

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:

```
suryaa@suryaa-HP-Pavilion-Laptop-15-eg2xxx:~$ java -version
openjdk version "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)
```

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
```

```
suryaa@suryaa-HP-Pavilion-Laptop-15-eg2xxx:~$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/suryaa/.ssh/id_rsa): /home/suryaa/hadoop
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/suryaa/hadoop
Your public key has been saved in /home/suryaa/hadoop.pub
The key fingerprint is:
SHA256:V3x8wE8DAjBYgUVK9siBaMvB9dKb5G9YWbg0h0v8Wt0 suryaa@suryaa-HP-Pavilion-Laptop-15-eg2xxx
The key's randomart image is:
+---[RSA 3072]----+
|.. o...+*Bo... oo |
|= .*o= o ....o.|
|o o. B o . o ooo|
| o = + + . . . .|
| 0 =S .           |
| . X ...         |
| o * . E         |
| +               |
| .               |
+---[SHA256]-----+
```

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:

```
suryaa@suryaa-HP-Pavilion-Laptop-15-eg2xxx:~$ ssh localhost
Welcome to Ubuntu 24.04 LTS (GNU/Linux 6.8.0-39-generic x86_64)

 * Documentation: https://help.ubuntu.com
 * Management: https://landscape.canonical.com
 * Support: https://ubuntu.com/pro

Expanded Security Maintenance for Applications is not enabled.

337 updates can be applied immediately.
185 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

8 additional security updates can be applied with ESM Apps.
Learn more about enabling ESM Apps service at https://ubuntu.com/esm

Last login: Mon Aug 12 23:07:04 2024 from 127.0.0.1
```

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

```
suryaa@suryaa-HP-Pavilion-Laptop-15-eg2xxx:~$ ls
'~.'                                mapper2.py          pig_1726204304999.log
'${system:java.io.tmpdir}'           mapper.py          pig_venv
data.txt                             metastore_db      Public
derby.log                            Music             reducer1.py
Desktop                            my_udf.py        script.pig
Documents                           Pictures         snap
Downloads                           pig               Templates
exp3                               pig_1726202834701.log
hadoop                            pig_1726203195922.log
hadoop-3.4.0                         pig_1726203409296.log
hadoop.pub                          pig_1726203590154.log
input.txt                           pig_1726203793878.log
mapper1.py                          pig_1726204274510.log
```

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:

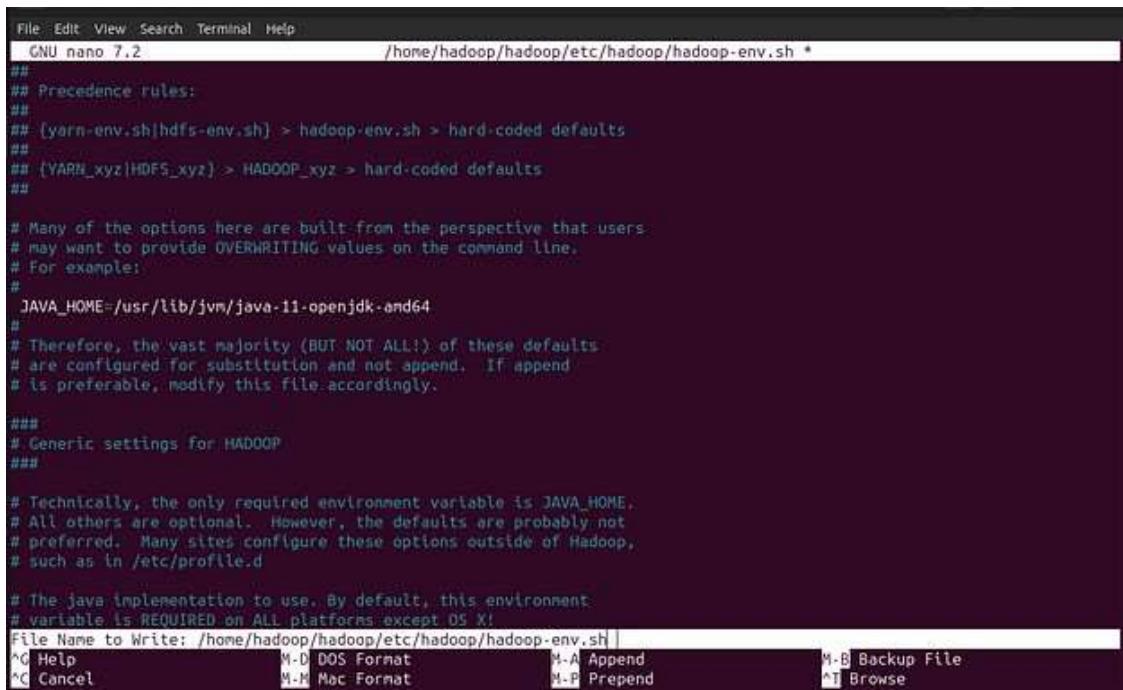
\$ source ~/.bashrc

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 ]
^G Help M-D DOS Format M-A Append M-B Backup File
^C Cancel M-M Mac Format M-P Prepend ^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

```
suryaa@suryaa-HP-Pavilion-Laptop-15-eg2xxx:~$ start-dfs.sh
Starting namenodes on [localhost]
Starting datanodes
Starting secondary namenodes [suryaa-HP-Pavilion-Laptop-15-eg2xxx]
suryaa@suryaa-HP-Pavilion-Laptop-15-eg2xxx:~$ jps
24304 SecondaryNameNode
23862 NameNode
24060 DataNode
24479 Jps
suryaa@suryaa-HP-Pavilion-Laptop-15-eg2xxx:~$ start-yarn.sh
Starting resourcemanager
Starting nodemanagers
```

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

\$ jps

```
suryaa@suryaa-HP-Pavilion-Laptop-15-eg2xxx:~$ jps
24304 SecondaryNameNode
23862 NameNode
24761 NodeManager
24601 ResourceManager
24060 DataNode
25231 Jps
```

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**](http://192.168.1.6:9870)

The screenshot shows the HDFS Health Overview page at localhost:9870/dfshealth.html#tab-overview. The top navigation bar includes tabs for Hadoop, Overview, Datanodes, Datanode Volume Failures, Snapshot, Startup Progress, and Utilities. The Overview tab is selected. Below the tabs, there's a summary section with the title "Overview 'localhost:9000' (✓active)". A table provides detailed information about the cluster's start time, version, compilation details, cluster ID, and block pool ID.

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-1797801860-127.0.1.1-1725252549180

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-1797801860-127.0.1.1-1725252549180

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: 11.4 GB (46.2%)	

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>

The screenshot shows the Hadoop Resource Manager interface at <http://192.168.1.6:8088/cluster>. The left sidebar includes links for Cluster (About, Nodes, Node Labels, Applications, Scheduler), Tools, and Help. The main content area displays Cluster Metrics, Cluster Nodes Metrics, and Scheduler Metrics. The Scheduler Metrics section shows the Capacity Scheduler configuration, including the scheduling resource type [memory-mb (unit=Mi), vcores] and minimum memory requirement <memory:1024, vCore:1>. A table below lists application entries, and a note at the bottom indicates 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:

```
suryaa@suryaa-HP-Pavilion-Laptop-15-eg2xxx:~$ hdfs dfs -ls /
Found 2 items
drwxr-wx-wx  - suryaa supergroup          0 2024-09-20 23:22 /tmp
drwxr-xr-x  - suryaa supergroup          0 2024-09-20 23:20 /user
```

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:

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

Step 15 – Stop Hadoop Cluster

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

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

