Fahad Fiaz – (303141) – G2

System Info:

Processor	i7-5500U , 2.40GHz
Cores	4
Operating system	Windows 64 Bit
Ram	8GB
Programming Language	Python 3.7.7

Q1:

Making directory in HDFS

```
PS E:\hadoop-2.7.0\hadoop-2.7.0> hadoop fs -mkdir /wordcount_ex/input
```

Copying text file to HDFS

```
PS E:\hadoop-2.7.0\hadoop-2.7.0> hadoop fs -put -f  E:\world_count\input_data\Words.txt /wordcount_ex/input
PS E:\hadoop-2.7.0\hadoop-2.7.0>
```

Running the prebuild World count program to count occurrence of words.

```
PS E:\hadoop-2.7.0\hadoop-2.7.0> bin\yarn jar E:\hadoop-2.7.0\hadoop-2.7.0\share\hadoop\mapreduce\hadoop-mapreduce-examples-2.7.0.jar w
ordcount /wordcount_ex/input/Words.txt WordCount_less_text
```

```
PS E:\hadoop-2.7.0\hadoop-2.7.0> hdfs dfs -cat wordCount_less
Author: 1
Brothers        2
EBook   1
Edgar   1
Edwardes        1
Fairy   2
Grimm   2
Grimms┐╜        2
Gutenberg       2
License 1
Marian  1
Project 2
Tales   1
Tales,  1
Taylor  1
The     3
This    1
Title:  1
Translator:     1
You     1
almost  1
and     2
anyone  1
anywhere        1
at      2
away    1
by      1
copy    1
cost    1
eBook   2
for     1
give    1
included        1
is      1
it      2
it,     1
may     1
no      2
of      3
online  1
```

/ Go!

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
drwxr-xr-x	fahad	supergroup	0 B	6/15/2020, 5:19:51 AM	0	0 B	sample
drwxr-xr-x	fahad	supergroup	0 B	6/15/2020, 5:55:33 AM	0	0 B	user
drwxr-xr-x	fahad	supergroup	0 B	6/15/2020, 5:42:32 AM	0	0 B	wordcount_ex

Hadoop, 2014.

Q2:

Mapper:

It will read data from STDIN, Split rows and output specific rows to STDOUT

```
for line in sys.stdin:
    line = line.split(',')
    try:
        print((line[3], line[6], line[8]))
    except Exception as e:
        print("Error: ", e)
```

Reducer:

It will read the results of mapper.py from STDIN (the output format of mapper.py and the expected input format of reducer.py must match).

Since the value received from STDIN is in string format so first we need to convert this string tuple to tuple.

```
line = literal_eval(line)  # convert string-tuple to tuple
try:
    line = (line[0], float(line[1]), float(line[2]))
except Exception as e:
    line = None
```

Following lines check whether if we are parsing a specific airport for 1st time or Not. If we are parsing it for first time it simply saves the airport data in "final" dictionary with airport name as key. If the airport name in new row is already in final dictionary then we compare current row data with already previous save data in final dictionary to find maximum, minimum, and average departure delay for each airport.

In "final" dictionary, key is airport names and values are tuple. In each tuple first column represent airport name, 2nd represent minimum, 3rd represent maximum, 4rth represent count of specific airport in dataset and 5th represent total arrival delay for specific airport.

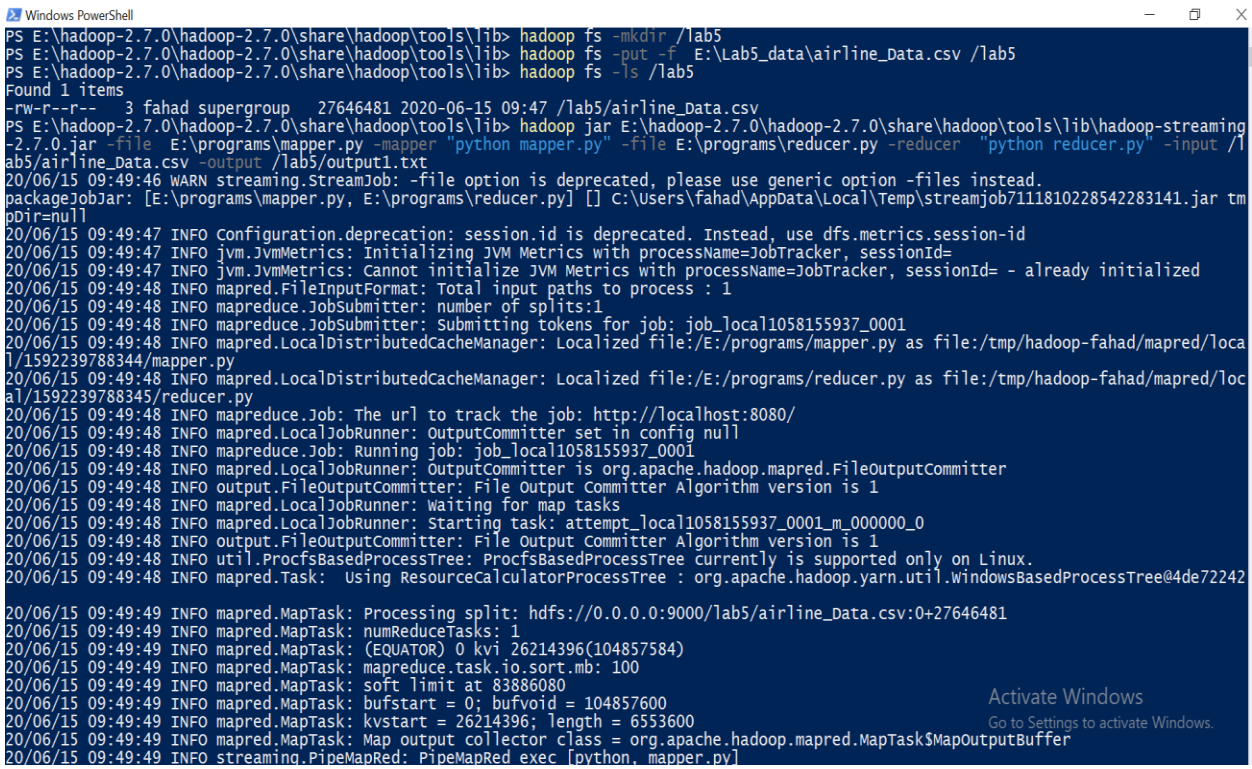
```
if line:
    if line[0] in final.keys():
        old_val = final[line[0]]
        airport = old_val[0]
        low = old_val[1]
        high = old_val[2]
        total = old_val[3] + line[1]
        count = old_val[4] + 1
        total_arrival_delay = old_val[5] + line[2]
```

