Reg.No.: 210701288

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:

```
thamizh@thamizh-HP-Pavilion-Laptop-15-eg2xxx ~
$ java --version
openjdk 21.0.4 2024-07-16
OpenJDK Runtime Environment (build 21.0.4+7-Ubuntu-1ubuntu224.04)
OpenJDK 64-Bit Server VM (build 21.0.4+7-Ubuntu-1ubuntu224.04, mixed mode, shari
ng)
```

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

```
thamizh@thamizh-HP-Pavilion-Laptop-15-eg2xxx ~
$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/thamizh/.ssh/id_rsa): y
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in y
Your public key has been saved in y.pub
The key fingerprint is:
SHA256:1FQBt0agN9e+YJz7EAjh5ni6Vv0wc6yYxJ7VfScchuw thamizh@thamizh-HP-Pavilion-Laptop
-15-eg2xxx
The key's randomart image is:
+---[RSA 3072]----+
|        . ++=.   |
|       . = o o   |
|        * +.+..  |
|       = o *ooo  |
|      ..S..+*+.. |
|       o+ *.E++.o|
|      .+ = Bo .o.|
|      ..= . .o   |
|     ..       .  |
+----[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

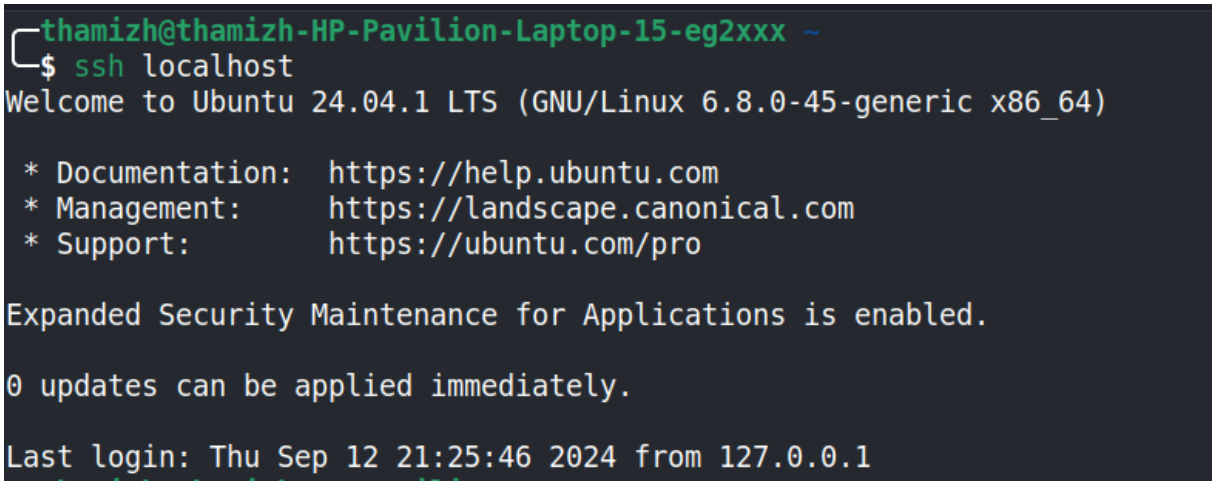
$ chmod 640 ~/.ssh/authorized_keys

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:



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:

$ wgethttps://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

```
thamizh@thamizh-HP-Pavilion-Laptop-15-eg2xxx ~
$ ls
'~'                                guice-5.0.1.jar          reducerWeather.py
'${system:java.io.tmpdir}'         hadoop-3.3.6             sample.l
288.l                              hadoop-3.3.6.tar.gz      snap
a.out                              hadoopdata               Templates
apache-hive-3.1.3-bin              lex.yy.c                 udf_example.pig
apache-hive-3.1.3-bin.tar.gz       libfprint                udp.py
apache-hive-3.1.3-bin.tar.gz.1     libinput-config          Untitled.ipynb
backend                            mapper.py                uppercase_udf.py
CD_UNIT-5.pptx.pdf                 mapperWeather.py         var.l
cglib-3.3.0.jar.1                  metastore_db             var.y
check.py                           Music                    Videos
demo_pig.pig                       nltk_data                weather_data.csv
demo.py                            Pictures                 WhiteSur-gtk-theme
derby.log                          pig-0.16.0               WhiteSur-icon-theme
Desktop                            pig-0.16.0.tar.gz        word_count.txt
Documents                          pig_1726261423895.log    y
Downloads                          pig_sample.txt           y.pub
eckey.pem                          process_data.py          y.tab.c
emp.json                           Public                   y.tab.h
fprintd                            reducer.py
```

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:

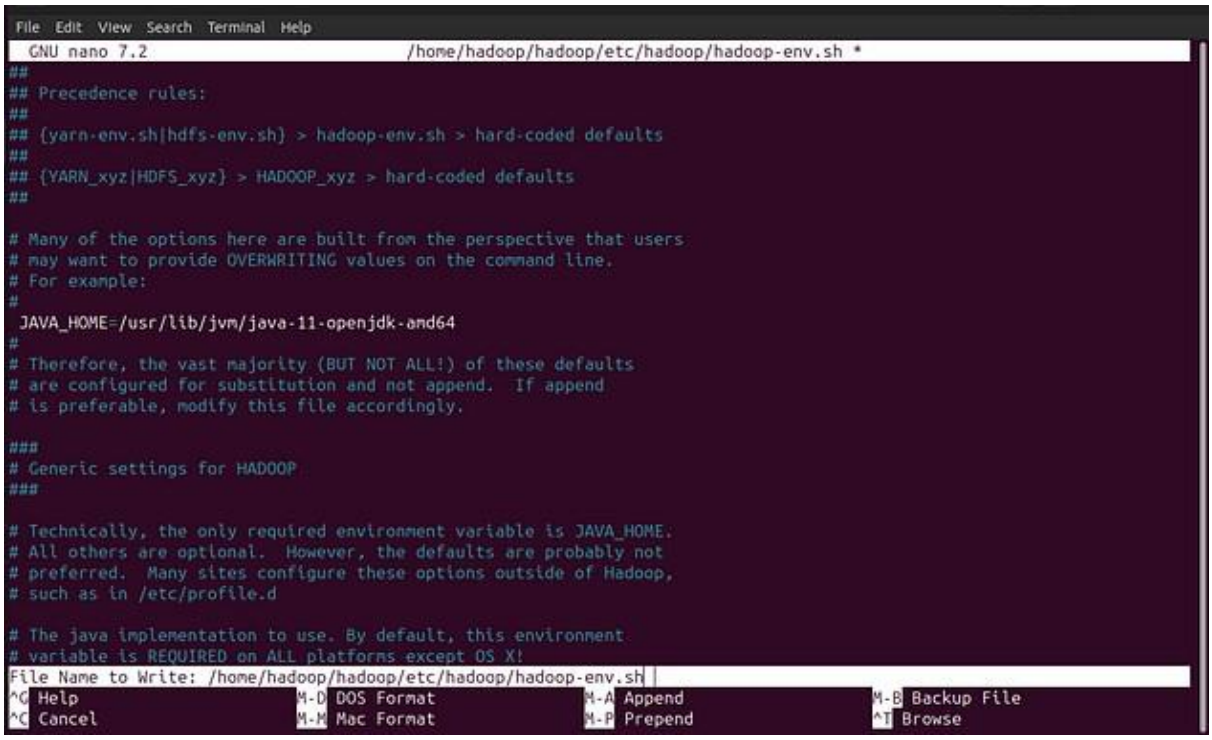
s$ source ~/.bashrc

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



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

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

