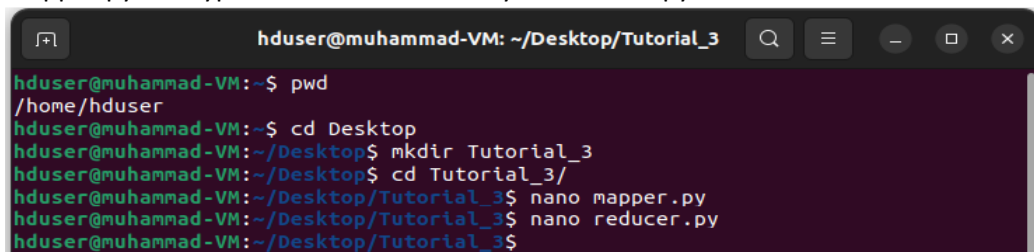


Tutorial 3

Hadoop Streaming using MapReduce Model

To demonstrate how the Hadoop streaming utility can run Python code as a MapReduce application on the Hadoop cluster, the **WordCount** application can be implemented as two Python programs: (**mapper.py** and **reducer.py**). **mapper.py** is the Python code that implements the logic in the map phase of **WordCount**. It reads data from **stdin**, splits the lines into words, and outputs each word with its intermediate count to **stdout**.

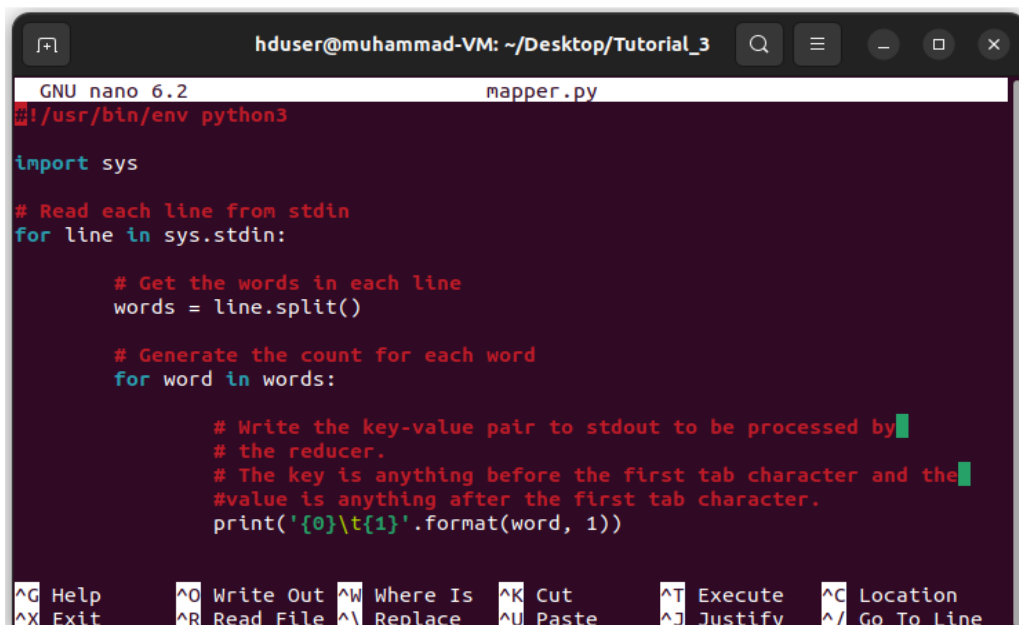
Create a folder **Tutorial_3** on Ubuntu Desktop or any other folder name of your choice and write the Python code in the following files named as **mapper.py** and **reducer.py**. The below screenshot showed the commands to create the folder (**Tutorial_3**) on Ubuntu Desktop. Then create the file **mapper.py** and type the code and similarly for **reducer.py**.



```
hduser@muhammad-VM: ~/Desktop/Tutorial_3
hduser@muhammad-VM:~$ pwd
/home/hduser
hduser@muhammad-VM:~$ cd Desktop
hduser@muhammad-VM:~/Desktop$ mkdir Tutorial_3
hduser@muhammad-VM:~/Desktop$ cd Tutorial_3/
hduser@muhammad-VM:~/Desktop/Tutorial_3$ nano mapper.py
hduser@muhammad-VM:~/Desktop/Tutorial_3$ nano reducer.py
hduser@muhammad-VM:~/Desktop/Tutorial_3$
```

