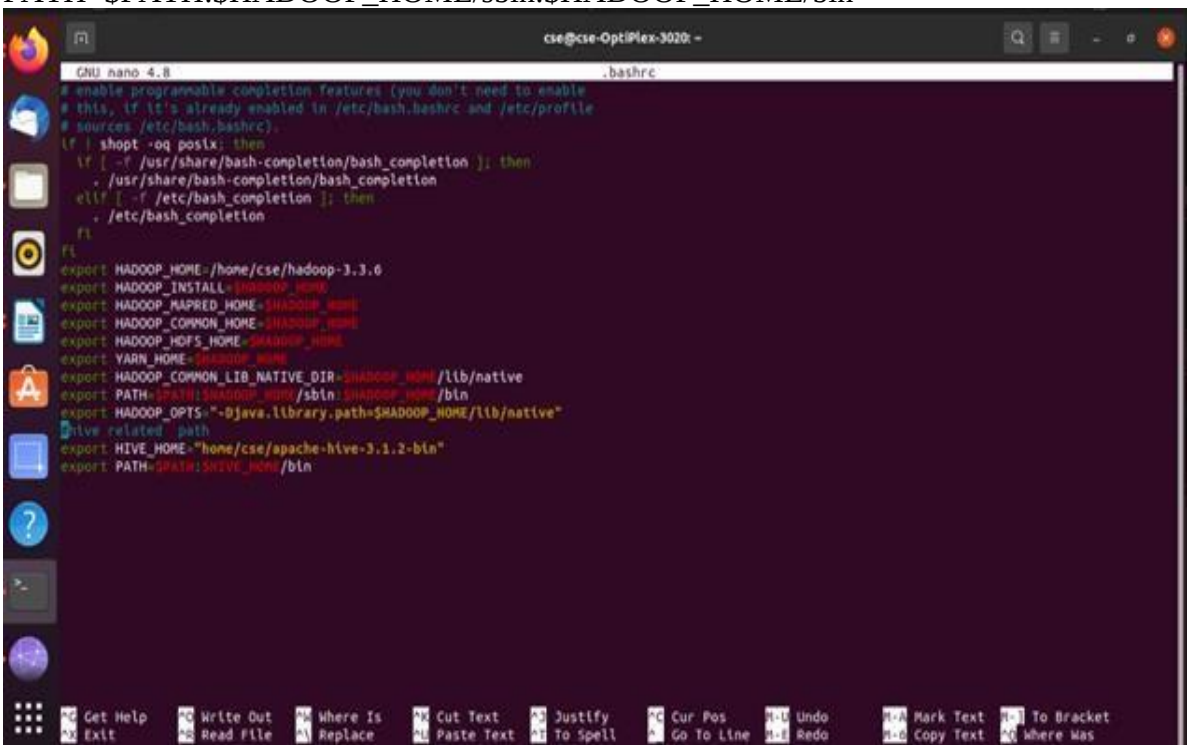
export HADOOP_MAPRED_HOME=$HADOOP_HOME

export HADOOP_COMMON_HOME=$HADOOP_HOME

export HADOOP_HDFS_HOME=$HADOOP_HOME

export YARN_HOME=$HADOOP_HOME

exportHADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native export PATH=$PATH:$HADOOP_HOME/sbin:$HADOOP_HOME/bin



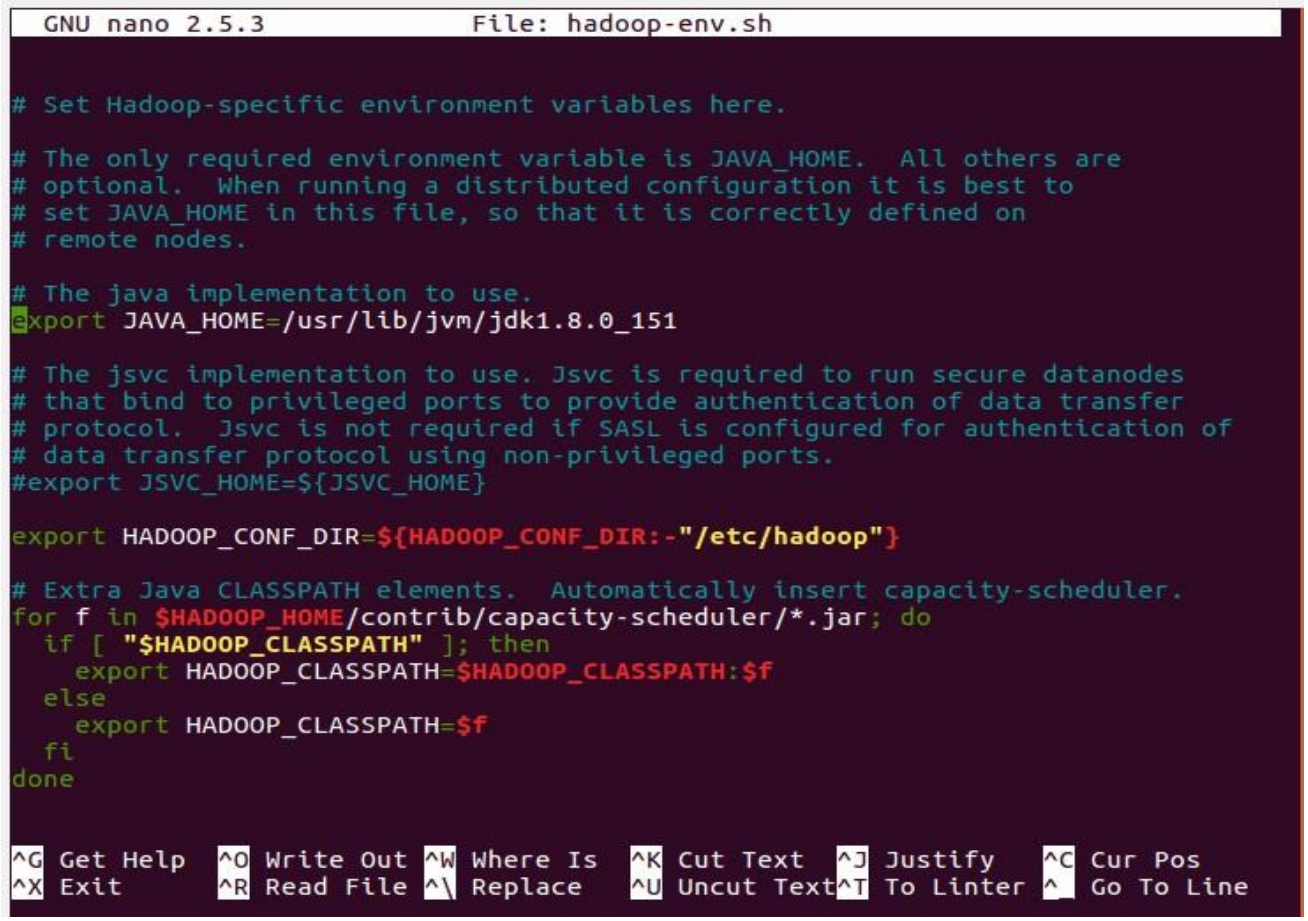
Source .bashrc in current login session in terminal

$source ~/.bashrc

b) Hadoop configuration file changes

Edit hadoop-env.sh

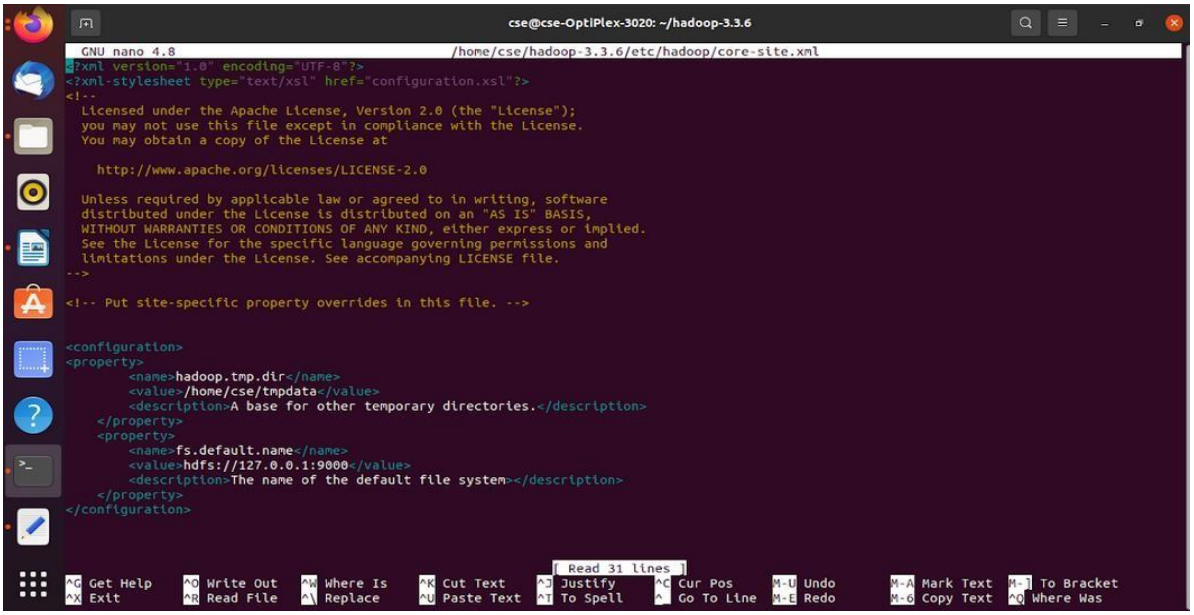
Edit hadoop-env.sh file which is in etc/hadoop inside the Hadoop installation directory.
$sudo nano $HADOOP_HOME/etc/hadoop/Hadoop-env.shThe
user can set JAVA_HOME:
export JAVA_HOME=<root directory of Java-installation> (eg:
/usr/lib/jvm/jdk1.8.0_151/)



Edit core-site.xml

$sudo nano $HADOOP_HOME/etc/hadoop/core-site.xml

```
<configuration>
<property>
<name>fs.defaultFS</name>
<value>hdfs://localhost:9000</value>
</property>
<property>
<name>hadoop.tmp.dir</name>
<value>/home/cse/hdata</value>
</property>
</configuration>
```

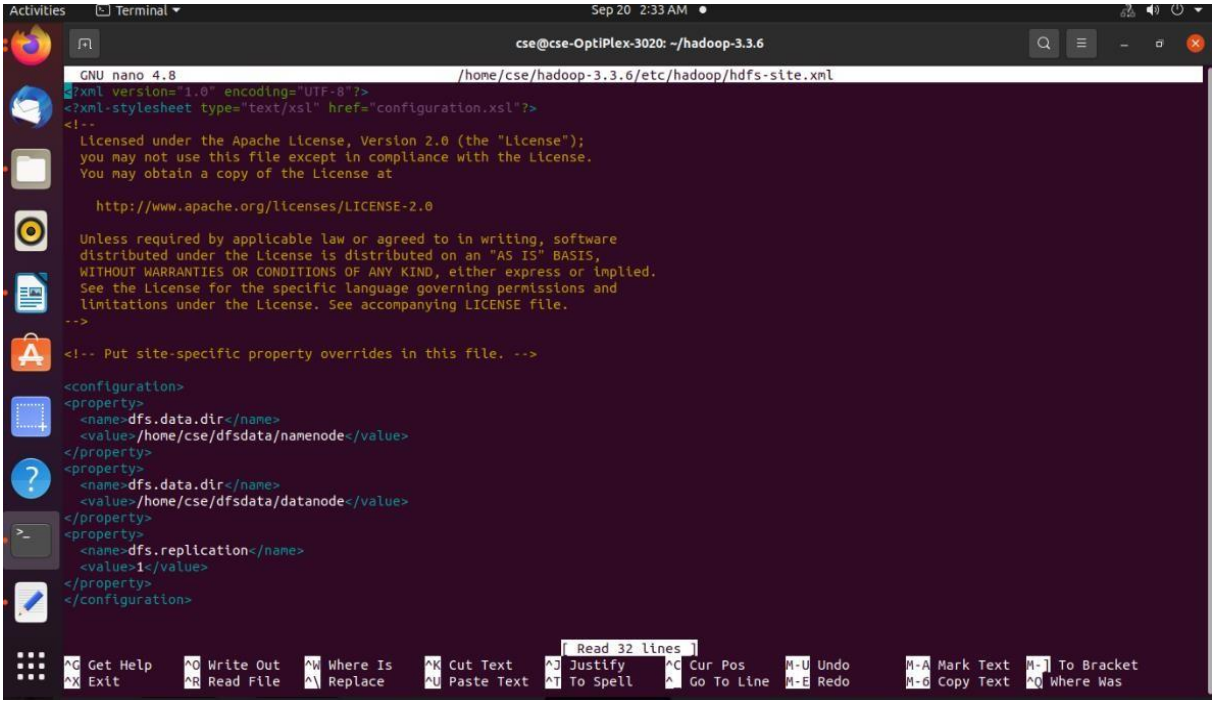


Edit hdfs-site.xml

$sudo nano $HADOOP_HOME/etc/hadoop/hdfs-site.xml

#Add below lines in this file(between "<configuration>" and "<"/configuration>")

```
<property>
  <name>dfs.data.dir</name>
  <value>/home/cse/dfsdata/namenode</value>
</property>
<property>
  <name>dfs.data.dir</name>
  <value>/home/cse/dfsdata/datanode</value>
</property>
<property>
  <name>dfs.replication</name>
  <value>1</value>
</property>
```

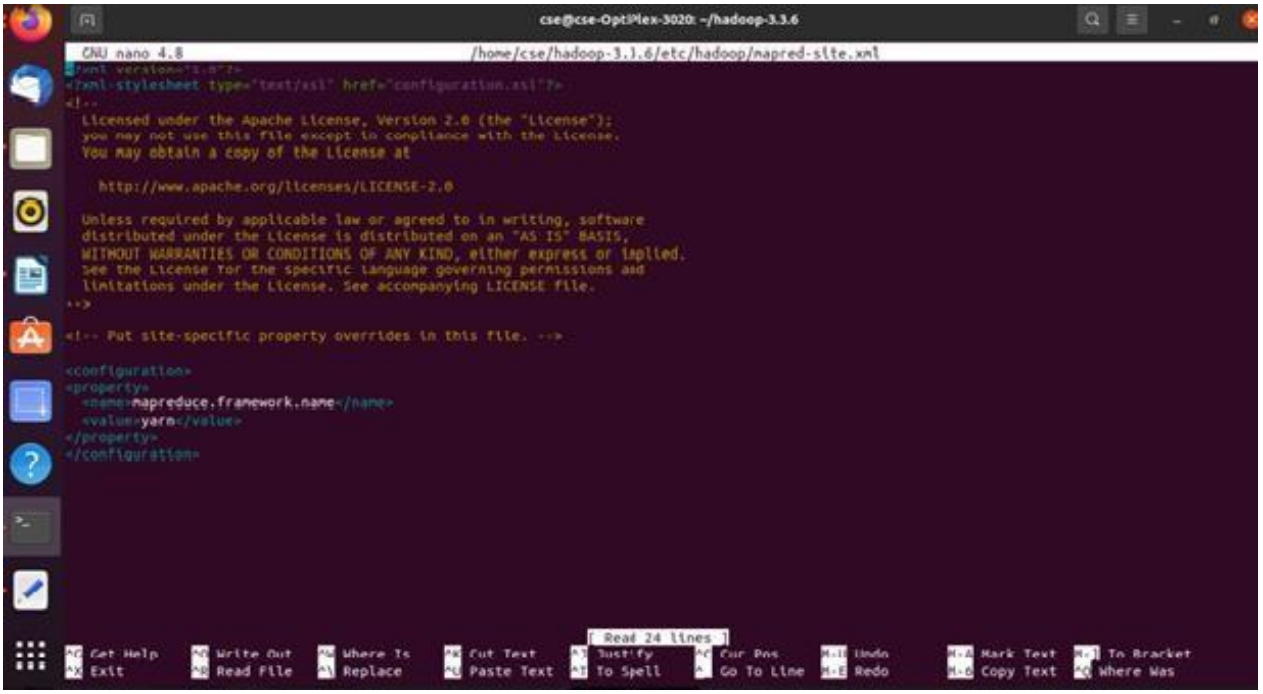


Edit mapred-site.xml

$sudo nano $HADOOP_HOME/etc/hadoop/mapred-site.xml

#Add below lines in this file(between "<configuration>" and "<"/configuration>")

```
<property>
  <name>mapreduce.framework.name</name>
  <value>yarn</value>
</property>
```