```
thamizh@thamizh-HP-Pavilion-Laptop-15-eg2xxx ~
$ start-all.sh
WARNING: Attempting to start all Apache Hadoop daemons as thamizh 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 [thamizh-HP-Pavilion-Laptop-15-eg2xxx]
Starting resourcemanager
Starting nodemanagers
```

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

$ jps

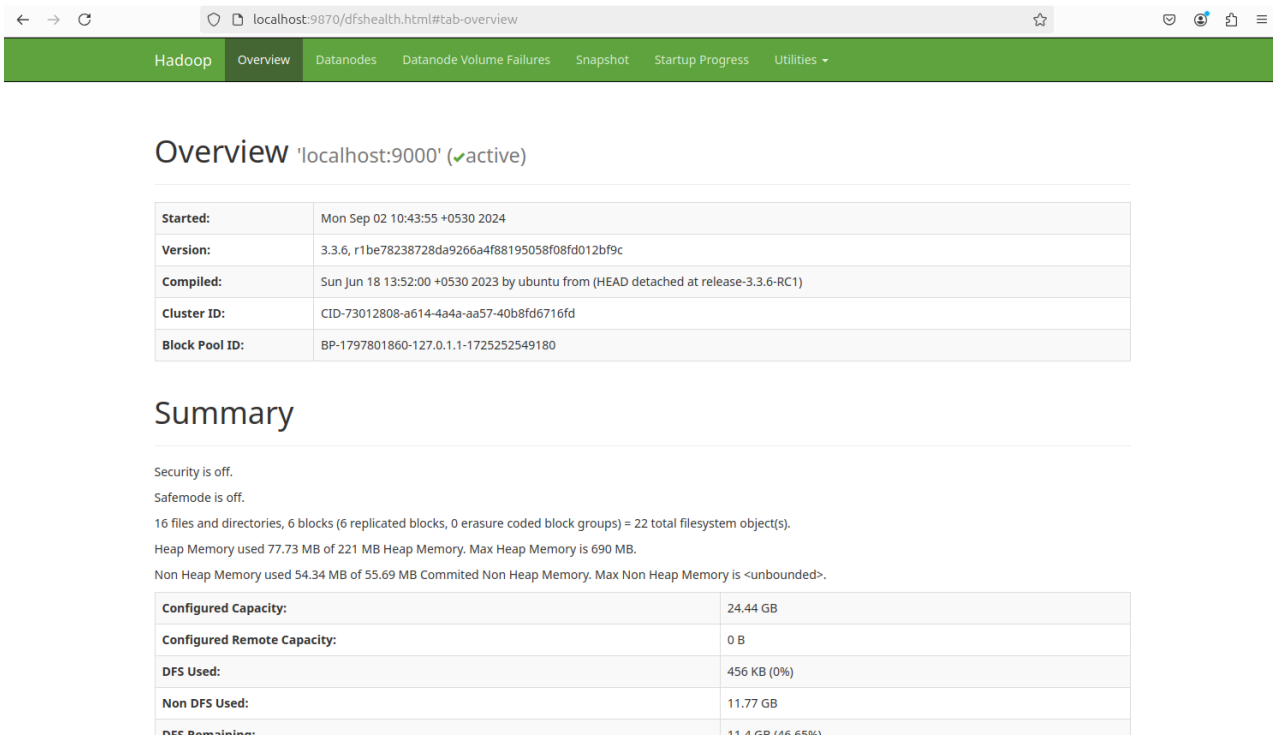
```
thamizh@thamizh-HP-Pavilion-Laptop-15-eg2xxx ~
$ jps
7333 ResourceManager
6568 NameNode
7065 SecondaryNameNode
7516 NodeManager
6796 DataNode
8012 Jps
```

Step 13 – Access Hadoop Namenode and Resource Manager

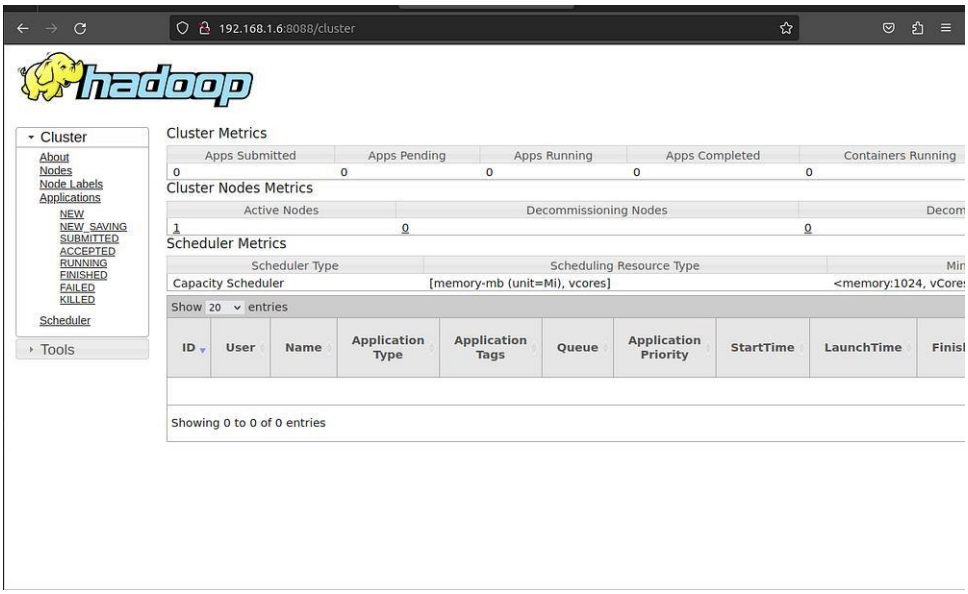
- First we need to know our ipaddress, In Ubuntu we need to install net-tools to run ipconfig command,
  If you installing net-tools for the first time switch to default user:
  $sudo apt install net-tools
- Then run ifconfig command to know our ip address: **ifconfig**

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



To access Resource Manage, open your web browser and visit the URL http://your-serverip:8088. You should see the following screen: http://192.168.16:8088



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:

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

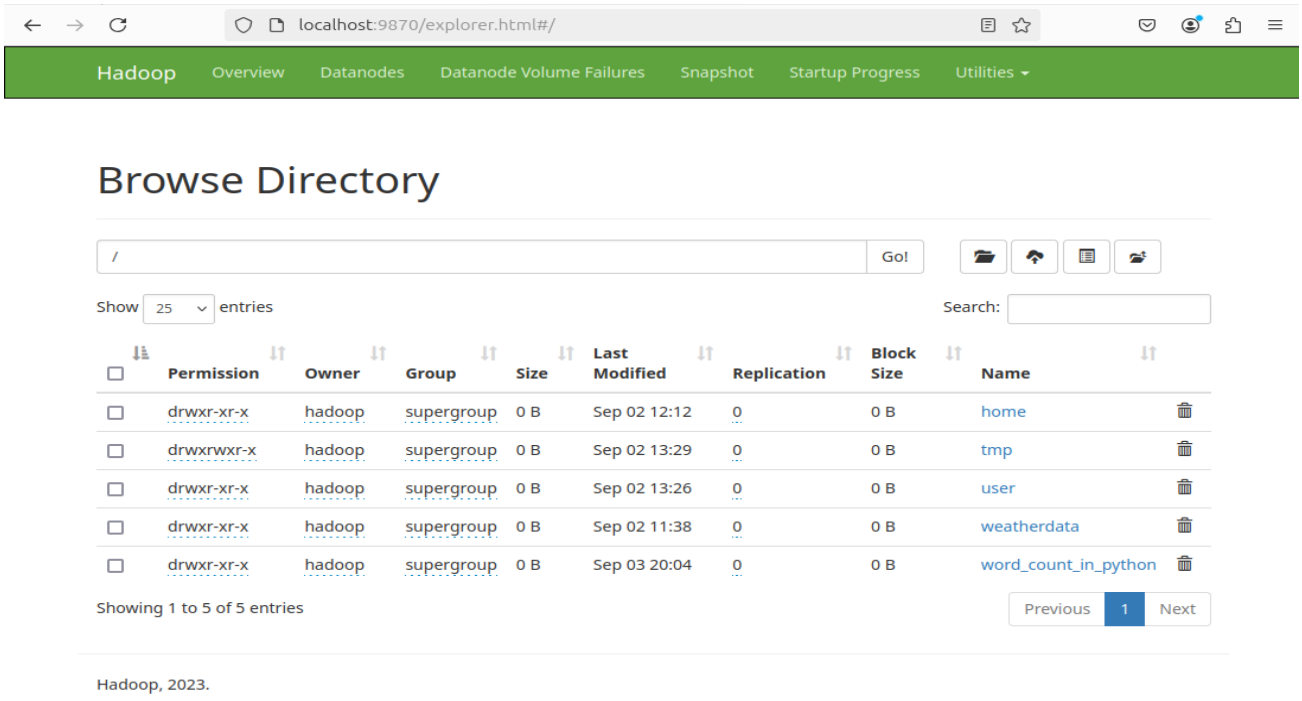
```
thamizh@thamizh-HP-Pavilion-Laptop-15-eg2xxx ~
$ hdfs dfs -ls /
Found 7 items
drwxr-xr-x   - thamizh supergroup          0 2024-09-20 11:53 /home
drwxr-xr-x   - thamizh supergroup          0 2024-09-19 11:00 /tmp
drwxr-xr-x   - thamizh supergroup          0 2024-09-14 02:36 /udfs
drwxr-xr-x   - thamizh supergroup          0 2024-09-14 02:36 /udfsp
drwxr-xr-x   - thamizh supergroup          0 2024-09-19 13:37 /user
drwxr-xr-x   - thamizh supergroup          0 2024-09-13 15:31 /weather_data
drwxr-xr-x   - thamizh supergroup          0 2024-09-13 08:45 /word_count_in_python
```

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.

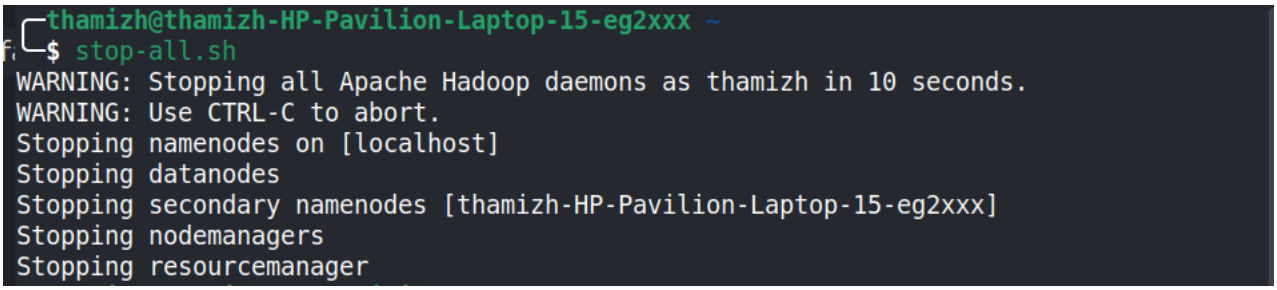
Go to the web interface, click on the Utilities => Browse the file system. You should see your directories which you have created earlier in the following screen:



Step 15 – Stop Hadoop Cluster

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

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



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

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.

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.

#!/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
```

```
thamizh@thamizh-HP-Pavilion-Laptop-15-eg2xxx ~
$ hadoop jar /home/thamizh/hadoop-3.3.6/share/hadoop/tools/lib/hadoop-streaming-3.3
.6.jar -input /word_count_in_python/word_count.txt -output /word_count_in_python/outp
ut -mapper /home/thamizh/mapper.py -reducer /home/thamizh/reducer.py
2024-09-23 13:32:39,684 INFO impl.MetricsConfig: Loaded properties from hadoop-metric
s2.properties
2024-09-23 13:32:39,783 INFO impl.MetricsSystemImpl: Scheduled Metric snapshot period
 at 10 second(s).
2024-09-23 13:32:39,783 INFO impl.MetricsSystemImpl: JobTracker metrics system starte
d
2024-09-23 13:32:39,805 WARN impl.MetricsSystemImpl: JobTracker metrics system alread
y initialized!
2024-09-23 13:32:40,080 INFO mapred.FileInputFormat: Total input files to process : 1
2024-09-23 13:32:40,178 INFO mapreduce.JobSubmitter: number of splits:1
2024-09-23 13:32:40,266 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_l
ocal1703582530_0001
2024-09-23 13:32:40,266 INFO mapreduce.JobSubmitter: Executing with tokens: []
2024-09-23 13:32:40,386 INFO mapreduce.Job: The url to track the job: http://localhos
t:8080/
2024-09-23 13:32:40,387 INFO mapreduce.Job: Running job: job_local1703582530_0001
2024-09-23 13:32:40,387 INFO mapred.LocalJobRunner: OutputCommitter set in config nul
l
2024-09-23 13:32:40,389 INFO mapred.LocalJobRunner: OutputCommitter is org.apache.had
oop.mapred.FileOutputCommitter
2024-09-23 13:32:40,394 INFO output.FileOutputCommitter: File Output Committer Algori
thm version is 2
```

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

```
thamizh@thamizh-HP-Pavilion-Laptop-15-eg2xxx ~
$ hdfs dfs -cat /word_count_in_python/new_output/part-00000                127 ↵
Callina 1
Finally 1
LA      2
Lookin  1
Lost    1
Made    1
Maria   2
Might   1
Tryna   1
dive    1
dough   1
for     2
in      2
it      1
make    1
marina  1
my      1
own     1
the     1
though  1
to      1
weed    1
without 1
```

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:

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.

```
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.