Part I: WordCount Frequency

1) Type the following Python code in the files in the Ubuntu VM using **nano/ gedit** editor as mentioned in the screenshot.



```
GNU nano 6.2 mapper.py
#!/usr/bin/env python3

import sys

# Read each line from stdin
for line in sys.stdin:

    # Get the words in each line
    words = line.split()

    # Generate the count for each word
    for word in words:

        # Write the key-value pair to stdout to be processed by
        # the reducer.
        # The key is anything before the first tab character and the
        # value is anything after the first tab character.
        print('{0}\t{1}'.format(word, 1))

^G Help      ^O Write Out ^W Where Is  ^K Cut       ^T Execute  ^C Location
^X Exit      ^R Read File ^_ Replace   ^U Paste     ^J Justify   ^_ Go To Line
```

symbol represents the comments in the Python.

reducer.py

```

GNU nano 6.2 reducer.py *
#!/usr/bin/env python3

import sys

curr_word = None
curr_count = 0

# Process each key-value pair from the mapper
for line in sys.stdin:

    # Get the key and value from the current line
    word, count = line.split('\t')

    # Convert the count to an int
    count = int(count)

    # If the current word is the same as the previous word,
    # increment its count, otherwise print the words count to stdout
    if word == curr_word:
        curr_count += count
    else:
        # Write word and its number of occurrences as a key-value
        # pair to stdout
        if curr_word:
            print('{0}\t{1}'.format(curr_word, curr_count))

        curr_word = word
        curr_count = count

# Output the count for the last word
if curr_word == word:
    print('{0}\t{1}'.format(curr_word, curr_count))

```

2) Before hadoop streaming, we provide the executable privileges to **mapper.py** and **reducer.py** files by using the command. (Check Tutorial 1 for further details).

```
$chmod 700 mapper.py
```

```
$chmod 700 reducer.py
```

```

hduser@muhammad-VM: ~/Downloads/Tutorial_3
hduser@muhammad-VM:~/Downloads$ cd Tutorial_3/
hduser@muhammad-VM:~/Downloads/Tutorial_3$ ls -l
total 144
-rw----- 1 hduser hadoopgroup 135287 Feb 25 2022 britney-spears.txt
-rw----- 1 hduser hadoopgroup    426 Feb 25 2022 mapper.py
-rw----- 1 hduser hadoopgroup    744 Feb 25 2022 reducer.py
hduser@muhammad-VM:~/Downloads/Tutorial_3$ chmod 700 mapper.py
hduser@muhammad-VM:~/Downloads/Tutorial_3$ chmod 700 reducer.py
hduser@muhammad-VM:~/Downloads/Tutorial_3$ ls -l
total 144
-rw----- 1 hduser hadoopgroup 135287 Feb 25 2022 britney-spears.txt
-rwx----- 1 hduser hadoopgroup    426 Feb 25 2022 mapper.py
-rwx----- 1 hduser hadoopgroup    744 Feb 25 2022 reducer.py
hduser@muhammad-VM:~/Downloads/Tutorial_3$

```

Check that the python code is working correctly or not. Use the following Linux command on Ubuntu terminal/ shell as mentioned below

```

hduser@muhammad-VM:~/Downloads/Tutorial_3$ echo 'jack be nimble jack be quick' | ./mapper.py | sort -t 1 | ./reducer.py
be      2
jack    2
nimble  1
quick   1
hduser@muhammad-VM:~/Downloads/Tutorial_3$

```

This shows that the syntax of the python code is correct. If you are facing difficulty in writing the Python code, you can download the files from Moodle and copy them into Tutorial_3 folder.