```
suryaa@suryaa-HP-Pavilion-Laptop-15-eg2xxx:~$ stop-all.sh
WARNING: Stopping all Apache Hadoop daemons as suryaa in 10 seconds.
WARNING: Use CTRL-C to abort.
Stopping namenodes on [localhost]
Stopping datanodes
Stopping secondary namenodes [suryaa-HP-Pavilion-Laptop-15-eg2xxx]
Stopping nodemanagers
Stopping resourcemanager
```

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.

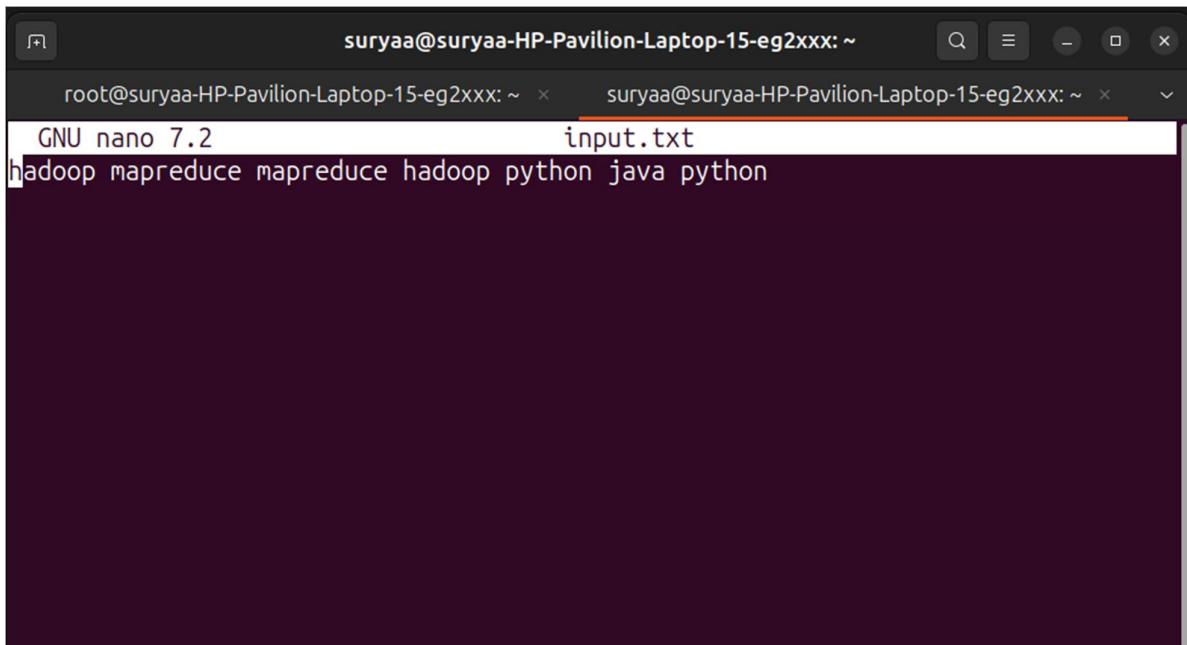
Procedure:**Step 1: Create Data File:**

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



```
suryaa@suryaa-HP-Pavilion-Laptop-15-eg2xxx: ~
root@suryaa-HP-Pavilion-Laptop-15-eg2xxx: ~ x suryaa@suryaa-HP-Pavilion-Laptop-15-eg2xxx: ~ x
GNU nano 7.2           input.txt
hadoop mapreduce mapreduce hadoop python java python
```

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
```

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
```

```
suryaa@suryaa-HP-Pavilion-Laptop-15-eg2xxx:~$ hdfs dfs -cat /output/part-00000
hadoop 2
java 1
mapreduce 2
python 2
```

Result:

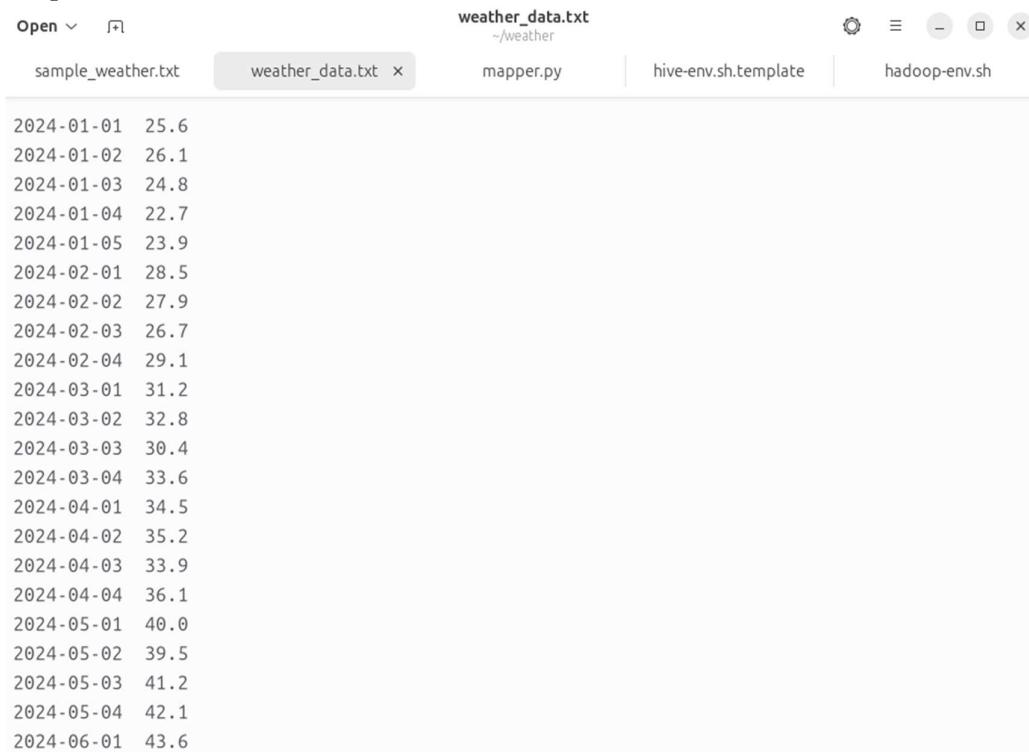
Thus, the program for basic Word Count Map Reduce has been executed successfully.

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:**

The screenshot shows a terminal window with the following details:

- File path: `~/weather`
- File name: `weather_data.txt`
- Content of the file:

```
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 go through the shuffle process 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

```
suryaa@suryaa-HP-Pavilion-Laptop-15-eg2xxx:~$ chmod 777 mapper1.py
suryaa@suryaa-HP-Pavilion-Laptop-15-eg2xxx:~$ chmod 777 reducer1.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
```

```
suryaa@suryaa-HP-Pavilion-Laptop-15-eg2xxx:~$ hadoop jar $HADOOP_HOME/share/hado
op/tools/lib/hadoop-streaming-3.4.0.jar -input /input -output /output
-mapper ./mapper2.py -reducer ./reducer1.py
packageJobJar: [/tmp/hadoop-unjar1557825331205723713/] [] /tmp/streamjob84806232
30645271256.jar tmpDir=null
2024-09-21 00:00:55,349 INFO client.DefaultNoHARMFailoverProxyProvider: Connecti
ng to ResourceManager at /0.0.0.0:8032
2024-09-21 00:00:55,448 INFO client.DefaultNoHARMFailoverProxyProvider: Connecti
ng to ResourceManager at /0.0.0.0:8032
2024-09-21 00:00:55,643 INFO mapreduce.JobResourceUploader: Disabling Erasure Co
ding for path: /tmp/hadoop-yarn/staging/suryaa/.staging/job_1726856840842_0002
2024-09-21 00:00:55,937 INFO mapred.FileInputFormat: Total input files to proces
s : 1
2024-09-21 00:00:56,013 INFO mapreduce.JobSubmitter: number of splits:2
2024-09-21 00:00:56,546 INFO mapreduce.JobSubmitter: Submitting tokens for job:
job_1726856840842_0002
2024-09-21 00:00:56,546 INFO mapreduce.JobSubmitter: Executing with tokens: []
2024-09-21 00:00:56,713 INFO conf.Configuration: resource-types.xml not found
2024-09-21 00:00:56,714 INFO resource.ResourceUtils: Unable to find 'resource-t
ypes.xml'.
```