```
#!/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))
```

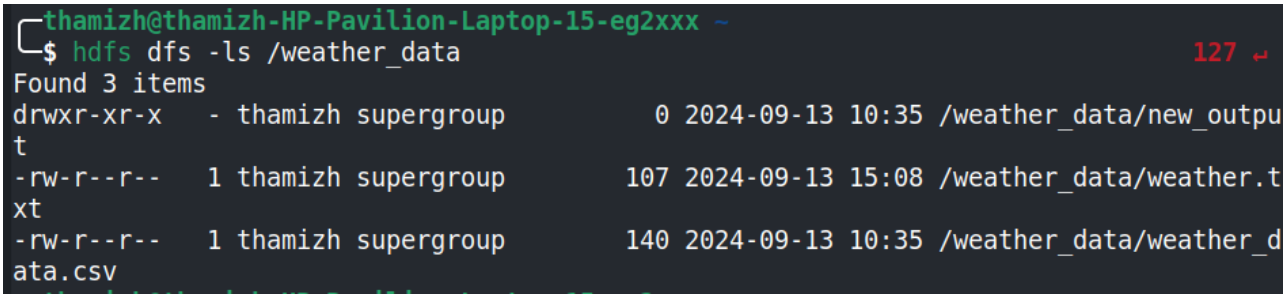
Step 4: Prepare Hadoop Environment:

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.



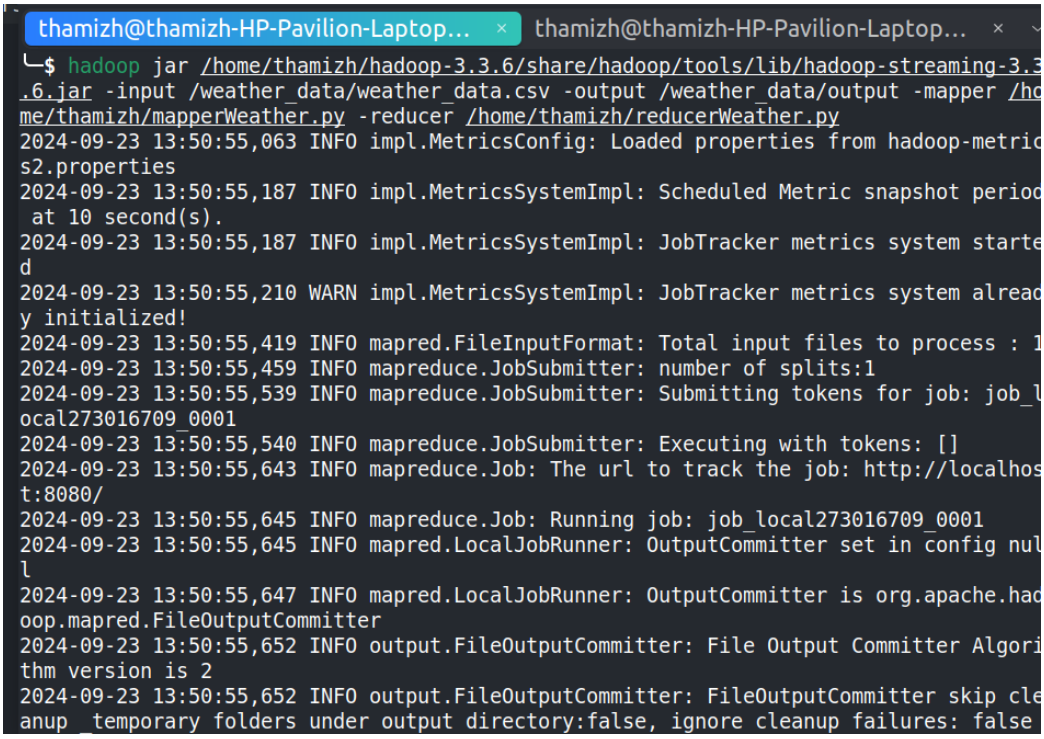
Step 7: Run the program using Hadoop Streaming:

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"
```

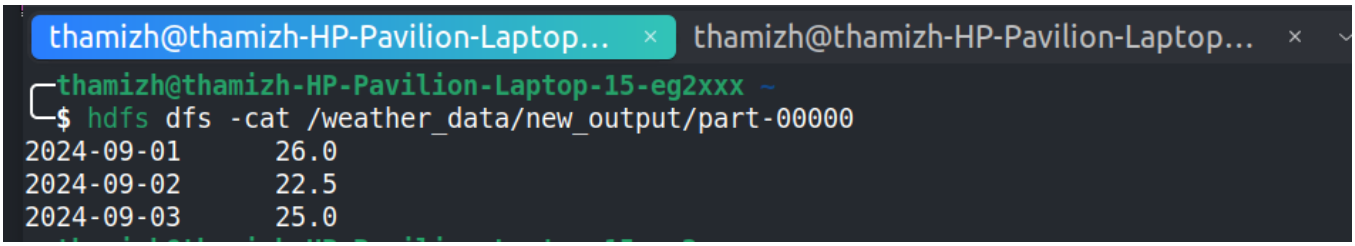
hdfs dfs -text /weatherdata/output/* > /home/sx/Downloads/outputfile.txt



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



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

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:

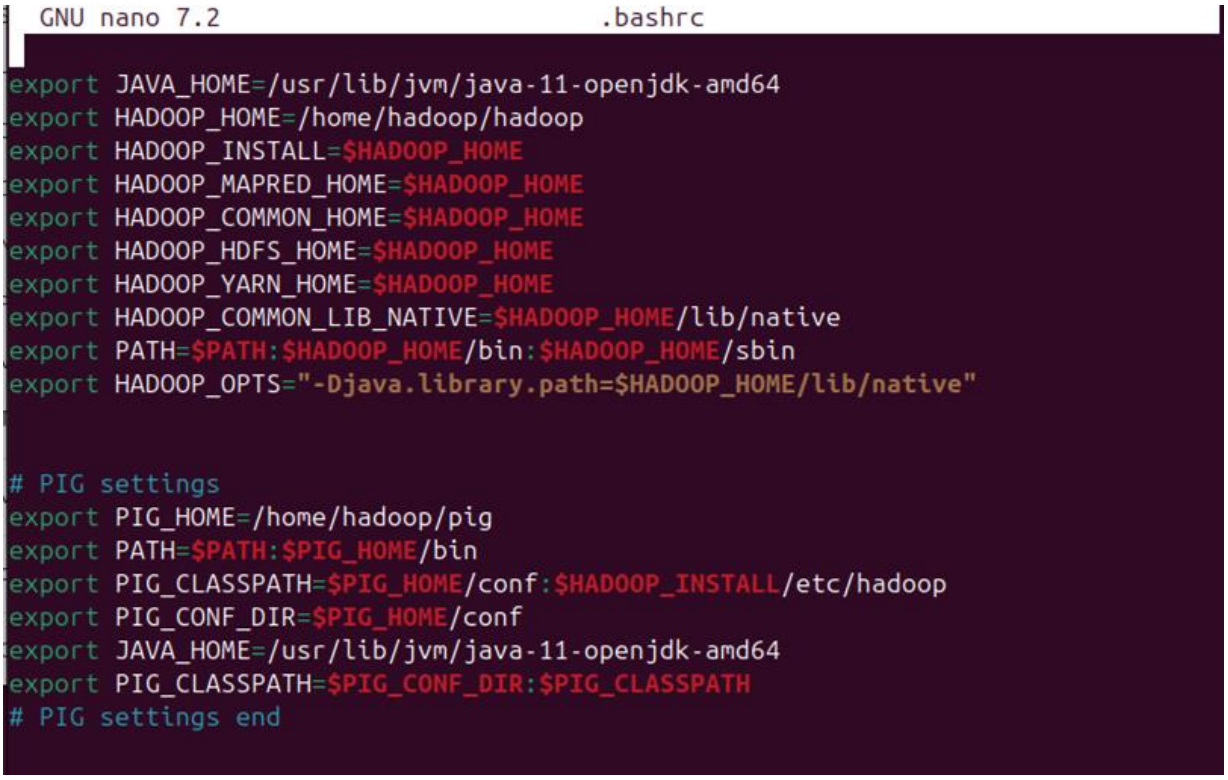
$ sudo mv /home/hadoop/pig-0.16.0 /home/hadoop/pig

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/hdoop/pigexport PATH=$PATH:$PIG_HOME/binexport PIG_CLASSPATH=$PIG_HOME/conf:$HADOOP_INSTALL/etc/hadoop/export PIG_CONF_DIR=$PIG_HOME/confexport JAVA_HOME=/usr/lib/jvm/java-8-openjdkamd64export PIG_CLASSPATH=$PIG_CONF_DIR:$PATH#PIG setting ends



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

```
thamizh@thamizh-HP-Pavilion-Laptop-15-eg2xxx ~
$ jps
47264 ResourceManager
65505 Jps
46996 SecondaryNameNode
47446 NodeManager
46728 DataNode
46494 NameNode
```

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

```
thamizh@thamizh-HP-Pavilion-Laptop-15-eg2xxx ~
$ pig
2024-09-23 13:53:29,765 INFO pig.ExecTypeProvider: Trying ExecType : LOCAL
2024-09-23 13:53:29,766 INFO pig.ExecTypeProvider: Trying ExecType : MAPREDUCE
2024-09-23 13:53:29,766 INFO pig.ExecTypeProvider: Picked MAPREDUCE as the ExecType
2024-09-23 13:53:29,818 [main] INFO  org.apache.pig.Main - Apache Pig version 0.16.0
(r1746530) compiled Jun 01 2016, 23:10:49
2024-09-23 13:53:29,818 [main] INFO  org.apache.pig.Main - Logging error messages to:
 /home/thamizh/pig_1727079809816.log
2024-09-23 13:53:29,841 [main] INFO  org.apache.pig.impl.util.Utils - Default bootup
file /home/thamizh/.pigbootup not found
2024-09-23 13:53:30,183 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation
 - mapred.job.tracker is deprecated. Instead, use mapreduce.jobtracker.address
2024-09-23 13:53:30,183 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation
 - fs.default.name is deprecated. Instead, use fs.defaultFS
2024-09-23 13:53:30,183 [main] INFO  org.apache.pig.backend.hadoop.executionengine.HE
xecutionEngine - Connecting to hadoop file system at: hdfs://localhost:9000
2024-09-23 13:53:30,596 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation
 - fs.default.name is deprecated. Instead, use fs.defaultFS
2024-09-23 13:53:30,611 [main] INFO  org.apache.pig.PigServer - Pig Script ID for the
 session: PIG-default-970afe7b-6a43-4413-8c39-e113ff6514d3
2024-09-23 13:53:30,611 [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,Sri

2,Vaish

3,Subhi

4,Priya

5,Sweatha

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

```
thamizh@thamizh-HP-Pavilion-Laptop-15-eg2xxx ~
$ pig demo_pig.pig
2024-09-23 13:55:18,214 INFO pig.ExecTypeProvider: Trying ExecType : LOCAL
2024-09-23 13:55:18,215 INFO pig.ExecTypeProvider: Trying ExecType : MAPREDUCE
2024-09-23 13:55:18,215 INFO pig.ExecTypeProvider: Picked MAPREDUCE as the ExecType
2024-09-23 13:55:18,239 [main] INFO  org.apache.pig.Main - Apache Pig version 0.16.0 (r1
746530) compiled Jun 01 2016, 23:10:49
2024-09-23 13:55:18,240 [main] INFO  org.apache.pig.Main - Logging error messages to: /h
ome/thamizh/pig_1727079918238.log
2024-09-23 13:55:18,558 [main] INFO  org.apache.pig.impl.util.Utils - Default bootup fil
e /home/thamizh/.pigbootup not found
2024-09-23 13:55:18,592 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation -
mapred.job.tracker is deprecated. Instead, use mapreduce.jobtracker.address
2024-09-23 13:55:18,593 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation -
fs.default.name is deprecated. Instead, use fs.defaultFS
2024-09-23 13:55:18,593 [main] INFO  org.apache.pig.backend.hadoop.executionengine.HExec
utionEngine - Connecting to hadoop file system at: hdfs://localhost:9000
2024-09-23 13:55:18,909 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation -
fs.default.name is deprecated. Instead, use fs.defaultFS
2024-09-23 13:55:18,921 [main] INFO  org.apache.pig.PigServer - Pig Script ID for the se
ssion: PIG-demo_pig.pig-537a5c63-a43e-4668-ba49-b280f87ecbed
2024-09-23 13:55:18,921 [main] WARN  org.apache.pig.PigServer - ATS is disabled since ya
rn.timeline-service.enabled set to false
2024-09-23 13:55:19,220 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation -
fs.default.name is deprecated. Instead, use fs.defaultFS
2024-09-23 13:55:19,279 [main] INFO  org.apache.pig.Main - Pig script completed in 1 sec
ond and 84 milliseconds (1084 ms)
```

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)
```

