

**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:**

```
sweatha@sweatha-HP-Pavilion-Laptop-15-eg2xxx:~$ java -version
openjdk version "1.8.0_422"
OpenJDK Runtime Environment (build 1.8.0_422-8u422-b05-1~24.04-b05)
OpenJDK 64-Bit Server VM (build 25.422-b05, mixed mode)
```

**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
```

```
sweatha@sweatha-HP-Pavilion-Laptop-15-eg2xxx:~$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/sweatha/.ssh/id_rsa):
/home/sweatha/.ssh/id_rsa already exists.
Overwrite (y/n)? y
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/sweatha/.ssh/id_rsa
Your public key has been saved in /home/sweatha/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:8cjzNmbyhR9DLOagkOedCRBVAsdNtfnRiyV8qUIKtg sweatha@sweatha-HP-Pavilion-Laptop-15-eg2xxx
The key's randomart image is:
+---[RSA 3072]---+
| o=o=... o.. |
| +.+. + = . |
| o. +. .o0 o |
| . E+.+ ..== |
| o.S o o.o |
| + + o = . |
| = . X . |
| . B 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:

```
sweatha@sweatha-HP-Pavilion-Laptop-15-eg2xxx:~$ ssh localhost
sweatha@localhost's password:
Welcome to Ubuntu 24.04.1 LTS (GNU/Linux 6.8.0-45-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.

0 updates can be applied immediately.

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

Last login: Tue Aug 20 11:22:10 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**

```
sweatha@sweatha-HP-Pavilion-Laptop-15-eg2xxx:~$ ls
'${system:java.io.tmpdir}'          Doc
ACKNOWLEDGMENTS                   Documents
apache-hive-3.1.3-bin              Downloads
apache-hive-3.1.3-bin.tar.gz       emp.json
build.xml                          google-cloud-sdk
CoreExposed.includes               google-cloud-sdk-433.0.0-linux-x86_64.tar.gz
db-derby-10.5.2.0-bin              grammar
db-derby-10.5.2.0-bin.tar.gz       hadoop
DEBIAN                            hadoop-3.4.0
Demo                               hadoop-data
derby.log                           hadoop-http-auth-signature-secret
derby.policy                        index.html
Desktop                           input.txt
```

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

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

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

**\$ nano ~/.bashrc**

Append the below lines to file.

```

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

```

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

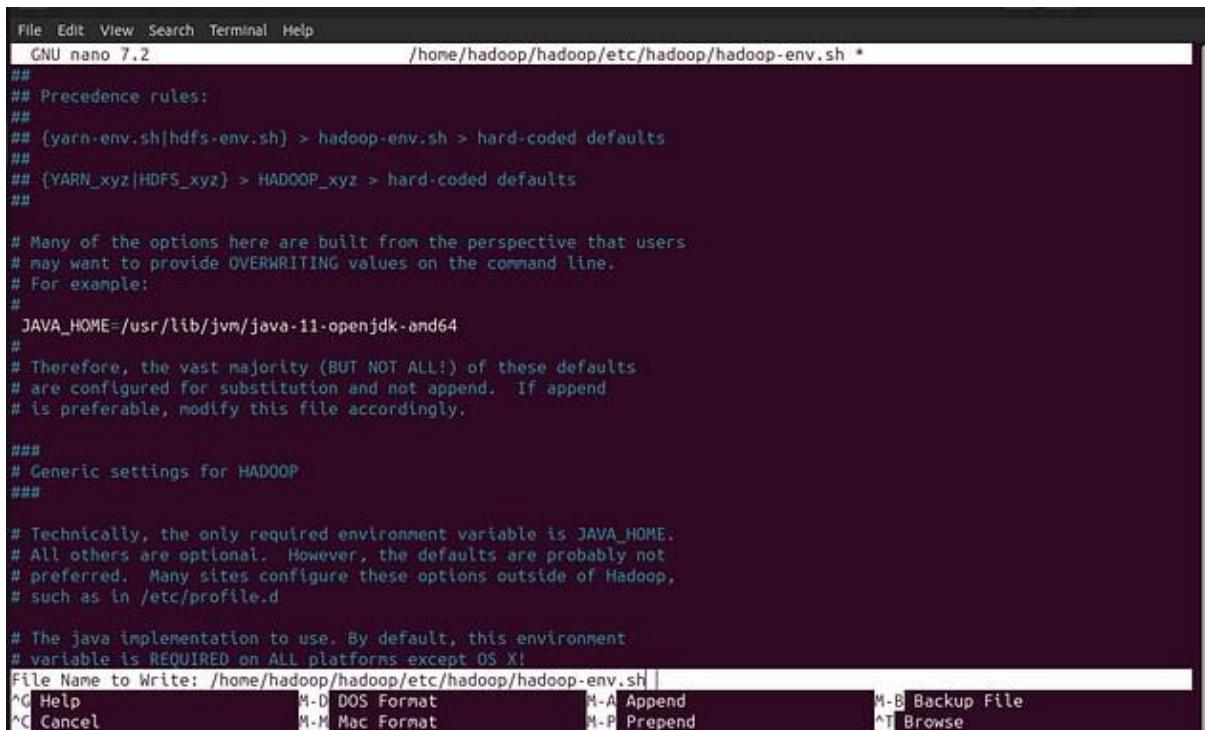
**s\$ source ~/.bashrc**

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

**\$HADOOP\_HOME/etc/hadoop/hadoop-env.sh**

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

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



```

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

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

### 
# Generic settings for HADOOP
###

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

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

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

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

**\$ jps**

```
sweatha@sweatha-HP-Pavilion-Laptop-15-eg2xxx:~$ jps
78114 DataNode
35715 ResourceManager
78569 Jps
37708 NodeManager
77951 NameNode
78398 SecondaryNameNode
```

### 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>

The screenshot shows the Hadoop NameNode Overview page. At the top, it says "Overview 'localhost:9000' (active)". Below that is a table with system details:

Started:	Mon Sep 02 18:10:05 +0530 2024
Version:	3.4.0, rbd8b7f308f626bb7791783192ee7a5dfaec760
Compiled:	Mon Mar 04 12:05:00 +0530 2024 by root from (HEAD detached at release-3.4.0-RC3)
Cluster ID:	CID-03b6980f-2609-4ba0-9788-d47f4a0a505b
Block Pool ID:	BP-1528023459-127.0.1.1-1725279877319

Below the table is a "Summary" section with the following details:

Configured Capacity:	77.95 GB
Configured Remote Capacity:	0 B
DFS Used:	1.79 MB (0%)
Non DFS Used:	16 GB
DFS Remaining:	57.95 GB (74.34%)
Block Pool Used:	1.79 MB (0%)
DataNodes usages% (Min/Median/Max/stdDev):	0.00% / 0.00% / 0.00% / 0.00%
Live Nodes	1 (Decommissioned: 0, In Maintenance: 0)
Dead Nodes	0 (Decommissioned: 0, In Maintenance: 0)
Decommissioning Nodes	0
Entering Maintenance Nodes	0
Total DataNode Volume Failures	0 / 0 / 0

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>

ID	User	Name	Application Type	Application Tags	Queue	Application Priority	StartTime	LaunchTime	FinishTime	State	FinalStatus	Ri Co
application_1725279194116_0005	sweatha	streamjob2425137979962209561.jar	MAPREDUCE		root.default	0	Mon Sep 2 19:02:59 +0550 2024	Mon Sep 2 19:02:59 +0550 2024	Mon Sep 2 19:03:09 +0550 2024	FINISHED	SUCCEEDED	N/A
application_1725279194116_0004	sweatha	streamjob1666684073044514851.jar	MAPREDUCE		root.default	0	Mon Sep 2 19:00:25 +0550 2024	Mon Sep 2 19:00:25 +0550 2024	Mon Sep 2 19:00:38 +0550 2024	FINISHED	FAILED	N/A
application_1725279194116_0003	sweatha	streamjob505449069662450369.jar	MAPREDUCE		root.default	0	Mon Sep 2 18:50:11 +0550 2024	Mon Sep 2 18:50:12 +0550 2024	Mon Sep 2 18:50:33 +0550 2024	FINISHED	FAILED	N/A
application_1725279194116_0002	sweatha	streamjob9103332973557097248.jar	MAPREDUCE		root.default	0	Mon Sep 2 18:31:26 +0550 2024	Mon Sep 2 18:31:26 +0550 2024	Mon Sep 2 18:31:47 +0550 2024	FINISHED	FAILED	N/A
application_1725279194116_0001	sweatha	QuasiMonteCarlo	MAPREDUCE		root.default	0	Mon Sep 2 18:14:37 +0550 2024	Mon Sep 2 18:14:38 +0550 2024	Mon Sep 2 18:15:01 +0550 2024	FINISHED	SUCCEEDED	N/A

## 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:

```
sweatha@sweatha-HP-Pavilion-Laptop-15-eg2xxx:~$ hdfs dfs -ls /
Found 5 items
drwxr-xr-x  - sweatha supergroup          0 2024-09-20 13:44 /home
drwxrwx-wx  - sweatha supergroup          0 2024-09-20 12:17 /tmp
drwxr-xr-x  - sweatha supergroup          0 2024-09-20 12:17 /user
drwxr-xr-x  - sweatha supergroup          0 2024-09-20 14:14 /weather_data
drwxr-xr-x  - sweatha supergroup          0 2024-09-20 13:57 /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:

	Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
<input type="checkbox"/>	drwxr-xr-x	sweatha	supergroup	0 B	Sep 20 13:44	0	0 B	home
<input type="checkbox"/>	drwxrwxrwx	sweatha	supergroup	0 B	Sep 20 12:17	0	0 B	tmp
<input type="checkbox"/>	drwxr-xr-x	sweatha	supergroup	0 B	Sep 20 12:17	0	0 B	user
<input type="checkbox"/>	drwxr-xr-x	sweatha	supergroup	0 B	Sep 20 14:14	0	0 B	weather_data
<input type="checkbox"/>	drwxr-xr-x	sweatha	supergroup	0 B	Sep 20 13:57	0	0 B	word_count_in_python

Showing 1 to 5 of 5 entries

Hadoop, 2024.

## Step 15 – Stop Hadoop Cluster

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

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

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

## Result:

The step-by-step installation and configuration of Hadoop on Ubuntu 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

