

Exp.No: 1**Downloading and installing Hadoop, Understanding different Hadoop modes, Startup scripts, Configuration files.****AIM:**

To Download and install Hadoop, Understanding different Hadoop modes, Startup scripts, Configuration files.

Procedure:**Step 1 : Install Java Development Kit**

The default Ubuntu repositories contain Java 8 and Java 11 both. But, Install Java 8 because hive only works on this version. Use the following command to install it.

```
$sudo apt update&&sudo apt install openjdk-8-jdk
```

Step 2 : Verify the Java version

Once installed, verify the installed version of Java with the following command: \$

java -version Output:

```
srimathy@srimathy-VirtualBox:~$ java --version
openjdk 11.0.24 2024-07-16
OpenJDK Runtime Environment (build 11.0.24+8-post-Ubuntu-1ubuntu324.04.1)
OpenJDK 64-Bit Server VM (build 11.0.24+8-post-Ubuntu-1ubuntu324.04.1, mixed mode, sharing)
srimathy@srimathy-VirtualBox:~$
```

Step 3: Install SSH

SSH (Secure Shell) installation is vital for Hadoop as it enables secure communication between nodes in the Hadoop cluster. This ensures data integrity, confidentiality, and allows for efficient distributed processing of data across the cluster. **\$sudo apt install ssh**

Step 4 : Create the hadoop user :

All the Hadoop components will run as the user that you create for Apache Hadoop, and the user will also be used for logging in to Hadoop's web interface. Run the command to create user and set password:

```
$ sudo adduser hadoop
```

Step 5 : Switch user

Switch to the newly created hadoop user:

```
$ su - hadoop
```

Step 6 : Configure SSH

Now configure password-less SSH access for the newly created hadoop user, so didn't enter the key to save file and passphrase. Generate an SSH keypair (generate Public and Private Key Pairs)first

```
$ ssh-keygen -t rsa
```

Step 7 : Set permissions :

Next, append the generated public keys from id_rsa.pub to authorized_keys and set proper permission:

```
$ cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
```

```
$ chmod 640 ~/.ssh/authorized_keys
```

Step 8 : SSH to the localhost

Next, verify the password less SSH authentication with the following command:

```
$ ssh localhost
```

You will be asked to authenticate hosts by adding RSA keys to known hosts. Type yes and hit Enter to authenticate the localhost:

Step 9 : Switch user

Again switch to hadoop. So, First, change the user to hadoop with the following command: \$ su-hadoop

Step 10 : Install hadoop

Next, download the latest version of Hadoop using the wget command:

```
$ wget https://downloads.apache.org/hadoop/common/hadoop-3.3.6/hadoop-3.3.6.tar.gz
```

Once downloaded, extract the downloaded file:

```
$ tar -xvzf hadoop-3.3.6.tar.gz
```

Next, rename the extracted directory to hadoop:

```
$ mv hadoop-3.3.6 hadoop
```

```
srimathy@srimathy-VirtualBox:~$ ls
apache-hive-3.1.3-bin          pig_1726303616365.log  pig_1726756386892.log
apache-hive-3.1.3-bin.tar.gz    pig_1726750475504.log  pig_1726756477847.log
demo_pig.pig                   pig_1726750699679.log  process_data.py
Desktop                         pig_1726751227654.log  Public
Documents                       pig_1726751435545.log  reducer.py
Downloads                       pig_1726753894589.log  reducer.py
emp.json                        pig_1726753946547.log  sample.txt
hadoop                          pig_1726754101784.log  snap
hadoop-3.3.6.tar.gz            pig_1726754461335.log  Templates
maper.py                         pig_1726754718301.log  udf_example.pig
mapper.py                        pig_1726755168443.log  upper_case.py
Music                            pig_1726755203738.log  uppercase_udf.py
myenv                            pig_1726755403570.log  Videos
Pictures                         pig_1726755578361.log  weather_data.txt
pig-0.16.0                       pig_1726755592192.log  word_count.txt
pig-0.16.0.tar.gz              pig_1726756208183.log
srimathy@srimathy-VirtualBox:~$
```

Next, you will need to configure Hadoop and Java Environment Variables on your system. Open the ~/.bashrc file in your favorite text editor. Use nano editior , to pasting the code we use ctrl+shift+v for saving the file ctrl+x and ctrl+y ,then hit enter:

Next, you will need to configure Hadoop and Java Environment Variables on your system.

Open the `~/.bashrc` file in your favorite text editor:

\$ nano ~/.bashrc

Append the below lines to file.

```
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
export HADOOP_HOME=/home/hadoop/hadoop
export HADOOP_INSTALL=$HADOOP_HOME
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME
export HADOOP_YARN_HOME=$HADOOP_HOME
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
export PATH=$PATH:$HADOOP_HOME/sbin:$HADOOP_HOME/bin
export HADOOP_OPTS="-Djava.library.path=$HADOOP_HOME/lib/native"
```

Save and close the file. Then, activate the environment variables with the following command:

s\$ source ~/.bashrc

Next, open the Hadoop environment variable file: **\$ nano**

\$HADOOP_HOME/etc/hadoop/hadoop-env.sh

Search for the “`export JAVA_HOME`” and configure it.

`JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64`

```
File Edit View Search Terminal Help
GNU nano 7.2 /home/hadoop/hadoop/etc/hadoop/hadoop-env.sh *
## Precedence rules:
## {yarn-env.sh|hdfs-env.sh} > hadoop-env.sh > hard-coded defaults
## {YARN_xyz|HDFS_xyz} > HADOOP_xyz > hard-coded defaults
## 

# Many of the options here are built from the perspective that users
# may want to provide OVERWRITING values on the command line.
# For example:
#
JAVA_HOME=/usr/lib/jvm/java-11-openjdk-amd64
#
# Therefore, the vast majority (BUT NOT ALL!) of these defaults
# are configured for substitution and not append. If append
# is preferable, modify this file accordingly.

### 
# Generic settings for HADOOP
###

# Technically, the only required environment variable is JAVA_HOME.
# All others are optional. However, the defaults are probably not
# preferred. Many sites configure these options outside of Hadoop,
# such as in /etc/profile.d

# The java implementation to use. By default, this environment
# variable is REQUIRED on ALL platforms except OS X!
File Name to Write: /home/hadoop/hadoop/etc/hadoop/hadoop-env.sh
^C Help M-D DOS Format M-A Append M-B Backup File
^Q Cancel M-M Mac Format M-P Prepend M-T Browse
```

Save and close the file when you are finished.

Step 11 : Configuring Hadoop :

First, you will need to create the namenode and datanode directories inside the Hadoop user home directory. Run the following command to create both directories:

```
$ cd hadoop/
$mkdir -p ~/hadoopdata/hdfs/{namenode,datanode}
```

- Next, edit the core-site.xml file and update with your system hostname:

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

Change the following name as per your system hostname:

```
<configuration>
  <property>
    <name>fs.defaultFS</name>
    <value>hdfs://localhost:9000</value>
  </property>
</configuration>
```

Save and close the file.

Then, edit the hdfs-site.xml file:

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

- Change the NameNode and DataNode directory paths as shown below:

```
<configuration>
  <property>
    <name>dfs.replication</name>
    <value>1</value>
  </property>

  <property>
    <name>dfs.namenode.name.dir</name>
    <value>file:///home/hadoop/hadoopdata/hdfs/namenode</value>
  </property>

  <property>
    <name>dfs.datanode.data.dir</name>
    <value>file:///home/hadoop/hadoopdata/hdfs/datanode</value>
  </property>
</configuration>
```

- Then, edit the mapred-site.xml file:

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

- Make the following changes:

```
<configuration>
  <property>
    <name>yarn.app.mapreduce.am.env</name>
    <value>HADOOP_MAPRED_HOME=$HADOOP_HOME/home/hadoop/hadoop/bin/hadoop</value>
  </property>
  <property>
    <name>mapreduce.map.env</name>
    <value>HADOOP_MAPRED_HOME=$HADOOP_HOME/home/hadoop/hadoop/bin/hadoop</value>
  </property>
  <property>
    <name>mapreduce.reduce.env</name>
    <value>HADOOP_MAPRED_HOME=$HADOOP_HOME/home/hadoop/hadoop/bin/hadoop</value>
  </property>
</configuration>
```

- Then, edit the yarn-site.xml file:
\$ nano \$HADOOP_HOME/etc/hadoop/yarnsite.xml
- Make the following changes:

```
<configuration>
  <property>
    <name>yarn.nodemanager.aux-services</name>
    <value>mapreduce_shuffle</value>
  </property>
</configuration>
```

Save the file and close it.

Step 12 – Start Hadoop Cluster

Before starting the Hadoop cluster. You will need to format the Namenode as a hadoop user.

Run the following command to format the Hadoop Namenode:

\$ hdfs namenode –format

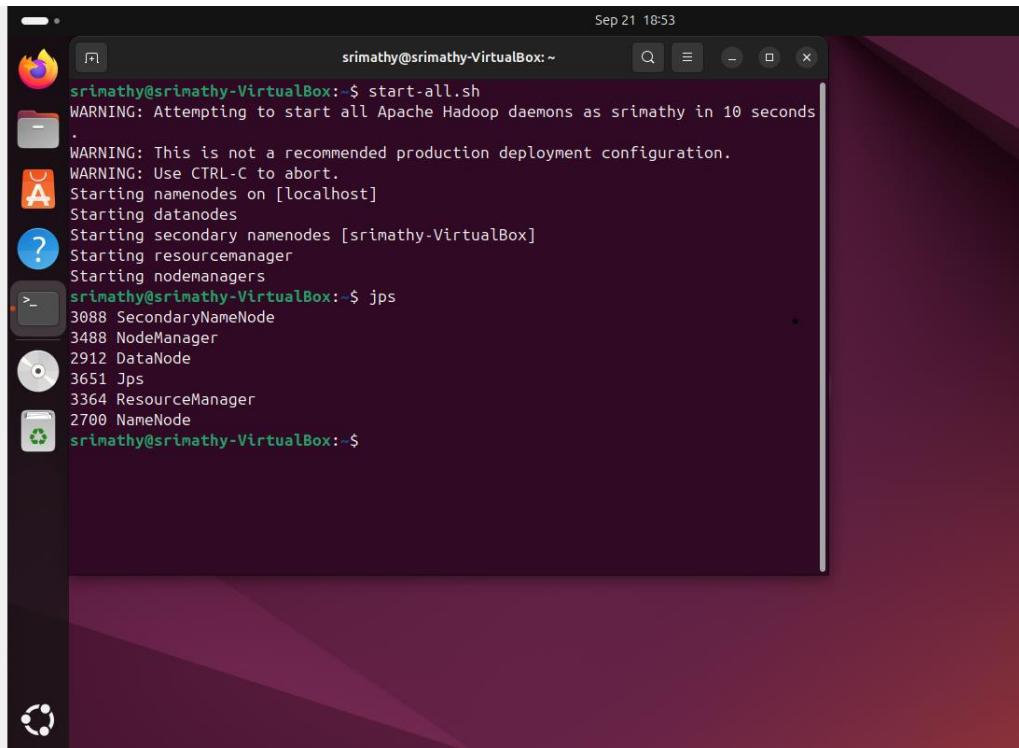
Once the namenode directory is successfully formatted with hdfs file system, you will see the message “Storage directory /home/hadoop/hadoopdata/hdfs/namenode has been successfully formatted “

Then start the Hadoop cluster with the following command.

\$ start-all.sh

You can now check the status of all Hadoop services using the jps command:

\$ jps

A screenshot of an Ubuntu desktop environment. In the foreground, a terminal window titled "srimathy@srimathy-VirtualBox: ~" is open, showing the output of the "start-all.sh" command. The terminal shows several "WARNING" messages about deployment configuration and starting various Hadoop daemons like namenodes, datanodes, secondary namenodes, resourcemanager, and nodemanagers. It also lists the process IDs (PIDs) for each started daemon. Below the terminal, the desktop background is visible with a dark purple gradient and some icons.

Step 13 – Access Hadoop Namenode and Resource Manager

- First we need to know our ipaddress, In Ubuntu we need to install net-tools to run ipconfig command,
If you installing net-tools for the first time switch to default user:
\$sudo apt install net-tools
- Then run ifconfig command to know our ip address: **ifconfig**

Here my ip address is 192.168.1.6.

- To access the Namenode, open your web browser and visit the URL <http://your-serverip:9870>.
- You should see the following screen:
<http://192.168.1.6:9870>

Overview 'localhost:9000' (✓active)

Started:	Mon Sep 02 10:43:55 +0530 2024
Version:	3.3.6, r1be78238728da9266a4f88195058f08fd012bf9c
Compiled:	Sun Jun 18 13:52:00 +0530 2023 by ubuntu from (HEAD detached at release-3.3.6-RC1)
Cluster ID:	CID-73012808-a614-4a4a-aa57-40b8fd6716fd
Block Pool ID:	BP-1797801860-127.0.1.1-172525249180

Summary

Security is off.
Safemode is off.
16 files and directories, 6 blocks (6 replicated blocks, 0 erasure coded block groups) = 22 total filesystem object(s).
Heap Memory used 77.73 MB of 221 MB Heap Memory. Max Heap Memory is 690 MB.
Non Heap Memory used 54.34 MB of 55.69 MB Committed Non Heap Memory. Max Non Heap Memory is <unbounded>.

Configured Capacity:	24.44 GB
Configured Remote Capacity:	0 B
DFS Used:	456 KB (0%)
Non DFS Used:	11.77 GB
DFS Remaining:	24.44 GB (100%)

To access Resource Manager, open your web browser and visit the URL <http://your-serverip:8088>. You should see the following screen: <http://192.168.16.8088>

hadoop

- Cluster
 - About
 - Nodes
 - Node Labels
 - Applications
 - NEW
 - NEW_SAVING
 - SUBMITTED
 - ASSIGNED
 - RUNNING
 - FINISHED
 - FAILED
 - KILLED
 - Scheduler
- Tools

Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running
0	0	0	0	0

Cluster Nodes Metrics

Active Nodes	Decommissioning Nodes	Decom
1	0	0

Scheduler Metrics

Scheduler Type	Scheduling Resource Type	Min
Capacity Scheduler	[memory-mb (unit=Mi), vcores]	<memory:1024, vCore:1>

Show 20 entries

ID	User	Name	Application Type	Application Tags	Queue	Application Priority	StartTime	LaunchTime	Finish
Showing 0 to 0 of 0 entries									

Step 14 – Verify the Hadoop Cluster

At this point, the Hadoop cluster is installed and configured. Next, we will create some directories in the HDFS filesystem to test the Hadoop.

Let's create some directories in the HDFS filesystem using the following command:

```
$ hdfsdfs -mkdir /test1
$ hdfsdfs -mkdir /logs
```

Next, run the following command to list the above directory:

Also, put some files to hadoop file system. For the example, putting log files from host machine to hadoop file system.

```
$ hdfs dfs -put /var/log/* /logs/
```

You can also verify the above files and directory in the Hadoop Namenode web interface.

Go to the web interface, click on the Utilities => Browse the file system. You should see your directories which you have created earlier in the following screen:

	Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
<input type="checkbox"/>	drwxr-xr-x	hadoop	supergroup	0 B	Sep 02 12:12	0	0 B	home
<input type="checkbox"/>	drwxrwxr-x	hadoop	supergroup	0 B	Sep 02 13:29	0	0 B	tmp
<input type="checkbox"/>	drwxr-xr-x	hadoop	supergroup	0 B	Sep 02 13:26	0	0 B	user
<input type="checkbox"/>	drwxr-xr-x	hadoop	supergroup	0 B	Sep 02 11:38	0	0 B	weatherdata
<input type="checkbox"/>	drwxr-xr-x	hadoop	supergroup	0 B	Sep 03 20:04	0	0 B	word_count_in_python

Step 15 – Stop Hadoop Cluster

To stop the Hadoop all services, run the following command:

```
$ stop-all.sh
```

```
srimathy@srimathy-VirtualBox:~$ jps
3088 SecondaryNameNode
3488 NodeManager
2912 DataNode
3651 Jps
3364 ResourceManager
2700 NameNode
srimathy@srimathy-VirtualBox:~$ stop-all.sh
WARNING: Stopping all Apache Hadoop daemons as srimathy in 10 seconds.
WARNING: Use CTRL-C to abort.
Stopping namenodes on [localhost]
Stopping datanodes
Stopping secondary namenodes [srimathy-VirtualBox]
Stopping nodemanagers
Stopping resourcemanager
srimathy@srimathy-VirtualBox:~$
```

Result:

The step-by-step installation and configuration of Hadoop on Ubutu linux system have been successfully completed.

Exp.No: 2**Run a basic Word Count Map Reduce program to understand Map Reduce Paradigm****AIM:**

To run a basic Word Count MapReduce program.

Procedure:**Step 1: Create Data File:**

Create a file named "word_count_data.txt" and populate it with text data that you wish to analyse.
Login with your hadoop user.

nano word_count.txt

Output: Type the below content in word_count.txt

The screenshot shows a terminal window with the nano 7.2 editor open. The file is named 'word_count.txt'. The content of the file is:

```
GNU nano 7.2                               word_count.txt
Hadoop Installation successfull
hadoop
hello
hi
hadoop
hello
```

The status bar at the bottom of the terminal window displays various keyboard shortcuts for nano editor commands, such as Help (^G), Write Out (^O), Where Is (^W), Cut (^K), Execute (^T), Location (^C), Exit (^X), Read File (^R), Replace (^|), Paste (^U), Justify (^J), and Go To Line (^/). The message '[Read 6 lines]' is also visible in the status bar.

Step 2: Mapper Logic - mapper.py:

Create a file named "mapper.py" to implement the logic for the mapper. The mapper will read input data from STDIN, split lines into words, and output each word with its count.

```

nano mapper.py
# Copy and paste the mapper.py code

#!/usr/bin/env python3
# import sys because we need to read and write data to STDIN and STDOUT
#!/usr/bin/python3
import sys
for line in sys.stdin:
    line = line.strip() # remove leading and trailing whitespace
    words = line.split() # split the line into words
    for word in words:
        print( '%s\t%s' % (word, 1))

```

Step 3: Reducer Logic - reducer.py:

Create a file named "reducer.py" to implement the logic for the reducer. The reducer will aggregate the occurrences of each word and generate the final output.

```

nano reducer.py
# Copy and paste the reducer.py code

```

reducer.py

```

#!/usr/bin/python3 from operator
import itemgetter import sys
current_word = None current_count
= 0 word = None for line in
sys.stdin: line = line.strip()
word, count = line.split('\t', 1)
try:
    count = int(count)
except ValueError:
    continue if current_word
== word: current_count
+= count else:
    if current_word:
        print( '%s\t%s' % (current_word, current_count))
    current_count = count current_word = word if
current_word == word: print( '%s\t%s' %
(current_word, current_count))

```

Step 4: Prepare Hadoop Environment:

Start the Hadoop daemons and create a directory in HDFS to store your data.

```
start-all.sh hdfsdfs -mkdir /word_count_in_python hdfsdfs -copyFromLocal
/path/to/word_count.txt/word_count_in_python
```

Step 6: Make Python Files Executable:

Give executable permissions to your mapper.py and reducer.py files.

```
chmod 777 mapper.py reducer.py
```

Step 7: Run Word Count using Hadoop Streaming:

Download the latest hadoop-streaming jar file and place it in a location you can easily access.