Reg.No.: 210701288

Create the udfs folder on hadoop

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

put the upppercase_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

```
thamizh@thamizh-HP-Pavilion-Laptop-15-eg2xxx ~
$ pig -f udf_example.pig
2024-09-23 13:56:20,591 INFO pig.ExecTypeProvider: Trying ExecType : LOCAL
2024-09-23 13:56:20,592 INFO pig.ExecTypeProvider: Trying ExecType : MAPREDUCE
2024-09-23 13:56:20,592 INFO pig.ExecTypeProvider: Picked MAPREDUCE as the ExecType
2024-09-23 13:56:20,616 [main] INFO  org.apache.pig.Main - Apache Pig version 0.16.0 (r1
746530) compiled Jun 01 2016, 23:10:49
2024-09-23 13:56:20,616 [main] INFO  org.apache.pig.Main - Logging error messages to: /h
ome/thamizh/pig_1727079980615.log
2024-09-23 13:56:20,846 [main] INFO  org.apache.pig.impl.util.Utils - Default bootup fil
e /home/thamizh/.pigbootup not found
2024-09-23 13:56:20,924 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation -
mapred.job.tracker is deprecated. Instead, use mapreduce.jobtracker.address
2024-09-23 13:56:20,924 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation -
fs.default.name is deprecated. Instead, use fs.defaultFS
2024-09-23 13:56:20,924 [main] INFO  org.apache.pig.backend.hadoop.executionengine.HExec
utionEngine - Connecting to hadoop file system at: hdfs://localhost:9000
2024-09-23 13:56:21,310 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation -
fs.default.name is deprecated. Instead, use fs.defaultFS
2024-09-23 13:56:21,323 [main] INFO  org.apache.pig.PigServer - Pig Script ID for the se
ssion: PIG-udf_example.pig-e8ea93d1-9cfb-49d5-b879-3fab58e434e4
2024-09-23 13:56:21,323 [main] WARN  org.apache.pig.PigServer - ATS is disabled since ya
rn.timeline-service.enabled set to false
2024-09-23 13:56:21,351 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation -
```

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

```
thamizh@thamizh-HP-Pavilion-Laptop-15-eg2xxx ~
$ hdfs dfs -ls /udfsp                                                          6
Found 1 items
drwxr-xr-x   - thamizh supergroup          0 2024-09-14 02:36 /udfsp/pig_output_data
thamizh@thamizh-HP-Pavilion-Laptop-15-eg2xxx ~
$ hdfs dfs -cat /udfsp/pig_output_data/part-m-00000
1,JOHN
2,JANE
3,JOE
4,EMMA
```

Result:

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

Reg.No.: 210701288

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:

$wgethttps://downloads.apache.org/hive/hive-3.1.2/apache-hive-3.1.2-bin.tar.gz

```
thamizh@thamizh-HP-Pavilion-Laptop-15-eg2xxx ~
$ wget https://archive.apache.org/dist/hive/hive-3.1.3/apache-hive-3.1.3-bin.tar.gz
--2024-09-23 13:58:48--  https://archive.apache.org/dist/hive/hive-3.1.3/apache-hive-3.1
.3-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... connected.
HTTP request sent, awaiting response... 200 OK
Length: 326940667 (312M) [application/x-gzip]
Saving to: 'apache-hive-3.1.3-bin.tar.gz.2'

     apache-hive-3.1.   0%[                          ] 343.63K   205KB/s              ^
```

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

```
thamizh@thamizh-HP-Pavilion-Laptop-15-eg2xxx ~
$ tar -xvf apache-hive-3.1.3-bin.tar.gz                                          2 ↵
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_bar.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

```
#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 MySQLby running below commands:*
$sudomysql

Pops command line interface for MySQLand 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
Replaceyour_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

```
thamizh@thamizh-HP-Pavilion-Laptop-15-eg2xxx ~/apache-hive-3.1.3-bin/bin
$ schematool -initSchema -dbType mysql
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/thamizh/apache-hive-3.1.3-bin/lib/log4j-slf4j-im
pl-2.17.1.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/thamizh/hadoop-3.3.6/share/hadoop/common/lib/slf
4j-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/metastore?createDatabaseIfNotExi
st=true
Metastore Connection Driver :    com.mysql.cj.jdbc.Driver
Metastore connection User:       root
```

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*

```
thamizh@thamizh-HP-Pavilion-Laptop-15-eg2xxx ~/apache-hive-3.1.3-bin/bin
$ hive                                                                    1 ↵
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/thamizh/apache-hive-3.1.3-bin/lib/log4j-slf4j-im
pl-2.17.1.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/thamizh/hadoop-3.3.6/share/hadoop/common/lib/slf
4j-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 = 392f1fb5-9bf3-43c8-b693-993cf9a470d2

Logging initialized using configuration in jar:file:/home/thamizh/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. Cons
ider using a different execution engine (i.e. spark, tez) or using Hive 1.X releases.
Hive Session ID = 458f1788-791d-44c3-b91f-dd74ee7eca99
hive> show databases;
OK
default
financials
Time taken: 0.417 seconds, Fetched: 2 row(s)
hive>
```

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

Reg.No.: 210701288

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;

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;
```

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

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

```
thamizh@thamizh-HP-Pavilion-Laptop-15-eg2xxx ~/apache-hive-3.1.3-bin/bin
$ hdfs dfs -cat /home/hadoop/emp.json