```
Open ▾ input.txt
hadoop-config.sh          hadoop-http-auth-signature-secret          input.txt
                           .                                     .
                           .                                     .
                           .                                     .
                           hello world
                           hello hadoop
                          .hadoop is great
                           .
```

**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
for line in sys.stdin:
    word, count = line.strip().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
hdfs dfs -mkdir /word_count_in_python
hdfs dfs -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
```

```
sweatha@sweatha-HP-Pavilion-Laptop-15-eg2xxx: $ hadoop jar $HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming*.jar -input /input/input.txt -output /output -mapper /home/sweatha/mapper.py -reducer /home/sweatha/reducer.py
packageJobJar: [/tmp/hadoop-unjar863823414226844503/] [] /tmp/streamjob2425137979962209561.jar tmpDir=null
2024-09-02 19:02:58,632 INFO client.DefaultNoHARNFailoverProxyProvider: Connecting to ResourceManager at /0.0.0.0:8032
2024-09-02 19:02:58,710 INFO client.DefaultNoHARNFailoverProxyProvider: Connecting to ResourceManager at /0.0.0.0:8032
2024-09-02 19:02:58,850 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /tmp/hadoop-yarn/staging/sweatha/.staging/job_1725279194116_0005
2024-09-02 19:02:59,002 INFO mapred.FileInputFormat: Total input files to process : 1
2024-09-02 19:02:59,036 INFO mapreduce.JobSubmitter: number of splits:2
2024-09-02 19:02:59,106 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1725279194116_0005
2024-09-02 19:02:59,106 INFO mapreduce.JobSubmitter: Executing with tokens: []
2024-09-02 19:02:59,223 INFO conf.Configuration: resource-types.xml not found
2024-09-02 19:02:59,224 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
2024-09-02 19:02:59,264 INFO impl.YarnClientImpl: Submitted application application_1725279194116_0005
2024-09-02 19:02:59,285 INFO mapreduce.Job: The url to track the job: http://sweatha-HP-Pavilion-Laptop-15-eg2xxx:8088/proxy/application_1725279194116_0005/
2024-09-02 19:02:59,285 INFO mapreduce.Job: Running job: job_1725279194116_0005
2024-09-02 19:03:03,362 INFO mapreduce.Job: Job job_1725279194116_0005 running in uber mode : false
2024-09-02 19:03:03,364 INFO mapreduce.Job: map 0% reduce 0%
2024-09-02 19:03:06,410 INFO mapreduce.Job: map 100% reduce 0%
2024-09-02 19:03:10,451 INFO mapreduce.Job: map 100% reduce 100%
2024-09-02 19:03:10,469 INFO mapreduce.Job: Job job_1725279194116_0005 completed successfully
2024-09-02 19:03:10,528 INFO mapreduce.Job: Counters: 54
    File System Counters
        FILE: Number of bytes read=75
        FILE: Number of bytes written=934304
        FILE: Number of read operations=0
        FILE: Number of large read operations=0
        FILE: Number of write operations=0
        HDFS: Number of bytes read=240
        HDFS: Number of bytes written=38
        HDFS: Number of read operations=11
        HDFS: Number of large read operations=0
        HDFS: Number of write operations=2
        HDFS: Number of bytes read erasure-coded=0
    Job Counters
        Launched map tasks=2
        Launched reduce tasks=1
        Data-local map tasks=2
        Total time spent by all maps in occupied slots (ms)=2802
        Total time spent by all reducers in occupied slots (ms)=1220
```

## 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
```

```
sweatha@sweatha-HP-Pavilion-Laptop-15-eg2xxx:~$ hadoop fs -ls /output/
Found 2 items
-rw-r--r-- 1 sweatha supergroup      0 2024-09-02 19:03 /output/_SUCCESS
-rw-r--r-- 1 sweatha supergroup 38 2024-09-02 19:03 /output/part-00000
sweatha@sweatha-HP-Pavilion-Laptop-15-eg2xxx:~$ hadoop fs -cat /output/part-00000
great    1
hadoop   2
hello    2
is       1
world   1
sweatha@sweatha-HP-Pavilion-Laptop-15-eg2xxx:~$
```

## 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 LibreOffice Calc spreadsheet titled "weather\_data.csv". The data is organized into three columns: "date", "temperature", and "humidity". The rows contain the following data:

	date	temperature	humidity
1	2024-09-01	25	60
2	2024-09-01	27	65
3	2024-09-02	23	55
4	2024-09-02	22	50
5	2024-09-03	24	70
6	2024-09-03	26	72
7			
8			
9			
10			
11			
12			
13			
14			
15			

**Step 2: Mapper Logic - mapper.py:**

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

```

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

#!/usr/bin/env python

import sys

# input comes from STDIN (standard input)
# the mapper will get daily max temperature and group it by month. so output will be
#(month,dailymax_temperature)
for line in sys.stdin:

```

```

# remove leading and trailing whitespace
line = line.strip() # split
the line into words words =
line.split()

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

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

```

### Step 3: Reducer Logic - reducer.py:

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

```

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

```

#### reducer.py

```

#!/usr/bin/env python

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

# input comes from STDIN for
line in sys.stdin:
    # remove leading and trailing whitespace line
= line.strip()
    # parse the input we got from mapper.py month,
daily_max = line.split('\t', 1)

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

```

```

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

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

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

```

#### **Step 4: Prepare Hadoop Environment:**

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

```
start-all.sh
```

#### **Step 6: Make Python Files Executable:**

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

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

```

sweatha@sweatha-HP-Pavilion-Laptop-15-eg2xxx:~$ hdfs dfs -put /home/sweatha/weather_data.csv /weather_data/
put: `/weather_data/': No such file or directory: `hdfs://localhost:9000/weather_data'
sweatha@sweatha-HP-Pavilion-Laptop-15-eg2xxx:~$ hdfs dfs -mkdir /weather_data
sweatha@sweatha-HP-Pavilion-Laptop-15-eg2xxx:~$ hdfs dfs -put /home/sweatha/weather_data.csv /weather_data/
2024-08-22 16:22:00,444 [HDFS-DataStreamer-Port-Stream-0] INFO DataStreamer - Execution

```

#### **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
```

```
sweatha@sweatha-HP-Pavilion-Laptop-15-eg2xxx: $ hadoop jar SHADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-.jar      -input /weather_data/weather_data.csv      -output /weather_dat
a/output      -mapper /home/sweatha/happerner.py      -reducer /home/sweatha/reducerw.py
packageJobJar: [/tmp/hadoop-unjar9181753598584659111/] [] /tmp/streamjob1366015123809532590.jar tmpDir=null
2024-09-02 22:37:07.619 INFO client.DefaultNoHARMFailoverProxyProvider: Connecting to ResourceManager at /0.0.0.0:8032
2024-09-02 22:37:07.717 INFO client.DefaultNoHARMFailoverProxyProvider: Connecting to ResourceManager at /0.0.0.0:8032
2024-09-02 22:37:08.014 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /tmp/hadoop-yarn/staging/sweatha/.staging/job_1725294573905_0004
2024-09-02 22:37:08.385 INFO mapred.FileInputFormat: Total input files to process : 1
2024-09-02 22:37:08.791 INFO mapreduce.JobSubmitter: number of splits:2
2024-09-02 22:37:08.947 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1725294573905_0004
2024-09-02 22:37:08.947 INFO mapreduce.JobSubmitter: Executing with tokens: []
2024-09-02 22:37:09.172 INFO conf.Configuration: resource-types.xml not found
2024-09-02 22:37:09.172 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
2024-09-02 22:37:09.233 INFO impl.YarnClientImpl: Submitted application application_1725294573905_0004
2024-09-02 22:37:09.264 INFO mapreduce.Job: The url to track the job: http://sweatha-HP-Pavilion-Laptop-15-eg2xxx:8088/proxy/application_1725294573905_0004/
2024-09-02 22:37:09.264 INFO mapreduce.Job: Running job: job_1725294573905_0004
2024-09-02 22:37:14.354 INFO mapreduce.Job: Job job_1725294573905_0004 running in uber mode : false
2024-09-02 22:37:14.354 INFO mapreduce.Job: map 0% reduce 0%
2024-09-02 22:37:19.466 INFO mapreduce.Job: map 100% reduce 0%
2024-09-02 22:37:24.528 INFO mapreduce.Job: map 100% reduce 100%
2024-09-02 22:37:25.552 INFO mapreduce.Job: Job job_1725294573905_0004 completed successfully
2024-09-02 22:37:25.685 INFO mapreduce.Job: Counters: 54
  File System Counters
    FILE: Number of bytes read=114
    FILE: Number of bytes written=932585
    FILE: Number of read operations=0
    FILE: Number of large read operations=0
    FILE: Number of write operations=0
    HDFS: Number of bytes read=416
```

## 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
```

```
sweatha@sweatha-HP-Pavilion-Laptop-15-eg2xxx: $ hdfs dfs -cat /weather_data/output/part-*
2024-09-01      26.0
2024-09-02      22.5
2024-09-03      25.0
sweatha@sweatha-HP-Pavilion-Laptop-15-eg2xxx: $
```

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:

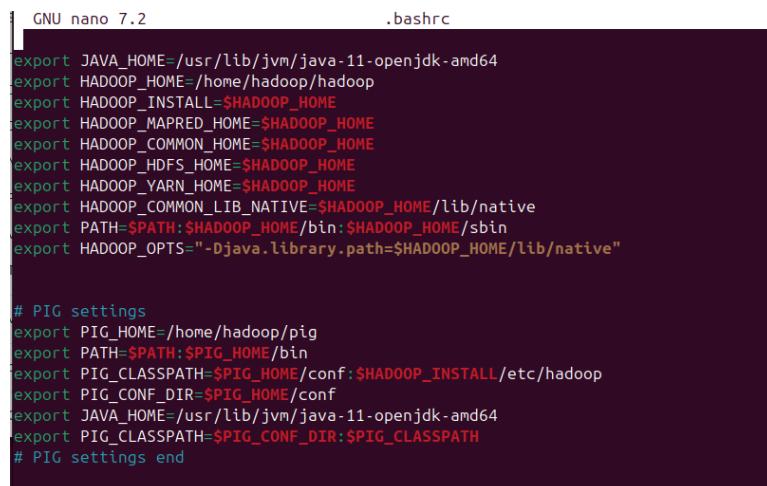
```
$ sudo mv /home/hadoop/pig-0.16.0 /home/hadoop/pig
```

**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 settings
export PIG_HOME=/home/hadoop/pig
export PATH=$PATH:$PIG_HOME/bin
export PIG_CLASSPATH=$PIG_HOME/conf:$HADOOP_INSTALL/etc/hadoop/export
export PIG_CONF_DIR=$PIG_HOME/conf
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
export PIG_CLASSPATH=$PIG_HOME:$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
```

```
sweatha@sweatha-HP-Pavilion-Laptop-15-eg2xxx:~$ jps
78114 DataNode
35715 ResourceManager
78569 Jps
37708 NodeManager
77951 NameNode
78398 SecondaryNameNode
```

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

```
sweatha@sweatha-HP-Pavilion-Laptop-15-eg2xxx:~$ pig -v
2024-09-03 10:59:19,620 INFO org.apache.pig.ExecTypeProvider: Trying ExecType : LOCAL
2024-09-03 10:59:19,620 INFO org.apache.pig.ExecTypeProvider: Trying ExecType : MAPREDUCE
2024-09-03 10:59:19,620 INFO org.apache.pig.ExecTypeProvider: Picked MAPREDUCE as the ExecType
2024-09-03 10:59:19,645 [main] INFO org.apache.pig.Main - Apache Pig version 0.17.0 (r1797306) compiled Jun 02 2017, 15:41:58
2024-09-03 10:59:19,645 [main] INFO org.apache.pig.Main - Logging error messages to: /home/sweatha/pig_1725341359642.log
2024-09-03 10:59:19,655 [main] INFO org.apache.pig.impl.util.Utils - Default bootstrap file /home/sweatha/.pigbootup not found
2024-09-03 10:59:19,816 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - mapred.job.tracker is deprecated. Instead, use mapreduce.jobtracker.address
2024-09-03 10:59:19,816 [main] INFO org.apache.pig.backend.hadoop.executionengine.HExecutionEngine - Connecting to hadoop file system at: hdfs://localhost:9000
2024-09-03 10:59:20,090 [main] INFO org.apache.pig.PigServer - Pig Script ID for the session: PIG-default-cc9e5bce-7e67-4f56-b672-df56764c0783
2024-09-03 10:59:20,090 [main] WARN org.apache.pig.PigServer - ATS is disabled since yarn.timeline-service.enabled set to false
grunt> nano script.pig
2024-09-03 10:59:20,201 [main] ERROR org.apache.pig.tools.grunt.Grunt - ERROR 1000: Error during parsing. Encountered "<IDENTIFIER> "nano "" at line 1, column 1.
Was expecting one of:
<EOF>
"Cat" ...
"cldir" ...
"fs" ...
"sh" ...
"cd" ...
"cp" ...
"copyFromLocal" ...
"copyToLocal" ...
"dump" ...
"\\"d" ...
"describe" ...
"\\"de" ...
"aliases" ...
"explain" ...
"\\"e" ...
"help" ...
"history" ...
"kill" ...
"ls" ...
"my" ...
"mkdir" ...
"pwd" ...
"quit" ...
"\\"q" ...
"register" ...
"rm" ...
"rmf" ...
"set" ...
"\\"set" ..."
```