Then run the Word Count program using Hadoop Streaming.

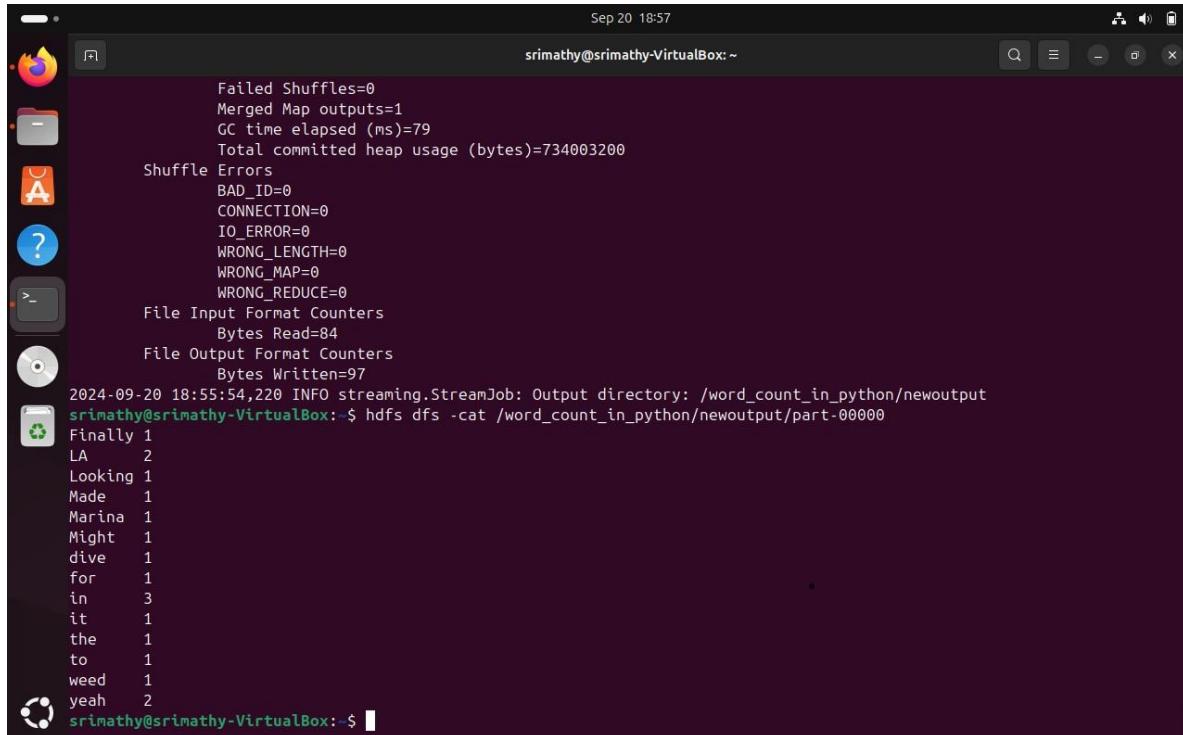
```
hadoop jar /path/to/hadoop-streaming-3.3.6.jar \
    -input
    /word_count_in_python/word_count_data.txt \
    -output /word_count_in_python/new_output \
    -mapper /path/to/mapper.py \
    -reducer /path/to/reducer.py
```

```
Sep 20 18:58
srimathy@srimathy-VirtualBox:~ 2024-09-20 18:55:27,420 INFO impl.MetricsConfig: Loaded properties from hadoop-metrics2.properties
2024-09-20 18:55:27,606 INFO impl.MetricsSystemImpl: Scheduled Metric snapshot period at 10 second(s).
2024-09-20 18:55:27,606 INFO impl.MetricsSystemImpl: JobTracker metrics system started
2024-09-20 18:55:27,631 WARN impl.MetricsSystemImpl: JobTracker metrics system already initialized!
2024-09-20 18:55:27,769 ERROR streaming.StreamJob: Error Launching job : Output directory hdfs://localhost:9000/word_count_in_python/new_output already exists
Streaming Command Failed!
srimathy@srimathy-VirtualBox:~$ hadoop jar /home/srimathy/hadoop/share/hadoop/tools/lib/hadoop-streaming-3.3.6.jar -input /word_count_python/word_count.txt -output /word_count_in_python/newoutput -mapper /home/srimathy/mapper.py -reducer /home/srimathy/reducer.py
2024-09-20 18:55:51,067 INFO impl.MetricsConfig: Loaded properties from hadoop-metrics2.properties
2024-09-20 18:55:51,229 INFO impl.MetricsSystemImpl: Scheduled Metric snapshot period at 10 second(s).
2024-09-20 18:55:51,229 INFO impl.MetricsSystemImpl: JobTracker metrics system started
2024-09-20 18:55:51,258 WARN impl.MetricsSystemImpl: JobTracker metrics system already initialized!
2024-09-20 18:55:51,668 INFO mapred.FileInputFormat: Total input files to process : 1
2024-09-20 18:55:51,767 INFO mapreduce.JobSubmitter: number of splits:1
2024-09-20 18:55:51,935 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_local986482978_0001
2024-09-20 18:55:51,935 INFO mapreduce.JobSubmitter: Executing with tokens: []
2024-09-20 18:55:52,120 INFO mapreduce.Job: The url to track the job: http://localhost:8080/
2024-09-20 18:55:52,122 INFO mapred.LocalJobRunner: OutputCommitter set in config null
2024-09-20 18:55:52,123 INFO mapreduce.Job: Running job: job_local986482978_0001
2024-09-20 18:55:52,133 INFO mapred.LocalJobRunner: OutputCommitter is org.apache.hadoop.mapred.FileOutputCommitter
2024-09-20 18:55:52,156 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 2
2024-09-20 18:55:52,156 INFO output.FileOutputCommitter: FileOutputCommitter skip cleanup _temporary folders under output directory:false, ignore cleanup failures: false
2024-09-20 18:55:52,242 INFO mapred.LocalJobRunner: Waiting for map tasks
2024-09-20 18:55:52,255 INFO mapred.LocalJobRunner: Starting task: attempt_local986482978_0001_m_000000_0
2024-09-20 18:55:52,329 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 2
2024-09-20 18:55:52,329 INFO output.FileOutputCommitter: FileOutputCommitter skip cleanup _temporary folders under output directory:false, ignore cleanup failures: false
2024-09-20 18:55:52,353 INFO mapred.Task: Using ResourceCalculatorProcessTree : [ ]
2024-09-20 18:55:52,375 INFO mapred.MapTask: Processing split: hdfs://localhost:9000/word_count_python/word_count.txt:0+84
```

Step 8: Check Output:

Check the output of the Word Count program in the specified HDFS output directory.

```
hdfs dfs -cat /word_count_in_python/new_output/part-00000
```



The screenshot shows a terminal window titled "srimathy@srimathy-VirtualBox: ~" running on a Linux desktop environment. The terminal displays the execution of a Hadoop streaming job and its word count results. The output includes various system metrics and the word counts for the input file "sherlock-holmes.txt".

```
Sep 20 18:57  
srimathy@srimathy-VirtualBox: ~  
Failed Shuffles=0  
Merged Map outputs=1  
GC time elapsed (ms)=79  
Total committed heap usage (bytes)=734003200  
Shuffle Errors  
BAD_ID=0  
CONNECTION=0  
IO_ERROR=0  
WRONG_LENGTH=0  
WRONG_MAP=0  
WRONG_REDUCE=0  
File Input Format Counters  
Bytes Read=84  
File Output Format Counters  
Bytes Written=97  
2024-09-20 18:55:54,220 INFO streaming.StreamJob: Output directory: /word_count_in_python/newoutput  
srimathy@srimathy-VirtualBox: $ hdfs dfs -cat /word_count_in_python/newoutput/part-00000  
Finally 1  
LA 2  
Looking 1  
Made 1  
Marina 1  
Might 1  
dive 1  
for 1  
in 3  
it 1  
the 1  
to 1  
weed 1  
yeah 2  
srimathy@srimathy-VirtualBox: ~$
```

Result:

Thus, the program for basic Word Count Map Reduce has been executed successfully.

Exp.No.: 3

Map Reduce program to process a weather dataset**AIM:**

To implement MapReduce program to process a weather dataset.

Procedure:**Step 1: Create Data File:**

Create a file named "word_count_data.txt" and populate it with text data that you wish to analyse.
Login with your hadoop user.

Download the dataset (weather data)**Output:**

```

Open ⓘ
weather_data.txt
~/weather
sample_weather.txt weather_data.txt × mapper.py hive-env.sh.template hadoop-env.sh

2024-01-01 25.6
2024-01-02 26.1
2024-01-03 24.8
2024-01-04 22.7
2024-01-05 23.9
2024-02-01 28.5
2024-02-02 27.9
2024-02-03 26.7
2024-02-04 29.1
2024-03-01 31.2
2024-03-02 32.8
2024-03-03 30.4
2024-03-04 33.6
2024-04-01 34.5
2024-04-02 35.2
2024-04-03 33.9
2024-04-04 36.1
2024-05-01 40.0
2024-05-02 39.5
2024-05-03 41.2
2024-05-04 42.1
2024-06-01 43.6

```

Step 2: Mapper Logic - mapper.py:

Create a file named "mapper.py" to implement the logic for the mapper. The mapper will read input data from STDIN, split lines into words, and output each word with its count.

```
nano mapper.py
```

```
# Copy and paste the mapper.py code
```

```
#!/usr/bin/env python
```

```
import sys
```

```
# input comes from STDIN (standard input)
```

```
# the mapper will get daily max temperature and group it by month. so output will be
(month,dailymax_temperature)
```

```

for line in sys.stdin:
    # remove leading and trailing whitespace
    line = line.strip()  # split
    the line into words  words =
    line.split()

    #See the README hosted on the weather website which help us understand how each
    position represents a column  month = line[10:12]  daily_max = line[38:45]  daily_max
    = daily_max.strip()

    # increase counters  for
    word in words:
        # write the results to STDOUT (standard output);
        # what we output here will be go through the shuffle proess and then
        # be the input for the Reduce step, i.e. the input for reducer.py
        #
        # tab-delimited; month and daily max temperature as output
        print ('%s\t%s' % (month ,daily_max))

```

Step 3: Reducer Logic - reducer.py:

Create a file named "reducer.py" to implement the logic for the reducer. The reducer will aggregate the occurrences of each word and generate the final output.

```

nano reducer.py
# Copy and paste the reducer.py code

```

reducer.py

```

#!/usr/bin/env python

from operator import itemgetter import sys
#reducer will get the input from stdid which will be a collection of key, value(Key=month , value=
daily max temperature)
#reducer logic: will get all the daily max temperature for a month and find max temperature for the
month
#shuffle will ensure that key are sorted(month)
current_month = None
current_max = 0 month =
None

# input comes from STDIN for
line in sys.stdin:
    # remove leading and trailing whitespace  line
    = line.strip()
    # parse the input we got from mapper.py  month,
    daily_max = line.split("\t", 1)

    # convert daily_max (currently a string) to float  try:

```

```

    daily_max = float(daily_max)    except
ValueError:
    # daily_max was not a number, so silently
    # ignore/discard this line
continue

    # this IF-switch only works because Hadoop shuffle process sorts map output
    # by key (here: month) before it is passed to the reducer
if current_month == month:      if daily_max > current_max:
current_max = daily_max    else:      if current_month:
    # write result to STDOUT
    print ('%s\t%s' % (current_month, current_max))
current_max = daily_max
current_month = month

# output of the last month if current_month == month:
print ('%s\t%s' % (current_month, current_max))

```

Step 4: Prepare Hadoop Environment:

Start the Hadoop daemons and create a directory in HDFS to store your data.

start-all.sh

Step 6: Make Python Files Executable:

Give executable permissions to your mapper.py and reducer.py files.

chmod 777 mapper.py reducer.py

Step 7: Run the program using Hadoop Streaming:

Download the latest hadoop-streaming jar file and place it in a location you can easily access.

Then run the program using Hadoop Streaming.

hadoop fs -mkdir -p /weatherdata

hadoop fs -copyFromLocal /home/sx/Downloads/dataset.txt /weatherdata

hdfs dfs -ls /weatherdata

hadoop jar /home/sx/hadoop-3.2.3/share/hadoop/tools/lib/hadoop-streaming-3.2.3.jar \
-input /weatherdata/dataset.txt \
-output /weatherdata/output \

```
-file "/home/sx/Downloads/mapper.py" \
-mapper "python3 mapper.py" \
-file "/home/sx/Downloads/reducer.py" \
-reducer "python3 reducer.py"
```

```
hdfs dfs -text /weatherdata/output/* > /home/sx/Downloads/outputfile.txt
```

```
srimathy@srimathy-VirtualBox:~$ hadoop jar /home/srimathy/hadoop/share/hadoop/tools/lib/hadoop-streaming-3.3.6.jar -input /weatherdata/weather_data.txt -output /weatherdata/output -file "/home/srimathy/mapper.py" -mapper "python3 mapper.py" -file "/home/srimathy/reducer.py" -reducer "python3 reducer.py"
2024-09-20 19:37:52,839 WARN streaming.StreamJob: -file option is deprecated, please use generic option -files instead.
packageJobJar: [/home/srimathy/mapper.py, /home/srimathy/reducer.py] [] /tmp/streamjob4361594942883870854.jar tmpDir=null
2024-09-20 19:37:54,189 INFO impl.MetricsConfig: Loaded properties from hadoop-metrics2.properties
2024-09-20 19:37:54,349 INFO impl.MetricsSystemImpl: Scheduled Metric snapshot period at 10 second(s).
2024-09-20 19:37:54,349 INFO impl.MetricsSystemImpl: JobTracker metrics system started
2024-09-20 19:37:54,380 WARN impl.MetricsSystemImpl: JobTracker metrics system already initialized!
2024-09-20 19:37:54,733 INFO mapred.FileInputFormat: Total input files to process : 1
2024-09-20 19:37:54,834 INFO mapreduce.JobSubmitter: number of splits:1
2024-09-20 19:37:55,022 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_local765019569_0001
2024-09-20 19:37:55,023 INFO mapreduce.JobSubmitter: Executing with tokens: []
2024-09-20 19:37:55,307 INFO mapred.LocalDistributedCacheManager: Localized file:/home/srimathy/mapper.py as file:/tmp/hadoop-srimathy/mapred/local/job_local765019569_0001_b9e636a9-f4fd-4eac-b0c4-8a016f36d246/mapper.py
2024-09-20 19:37:55,337 INFO mapred.LocalDistributedCacheManager: Localized file:/home/srimathy/reducer.py as file:/tmp/hadoop-srimathy/mapred/local/job_local765019569_0001_1bd57d98-d35f-4743-b069-89482576cc58/reducer.py
2024-09-20 19:37:55,525 INFO mapreduce.Job: The url to track the job: http://localhost:8080/
2024-09-20 19:37:55,527 INFO mapred.LocalJobRunner: OutputCommitter set in config null
2024-09-20 19:37:55,535 INFO mapreduce.Job: Running job: job_local765019569_0001
2024-09-20 19:37:55,554 INFO mapred.LocalJobRunner: OutputCommitter is org.apache.hadoop.mapred.FileOutputCommitter
2024-09-20 19:37:55,564 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 2
2024-09-20 19:37:55,564 INFO output.FileOutputCommitter: FileOutputCommitter skip cleanup _temporary folders under output directory:false, ignore cleanup failures: false
2024-09-20 19:37:55,658 INFO mapred.LocalJobRunner: Waiting for map tasks
2024-09-20 19:37:55,666 INFO mapred.LocalJobRunner: Starting task: attempt_local765019569_0001_m_000000_0
2024-09-20 19:37:55,734 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 2
2024-09-20 19:37:55,734 INFO output.FileOutputCommitter: FileOutputCommitter skip cleanup _temporary folders under output directory:false, ignore cleanup failures: false
2024-09-20 19:37:55,767 INFO mapred.Task: Using ResourceCalculatorProcessTree : []
2024-09-20 19:37:55,783 INFO mapred.MapTask: Processing split: hdfs://localhost:9000/weatherdata/weather_data.txt:0+250
2024-09-20 19:37:55 835 INFO mapred.MapTask: numReduceTasks: 1
```

Step 8: Check Output:

Check the output of the program in the specified HDFS output directory.

```
hdfs dfs -text /weatherdata/output/* > /home/sx/Downloads/output/ /part-00000
```

```
Sep 20 19:40
srimathy@srimathy-VirtualBox:~
```

```
Failed Shuffles=0
Merged Map outputs=1
GC time elapsed (ms)=50
Total committed heap usage (bytes)=734003200
Shuffle Errors
BAD_ID=0
CONNECTION=0
IO_ERROR=0
WRONG_LENGTH=0
WRONG_MAP=0
WRONG_REDUCE=0
File Input Format Counters
Bytes Read=250
File Output Format Counters
Bytes Written=270
2024-09-20 19:37:57,601 INFO streaming.StreamJob: Output directory: /weatherdata/output
srimathy@srimathy-VirtualBox:~$ hdfs dfs -ls /weatherdata/output
Found 2 items
-rw-r--r-- 3 srimathy supergroup 0 2024-09-20 19:37 /weatherdata/output/_SUCCESS
-rw-r--r-- 3 srimathy supergroup 270 2024-09-20 19:37 /weatherdata/output/part-00000
```

01	26.1
02	29.1
03	33.6
04	36.1
05	42.1
06	44.0

After copy and paste the above output in your local file give the below command to remove the directory from hdfs : hadoop fs -rm -r /weatherdata/output

Result:

Thus, the program for weather dataset using Map Reduce has been executed successfully.