[
    {"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"}
]
```

RESULT:

Thus to import a JASON file from the command line and apply the following actions with the

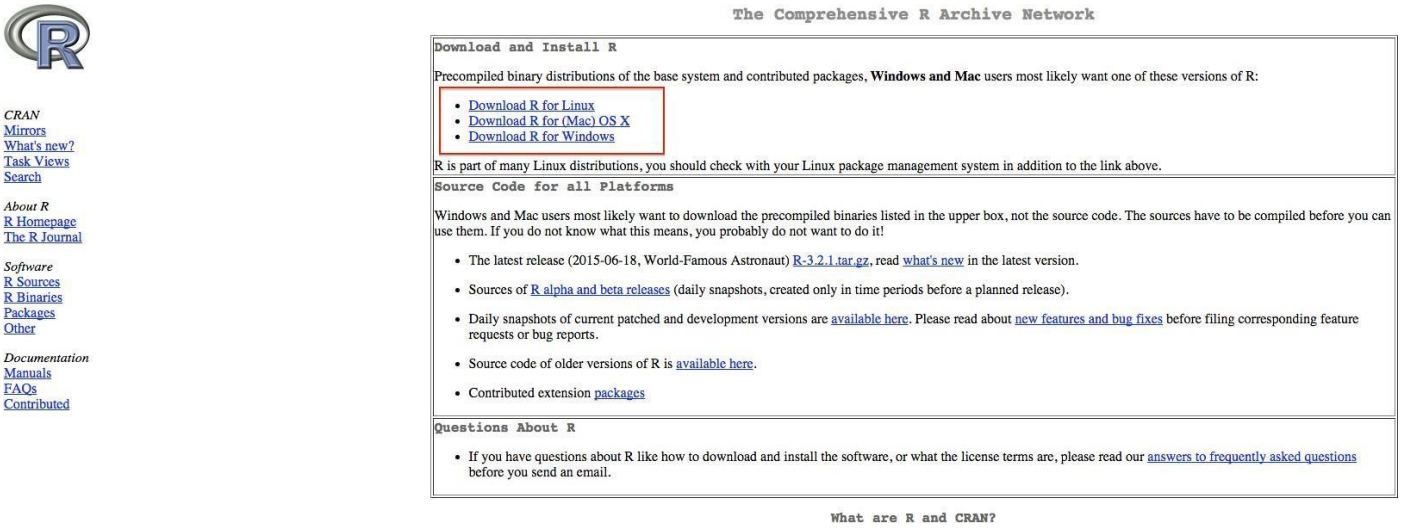
data present in the JASON file where, projection, aggregation, remove, count, limit, skip and sort has been executed and verified successfully.

Reg.No: 210701288

Installation guide for R and RStudio

Step 1 – Install R

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



The Comprehensive R Archive Network

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
Contributed

Download and Install R

Precompiled binary distributions of the base system and contributed packages, **Windows and Mac** users most likely want one of these versions of R:

- Download R for Linux
- Download R for (Mac) OS X
- Download R for Windows

R is part of many Linux distributions, you should check with your Linux package management system in addition to the link above.

Source Code for all Platforms

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!

- The latest release (2015-06-18, World-Famous Astronaut) R-3.2.1.tar.gz, read what's new in the latest version.
- Sources of R alpha and beta releases (daily snapshots, created only in time periods before a planned release).
- Daily snapshots of current patched and development versions are available here. Please read about new features and bug fixes before filing corresponding feature requests or bug reports.
- Source code of older versions of R is available here.
- Contributed extension packages

Questions About R

- If you have questions about R like how to download and install the software, or what the license terms are, please read our answers to frequently asked questions before you send an email.

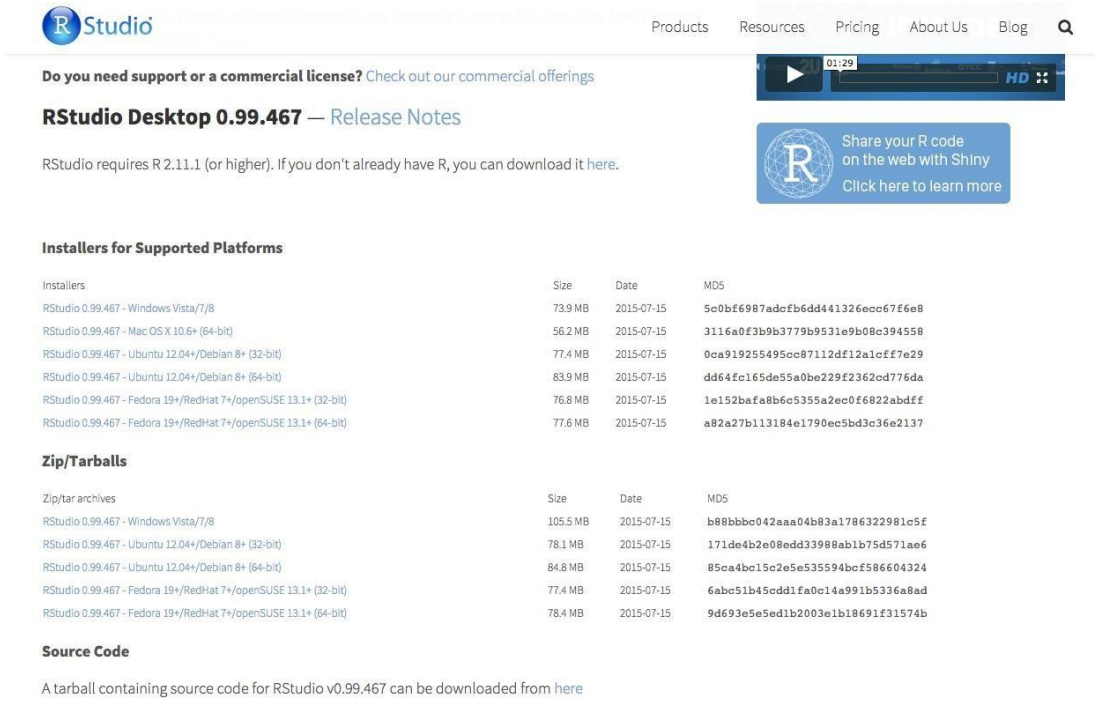
What are R and CRAN?

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/



RStudio

Products Resources Pricing About Us Blog

Do you need support or a commercial license? Check out our commercial offerings

RStudio Desktop 0.99.467 — Release Notes

RStudio requires R 2.11.1 (or higher). If you don't already have R, you can download it here.

Share your R code on the web with Shiny
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	5c0bf6987adcfb6dd441326ecc67f6e8
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+/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

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	85ca4bc15c2e5e535594bcf586604324
RStudio 0.99.467 - Fedora 19+/RedHat 7+/openSUSE 13.1+ (32-bit)	77.4 MB	2015-07-15	6abc51b45cdd1fa0c14a991b5336a8ad
RStudio 0.99.467 - Fedora 19+/RedHat 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

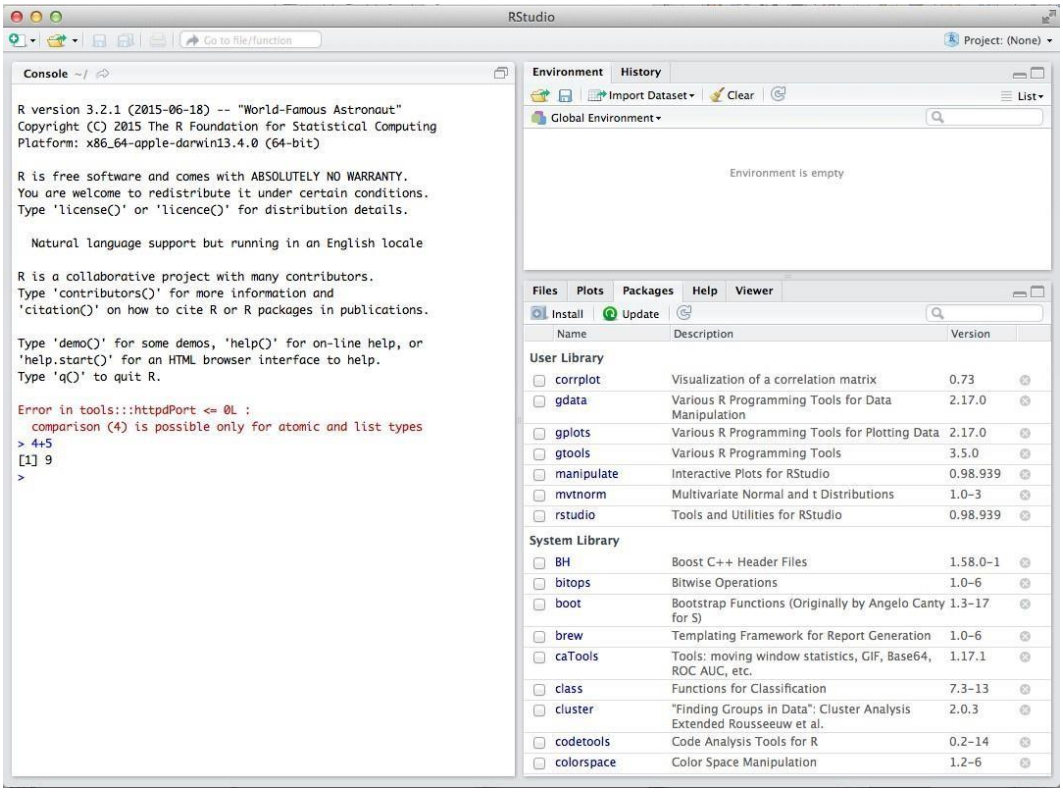


Figure 3. Running R with RStudio

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).

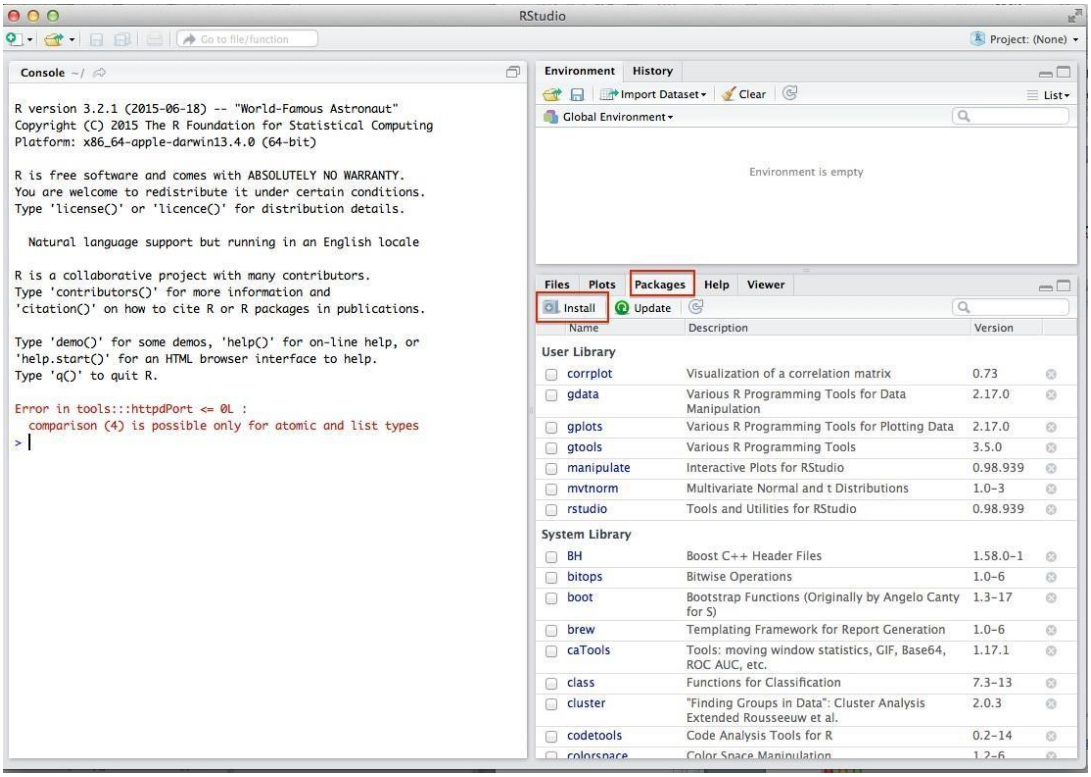


Figure 4. Click on Install to install R packages.

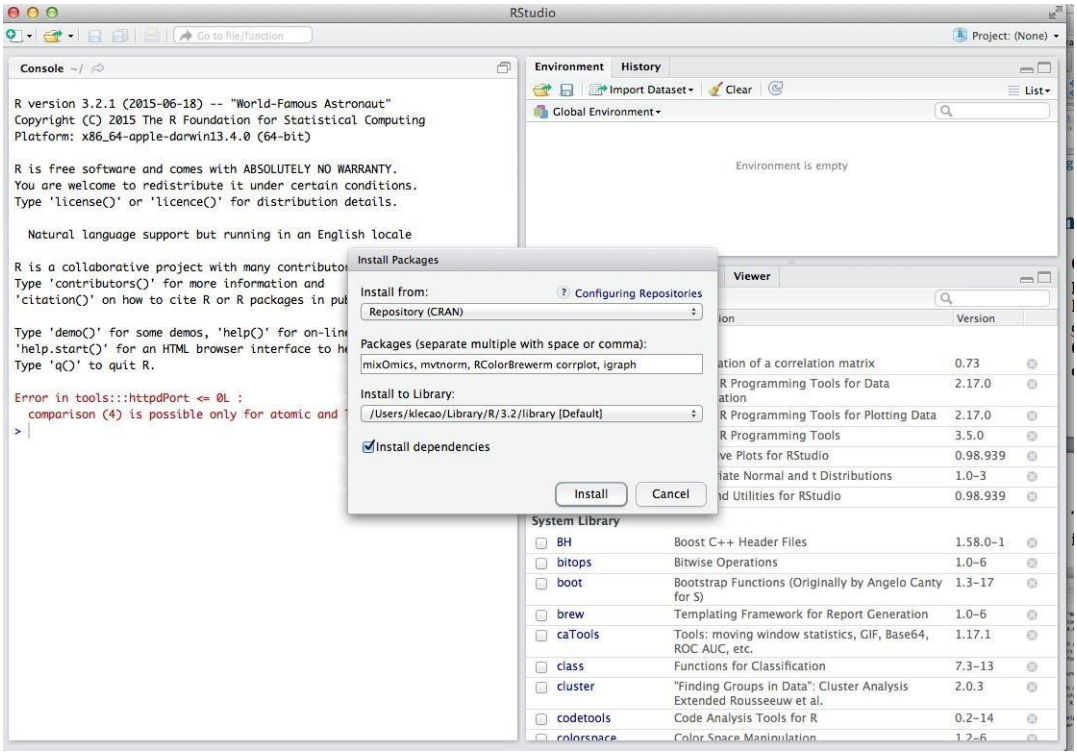


Figure 5. Specify the list of packages to be installed

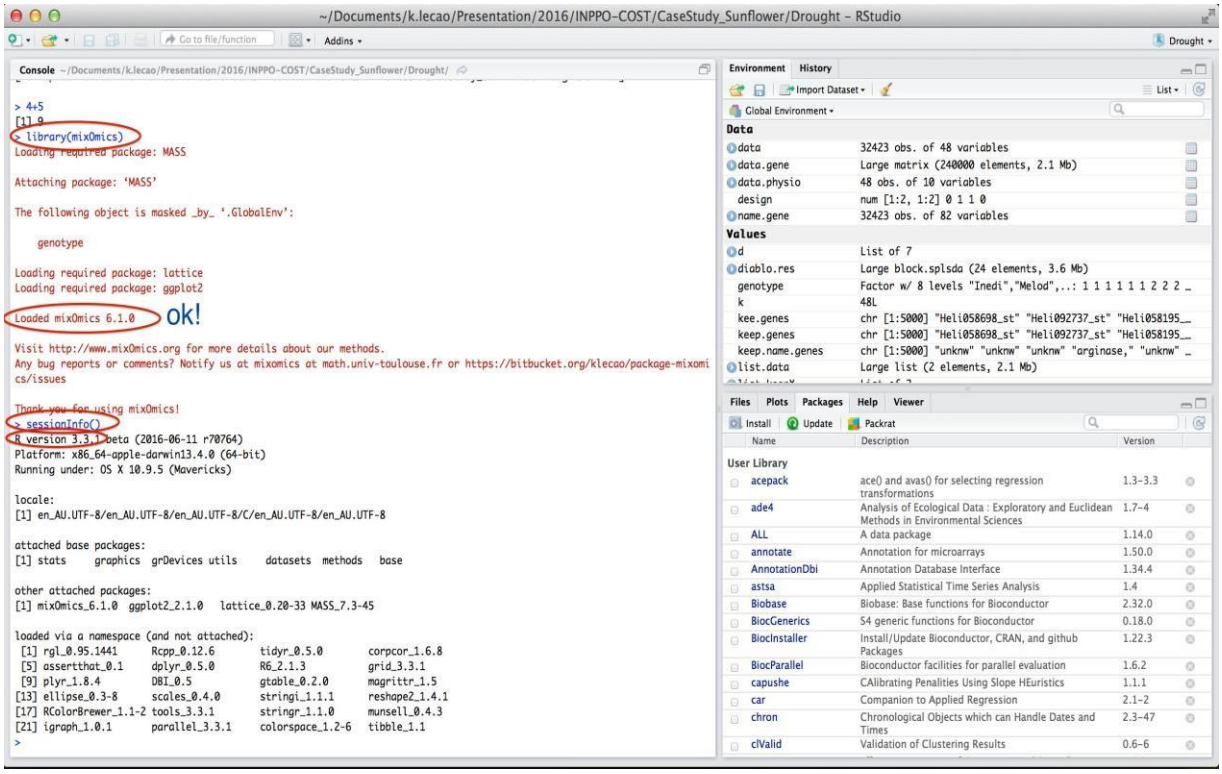


Figure 6. Check that the package mixOmics is installed and has the version 6.1.0.

Reg.No: 210701288

Exp.No: 7

IMPLEMENT LINEAR AND LOGISTIC REGRESSION

AIM:

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

PROCEDURE:

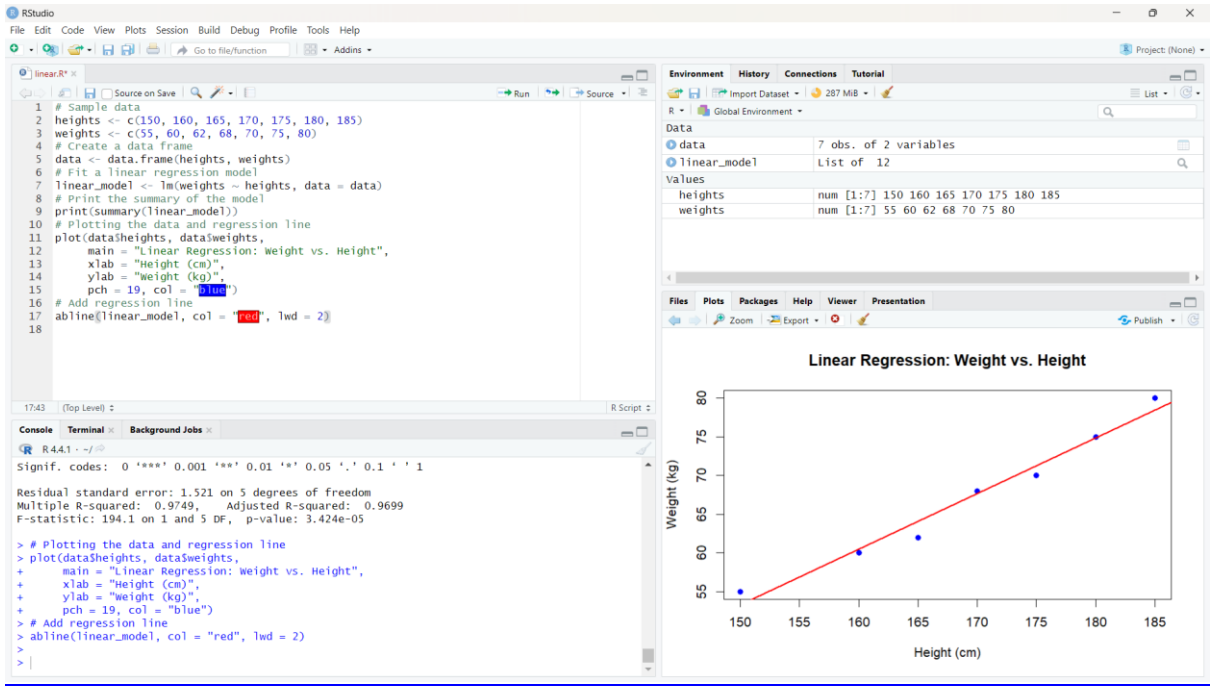
1. Create sample data for heights and weights, fit a linear regression model, and plot the data with the regression line.
2. Use the sample data to create a data frame for the regression model.
3. Fit the linear regression model using the `lm()` function and display the summary.
4. Plot the data points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