3) The MapReduce framework's streaming functionality is one of the properties that can be configured in Hadoop using the **mapred-site.xml** file. Without writing a Java code, Hadoop Streaming is a tool that lets you design and execute MapReduce tasks using arbitrary executables as the mapper and/or reducer. Before the execution of Hadoop streaming, you must update **mapred-site.xml** file in the folder at the location **/usr/local/hadoop/etc/hadoop/**

Comment the already written property in the **mapred.xml** file and write the below mentioned code. The file after update looks like as mentioned below

nano /usr/local/hadoop/etc/hadoop/mapred-site.xml

```

GNU nano 4.8 /usr/local/hadoop/etc/hadoop/mapred-site.xml
-->

<!-- Put site-specific property overrides in this file. -->

<configuration>
<!--
<property>
  <name>mapreduce.framework.name</name>
  <value>yarn</value>
</property>
-->

<property>
  <name>yarn.app.mapreduce.am.env</name>
  <value>HADOOP_MAPRED_HOME=${HADOOP_HOME}</value>
</property>
<property>
  <name>mapreduce.map.env</name>
  <value>HADOOP_MAPRED_HOME=${HADOOP_HOME}</value>
</property>
<property>
  <name>mapreduce.reduce.env</name>
  <value>HADOOP_MAPRED_HOME=${HADOOP_HOME}</value>
</property>
</configuration>

^G Get Help  ^O Write Out  ^W Where Is  ^K Cut Text  ^J Justify   ^C Cur Pos   M-U Undo
^X Exit      ^R Read File  ^\ Replace   ^U Paste Text ^T To Spell  ^_ Go To Line M-E Redo

```

Comment the previous property in the **mapred-site.xml** file and append as the three new properties for Hadoop environment as shown in the above screenshot. These are essential for the execution of Hadoop streaming jobs.

4) Download a text input file from the Moodle named as **britney-spears.txt** (Song lyrics). This file will store in the Downloads folder in Ubuntu VM automatically. Move this text file from your Downloads folder into **hdfs** for Map Reduce streaming jobs. The steps to move the file from Downloads folder to the "user1" folder on **hdfs** is shown below

```

hduser@muhammad-VM: ~/Downloads/Tutorial_3
hduser@muhammad-VM:~/Downloads/Tutorial_3$ jps
7443 NameNode
8071 NodeManager
7594 DataNode
7947 ResourceManager
9579 Jps
7775 SecondaryNameNode
hduser@muhammad-VM:~/Downloads/Tutorial_3$ hadoop fs -ls /
hduser@muhammad-VM:~/Downloads/Tutorial_3$ hadoop fs -mkdir /user1
hduser@muhammad-VM:~/Downloads/Tutorial_3$ hadoop fs -ls /
Found 1 items
drwxr-xr-x  - hduser supergroup          0 2024-01-30 23:28 /user1
hduser@muhammad-VM:~/Downloads/Tutorial_3$ ls
britney-spears.txt  mapper.py  reducer.py
hduser@muhammad-VM:~/Downloads/Tutorial_3$ hadoop fs -put ./britney-spears.txt /user1
hduser@muhammad-VM:~/Downloads/Tutorial_3$ hadoop fs -ls /user1
Found 1 items
-rw-r--r--  1 hduser supergroup    135287 2024-01-30 23:28 /user1/britney-spears.txt
hduser@muhammad-VM:~/Downloads/Tutorial_3$

```

\$cd Downloads

```
$hadoop fs -put ./britney-spears.txt /user1
```

```
$hadoop fs -ls /user1
```

If the **user1** is not present on Hadoop distributed file system (hdfs), create a folder **user1** on the Hadoop distributed file system using the command.

```
$hadoop fs -mkdir /user1
```