Exp.No.: 4

Create UDF in PIG

Step-by-step installation of Apache Pig on Hadoop cluster on Ubuntu Pre-requisite:

- Ubuntu 16.04 or higher version running (I have installed Ubuntu on Oracle VM (Virtual Machine) VirtualBox),
- Run Hadoop on ubuntu (I have installed Hadoop 3.2.1 on Ubuntu 16.04). You may refer to my blog “How to install Hadoop installation” click [here](#) for Hadoop installation).

Pig installation steps

Step 1: Login into Ubuntu

```
hadoop@hadoop-VirtualBox:~$ wget https://dlcdn.apache.org/pig/pig-0.16.0/pig-0.16.0.tar.gz
$: command not found
hadoop@hadoop-VirtualBox:~$ wget https://dlcdn.apache.org/pig/pig-0.16.0/pig-0.16.0.tar.gz
--2022-06-21 11:57:52-- https://dlcdn.apache.org/pig/pig-0.16.0/pig-0.16.0.tar.gz
Resolving dlcdn.apache.org (dlcdn.apache.org)... 151.101.2.132, 2a04:4e42::644
Connecting to dlcdn.apache.org (dlcdn.apache.org)|151.101.2.132|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 177279333 (169M) [application/x-gzip]
Saving to: 'pig-0.16.0.tar.gz.1'

pig-0.16.0.tar.gz.1 94%[=====] 158.94M 5.19MB/s eta 2s
```

Step 2: Go to <https://pig.apache.org/releases.html> and copy the path of the latest version of pig that you want to install. Run the following comment to download Apache Pig in Ubuntu:

```
$ wget https://dlcdn.apache.org/pig/pig-0.16.0/pig-0.16.0.tar.gz
```

Step 3: To untar pig-0.16.0.tar.gz file run the following command:

```
$ tar xvzf pig-0.16.0.tar.gz
```

Step 4: To create a pig folder and move pig-0.16.0 to the pig folder, execute the following command:

```
$ sudo mv /home/hadoop/pig-0.16.0 /home/hadoop/pig
```

Step 5: Now open the .bashrc file to edit the path and variables/settings for pig. Run the following command:

```
$ sudo nano .bashrc
```

Add the below given to .bashrc file at the end and save the file.

```
#PIG settingsexport PIG_HOME=/home/hadoop/pigexport
PATH=$PATH:$PIG_HOME/binexport
```

```
PIG_CLASSPATH=$PIG_HOME/conf:$HADOOP_INSTALL/etc/hadoop/export
PIG_CONF_DIR=$PIG_HOME/conf/export JAVA_HOME=/usr/lib/jvm/java-8-
openjdk-amd64export PIG_CLASSPATH=$PIG_CONF_DIR:$PATH#PIG setting ends
```

```
GNU nano 7.2 .bashrc

export JAVA_HOME=/usr/lib/jvm/java-11-openjdk-amd64
export HADOOP_HOME=/home/hadoop/hadoop
export HADOOP_INSTALL=$HADOOP_HOME
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME
export HADOOP_YARN_HOME=$HADOOP_HOME
export HADOOP_COMMON_LIB_NATIVE=$HADOOP_HOME/lib/native
export PATH=$PATH:$HADOOP_HOME/bin:$HADOOP_HOME/sbin
export HADOOP_OPTS="-Djava.library.path=$HADOOP_HOME/lib/native"

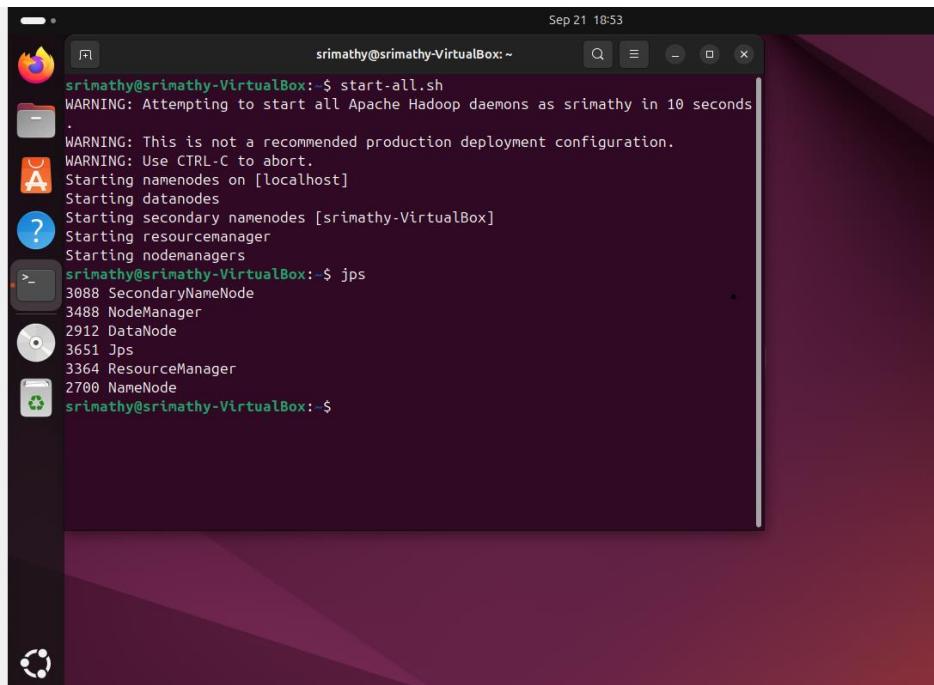
# PIG settings
export PIG_HOME=/home/hadoop/pig
export PATH=$PATH:$PIG_HOME/bin
export PIG_CLASSPATH=$PIG_HOME/conf:$HADOOP_INSTALL/etc/hadoop
export PIG_CONF_DIR=$PIG_HOME/conf
export JAVA_HOME=/usr/lib/jvm/java-11-openjdk-amd64
export PIG_CLASSPATH=$PIG_CONF_DIR:$PIG_CLASSPATH
# PIG settings end
```

Step 6: Run the following command to make the changes effective in the .bashrc file:

```
$ source .bashrc
```

Step 7: To start all Hadoop daemons, navigate to the hadoop-3.2.1/sbin folder and run the following commands:

```
$ ./start-dfs.sh$ ./start-yarn$ jps
```



Step 8: Now you can launch pig by executing the following command: \$ pig

```
Sep 19 18:23
srimathy@srimathy-VirtualBox:~$ ls
demo_pig.pig  mapper.py  pig-0.16.0.tar.gz  snap
Desktop       Music      pig_1726303616365.log  Templates
Documents     Pictures   reducer.py        uppercase_udf.py
Downloads    sample.txt  word_count.txt
hadoop        pig-0.16.0
srimathy@srimathy-VirtualBox:~$ pig
2024-09-19 18:23:14,166 INFO pig.ExecTypeProvider: Trying ExecType : LOCAL
2024-09-19 18:23:14,168 INFO pig.ExecTypeProvider: Trying ExecType : MAPREDUCE
2024-09-19 18:23:14,168 INFO pig.ExecTypeProvider: Picked MAPREDUCE as the ExecType
2024-09-19 18:23:14,254 [main] INFO org.apache.pig.Main - Apache Pig version 0.16.0 (r1746530) compiled Jun 01 2016, 23:10:49
2024-09-19 18:23:14,254 [main] INFO org.apache.pig.Main - Logging error messages to: /home/srimathy/pig_1726750394238.log
2024-09-19 18:23:14,304 [main] INFO org.apache.pig.impl.util.Utils - Default bootup file /home/srimathy/.pigbootup not found
2024-09-19 18:23:14,883 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - mapred.job.tracker is deprecated. Instead, use mapreduce.jobtracker.address
2024-09-19 18:23:14,883 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instead, use fs.defaultFS
2024-09-19 18:23:14,883 [main] INFO org.apache.pig.backend.hadoop.executionengine.HExecutionEngine - Connecting to hadoop file system at: hdfs://localhost:9000
2024-09-19 18:23:16,176 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instead, use fs.defaultFS
2024-09-19 18:23:16,219 [main] INFO org.apache.pig.PigServer - Pig Script ID for the session: PIG-default-2ad3d2fa-40c2-4d2e-8a30-746521057da9
2024-09-19 18:23:16,219 [main] WARN org.apache.pig.PigServer - ATS is disabled since yarn.timeline-service.enabled set to false
grunt> 
```

Step 9: Now you are in pig and can perform your desired tasks on pig. You can come out of the pig by the quit command:

> quit;

CREATE USER DEFINED FUNCTION(UDF)

Aim :

To create User Define Function in Apache Pig and execute it on map reduce.

PROCEDURE:

Create a sample text file

```
hadoop@Ubuntu:~/Documents$ nano sample.txt
```

Paste the below content to sample.txt

1,John

2,Joe

3,Emma

4,Peter

```
hadoop@Ubuntu:~/Documents$ hadoop fs -put sample.txt /home/hadoop/piginput/
```

Create PIG File

```
hadoop@Ubuntu:~/Documents$ nano demo_pig.pig
```

paste the below the content to demo_pig.pig

-- Load the data from HDFS

```
data = LOAD '/home/hadoop/piginput/sample.txt' USING PigStorage(',') AS (id:int>
```

-- Dump the data to check if it was loaded correctly

```
DUMP data;
```

----- Run

the above file

```
hadoop@Ubuntu:~/Documents$ pig demo_pig.pig
```

```

Sep 19 19:05
srimathy@srimathy-VirtualBox: ~

Job Stats (time in seconds):
JobId Maps Reduces MaxMapTime MinMapTime AvgMapTime MedianMapTime MaxReduceTime MinReduceTime
AvgReduceTime MedianReducetime Alias Feature Outputs
job_local663734817_0001 1 0 n/a n/a n/a n/a 0 00 0 data MAP_ONLY
hdfs://localhost:9000/tmp/temp-1542450718/tmp1069937356,

Input(s):
Successfully read 4 records (5378235 bytes) from: "/user/srimathy/simple.txt"

Output(s):
Successfully stored 4 records (5378230 bytes) in: "hdfs://localhost:9000/tmp/temp-1542450718/tmp1069937356"

Counters:
Total records written : 4
Total bytes written : 5378230
Spillable Memory Manager spill count : 0
Total bags proactively spilled: 0
Total records proactively spilled: 0

Job DAG:
job_local663734817_0001

2024-09-19 19:04:41,032 [main] WARN org.apache.hadoop.metrics2.impl.MetricsSystemImpl - JobTracker metrics system already initialized!
2024-09-19 19:04:41,037 [main] WARN org.apache.hadoop.metrics2.impl.MetricsSystemImpl - JobTracker metrics system already initialized!
2024-09-19 19:04:41,041 [main] WARN org.apache.hadoop.metrics2.impl.MetricsSystemImpl - JobTracker metrics system already initialized!
2024-09-19 19:04:41,056 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - Success!
2024-09-19 19:04:41,062 [main] INFO org.apache.hadoop.conf.Configuration - fs.default.name is deprecated. T

```

Create udf file an save as uppercase_udf.py

uppercase_udf.py

```

def uppercase(text): return text.upper()

if __name__ == "__main__":
    import sys
    for line in sys.stdin:
        line = line.strip()
        result = uppercase(line)
        print(result)

```

Create the udfs folder on hadoop

hadoop@Ubuntu:~/Documents\$ hadoop fs -mkdir /home/hadoop/udfs

put the uppercase_udf.py in to the abv folder

hadoop@Ubuntu:~/Documents\$ hdfs dfs -put uppercase_udf.py /home/hadoop/udfs/

hadoop@Ubuntu:~/Documents\$ nano udf_example.pig copy and paste the below content on udf_example.pig

-- Register the Python UDF script

```
REGISTER 'hdfs:///home/hadoop/udfs/uppercase_udf.py' USING jython AS udf;
```

-- Load some data

```
data = LOAD 'hdfs:///home/hadoop/sample.txt' AS (text:chararray);
```

-- Use the Python UDF

```
uppercased_data = FOREACH data GENERATE udf.uppercase(text) AS uppercase_text;
```

-- Store the result

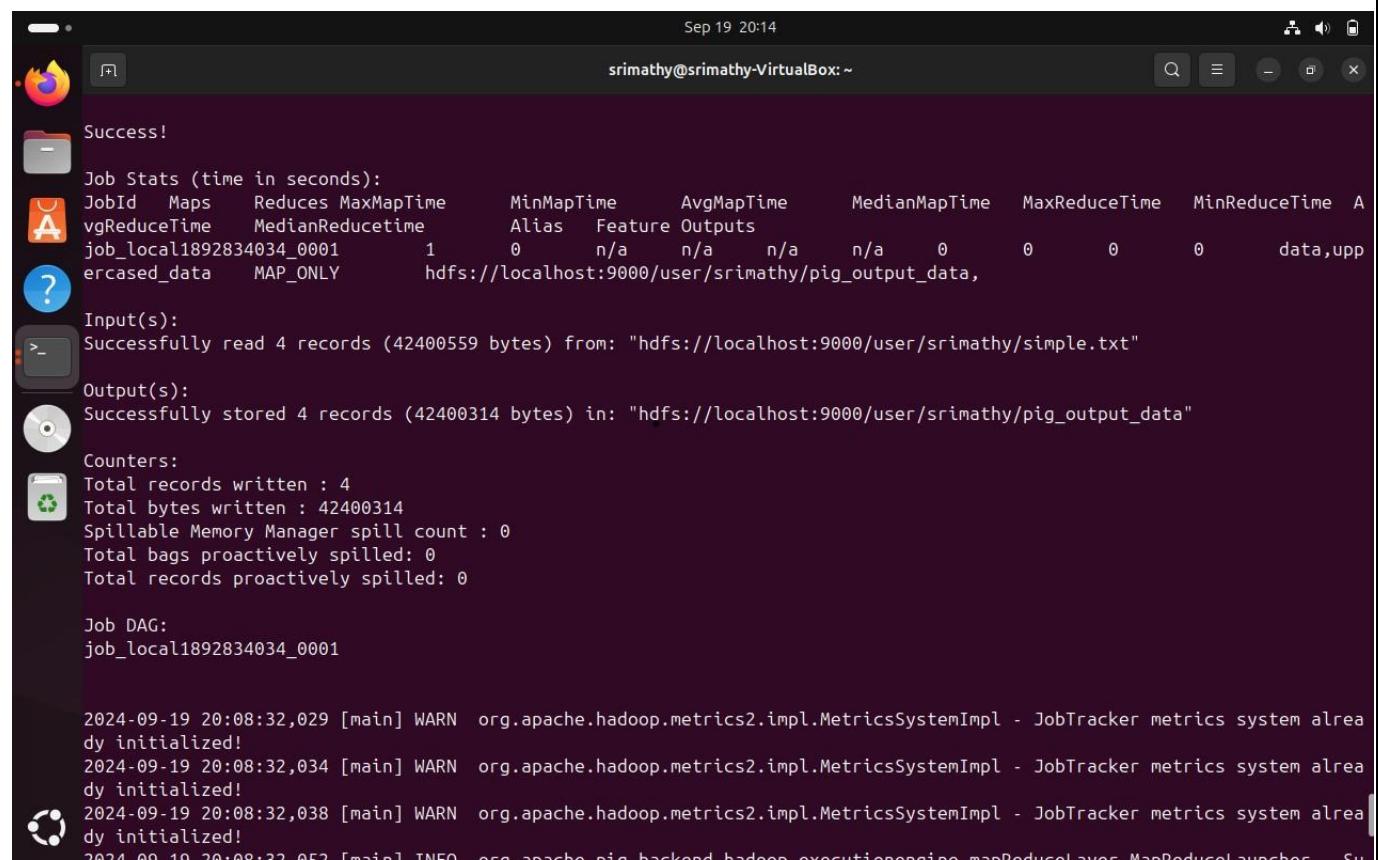
```
STORE uppercased_data INTO 'hdfs:///home/hadoop/pig_output_data';
```

place sample.txt file on hadoop

```
hadoop@Ubuntu:~/Documents$ hadoop fs -put sample.txt /home/hadoop/
```

To Run the pig file

```
hadoop@Ubuntu:~/Documents$ pig -f udf_example.pig
```



```
Sep 19 20:14
srimathy@srimaly-VirtualBox:~

Success!
Job Stats (time in seconds):
JobId   Maps   Reduces MaxMapTime      MinMapTime      AvgMapTime      MedianMapTime    MaxReduceTime   MinReduceTime   A
vgReduceTime   MedianReducetime   Alias   Feature Outputs
job_local1892834034_0001      1       0       n/a       n/a       n/a       0       0       0       0       data,upp
ercased_data   MAP_ONLY          hdfs://localhost:9000/user/srimathy/pig_output_data,
Input(s):
Successfully read 4 records (42400559 bytes) from: "hdfs://localhost:9000/user/srimathy/simple.txt"
Output(s):
Successfully stored 4 records (42400314 bytes) in: "hdfs://localhost:9000/user/srimathy/pig_output_data"
Counters:
Total records written : 4
Total bytes written : 42400314
Spillable Memory Manager spill count : 0
Total bags proactively spilled: 0
Total records proactively spilled: 0
Job DAG:
job_local1892834034_0001

2024-09-19 20:08:32,029 [main] WARN org.apache.hadoop.metrics2.impl.MetricsSystemImpl - JobTracker metrics system already initialized!
2024-09-19 20:08:32,034 [main] WARN org.apache.hadoop.metrics2.impl.MetricsSystemImpl - JobTracker metrics system already initialized!
2024-09-19 20:08:32,038 [main] WARN org.apache.hadoop.metrics2.impl.MetricsSystemImpl - JobTracker metrics system already initialized!
2024-09-19 20:08:32,052 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - Su
```