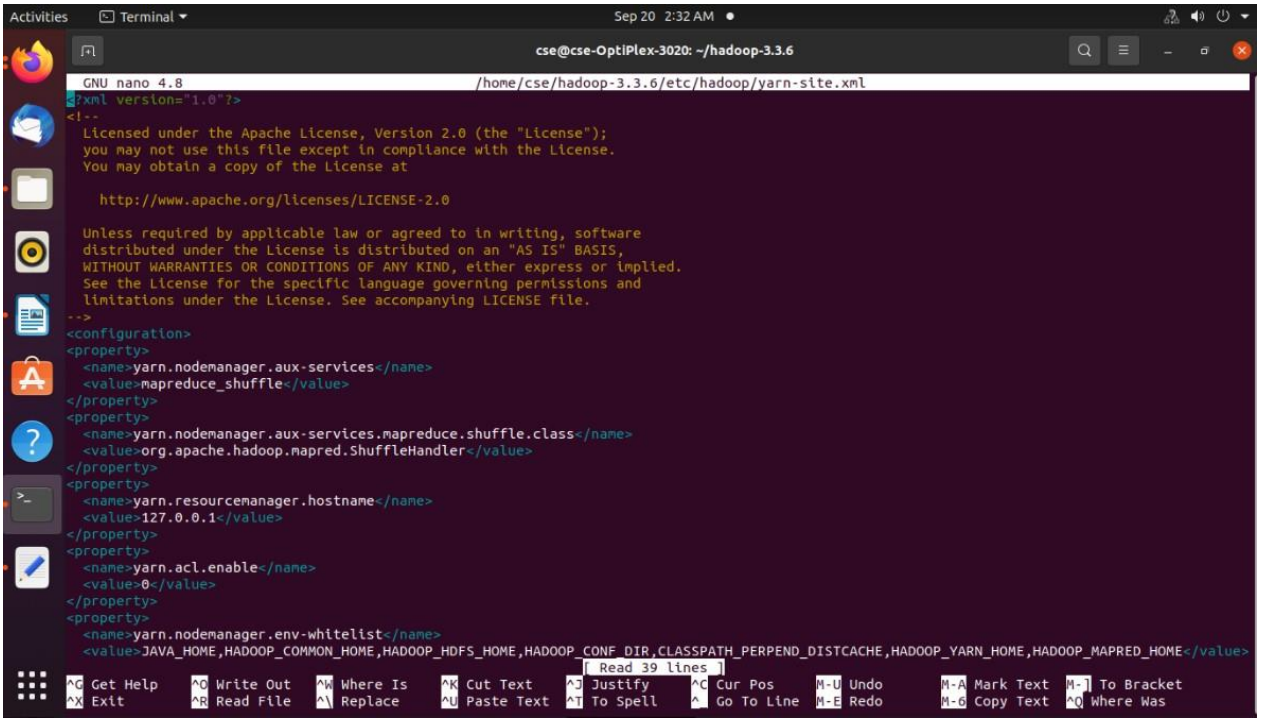


Edit yarn-site.xml

$sudo nano $HADOOP_HOME/etc/hadoop/yarn-site.xml

#Add below lines in this file(between "<configuration>" and "<"/configuration>")

```
<property>
  <name>yarn.nodemanager.aux-services</name>
  <value>mapreduce_shuffle</value>
</property>
<property>
  <name>yarn.nodemanager.aux-services.mapreduce.shuffle.class</name>
  <value>org.apache.hadoop.mapred.ShuffleHandler</value>
</property>
<property>
  <name>yarn.resourcemanager.hostname</name>
  <value>127.0.0.1</value>
</property>
<property>
  <name>yarn.acl.enable</name>
  <value>0</value>
</property>
<property>
  <name>yarn.nodemanager.env-whitelist</name>
<value>JAVA_HOME,HADOOP_COMMON_HOME,HADOOP_HDFS_HOME,HADO
OP_CONF_DIR,CLASSPATH_PERPEND_DISTCACHE,HADOOP_YARN_HOME,HA
DOOP_MAPRED_HOME</value>
</property>
```

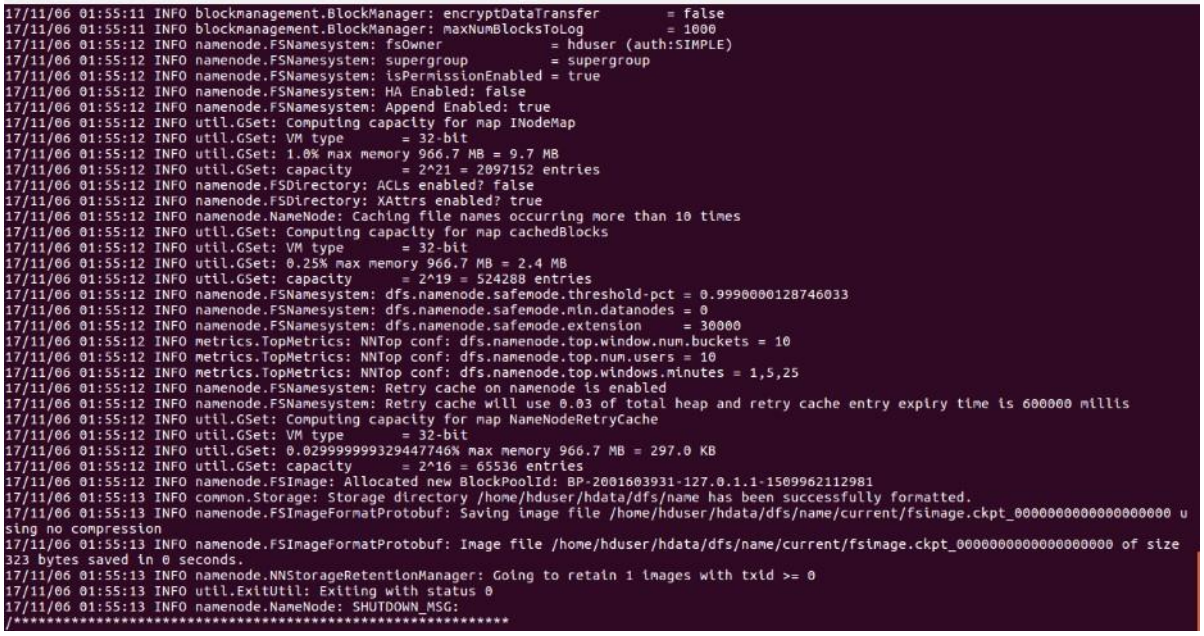


Step 2: Start the cluster

We will now start the single node cluster with the following commands.

a) Format the namenode

$hdfs namenode –format



b) Start the HDFS

$start-all.sh

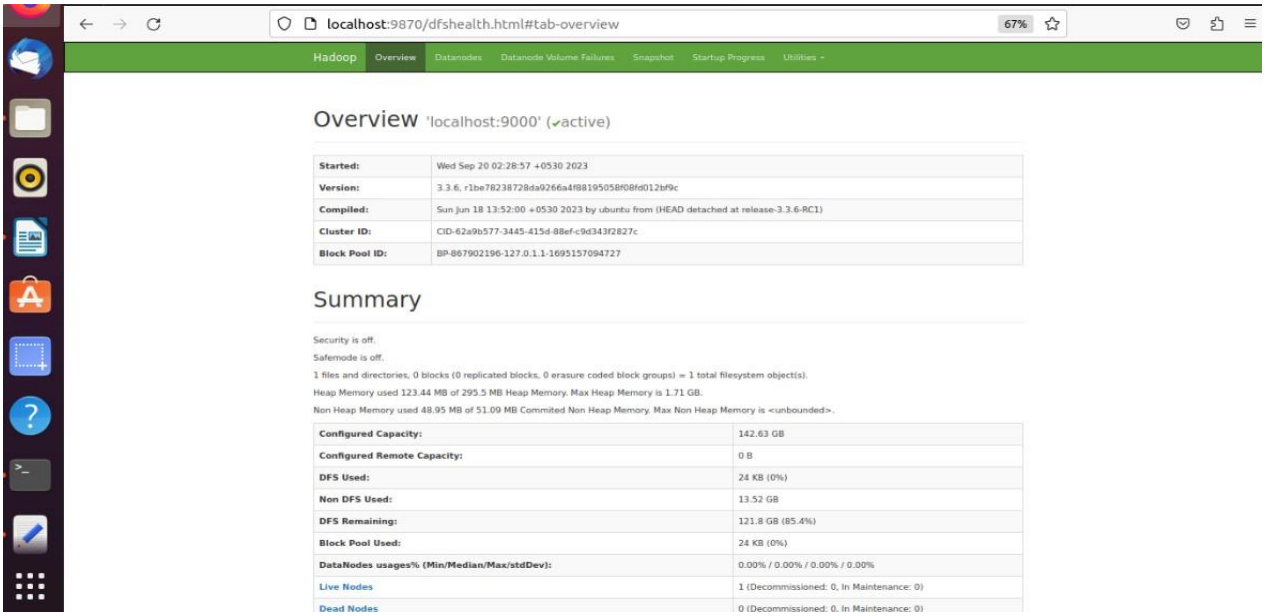
c)Verify if all process started

$ jps

6775 DataNode
7209 ResourceManager
7017 SecondaryNameNode
6651 NameNode
7339 NodeManager
7663 Jps

```
cse@cse-OptiPlex-3020:~/hadoop-3.3.6$ start-all.sh
WARNING: Attempting to start all Apache Hadoop daemons as cse in 10 seconds.
WARNING: This is not a recommended production deployment configuration.
WARNING: Use CTRL-C to abort.
Starting namenodes on [localhost]
Starting datanodes
Starting secondary namenodes [cse-OptiPlex-3020]
cse-OptiPlex-3020: Warning: Permanently added 'cse-optiplex-3020' (ECDSA) to the list of known hosts.
Starting resourcemanager
Starting nodemanagers
cse@cse-OptiPlex-3020:~/hadoop-3.3.6$ jps
3969 SecondaryNameNode
3587 NameNode
4166 ResourceManager
4297 NodeManager
3723 DataNode
4637 Jps
cse@cse-OptiPlex-3020:~/hadoop-3.3.6$
```

d)Web interface-For viewing Web UI of NameNode
visit : (http://localhost:9870)



RESULT:

Thus, Downloaded, and installed Hadoop and understand different Hadoop modes Startup scripts, Configuration files are successfully implemented

POSSIBLE VIVA QUESTIONS:

Pre-Viva Questions:

1. What is Hadoop and what are its main components?
2. How do you download and install Hadoop?
3. What are the different Hadoop modes?
4. What are the startup scripts for Hadoop?
5. What are the main Hadoop configuration files?

Post-Viva Questions:

1. Explain the process of downloading and installing Hadoop.
2. What are the different modes in which Hadoop can be configured?
3. Describe the purpose of startup scripts in Hadoop.
4. What are the main Hadoop configuration files and what do they configure?
5. How does the pseudo-distributed mode differ from the fully distributed mode in Hadoop?

EXPT.NO.2	Hadoop Implementation of file management tasks, such as Adding files anddirectories, retrieving files and Deleting files

AIM:

To implement the following file management tasks in Hadoop:

1. Adding files and directories
2. Retrieving files
3. Deleting Files

DESCRIPTION:-

HDFS is a scalable distributed filesystem designed to scale to petabytes of data while running on top of the underlying filesystem of the operating system. HDFS keeps track of where the data resides in a network by associating the name of its rack (or network switch) with the dataset. This allows Hadoop to efficiently scheduletasks to those nodes that contain data, or which are nearest to it, optimizing bandwidth utilization. Hadoop provides a set of command line utilities that work similarly to the Linux file commands, and serve as your primary interface with HDFS.

We‘re going to have a look into HDFS by interacting with it from the command line.

We will take a look at the most common file management tasks in Hadoop, which include:

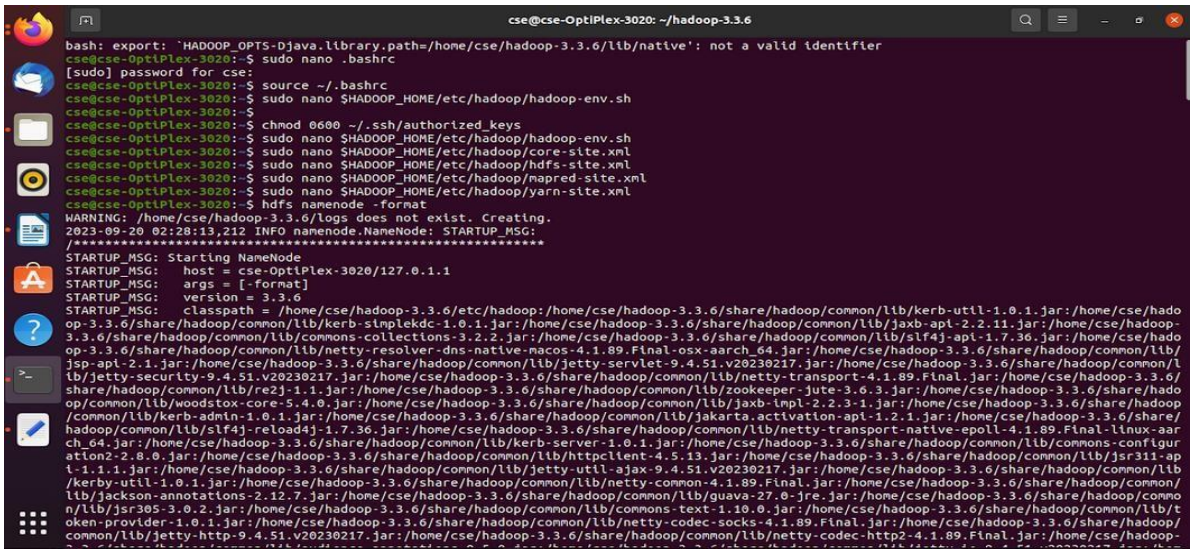
1. Adding files and directories to HDFS
2. Retrieving files from HDFS to local filesystem
3. Deleting files from HDFS

SYNTAX AND COMMANDS TO ADD, RETRIEVE AND DELETE DATA FROMHDFS

Step 1: Starting HDFS

Initially you have to format the configured HDFS file system, open namenode (HDFSserver), and execute the following command.

$ hadoop namenode -format



After formatting the HDFS, start the distributed file system. The following command will start the namenode as well as the data nodes as cluster.

$ start-dfs.sh

Listing Files in HDFS

After loading the information in the server, we can find the list of files in a directory, status of a file, using ls Given below is the syntax of ls that you can pass to a directory or a filename as an argument.

$ $HADOOP_HOME/bin/hadoop fs -ls <args>

```
ambal2@Ubuntu:~$ hadoop fs -ls
2023-10-12 11:37:33,233 WARN util.NativeCodeLoader: Unable to load native-hadoop
 library for your platform... using builtin-java classes where applicable
Found 4 items
drwxr-xr-x   - ambal2 supergroup          0 2023-10-11 13:04 bigdata
drwxr-xr-x   - ambal2 supergroup          0 2023-10-12 11:35 new
drwxr-xr-x   - ambal2 supergroup          0 2023-10-09 11:39 sqoop
drwxr-xr-x   - ambal2 supergroup          0 2023-10-09 12:41 sqoop1
ambal2@Ubuntu:~$
```

Inserting Data into HDFS

Assume we have data in the file called file.txt in the local system which is ought to be saved in the hdfs file system. Follow the steps given below to insert the required file in the Hadoop file system.

Step-2: Adding Files and Directories to HDFS

$ $HADOOP_HOME/bin/hadoop fs -mkdir /user/input

```
ambal2@Ubuntu:~$ hadoop fs -mkdir new
2023-10-12 11:33:52,575 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where


ambal2@Ubuntu:~$
ambal2@Ubuntu:~$
ambal2@Ubuntu:~$ hadoop fs -ls
2023-10-12 11:37:33,233 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where
Found 4 items
drwxr-xr-x   - ambal2 supergroup          0 2023-10-11 13:04 bigdata
drwxr-xr-x   - ambal2 supergroup          0 2023-10-12 11:35 new
drwxr-xr-x   - ambal2 supergroup          0 2023-10-09 11:39 sqoop
drwxr-xr-x   - ambal2 supergroup          0 2023-10-09 12:41 sqoop1
```

Transfer and store a data file from local systems to the Hadoop file system using the put command.

$ $HADOOP_HOME/bin/hadoop fs -put /home/file.txt /user/input

Step 3 :You can verify the file using ls command.

$ $HADOOP_HOME/bin/hadoop fs -ls /user/input

Step 4 Retrieving Data from HDFS

Assume we have a file in HDFS called outfile. Given below is a simple demonstration for retrieving the required file from the Hadoop file system.

Initially, view the data from HDFS using cat command.

$ $HADOOP_HOME/bin/hadoop fs -cat /user/output/outfile

Get the file from HDFS to the local file system using get command.

$ $HADOOP_HOME/bin/hadoop fs -get /user/output/ /home/hadoop_tp/

Step-5: Deleting Files from HDFS

$ hadoop fs -rm file.txt

Step 6: Shutting Down the HDFS

You can shut down the HDFS by using the following command.

$ stop-dfs.sh

RESULT:

Thus, the Installing of Hadoop in three operating modes has been successfully completed.

POSSIBLE VIVA QUESTIONS:

Pre-Viva Questions:

1. How do you add files to HDFS in Hadoop?
2. How do you retrieve files from HDFS?
3. What command would you use to list the contents of a directory in HDFS?
4. How do you check the current usage of HDFS space?
5. What command would you use to move a file from one location to another within HDFS?

Post-Viva Questions:

1. Explain the steps to add a file to HDFS. What commands do you use, and what are the necessary considerations?
2. What are the potential error messages you might encounter, and how do you resolve them?
3. How do you retrieve a file from HDFS to the local file system? Provide the command and explain any options you might use.
4. What command would you use to delete a file or directory in HDFS? Explain the different options available for this command.
5. How do you verify the contents of a directory in HDFS? What are the possible outputs of this command?

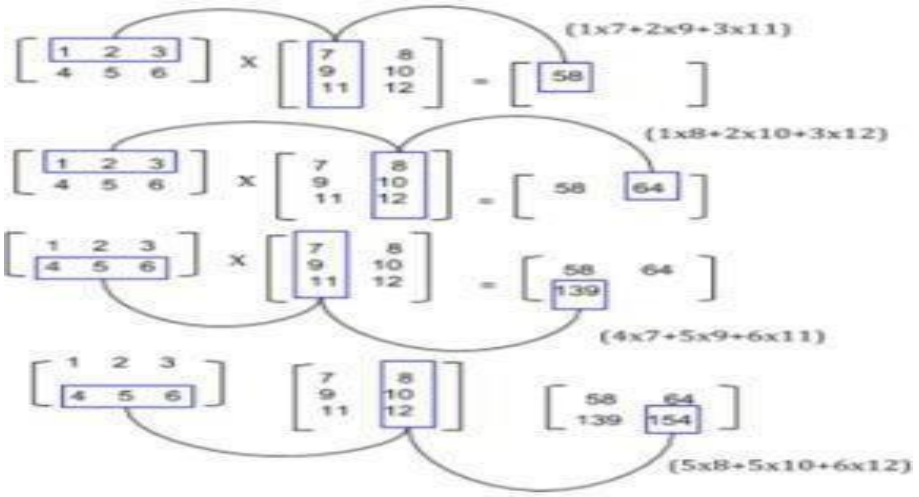
EXPT.NO.3	Implement of Matrix Multiplication with Hadoop Map Reduce

AIM:

To Develop a MapReduce program to implement Matrix Multiplication.