5) Now the input file (**britney-spears.txt**) is ready for the Hadoop streaming, type the following command as mentioned below highlighted with an arrow

```

huser@muhammad-VM: ~/Desktop/Tutorial_3
huser@muhammad-VM:~/Desktop/Tutorial_3$ jps
5107 SecondaryNameNode
5507 NodeManager
6133 Jps
4648 NameNode
5371 ResourceManager
4875 DataNode
huser@muhammad-VM:~/Desktop/Tutorial_3$ hadoop fs -ls /user1
Found 1 items
-rw-r--r-- 1 huser supergroup 135287 2024-08-20 21:59 /user1/britney-spears.txt
huser@muhammad-VM:~/Desktop/Tutorial_3$ hadoop jar $HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-3.4.0.jar
-mapper ./mapper.py -reducer ./reducer.py -input /user1/britney-spears.txt -output /output1
2024-08-20 22:04:19,031 INFO impl.MetricsConfig: Loaded properties from hadoop-metrics2.properties
2024-08-20 22:04:19,096 INFO impl.MetricsSystemImpl: Scheduled Metric snapshot period at 10 second(s).
2024-08-20 22:04:19,096 INFO impl.MetricsSystemImpl: JobTracker metrics system started
2024-08-20 22:04:19,109 WARN impl.MetricsSystemImpl: JobTracker metrics system already initialized!
2024-08-20 22:04:19,265 INFO mapred.FileInputFormat: Total input files to process : 1
2024-08-20 22:04:19,328 INFO mapreduce.JobSubmitter: number of splits:1
2024-08-20 22:04:19,446 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_local934454541_0001
2024-08-20 22:04:19,447 INFO mapreduce.JobSubmitter: Executing with tokens: []
2024-08-20 22:04:19,565 INFO mapreduce.Job: The url to track the job: http://localhost:8080/
2024-08-20 22:04:19,572 INFO mapred.LocalJobRunner: OutputCommitter set in config null
2024-08-20 22:04:19,575 INFO mapred.LocalJobRunner: OutputCommitter is org.apache.hadoop.mapred.FileOutputCommitter
2024-08-20 22:04:19,585 INFO mapreduce.Job: Running job: job_local934454541_0001
2024-08-20 22:04:19,587 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 2
2024-08-20 22:04:19,587 INFO output.FileOutputCommitter: FileOutputCommitter skip cleanup _temporary folders under

```

6) The separate command for Hadoop streaming from the above screenshot is mentioned below

```
huser@muhammad-VM:~/Desktop/Tutorial_3$ hadoop jar $HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-3.4.0.jar
-mapper ./mapper.py -reducer ./reducer.py -input /user1/britney-spears.txt -output /output1
```

Note: The last part of the message will be displayed after the successful execution of Hadoop streaming job as mentioned below

7) The output of the Hadoop streaming job will be stored in the output folder (named as **“output1”** in the above command) on the Hadoop cluster. It is not possible to reuse or rewrite into the output1 folder name on hdfs. It is necessary for either creating a new folder name or delete the existing one and start over.

You can display the contents of the files by using the following commands as mentioned below using a dotted-dashed arrow as shown in the screenshot.

```

hduser@muhammad-VM: ~/Desktop/Tutorial_3
reduce input records=27731
Reduce output records=2743
Spilled Records=55862
Shuffled Maps =1
Failed Shuffles=0
Merged Map outputs=1
GC time elapsed (ms)=11
Total committed heap usage (bytes)=541065216

Shuffle Errors
BAD_ID=0
CONNECTION=0
IO_ERROR=0
WRONG_LENGTH=0
WRONG_MAP=0
WRONG_REDUCE=0

File Input Format Counters
Bytes Read=135287
File Output Format Counters
Bytes Written=24095
2024-08-20 22:04:21,605 INFO streaming.StreamJob: Output directory: /output1
hduser@muhammad-VM:~/Desktop/Tutorial_3$ hadoop fs -ls /output1
Found 2 items
-rw-r--r-- 1 hduser supergroup          0 2024-08-20 22:04 /output1/_SUCCESS
-rw-r--r-- 1 hduser supergroup    24095 2024-08-20 22:04 /output1/part-00000
hduser@muhammad-VM:~/Desktop/Tutorial_3$ hadoop fs -cat /output1/part-00000
"Because      1
"But          1
"Can          2
"Ladies       1
"hey,         1
&             1
'Cause       35
'Cause,       4
'Most         1
'Till         3
'You          1
'bout        13
'cause       20
'cross        1
'em          22

```

The wordcount example using Hadoop streaming is successfully completed as clearly illustrated in the output as clearly shown in the screenshot.

You have finished a successful execution of Hadoop streaming job. Now you can consider a large data input file and count the frequency of various words.

- If you would like to see the details of Hadoop cluster, you can explore further details of hadoop cluster by using **localhost:9870** on the web browser (mozilla firefox or google chrome on Ubuntu VM). Explore different options of hadoop in the browser.