```
# 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)
```

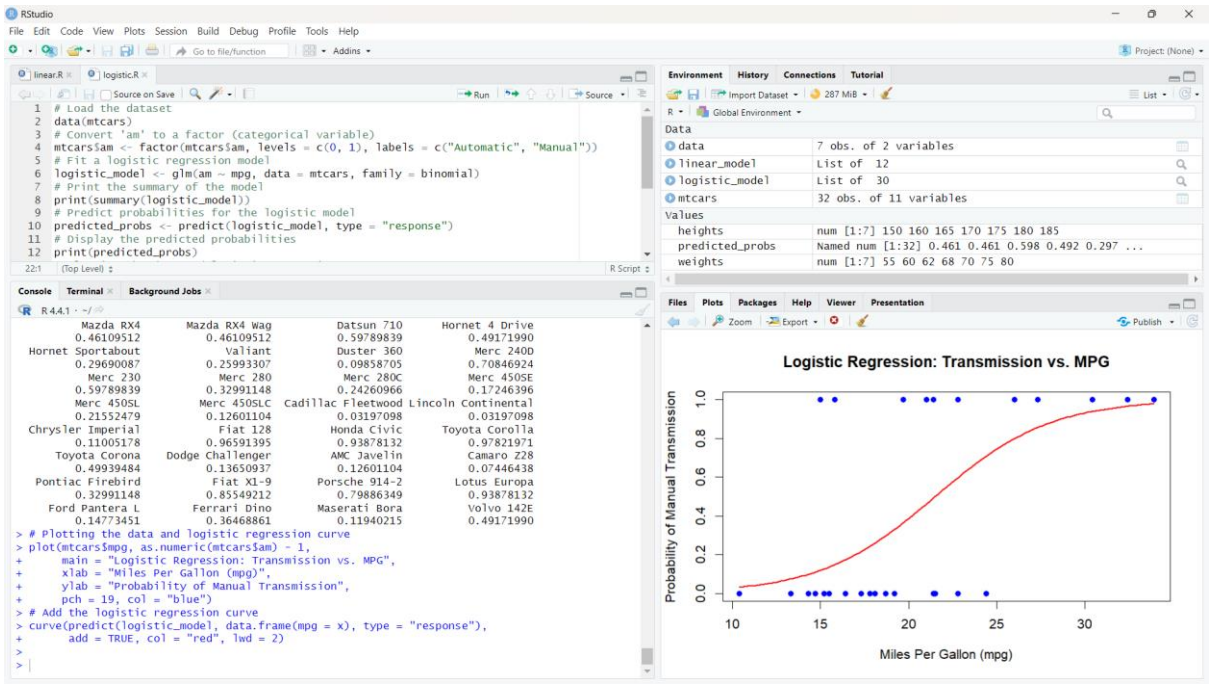
OUTPUT:



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.

Reg.No: 210701288

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.

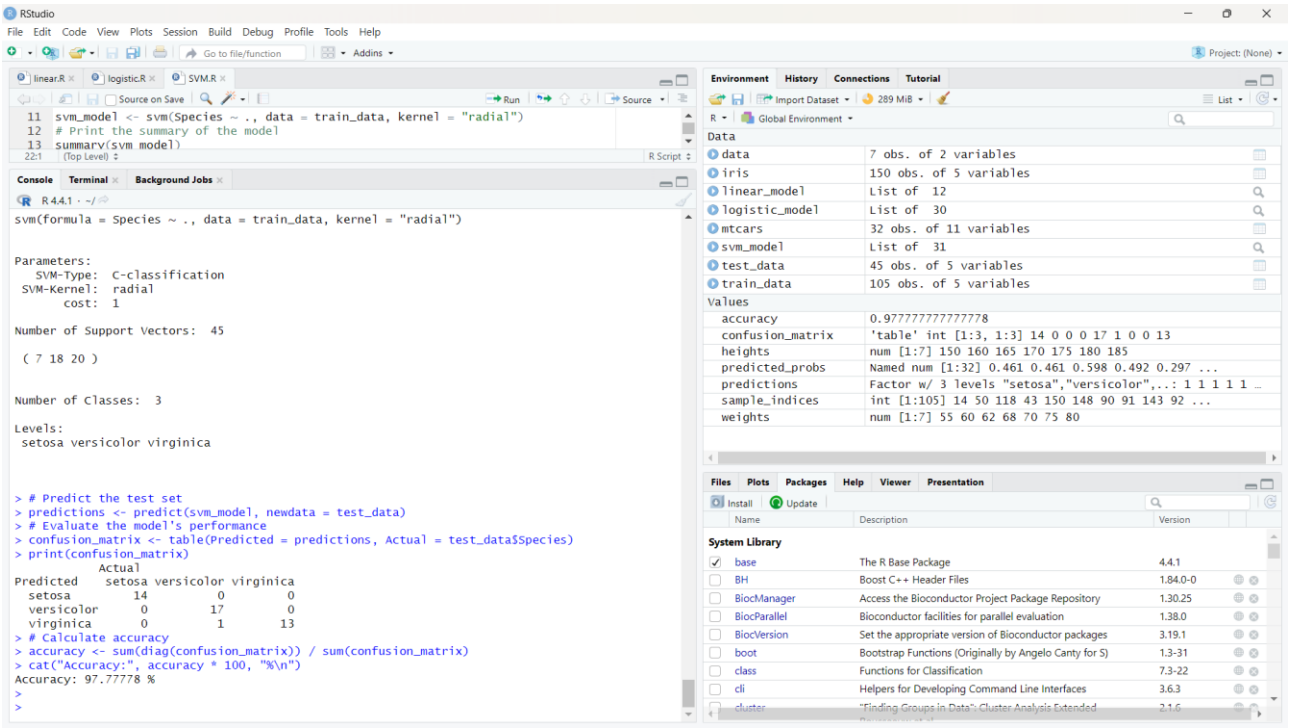
PROGRAM CODE:

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:

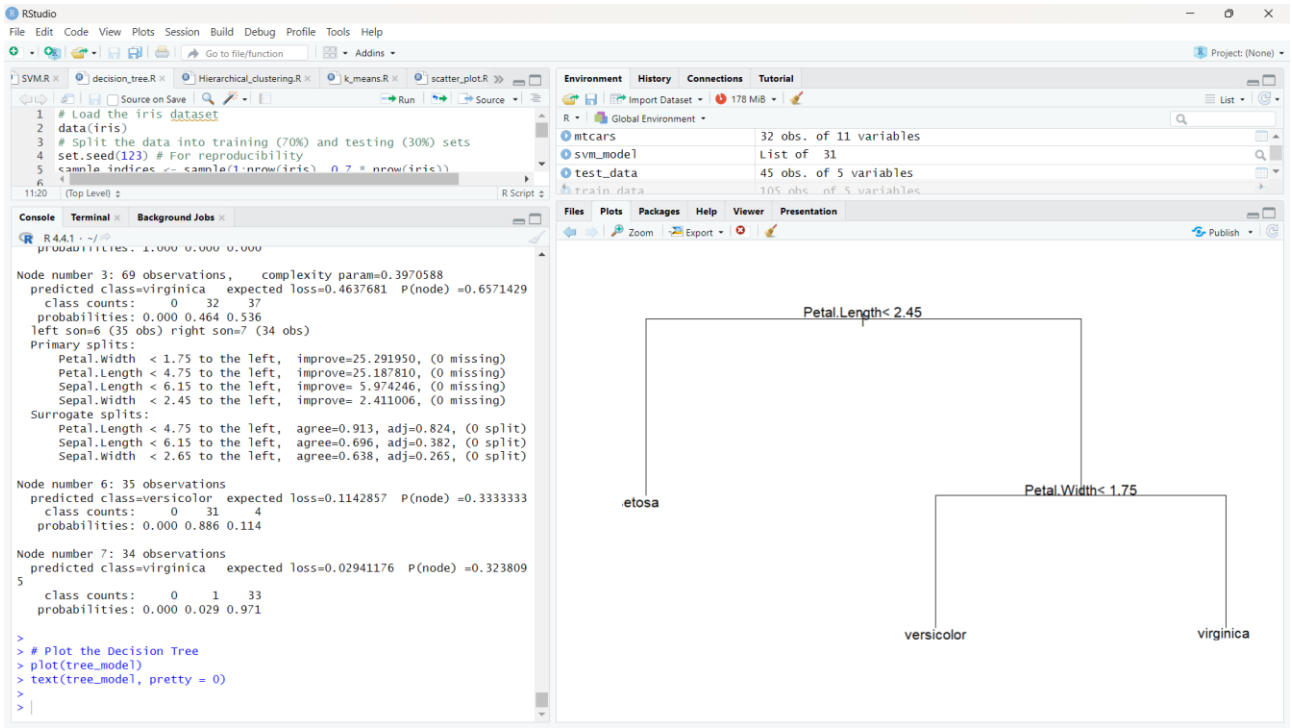


b) Decision tree in R

```
# 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:



RESULT:

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

Reg.No: 210701288

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.