To check the output file is created

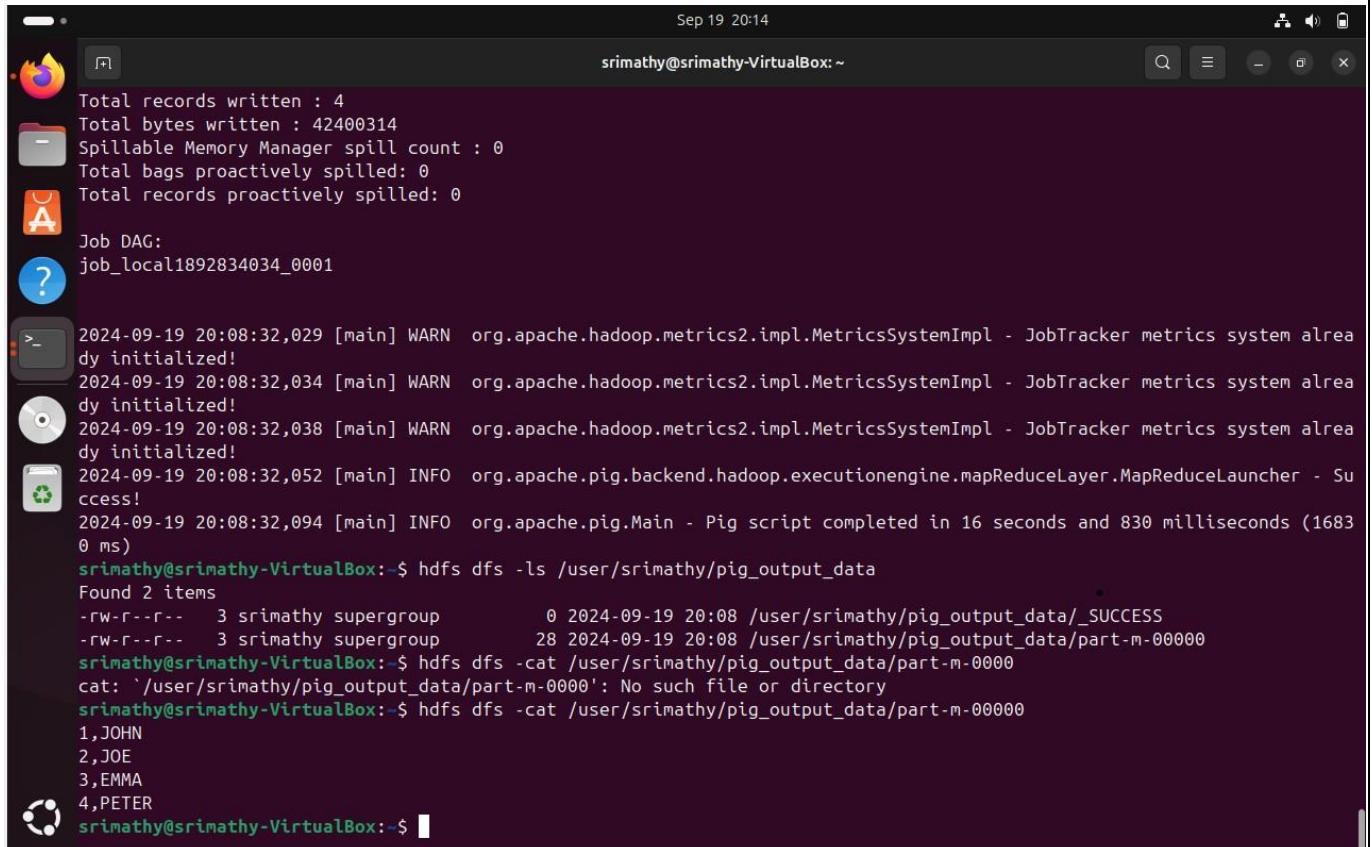
```
hadoop@Ubuntu:~/Documents$ hdfs dfs -ls /home/hadoop/pig_output_data
```

Found 2 items

If you need to examine the files in the output folder, use:

To view the output

```
hadoop@Ubuntu:~/Documents$ hdfs dfs -cat /home/hadoop/pig_output_data/part-m00000
```



The screenshot shows a terminal window titled 'srimathy@srimathy-VirtualBox: ~'. The terminal displays the following command and its output:

```

Total records written : 4
Total bytes written : 42400314
Spillable Memory Manager spill count : 0
Total bags proactively spilled: 0
Total records proactively spilled: 0

Job DAG:
job_local1892834034_0001

2024-09-19 20:08:32,029 [main] WARN org.apache.hadoop.metrics2.impl.MetricsSystemImpl - JobTracker metrics system already initialized!
2024-09-19 20:08:32,034 [main] WARN org.apache.hadoop.metrics2.impl.MetricsSystemImpl - JobTracker metrics system already initialized!
2024-09-19 20:08:32,038 [main] WARN org.apache.hadoop.metrics2.impl.MetricsSystemImpl - JobTracker metrics system already initialized!
2024-09-19 20:08:32,052 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - Success!
2024-09-19 20:08:32,094 [main] INFO org.apache.pig.Main - Pig script completed in 16 seconds and 830 milliseconds (1683 0 ms)

srimathy@srimathy-VirtualBox:~$ hdfs dfs -ls /user/srimathy/pig_output_data
Found 2 items
-rw-r--r-- 3 srimathy supergroup 0 2024-09-19 20:08 /user/srimathy/pig_output_data/_SUCCESS
-rw-r--r-- 3 srimathy supergroup 28 2024-09-19 20:08 /user/srimathy/pig_output_data/part-m-00000
srimathy@srimathy-VirtualBox:~$ hdfs dfs -cat /user/srimathy/pig_output_data/part-m-00000
cat: '/user/srimathy/pig_output_data/part-m-00000': No such file or directory
srimathy@srimathy-VirtualBox:~$ hdfs dfs -cat /user/srimathy/pig_output_data/part-m-00000
1,JOHN
2,JOE
3,EMMA
4,PETER
srimathy@srimathy-VirtualBox:~$
```

Result:

Thus the program to create User Define Function in Apache Pig and execute it on map reduce has been done successfully.

Exp.No.:5**Installation of Hive on Ubuntu****Aim:**

To Download and install Hive, Understanding Startup scripts, Configuration files.

Procedure:**Step 1: Download and extract it**

Download the Apache hive and extract it use tar, the commands given below:

```
$ wget https://downloads.apache.org/hive/hive-3.1.2/apache-hive-3.1.2-bin.tar.gz
```

```
/apache-hive-3.1.2-bin.tar.gz
--2024-09-02 12:26:15-- https://archive.apache.org/dist/hive/hive-3.1.2/apache-
hive-3.1.2-bin.tar.gz
Resolving archive.apache.org (archive.apache.org)... 65.108.204.189, 2a01:4f9:1a
:a084::2
Connecting to archive.apache.org (archive.apache.org)|65.108.204.189|:443... con-
nected.
HTTP request sent, awaiting response... 200 OK
Length: 278813748 (266M) [application/x-gzip]
Saving to: 'apache-hive-3.1.2-bin.tar.gz'

apache-hive-3.1.2-b 100%[=====] 265.90M 1.20MB/s    in 2m 57s

2024-09-02 12:29:13 (1.50 MB/s) - 'apache-hive-3.1.2-bin.tar.gz' saved [27881374
8/278813748]
```

```
$ tar -xvf apache-hive-3.1.2-bin.tar.gz
```

```
apache-hive-3.1.2-bin/LICENSE
apache-hive-3.1.2-bin/NOTICE
apache-hive-3.1.2-bin/RELEASE_NOTES.txt
apache-hive-3.1.2-bin/binary-package-licenses/asm-LICENSE
apache-hive-3.1.2-bin/binary-package-licenses/com.google.protobuf-LICENSE
apache-hive-3.1.2-bin/binary-package-licenses/com.ibm.icu.icu4j-LICENSE
apache-hive-3.1.2-bin/binary-package-licenses/com.sun.jersey-LICENSE
apache-hive-3.1.2-bin/binary-package-licenses/com.thoughtworks.paranamer-LICENSE
apache-hive-3.1.2-bin/binary-package-licenses/javax.transaction.transaction-api-
LICENSE
apache-hive-3.1.2-bin/binary-package-licenses/javolution-LICENSE
apache-hive-3.1.2-bin/binary-package-licenses/jline-LICENSE
apache-hive-3.1.2-bin/binary-package-licenses/NOTICE
apache-hive-3.1.2-bin/binary-package-licenses/org.abego.treelayout.core-LICENSE
apache-hive-3.1.2-bin/binary-package-licenses/org.antlr-LICENSE
apache-hive-3.1.2-bin/binary-package-licenses/org.antlr.antlr4-LICENSE
```

Step 2: Place different configuration properties in Apache Hive

In this step, we are going to do two things o Placing

Hive Home path in bashrc file

```
$ nano .bashrc
```

And append the below lines in it

```
#HIVE settings
export HIVE_HOME=/home/hadoop/apache-hive-3.1.2
export PATH=$PATH:$HIVE_HOME/bin
#HIVE settings end
```

2. Exporting **Hadoop path in Hive-config.sh** (To communicate with the Hadoop eco system we are defining Hadoop Home path in hive config field) **Open the hiveconfig.sh as shown in below \$cd apache-hive-3.1.2-bin/bin**

```
$cp hive-env.sh.template hive-env.sh
$nano hive-env.sh
Append the below commands on it      export
HADOOP_HOME=/home/Hadoop/Hadoop
export HIVE_CONF_DIR=/home/Hadoop/apache-hive-3.1.2/conf
```

```
# Set HADOOP_HOME to point to a specific hadoop install directory
# HADOOP_HOME=${bin}/../../hadoop
export HADOOP_HOME=/home/hadoop/hadoop

# Hive Configuration Directory can be controlled by:
# export HIVE_CONF_DIR=
export HIVE_CONF_DIR=/home/hadoop/apache-hive-3.1.2-bin/conf
# Folder containing extra libraries required for hive compilation/execution can be controlled by:
```

Step 3: Install mysql

1. Install mysql in Ubuntu by running this command:

```
$sudo apt update
$sudo apt install mysql-server
```

2. Alter username and password for MySQL by running below commands:

```
$sudomysql
```

Pops command line interface for MySQL and run the below SQL queries to change username and set password

```
mysql> SELECT user, host, plugin FROM mysql.user WHERE user = 'root';
mysql> ALTER USER 'root'@'localhost' IDENTIFIED WITH 'mysql_native_password' BY
'your_new_password';
mysql> FLUSH PRIVILEGES;
```

Step 4: Config hive-site.xml

Config the hive-site.xml by appending this xml code and change the username and password according to your MySQL.

```
$cd apache-hive-3.1.2-bin/bin
$cp hive-default.xml.template hive-site.xml
$nano hive-site.xml
```

Append these lines into it

Replace root as your username of MySQL

Replace your_new_password as with your password of MySQL

```
<configuration>
```

```
<property>
```

```
    <name>javax.jdo.option.ConnectionURL</name>
    <value>jdbc:mysql://localhost/metastore?createDatabaseIfNotExist=true</value>
  </property>
```

```

<property>
<name>javax.jdo.option.ConnectionDriverName</name>
<value>com.mysql.cj.jdbc.Driver</value>
</property>

<property>
<name>javax.jdo.option.ConnectionUserName</name>
<value>root</value>
</property>

<property>
<name>javax.jdo.option.ConnectionPassword</name>
<value>your_new_password</value>
</property>

<property>
<name>datanucleus.autoCreateSchema</name>
<value>true</value>
</property>

<property>
<name>datanucleus.fixedDatastore</name>
<value>true</value>
</property>

<property>
<name>datanucleus.autoCreateTables</name>
<value>True</value>
</property>

</configuration>

```

Step 5: Setup MySQL java connector:

First, you'll need to download the MySQL Connector/J, which is the JDBC driver for MySQL. You can download it from the below link

https://drive.google.com/file/d/1QFhB7Kvcat7a4LzDRe6GcmZva1yAxKz/view?usp=drive_link

Copy the downloaded MySQL Connector/J JAR file to the Hive library directory. By default, the Hive library directory is usually located at /path/to/apache-hive-3.1.2/lib/on Ubuntu. Use the following command to copy the JAR file:

\$sudo cp /path/to/mysql-connector-java-8.0.15.jar /path/to/apache-hive-3.1.2/lib/ Replace /path/to/ with the actual path to the JAR file.

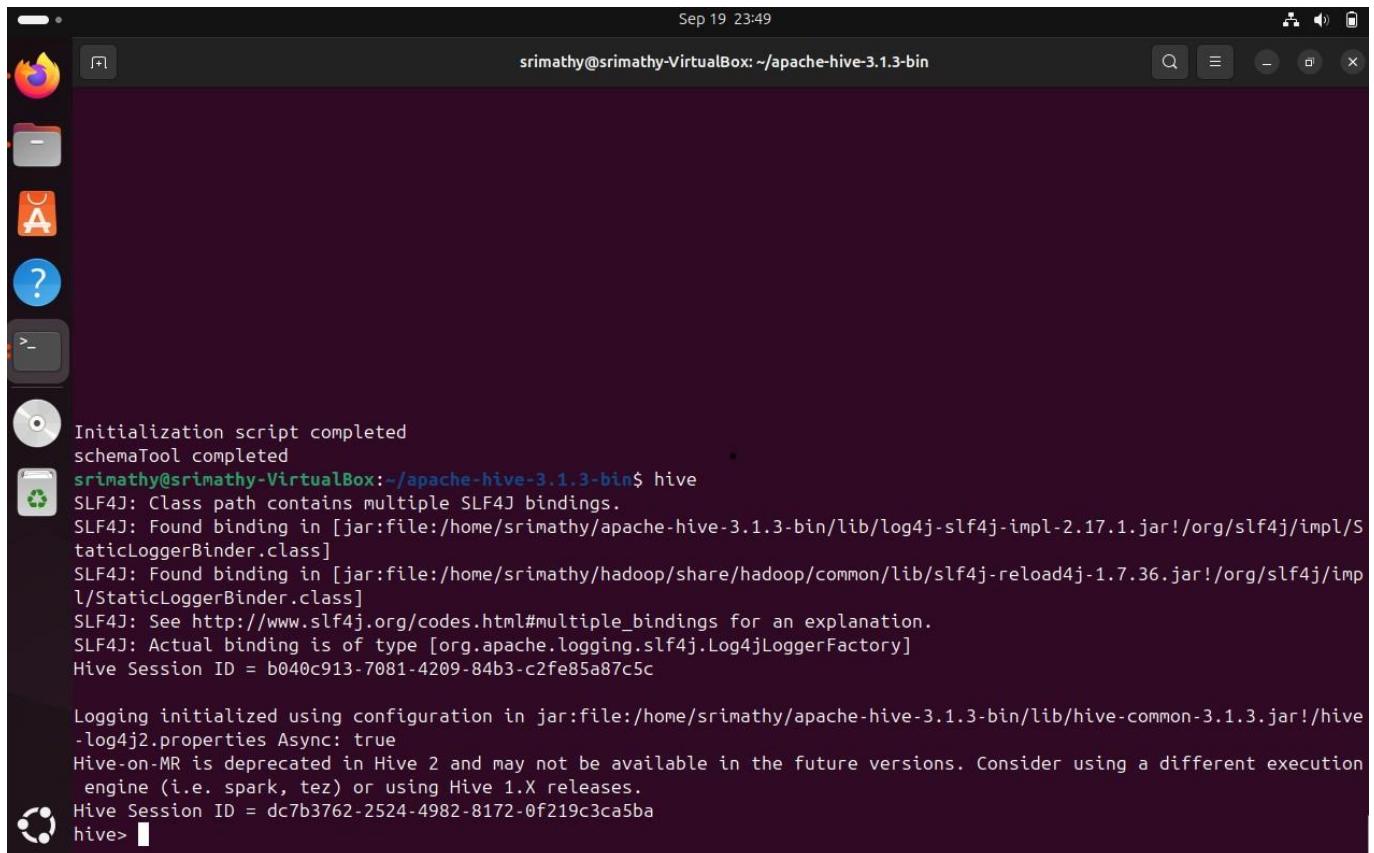
Step 6:Initialize the Hive Metastore Schema:

Run the following command to initialize the Hive metastore schema:

\$\$HIVE_HOME/bin/schematool -initSchema -dbTypemysql

Step 7: Start hive:

You can test Hive by running the Hive shell: Copy code hive You should be able to run Hive queries, and metadata will be stored in your MySQL database. `$hive`



The screenshot shows a terminal window on a Linux desktop environment. The terminal title is "srimathy@srimathy-VirtualBox: ~/apache-hive-3.1.3-bin". The window contains the following text output from the Hive installation:

```
Sep 19 23:49
Initialization script completed
schemaTool completed
srimathy@srimathy-VirtualBox:~/apache-hive-3.1.3-bin$ hive
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/srimathy/apache-hive-3.1.3-bin/lib/log4j-slf4j-impl-2.17.1.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/srimathy/hadoop/share/hadoop/common/lib/slf4j-reload4j-1.7.36.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 = b040c913-7081-4209-84b3-c2fe85a87c5c

Logging initialized using configuration in jar:file:/home/srimathy/apache-hive-3.1.3-bin/lib/hive-common-3.1.3.jar!/hive-log4j2.properties Async: true
Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e. spark, tez) or using Hive 1.X releases.
Hive Session ID = dc7b3762-2524-4982-8172-0f219c3ca5ba
hive> 
```

Result:

Thus, the Apache Hive installation is completed successfully on Ubuntu.

Exp.No.: 5a**Design and test various schema models to optimize data storage and retrieval Using Hive****Aim:**

To Design and test various schema models to optimize data storage and retrieval Using Hbase.