**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
```

```
sweatha@sweatha-HP-Pavilion-Laptop-15-eg2xxx:~$ nano script.py
sweatha@sweatha-HP-Pavilion-Laptop-15-eg2xxx:~$ pig -x local script.py
2024-09-03 11:04:44,433 INFO pig.ExecTypeProvider: Trying ExecType : LOCAL
2024-09-03 11:04:44,434 INFO pig.ExecTypeProvider: Picked LOCAL as the ExecType
2024-09-03 11:04:44,466 [main] INFO org.apache.pig.Main - Apache Pig version 0.17.0 (r1797386) compiled Jun 02 2017, 15:41:58
2024-09-03 11:04:44,466 [main] INFO org.apache.pig.Main - Logging error messages to: /home/sweatha/pig_1725341684459.log
2024-09-03 11:04:44,511 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - user.name is deprecated. Instead, use mapreduce.job.user.name
2024-09-03 11:04:44,647 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - Default bootstrap file /home/sweatha/.pigbootup not found
2024-09-03 11:04:44,647 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - mapred.job.tracker is deprecated. Instead, use mapreduce.jobtracker.address
2024-09-03 11:04:44,649 [main] INFO org.apache.pig.backend.hadoop.executionengine.HExecutionEngine - Connecting to hadoop file system at: file:///
2024-09-03 11:04:44,662 [main] INFO org.apache.pig.PigServer - Pig Script ID for the session: PIG-script.pig-7fb48a5-7499-429f-a20a-c004b1b3de8
2024-09-03 11:04:44,662 [main] WARN org.apache.pig.PigServer - ATS is defined since yarn.timeline-service.enabled.set to false
2024-09-03 11:04:44,691 [main] INFO org.apache.pig.scripting.jython.JythonScriptEngine - created tmp.python.cachedir=/tmp/pig_jython_1259722673247941174
2024-09-03 11:04:47,075 [main] INFO org.apache.pig.scripting.jython.JythonScriptEngine - Register scripting UDF: pycscript.to_upper
2024-09-03 11:04:47,188 [main] INFO org.apache.pig.scripting.jython.JythonFunction - Schema 'word:chararray' defined for func to_upper
2024-09-03 11:04:47,242 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - mapred.textoutputformat.separator is deprecated. Instead, use mapreduce.output.textoutputformat.separator
2024-09-03 11:04:47,254 [main] INFO org.apache.pig.tools.pigstats.ScriptState - Pig features used in the script: UNKNOWN
2024-09-03 11:04:47,284 [main] INFO org.apache.pig.newplan.logical.optimizer.LogicalPlanOptimizer - RULES_ENABLED=[AddForEach, ColumnMapKeyPrune, ConstantCalculator, GroupByConstParallelSetter, LimitOptimizer, LoadTypeCastInserter, MergeFilter, MergeForEach, NestedLimitOptimizer, PartitionFilterOptimizer, PredicatePushdownOptimizer, PushDownForEachFlatten, PushUpFilter, StreamInserter]
2024-09-03 11:04:47,338 [main] INFO org.apache.pig.impl.util.SpillableMemoryManager - Selected heap (PS Old Gen) of size 699400192 to monitor. collectionUsageThreshold = 489580128, usageThreshold = 489580128
2024-09-03 11:04:47,389 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MRCompiler - File concatenation threshold: 100 optimistic? false
2024-09-03 11:04:47,402 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MultiQueryOptimizer - MR plan size before optimization: 1
2024-09-03 11:04:47,402 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MultiQueryOptimizer - MR plan size after optimization: 1
2024-09-03 11:04:47,449 [main] INFO org.apache.hadoop.metrics2.impl.MetricsConfig - Loaded properties from hadoop-metrics2.properties
2024-09-03 11:04:47,548 [main] INFO org.apache.hadoop.metrics2.impl.MetricsSystemImpl - Scheduled Metric snapshot period at 10 second(s).
2024-09-03 11:04:47,548 [main] INFO org.apache.hadoop.metrics2.impl.MetricsSystemImpl - JobTracker metrics system started
2024-09-03 11:04:47,577 [main] INFO org.apache.pig.tools.pigstats.mapreduce.MRScriptState - Pig script settings are added to the job
2024-09-03 11:04:47,582 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - mapred.job.reduce.markreset.buffer.percent is deprecated. Instead, use mapreduce.reduce.markreset.buffer.percent
2024-09-03 11:04:47,583 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.JobControlCompiler - mapred.job.reduce.markreset.buffer.percent is not set, set to default 0.3
2024-09-03 11:04:47,584 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - mapred.output.compress is deprecated. Instead, use mapreduce.output.fileoutputformat.compress
2024-09-03 11:04:47,595 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.JobControlCompiler - Setting up single store job
2024-09-03 11:04:47,605 [main] INFO org.apache.pig.data.SchemaTupleFrontend - Key [pig.schematuple] is false, will not generate code.
2024-09-03 11:04:47,605 [main] INFO org.apache.pig.data.SchemaTupleFrontend - Starting process to move generated code to distributed cacche
2024-09-03 11:04:47,605 [main] INFO org.apache.pig.data.SchemaTupleFrontend - Distributed cache not supported or needed in local mode. Setting key [pig.schematuple.local.dir] with code temp directory: /tmp/1725341687684-8
2024-09-03 11:04:47,638 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - 1 map-reduce job(s) waiting for submission.
2024-09-03 11:04:47,648 [JobControl] WARN org.apache.hadoop.metrics2.impl.MetricsSystemImpl - JobTracker metrics system already initialized!
2024-09-03 11:04:47,668 [JobControl] INFO org.apache.hadoop.conf.Configuration.deprecation - mapred.task.id is deprecated. Instead, use mapreduce.task.attempt.id
```

## 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';
```