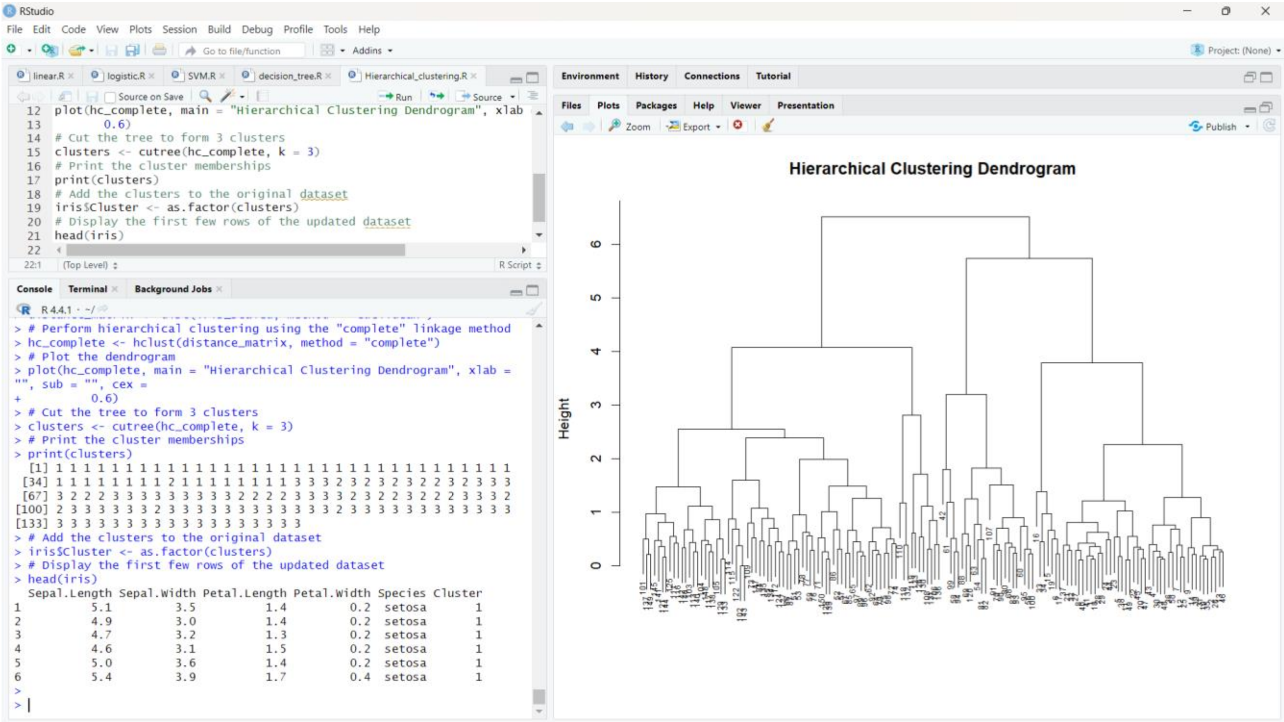
PROGRAM CODE:

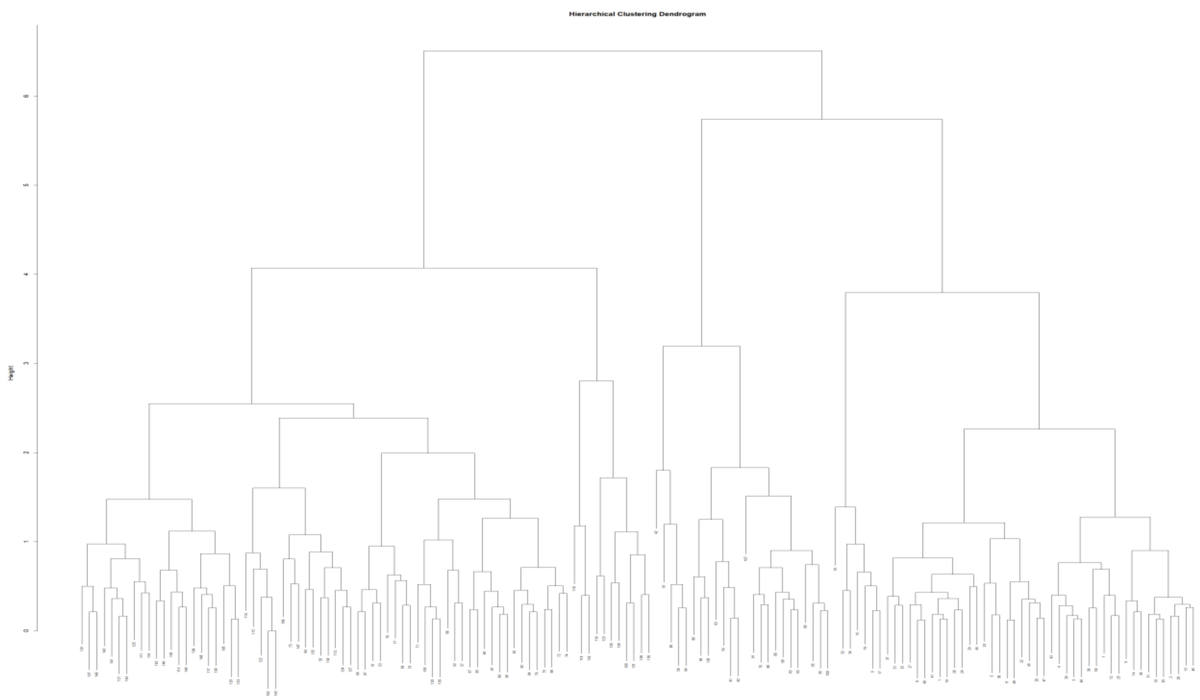
a) HIERARCHIAL CLUSTERING

```
# 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:

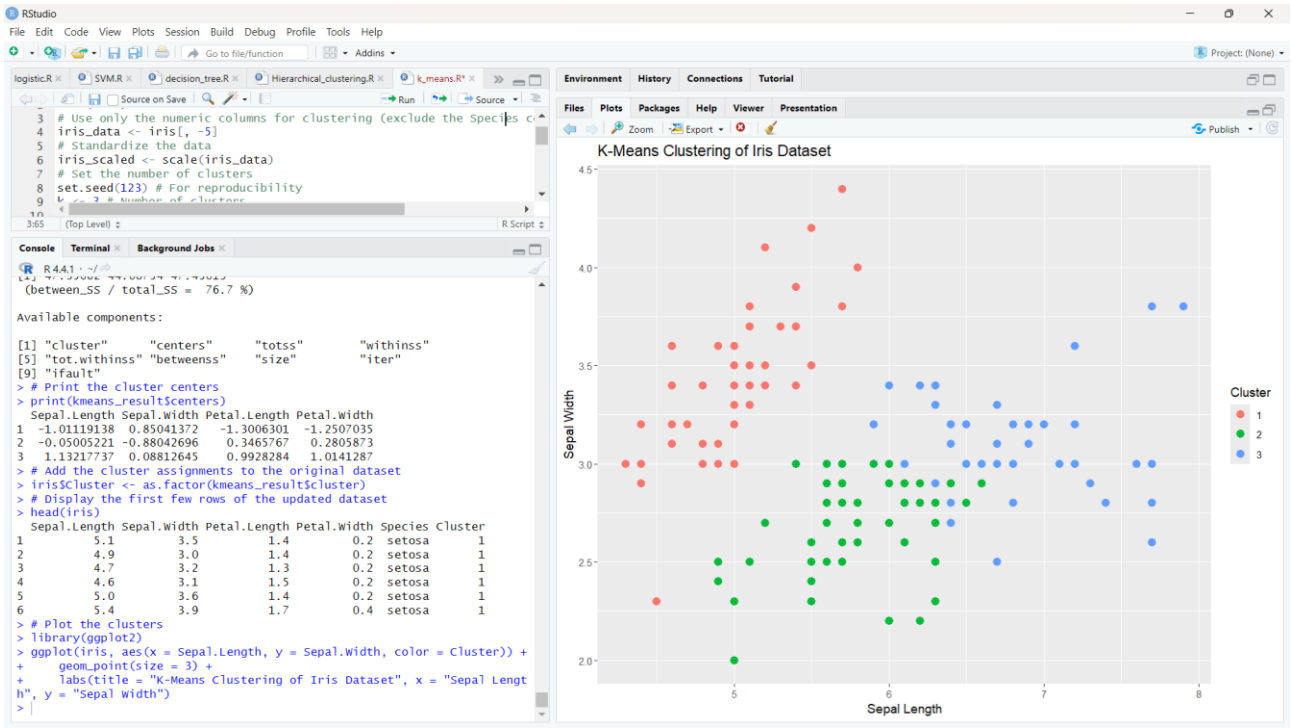




b) K-MEANS CLUSTERING

```
# 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.

Reg.No: 210701288

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.