Description:

In **mathematics**, **matrix multiplication** or the **matrix product** is a binary operationthat produces a matrix from two matrices. The definition is motivated by linear equations and linear transformations on vectors, which have numerous applicationsin applied mathematics,physics, and engineering. In more detail, if **A** is an $n \times m$ matrix and **B** is an $m \times p$ matrix, their matrix product **AB** is an $n \times p$ matrix, in which the m entries across a row of **A** are multiplied with the m entries down a column of **B** and summed to produce an entry of **AB**. When two linear transformations are represented by matrices, then the matrix productrepresents the composition of the two transformations.



Algorithm for Map Function.

a. for each element mij of M do
produce (key,value) pairs as ((i,k), (M,j,mij), for k=1,2,3,.. upto the number of columns of N

b. for each element njk of N do
produce (key,value) pairs as ((i,k),(N,j,Njk), for i = 1,2,3,.. Upto the number ofrows of M.

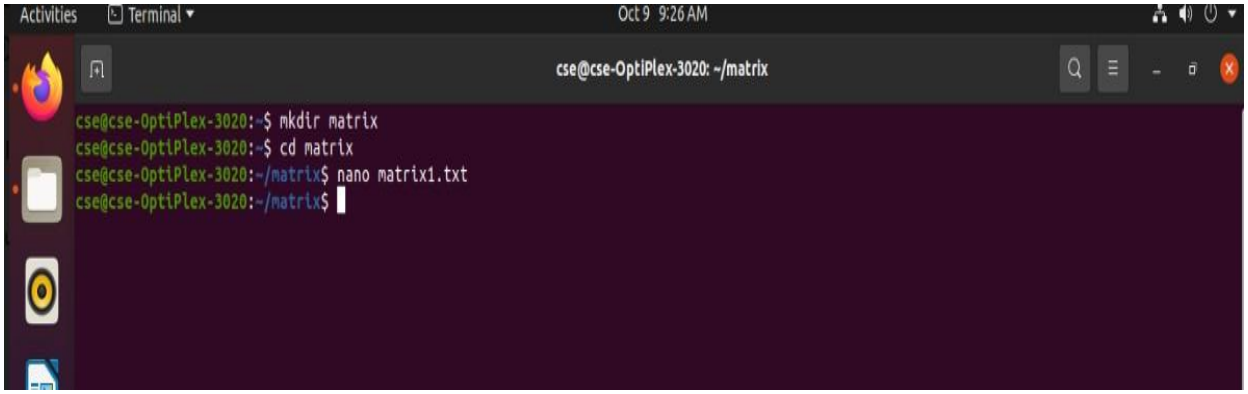
c. return Set of (key,value) pairs that each key (i,k), has list with values (M,j,mij) and (N, j,njk) for all possible values of j.

Algorithm for Reduce Function.

d. for each key (i,k) do
e. sort values begin with M by j in listM sort values begin with N by j in listN
   multiply mij and njk for jth value of each list
f. sum up mij x njk return (i,k), Σj=1 mij x njk

Step 1. Creating directory for matrix

Then open matrix1.txt and matrix2.txt put the values in that text files



Step 2. Creating Mapper file for Matrix Multiplication.

```
#!/usr/bin/env python
import sys
cache_info = open("cache.txt").readlines()[0].split(",")
row_a, col_b = map(int,cache_info)
for line in sys.stdin:
        matrix_index, row, col, value = line.rstrip().split(",")if
        matrix_index == "A":
                for i in xrange(0,col_b):
                        key = row + "," + str(i)
                        print "%s\t%s\t%s"%(key,col,value)
        else:
                for j in xrange(0,row_a):
                        key = str(j) + "," + col
                        print "%s\t%s\t%s"%(key,row,value)
```

Step 3. Creating reducer file for Matrix Multiplication.

```
#!/usr/bin/env python
import sys
from operator import itemgetter
prev_index = None
value_list = []
for line in sys.stdin:
    curr_index, index, value = line.rstrip().split("\t")
    index, value = map(int,[index,value])
    if curr_index == prev_index: value_list.append((index,value))
    else:
        if prev_index:
            value_list = sorted(value_list,key=itemgetter(0))i =
            0
            result = 0
            while i < len(value_list) - 1:
                if value_list[i][0] == value_list[i + 1][0]:
```

```
                result += value_list[i][1]*value_list[i + 1][1]i
                += 2
            else:
                i += 1
        print "%s,%s"%(prev_index,str(result))prev_index =
    curr_index
    value_list = [(index,value)]

if curr_index == prev_index:
    value_list = sorted(value_list,key=itemgetter(0))i =
    0
    result = 0
    while i < len(value_list) - 1:
        if value_list[i][0] == value_list[i + 1][0]:
            result += value_list[i][1]*value_list[i + 1][1]i
            += 2
        else:
            i += 1
    print "%s,%s"%(prev_index,str(result))
```

Step 4. To view this file using cat command

$cat *.txt |python mapper.py

$ chmod +x ~/Desktop/mr/matrix/Mapper.py

$ chmod +x ~/Desktop/mr/matrixl/Reducer.py

$ hadoop jar /usr/lib/hadoop-mapreduce/hadoop-streaming.jar \

> -input /user/cse/matrices/ \

> -output /user/cse/mat_output \

> -mapper ~/Desktop/mr/matrix/Mapper.py \

> -reducer ~/Desktop/mr/matrix/Reducer.py

Step 5: To view this full output

RESULT:

Thus, the MapReduce program to implement Matrix Multiplication was successfully executed.

POSSIBLE VIVA QUESTIONS:

Pre-Viva Questions:

1. How do you handle the input format for matrices in Hadoop MapReduce? What does each line of the input represent?
2. How does Hadoop ensure data locality and fault tolerance during the matrix multiplication process?
3. What are some potential challenges or pitfalls when implementing matrix multiplication using Hadoop MapReduce, and how can they be mitigated?
4. What is the significance of the shuffle and sort phase in Hadoop MapReduce, especially in the context of matrix multiplication?
5. How do you handle matrix sparsity in Hadoop MapReduce for matrix multiplication?

Post-Viva Questions:

1. What are the primary challenges you faced while implementing matrix multiplication using Hadoop MapReduce, and how did you overcome them?
2. How did you handle the input and output data formats in your Hadoop MapReduce implementation? What considerations were made for efficient data parsing and writing?
3. What are the implications of matrix sparsity on your Hadoop MapReduce implementation? How did you manage sparse matrices?
4. What are the key differences between your initial design and the final implementation of matrix multiplication in Hadoop MapReduce?
5. What future improvements or features would you consider adding to your matrix multiplication implementation to enhance its performance and usability?

EXPT.NO.4	Run a basic Word Count Map Reduce program to understand Map ReduceParadigm

AIM:

To Develop a MapReduce program to calculate the frequency of a given wordin a given file.

Map Function – It takes a set of data and converts it into another set ofdata, where individual elements are broken down into tuples (Key-Value pair).

Example – (Map function in Word Count)

Input

Set of data

Bus, Car, bus, car, train, car, bus, car, train, bus, TRAIN,BUS, buS, caR, CAR, car, BUS,TRAIN

Output

Convert into another set

of data(Key,Value)

(Bus,1), (Car,1), (bus,1), (car,1), (train,1), (car,1), (bus,1), (car,1), (train,1), (bus,1),

(TRAIN,1),(BUS,1), (buS,1), (caR,1), (CAR,1), (car,1), (BUS,1), (TRAIN,1)

Reduce Function – Takes the output from Map as an input and combinesthose

data tuplesinto a smaller set of tuples.

Example – (Reduce function in Word Count)

Input Set of

Tuples(output of

Map function)

(Bus,1), (Car,1), (bus,1), (car,1), (train,1), (car,1), (bus,1), (car,1),

(train,1), (bus,1),(TRAIN,1),(BUS,1),

(buS,1),(caR,1),(CAR,1), (car,1), (BUS,1), (TRAIN,1)

Output **Converts into smaller set of tuples**

(BUS,7), (CAR,7), (TRAIN,4)

Workflow of MapReduce consists of 5 steps

1. **Splitting –** The splitting parameter can be anything, e.g. splitting by space,comma, semicolon, or even by a new line ('\n').
2. **Mapping –** as explained above
3. Intermediate splitting – the entire process in parallel on different clusters. In orderto group them in "Reduce Phase" the similar KEY data should be on same cluster.
4. **Reduce –** it is nothing but mostly group by phase
5. **Combining –** The last phase where all the data (individual result set from eachcluster) is combine together to form a Result

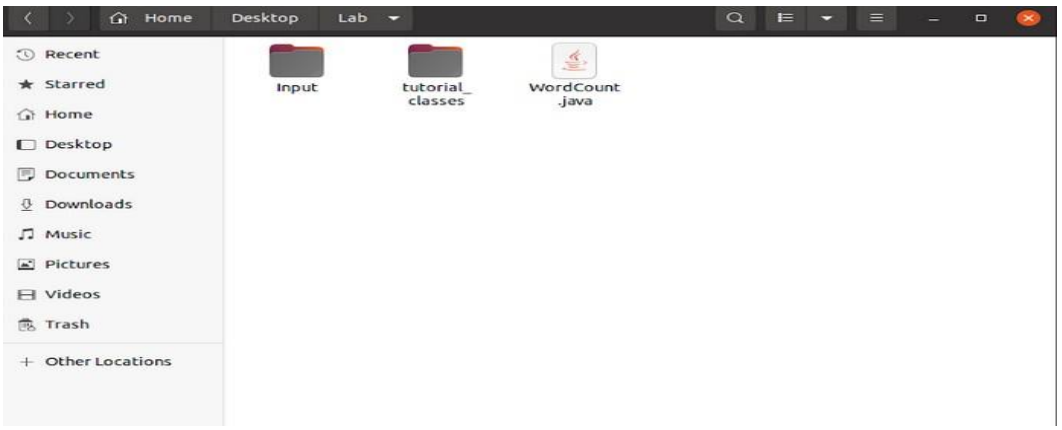
Now Let's See the Word Count Program in Java

Step1 : Make sure Hadoop and Java are installed properly

hadoop version

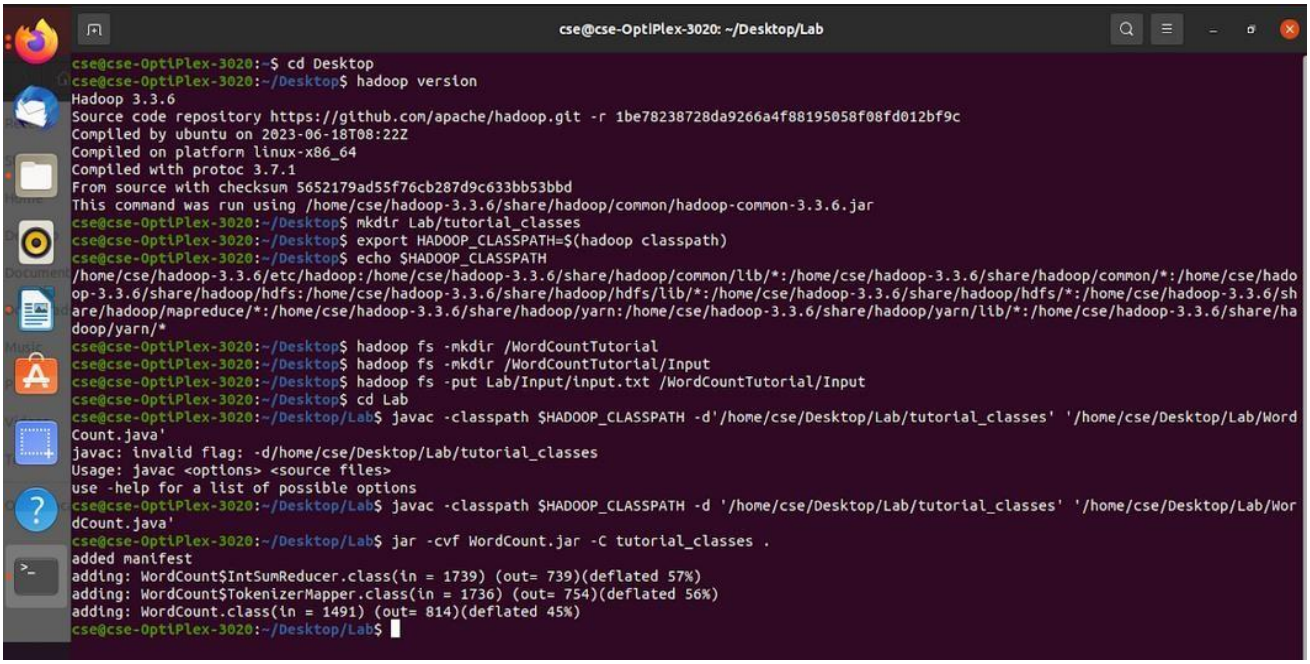
javac –version

Step 2. Create a directory on the Desktop named Lab and inside it create two folders;
one called "Input" and the other called "tutorial_classes". [You can do this step using GUI normally or through terminalcommands]
cd Desktop mkdir
Lab mkdir
Lab/Input
mkdir Lab/tutorial_classes
Step 3. Add the file attached with this document "WordCount.java" in the directory Lab
Step 4. Add the file attached with this document "input.txt" inthe directory Lab/Input.

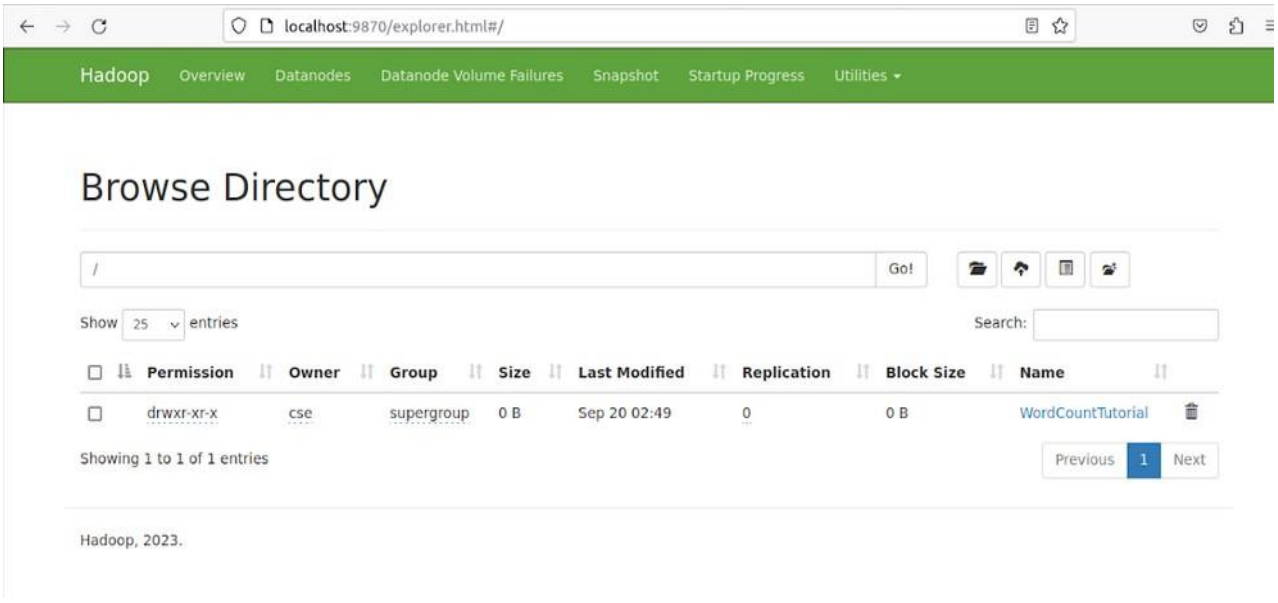


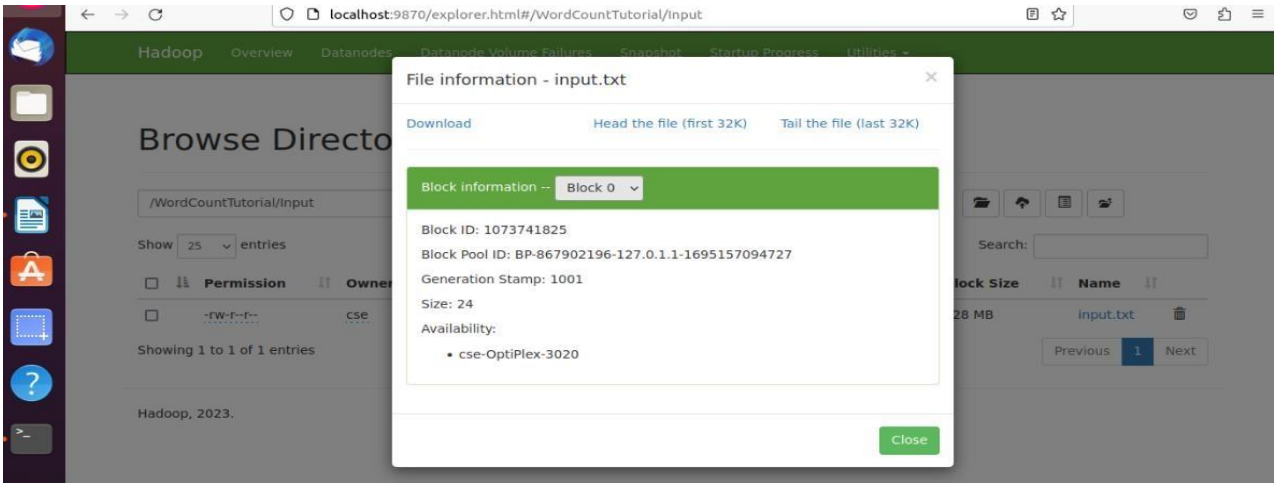
Step 5.
Type the following command to export the hadoopclasspath into bash.
export HADOOP_CLASSPATH=$(hadoop classpath)
Make sure it is now exported.
echo $HADOOP_CLASSPATH
Step 6. It is time to create these directories on HDFS rather thanlocally. Type the following commands.
hadoop fs -mkdir /WordCountTutorial hadoop fs -mkdir /WordCountTutorial/Input

hadoop fs -put Lab/Input/input.txt /WordCountTutorial/Input



Step 7. Go to localhost:9870 from the browser, Open"Utilities→ Browse File System"and you should see the directories and files we placed in the file system.