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
```

```
Bytes Read=159
File Output Format Counters
  Bytes Written=32
2024-09-12 22:21:38,945 INFO streaming.StreamJob: Output directory: /output/average_temperature
suryaa@suryaa-HP-Pavilion-Laptop-15-eg2xxx:~$ hdfs dfs -cat /output/average_temperature/part-00000
2024-09-01      31.0
2024-09-02      28.5
```

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=$PIG_HOME/confexport JAVA_HOME=/usr/lib/jvm/java-8-openjdkam64export PIG_CLASSPATH=$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
```

```

suryaa@suryaa-HP-Pavilion-Laptop-15-eg2xxx:~$ start-all.sh
WARNING: Attempting to start all Apache Hadoop daemons as suryaa 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 [suryaa-HP-Pavilion-Laptop-15-eg2xxx]
Starting resourcemanager
Starting nodemanagers
suryaa@suryaa-HP-Pavilion-Laptop-15-eg2xxx:~$ jps
9361 SecondaryNameNode
10179 Jps
9124 DataNode
8934 NameNode
9788 NodeManager
9630 ResourceManager

```

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

```
suryaa@suryaa-HP-Pavilion-Laptop-15-eg2xxx:~$ pig
2024-09-21 08:25:55,958 INFO pig.ExecTypeProvider: Trying ExecType : LOCAL
2024-09-21 08:25:55,959 INFO pig.ExecTypeProvider: Trying ExecType : MAPREDUCE
2024-09-21 08:25:55,959 INFO pig.ExecTypeProvider: Picked MAPREDUCE as the ExecT
ype
2024-09-21 08:25:56,004 [main] INFO org.apache.pig.Main - Apache Pig version 0.
17.0 (r1797386) compiled Jun 02 2017, 15:41:58
2024-09-21 08:25:56,004 [main] INFO org.apache.pig.Main - Logging error message
s to: /home/suryaa/pig_1726887355997.log
2024-09-21 08:25:56,026 [main] INFO org.apache.pig.impl.util.Utils - Default bo
otup file /home/suryaa/.pigbootup not found
2024-09-21 08:25:56,238 [main] INFO org.apache.hadoop.conf.Configuration.deprec
ation - mapred.job.tracker is deprecated. Instead, use mapreduce.jobtracker.add
ress
2024-09-21 08:25:56,238 [main] INFO org.apache.pig.backend.hadoop.executionengi
ne.HExecutionEngine - Connecting to hadoop file system at: hdfs://localhost:9000
2024-09-21 08:25:56,592 [main] INFO org.apache.pig.PigServer - Pig Script ID fo
r the session: PIG-default-9c5ce4bb-377d-4e2c-9e3b-ca0eaaf3e2bf
2024-09-21 08:25:56,592 [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,Suryaa

2,CSE

3,Hadoop

4,Pig

5,udf

```
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
```

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

```
suryaa@suryaa-HP-Pavilion-Laptop-15-eg2xxx:~/Downloads/pig-0.17.0$ pig -x local /home/suryaa/Downloads/pig-0.17.0/script.pig
2024-09-21 08:33:10,565 INFO pig.ExecTypeProvider: Trying ExecType : LOCAL
2024-09-21 08:33:10,565 INFO pig.ExecTypeProvider: Picked LOCAL as the ExecType
2024-09-21 08:33:10,610 [main] INFO org.apache.pig.Main - Apache Pig version 0.17.0 (r1797386) compiled Jun 02 2017, 15:41:58
2024-09-21 08:33:10,610 [main] INFO org.apache.pig.Main - Logging error messages to: /home/suryaa/Downloads/pig-0.17.0/pig_17...
2024-09-21 08:33:10,631 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - user.name is deprecated. Instead, use mapreduce.job.user.name
2024-09-21 08:33:10,737 [main] INFO org.apache.pig.impl.util.Utils - Default bootup file /home/suryaa/.pigbootup not found
2024-09-21 08:33:10,773 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - mapred.job.tracker is deprecated. Instead, use mapreduce.job.tracker
2024-09-21 08:33:10,775 [main] INFO org.apache.pig.backend.hadoop.executionengine.HExecutionEngine - Connecting to hadoop file system at /tmp
2024-09-21 08:33:10,796 [main] INFO org.apache.pig.PigServer - Pig Script ID for the session: PIG-script.pig-d9bd4342-7042-4a...
2024-09-21 08:33:10,796 [main] WARN org.apache.pig.PigServer - ATS is disabled since yarn.timeline-service.enabled set to false
2024-09-21 08:33:10,843 [main] INFO org.apache.pig.scripting.jython.JythonScriptEngine - created tmp python.cachedir=/tmp/pig...
2024-09-21 08:33:13,135 [main] INFO org.apache.pig.scripting.jython.JythonScriptEngine - Register scripting UDF: my_udf.to_upper
2024-09-21 08:33:13,501 [main] INFO org.apache.pig.scripting.jython.JythonFunction - No schema defined for function 'to_upper'
2024-09-21 08:33:13,571 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - mapred.textoutputformat.separator is de...
2024-09-21 08:33:13,586 [main] ERROR org.apache.pig.tools.grunt.Grunt - ERROR 6000: <file:/home/suryaa/Downloads/pig-0.17.0/sc...
0> Output Location Validation Failed for: 'file:///home/suryaa/Downloads/pig-0.17.0/output' More info to follow:
Output directory file:/home/suryaa/Downloads/pig-0.17.0/pig_1726887790608.log
Details at logfile: /home/suryaa/Downloads/pig-0.17.0/pig_1726887790608.log
2024-09-21 08:33:13,609 [main] INFO org.apache.pig.Main - Pig script completed in 3 seconds and 171 milliseconds (3171 ms)
```

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-m-00000

```
suryaa@suryaa-HP-Pavilion-Laptop-15-eg2xxx:~/Downloads/pig-0.17.0/output$ cat /home/suryaa/Downloads/pig-0.17.0/output/part-m-00000
SURYAA
CSE
HADOOP
PIG
UDF
suryaa@suryaa-HP-Pavilion-Laptop-15-eg2xxx:~/Downloads/pig-0.17.0/output$
```

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
```