8) If you would like download the processed data from the hdfs drive to local drive for your continuous assessment. You can use the following commands as mentioned below

```

$cd /home/hduser/Desktop/Tutorial_3/
$hadoop fs -ls /output1

```

This command will copy the folder from hdfs to your local Tutorial_3 folder on Ubuntu VM.

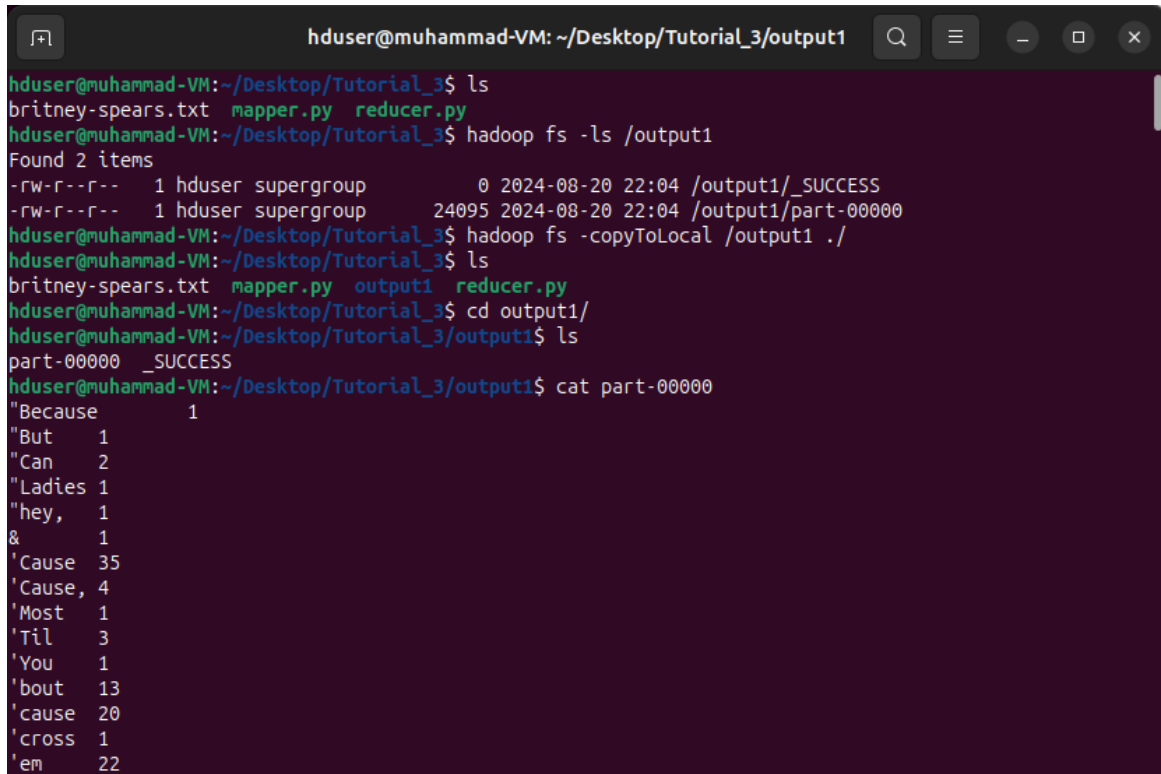
```

$hadoop fs -copyToLocal /output1 ./
$ls
$cd output1