Step 8. Then, back to local machine where we will compile theWordCount.java file. Assuming we are currently in the Desktop directory.

cd Lab

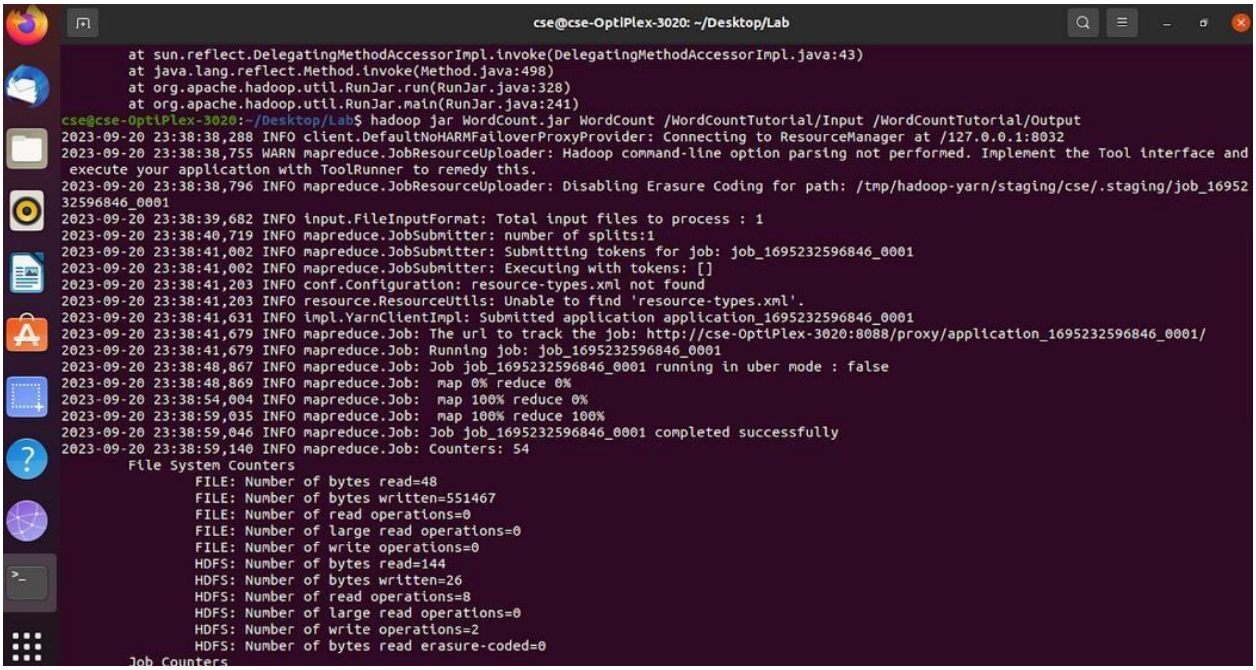
javac -classpath $HADOOP_CLASSPATH -d tutorial_classes WordCount.java

Put the output files in one jar file (There is a dot at the end)

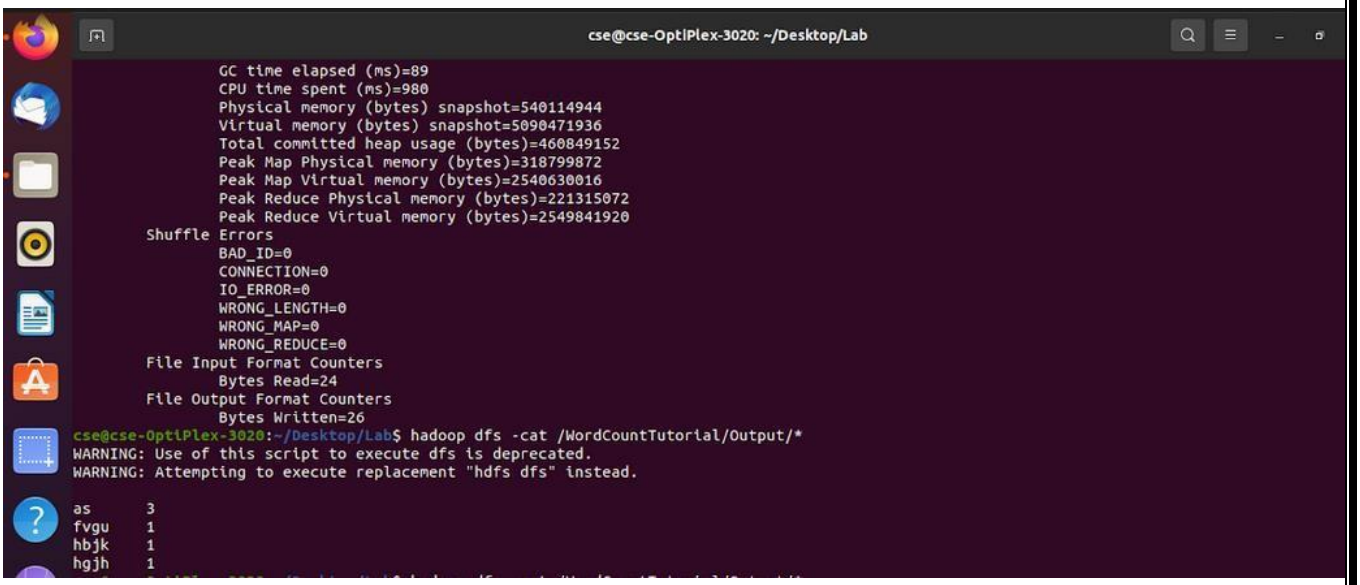
jar -cvf WordCount.jar -C tutorial_classes .

Step 9. Now, we run the jar file on Hadoop.

hadoop jar WordCount.jar WordCount /WordCountTutorial/Input /WordCountTutorial/Output



Step 10. Output the result:



hadoop dfs -cat /WordCountTutorial/Output/*

Program: Step 5. Type following Program :

package PackageDemo; import
java.io.IOException;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;

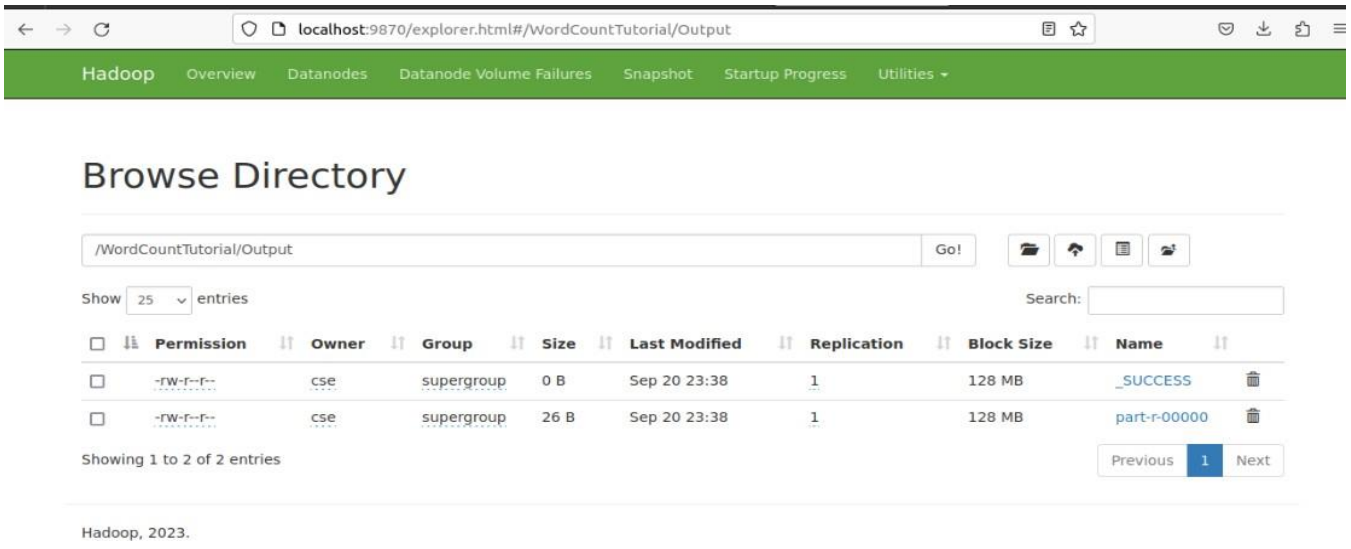
```
import
org.apache.hadoop.io.LongWritable;
import  org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import  org.apache.hadoop.mapreduce.lib.input.FileInputFormat;  import
org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;         import
org.apache.hadoop.util.GenericOptionsParser;
public class WordCount {
public static void main(String [] args) throws Exception
{
Configuration c=new  Configuration();String[]
files=new
GenericOptionsParser(c,args).getRemainingArgs(); Path
input=new Path(files[0]);
Path output=new Path(files[1]);
Job j=new Job(c,"wordcount");
j.setJarByClass(WordCount.c
lass);
j.setMapperClass(MapForWordCount.class);
j.setReducerClass(ReduceForWordCount.class);
j.setOutputKeyClass(Text.class);
j.setOutputValueClass(IntWritable.class);
FileInputFormat.addInputPath(j, input);
FileOutputFormat.setOutputPath(j, output);
System.exit(j.waitForCompletion(true)?0:1);
}
public static class MapForWordCount extends Mapper<LongWritable, Text,Text,IntWritable>{
public void map(LongWritable  key, Text  value, Context  con) throws IOException,
InterruptedException
{
String line = value.toString();
String[]
words=line.split(",");
for(String word: words
)
{
   Text outputKey = new
   Text(word.toUpperCase().trim());IntWritable
   outputValue = new IntWritable(1);
   con.write(outputKey, outputValue);
}
}
}
public static class ReduceForWordCount extends Reducer<Text, IntWritable,Text,IntWritable>
{
public void reduce(Text word, Iterable<IntWritable> values, Context con)throwsIOException,
InterruptedException
{
int sum = 0;
for(IntWritable value : values)
{
sum += value.get();
}
```

```
    con.write(word, new IntWritable(sum));
    }
    }
    }
```

The output is stored in /r_output/part-00000

```
cse@cse-OptiPlex-3020:~/Desktop/Lab$ hadoop dfs -cat /WordCountTutorial/Output/*
WARNING: Use of this script to execute dfs is deprecated.
WARNING: Attempting to execute replacement "hdfs dfs" instead.

as      3
fvgu    1
hbjk    1
hgjh    1
```

OUTPUT:



RESULT:

Thus, the Word Count Map Reduce program to understand Map Reduce Paradigm was successfully executed.

POSSIBLE VIVA QUESTIONS:

Pre-Viva Questions:

1. What is the role of the Mapper and Reducer in the Word Count MapReduce program? How do they interact with each other?
2. What are the input and output formats of the Word Count MapReduce program? How do you structure the data for processing?
3. How do you configure and run a basic Word Count MapReduce job in Hadoop? What are the essential steps and commands?
4. Can you explain the overall workflow of a basic Word Count MapReduce program? How does it process the input data to produce the output?
5. How do you handle Word Count MapReduce program?

Post-Viva Questions:

1. What were the key challenges you faced when implementing the Word Count MapReduce program, and how did you address them?
2. How did you ensure the correctness of the word count results in your MapReduce job? What validation techniques did you use?
3. Explain how the performance of your Word Count MapReduce job could be improved. What optimization techniques did you apply or consider?
4. Describe the process of handling errors or failures in your MapReduce job. What strategies did you use for error handling and job recovery?
5. How did you monitor and manage the Hadoop MapReduce job execution? What tools or techniques did you use to track the job's progress and diagnose issues?

EXPT.NO.5	Installation hive

AIM:

To installing hive with example

PROCEDURE:

Steps for hive installation

Download and Unzip Hive
Edit .bashrc file
Edit hive-config.sh file
Create Hive directories in HDFS
Initiate Derby database
Configure hive-site.xml file

Step 1:

download and unzip Hive

==========

wget https://downloads.apache.org/hive/hive-3.1.2/apache-hive-3.1.2-bin.tar.gztar xzf apache-hive-3.1.2-bin.tar.gz

Step 2:

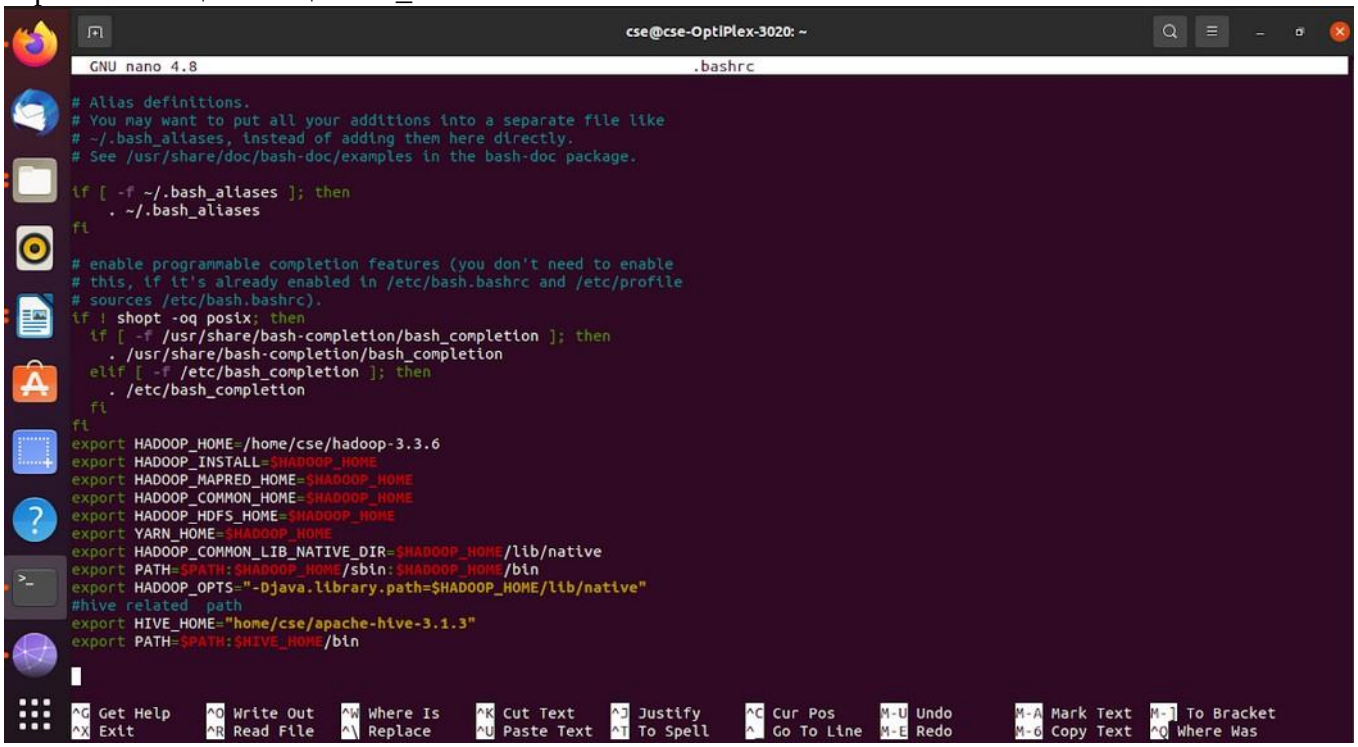
Edit .bashrc file

==========

sudo nano .bashrc

export HIVE_HOME= /home/hdoop/apache-hive-3.1.2-bin

export PATH=$PATH:$HIVE_HOME/bin



Step 3:

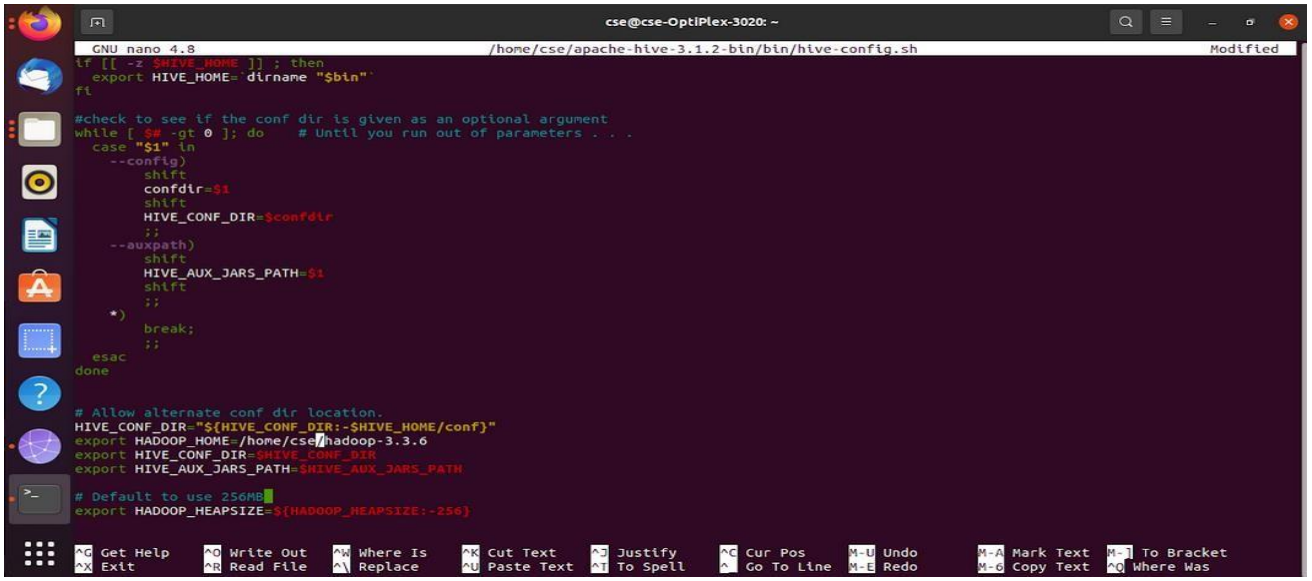
source ~/.bashrc

Step 4:

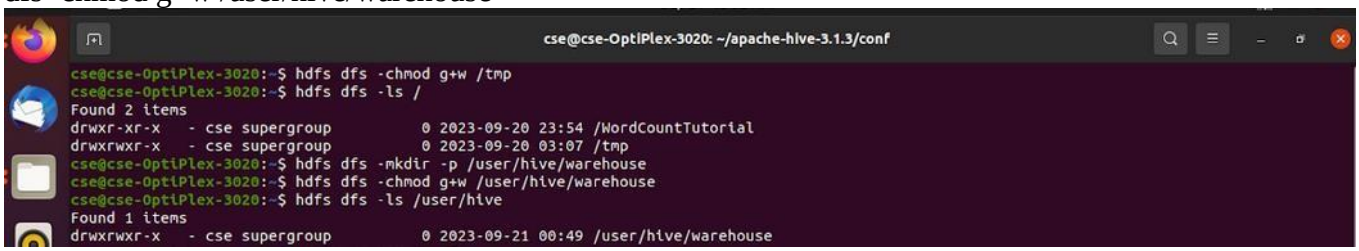
Edit hive-config.sh file

==========

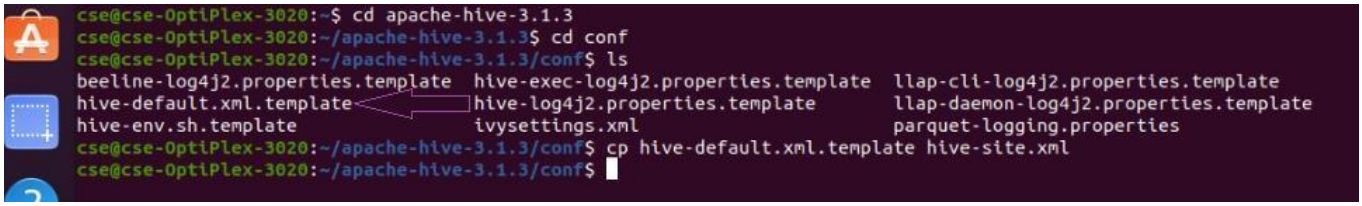
sudo nano $HIVE_HOME/bin/hive-config.sh export
HADOOP_HOME=/home/cse/hadoop-3.3.6



Step 5:
Create Hive directories in HDFS
==========
hdfs dfs -mkdir /tmp
hdfs dfs -chmod g+w /tmp
hdfs dfs -mkdir -p /user/hive/warehouse hdfs
dfs -chmod g+w /user/hive/warehouse



Step 6:
Fixing guava problem – Additional step
==========
rm $HIVE_HOME/lib/guava-19.0.jar
cp $HADOOP_HOME/share/hadoop/hdfs/lib/guava-27.0-jre.jar $HIVE_HOME/lib/
Step 7: Configure hive-site.xml File (Optional)
Use the following command to locate the correct file:cd
$HIVE_HOME/conf
List the files contained in the folder using the ls command.