Procedure:**Step 1: Start Hive**

Open a terminal and start Hive by running:

\$hive

```

Initialization script completed
SchemaTool completed
srimathy@srimathy-VirtualBox:~/apache-hive-3.1.3-bin$ hive
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/srimathy/apache-hive-3.1.3-bin/lib/log4j-slf4j-impl-2.17.1.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/srimathy/hadoop/share/hadoop/common/lib/slf4j-reload4j-1.7.36.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 = b040c913-7081-4209-84b3-c2fe85a87c5c

Logging initialized using configuration in jar:file:/home/srimathy/apache-hive-3.1.3-bin/lib/hive-common-3.1.3.jar!/hive-log4j2.properties Async: true
Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e. spark, tez) or using Hive 1.X releases.
Hive Session ID = dc7b3762-2524-4982-8172-0f219c3ca5ba
hive> 

```

Step 2: Create a Database

Create a new database in Hive: `hive>CREATE DATABASE financials;`

```

hive> CREATE DATABASE financials;
OK
Time taken: 0.063 seconds

```

Step 3: Use the Database:

Switch to the newly created database: `hive>use financials;`

Step 4: Create a Table:

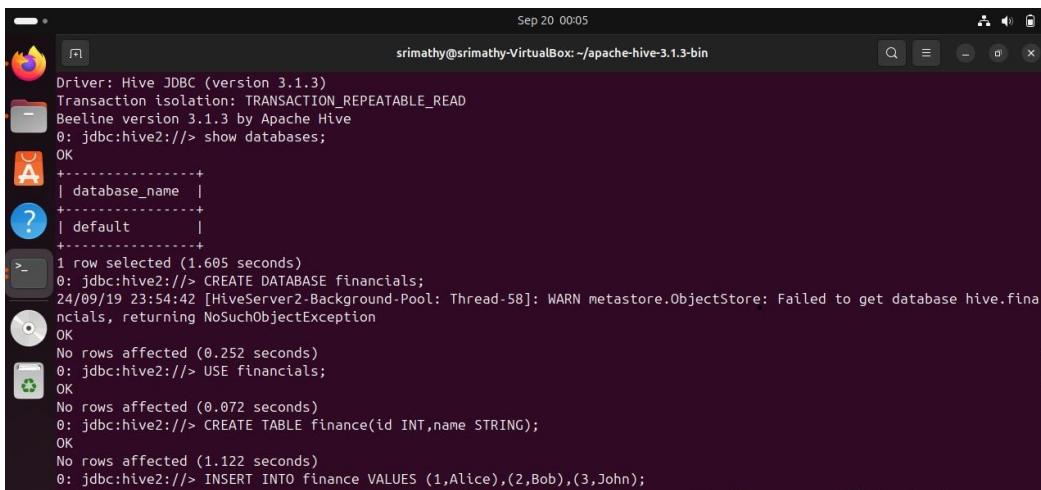
Create a simple table in your database:

```
hive>CREATE TABLE finance_table( id INT, name STRING );
```

Step 5: Load Sample Data:

You can insert sample data into the table:

```
hive>INSERT INTO finance_tableVALUES (1, 'Alice'), (2, 'Bob'), (3, 'Charlie');
```



```

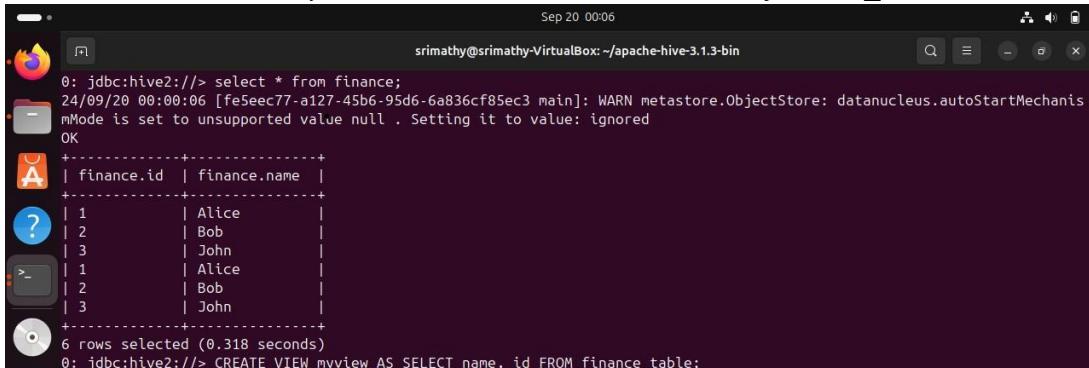
Driver: Hive JDBC (version 3.1.3)
Transaction isolation: TRANSACTION_REPEATABLE_READ
Beeline version 3.1.3 by Apache Hive
0: jdbc:hive2://> show databases;
OK
+-----+
| database_name |
+-----+
| default      |
+-----+
1 row selected (1.605 seconds)
0: jdbc:hive2://> CREATE DATABASE financials;
24/09/19 23:54:42 [HiveServer2-Background-Pool: Thread-58]: WARN metastore.ObjectStore: Failed to get database hive.financials, returning NoSuchObjectException
OK
No rows affected (0.252 seconds)
0: jdbc:hive2://> USE financials;
OK
No rows affected (0.072 seconds)
0: jdbc:hive2://> CREATE TABLE finance(id INT,name STRING);
OK
No rows affected (1.122 seconds)
0: jdbc:hive2://> INSERT INTO finance VALUES (1,Alice),(2,Bob),(3,John);

```

Step 6: Query Your Data

Use SQL-like queries to retrieve data from your table:

hive>CREATE VIEW myview AS SELECT name, id FROM finance_table;



```

0: jdbc:hive2://> select * from finance;
24/09/20 00:00:06 [feSeec77-a127-45b6-95d6-6a836cf85ec3 main]: WARN metastore.ObjectStore: datanucleus.autoStartMechanismMode is set to unsupported value null . Setting it to value: ignored
OK
+-----+-----+
| finance.id | finance.name |
+-----+-----+
| 1          | Alice       |
| 2          | Bob         |
| 3          | John        |
| 1          | Alice       |
| 2          | Bob         |
| 3          | John        |
+-----+-----+
6 rows selected (0.318 seconds)
0: jdbc:hive2://> CREATE VIEW myview AS SELECT name, id FROM finance_table;

```

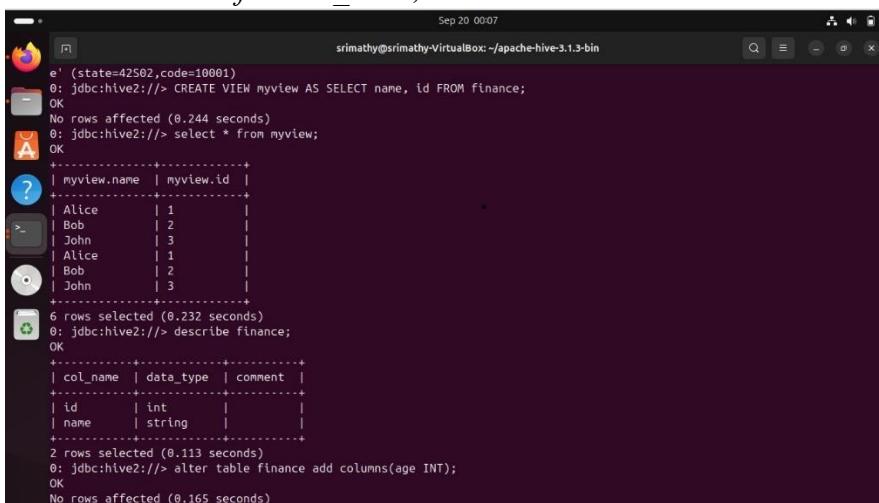
Step 7: View the data:

To see the data in the view, you would need to query the view `hive>SELECT*FROM myview;`

Step 8: Describe a Table:

You can describe the structure of a table using the `DESCRIBE` command:

hive>`DESCRIBE finance_table;`



```

e' (state=42502,code=10001)
0: jdbc:hive2://> CREATE VIEW myview AS SELECT name, id FROM finance;
OK
No rows affected (0.244 seconds)
0: jdbc:hive2://> select * from myview;
OK
+-----+-----+
| myview.name | myview.id |
+-----+-----+
| Alice       | 1          |
| Bob         | 2          |
| John        | 3          |
| Alice       | 1          |
| Bob         | 2          |
| John        | 3          |
+-----+-----+
6 rows selected (0.232 seconds)
0: jdbc:hive2://> describe finance;
OK
+-----+-----+-----+
| col_name | data_type | comment |
+-----+-----+-----+
| id       | int       |          |
| name     | string    |          |
+-----+-----+-----+
2 rows selected (0.113 seconds)
0: jdbc:hive2://> alter table finance add columns(age INT);
OK
No rows affected (0.165 seconds)

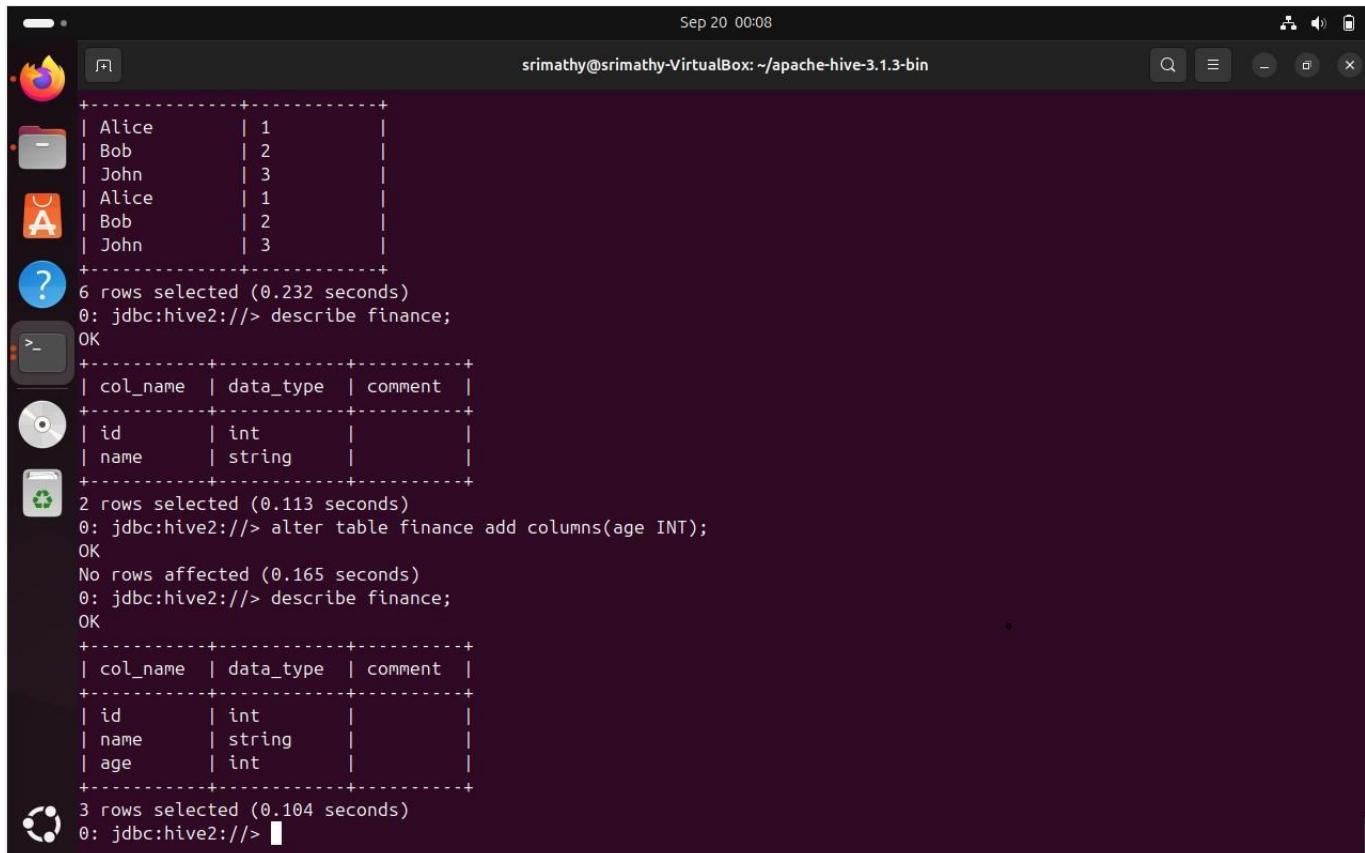
```

Step 9: Alter a Table:

You can alter the table structure by adding a new column: `hive>ALTER TABLE finance_table ADD COLUMNS (age INT);`

Step 10: Quit Hive:

To exit the Hive CLI, simply type: `hive>quit;`



The screenshot shows a terminal window titled "srimathy@srimathy-VirtualBox: ~/apache-hive-3.1.3-bin". The session starts with a query to a table named "finance" which contains three rows of data with columns "id" and "name". Then, the user runs the command `describe finance;` which shows the table structure with two columns: "id" (int) and "name" (string). Next, the user runs the command `alter table finance add columns(age INT);` followed by `OK`. A message indicates "No rows affected (0.165 seconds)". Finally, the user runs `describe finance;` again, and the output shows the table now has three columns: "id" (int), "name" (string), and "age" (int). The session ends with `3 rows selected (0.104 seconds)` and the prompt `0: jdbc:hive2://> [REDACTED]`.

```
+-----+-----+
| Alice | 1   |
| Bob   | 2   |
| John  | 3   |
+-----+
6 rows selected (0.232 seconds)
0: jdbc:hive2://> describe finance;
OK
+-----+-----+-----+
| col_name | data_type | comment |
+-----+-----+-----+
| id       | int        |          |
| name     | string     |          |
+-----+-----+-----+
2 rows selected (0.113 seconds)
0: jdbc:hive2://> alter table finance add columns(age INT);
OK
No rows affected (0.165 seconds)
0: jdbc:hive2://> describe finance;
OK
+-----+-----+-----+
| col_name | data_type | comment |
+-----+-----+-----+
| id       | int        |          |
| name     | string     |          |
| age      | int        |          |
+-----+-----+-----+
3 rows selected (0.104 seconds)
0: jdbc:hive2://> [REDACTED]
```

Result:

Thus, the usage of various commands in Hive has been successfully completed.

Ex.No.: 6

Import a JASON file from the command line. Apply the following actions with the data present in the JASON file where, projection, aggregation, remove, count, limit, skip and sort

AIM:

To import a JASON file from the command line and apply the following actions with the data present in the JASON file where, projection, aggregation, remove, count, limit, skip and sort.

PROCEDURE:

1. Required Packages Installation

- Install Pandas

Pandas is required for manipulating and analyzing data.

Installation:

- pip install pandas

- Install HDFS

HDFS provides a Python interface to interact with Hadoop Distributed File System (HDFS).

Installation:

- pip install hdfs

- Optional Packages

These packages may help when working with large datasets or different formats:

o PyArrow (for Apache Arrow support):

- pip install pyarrow

o HDFS3 (alternative to HDFS):

- pip install hdfs3

2. Create a json file (for example: emp.json) with the following content:

```
[{"name": "Alice", "salary": 60000, "department": "HR"},  
 {"name": "Bob", "salary": 55000, "department": "Finance"},  
 {"name": "Charlie", "salary": 70000, "department": "IT"},  
 {"name": "David", "salary": 45000, "department": "Sales"},  
 {"name": "Eve", "salary": 80000, "department": "IT"}]
```

3. Copy the json file to the hdfs directory using the command:

```
$ hdfs dfs copyFromLocal /path/to/emp.json /home/hadoop
```

Also give the necessary permissions if not already given using the command:

```
$ hdfs dfs -chmod 777 /home/hadoop
```

4. Python Script: process_data.py

The following script reads a JSON file from HDFS, processes it using Pandas, and performs several operations such as projection, aggregation, counting, limiting, skipping, and filtering.

```

#process_data.py
from hdfs import InsecureClient
import pandas as pd
import json
# Connect to HDFS
hdfs_client = InsecureClient('http://localhost:9870', user='hdfs')
# Read JSON data from HDFS
try:
    with hdfs_client.read('/home/hadoop/emp.json', encoding='utf-8') as reader:
        json_data = reader.read() # Read the raw data as a string
        if not json_data.strip(): # Check if data is empty
            raise ValueError("The JSON file is empty.")
        print(f"Raw JSON Data: {json_data[:1000]}") # Print first 1000 characters for debugging
        data = json.loads(json_data) # Load the JSON data
except json.JSONDecodeError as e:
    print(f"JSON Decode Error: {e}")
    exit(1)
except Exception as e:
    print(f"Error reading or parsing JSON data: {e}")
    exit(1)
# Convert JSON data to DataFrame
try:
    df = pd.DataFrame(data)
except ValueError as e:
    print(f"Error converting JSON data to DataFrame: {e}")
    exit(1)
# Projection: Select only 'name' and 'salary' columns
projected_df = df[['name', 'salary']]
# Aggregation: Calculate total salary
total_salary = df['salary'].sum()
# Count: Number of employees earning more than 50000
high_earners_count = df[df['salary'] > 50000].shape[0]
# Limit: Get the top 5 highest earners
top_5_earners = df.nlargest(5, 'salary')
# Skip: Skip the first 2 employees
skipped_df = df.iloc[2:]
# Remove: Remove employees from a specific department (e.g., 'Sales')
filtered_df = df[df['department'] != 'IT']
# Save the filtered result back to HDFS
filtered_json = filtered_df.to_json(orient='records')
try:
    with hdfs_client.write('/home/hadoop/filtered_employees.json', encoding='utf-8', overwrite=True) as writer:
        writer.write(filtered_json)
        print("Filtered JSON file saved successfully.")
except Exception as e:

```

```

print(f'Error saving filtered JSON data: {e}')
exit(1)
# Print results
print(f'Projection: Select only name and salary columns\n{projected_df}')
print(f'Aggregation: Total Salary: {total_salary}')
print(f'Number of High Earners (>50000): {high_earners_count}')
print(f'Top 5 Earners: \n{top_5_earners}')
print(f'Skipped DataFrame (First 2 rows skipped): \n{skipped_df}')
print(f'Filtered DataFrame (IT department removed): \n{filtered_df}')

```

5. Run the Script

Execute the Python script by running the following command in your terminal:

```

Sep 21 17:51 srimathy@srimathy-VirtualBox:~ 
[{"name": "David", "salary": 45000, "department": "Sales"}, {"name": "Eve", "salary": 80000, "department": "IT"}]

Filtered JSON file saved successfully.
Projection: Select only name and salary columns
      name  salary
0    Alice   60000
1     Bob   55000
2  Charlie   70000
3   David   45000
4     Eve   80000

Aggregation: Total Salary: 310000
Number of High Earners (>50000): 4
Top 5 Earners:
      name  salary  department
4     Eve   80000        IT
2  Charlie   70000        IT
0   Alice   60000        HR
1     Bob   55000    Finance
3   David   45000    Sales

Skipped DataFrame (First 2 rows skipped):
      name  salary  department
2  Charlie   70000        IT
3   David   45000    Sales
4     Eve   80000        IT

Filtered DataFrame (IT department removed):
      name  salary  department
0   Alice   60000        HR
1     Bob   55000    Finance
3   David   45000    Sales
(myenv) srimathy@srimathy-VirtualBox:~$ 

```

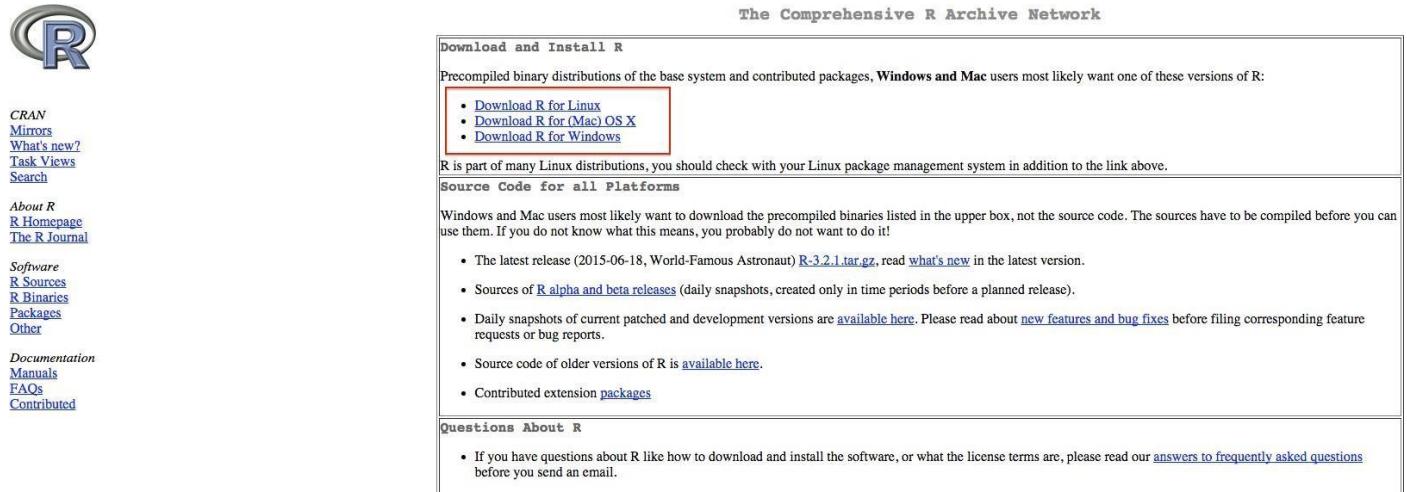
RESULT:

Thus to import a JASON file from the command line and apply the following actions with the data present in the JASON file where, projection, aggregation, remove, count, limit, skip and sort has been executed and verified successfully.

