EXP 3: **Map Reduce program to process a weather dataset.**

AIM:

To implement MapReduce program to process a weather dataset.

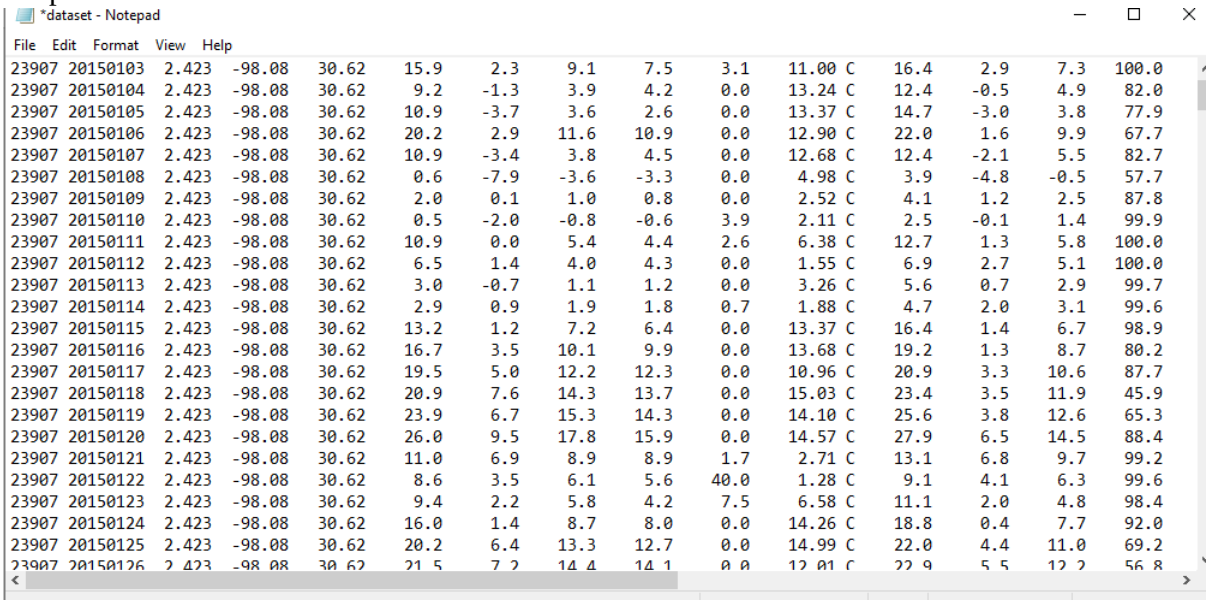
Procedure:

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:



23907	20150103	2.423	-98.08	30.62	15.9	2.3	9.1	7.5	3.1	11.00 C	16.4	2.9	7.3	100.0
23907	20150104	2.423	-98.08	30.62	9.2	-1.3	3.9	4.2	0.0	13.24 C	12.4	-0.5	4.9	82.0
23907	20150105	2.423	-98.08	30.62	10.9	-3.7	3.6	2.6	0.0	13.37 C	14.7	-3.0	3.8	77.9
23907	20150106	2.423	-98.08	30.62	20.2	2.9	11.6	10.9	0.0	12.90 C	22.0	1.6	9.9	67.7
23907	20150107	2.423	-98.08	30.62	10.9	-3.4	3.8	4.5	0.0	12.68 C	12.4	-2.1	5.5	82.7
23907	20150108	2.423	-98.08	30.62	0.6	-7.9	-3.6	-3.3	0.0	4.98 C	3.9	-4.8	-0.5	57.7
23907	20150109	2.423	-98.08	30.62	2.0	0.1	1.0	0.8	0.0	2.52 C	4.1	1.2	2.5	87.8
23907	20150110	2.423	-98.08	30.62	0.5	-2.0	-0.8	-0.6	3.9	2.11 C	2.5	-0.1	1.4	99.9
23907	20150111	2.423	-98.08	30.62	10.9	0.0	5.4	4.4	2.6	6.38 C	12.7	1.3	5.8	100.0
23907	20150112	2.423	-98.08	30.62	6.5	1.4	4.0	4.3	0.0	1.55 C	6.9	2.7	5.1	100.0
23907	20150113	2.423	-98.08	30.62	3.0	-0.7	1.1	1.2	0.0	3.26 C	5.6	0.7	2.9	99.7
23907	20150114	2.423	-98.08	30.62	2.9	0.9	1.9	1.8	0.7	1.88 C	4.7	2.0	3.1	99.6
23907	20150115	2.423	-98.08	30.62	13.2	1.2	7.2	6.4	0.0	13.37 C	16.4	1.4	6.7	98.9
23907	20150116	2.423	-98.08	30.62	16.7	3.5	10.1	9.9	0.0	13.68 C	19.2	1.3	8.7	80.2
23907	20150117	2.423	-98.08	30.62	19.5	5.0	12.2	12.3	0.0	10.96 C	20.9	3.3	10.6	87.7
23907	20150118	2.423	-98.08	30.62	20.9	7.6	14.3	13.7	0.0	15.03 C	23.4	3.5	11.9	45.9
23907	20150119	2.423	-98.08	30.62	23.9	6.7	15.3	14.3	0.0	14.10 C	25.6	3.8	12.6	65.3
23907	20150120	2.423	-98.08	30.62	26.0	9.5	17.8	15.9	0.0	14.57 C	27.9	6.5	14.5	88.4
23907	20150121	2.423	-98.08	30.62	11.0	6.9	8.9	8.9	1.7	2.71 C	13.1	6.8	9.7	99.2
23907	20150122	2.423	-98.08	30.62	8.6	3.5	6.1	5.6	40.0	1.28 C	9.1	4.1	6.3	99.6
23907	20150123	2.423	-98.08	30.62	9.4	2.2	5.8	4.2	7.5	6.58 C	11.1	2.0	4.8	98.4
23907	20150124	2.423	-98.08	30.62	16.0	1.4	8.7	8.0	0.0	14.26 C	18.8	0.4	7.7	92.0
23907	20150125	2.423	-98.08	30.62	20.2	6.4	13.3	12.7	0.0	14.99 C	22.0	4.4	11.0	69.2
23907	20150126	2.423	-98.08	30.62	21.5	7.2	14.4	14.1	0.0	12.01 C	22.9	5.5	12.2	56.8