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

```
suryaa@suryaa-HP-Pavilion-Laptop-15-eq2xxx:~/Downloads$ tar xvf apache-hive-3.1.3-bin.tar.gz
apache-hive-3.1.3-bin/LICENSE
apache-hive-3.1.3-bin/RELEASE_NOTES.txt
apache-hive-3.1.3-bin/NOTICE
apache-hive-3.1.3-bin/binary-package-licenses/com.thoughtworks.paranamer-LICENSE
apache-hive-3.1.3-bin/binary-package-licenses/org.codehaus.janino-LICENSE
apache-hive-3.1.3-bin/binary-package-licenses/org.jamon.jamon-runtime-LICENSE
apache-hive-3.1.3-bin/binary-package-licenses/org.mozilla.rhino-LICENSE
apache-hive-3.1.3-bin/binary-package-licenses/org.jruby-LICENSE
apache-hive-3.1.3-bin/binary-package-licenses/jline-LICENSE
apache-hive-3.1.3-bin/binary-package-licenses/org.antlr-LICENSE
apache-hive-3.1.3-bin/binary-package-licenses/org.abego.treelayout.core-LICENSE
apache-hive-3.1.3-bin/binary-package-licenses/sqlline-LICENSE
apache-hive-3.1.3-bin/binary-package-licenses/org.slf4j-LICENSE
apache-hive-3.1.3-bin/binary-package-licenses/org.antlr.antlr4-LICENSE
apache-hive-3.1.3-bin/binary-package-licenses/com.google.protobuf-LICENSE
apache-hive-3.1.3-bin/binary-package-licenses/com.sun.jersey-LICENSE
apache-hive-3.1.3-bin/binary-package-licenses/javolution-LICENSE
apache-hive-3.1.3-bin/binary-package-licenses/NOTICE
apache-hive-3.1.3-bin/binary-package-licenses/asm-LICENSE
apache-hive-3.1.3-bin/binary-package-licenses/com.ibm.icu.icu4j-LICENSE
apache-hive-3.1.3-bin/binary-package-licenses/org.antlr.stringtemplate-LICENSE
apache-hive-3.1.3-bin/binary-package-licenses/javax.transaction.transaction-api-LICENSE
apache-hive-3.1.3-bin/examples/files/avro_historical_timestamp_legacy.avro
apache-hive-3.1.3-bin/examples/files/hive-626-hive.txt
```

Step 2: Place different configuration properties in Apache Hive

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

Hive Home path in bashrc file

```
$ nano .bashrc
```

And append the below lines in it

```
export HIVE_HOME=/home/suryaa/Downloads/apache-hive-3.1.3-bin
export PATH=$PATH:$HIVE_HOME/bin
```

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/1QFhB7Kvcat7a4LzDRe6GcmZvalyAxKz/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

```

suryaa@suryaa-HP-Pavilion-Laptop-15-eg2xxx:~/Downloads$ $HIVE_HOME/bin/schematool -initSchema -dbType mysql --verbose
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/suryaa/Downloads/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/suryaa/hadoop-3.4.0/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]
Metastore connection URL:      jdbc:mysql://localhost:3306/hive_db
Metastore Connection Driver :   com.mysql.cj.jdbc.Driver
Metastore connection User:     suryaa
Sat Sep 21 08:38:58 IST 2024 WARN: Caught while disconnecting...

```

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`

```
surya@surya-HP-Pavilion-Laptop-15-eg2xxx:~/Downloads$ hive
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/suryaa/Downloads/apache-hive-3.1.3-bin/lib/log4j-slf4j-impl-2.17.1.jar!/org/slf4j/impl/StaticLoggerFactory.class]
SLF4J: Found binding in [jar:file:/home/suryaa/hadoop-3.4.0/share/hadoop/common/lib/slf4j-reload4j-1.7.36.jar!/org/slf4j/impl/StaticLoggerFactory.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 = 6fbfe530-4714-4eef-98c5-d5f15bd935d0

Logging initialized using configuration in jar:file:/home/suryaa/Downloads/apache-hive-3.1.3-bin/lib/hive-common-3.1.3.jar!/hive-log4j.properties: 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. Flink, Tez or Spark) instead of Hive 1.X releases.
Hive Session ID = 01cc9a54-b38b-4ada-8de5-ace977ded1ee
hive> show databases;
OK
default
financials
Time taken: 0.36 seconds, Fetched: 2 row(s)
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

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;

```
hive> use financials;
OK
Time taken: 0.57 seconds
```

Step 4: Create a Table:

Create a simple table in your database:

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

```
hive> CREATE TABLE finance_table( id INT, name STRING );
OK
Time taken: 2.013 seconds
```

Step 5: Load Sample Data:

You can insert sample data into the table:

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

```

hive> INSERT INTO finance_table VALUES
    > (1,'Alice')
    > ,
    > (2,'Bob'),
    > (3,'Charlie');
Query ID = hadoop_20240911171244_304f3e60-6937-434c-acb2-d71be2797182
Total jobs = 3
Launching Job 1 out of 3
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Job running in-process (local Hadoop)
2024-09-11 17:12:54,138 Stage-1 map = 0%,  reduce = 0%
2024-09-11 17:12:57,541 Stage-1 map = 100%,  reduce = 100%
Ended Job = job_local1825573535_0001
Stage-4 is selected by condition resolver.
Stage-3 is filtered out by condition resolver.
Stage-5 is filtered out by condition resolver.
Moving data to directory hdfs://localhost:9000/user/hive/warehouse/financials.db/finance_table/.hive-staging_hive_2024-9-11_17-12-44_558_5675160086864575725-1-ext-10000
Loading data to table financials.finance_table
MapReduce Jobs Launched:
Stage-Stage-1:  HDFS Read: 0 HDFS Write: 208 SUCCESS
Total MapReduce CPU Time Spent: 0 msec
OK
Time taken: 13.965 seconds

```

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;
```

```

hive> CREATE VIEW myview AS SELECT name, id FROM finance_table;
OK
Time taken: 0.244 seconds

```

Step 7: View the data:

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

```

hive> SELECT*FROM myview;
OK
Alice 1
Bob 2
Charlie 3
Time taken: 0.22 seconds, Fetched: 3 row(s)

```

Step 8: Describe a Table:

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

```
hive>DESCRIBE finance_table;
```

```

hive> DESCRIBE finance_table;
OK
id          int
name        string
age         int
Time taken: 0.729 seconds, Fetched: 3 row(s)

```

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);`

```
hive> ALTER TABLE finance_table ADD COLUMNS (age INT);
OK
Time taken: 0.188 seconds
```

Step 10: Quit Hive:

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

```
hive> quit;
hadoop@priyav-VirtualBox:~$
```

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:

Step 1: Install Required Packages

Install the necessary packages using pip:

```
$ pip install pandas --break-system-packages
```

Step 2: Verify Package Installation

Verify that the required packages are installed:

```
$ python
>>> import pandas as pd
>>> from hdfs import InsecureClient
>>> print("Pandas version:", pd.__version__)
>>> client = InsecureClient('http://localhost:9870', user='hadoop')
>>> print("HDFS status:", client.status('/'))
>>> exit()
```

Step 3: Create process_data.py File

Create the Python script for processing data:

```
$ nano process_data.py
```

Paste the following code into the file:

```
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()
        if not json_data.strip():
            raise ValueError("The JSON file is empty.")
        data = json.loads(json_data)
except Exception as e:
```

```

print(f"Error reading or parsing JSON data: {e}")
exit(1)

# Convert JSON data to DataFrame
df = pd.DataFrame(data)

# Projection: Select 'name' and 'salary'
projected_df = df[['name', 'salary']]

# Aggregation: Calculate total salary
total_salary = df['salary'].sum()

# Count: Employees earning more than 50000
high_earners_count = df[df['salary'] > 50000].shape[0]

# Limit: Top 5 highest earners
top_5_earners = df.nlargest(5, 'salary')

# Skip: Skip the first 2 employees
skipped_df = df.iloc[2:]

# Remove: Filter out employees from IT department
filtered_df = df[df['department'] != 'IT']

# Save the filtered data 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)
except Exception as e:
    print(f"Error saving filtered JSON data: {e}")