Installation guide for R and RStudio

Step 1 – Install R

1. Download the R installer from <https://cran.r-project.org/>



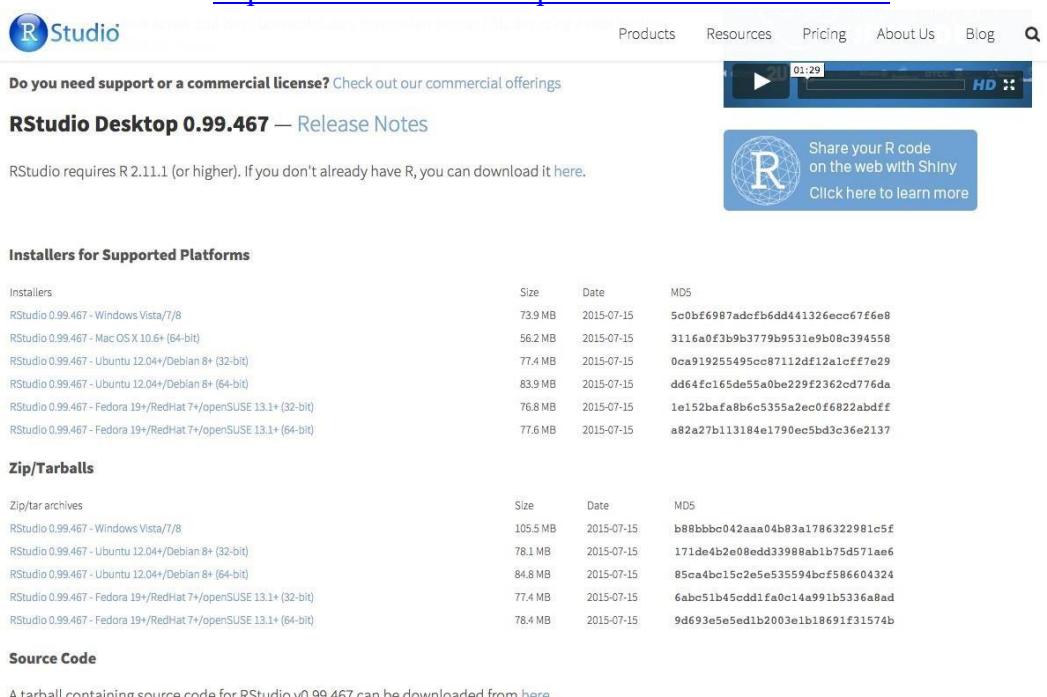
The screenshot shows the CRAN (Comprehensive R Archive Network) homepage. On the left, there's a sidebar with links like 'CRAN Mirrors', 'What's new?', 'Task Views', 'Search', 'About R', 'R Homepage', 'The R Journal', 'Software', 'R Sources', 'R Binaries', 'Packages', 'Other', 'Documentation', 'Manuals', 'FAQs', and 'Contributed'. The main content area has a heading 'Download and Install R' with a sub-section 'Precompiled binary distributions of the base system and contributed packages, Windows and Mac users most likely want one of these versions of R:'. It lists three options: 'Download R for Linux', 'Download R for (Mac) OS X', and 'Download R for Windows'. Below this, it says 'R is part of many Linux distributions, you should check with your Linux package management system in addition to the link above.' A section titled 'Source Code for all Platforms' follows, with a note that Windows and Mac users should download precompiled binaries. It lists several bullet points about the latest release, alpha/beta releases, daily snapshots, and source code for older versions. Another section 'Questions About R' contains a bullet point about frequently asked questions. At the bottom, there's a link to 'What are R and CRAN?'

Figure 1. Screenshot of <http://cran.csiro.au/>

2. Run the installer. Default settings are fine. If you do not have admin rights on your laptop, then ask you local IT support. In that case, it is important that you also ask them to give you full permissions to the R directories. Without this, you will not be able to install additional packages later

Step 2 – Install RStudio

1. Download RStudio: <https://www.rstudio.com/products/rstudio/download/>



The screenshot shows the RStudio download page. At the top, there's a logo for 'R Studio' and navigation links for 'Products', 'Resources', 'Pricing', 'About Us', 'Blog', and a search icon. A video player window is visible in the top right. Below the navigation, a message says 'Do you need support or a commercial license? Check out our commercial offerings' and a link to 'RStudio Desktop 0.99.467 — Release Notes'. A note states 'RStudio requires R 2.11.1 (or higher). If you don't already have R, you can download it here.' To the right, there's a 'Shiny' section with a button 'Click here to learn more'.

Installers for Supported Platforms

Installers	Size	Date	MD5
RStudio 0.99.467 - Windows Vista/7/8	73.9 MB	2015-07-15	5e0bf6987adcfb6dd441326ecc67f6eb
RStudio 0.99.467 - Mac OS X 10.6+ (64-bit)	56.2 MB	2015-07-15	3116a0f3b9b3779b9531e9b08c394558
RStudio 0.99.467 - Ubuntu 12.04+/Debian 8+ (32-bit)	77.4 MB	2015-07-15	0ca919255495cc87112df12a1cff7e29
RStudio 0.99.467 - Ubuntu 12.04+/Debian 8+ (64-bit)	83.9 MB	2015-07-15	dd64fc165de55a0be229f2362cd776da
RStudio 0.99.467 - Fedora 19+/Red Hat 7+/openSUSE 13.1+ (32-bit)	76.8 MB	2015-07-15	1e152bafaf8b6c5355a2ec0f6822abdf
RStudio 0.99.467 - Fedora 19+/Red Hat 7+/openSUSE 13.1+ (64-bit)	77.6 MB	2015-07-15	a82a27b113184e1790ec5bd3c36e2137

Zip/Tarballs

Zip/tar archives	Size	Date	MD5
RStudio 0.99.467 - Windows Vista/7/8	105.5 MB	2015-07-15	b88bbbc042aaa04b83a1786322981c5f
RStudio 0.99.467 - Ubuntu 12.04+/Debian 8+ (32-bit)	78.1 MB	2015-07-15	171de4b2e08edd33988ab1b75d571ae6
RStudio 0.99.467 - Ubuntu 12.04+/Debian 8+ (64-bit)	84.8 MB	2015-07-15	85cadbe15c2e5e535594bcf586604324
RStudio 0.99.467 - Fedora 19+/Red Hat 7+/openSUSE 13.1+ (32-bit)	77.4 MB	2015-07-15	6abc51b45cdd1fa0c14a991b5336a8ad
RStudio 0.99.467 - Fedora 19+/Red Hat 7+/openSUSE 13.1+ (64-bit)	78.4 MB	2015-07-15	9d693e5e5ed1b2003e1b18691f31574b

Source Code

A tarball containing source code for RStudio v0.99.467 can be downloaded from [here](#)

Figure 2. Download RStudio on <https://www.rstudio.com/products/rstudio/download/>

2. Once the installation of R has completed successfully (and not before), run the RStudio installer.
3. If you do not have administrative rights on your laptop, step 2 may fail. Ask your IT Support or download a pre-built zip archive of RStudio which doesn't need installing. The link for this is towards the bottom of the download page, highlighted in Image 2.
 - a. Download the appropriate archive for your system (Windows/Linux only – the Mac version can be installed into your personal “Applications” folder without admin rights).
 - b. Double clicking on the zip archive should automatically unpack it on most Windows machines.

Step 3 – Check that R and RStudio are working

1. Open RStudio. It should open a window that looks similar to image 3 below.
2. In the left hand window, by the ‘>’ sign, type ‘4+5’(without the quotes) and hit enter. An output line reading ‘[1] 9’ should appear. This means that R and RStudio are working.
3. If this is not successful, contact us or your local IT support for further advice

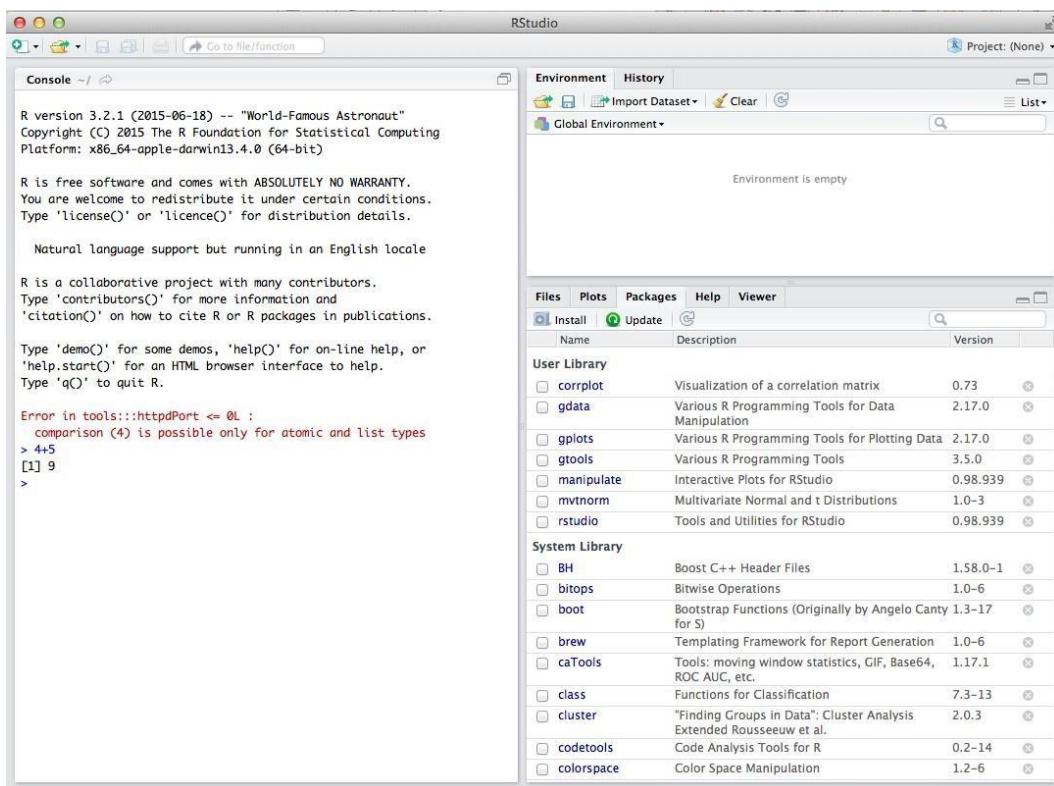


Figure 3. Running R with RStudio

Step 4 – Install R packages required for the workshop

1. Click on the tab ‘ Packages’ then ‘Install’ as shown in Image 4. Or Tools ---> Install packages.
2. Install the following packages: mixOmics **version 6.1.0**, mvtnorm, RColorBrewer, corrplot, igraph (see Image 4). For apple mac users, if you are unable to install the mixOmics imported library rgl, you will need to install the XQuartz software first
<https://www.xquartz.org/>
3. Check that the packages are installed by typing ‘library(mixOmics)’ (without the quotes) in the prompt and press enter (see Image 5).
4. Then type ‘sessionInfo()’ and check that mixOmics version 6.1.0 has been installed (image 6).

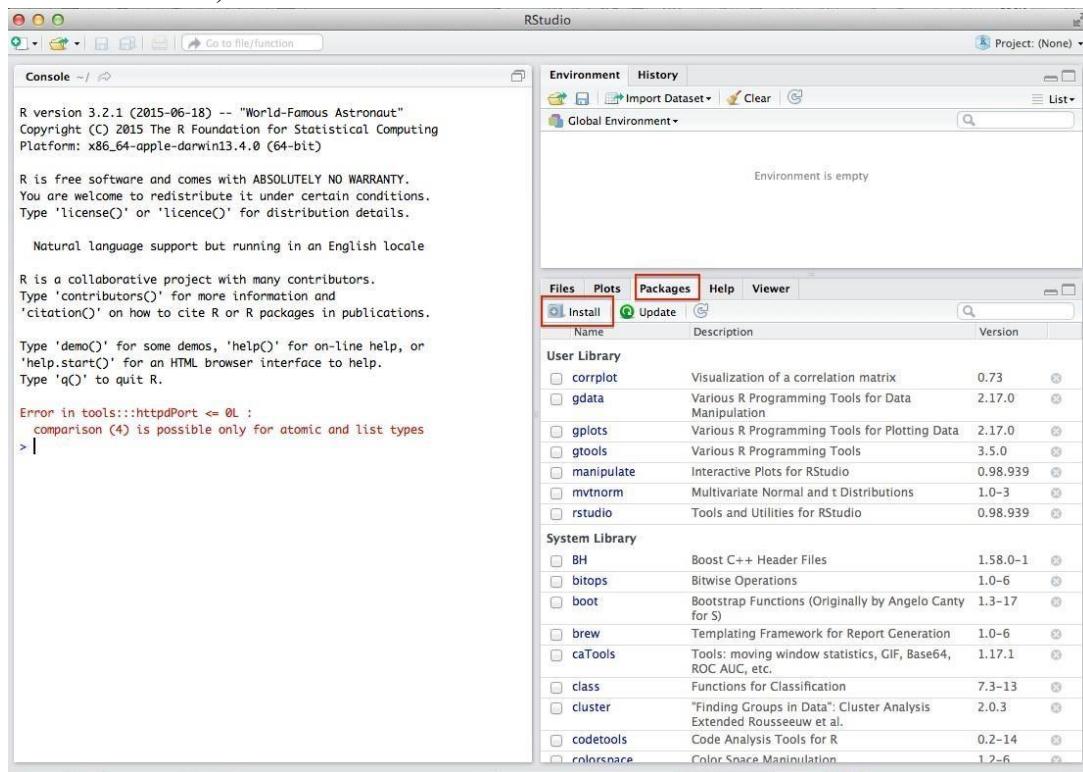


Figure 4. Click on Install to install R packages.

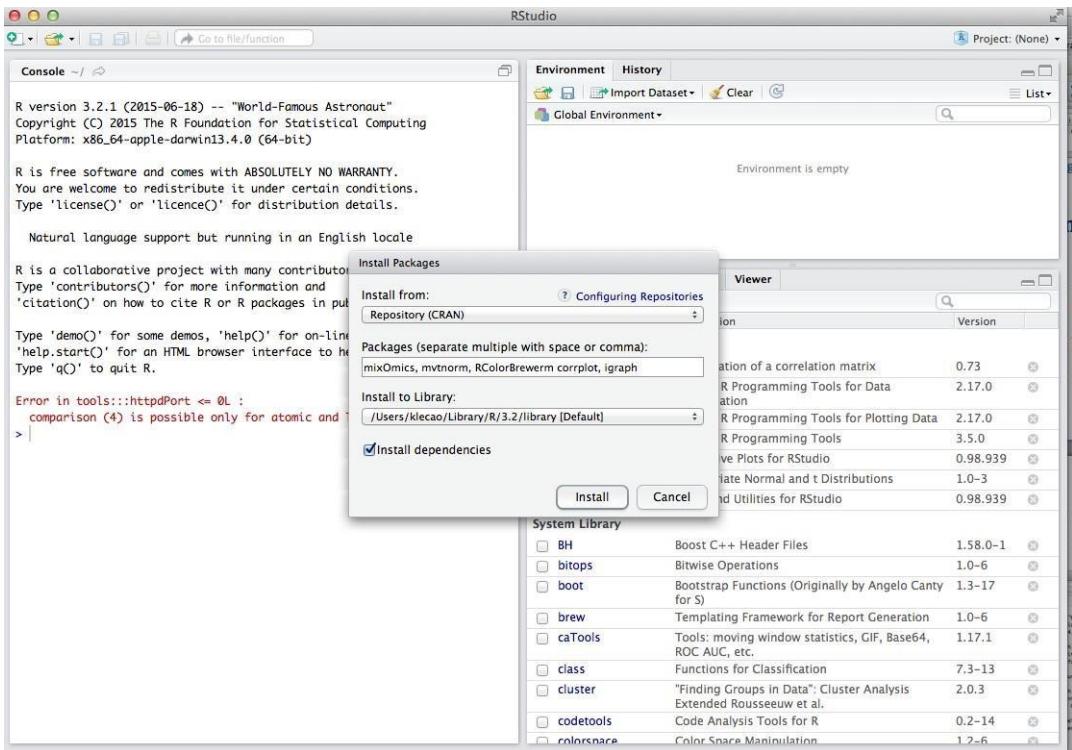


Figure 5. Specify the list of packages to be installed

```

~/Documents/k.lecao/Presentation/2016/INPPO-COST/CaseStudy_Sunflower/Drought - RStudio
Addins ▾
Console ~/Documents/k.lecao/Presentation/2016/INPPO-COST/CaseStudy_Sunflower/Drought/ ▾ Go to file/function ▾ Addins ▾
> 4:5
[1] 9
<library(mixOmics)
Loading required package: MASS
Attaching package: 'MASS'
The following object is masked _by_ '.GlobalEnv':
  genotype
Loading required package: lattice
Loading required package: grid
Loaded mixOmics 6.1.0
Visit http://www.mixOmics.org for more details about our methods.
Any bug reports or comments? Notify us at mixomics at math.univ-toulouse.fr or https://bitbucket.org/klecao/package-mixomi
cs/issues
Thank you for using mixOmics!
> sessionInfo()
R version 3.3.1 beta (2016-06-11 r70764)
Platform: x86_64-apple-darwin13.4.0 (64-bit)
Running under: OS X 10.9.5 (Mavericks)

locale:
[1] en_AU.UTF-8/en_AU.UTF-8/C/en_AU.UTF-8/en_AU.UTF-8

attached base packages:
[1] stats      graphics    grDevices   utils      datasets    methods     base

other attached packages:
[1] mixOmics_6.1.0  gplots_2.2.1.0  lattice_0.20-33 MASS_7.3-45

loaded via a namespace (and not attached):
[1] rgl_0.95.1441  Rcpp_0.12.6     tidyr_0.5.0    corpcor_1.6.8
[5] assertthat_0.1  dplyr_0.5.0    R6_2.1.3       grid_3.3.1
[9] plyr_1.8.4     DBI_0.5       gtable_0.2.0   magrittr_1.5
[13] ellipse_0.3-8  scales_0.4.0   stringi_1.1.1  reshape2_1.4.1
[17] RColorBrewer_1.1-2 tools_3.3.1    stringr_1.1.0  munsell_0.4.3
[21] igraph_1.0.1   parallel_3.3.1 colorspace_1.2-6 tibble_1.1
>

```

The screenshot shows the RStudio interface with the 'Console' tab active. In the console, the user has run the command `library(mixOmics)` and received the response `Loaded mixOmics 6.1.0`. A red oval highlights this message. Below it, the user runs `sessionInfo()` and receives detailed information about the R environment, including the version of mixOmics (3.3.1 beta). Another red oval highlights the mixOmics version in the output. The 'Environment' tab is selected in the top right, showing the global environment with various objects like 'data', 'diabolo.res', 'genotype', 'k', 'kee.genes', 'keep.genes', 'keep.name.genes', 'list.data', etc. The 'Values' section shows the size and type of each object. The 'User Library' section lists all the packages loaded via namespaces, with mixOmics being one of them.