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.

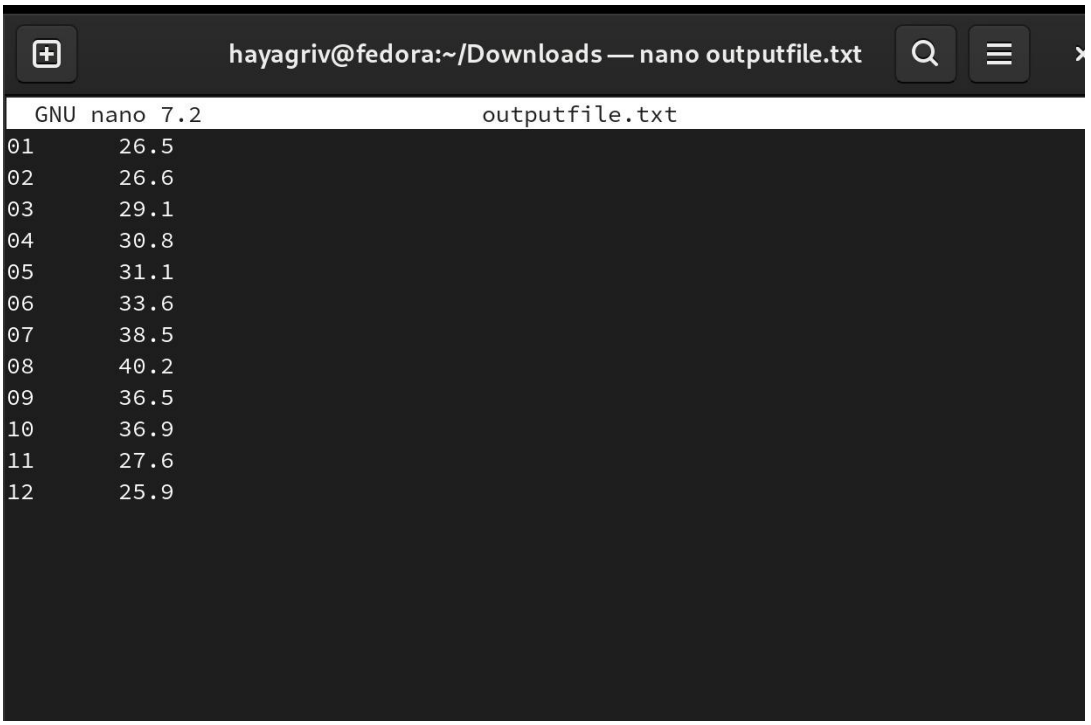
Then run the program using Hadoop Streaming. hadoop

```
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
```

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
```



Result:

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