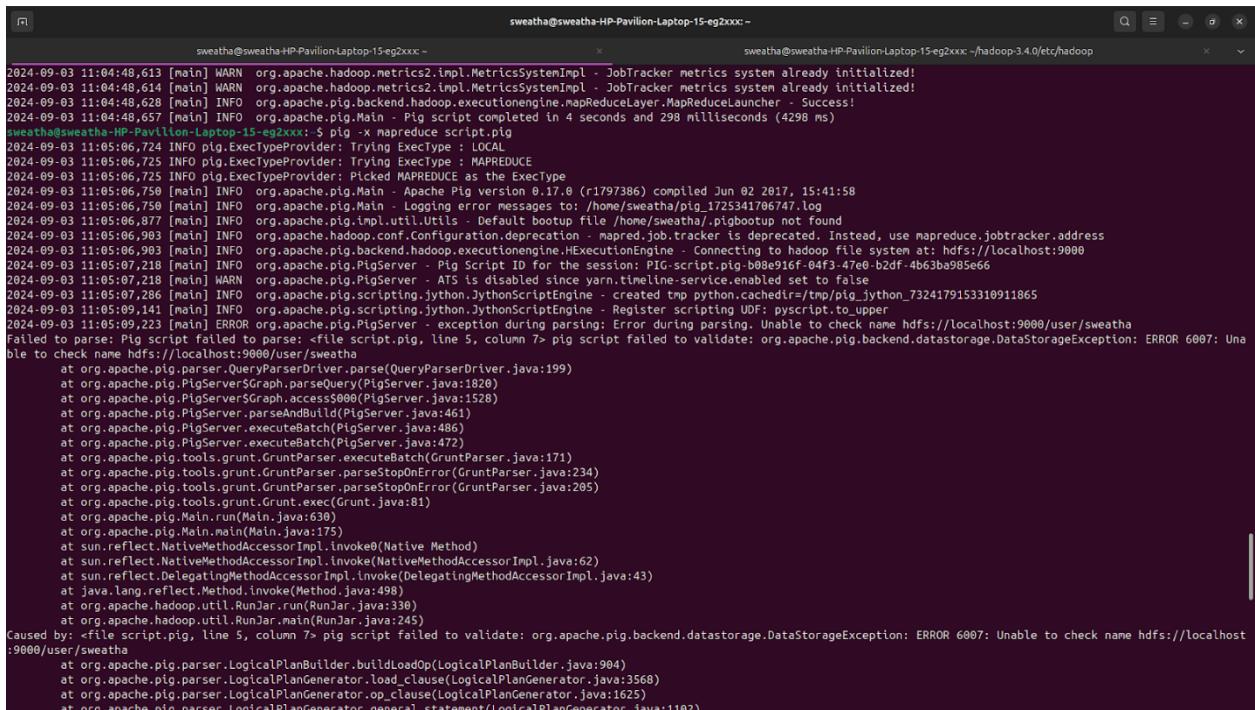
---

### 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
```



```
sweatha@sweatha-HP-Pavilion-Laptop-15-eg2xxx: ~
```

```
sweatha@sweatha-HP-Pavilion-Laptop-15-eg2xxx: ~/hadoop-3.4.0/etc/hadoop
```

```
2024-09-03 11:04:48,613 [main] WARN org.apache.hadoop.metrics2.impl.MetricsSystemImpl - JobTracker metrics system already initialized!
2024-09-03 11:04:48,614 [main] WARN org.apache.hadoop.metrics2.impl.MetricsSystemImpl - JobTracker metrics system already initialized!
2024-09-03 11:04:48,628 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - Success!
2024-09-03 11:04:48,657 [main] INFO org.apache.pig.Main - Pig script completed in 4 seconds and 298 milliseconds (4298 ms)
sweatha@sweatha-HP-Pavilion-Laptop-15-eg2xxx: $ pig -x mapreduce script.pig
2024-09-03 11:05:06,724 INFO pig.ExecTypeProvider: Trying ExecType : LOCAL
2024-09-03 11:05:06,725 INFO pig.ExecTypeProvider: Trying ExecType : MAPREDUCE
2024-09-03 11:05:06,758 INFO pig.ExecTypeProvider: Picked MAPREDUCE as the ExecType
2024-09-03 11:05:06,758 [main] INFO org.apache.pig.Main - Apache Pig version 0.17.0 (r1797386) compiled Jun 02 2017, 15:41:58
2024-09-03 11:05:06,758 [main] INFO org.apache.pig.Main - Logging error messages to: /home/sweatha/pig_1725341796747.log
2024-09-03 11:05:06,877 [main] INFO org.apache.pig.impl.util.Utils - Default bootup file /home/sweatha/.pigbootup not found
2024-09-03 11:05:06,903 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - mapred.job.tracker is deprecated. Instead, use mapreduce.jobtracker.address
2024-09-03 11:05:06,903 [main] INFO org.apache.pig.backend.hadoop.executionengine.HExecutionEngine - Connecting to hadoop file system at: hdfs://localhost:9000
2024-09-03 11:05:07,218 [main] INFO org.apache.pig.PigServer - Pig Script ID for the session: PIG-script.pig_b08e916f-04f3-47e0-b2df-4b63ba985e66
2024-09-03 11:05:07,218 [main] WARN org.apache.pig.PigServer - ATS is disabled since yarn.timeline-service.enabled set to false
2024-09-03 11:05:07,286 [main] INFO org.apache.pig.scripting.jython.JythonScriptEngine - created tmp python.cachedir=/tmp/pig_jython_7324179153310911865
2024-09-03 11:05:09,141 [main] INFO org.apache.pig.scripting.jython.JythonScriptEngine - Register scripting UDF: pyscript.to_upper
2024-09-03 11:05:09,223 [main] ERROR org.apache.pig.PigServer - exception during parsing; Error during parsing. Unable to check name hdfs://localhost:9000/user/sweatha
Failed to parse: Pig script failed to parse: file script.pig, line 5, column 7> pig script failed to validate: org.apache.pig.backend.datastorage.DataStorageException: ERROR 6007: Unable to check name hdfs://localhost:9000/user/sweatha
at org.apache.pig.parser.QueryParserDriver.parse(QueryParserDriver.java:199)
at org.apache.pig.PigServer$Graph.parseQuery(PigServer.java:1820)
at org.apache.pig.PigServer$Graph.access$000(PigServer.java:1528)
at org.apache.pig.PigServer.parseAndBuild(PigServer.java:461)
at org.apache.pig.PigServer.executeBatch(PigServer.java:486)
at org.apache.pig.PigServer.executeBatch(PigServer.java:472)
at org.apache.pig.tools.grunt.GruntParser.executeBatch(GruntParser.java:171)
at org.apache.pig.tools.grunt.GruntParser.parseStopOnError(GruntParser.java:234)
at org.apache.pig.tools.grunt.GruntParser.parseStopOnError(GruntParser.java:205)
at org.apache.pig.tools.grunt.Grunt.exec(Grunt.java:81)
at org.apache.pig.Main.main(Main.java:630)
at org.apache.pig.Main.main(Main.java:175)
at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
at java.lang.reflect.Method.invoke(Method.java:498)
at org.apache.hadoop.util.RunJar.run(RunJar.java:38)
at org.apache.hadoop.util.RunJar.main(RunJar.java:245)
Caused by: <file script.pig, line 5, column 7> pig script failed to validate: org.apache.pig.backend.datastorage.DataStorageException: ERROR 6007: Unable to check name hdfs://localhost:9000/user/sweatha
at org.apache.pig.parser.LogicalPlanBuilder.buildLoadOp(LogicalPlanBuilder.java:904)
at org.apache.pig.parser.LogicalPlanGenerator.load_clause(LogicalPlanGenerator.java:3568)
at org.apache.pig.parser.LogicalPlanGenerator.op_clause(LogicalPlanGenerator.java:1625)
at org.apache.pig.parser.LogicalPlanGenerator.general_statement(LogicalPlanGenerator.java:1102)
```

---

### 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
```

```
sweatha@sweatha-HP-Pavilion-Laptop-15-eg2xxx:~$ cat /home/sweatha/output_p/part-m-00000
HELLO WORLD
HELLO HADOOP
HADOOP IS GREAT
sweatha@sweatha-HP-Pavilion-Laptop-15-eg2xxx:~$ ~
```

**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
```

**Step 2: Place different configuration properties in Apache Hive**

In this step, we are going to do two things  
o Placing  
    Hive Home path in bashrc file

```
$ nano .bashrc
```

*And append the below lines in it*

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

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

```
$cp hive-env.sh.template hive-env.sh
```

```
$nano hive-env.sh
```

*Append the below commands on it export*

```
HADOOP_HOME=/home/Hadoop/Hadoop
```

```
export HIVE_CONF_DIR=/home/Hadoop/apache-hive-3.1.2/conf
```

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

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

**Step 3: Install mysql**

1. Install mysql in Ubuntu by running this command:

```
$sudo apt update
```

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

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

*\$sudomysql*

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

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

#### **Step 4:Config hive-site.xml**

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

*\$cd apache-hive-3.1.2-bin/bin*

*\$cp hive-default.xml.template hive-site.xml*

*\$nano hive-site.xml*

*Append these lines into it*

*Replace root as your username of MySQL*

*Replace your\_new\_password as with your password of MySQL*

*<configuration>*

*<property>*

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

*<property>*

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

*<property>*

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

*<property>*

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

*<property>*

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

*<property>*

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

```

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

</configuration>

```

### Step 5: Setup MySQL java connector:

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

[https://drive.google.com/file/d/1QFhB7Kvcat7a4LzDRe6GcmZvalyAxKz/view?usp=drive\\_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`

```

sweatha@sweatha-HP-Pavilion-Laptop-15-egxxxx: ~ $ SHIVE_HOME/bin/hiveserver2 -start
SLF4J: Found binding in [jar:file:/home/sweatha/apache-hive-4.0.0-bin/lib/log4j-slf4j-impl-2.18.0.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/sweatha/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]
HiveServer2 running as process 8206. Stop it first.
    PID TTY      STAT   TIME COMMAND
 8206 pts/6    Sl     0:18 /usr/lib/jvm/java-8-openjdk-amd64/bin/java -Dproc_jar -Djava.library.path=/home/sweatha/hadoop-3.4.0/lib/native -Dproc_hiveserver2 -Dlog4j2.formatMsgNoLook
sweatha@sweatha-HP-Pavilion-Laptop-15-egxxxx: ~ $ SHIVE_HOME/bin/hiveserver2 -start
SLF4J: Class path contains multiple slf4j bindings.
SLF4J: Found binding in [jar:file:/home/sweatha/apache-hive-4.0.0-bin/lib/log4j-slf4j-impl-2.18.0.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/sweatha/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]
SLF4J: Class path contains multiple slf4j bindings.
SLF4J: Found binding in [jar:file:/home/sweatha/apache-hive-4.0.0-bin/lib/log4j-slf4j-impl-2.18.0.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/sweatha/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]
HiveServer2 running as process 8206. Stop it first.
    PID TTY      STAT   TIME COMMAND
 8206 pts/6    Sl     0:18 /usr/lib/jvm/java-8-openjdk-amd64/bin/java -Dproc_jar -Djava.library.path=/home/sweatha/hadoop-3.4.0/lib/native -Dproc_hiveserver2 -Dlog4j2.formatMsgNoLook
sweatha@sweatha-HP-Pavilion-Laptop-15-egxxxx: ~ $ ps aux | grep HiveServer2
sweatha  9724  0.0  0.0  9280  2048 pts/7    S+  21:48  0:00 grep --color=auto HiveServer2
sweatha@sweatha-HP-Pavilion-Laptop-15-egxxxx: ~ $ kill 9206
sweatha@sweatha-HP-Pavilion-Laptop-15-egxxxx: ~ $ SHIVE_HOME/bin/hiveserver2
SLF4J: Class path contains multiple slf4j bindings.
SLF4J: Found binding in [jar:file:/home/sweatha/apache-hive-4.0.0-bin/lib/log4j-slf4j-impl-2.18.0.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/sweatha/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]
SLF4J: Class path contains multiple slf4j bindings.
SLF4J: Found binding in [jar:file:/home/sweatha/apache-hive-4.0.0-bin/lib/log4j-slf4j-impl-2.18.0.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/sweatha/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]
2024-09-19 21:49:18: Starting HiveServer2
SLF4J: Class path contains multiple slf4j bindings.
SLF4J: Found binding in [jar:file:/home/sweatha/apache-hive-4.0.0-bin/lib/log4j-slf4j-impl-2.18.0.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/sweatha/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]
Hive Session ID = 6010a53b-90da-4c57-ae42-d7de334e57c

```

### 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`

```
sweatha@sweatha-HP-Pavilion-Laptop-15-eg2xxx: ~/hadoop-3.4.0/etc/hadoop$  
sweatha@sweatha-HP-Pavilion-Laptop-15-eg2xxx: ~$  
Ended Job = job_1726814727695_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_2024-09-20_12-17-37_873_7283605356994913969-1/  
-ext-10000  
Loading data to table financials.finance_table  
MapReduce Jobs Launched:  
Stage-Stage-1: Map: 1 Reduce: 1 Cumulative CPU: 5.75 sec HDFS Read: 15693 H  
DFS Write: 291 SUCCESS  
Total MapReduce CPU Time Spent: 5 seconds 750 msec  
OK  
Time taken: 28.445 seconds  
hive> CREATE VIEW myview AS SELECT name, id FROM finance_table;  
OK  
Time taken: 0.37 seconds  
hive> SELECT*FROM myview;  
OK  
Alice 1  
Bob 2  
Charlie 3  
Time taken: 0.3 seconds, Fetched: 3 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;
```

**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:****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
```

```
sweatha@sweatha-HP-Pavilion-Laptop-15-eg2xxx: $ 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"}
]
sweatha@sweatha-HP-Pavilion-Laptop-15-eg2xxx: $ █
```

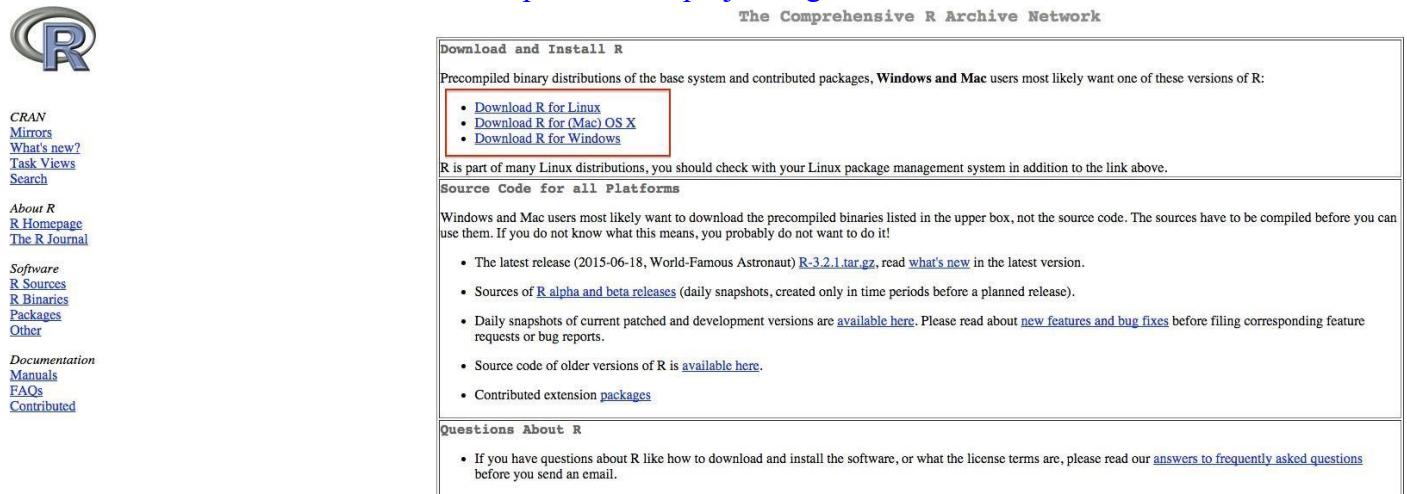
### RESULT:

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

## Installation guide for R and RStudio

### Step 1 – Install R

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



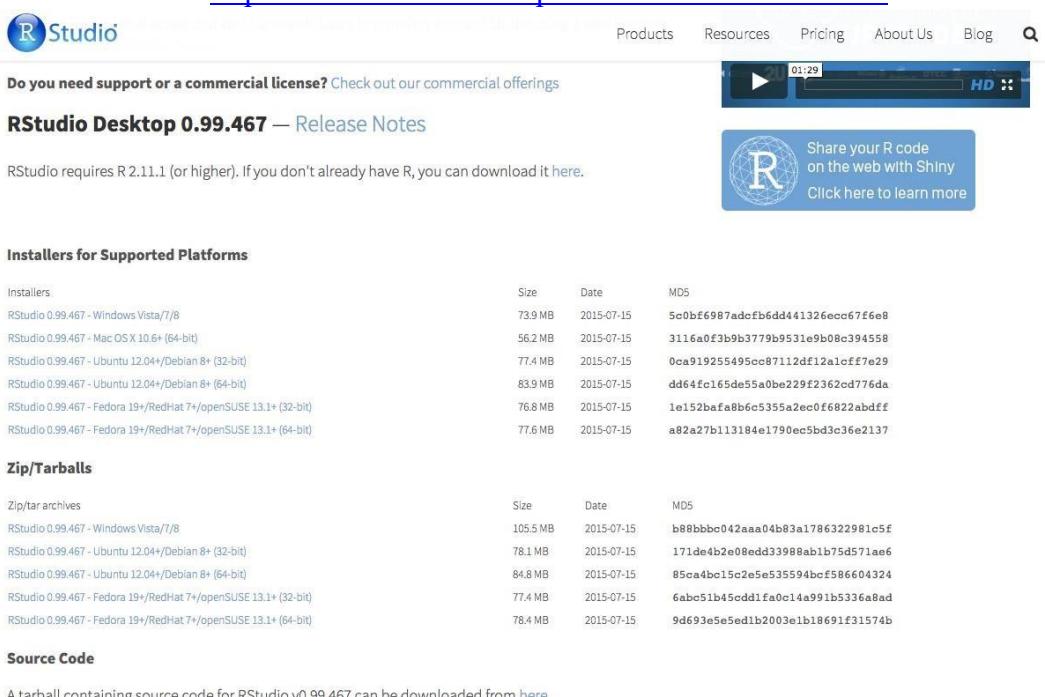
The screenshot shows the CRAN (Comprehensive R Archive Network) homepage. On the left, there's a sidebar with links for CRAN Mirrors, What's new?, Task Views, Search, About R, R Homepage, The R Journal, Software, R Sources, R Binaries, Packages, Other, Documentation, Manuals, FAQs, and Contributed. The main content area has a large 'Download and Install R' section with a box for precompiled binary distributions for Linux, Mac OS X, and Windows. Below this, a note says R is part of many Linux distributions. A 'Source Code for all Platforms' section follows, with a note that Windows and Mac users want precompiled binaries. It lists several bullet points about the latest release, alpha/beta releases, daily snapshots, and source code for older versions. At the bottom is a 'Questions About R' section with a single bullet point about sending emails.

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

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

### Step 2 – Install RStudio

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



The screenshot shows the RStudio download page. At the top, there's a navigation bar with links for Products, Resources, Pricing, About Us, Blog, and a search icon. A video player window is visible in the top right. Below the navigation, a message encourages users to check out commercial offerings. A 'RStudio Desktop 0.99.467 — Release Notes' section follows, with a note that RStudio requires R 2.11.1 or higher. To the right, there's a 'Share your R code on the web with Shiny' section with a 'Click here to learn more' button. The main content area has two tables: one for 'Installers for Supported Platforms' and another for 'Zip/Tarballs'. Both tables list file names, sizes, dates, and MD5 checksums for various operating systems. At the bottom, there's a 'Source Code' section with a note about downloading the tarball.

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

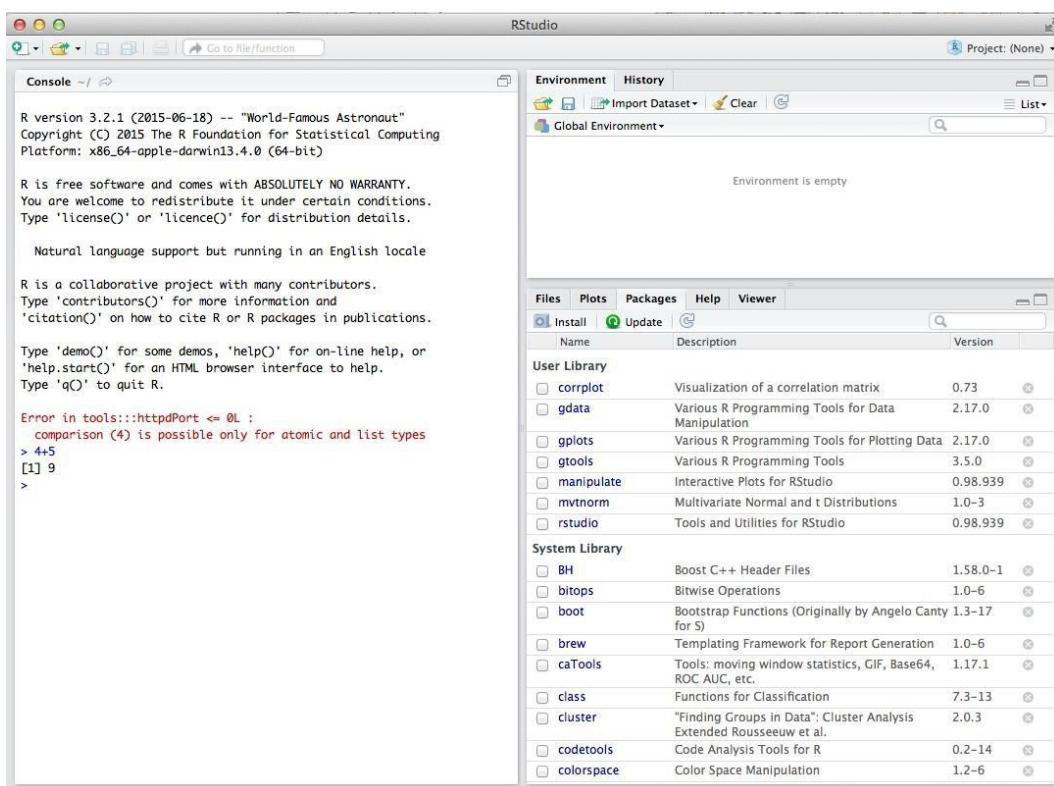


Figure 3. Running R with RStudio

### 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).

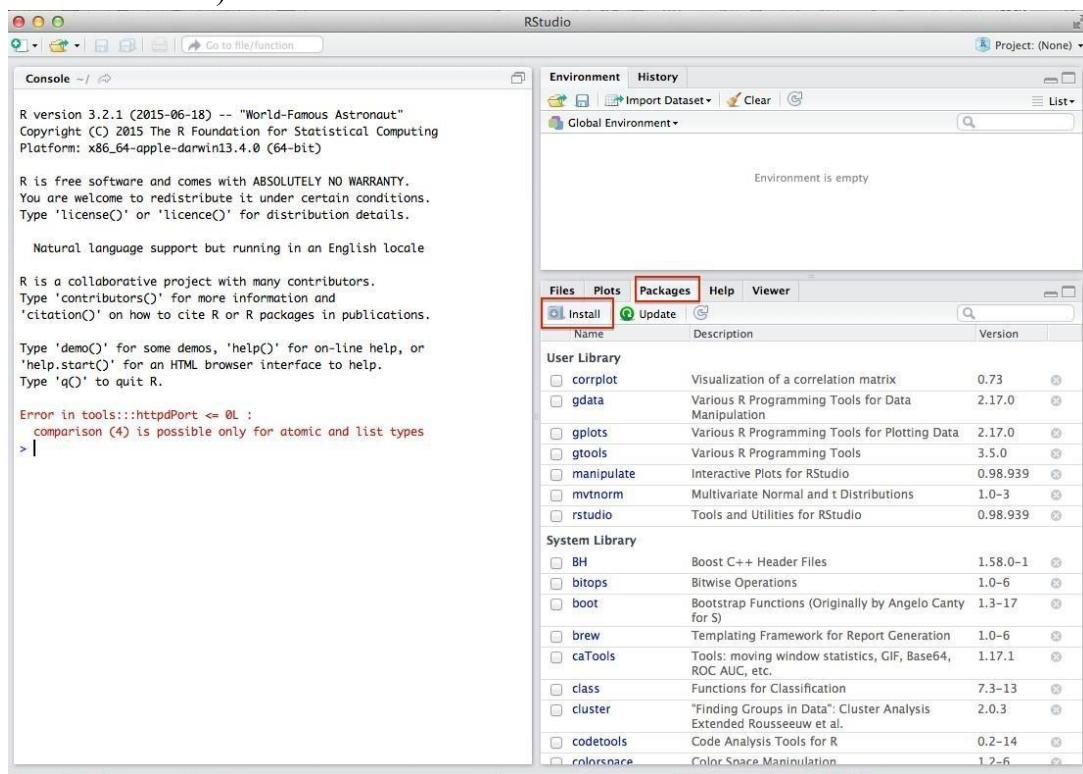


Figure 4. Click on Install to install R packages.