```

Step 4: Run the Script

Execute the script to process the data:

\$ python3 process_data.py

Step 5: To view the output

```
hadoop@Ubuntu:~/Documents$ hdfs dfs -cat /home/hadoop/emp.json
```

```
suryaa@suryaa-HP-Pavilion-Laptop-15-eq2xxx: $ python3 process_data.py
Raw JSON Data: [
    {
        "name": "John Doe",
        "age": 30,
        "department": "HR",
        "salary": 50000
    },
    {
        "name": "Jane Smith",
        "age": 25,
        "department": "IT",
        "salary": 60000
    },
    {
        "name": "Alice Johnson",
        "age": 35,
        "department": "Finance",
        "salary": 70000
    },
    {
        "name": "Bob Brown",
        "age": 28,
        "department": "Marketing",
        "salary": 55000
    },
    {
        "name": "Charlie Black",
        "age": 45,
        "department": "IT",
        "salary": 80000
    }
]
Filtered JSON file saved successfully.
```

Screenshot captured
You can paste the image from the clipboard.

```
Filtered JSON file saved successfully.
Projection: Select only name and salary columns
      name   salary
0  John Doe  50000
1  Jane Smith  60000
2 Alice Johnson  70000
3  Bob Brown  55000
4 Charlie Black  80000
Aggregation: Calculate total salary
Total Salary: 315000

Count: Number of employees earning more than 50000
Number of High Earners (>50000): 4

Top 5 highest salary
Top 5 Earners:
      name   age department   salary
4 Charlie Black  45        IT  80000
2 Alice Johnson  35    Finance  70000
1  Jane Smith  25        IT  60000
3  Bob Brown  28  Marketing  55000
0  John Doe  30        HR  50000

Skipped DataFrame (First 2 rows skipped):
      name   age department   salary
2 Alice Johnson  35    Finance  70000
3  Bob Brown  28  Marketing  55000
4 Charlie Black  45        IT  80000

Filtered DataFrame (IT department removed):
      name   age department   salary
0  John Doe  30        HR  50000
2 Alice Johnson  35    Finance  70000
3  Bob Brown  28  Marketing  55000
suryaa@suryaa-HP-Pavilion-Laptop-15-eq2xxx: $
```

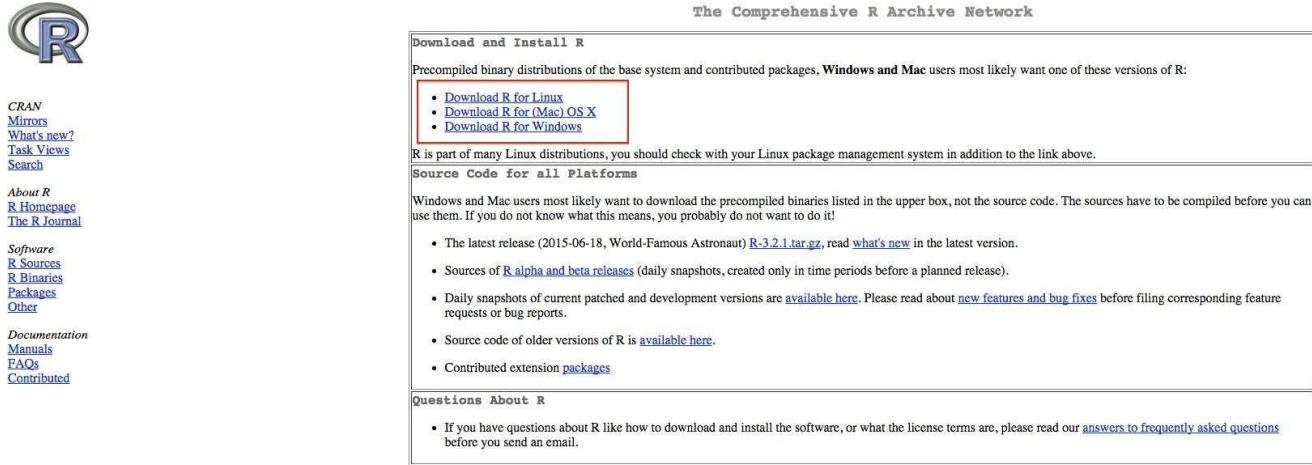
RESULT:

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/>



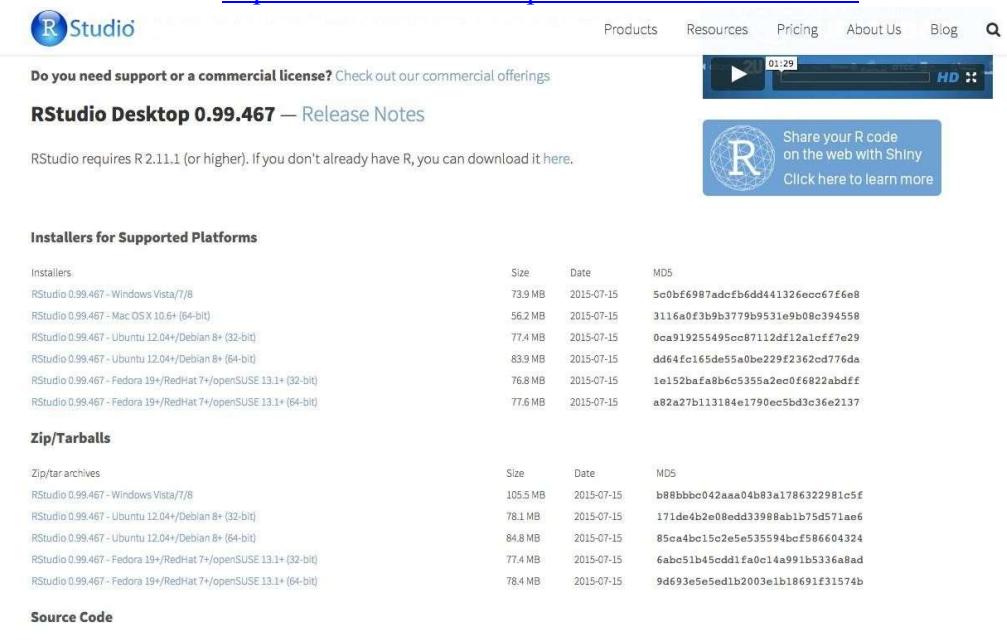
The screenshot shows the official Comprehensive R Archive Network (CRAN) website. On the left, there's a sidebar with links for CRAN Mirrors, What's new?, Task Views, About R, R Homepage, R Journal, Software, R Sources, R Binaries, Packages, Documentation, Manuals, FAQs, and Contributed. The main content area has a heading 'Download and Install R'. It says: 'Precompiled binary distributions of the base system and contributed packages. Windows and Mac users most likely want one of these versions of R:' followed by a list: '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.' Under 'Source Code for all Platforms', it says: 'Windows and Mac users most likely want to download the precompiled binaries listed in the upper box, not the source code. The sources have to be compiled before you can use them. If you do not know what this means, you probably do not want to do it!' followed by a list of items related to source code and releases. At the bottom, there's a 'Questions About R' section with a bullet point about frequently asked questions.

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

- 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 navigation bar with links for Products, Resources, Pricing, About Us, Blog, and a search icon. Below the navigation, there's a video player showing a video titled 'RStudio Desktop 0.99.467 — Release Notes'. The main content area has a heading 'Installers for Supported Platforms' with a table showing file sizes, dates, and MD5 checksums for various RStudio installers across different platforms. There's also a 'Zip/Tarballs' section with a similar table. At the bottom, there's a 'Source Code' section with a note about downloading the tarball containing source code.

