**EXP 3** 210701224

**Map Reduce program to process a weather dataset.**

**Aim:**

To implement MapReduce program to process a weather dataset

**Procedure:**

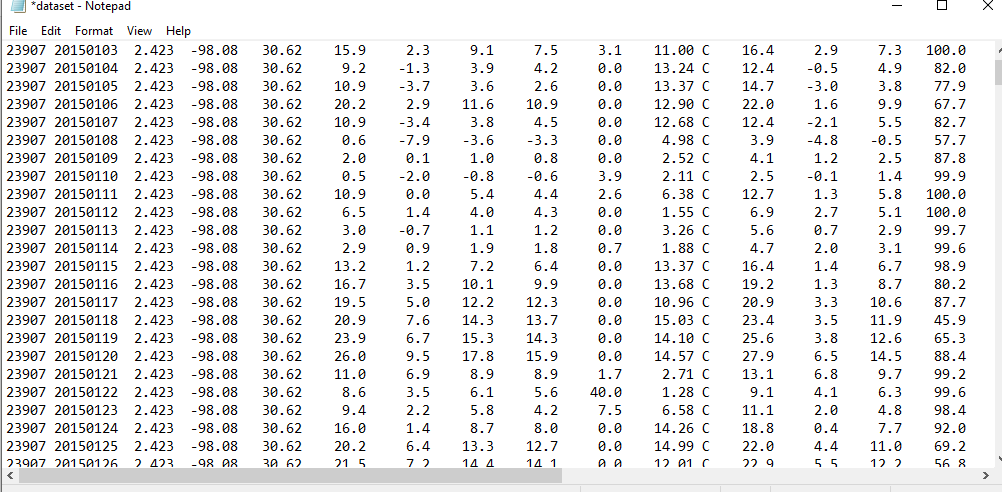
**Step 1: Create Data File:**

Create a file named "word\_count\_data.txt" and populate it with text data that you wish to analyse.

Login with your hadoop user.

# Download the dataset (weather data)

Output:



# Step 2: Mapper Logic - mapper.py:

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

nano mapper.py

# Copy and paste the mapper.py code

#!/usr/bin/env python import sys

# input comes from STDIN (standard input)

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

for line in sys.stdin:

# remove leading and trailing whitespace line = line.strip()

# split the line into words words = line.split()

#See the README hosted on the weather website which help us understand how each position represents a column

month = line[10:12] daily\_max = line[38:45] daily\_max = daily\_max.strip() # increase counters

for word in words:

# write the results to STDOUT (standard output);

# what we output here will be go through the shuffle proess and then # be the input for the Reduce step, i.e. the input for reducer.py

#

# tab-delimited; month and daily max temperature as output print ('%s\t%s' % (month ,daily\_max))

.

# Step 3: Reducer Logic - reducer.py:

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

nano reducer.py

# Copy and paste the reducer.py code

# reducer.py

#!/usr/bin/env python

from operator import itemgetter import sys

#reducer will get the input from stdid which will be a collection of key, value(Key=month , value= daily max temperature)

#reducer logic: will get all the daily max temperature for a month and find max temperature for the month

#shuffle will ensure that key are sorted(month) current\_month = None

current\_max = 0 month = None

# input comes from STDIN for line in sys.stdin:

# remove leading and trailing whitespace line = line.strip()

# parse the input we got from mapper.py month, daily\_max = line.split('\t', 1)

# convert daily\_max (currently a string) to float try:

daily\_max = float(daily\_max) except ValueError:

# daily\_max was not a number, so silently # ignore/discard this line

continue

# this IF-switch only works because Hadoop shuffle process sorts map output # by key (here: month) before it is passed to the reducer

if current\_month == month:

if daily\_max > current\_max: current\_max = daily\_max

else:

if current\_month:

# write result to STDOUT

print ('%s\t%s' % (current\_month, current\_max)) current\_max = daily\_max

current\_month = month

# output of the last month

if current\_month == month:

print ('%s\t%s' % (current\_month, current\_max))

# Step 4: Prepare Hadoop Environment:

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

start-all.sh

# Step 6: Make Python Files Executable:

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

chmod 777 mapper.py reducer.py

# Step 7: Run the program using Hadoop Streaming:

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

access.