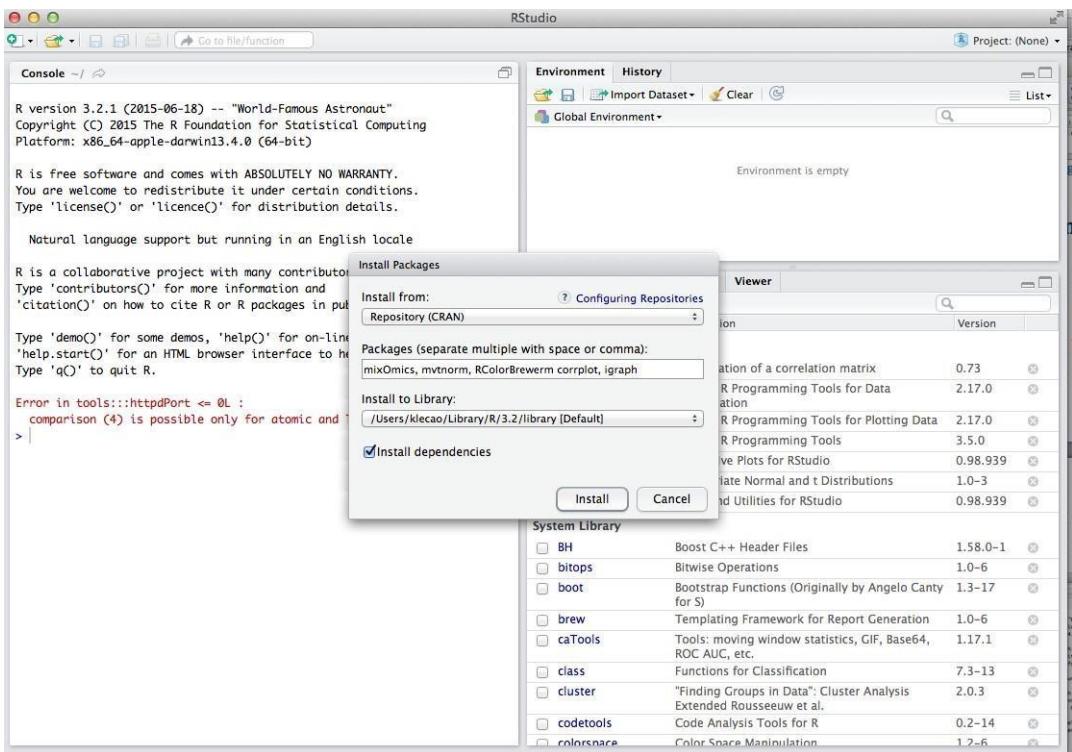


Figure 5. Specify the list of packages to be installed

The screenshot shows the RStudio interface with the following details:

- Console:**

```
> library(mixOmics)
[1] 9
<library(mixOmics)
Loading required package: MASS
Attaching package: 'MASS'
The following object is masked _by_ '.GlobalEnv':
  genotype
Loading required package: lattice
Loading required package: grid
[1] Loaded mixOmics 6.1.0
[1] ok!
```
- Environment:** Shows the global environment with objects like `data`, `data.gene`, `data.physio`, `design`, `name.gene`, `d`, `diabolo.res`, `genotype`, `k`, `kee.genes`, `keep.genes`, `keep.name.genes`, `list.data`.
- Packages:** Shows the installed packages:

Name	Description	Version
acepack	ace() and avas() for selecting regression transformations	1.3-3.3
ade4	Analysis of Ecological Data : Exploratory and Euclidean Methods in Environmental Sciences	1.7-4
ALL	A data package	1.14.0
annotate	Annotation for microarrays	1.50.0
AnnotationDbi	Annotation Database Interface	1.34.4
astsa	Applied Statistical Time Series Analysis	1.4
Biobase	Biobase: Base functions for Bioconductor	2.32.0
BiocGenerics	S4 generic functions for Bioconductor	0.18.0
BioInstaller	Install/Update Bioconductor, CRAN, and github Packages	1.22.3
BioParallel	Bioconductor facilities for parallel evaluation	1.6.2
capushe	Calibrating Penalties Using Slope Heuristics	1.1.1
car	Companion to Applied Regression	2.1-2
chron	Chronological Objects which can Handle Dates and Times	2.3-47
cisValid	Validation of Clustering Results	0.6-6

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

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

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

**PROCEDURE:**

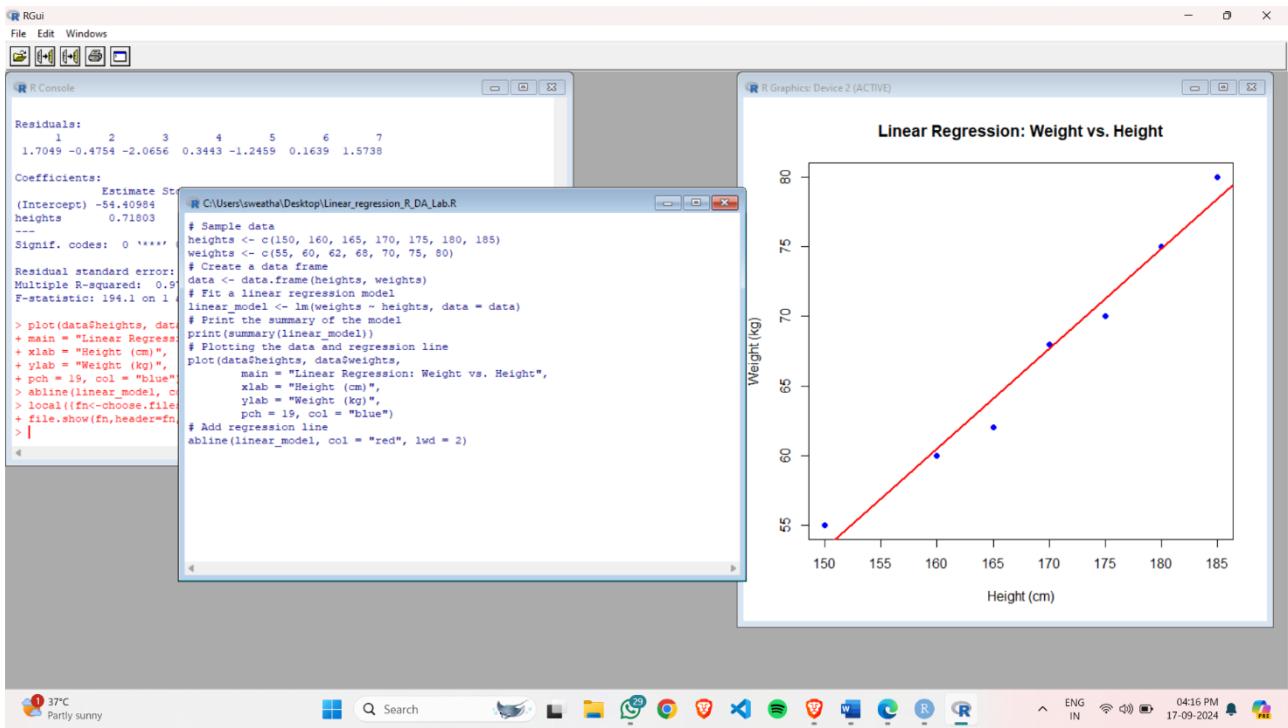
1. Create sample data for heights and weights, fit a linear regression model, and plot the data with the regression line.
2. Use the sample data to create a data frame for the regression model.
3. Fit the linear regression model using the `lm()` function and display the summary.
4. Plot the data points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

**PROGRAM CODE:****a) Linear regression**

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

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

## **OUTPUT:**



### **b) Logistic regression**

```
# Logistic Regression  
  
data(mtcars)  
  
mtcars$am <- factor(mtcars$am, levels = c(0, 1), labels = c("Automatic", "Manual"))  
  
logistic_model <- glm(am ~ mpg, data = mtcars, family = binomial)  
  
print(summary(logistic_model))
```

```
# Plotting Logistic Regression

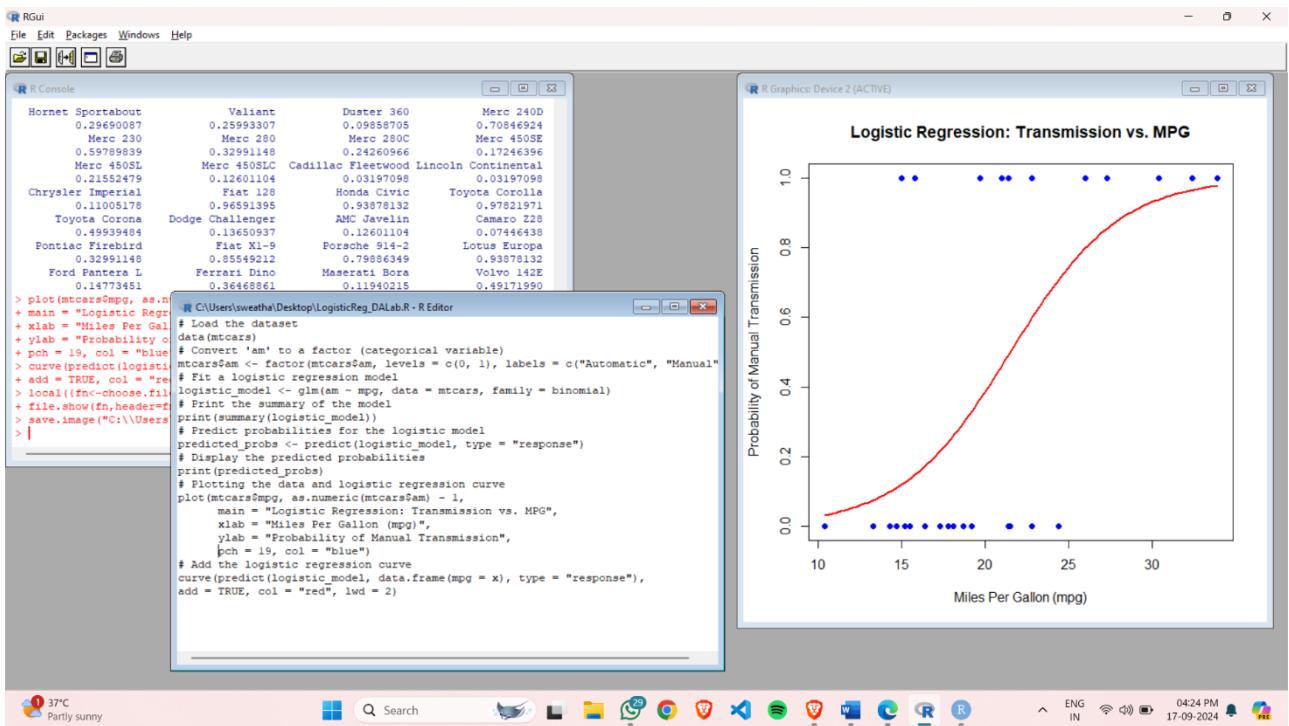
predicted_probs <- predict(logistic_model, type = "response")

print(predicted_probs)

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

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

## OUTPUT:



## RESULT:

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

**Exp.No: 8**

## **IMPLEMENT SVM/DECISION TREE CLASSIFICATION TECHNIQUES**

**AIM:**

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

**PROCEDURE:**

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

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

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

# Load the iris dataset
data(iris)

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

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

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

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

```

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

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

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

```

## OUTPUT:

The screenshot shows the RStudio interface with two main windows open:

- R Console:** Displays the command-line interface where the SVM code was run. It shows the number of support vectors (45), the number of classes (3), and the levels (setosa, versicolor, virginica). The confusion matrix and accuracy are also printed.
- R Editor:** Displays the R script used to build the SVM model. The script includes comments explaining the steps: installing e1071, loading iris dataset, splitting it into training (70%) and testing (30%) sets, fitting an SVM model with radial kernel, printing the summary, predicting the test set, and finally calculating and printing the accuracy.

## 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:

The screenshot shows the RStudio interface with two windows open. The left window is the R Console, displaying the R code used to build and evaluate a decision tree on the Iris dataset. The right window is the R Graphics Device 2 (ACTIVE), showing a decision tree diagram. The tree has a root node labeled 'Petal Length < 2.45'. This node branches into two: one leading to 'setosa' and another leading to a node labeled 'Petal Width < 1.75'. This second node further branches into 'versicolor' and 'virginica'.