Use the hive-default.xml.template to create the hive-site.xml file:

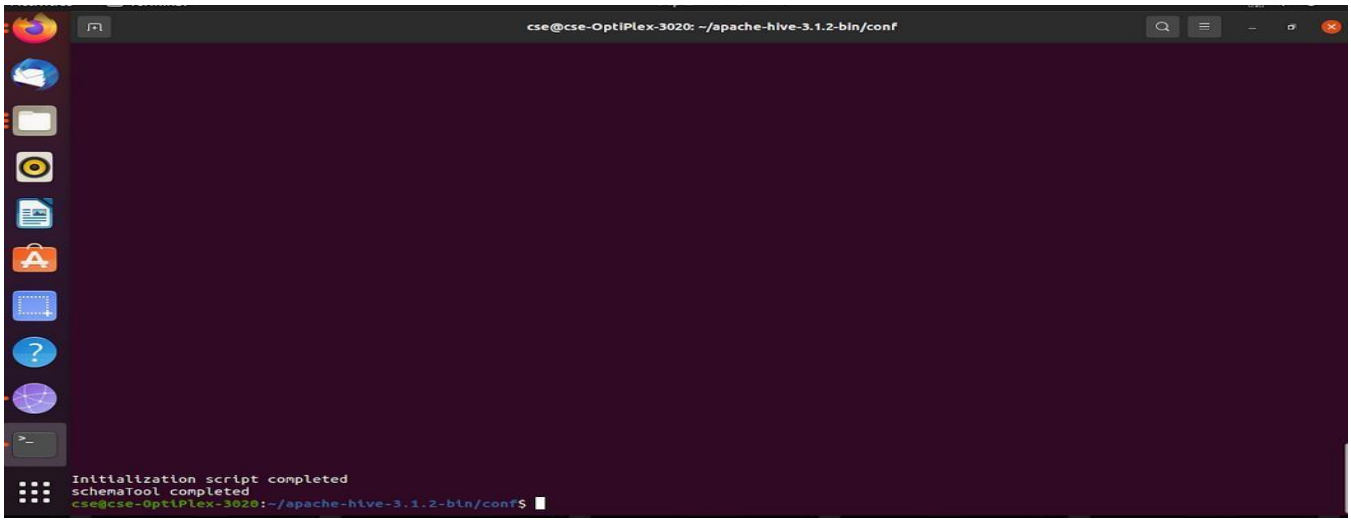
cp hive-default.xml.template hive-site.xml
Access the hive-site.xml file using the nano text editor:sudo
nano hive-site.xml

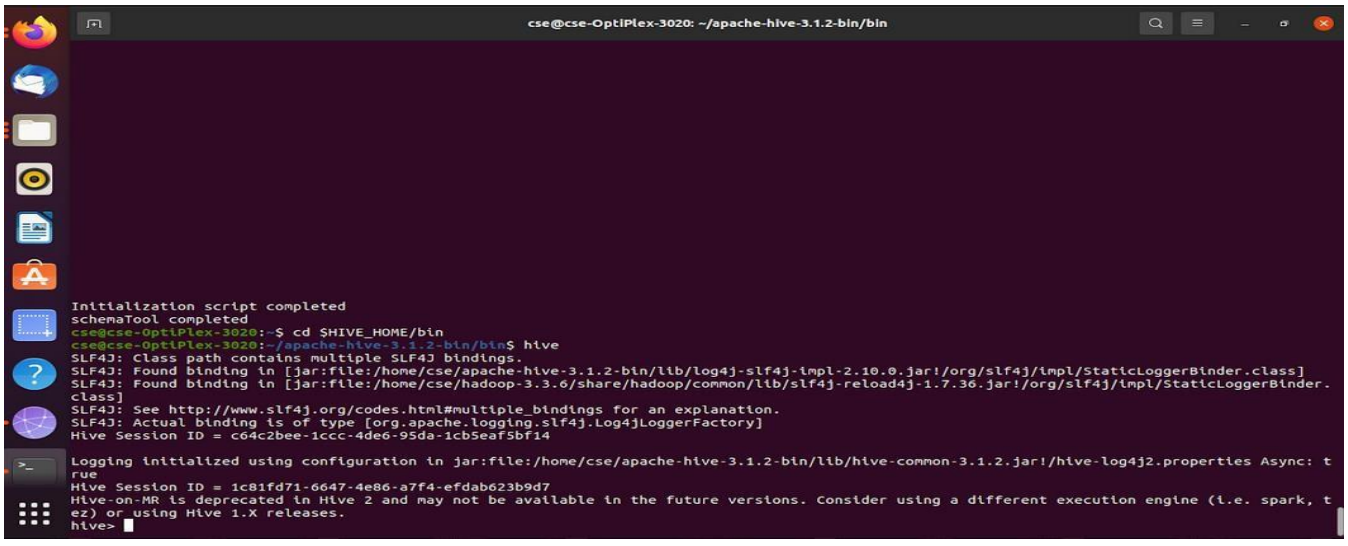
```
    <value>DERBY</value>
    <description>
      Expects one of [derby, oracle, mysql, mssql, postgres].
      Type of database used by the metastore. Information schema &amp;amp; JDBCSto
    </description>
  </property>
  <property>
    <name>hive.metastore.warehouse.dir</name>
    <value>/user/hive/warehouse</value>
    <description>location of default database for the warehouse</description>
  </property>
  <property>
    <name>hive.metastore.warehouse.external.dir</name>
    <value/>
    <description>Default location for external tables created in the warehouse
  </property>
  <property>
    <name>hive.metastore.uris</name>
    <value/>
    <description>Thrift URI for the remote metastore. Used by metastore client
  </property>
```

Step 8: Initiate Derby Database

==========

$HIVE_HOME/bin/schematool -dbType derby –initSchema





RESULT:

Thus, the installation hive was successfully installed and executed.

POSSIBLE VIVA QUESTIONS:

Pre-Viva Questions:

1. What are the prerequisites for installing Apache Hive on a Hadoop cluster?
2. What are the key configuration files you need to modify?
3. How do you configure the Hive Metastore database? What are the common issues you might face during this configuration?
4. What are some common practice examples you might run after installing Hive to verify that it is working correctly?
5. How would you troubleshoot Hive installation issues? What steps would you take to diagnose and resolve common problems?

Post-Viva Questions:

1. What were the key challenges you faced during the installation of Apache Hive, and how did you resolve them?
2. What tests or examples did you use to confirm that Hive was installed and configured properly?
3. How you validated the connection between Hive and the Metastore database. What tools or commands did you use for this validation?
4. What changes did you make to improve Hive performance?

EXPT.NO.6	Installation of HBase, Installing thrift along with Practice examples

AIM:

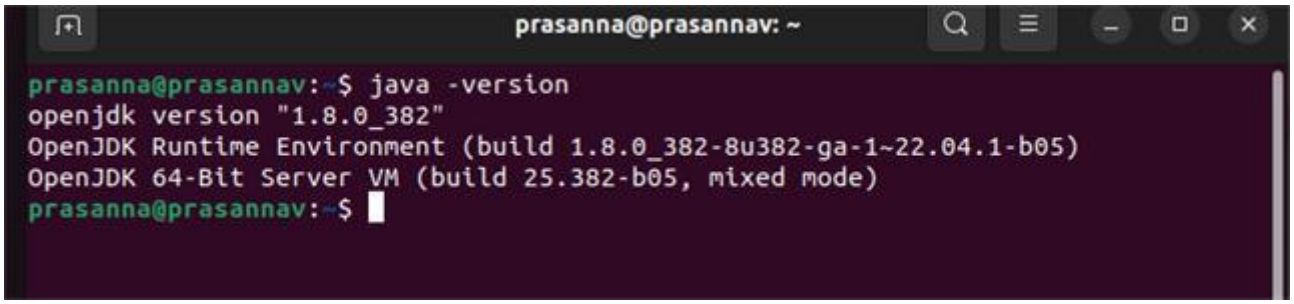
To Install HBase on Ubuntu 18.04 HBase in Standalone Mode

PROCEDURE:

Pre-requisite:

Ubuntu 16.04 or higher installed on a virtual machine**.**

Step-1: Make sure that java has installed in your machine to verify that run java –version

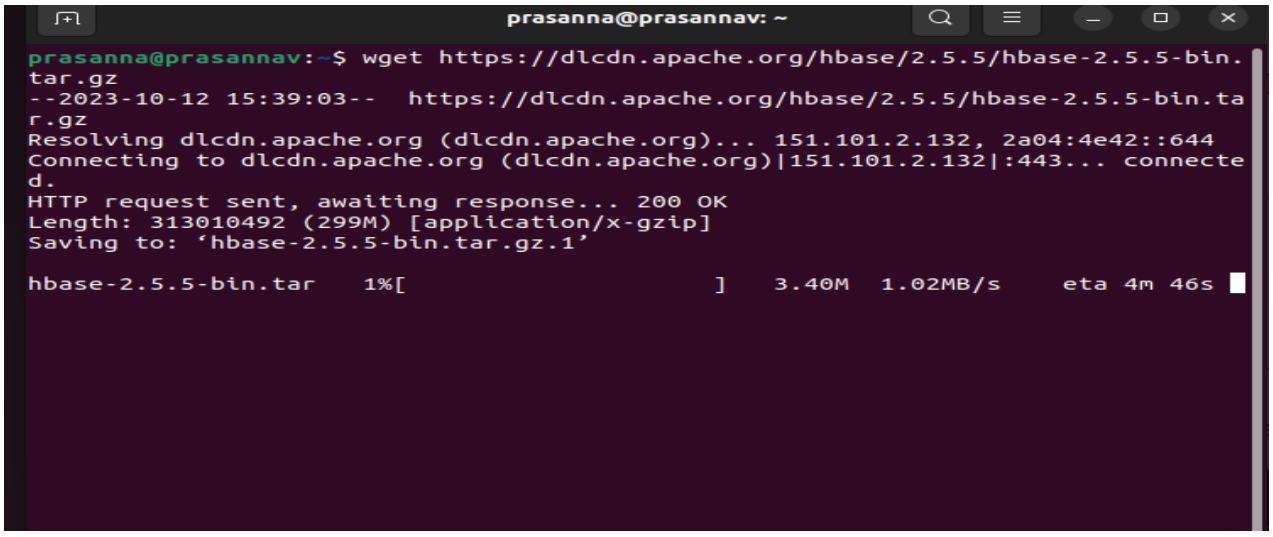


If any Error Occured While Execute this command , then java is not installed in your systemTo

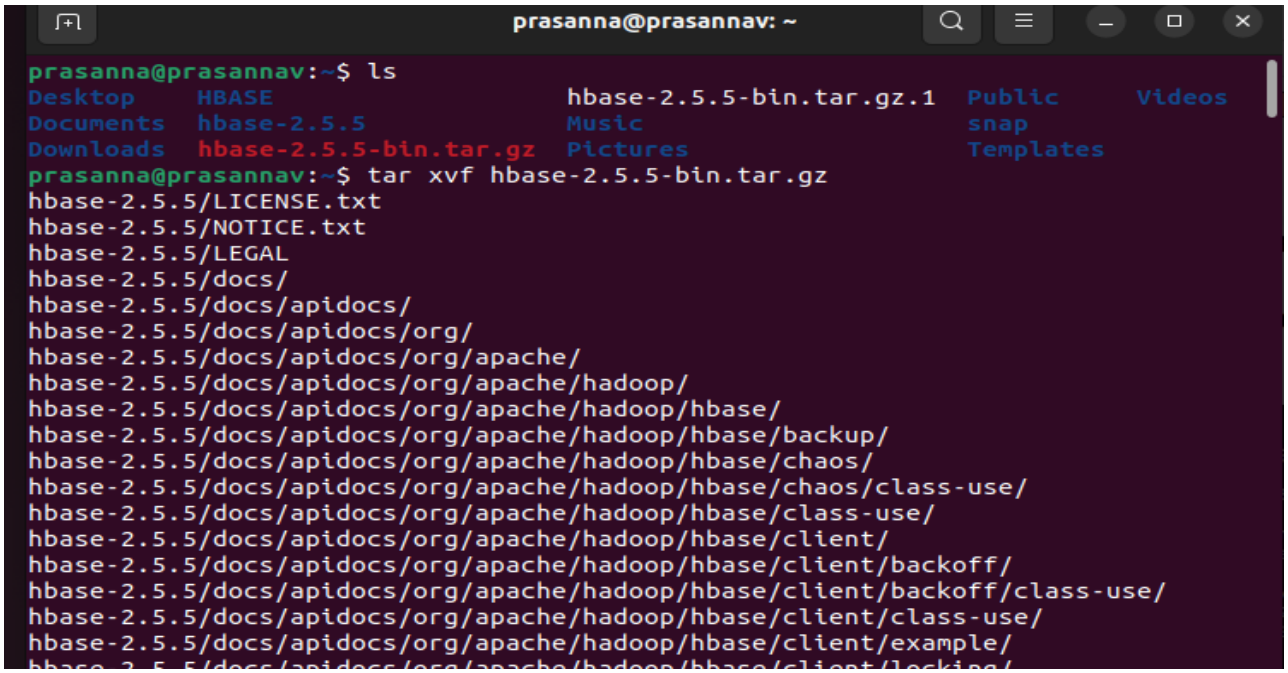
Install Java sudo apt install openjdk-8-jdk -y

Step-2: Download Hbase

wget https://dlcdn.apache.org/hbase/2.5.5/hbase-2.5.5-bin.tar.gz

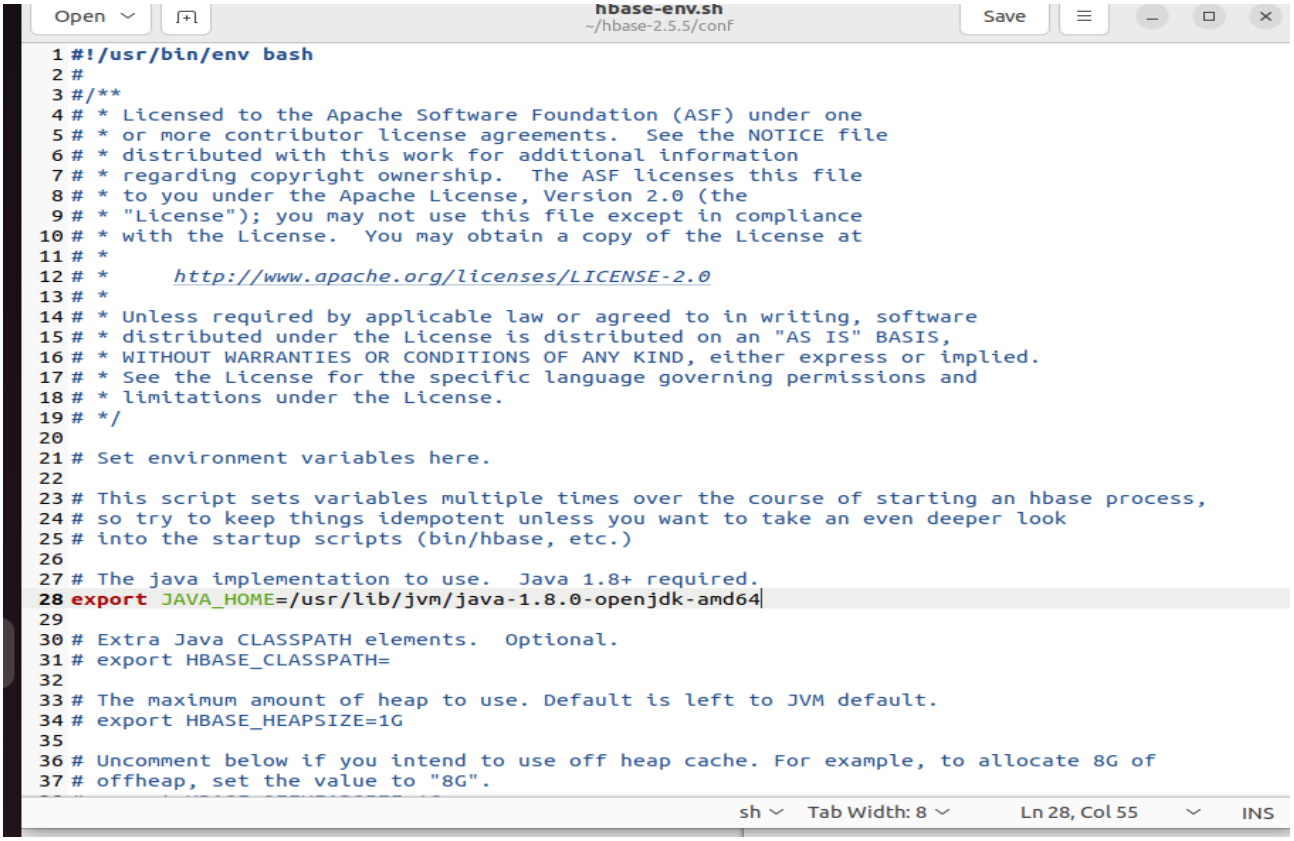


Step-3: Extract The hbase-2.5.5-bin.tar.gz file by using the command tar xvf hbase-2.5.5-bin.tar.gz