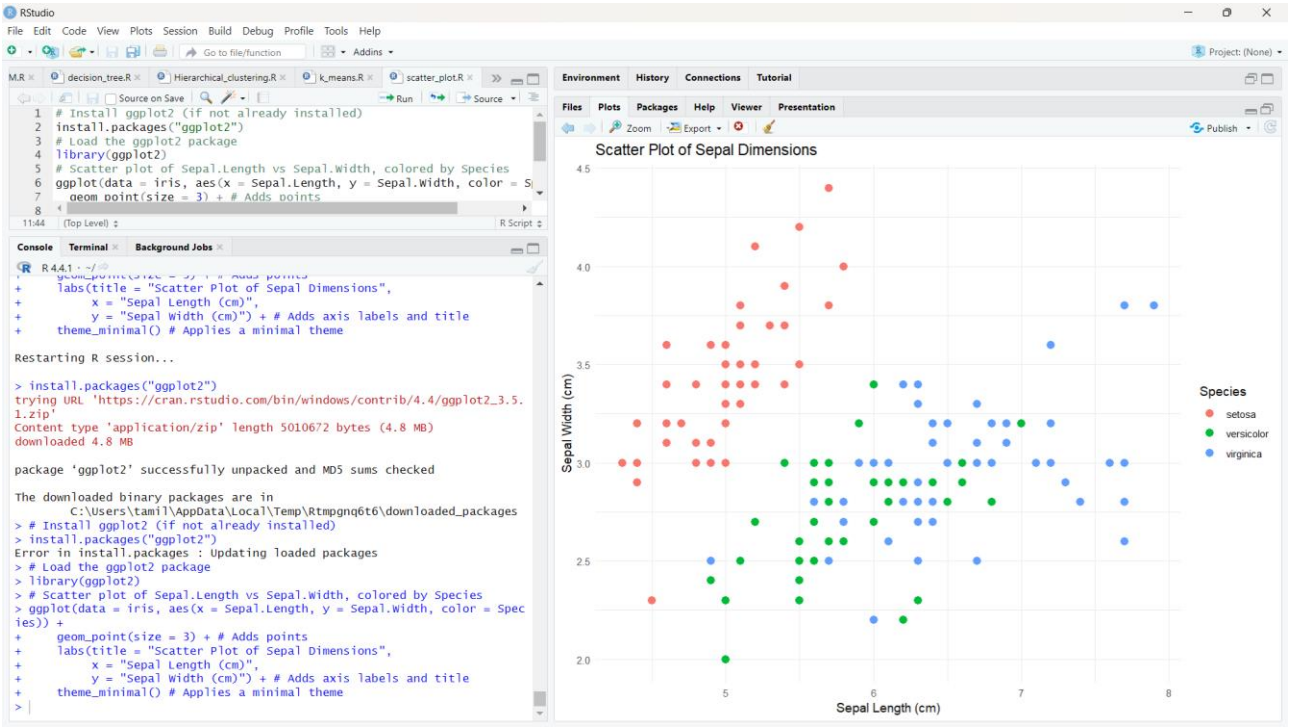
PROCEDURE:

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
```

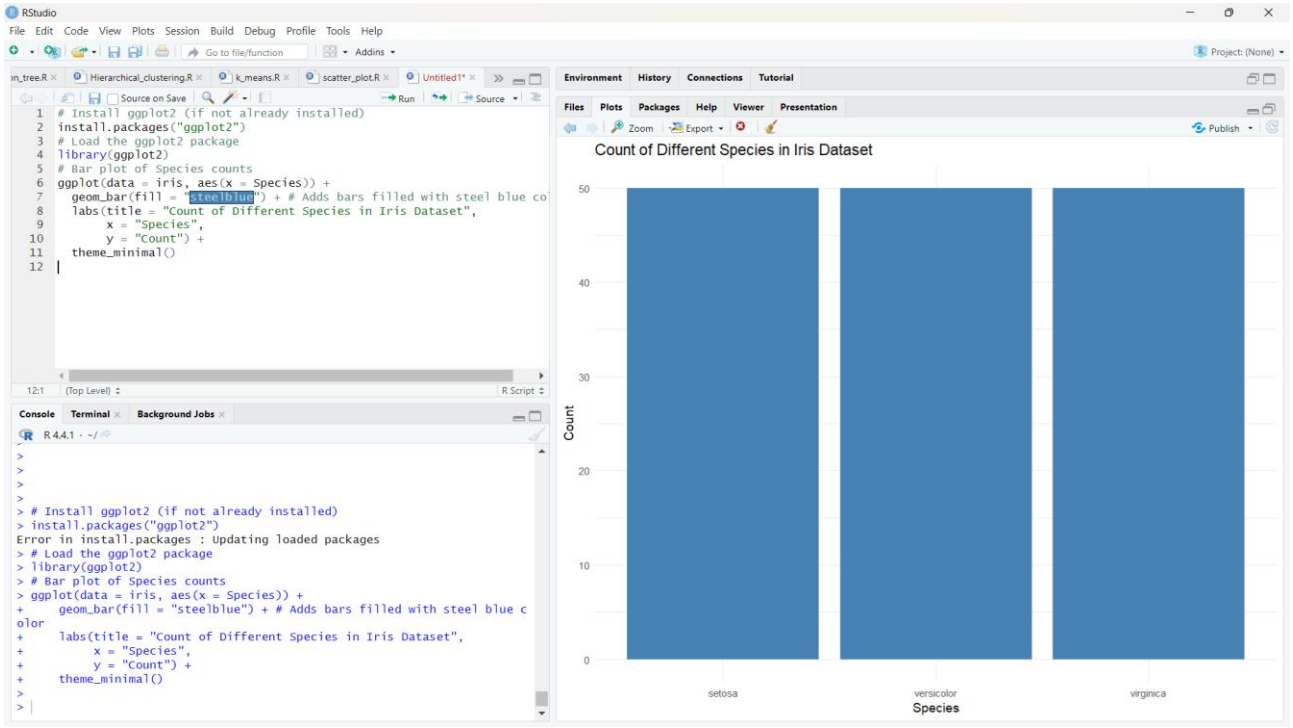
OUTPUT:



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()
```

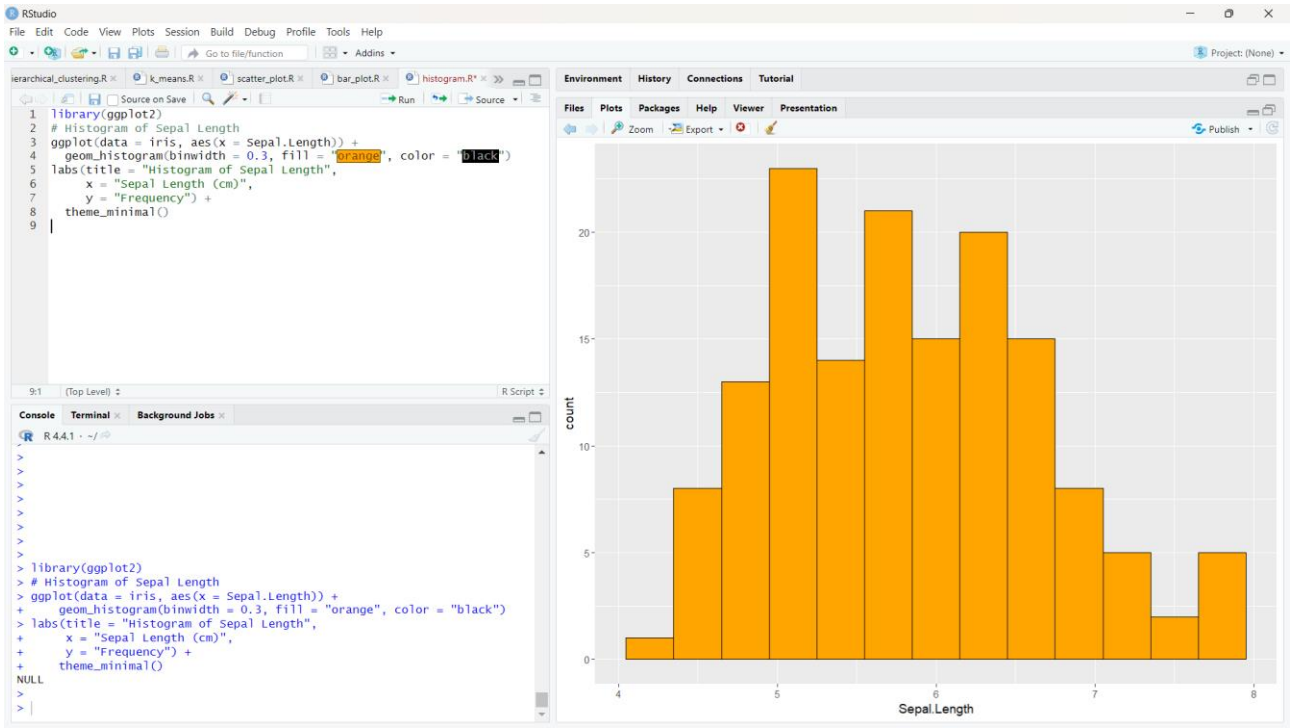
OUTPUT:



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()
```

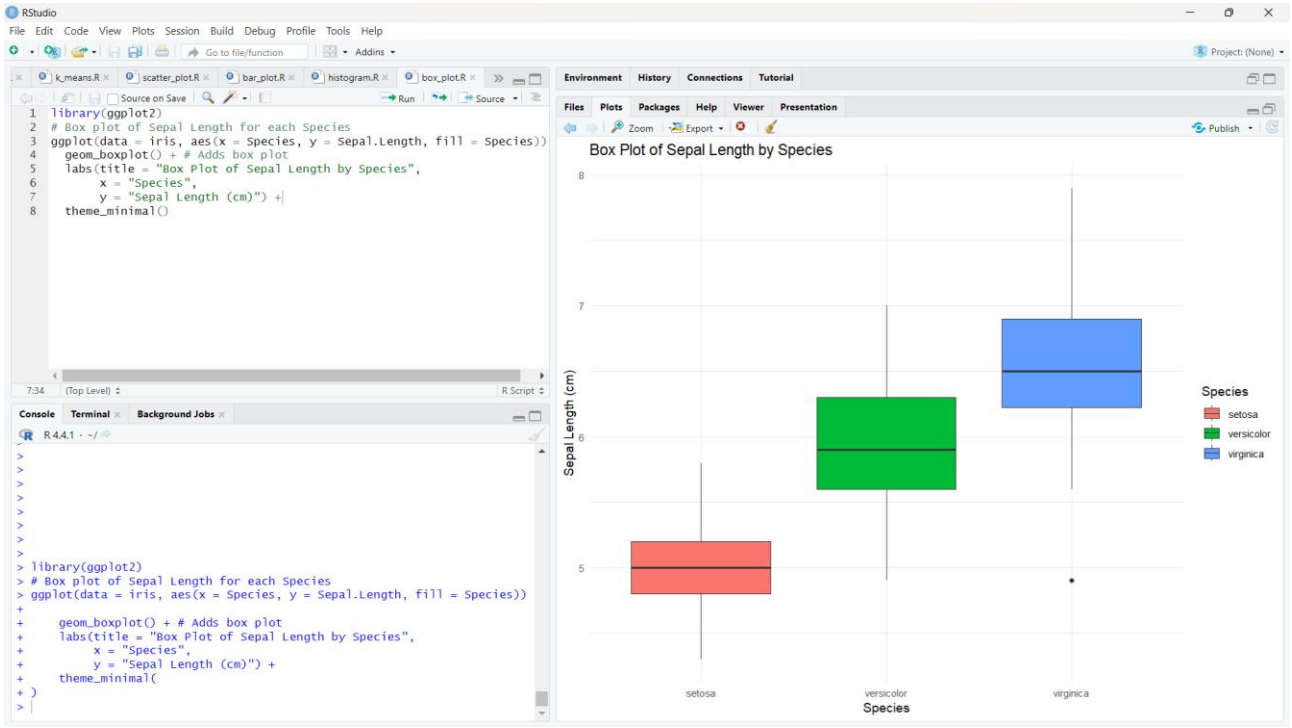
OUTPUT:



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.