```

RGui
File History Resize Windows
R R Console
R Graphics Device 2 (ACTIVE)

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

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

> plot(tree_model)
> text(tree_model, pretty = 0)
> predictions <- predict(tree_model, newdata = test_data, type = "class")
> confusion_matrix <- table(Predicted = predictions, Actual = test_data$Species)
> print(confusion_matrix)
      Actual
Predicted   setosa versicolor virginica
  setosa       14          0         0
  versicolor      0          0         0
  virginica       0          0         0
> accuracy <- sum(diag(confusion_matrix))
> cat("Accuracy:", accuracy * 100, "%\n")
Accuracy: 97.77778 %

```

## RESULT:

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

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

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

**PROCEDURE:**

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

**PROGRAM CODE:****a) HIERARCHIAL CLUSTERING**

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

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

# Standardize the data
iris_scaled <- scale(iris_data)

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

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

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

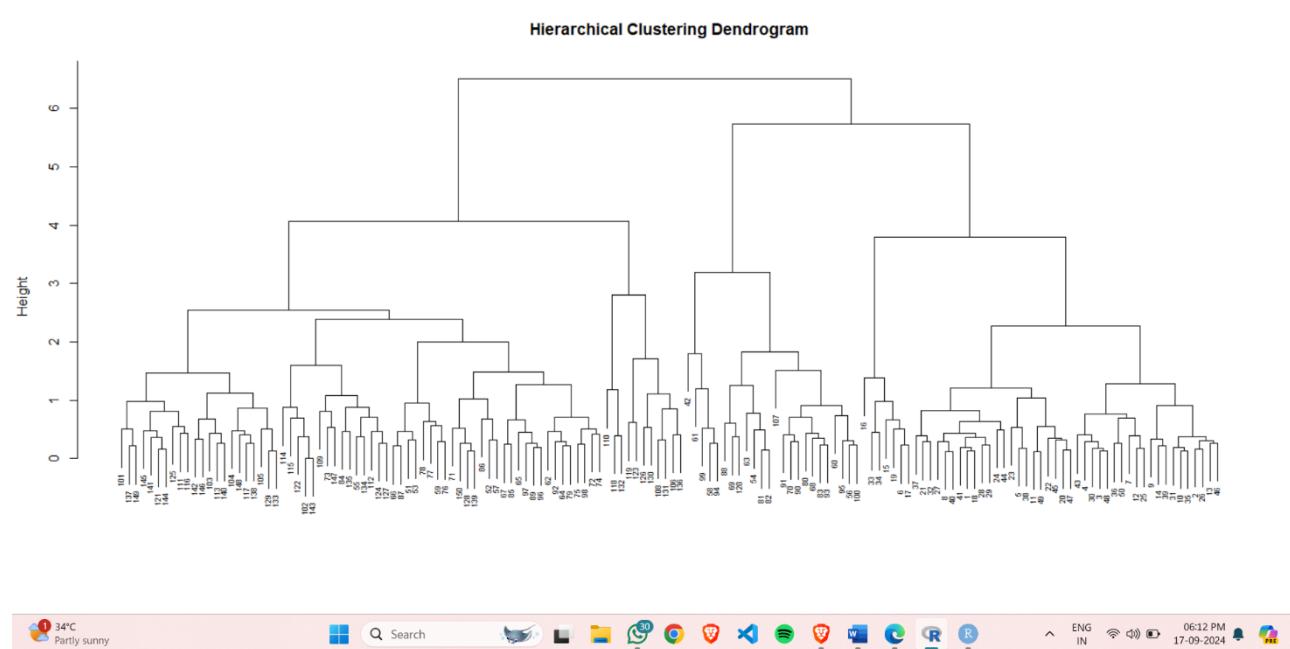
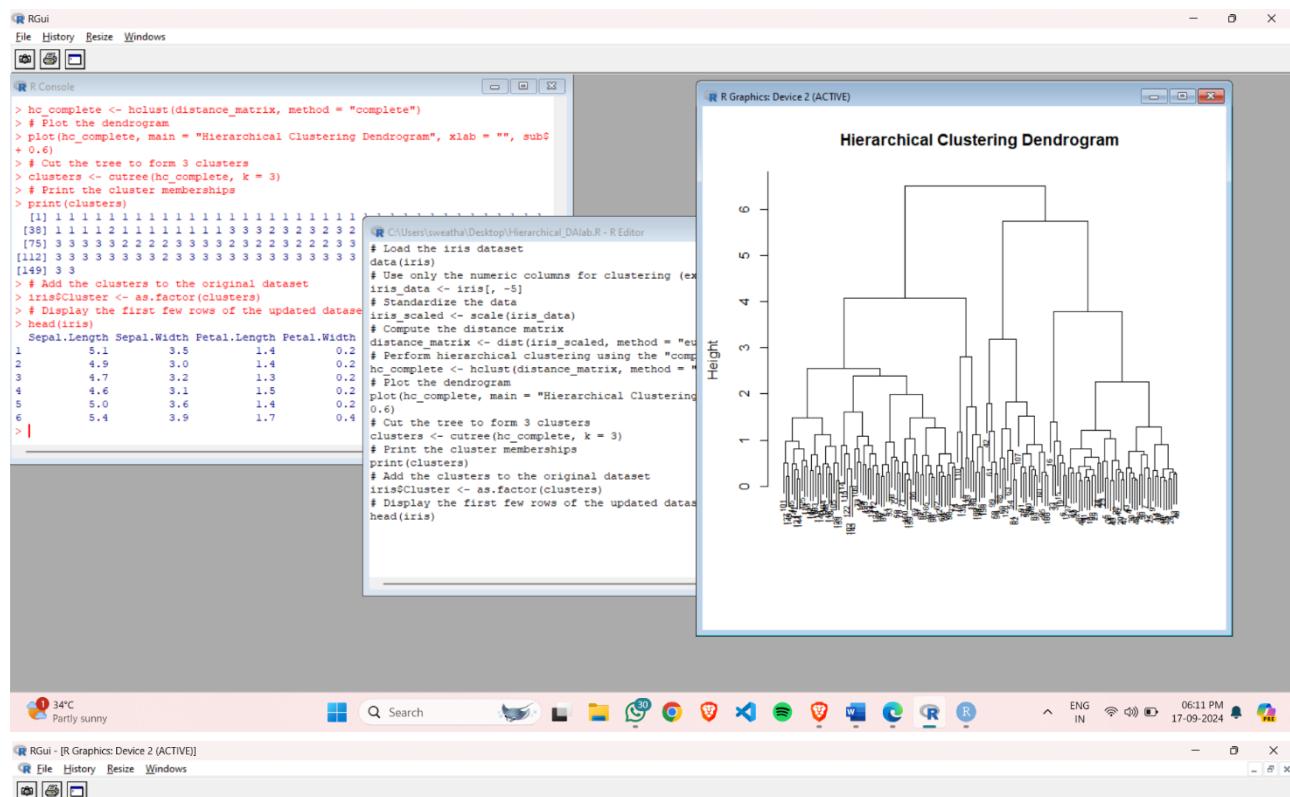
# Cut the tree to form 3 clusters
clusters <- cutree(hc_complete, k = 3)

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

```
# Add the clusters to the original dataset
iris$Cluster <- as.factor(clusters)

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

## OUTPUT:



**b) K-MEANS CLUSTERING**

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

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

# Standardize the data
iris_scaled <- scale(iris_data)

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

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

# Print the K-Means result
print(kmeans_result)

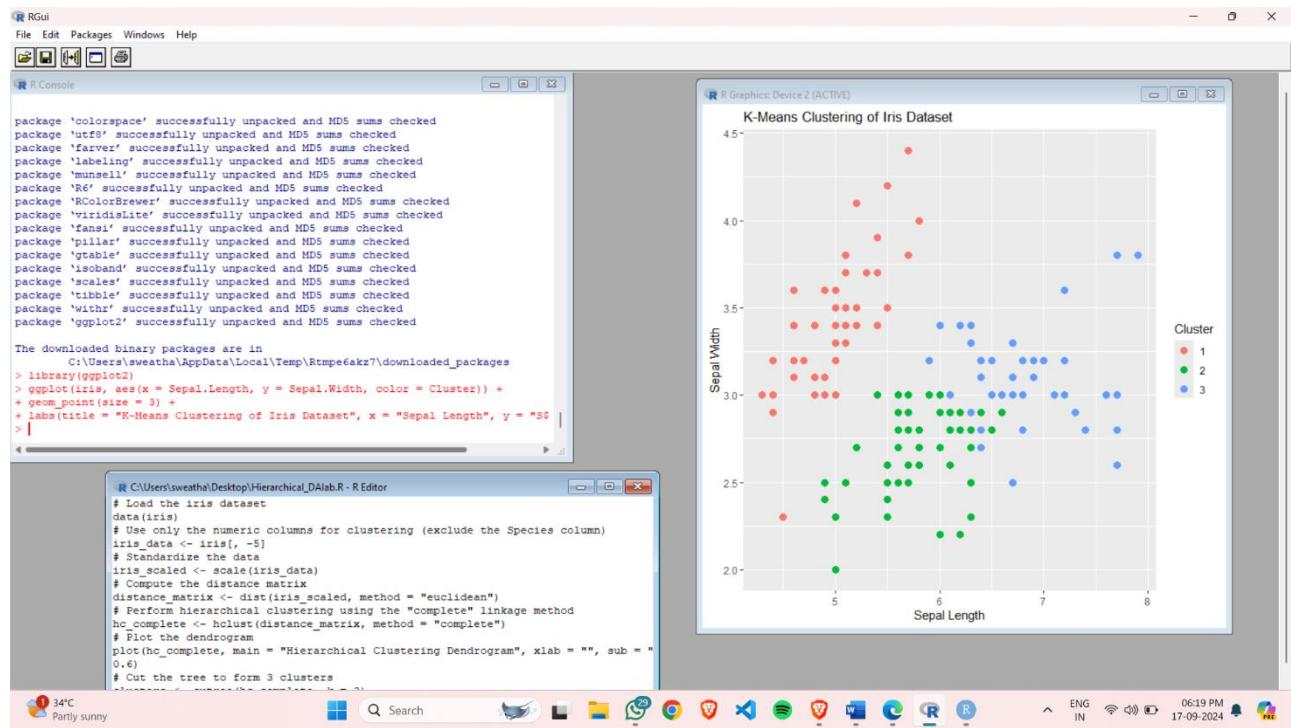
# Print the cluster centers
print(kmeans_result$centers)