Installers	Size	Date	MD5
RStudio 0.99.467 - Windows Vista/7/8	73.9 MB	2015-07-15	5c0bf6987adcfb6dd441326ecc67f6e8
RStudio 0.99.467 - Mac OS X 10.6+ (64-bit)	56.2 MB	2015-07-15	3116a0f3b9b3779b9531e9b08c39458
RStudio 0.99.467 - Ubuntu 12.04+/Debian 8+ (32-bit)	77.4 MB	2015-07-15	0ca919255495cc87112df12a1cff7e28
RStudio 0.99.467 - Ubuntu 12.04+/Debian 8+ (64-bit)	83.9 MB	2015-07-15	dd64fc165de55a0be229f2362cd776da
RStudio 0.99.467 - Fedora 19+/RedHat 7+/openSUSE 13.1+ (32-bit)	76.8 MB	2015-07-15	1e152bafa8b6c5355a2ec0f6822abdff
RStudio 0.99.467 - Fedora 19+/RedHat 7+/openSUSE 13.1+ (64-bit)	77.6 MB	2015-07-15	a82a27b113184e1790ec5bd3c36e2137

Zip/Tarballs	Size	Date	MD5
RStudio 0.99.467 - Windows Vista/7/8	105.5 MB	2015-07-15	b8bbbc042aa04b83a1786322981c5f
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	85ca4bc15c2e5e35594bcff586604324
RStudio 0.99.467 - Fedora 19+/RedHat 7+/openSUSE 13.1+ (32-bit)	77.4 MB	2015-07-15	6abc51b45cd1fa0c14a991b5336a8ad
RStudio 0.99.467 - Fedora 19+/RedHat 7+/openSUSE 13.1+ (64-bit)	78.4 MB	2015-07-15	9d693e5e5ed1b2003e1b18691e31574b

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

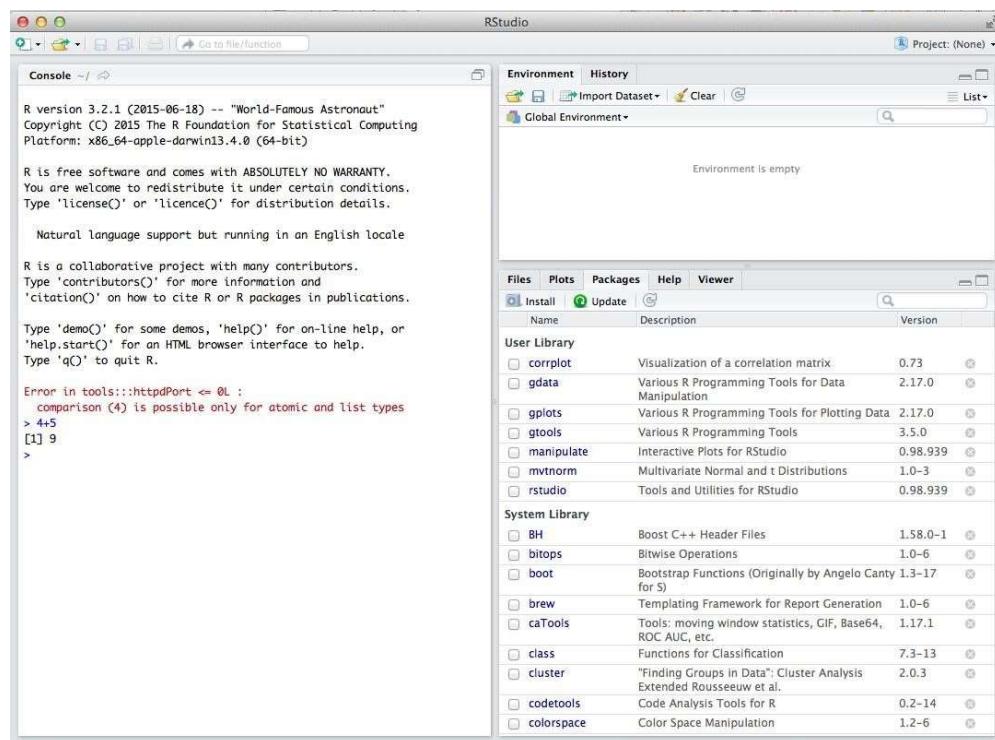


Figure 3. Running R with RStudio

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).

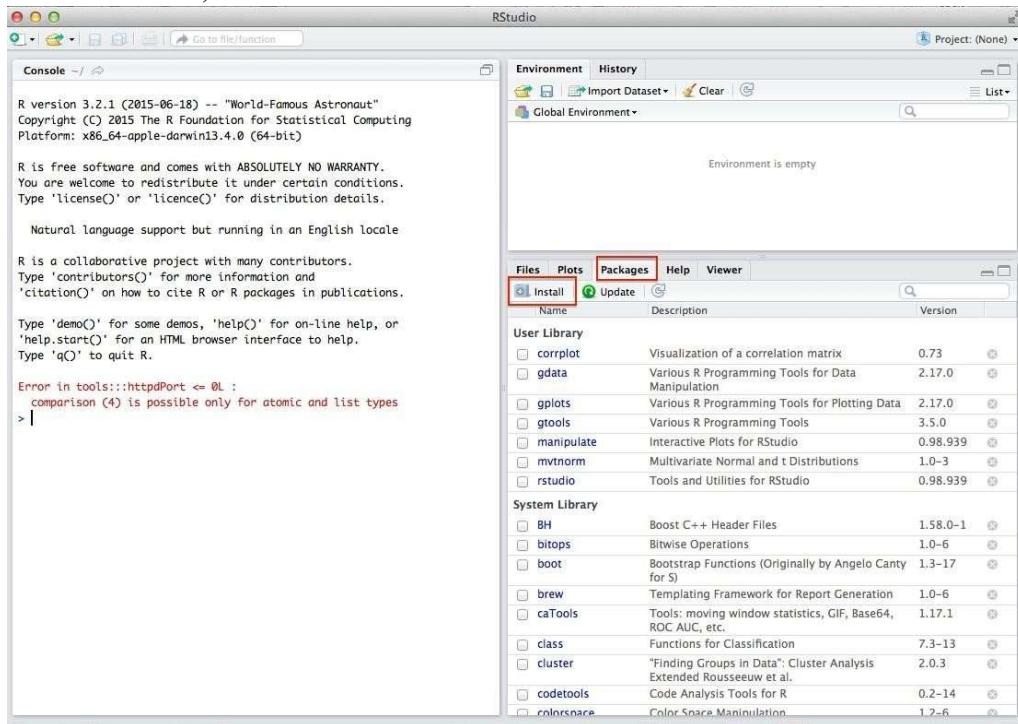


Figure 4. Click on Install to install R packages.