Figure 6. Check that the package mixOmics is installed and has the version 6.1.0.

Exp.No: 7**IMPLEMENT LINEAR AND LOGISTIC REGRESSION****AIM:**

To write an R code to implement linear and logistic regression.

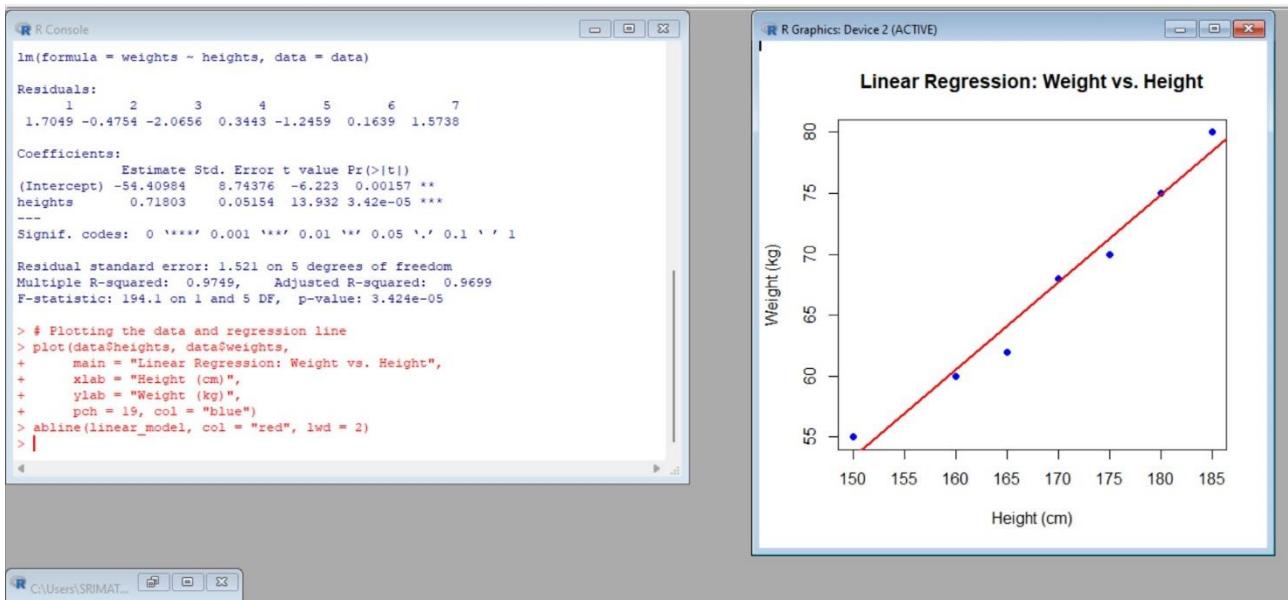
PROCEDURE:

1. Create sample data for heights and weights, fit a linear regression model, and plot the data with the regression line.
2. Use the sample data to create a data frame for the regression model.
3. Fit the linear regression model using the `lm()` function and display the summary.
4. Plot the data points and add the regression line using the `plot()` and `abline()` functions.
5. Load the `mtcars` dataset, convert the 'am' variable to a factor, fit a logistic regression model using the `glm()` function, and plot the probabilities.

PROGRAM CODE:**a) Linear regression**

```
# Linear Regression
heights <- c(150, 160, 165, 170, 175, 180, 185)
weights <- c(55, 60, 62, 68, 70, 75, 80)
data <- data.frame(heights, weights)
linear_model <- lm(weights ~ heights, data = data)
print(summary(linear_model))

# Plotting Linear Regression
plot(data$heights, data$weights,
      main = "Linear Regression: Weight vs. Height",
      xlab = "Height (cm)",
      ylab = "Weight (kg)",
      pch = 19, col = "blue")
abline(linear_model, col = "red", lwd = 2)
```

OUTPUT:**b) Logistic regression**

```
# Logistic Regression
data(mtcars)

mtcars$am <- factor(mtcars$am, levels = c(0, 1), labels = c("Automatic", "Manual"))

logistic_model <- glm(am ~ mpg, data = mtcars, family = binomial)

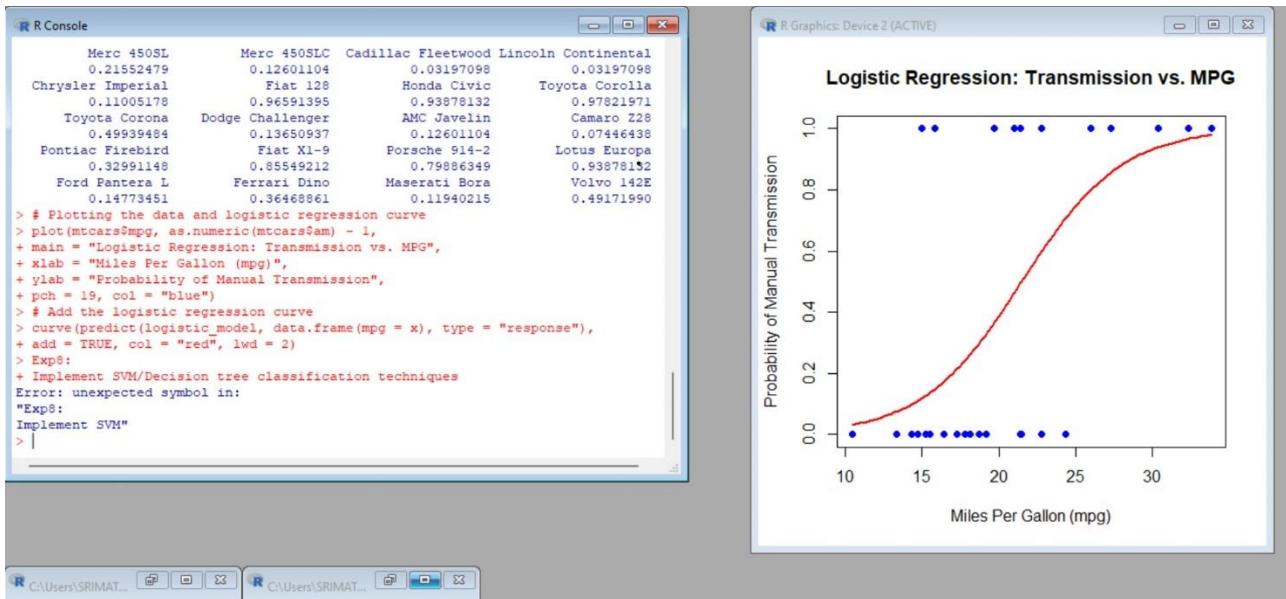
print(summary(logistic_model))
```

```
# Plotting Logistic Regression
predicted_probs <- predict(logistic_model, type = "response")
print(predicted_probs)

plot(mtcars$mpg, as.numeric(mtcars$am) - 1,
     main = "Logistic Regression: Transmission vs. MPG",
     xlab = "Miles Per Gallon (mpg)",
     ylab = "Probability of Manual Transmission",
     pch = 19, col = "blue")

curve(predict(logistic_model, data.frame(mpg = x), type = "response"),
       add = TRUE, col = "red", lwd = 2)
```

OUTPUT:



RESULT:

Thus the R program to implement Linear and Logistic Regression has been executed and verified successfully.

Exp.No: 8

IMPLEMENT SVM/DECISION TREE CLASSIFICATION TECHNIQUES

AIM:

To write an R code to implement SVM/decision tree classification techniques.

PROCEDURE:

1. Install and load the required packages (e1071 for SVM and rpart for Decision Tree) and load the iris dataset.
2. Split the dataset into training (70%) and testing (30%) sets using a reproducible random sampling method.
3. Fit the SVM model with a radial kernel using the training data, print the model summary, and evaluate its performance using a confusion matrix and accuracy calculation.
4. Fit the Decision Tree model using the rpart function with the training data, print the model summary, visualize the tree, and evaluate its performance using a confusion matrix and accuracy calculation.
5. Predict the test set results for both SVM and Decision Tree models and assess their accuracy.

PROGRAM CODE:**a) SVM IN R**

```
# Install and load the e1071 package (if not already installed)
install.packages("e1071")
library(e1071)

# Load the iris dataset
data(iris)

# Inspect the first few rows of the dataset
head(iris)

# Split the data into training (70%) and testing (30%) sets
set.seed(123) # For reproducibility
sample_indices <- sample(1:nrow(iris), 0.7 * nrow(iris))
train_data <- iris[sample_indices, ]
test_data <- iris[-sample_indices, ]

# Fit the SVM model
svm_model <- svm(Species ~ ., data = train_data, kernel = "radial")

# Print the summary of the model
summary(svm_model)
```

```
# Predict the test set
predictions <- predict(svm_model, newdata = test_data)

# Evaluate the model's performance
confusion_matrix <- table(Predicted = predictions, Actual = test_data$Species)
print(confusion_matrix)

# Calculate accuracy
accuracy <- sum(diag(confusion_matrix)) / sum(confusion_matrix)
cat("Accuracy:", accuracy * 100, "%\n")
```

OUTPUT:

```
> # Split the data into training (70%) and testing (30%) sets
> set.seed(123) # For reproducibility
>
> sample_indices <- sample(1:nrow(iris), 0.7 * nrow(iris))
> train_data <- iris[sample_indices, ]
> test_data <- iris[-sample_indices, ]
> # Fit the SVM model
> svm_model <- svm(Species ~ ., data = train_data, kernel = "radial")
Error in svm(Species ~ ., data = train_data, kernel = "radial") :
  could not find function "svm"
> library(e1071)
>
> library(e1071)
> # Fit the SVM model
> svm_model <- svm(Species ~ ., data = train_data, kernel = "radial")
> # Print the summary of the model
> summary(svm_model)

Call:
svm(formula = Species ~ ., data = train_data, kernel = "radial")

Parameters:
  SVM-Type: C-classification
  SVM-Kernel: radial
  cost: 1
```

	setosa	versicolor	virginica
Predicted	14	0	0
Actual	0	17	0
setosa	0	1	13

```
> # Predict the test set
> predictions <- predict(svm_model, newdata = test_data)
> # Evaluate the model's performance
> confusion_matrix <- table(Predicted = predictions, Actual = test_data$Species)
> print(confusion_matrix)

Number of Support Vectors: 45
( 7 18 20 )

Number of classes: 3
Levels:
setosa versicolor virginica
```

b) Decision tree in R

```
# Install and load the rpart package (if not already installed)
install.packages("rpart")
library(rpart)

# Load the iris dataset
data(iris)

# Split the data into training (70%) and testing (30%) sets
set.seed(123) # For reproducibility
sample_indices <- sample(1:nrow(iris), 0.7 * nrow(iris))
train_data <- iris[sample_indices, ]
test_data <- iris[-sample_indices, ]

# Fit the Decision Tree model
tree_model <- rpart(Species ~ ., data = train_data, method = "class")

# Print the summary of the model
summary(tree_model)

# Plot the Decision Tree
plot(tree_model)
```

```

text(tree_model, pretty = 0)

# Predict the test set
predictions <- predict(tree_model, newdata = test_data, type = "class")

# Evaluate the model's performance
confusion_matrix <- table(Predicted = predictions, Actual = test_data$Species)
print(confusion_matrix)

# Calculate accuracy
accuracy <- sum(diag(confusion_matrix)) / sum(confusion_matrix)
cat("Accuracy:", accuracy * 100, "%\n")

```

OUTPUT:

```

package 'rpart' successfully unpacked and MD5 sums checked
The downloaded binary packages are in
  C:/Users/SRIMATHY.R/AppData/Local/Temp/RtmpGs71Wc/downloaded_packages
> library(rpart)
> # Load the iris dataset
> data(iris)
> # Split the data into training (70%) and testing (30%) sets
> set.seed(123) # For reproducibility
> sample_indices <- sample(1:nrow(iris), 0.7 * nrow(iris))
> train_data <- iris[sample_indices, ]
> test_data <- iris[-sample_indices, ]
> # Fit the Decision Tree model
> tree_model <- rpart(Species ~ ., data = train_data, method = "class")
> # Print the summary of the model
> summary(tree_model)
Call:
rpart(formula = Species ~ ., data = train_data, method = "class")
n= 105

Surrogate splits:
  Petal.Length < 4.75 to the left, agree=0.913, adj=0.824, (0 split)
  Sepal.Length < 6.15 to the left, agree=0.696, adj=0.382, (0 split)
  Sepal.Width < 2.65 to the left, agree=0.638, adj=0.265, (0 split)

Node number 6: 35 observations
  predicted class=versicolor expected loss=0.1142857 P(node) =0.3333333
  class counts: 0 31 4
  probabilities: 0.000 0.886 0.114

Node number 7: 34 observations
  predicted class=virginica expected loss=0.02941176 P(node) =0.3238095
  class counts: 0 1 33
  probabilities: 0.000 0.029 0.971

> # Plot the Decision Tree
> plot(tree_model)
> text(tree_model, pretty = 0)
> # Predict the test set
> predictions <- predict(tree_model, newdata = test_data, type = "class")
> # Evaluate the model's performance
> confusion_matrix <- table(Predicted = predictions, Actual = test_data$Species)
> print(confusion_matrix)
  Actual
Predicted   setosa versicolor virginica
  setosa      14          0          0
  versicolor     0         18          1
  virginica      0          0         12
> # Calculate accuracy
> accuracy <- sum(diag(confusion_matrix)) / sum(confusion_matrix)
> cat("Accuracy:", accuracy * 100, "%\n")
Accuracy: 97.7778 %

```

```

CP nsplit rel error xerror xstd
1 0.5294118    0 1.0000000  1.2058824 0.06232572
2 0.3970588    1 0.47058824 0.5441176 0.07198662
3 0.0100000    2 0.07352941 0.1176471 0.03997857

Variable importance
Petal.Width Petal.Length Sepal.Length Sepal.Width
            34           32           21           13

Node number 1: 105 observations, complexity param=0.5294118
predicted class=virginica expected loss=0.647619 P(node) =1
  class counts: 36 32 37
  probabilities: 0.343 0.305 0.352
  left son=2 (36 obs), right son=3 (69 obs)
Primary splits:
  Petal.Length < 2.45 to the left,  improve=35.54783, (0 missing)
  Petal.Width < 0.8 to the left,  improve=35.54783, (0 missing)
  Sepal.Length < 5.45 to the left,  improve=24.79179, (0 missing)
  Sepal.Width < 3.25 to the right,  improve=12.34670, (0 missing)
Surrogate splits:
  Petal.Width < 0.8 to the left,  agree=1.000, adj=1.000, (0 split)
  Sepal.Length < 5.45 to the left,  agree=0.924, adj=0.778, (0 split)
  Sepal.Width < 3.25 to the right,  agree=0.819, adj=0.472, (0 split)

Node number 2: 36 observations
predicted class=setosa expected loss=0 P(node) =0.3428571
  class counts: 36 0 0
  probabilities: 1.000 0.000 0.000

Node number 3: 69 observations, complexity param=0.3970588
predicted class=virginica expected loss=0.4637681 P(node) =0.6571429

```

RESULT:

Thus the R program to implement SVM/decision tree classification techniques has been executed and verified successfully.

Exp.No: 9**IMPLEMENT CLUSTERING TECHNIQUES – HIERARCHICAL AND KMEANS****AIM:**

To write an R code to implement hierarchical and k-means clustering techniques.

PROCEDURE:

1. Load the iris dataset and use only the numeric columns for clustering by excluding the Species column.
2. Standardize the data to ensure all variables have equal weight in the clustering process.
3. Compute the distance matrix using the Euclidean method and perform hierarchical clustering using the "complete" linkage method, plot the dendrogram, and cut the tree to form 3 clusters.
4. Perform K-means clustering by setting the number of clusters, run the clustering algorithm, and add cluster assignments to the original dataset.
5. Display the first few rows of the updated dataset and plot the clusters using ggplot2 for visualization.