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

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

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

## OUTPUT:



## RESULT:

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

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

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

**PROCEDURE:**

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

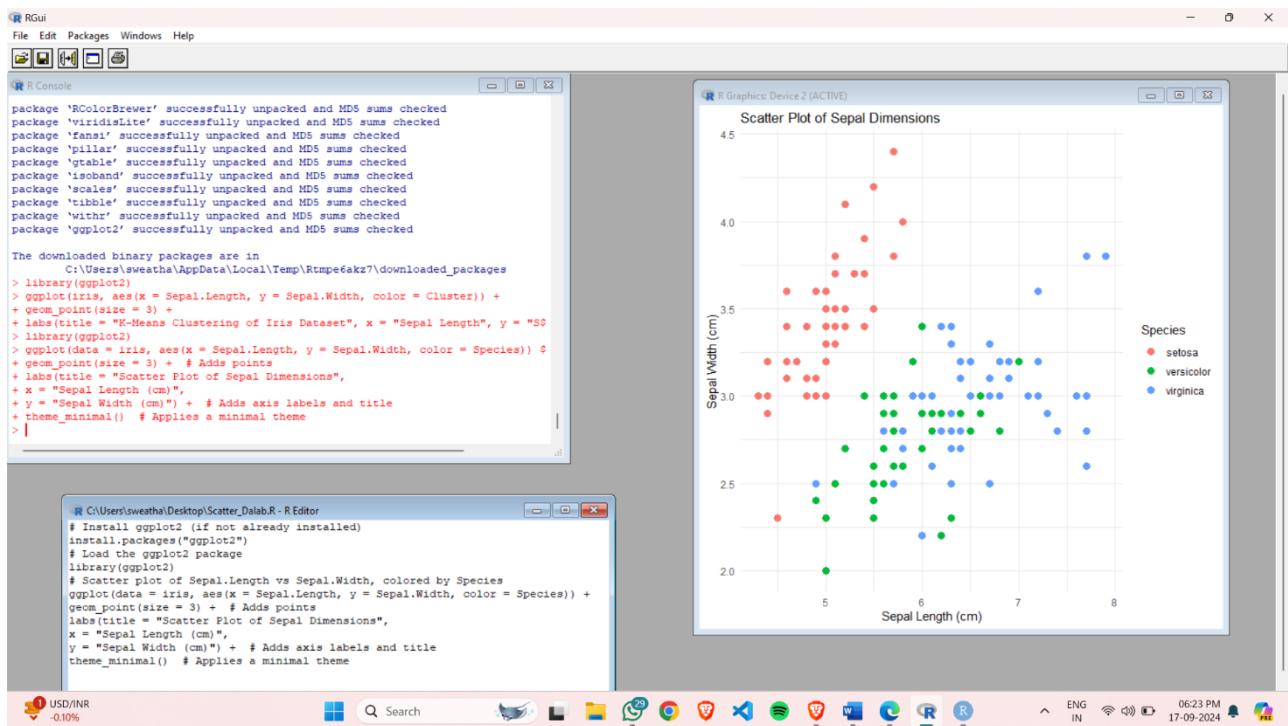
**1) SCATTER PLOT**

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

# Load the ggplot2 package
library(ggplot2)

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

## OUTPUT:

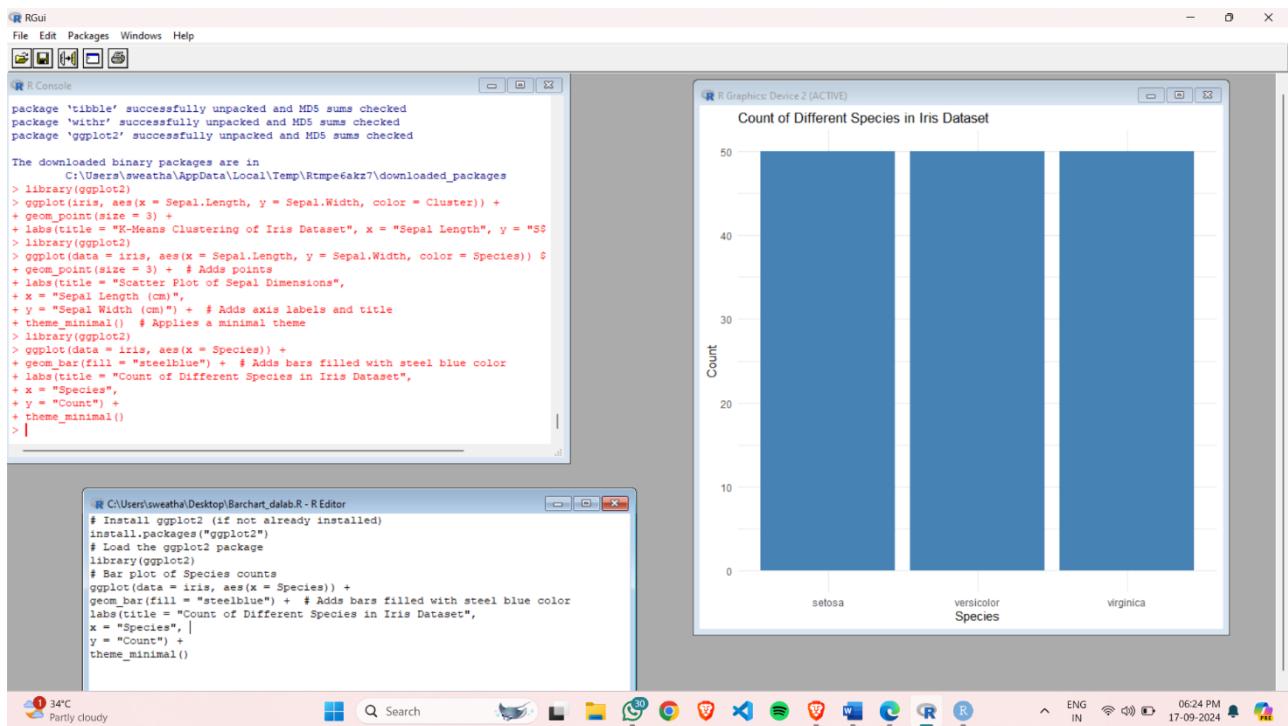


## 2) BAR CHART

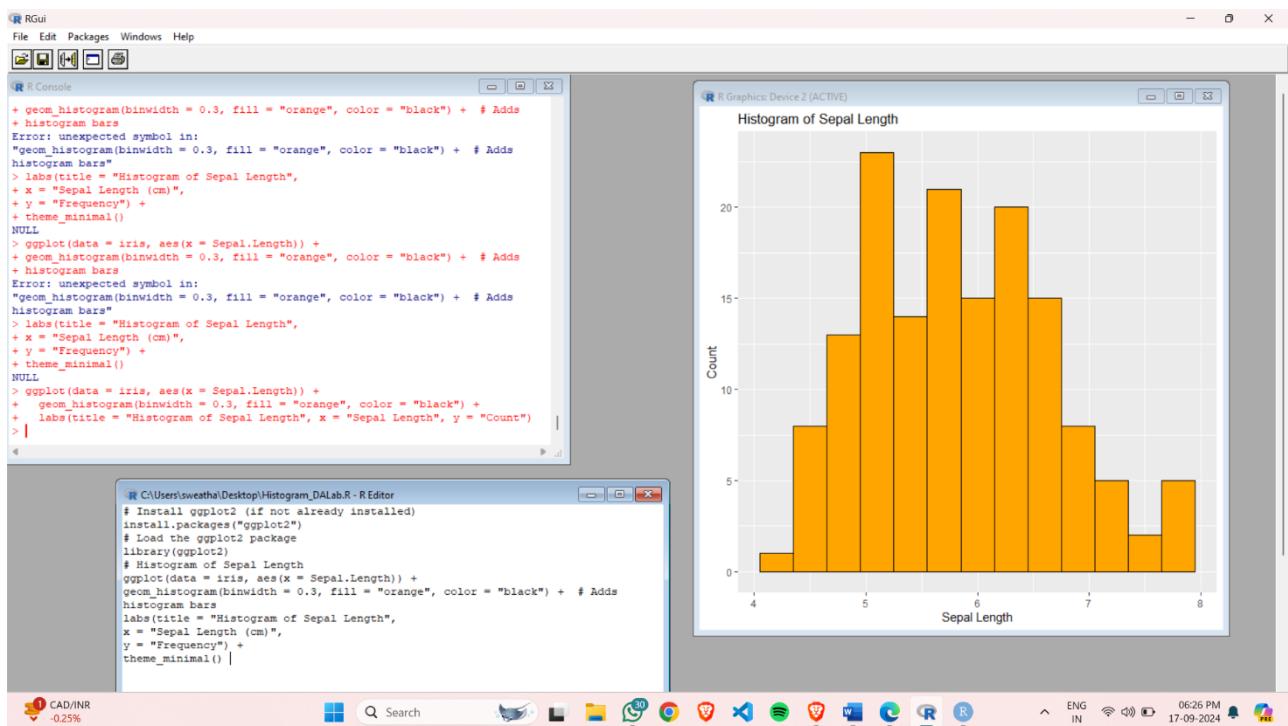
```
# Install ggplot2 (if not already installed)
install.packages("ggplot2")

# Load the ggplot2 package
library(ggplot2)

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

**OUTPUT:****3) HISTOGRAM**

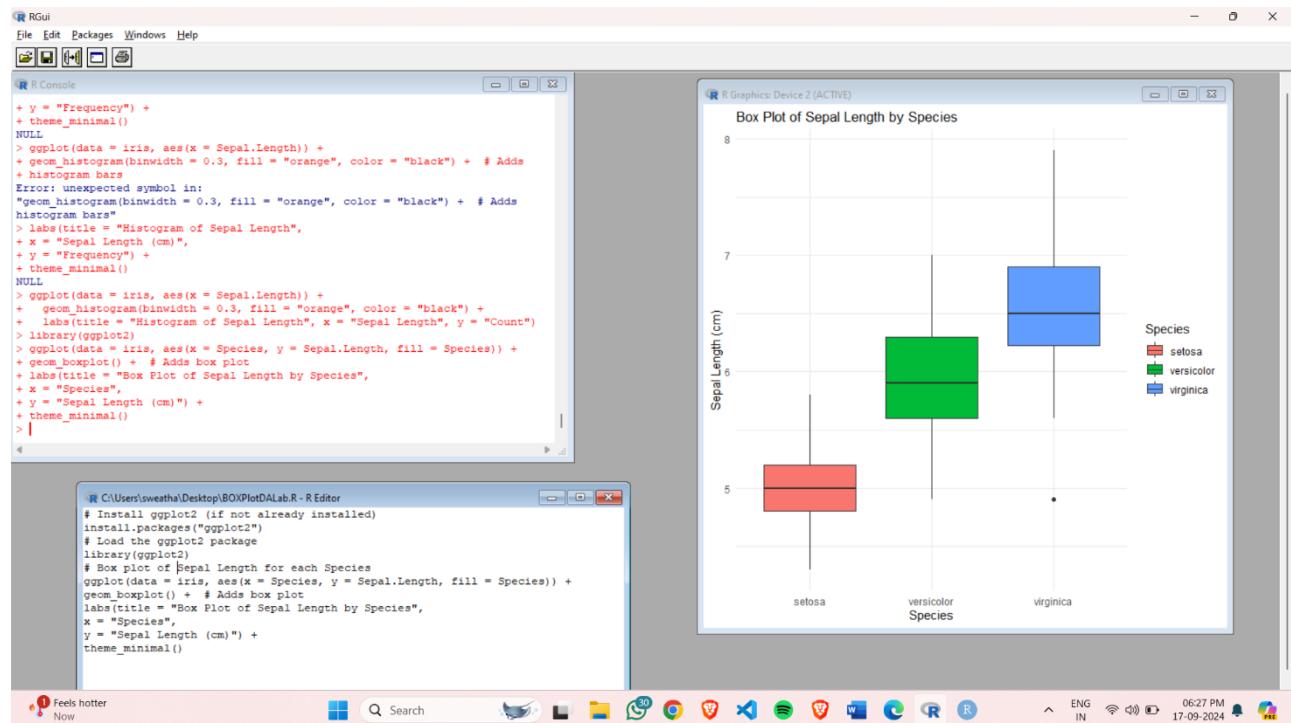
```
# Histogram of Sepal Length
ggplot(data = iris, aes(x = Sepal.Length)) +
  geom_histogram(binwidth = 0.3, fill = "orange", color = "black") + # Adds histogram bars
  labs(title = "Histogram of Sepal Length",
       x = "Sepal Length (cm)",
       y = "Frequency") +
  theme_minimal()
```

**OUTPUT:**

#### 4)BOX PLOT

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

#### OUTPUT:



#### RESULT:

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