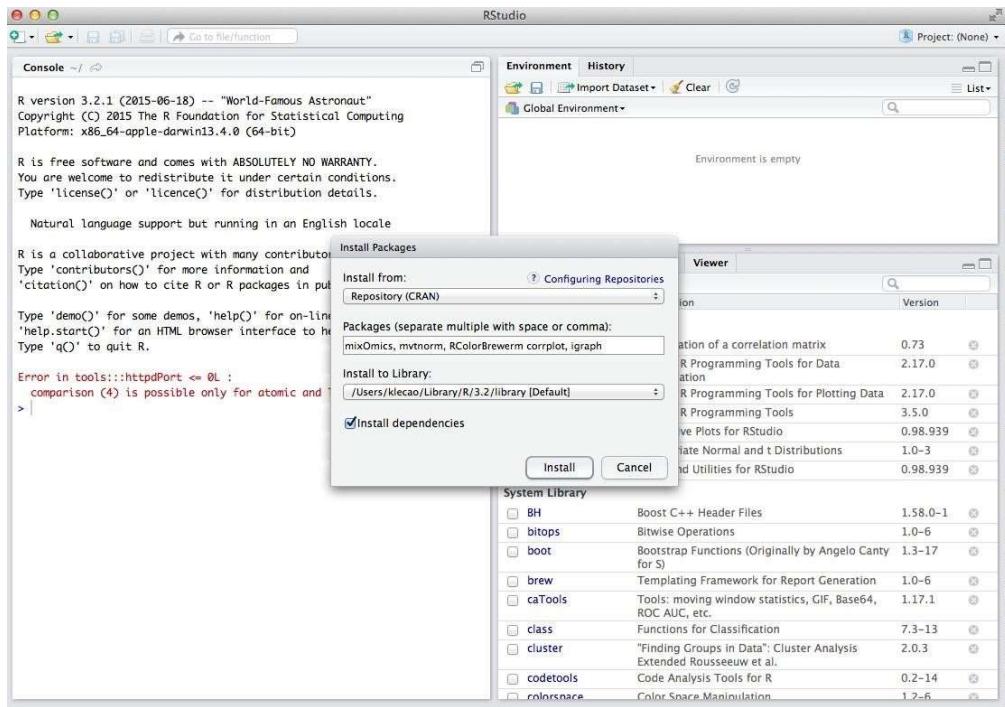


Figure 5. Specify the list of packages to be installed

The screenshot shows the RStudio interface with the following details:

- Console:** Displays the R session output. It shows the command `library(mixOmics)` being run, followed by a message indicating it is loading the MASS package. A red circle highlights the text "Loaded mixOmics 6.1.0 ok!".
- Environment:** Shows the global environment with various objects listed. A red circle highlights the "mixOmics" object.
- Packages:** Shows the installed packages. A red circle highlights the "mixOmics" package entry, which is version 6.1.0.

```

~/Documents/k.lecao/Presentation/2016/INPO-COST/CaseStudy_Sunflower/Drought - RStudio
[1] > library(mixOmics)
[1] "Loading required package: MASS"
[1] "Attaching package: 'MASS'"
[1] "The following object is masked _by_ '.GlobalEnv':"
[1] "genotype"
[1] "Loading required package: lattice"
[1] "Loading required package: ggplot2"
[1] "Loaded mixOmics 6.1.0 ok!"

Visit http://www.mixOmics.org for more details about our methods.
Any bug reports or questions? Notify us at mixomics at math.univ-toulouse.fr or https://bitbucket.org/klecao/package-mixomcs/issues

Thank you for using mixOmics!
sessionInfo()
#> version 3.3.2-patched (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/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  ggplot2_2.2.1.0  lattice_0.20-33 MASS_7.3-4.5

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
>

```

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:

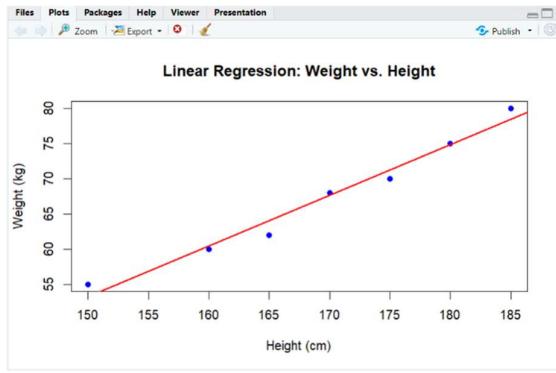
1. Create sample data for heights and weights, fit a linear regression model, and plot the data with the regression line.
2. Use the sample data to create a data frame for the regression model.
3. Fit the linear regression model using the `lm()` function and display the summary.
4. Plot the data points 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:



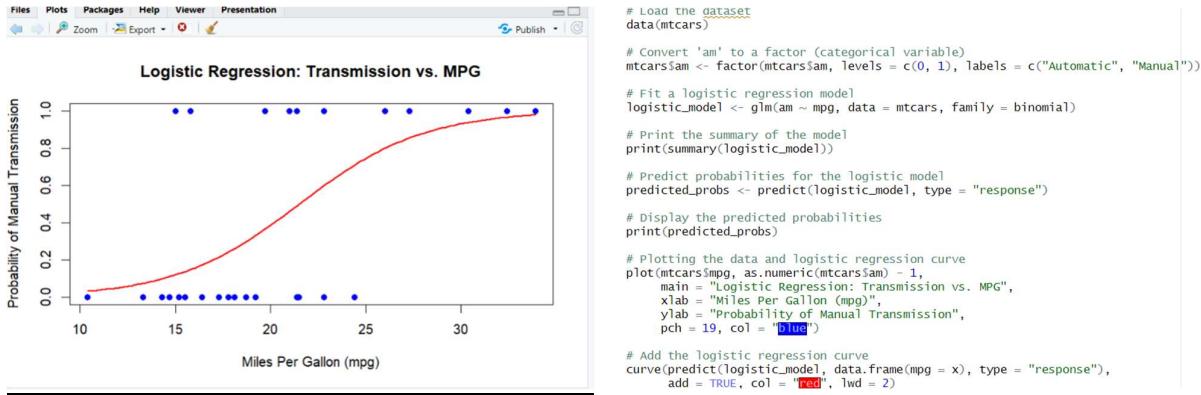
```
ex1.R* [1]
1 # Sample data
2 heights <- c(150, 160, 165, 170, 175, 180, 185)
3 weights <- c(55, 60, 62, 68, 70, 75, 80)
4
5 # Create a data frame
6 data <- data.frame(heights, weights)
7
8 # Fit a linear regression model
9 linear_model <- lm(weights ~ heights, data = data)
10
11 # Print the summary of the model
12 print(summary(linear_model))
13
14 # Plotting the data and regression line
15 plot(data$heights, data$weights,
16       main = "Linear Regression: Weight vs. Height",
17       xlab = "Height (cm)",
18       ylab = "Weight (kg)",
19       pch = 19, col = "blue")
20
21 # Add regression line
22 abline(linear_model, col = "red", lwd = 2)
23 |
```

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:

```

Number of Support Vectors: 45
( 7 18 20 )

Number of Classes: 3
Levels:
setosa versicolor virginica

> # 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)
      Actual
Predicted   setosa versicolor virginica
setosa       14        0        0
versicolor     0       17        0
virginica      0        1       13
> # Calculate accuracy
> accuracy <- sum(diag(confusion_matrix)) / sum(confusion_matrix)
> cat("Accuracy:", accuracy * 100, "%\n")
Accuracy: 97.77778 %
>

# 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")
|
```

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)

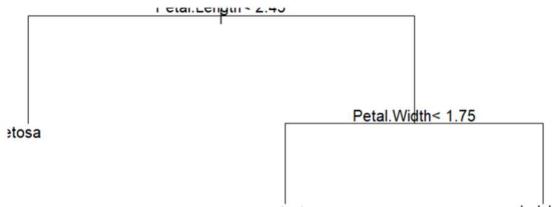
# Predict the test set
predictions <- predict(tree_model, newdata = test_data, type = "class")

# 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:



```

# 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)

```

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)