```

```
$cat part-00000
```

The screenshot for the above-mentioned commands is shown below.



```

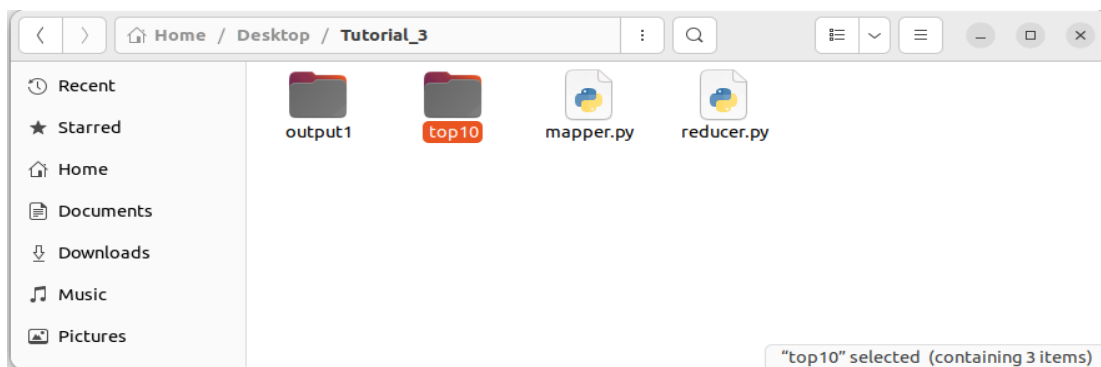
hduser@muhammad-VM: ~/Desktop/Tutorial_3/output1
hduser@muhammad-VM:~/Desktop/Tutorial_3$ ls
britney-spears.txt mapper.py reducer.py
hduser@muhammad-VM:~/Desktop/Tutorial_3$ hadoop fs -ls /output1
Found 2 items
-rw-r--r-- 1 hduser supergroup          0 2024-08-20 22:04 /output1/_SUCCESS
-rw-r--r-- 1 hduser supergroup    24095 2024-08-20 22:04 /output1/part-00000
hduser@muhammad-VM:~/Desktop/Tutorial_3$ hadoop fs -copyToLocal /output1 ./
hduser@muhammad-VM:~/Desktop/Tutorial_3$ ls
britney-spears.txt mapper.py output1 reducer.py
hduser@muhammad-VM:~/Desktop/Tutorial_3$ cd output1/
hduser@muhammad-VM:~/Desktop/Tutorial_3/output1$ ls
part-00000 _SUCCESS
hduser@muhammad-VM:~/Desktop/Tutorial_3/output1$ cat part-00000
"Because      1
"But          1
"Can          2
"Ladies       1
"hey,         1
&             1
'Cause        35
'Cause,       4
'Most         1
'Til          3
'You          1
'bout         13
'cause        20
'cross        1
'em           22

```

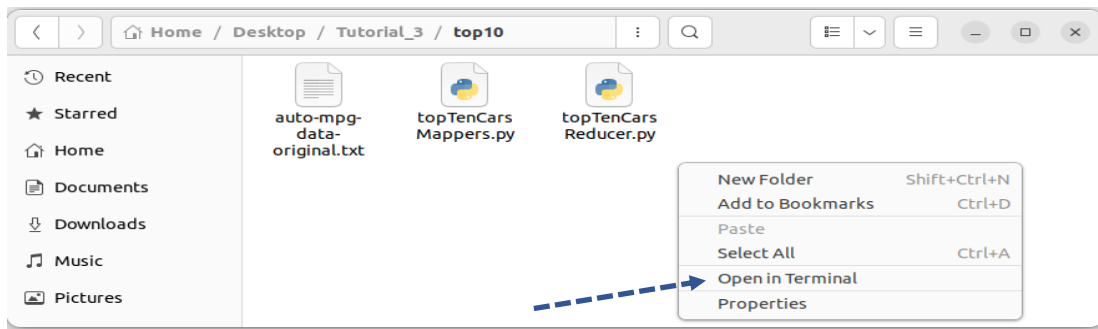
Now create another folder in Tutorial_3 and named as “**top10**”.

Part II: Top10 Design Pattern

1) Download the folder **top10** MapReduce Design pattern from the Moodle. Unzip the folder and move the folder to **Tutorial_3** on Ubuntu Desktop as shown below.