PROGRAM CODE:**a) HIERARCHIAL CLUSTERING**

```
# Load the iris dataset
data(iris)

# Use only the numeric columns for clustering (exclude the Species column)
iris_data <- iris[, -5]

# Standardize the data
iris_scaled <- scale(iris_data)

# Compute the distance matrix
distance_matrix <-
dist(iris_scaled, method = "euclidean")

# Perform hierarchical clustering using the "complete" linkage method
hc_complete <- hclust(distance_matrix, method = "complete")

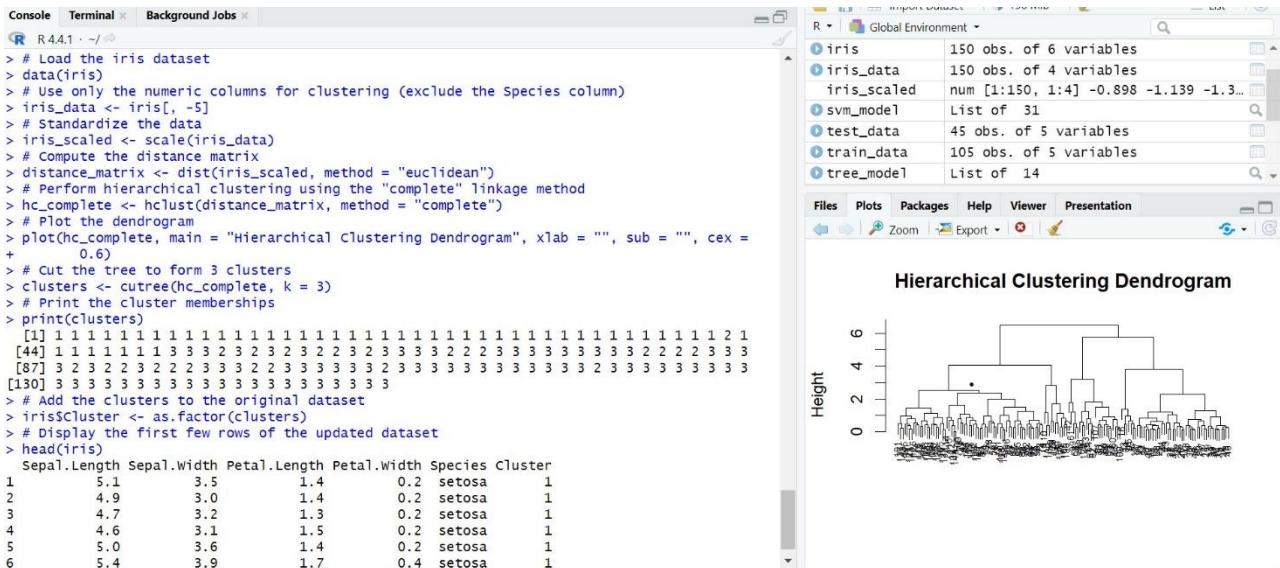
# Plot the dendrogram
plot(hc_complete, main = "Hierarchical Clustering Dendrogram",
      xlab = "", sub = "", cex =
      0.6)

# Cut the tree to form 3 clusters
clusters <- cutree(hc_complete, k = 3)

# Print the cluster memberships
print(clusters)
```

```
# Add the clusters to the original dataset
iris$Cluster <- as.factor(clusters)
# Display the first few rows of the updated dataset
head(iris)
```

OUTPUT:



b) K-MEANS CLUSTERING

```
# Load the iris dataset
data(iris)

# Use only the numeric columns for clustering (exclude the Species column)
iris_data <- iris[, -5]

# Standardize the data
iris_scaled <- scale(iris_data)

# Set the number of clusters
set.seed(123) # For reproducibility
k <- 3 # Number of clusters

# Perform K-Means clustering
kmeans_result <- kmeans(iris_scaled, centers = k, nstart = 25)

# Print the K-Means result
print(kmeans_result)
```

```

# Print the cluster centers
print(kmeans_result$centers)

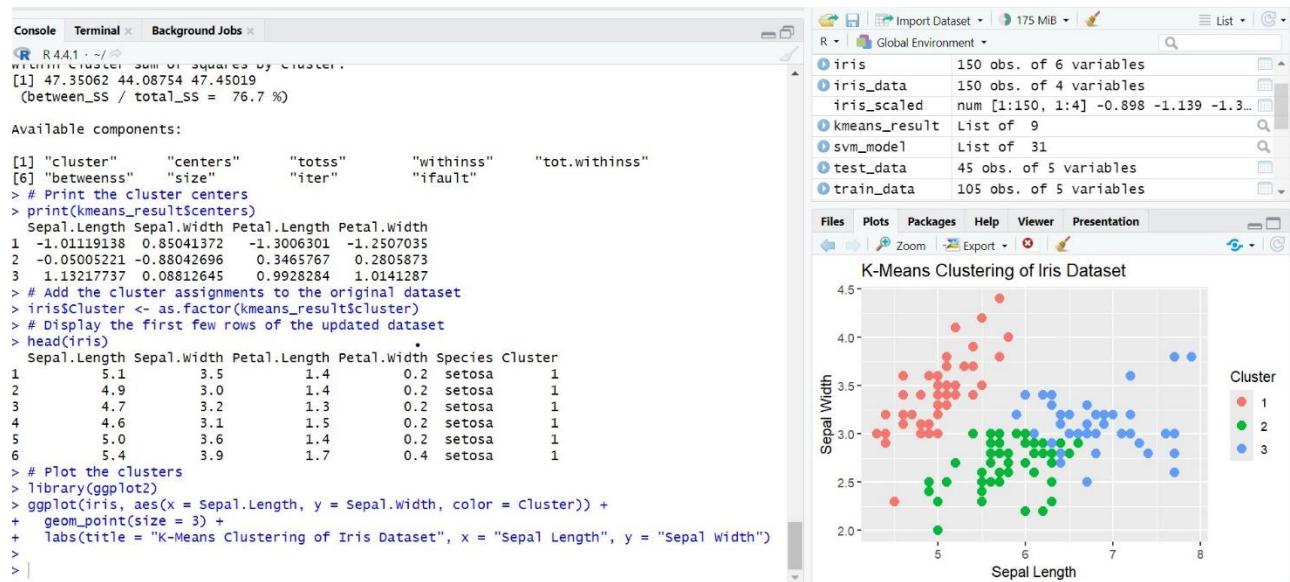
# Add the cluster assignments to the original dataset
iris$Cluster <- as.factor(kmeans_result$cluster)

# Display the first few rows of the updated dataset
head(iris)

# Plot the clusters
library(ggplot2)
ggplot(iris, aes(x = Sepal.Length, y = Sepal.Width, color = Cluster)) +
  geom_point(size = 3) +
  labs(title = "K-Means Clustering of Iris Dataset", x = "Sepal Length", y = "Sepal Width")

```

OUTPUT:



RESULT:

Thus the R program to implement hierarchical and k-means clustering techniques has been executed and verified successfully.

Exp.No:10**VISUALIZE DATA USING ANY PLOTTING FRAMEWORK****AIM:**

To write an R code to visualize data using plotting framework such as scatter plot, bar char, histogram and box plot.

PROCEDURE:

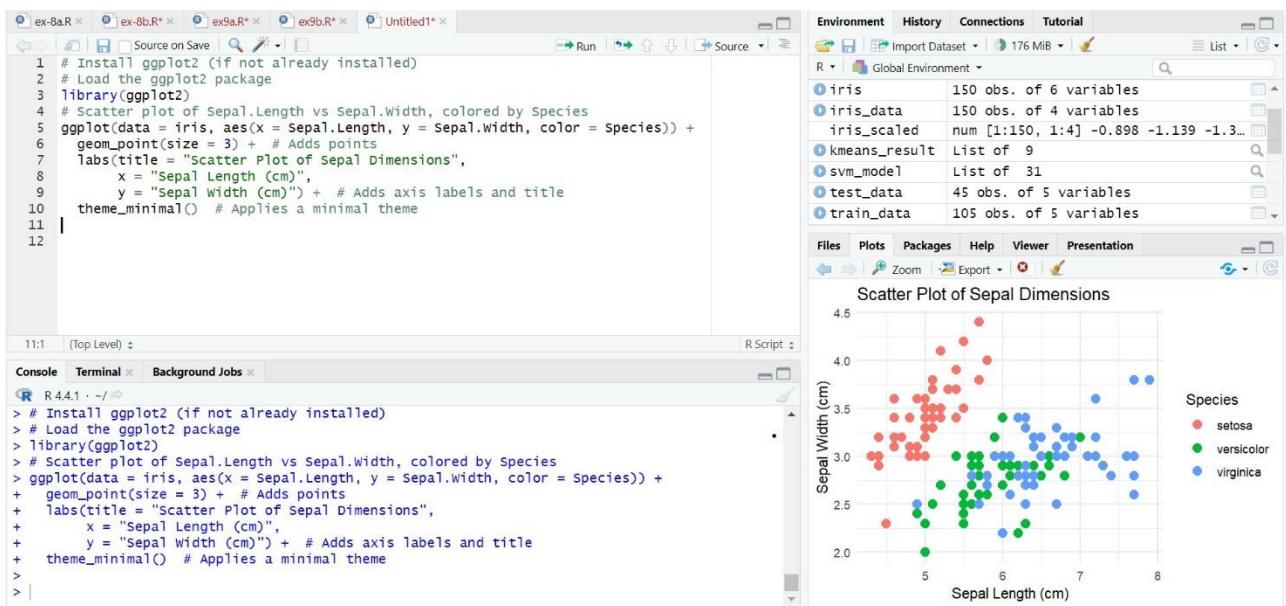
1. Install and Load ggplot2: Ensure the ggplot2 package is installed and loaded to use its plotting functions.
2. Scatter Plot: Create a scatter plot of Sepal Length vs. Sepal Width, colored by Species, to visualize the relationship between these two variables across different species in the iris dataset.
3. Bar Chart: Generate a bar chart to show the count of different Species in the iris dataset, using bars filled with a specified color to represent the counts.
4. Histogram: Create a histogram of Sepal Length to visualize the frequency distribution of this variable within the dataset, specifying the bin width and colors for the histogram bars.
5. Box Plot: Plot a box plot of Sepal Length for each Species to compare the distribution and central tendency of Sepal Length across the different species in the dataset.

1) SCATTER PLOT

```
# Install ggplot2 (if not already installed)
install.packages("ggplot2")

# Load the ggplot2 package
library(ggplot2)

# Scatter plot of Sepal.Length vs Sepal.Width, colored by Species
ggplot(data = iris, aes(x = Sepal.Length, y = Sepal.Width, color = Species)) +
  geom_point(size = 3) + # Adds points
  labs(title = "Scatter Plot of Sepal Dimensions",
       x = "Sepal Length (cm)",
       y = "Sepal Width (cm)") + # Adds axis labels and title
  theme_minimal() # Applies a minimal theme
```

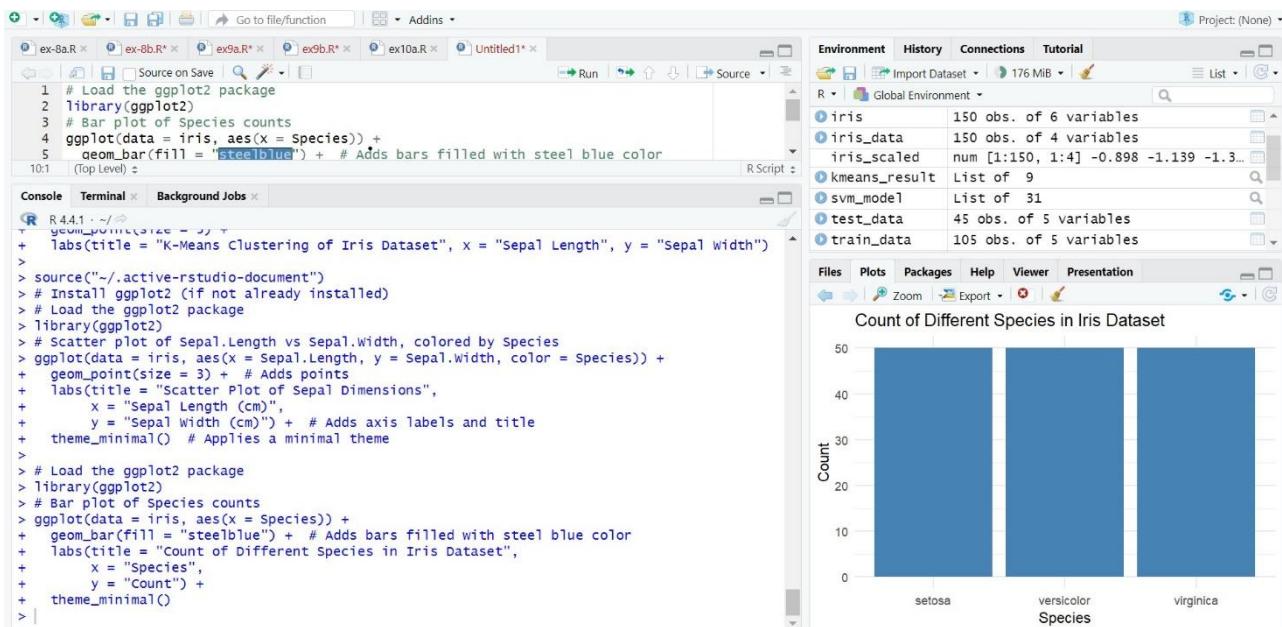
OUTPUT:**2) BAR CHART**

```
# Install ggplot2 (if not already installed)
install.packages("ggplot2")

# Load the ggplot2 package
library(ggplot2)

# Bar plot of Species counts
ggplot(data = iris, aes(x = Species)) +
  geom_bar(fill = "steelblue") + # Adds bars filled with steel blue color
  labs(title = "Count of Different Species in Iris Dataset",
       x = "Species",
       y = "Count") +
  theme_minimal()
```

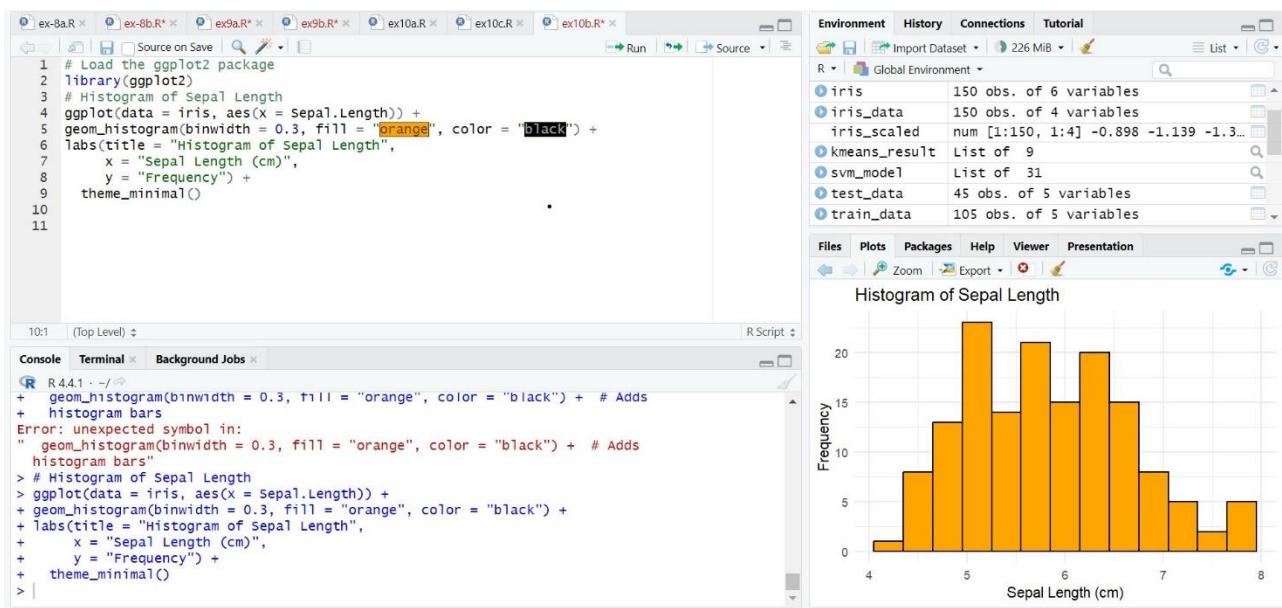
OUTPUT:



3) HISTOGRAM

```
# Histogram of Sepal Length
ggplot(data = iris, aes(x = Sepal.Length)) +
  geom_histogram(binwidth = 0.3, fill = "orange", color = "black") + # Adds histogram bars
  labs(title = "Histogram of Sepal Length",
       x = "Sepal Length (cm)",
       y = "Frequency") +
  theme_minimal()
```

OUTPUT:

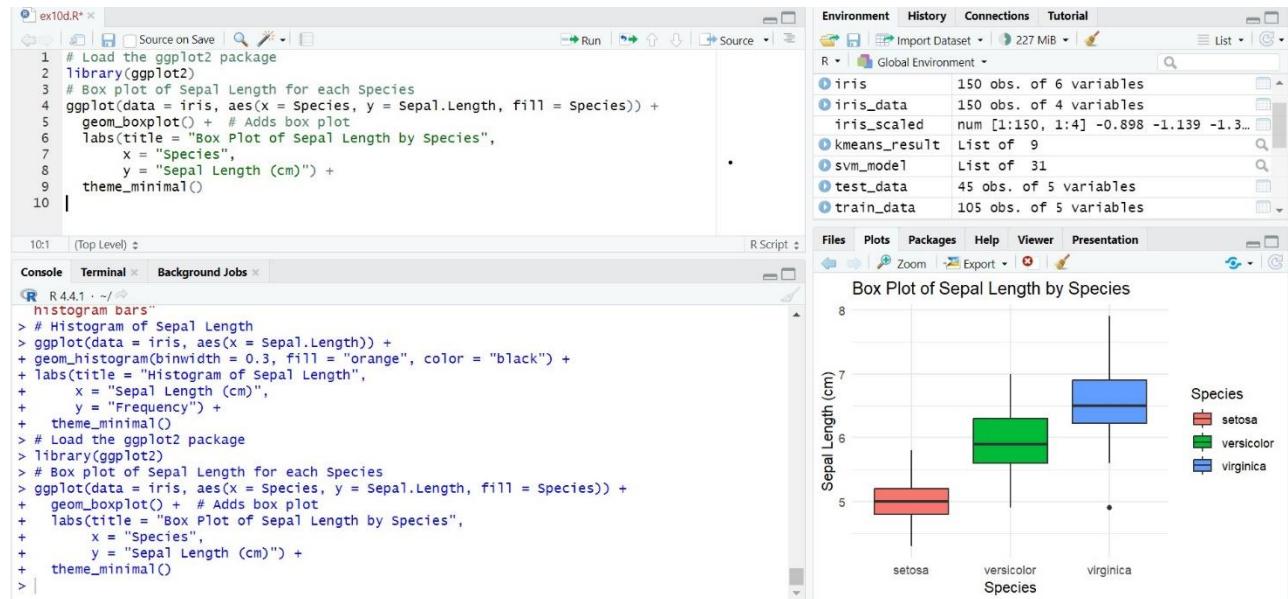


4) BOX PLOT

```
# Box plot of Sepal Length for each Species
ggplot(data = iris, aes(x = Species, y = Sepal.Length, fill = Species)) +
  geom_boxplot() + # Adds box plot
  labs(title = "Box Plot of Sepal Length by Species",
       x = "Species",
```

```
y = "Sepal Length (cm)") +
theme_minimal()
```

OUTPUT:



RESULT:

Thus the R program to visualize data using plotting framework such as scatter plot, bar char, histogram and box plot has been executed and verified successfully.