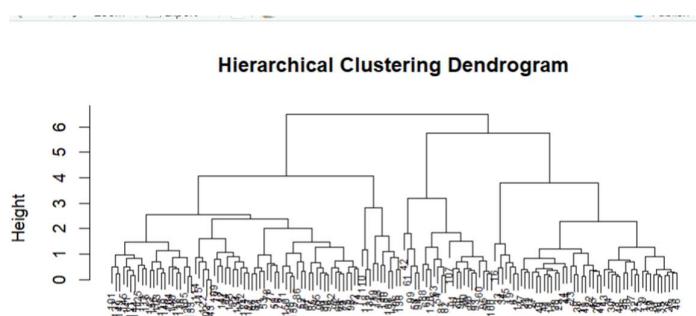
# Cut the tree to form 3 clusters
clusters <- cutree(hc_complete, k = 3)
```

```
# 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:

```
# 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)]
```



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")

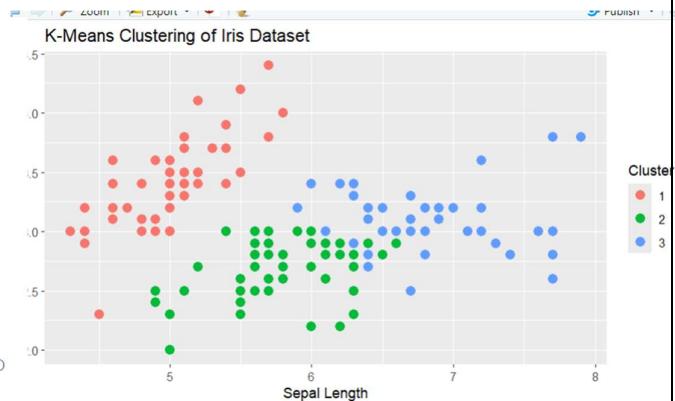
```

OUTPUT:

```

# 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")

```



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

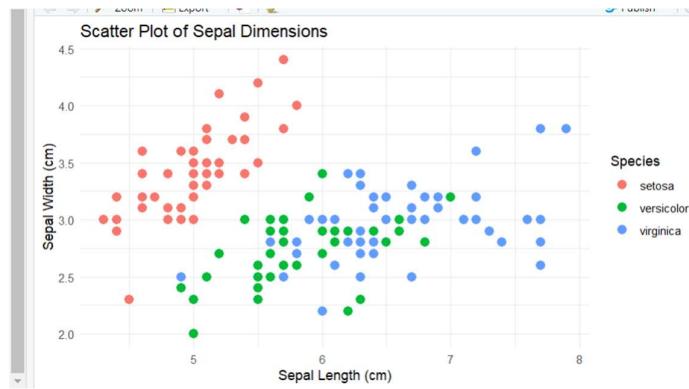
```
# Install ggplot2 (if not already installed)
install.packages("ggplot2")

# 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:

```
# 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 |
```



2) BAR CHART

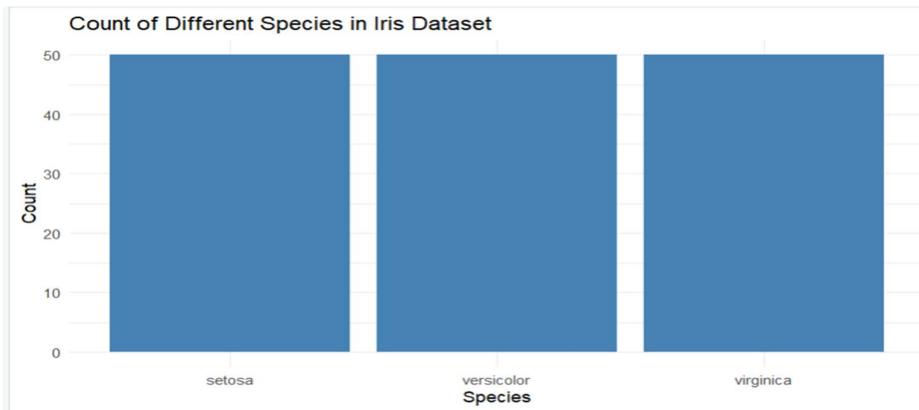
```
# Install ggplot2 (if not already installed)
install.packages("ggplot2")

# 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:

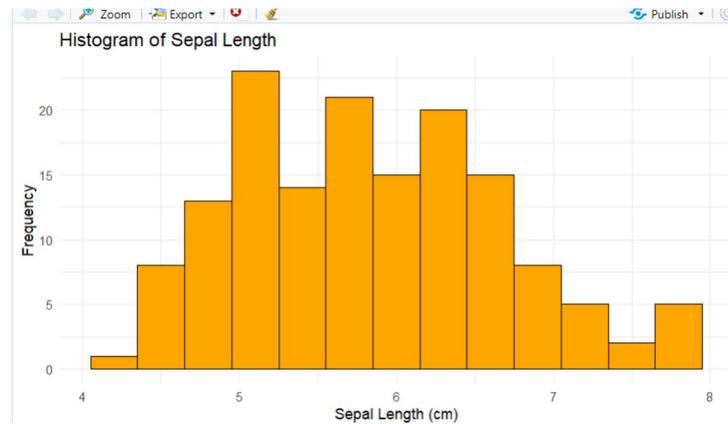
```
# Install ggplot2 (if not already installed)
install.packages("ggplot2")
# Load the ggplot2 package
library(ggplot2)
# Bar plot of Species counts
ggplot(data = iris, aes(x = Species)) +
  geom_bar(fill = "steelblue") + # Adds bars filled with steel blue color
  labs(title = "Count of Different Species in Iris Dataset",
       x = "Species",
       y = "Count") +
  theme_minimal()
```

**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:

```
1 library(ggplot2)
2 # Histogram of Sepal Length
3 ggplot(data = iris, aes(x = Sepal.Length)) +
4   geom_histogram(binwidth = 0.3, fill = "orange", color = "black") + # Adds
5   histogram bars
6   labs(title = "Histogram of Sepal Length",
7       x = "Sepal Length (cm)",
8       y = "Frequency") +
9   theme_minimal()
```

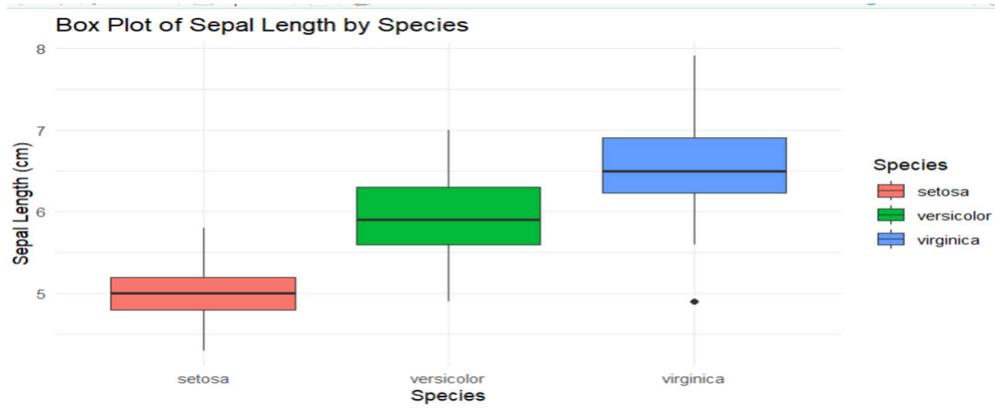


4)BOX PLOT

```
# 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:

```
library(ggplot2)  
# 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()
```

**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.