Step-4: goto hbase2.5.5/conf folder and open hbase-env.sh file

export JAVA_HOME=/usr/lib/jvm/java-1.8.0-openjdk-amd64



Step-5 : Edit .bashrc file

and then open .bashrc file and mention HBASE_HOME path as shown in below

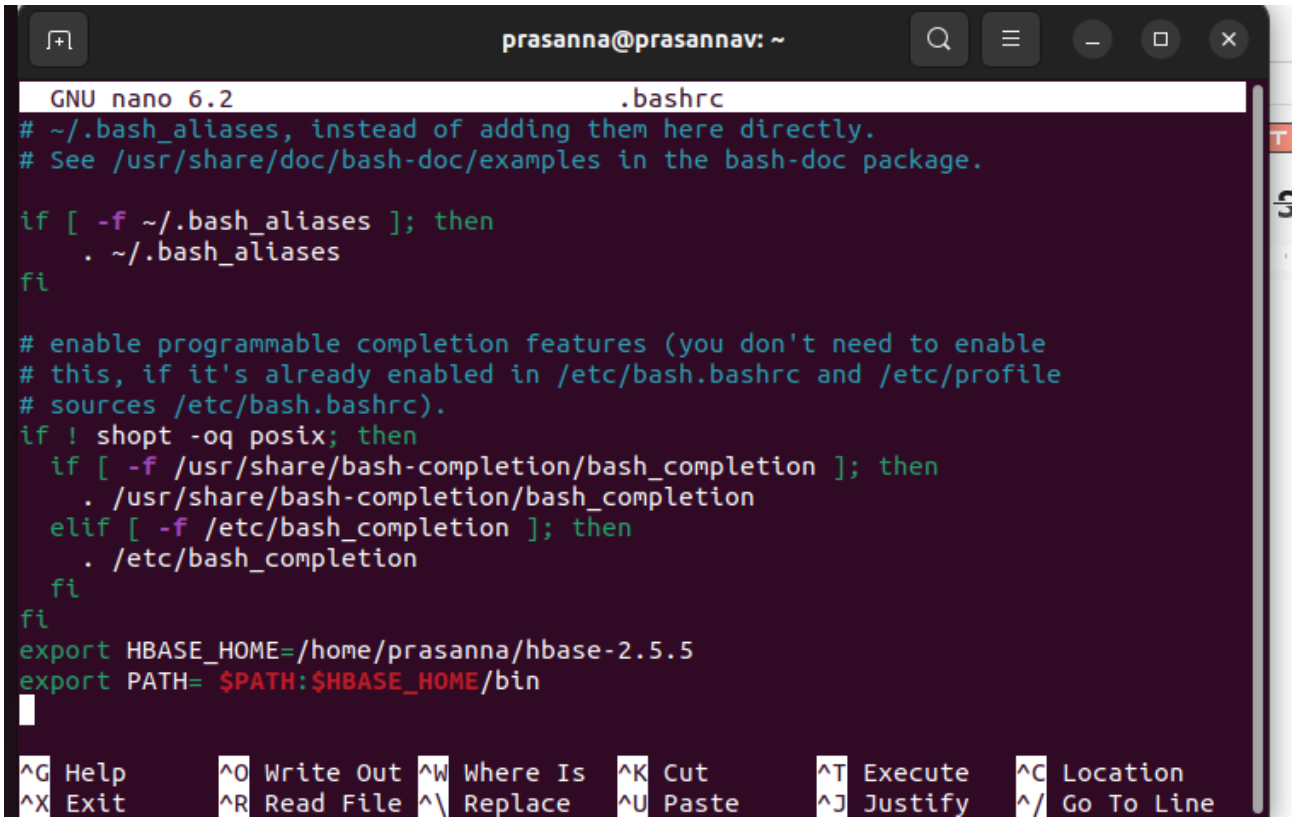
export HBASE_HOME=/home/prasanna/hbase-2.5.5

here you can change name according to your local machine name

eg : export HBASE_HOME=/home/<your_machine_name>/hbase-2.5.5

export PATH= $PATH:$HBASE_HOME/bin

Note:*make sure that the hbase-2.5.5 folderin home directory before setting HBASE_HOME path , if not then move the hbase-2.5.5 file to home directory*



Step-6 : Add properties in the hbase-site.xml

gedit hbase-site,.xml

put the below property between the <configuratio></configuration> tag

<property>
<name>hbase.rootdir</name>
<value>file:///home/prasanna/HBASE/hbase</value>
</property>
<property>
<name>hbase.zookeeper.property.dataDir</name>
<value>/home/prasanna/HBASE/zookeeper</value>
</property>

Step-7: Goto To /etc/ folderand run the following command and configure

$getit hosts

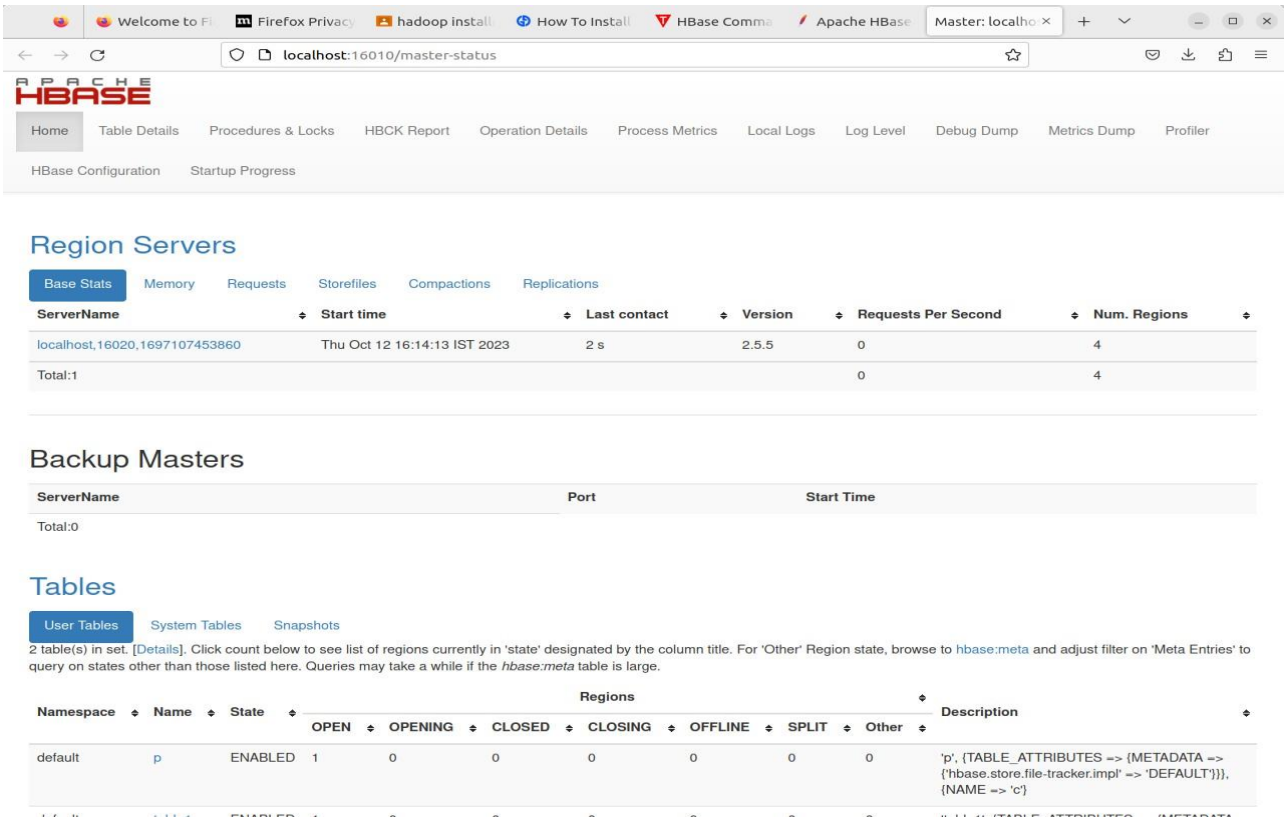
change in line no-2 by default the ip is 127.0.1.1
change it to 127.0.0.1 in second line only
step-8: starting hbase
goto hbase-2.5.5/bin folder

```
prasanna@prasannav:~$ cd hbase-2.5.5/bin/
prasanna@prasannav:~/hbase-2.5.5/bin$ ./start-hbase.sh
running master, logging to /home/prasanna/hbase-2.5.5/logs/hbase-prasanna-master
-prasannav.out
prasanna@prasannav:~/hbase-2.5.5/bin$
```

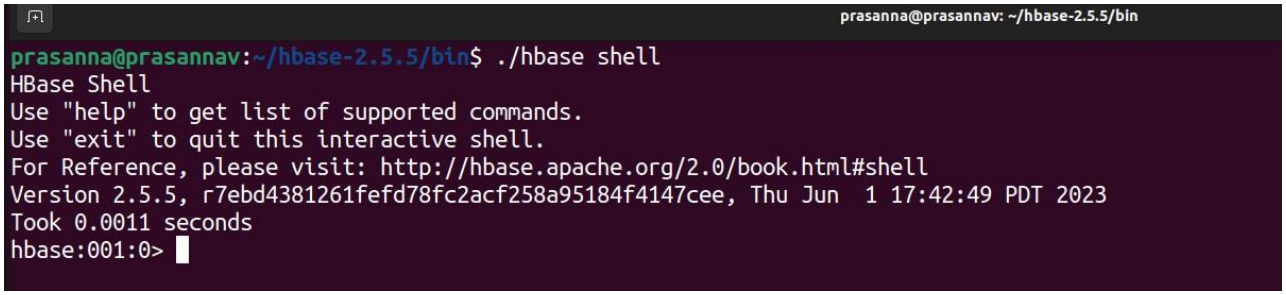
After this run jps command to ensure that hbase is running

```
prasanna@prasannav:~/hbase-2.5.5/bin$ jps
5126 Jps
4729 HMaster
prasanna@prasannav:~/hbase-2.5.5/bin$
```

run http://localhost:16010 ***to see hbase web UI***



Step-9: accessing hbase shell by running ./hbase shell command



RESULT:

Thus, the HBase was successfully installed on Ubuntu 18.04.

POSSIBLE VIVA QUESTIONS:

Pre-Viva Questions:

1. What are the prerequisites for installing HBase on a cluster?
2. What are the key configuration files for HBase, and what settings do you need to configure in them?
3. List the process of setting up the HBase Metastore. What are the key steps involved?
4. What are the steps to start HBase after installation?
5. How do you verify that HBase is installed and running correctly?

Post-Viva Questions:

1. What challenges did you encounter during the HBase installation and configuration? How did you resolve them?
2. How did you test the functionality of your HBase installation? What methods did you use to ensure everything was working correctly?
3. Mention any performance optimizations you made post-installation.
4. What tools or methods did you use to monitor HBase performance and health? How did you handle any issues that arose?
5. What are the steps for installing Apache Thrift on a system?

EXPT.NO.6(b)	HBase, Installing thrift along with Practice examples

AIM:

To Install HBase on Ubuntu 18.04 HBase in Standalone Mod

EXAMPLE:

1) To create Table

syntax:

create ‘Table_Name’,’col_fam_1’,’col_fam_1’,..... ’col_fam-n’

code :

create 'aamec','dept','year'

```
hbase:007:0> create 'aamec','dept','year'
2023-10-13 12:04:54,143 INFO  [main] client.HBaseAdmin (HBaseAdmin.java:postOper
ationResult(3591)) - Operation: CREATE, Table Name: default:aamec, procId: 100 c
ompleted
Created table aamec
Took 0.6840 seconds
=> Hbase::Table - aamec
hbase:008:0>
```

2) List All Tables

code :

list

```
hbase:010:0> list
TABLE
aamec
amazon
college
prasanna
table_name
5 row(s)
Took 0.0133 seconds
=> ["aamec", "amazon", "college", "prasanna", "table_name"]
hbase:011:0>
```

3) insert data

syntax:

put ‘table_name’,’row_key’,’column_family:attribute’,’value’

here row_key is a unique key to retrive data

code :

this data will enter data into the dept column family

put 'aamec','cse','dept:studentname','prasanna'
put 'aamec','cse','dept:year','third'
put 'aamec','cse','dept:section','A'

```
hbase:008:0> put 'aamec','cse','dept:studentname','prasanna'
Took 0.0240 seconds
hbase:009:0> put 'aamec','cse','dept:year','third'
Took 0.0072 seconds
hbase:010:0> put 'aamec','cse','dept:section','A'
Took 0.0342 seconds
hbase:011:0>
```

This data will enter data into the year column family

put 'aamec','cse','year:joinedyear','2021'
put 'aamec','cse','year:finishingyear','2025'

```
hbase:011:0> put 'aamec','cse','year:joinedyear','2021'
Took 0.0739 seconds
hbase:012:0> put 'aamec','cse','year:finishingyear','2025'
Took 0.0411 seconds
hbase:013:0>
```

4) Scan Table

same as desc in RDBMS

syntax:

scan ‘table_name’

code:

scan ‘aamec’

```
hbase:022:0> scan 'aamec'
ROW                     COLUMN+CELL
 cse                    column=dept:section, timestamp=2023-10-13T12:14:26.734, value=A
 cse                    column=dept:studentname, timestamp=2023-10-13T12:13:11.914, value=pras
                        anna
 cse                    column=dept:year, timestamp=2023-10-13T12:13:41.018, value=third
 cse                    column=year:finishingyear, timestamp=2023-10-13T12:16:57.291, value=20
                        25
 cse                    column=year:joinedyear, timestamp=2023-10-13T12:16:41.876, value=2021
 it                     column=dept:section, timestamp=2023-10-13T12:20:40.016, value=A
 it                     column=dept:studentname, timestamp=2023-10-13T12:20:06.012, value=amba
                        lavanan
 it                     column=dept:year, timestamp=2023-10-13T12:20:19.978, value=third
 it                     column=year:finishingyear, timestamp=2023-10-13T12:21:02.654, value=20
                        25
 it                     column=year:joinedyear, timestamp=2023-10-13T12:20:53.094, value=2021
2 row(s)
Took 0.0810 seconds
hbase:023:0>
```

5) To get specific data

syntax:

get ‘table_name’,’row_key’,[optional column family: attribute]

code :

get ‘aamec’,’cse’

```
hbase:023:0> get 'aamec','cse'
COLUMN                          CELL
 dept:section                   timestamp=2023-10-13T12:14:26.734, value=A
 dept:studentname               timestamp=2023-10-13T12:13:11.914, value=prasanna
 dept:year                      timestamp=2023-10-13T12:13:41.018, value=third
 year:finishingyear             timestamp=2023-10-13T12:16:57.291, value=2025
 year:joinedyear                timestamp=2023-10-13T12:16:41.876, value=2021
1 row(s)
Took 0.0908 seconds
```

6.update table value

The same put command is used to update the table value ,if the row key is aldready present in the database then it will update data according to the value ,if not present the it will create new row with the given row key

```
hbase:025:0> put 'aamec','cse','dept:section','B'
Took 0.0134 seconds
hbase:026:0>
```

previously the value for the section in cse is A ,But after running this command the value will be changed into B

```
hbase:026:0> get 'aamec','cse'
COLUMN                          CELL
 dept:section                   timestamp=2023-10-13T12:30:57.010, value=B
 dept:studentname               timestamp=2023-10-13T12:13:11.914, value=prasanna
 dept:year                      timestamp=2023-10-13T12:13:41.018, value=third
 year:finishingyear             timestamp=2023-10-13T12:16:57.291, value=2025
 year:joinedyear                timestamp=2023-10-13T12:16:41.876, value=2021
1 row(s)
Took 0.0506 seconds
hbase:027:0>
```

7)To Delete Data

syntax:

delete ‘table_name’,’row_key’,’column_family:attribute’

code :

delete 'aamec','cse','year:joinedyear'

```
hbase:027:0> delete 'aamec','cse','year:joinedyear'
Took 0.0138 seconds
hbase:028:0> get 'aamec','cse'
COLUMN                          CELL
 dept:section                   timestamp=2023-10-13T12:30:57.010, value=B
 dept:studentname               timestamp=2023-10-13T12:13:11.914, value=prasanna
 dept:year                      timestamp=2023-10-13T12:13:41.018, value=third
 year:finishingyear             timestamp=2023-10-13T12:16:57.291, value=2025
1 row(s)
Took 0.0686 seconds
hbase:029:0>
```

8.De

lete Table

first we need to disable the table before dropping it

To Disable:

syntax:

disable ‘table_name’

code:

disable ‘aamec’

```
hbase:029:0> disable 'aamec'
2023-10-13 12:42:08,027 INFO  [main] client.HBaseAdmin (HBaseAdmin.java:rpcCall(926)) - Started
disable of aamec
2023-10-13 12:42:08,699 INFO  [main] client.HBaseAdmin (HBaseAdmin.java:postOperationResult(3591
)) - Operation: DISABLE, Table Name: default:aamec, procId: 106 completed
Took 0.7578 seconds
hbase:030:0>
```

RESULT:

Thus, the HBase was successfully installed with an example on Ubuntu 18.04.