When you open **top10** folder inside **Tutorial_3**, three files (Mapper, Reducer Python files and text file) are present as shown below



Right click inside the folder and open the terminal as shown below

```

hduser@muhammad-VM: ~/Desktop/Tutorial_3/top10
hduser@muhammad-VM:~/Desktop/Tutorial_3/top10$ ls
auto-mpg-data-original.txt  topTenCarsMappers.py  topTenCarsReducer.py
hduser@muhammad-VM:~/Desktop/Tutorial_3/top10$ jps
4272 Jps
3648 ResourceManager
3235 DataNode
3466 SecondaryNameNode
3053 NameNode
3790 NodeManager
hduser@muhammad-VM:~/Desktop/Tutorial_3/top10$ hadoop jar $HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-3.4.0.jar
-mapper ./topTenCarsMappers.py -reducer ./topTenCarsReducer.py -input /user1/auto-mpg-data-original.txt -output /output2
2024-08-27 10:10:58,144 INFO impl.MetricsConfig: Loaded properties from hadoop-metrics2.properties
2024-08-27 10:10:58,228 INFO impl.MetricsSystemImpl: Scheduled Metric snapshot period at 10 second(s).
2024-08-27 10:10:58,229 INFO impl.MetricsSystemImpl: JobTracker metrics system started
2024-08-27 10:10:58,247 WARN impl.MetricsSystemImpl: JobTracker metrics system already initialized!
2024-08-27 10:10:58,406 INFO mapred.FileInputFormat: Total input files to process : 1
2024-08-27 10:10:58,452 INFO mapreduce.JobSubmitter: number of splits:1
2024-08-27 10:10:58,551 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_local1154753362_0001
2024-08-27 10:10:58,555 INFO mapreduce.JobSubmitter: Executing with tokens: []
2024-08-27 10:10:58,678 INFO mapreduce.Job: The url to track the job: http://localhost:8080/
2024-08-27 10:10:58,683 INFO mapred.LocalJobRunner: OutputCommitter set in config null
2024-08-27 10:10:58,685 INFO mapreduce.Job: Running job: job_local1154753362_0001
2024-08-27 10:10:58,687 INFO mapred.LocalJobRunner: OutputCommitter is org.apache.hadoop.mapred.FileOutputCommitter
2024-08-27 10:10:58,700 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 2

```

2) After successful completion of MapReduce job, the following screenshot shows that **output2** folder is created on **hdfs**. We can display the output by using the **hadoop fs \$!ls** command.

```

hduser@muhammad-VM: ~/Desktop/Tutorial_3/top10
Spilled Records=20
Shuffled Maps =1
Failed Shuffles=0
Merged Map outputs=1
GC time elapsed (ms)=10
Total committed heap usage (bytes)=483393536
Shuffle Errors
BAD_ID=0
CONNECTION=0
IO_ERROR=0
WRONG_LENGTH=0
WRONG_MAP=0
WRONG_REDUCE=0
File Input Format Counters
Bytes Read=19362
File Output Format Counters
Bytes Written=495
2024-08-27 10:10:59,701 INFO streaming.StreamJob: Output directory: /output2
hduser@muhammad-VM:~/Desktop/Tutorial_3/top10$ hadoop fs -ls /output2
Found 2 items
-rw-r--r-- 1 hduser supergroup 0 2024-08-27 10:10 /output2/_SUCCESS
-rw-r--r-- 1 hduser supergroup 495 2024-08-27 10:10 /output2/part-000000
hduser@muhammad-VM:~/Desktop/Tutorial_3/top10$ hadoop fs -cat /output2/part-000000
13 8 400 175 5140 12 71 1 "pontiac safari (sw)"
11 8 400 150 4997 14 73 1 "chevrolet impala"
12 8 383 180 4955 11.5 71 1 "dodge monaco (sw)"
12 8 429 198 4952 11.5 73 1 "mercury marquis brougham"
12 8 455 225 4951 11 73 1 "buick electra 225 custom"
12 8 400 167 4906 12.5 73 1 "ford country"
13 8 400 170 4746 12 71 1 "ford country squire (sw)"
13 8 440 215 4735 11 73 1 "chrysler new yorker brougham"
9 8 304 193 4732 18.5 70 1 "hi 1200d"
13 8 350 150 4699 14.5 74 1 "buick century luxury (sw)"
hduser@muhammad-VM:~/Desktop/Tutorial_3/top10$

```

3) The output is based on the top 10 records based on the dataset as shown in the above screenshot.

References:

- <https://hadoop.apache.org/docs/current/hadoop-streaming/HadoopStreaming.html>
- <https://github.com/Virksaabnavjot/Mapreduce-Python/tree/master>