Then run the program using Hadoop Streaming. hadoop fs -mkdir -p /weatherdata

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

hadoop jar /home/sx/hadoop-3.2.3/share/hadoop/tools/lib/hadoop-streaming-3.2.3.jar \

-input /weatherdata/dataset.txt \

-output /weatherdata/output \

-file "/home/sx/Downloads/mapper.py" \

-mapper "python3 mapper.py" \

-file "/home/sx/Downloads/reducer.py" \

-reducer "python3 reducer.py"

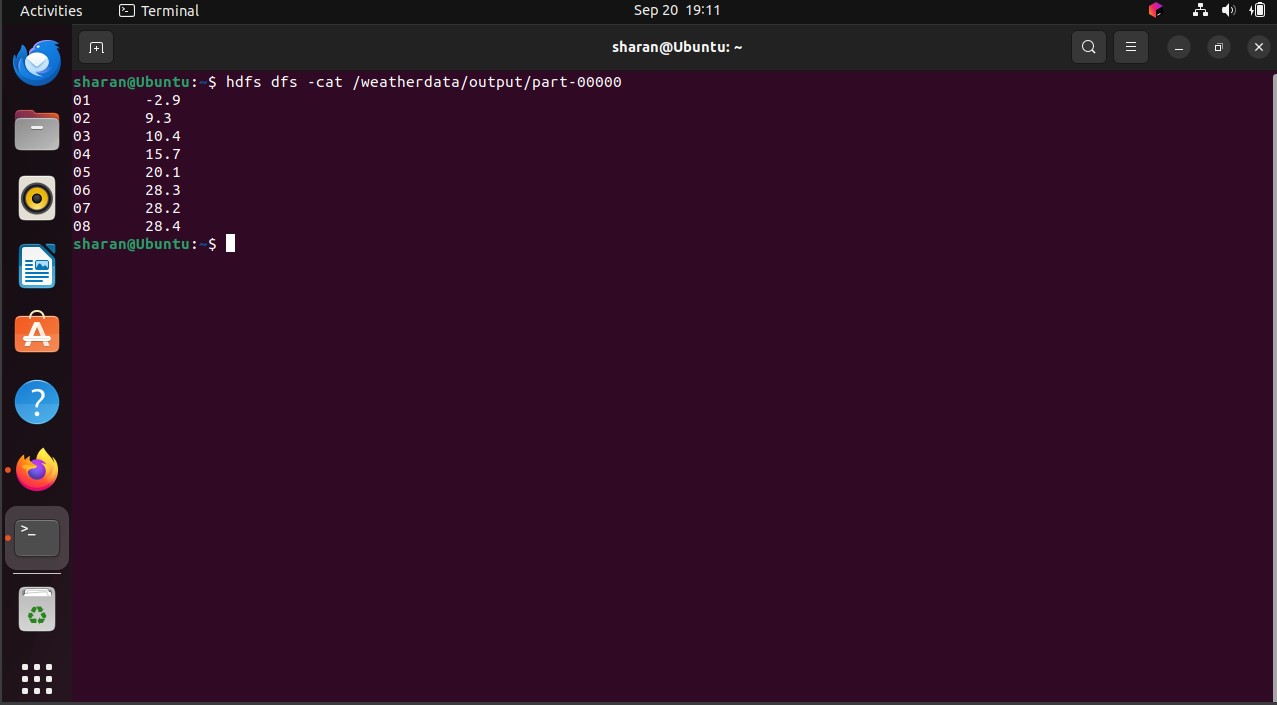
hdfs dfs -text /weatherdata/output/\* > /home/sx/Downloads/outputfile.txt

|  |
| --- |
|  |
| jbdkb |
| ouOutput: |

# Step 8: Check Output:

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

**OUTPUT:**



**Result:**

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