EXPT.NO.7	Practice importing and exporting data from various databases.

AIM:

To import or export, the order of columns in MySQL and Hive

Pre-requisite

Hadoop and Java
MySQL
Hive
SQOOP

Step 1:To start hdfs

```
ambal2@Ubuntu:~$ start-all.sh
WARNING: Attempting to start all Apache Hadoop daemons as ambal2 in 10 seconds.
WARNING: This is not a recommended production deployment configuration.
WARNING: Use CTRL-C to abort.
Starting namenodes on [localhost]
Starting datanodes
```

Step 2: MySQL Installation

sudo apt install mysql-server (use this command to install MySQL server)

```
root@Ubuntu:/home/ambal2# mysql
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 9
Server version: 8.0.34-0ubuntu0.22.04.1 (Ubuntu)

Copyright (c) 2000, 2023, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

COMMANDS:
~$ sudo su

After this enter your linux user password,then the root mode will be open here we don’tneed any authentication for mysql.

~root$ mysql

Creating user profiles and grant them permissions:

Mysql> CREATE USER ‘bigdata'@'localhost' IDENTIFIED BY ‘bigdata’;
Mysql>grant all privileges on *.* to bigdata@localhost;

Note: This step is not required if you just use the root user to make CRUDoperations in the MySQL

Mysql> CREATE USER ‘bigdata’@’127.0.0.1' IDENTIFIED BY ‘ bigdata’;
Mysql>grant all privileges on *.* to bigdata@127.0.0.1;

Note: Here, *.* means that the user we create has all the privileges on all the tablesof all the databases.

Now, we have created user profiles which will be used to make CRUD operations in themysql

Step 3: Create a database and table and insert data.

Example:

create database Employe;

create table Employe.Emp(author_name varchar(65), total_no_of_articles int, phone_noint, address varchar(65));

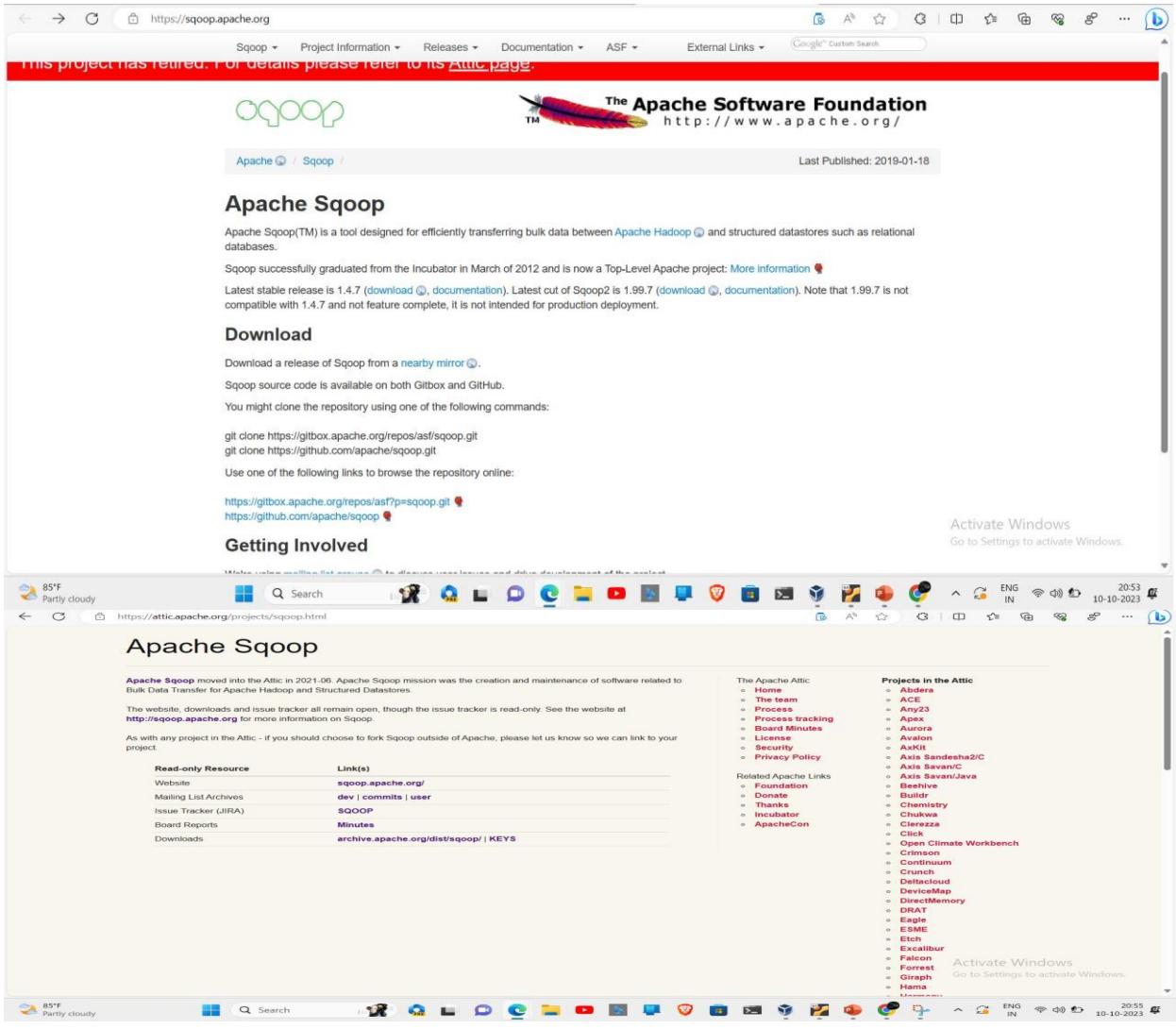
insert into Emp values(“Rohan”,10,123456789,”Lucknow”);

Step 3: Create a database and table in the hive where data should be imported.

create table geeks_hive_table(name string, total_articles int, phone_no int, address string)row format delimited fields terminated by ‘,’;

```
mysql> insert into dell values('inspiron',3505);
Query OK, 1 row affected (0.12 sec)

mysql> insert into dell values('alienware',5005);
Query OK, 1 row affected (0.03 sec)

mysql> insert into dell values('inspiron',3550);
Query OK, 1 row affected (0.02 sec)
```

Step 4: SQOOP INSTALLATION :



After downloading the sqoop , go to the directory where we downloaded the sqoopand then extract it using the following command :

$ tar -xvf sqoop-1.4.4.bin_____hadoop-2.0.4-alpha.tar.gz

Then enter into the super user : $ su

Next to move that to the usr/lib which requires a super user privilege

$ mv sqoop-1.4.4.bin_____hadoop-2.0.4-alpha /usr/lib/sqoop
Then exit : $ exit

Goto .bashrc: $ sudo nano .bashrc , and then add the following

export SQOOP_HOME=/usr/lib/sqoop

export PATH=$PATH:$SQOOP_HOME/bin

$ source ~/.bashrc
Then configure the sqoop, goto the directory of the config folder of sqoop_home and then move the contents of template file to the environment file.

$ cd $SQOOP_HOME/conf

$ mv sqoop-env-template.sh sqoop-env.sh

Then open the sqoop-environment file and then add the following,

export HADOOP_COMMON_HOME=/usr/local/Hadoop

export HADOOP_MAPRED_HOME=/usr/local/hadoop

Note : Here we add the path of the Hadoop libraries and files and it may different from the path which we mentioned here. So, add the Hadoop path based on your installation.

Step 5: Download and Configure mysql-connector-java :
We can download mysql-connector-java-5.1.30.tar.gz file from the following link.

Next, to extract the file and place it to the lib folder of sqoop
$ tar -zxf mysql-connector-java-5.1.30.tar.gz
$ su
$ cd mysql-connector-java-5.1.30
$ mv mysql-connector-java-5.1.30-bin.jar /usr/lib/sqoop/lib
Note : This is library file is very important don't skip this step because it contains the libraries to connect the mysql databases to jdbc.
Verify sqoop: sqoop-version Step 3: hive database Creation
hive> create database sqoop_example;
hive>use sqoop_example;
hive>create table sqoop(usr_name string,no_ops int,ops_names string);
Hive commands much more alike mysql commands. Here, we just create the structure to store the datawhich we want to import in hive.

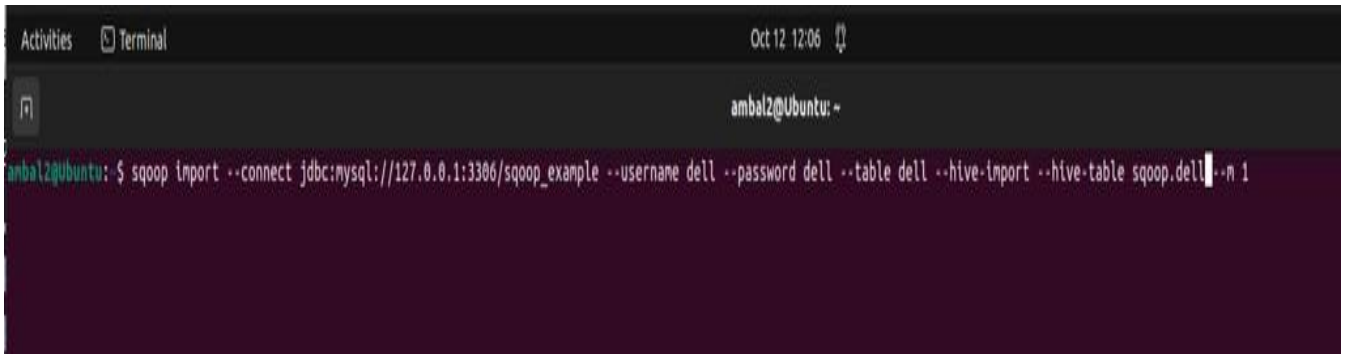
```
ambal2@Ubuntu: $ hive
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/ambal2/apache-hive-3.1.2-bin/lib/log4j-slf4j-impl-2.10.0.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/ambal2/hadoop-3.2.3/share/hadoop/common/lib/slf4j-log4j12-1.7.25.jar!/org/slf4j/impl/StaticLoggerBinder.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]
Hive Session ID = 1fb24ab2-af10-4d03-948f-73de05944193

Logging initialized using configuration in jar:file:/home/ambal2/apache-hive-3.1.2-bin/lib/hive-common-3.1.2.jar!/hive-log4j2.properties Async: true
Hive Session ID = 63f5f215-bf1c-4eb8-a6b5-01338cc55110
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 us
hive>
```

```
hive> show databases;
OK
default
sqoop
Time taken: 0.683 seconds, Fetched: 2 row(s)
hive> use sqoop;
OK
Time taken: 0.08 seconds
hive> show tables;
OK
bigdata
sqoop
Time taken: 0.148 seconds, Fetched: 2 row(s)
hive> create table dell(mdl_name string,mdl_num int);
OK
Time taken: 2.564 seconds
hive>
```

Hive>show database Hive>use sqoop
Hive>create table dell(mdl_name string,mdl_num int);

Step 6: Importing data from MySQL to hive :

sqoop import --connect \
jdbc:mysql://127.0.0.1:3306/database_name_in_mysql \
--username root --password cloudera \
--table table_name_in_mysql \
--hive-import --hive-table database_name_in_hive.table_name_in_hive \
--m 1



```
        Job Counters
                Launched map tasks=1
                Other local map tasks=1
                Total time spent by all maps in occupied slots (ms)=8912
                Total time spent by all reduces in occupied slots (ms)=0
                Total time spent by all map tasks (ms)=8912
                Total vcore-milliseconds taken by all map tasks=8912
                Total megabyte-milliseconds taken by all map tasks=9125888
        Map-Reduce Framework
                Map input records=3
                Map output records=3
                Input split bytes=87
                Spilled Records=0
                Failed Shuffles=0
                Merged Map outputs=0
                GC time elapsed (ms)=141
                CPU time spent (ms)=2800
                Physical memory (bytes) snapshot=223498240
                Virtual memory (bytes) snapshot=2542714880
                Total committed heap usage (bytes)=136839168
                Peak Map Physical memory (bytes)=223498240
                Peak Map Virtual memory (bytes)=2542714880
        File Input Format Counters
                Bytes Read=0
        File Output Format Counters
                Bytes Written=43
2023-10-12 12:15:07,009 INFO mapreduce.ImportJobBase: Transferred 43 bytes in 31.8907 seconds (1.3484 bytes/sec)
2023-10-12 12:15:07,047 INFO mapreduce.ImportJobBase: Retrieved 3 records.
Thu Oct 12 12:15:07 IST 2023 WARN: Establishing SSL connection without server's identity verification is not recommended. According
ished by default if explicit option isn't set. For compliance with existing applications not using SSL the verifyServerCertificate
 useSSL=false, or set useSSL=true and provide truststore for server certificate verification.
2023-10-12 12:15:07,188 INFO manager.SqlManager: Executing SQL statement: SELECT t.* FROM `dell` AS t LIMIT 1
2023-10-12 12:15:07,282 INFO hive.HiveImport: Loading uploaded data into Hive
2023-10-12 12:15:09,556 INFO hive.HiveImport: SLF4J: Class path contains multiple SLF4J bindings.
2023-10-12 12:15:09,557 INFO hive.HiveImport: SLF4J: Found binding in [jar:file:/home/ambal2/apache-hive-3.1.2-bin/lib/log4j-slf4j-
2023-10-12 12:15:09,557 INFO hive.HiveImport: SLF4J: Found binding in [jar:file:/home/ambal2/hadoop-3.2.3/share/hadoop/common/lib/s
2023-10-12 12:15:09,557 INFO hive.HiveImport: SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
2023-10-12 12:15:09,562 INFO hive.HiveImport: SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
```

OUTPUT:

```
ambal2@Ubuntu:~$ hive
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/ambal2/apache-hive-3.1.2-bin/lib/log4j-slf4j-impl-2.10.0.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/ambal2/hadoop-3.2.3/share/hadoop/common/lib/slf4j-log4j12-1.7.25.jar!/org/slf4j/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]
Hive Session ID = ca95a42a-a85e-4d00-948a-c435099df78f

Logging initialized using configuration in jar:file:/home/ambal2/apache-hive-3.1.2-bin/lib/hive-common-3.1.2.jar!/hive-log4j2.properties Async: true
Hive Session ID = a1776f23-c763-4313-a2c9-e3bc02cb423e
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
hive> show databases;
OK
default
sqoop
Time taken: 0.573 seconds, Fetched: 2 row(s)
hive> use sqoop;
OK
Time taken: 0.073 seconds
hive> select * from dell;
OK
inspiron        3505
alienware       5005
inspiron        3550
Time taken: 3.087 seconds, Fetched: 3 row(s)
hive>
```

RESULT:

Thus, the import and export, the order of columns in MySQL queries are exportedto hive successfully.

POSSIBLE VIVA QUESTIONS:

Pre-Viva Questions:

1. What are the common methods for importing data into a relational database? What tools or commands are typically used?
2. Mention the process of exporting data from a database.
3. What steps and commands are typically involved for exporting the database?
4. How do you handle data format compatibility when importing data into a database?
5. What are some common issues you might encounter during data import/export operations, and how would you resolve them?

Post-Viva Questions:

1. What were the main challenges you faced during the data import/export process, and how did you address them?
2. How did you verify the correctness and completeness of the data after importing or exporting? What methods did you use?
3. How did the optimizations improve the process after importing or exporting the data?
4. What tools or methods did you use to monitor the progress and performance of data import/export operations?
5. How did you address any issues identified during monitoring of the data?

EXPT.NO.8	Implement a program using Hive indexes

AIM:

To implement a program using Hive indexes.

PROCEDURE:

1. Creating an Index

Creating an index means creating a pointer on a particular column of a table. Its syntax is as follows:

CREATE INDEX index_name
ON TABLE base_table_name (col_name, ...) AS 'index.handler.class.name'
[WITH DEFERRED REBUILD]
[IDXPROPERTIES (property_name=property_value, ...)] [IN TABLE index_table_name]
[PARTITIONED BY (col_name, ...)] [
[ROW FORMAT ...] STORED AS ...
| STORED BY ...
]
[LOCATION hdfs_path] [TBLPROPERTIES (...)]

Example

Use the same employee table that we have used earlier with the fields Id, Name, Salary, Designation, and Dept. Create an index named index_salary on the salary column of the employee table.

The following query creates an index:

hive> CREATE INDEX inedx_salary ON TABLE employee(salary) AS 'org.apache.hadoop.hive.ql.index.compact.CompactIndexHandler';

2. Dropping an Index

The following syntax is used to drop an index: DROP INDEX <index_name> ON <table_name>

The following query drops an index named index_salary: hive> DROP INDEX index_salary ON employee;

The following query drops a view named as emp_30000: hive> DROP VIEW emp_30000;

RESULT:

Thus, the index is created and dropped using Hive Concept.

EXPT.NO.9	Implement a program using Hive Views

AIM:

To implement a program using Hive views

PROCEDURE:

1. Creating a View

Create a view at the time of executing a SELECT statement. The syntax is as follows:

CREATE VIEW [IF NOT EXISTS] view_name [(column_name [COMMENT column_comment], ...)

]

[COMMENT table_comment] AS SELECT ...

Consider the employee table as given below, with the fields Id, Name, Salary, Designation, and Dept. Generate a query to retrieve the employee details who earn a salary of more than Rs 30000.

Store the RESULT in a view named **emp_30000.**

ID	Name	Salary	Designation	Dept
1201	Gopal	45000	Technical manager	TP
1202	Manisha	45000	Proofreader	PR
1203	Masthanvali	40000	Technical writer	TP
1204	Krian	40000	Hr Admin	HR
1205	Kranthi	30000	Op Admin	Admin

The following query retrieves the employee details using the above scenario:

```
hive> CREATE VIEW emp_30000 AS
SELECT * FROM employee
WHERE salary>30000;
```

2. Dropping a View

Use the following syntax to drop a view: DROP VIEW view_name

RESULT:

Thus, the view is created and dropped using Hive Concept.

EXPT.NO.10	Implement a program using Hive External Table by accessing the external file created by Pig or any other tool.

AIM:

To implement a program using Hive external table by accessing the external file created by Pig or any other tool.

DESCRIPTION:

The external table allows us to create and access a table and a data externally. The **external** keyword is used to specify the external table, whereas the **location** keyword is used to determine the location of loaded data.As the table is external, the data is not present in the Hive directory. Therefore, if we try to drop the table, the metadata of the table will be deleted, but the data still exists.

PROCEDURE:

1. Creating a Table

For creating an external table, we use the flowing command:

```
>CREATE EXTERNAL TABLE student_external (
name string, class ARRAY,
gender_age STRUCT, subj_score MAP
)
COMMENT ' External student table' ROW FORMAT DELIMITED
FIELDS TERMINATED BY '|'
COLLECTION ITEMS TERMINATED BY ',' MAP KEYS TERMINATED BY ':'
STORED AS TEXTFILE;
```

LOCATION '/user/tables/students'; The external table has 4 columns-

Column	Type
name	String
class	String array
gender_age	Struct (to hold different data types in one structure)
subj_score	MAP(to hold subjects and their scores)

2. Load the data and populate it.

Loading the Data

To load the data, we use the following command:

```
>LOAD DATA LOCAL INPATH '/home/Hadoop/student.txt' OVERWRITE INTO TABLE
student_external
```

To import data from a CSV file into an external table.

Step 1: Prepare the Data File

1. Create a CSV file titled „countries.csv“: sudo nano countries.csv

2. For each country in the list, write a row number, the country‟s name, its capital city, and its population in millions:

1,USA,Washington,328 2,France,Paris,67 3,Spain,Madrid,47 4,Russia,Moscow,145 5,Indonesia,Jakarta,267 6,Nigeria,Abuja,196

3. Save the file and make a note of its location.

Step 2: Import the File to HDFS

1. Create an HDFS directory. You will use this directory as an HDFS location of the file you created.

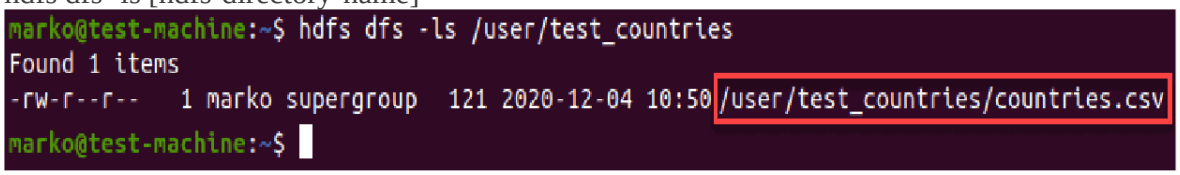
hdfs dfs -mkdir [hdfs-directory-name]

2. Import the CSV file into HDFS:

hdfs dfs -put [original-file-location] [hdfs-directory-name]

3. Use the -ls command to verify that the file is in the HDFS folder:

hdfs dfs -ls [hdfs-directory-name]



The output displays all the files currently in the directory.

Step 3: Create an External Table

1. After you import the data file to HDFS, initiate Hive and use the syntax to create an external table.

2. To verify that the external table creation was successful, type:

select * from [external-table-name];

The output should list the data from the CSV file you imported into the table:

3. If you wish to create a managed table using the data from an external table, type:

create table if not exists [managed-table-name](

[column1-name] [column1-type], [column2-name] [var2-name], …) comment '[comment]';

4. Next, import the data from the external table:

insert overwrite table [managed-table-name] select * from [external-table-name];

5. Verify that the data is successfully inserted into the managed table.

select * from [managed-table-name];

RESULT:

Thus, a program using Hive external table by accessing the external file created by Pig or any other tool is implemented.

EXPT.NO.11	Program using Hive scripts and aggregate functions

AIM:

To write a program using Hive scripts and aggregate functions.

PROCEDURE:

Aggregate Functions

To fetch the SUM of an employee of an employee

jdbc:hive2://> select sum(salary) from employee;

jdbc:hive2://> select sum(distinct salary) from employee;

jdbc:hive2://> select age,sum(salary) from employee group by age;

To fetch the maximum salary of an employee.

hive> select max(Salary) from employee_data;

To fetch the minimum salary of an employee.

hive> select min(Salary) from employee_data;

To fetch the minimum salary of an employee.

hive> select min(Salary) from employee_data;

To fetch the standard deviation salary of an employee.

hive://> select stddev_samp(salary) from employee_data;

To fetch the average salary of an employee.

hive://>select avg(salary) from employee group by age; hive://>select avg(distinct salary) from employee group by age; hive://> select age,avg(salary) from employee group by age;

To fetch the count of salary of an employee.

dbc:hive2://> select count(*) from employee;

jdbc:hive2://> select count(salary) from employee;

select count(distinct gender, salary) from employee;

HIVE SCRIPTS AND ITS EXECUTION

Step 1: Writing a Hive script.

To write the Hive Script the file should be saved with .sql extension. Open a terminal and give the following command to create a Hive Script.

Command:

sudo gedit sample.sql

On executing the above command, it will open this file in gedit

1. Create the Data to store into the Table.

To load the data into the table first we need to create an input file which contains the records that need to be inserted in the table.

Let us create an input file. Command: sudo gedit input.txt

Edit the contents into it you want to store into table

2. Creating the Table in Hive:

Command: create table product (productid: int, productname: string, price: float, category: string) rows format delimited fields terminated by „„ ;

Here, product is the table name and { productid, productname, price, category} are the columns of this table.

Fields terminated by „„ indicate that the columns in the input file are separated by the symbol „„. By default the records in the input file are separated by a new line.

3. Describing the Table:

Command: describe product

4. Retrieving the Data:

To retrieve the data, the select command is used. Command: Select * from product;

The above command is used to retrieve the value of all the columns present in the table. Now, we are done with writing the Hive script. The file sample.sql can now be saved.

Step 2: Running the Hive Script

The following is the command to run the Hive script:

Command: hive –f /home/sample.sql

While executing the script, make sure that the entire path of the location of the Script file is present.

RESULT:

Thus, a program using Hive scripts and aggregate functions is implemented successfully.

POSSIBLE VIVA QUESTIONS:

Pre-Viva Questions:

1. What is the purpose of using an external table in Hive, and how does it differ from a managed table?
2. How do you define an external table in Hive? What are the key components of the CREATE EXTERNAL TABLE statement?
3. What considerations should be made when creating an external table that will access files generated by Pig?
4. How you can ensure data compatibility between Pig and Hive when using external tables.
5. What steps are involved in creating an external table in Hive that accesses a file created by another tool, such as a CSV file generated by a script?

Post-Viva Questions:

1. What were the main issues encountered during the creation and use of an external table accessing files from Pig, and how were they resolved?
2. How did you verify that the data in the Hive external table correctly reflects the data in the external file?
3. Discuss any performance considerations you encountered when working with external tables in Hive. How did you address these considerations?
4. What steps did you take to ensure compatibility with the Hive external table?
5. What tools or methods did you use to monitor and troubleshoot issues with Hive external tables accessing external files?

INDUSTRY APPLICATION BASED EXPERIMENTS

EXPT.NO.1	Visualize Data Using Basic Plotting Techniques

AIM:

To create an application that takes the Visualize Data Using Basic Plotting Techniques.

PROCEDURE:

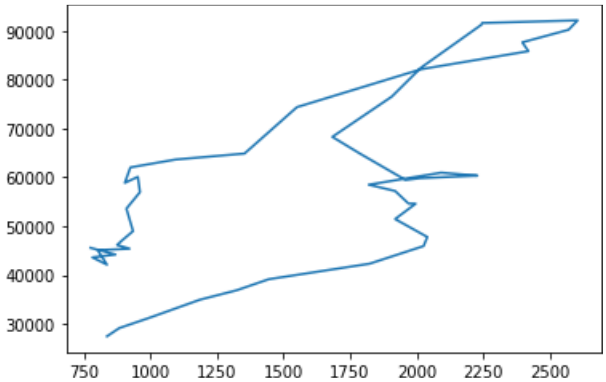
- Download any of the dataset from datasite in CSV format.
- Load the dataset.
- Draw different types of graph for the dataset.

SAMPLE CODING:

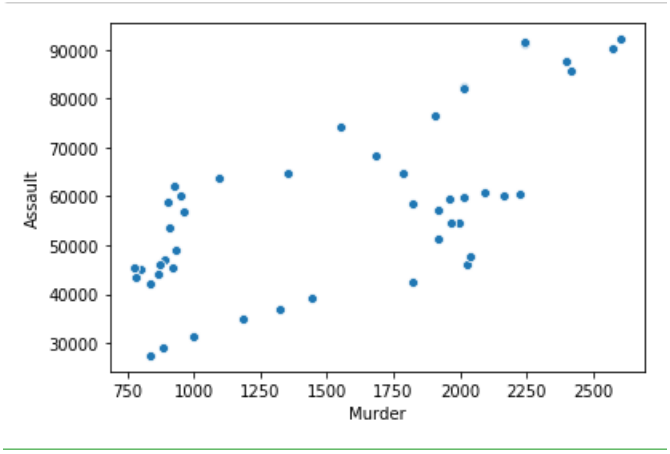
```
import pandas as pb
import matplotlib.pyplot as plt
import seaborn as sns
crime=pb.read_csv('crime.csv')
crime
```

	Year	Population	Murder	Rape	Robbery	Assault	Burglary	CarTheft
0	1965	18073000	836	2320	28182	27464	183443	58452
1	1966	18258000	882	2439	30098	29142	196127	64368
2	1967	18336000	996	2665	40202	31261	219157	83775
3	1968	18113000	1185	2527	59857	34946	250918	104877
4	1969	18321000	1324	2902	64754	36890	248477	115400
5	1970	18190740	1444	2875	81149	39145	267474	125674
6	1971	18391000	1823	3225	97682	42318	273704	127658
7	1972	18366000	2026	4199	86391	45926	239886	105081
8	1973	18265000	2040	4852	80795	47781	246246	112328
9	1974	18111000	1919	5240	86814	51454	271824	104095
10	1975	18120000	1996	5099	93499	54593	301996	116274
11	1976	18084000	1969	4663	95718	54638	318919	133504
12	1977	17924000	1919	5272	84703	57193	309735	133669
13	1978	17748000	1820	5168	83785	58484	292956	119264
14	1979	17649000	2092	5394	93471	60949	308302	124343
15	1980	17506690	2228	5405	112273	60329	360925	133041
16	1981	17594000	2166	5479	120344	60189	350422	136849
17	1982	17659000	2013	5159	107843	59818	295245	137880
18	1983	17667000	1958	5296	94783	59452	249115	127861
19	1984	17735000	1786	5599	89900	64872	222956	115392
20	1985	17783000	1683	5706	89706	68270	219633	106537

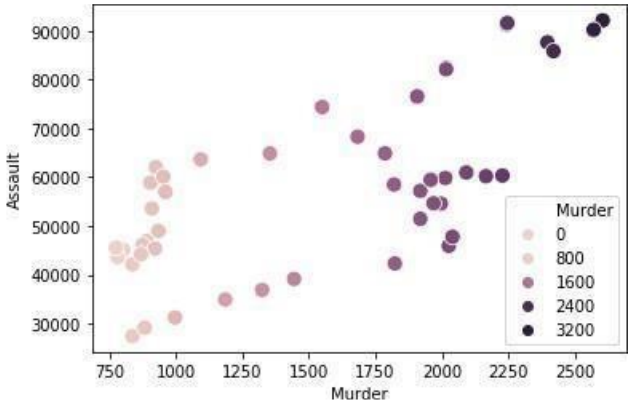
```
plt.plot(crime.Murder,crime.Assault);
```



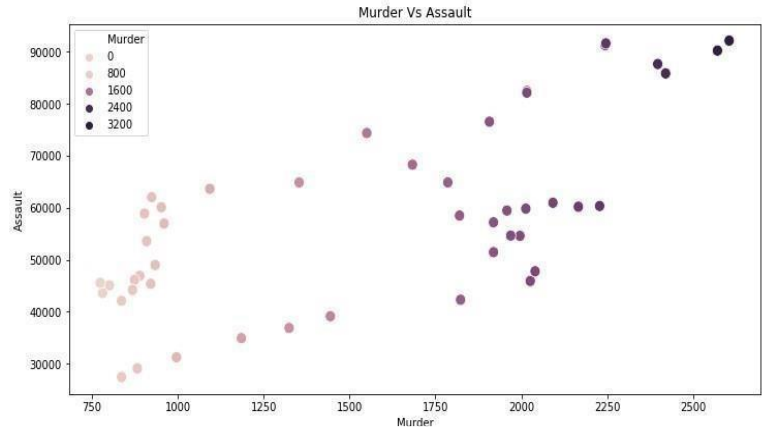

```
import seaborn as sns

sns.scatterplot(crime.Murder,crime.Assault);
```



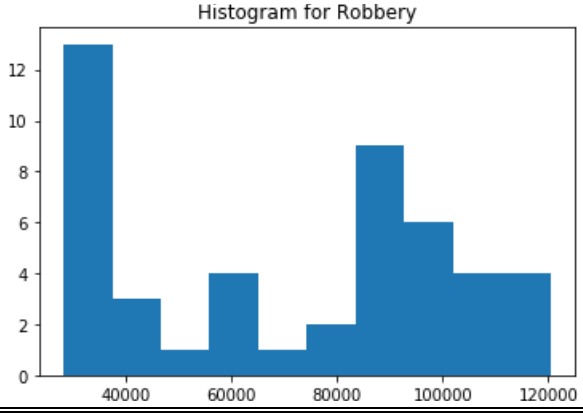

```
sns.scatterplot(crime.Murder,crime.Assault,hue=crime.Murder,s=100);
```



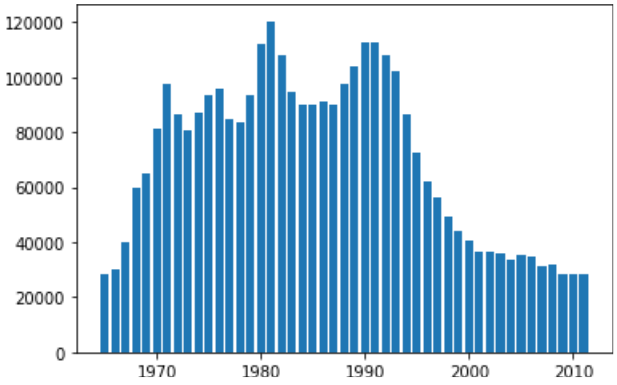

```
plt.figure(figsize=(12,6))
plt.title('Murder Vs Assault')
sns.scatterplot(crime.Murder,crime.Assault,hue=crime.Murder,s=100);
```



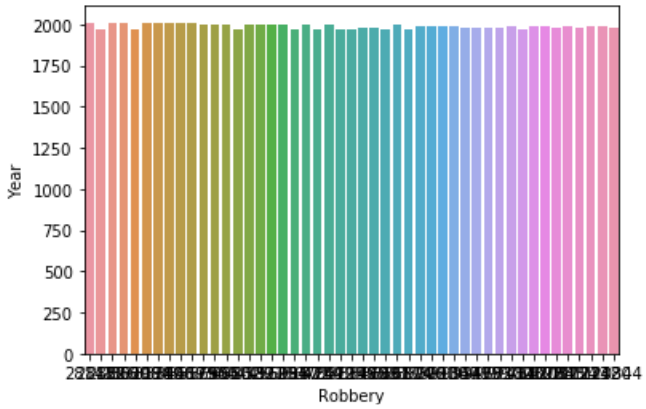

```
plt.title('Histogram for Robbery')

plt.hist(crime.Robbery);
```



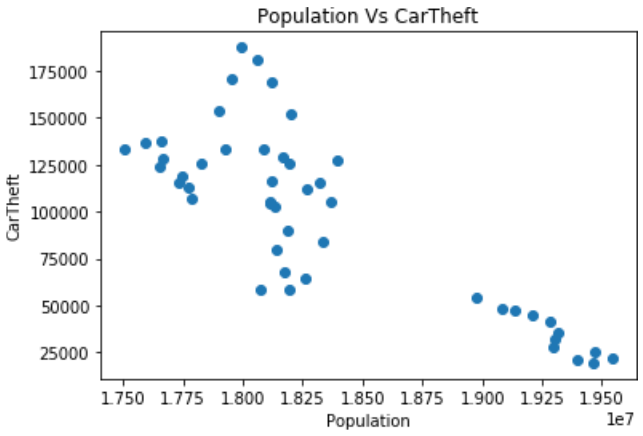
plt.bar(crime_bar.index,crime_bar.Robbery);



sns.barplot('Robbery','Year',data=crime);




```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
data=pd.read_csv('crime.csv')
x=data.Population
y=data.CarTheft
plt.scatter(x,y)
plt.xlabel('Population')
plt.ylabel('CarTheft')
plt.title('Population Vs CarTheft') plt.show();
```



RESULT:

Thus, the application to visualize the dataset using Python is completed successfully.

EXPT.NO.2	Implement Clustering Techniques Using SPARK.

AIM:

To create a clustering using SPARK.

PROGRAM:

```
# Loads data.
dataset = spark.read.format("libsvm").load("data/mllib/sample_kmeans_data.txt") # Trains a k-means model.
kmeans = KMeans().setK(2).setSeed(1) model = kmeans.fit(dataset)
# Evaluate clustering by computing Within Set Sum of Squared Errors. wssse = model.computeCost(dataset)
print("Within Set Sum of Squared Errors = " + str(wssse)) # Shows the RESULT.
centers = model.clusterCenters() print("Cluster Centers: ")
for center in centers:
print(center)
```

OUTPUT:

Cluster centers

RESULT:

Thus, the program t**o** create a clustering using SPARK**.**