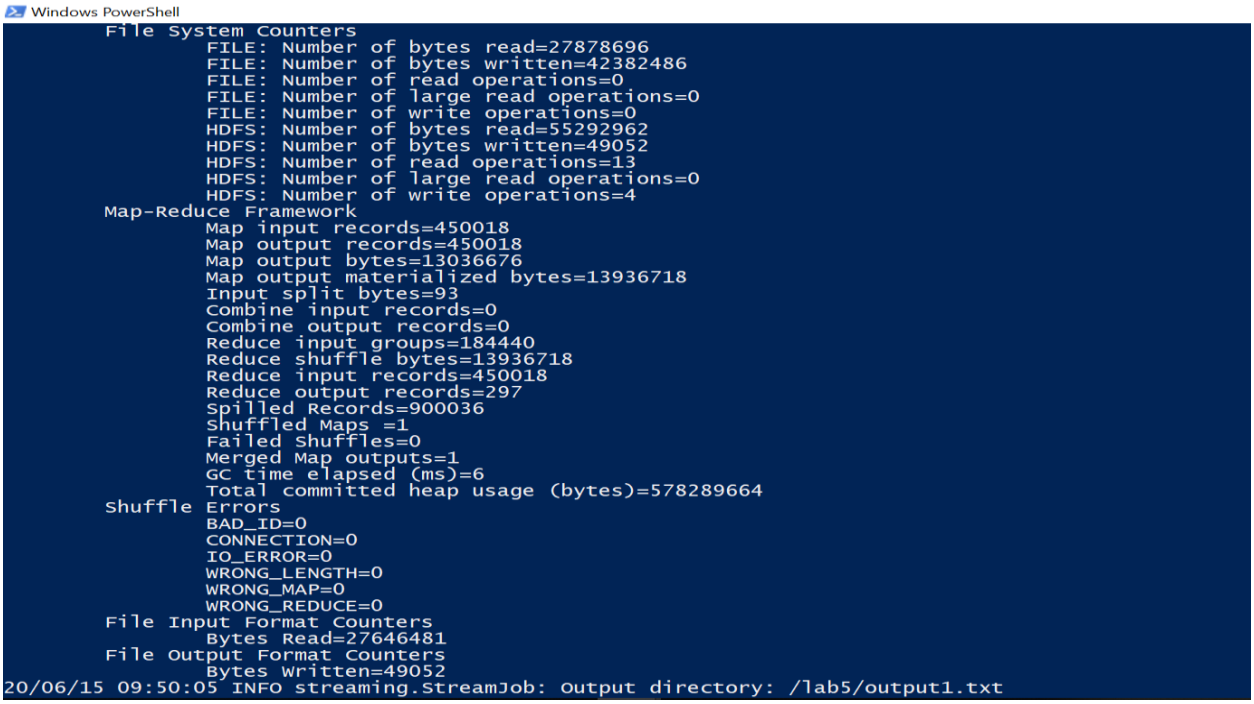
```
        if line[1] < old_val[1]:
            low = line[1]
        if line[1] > old_val[2]:
            high = line[1]

        final[line[0]] = (airport, low, high, total, count, total_arrival_delay)

    else:
        # airport_Names.append(line[0])
        final[line[0]] = (line[0], 9999, -9999, 0, 1, 0)
```

Show the final results:

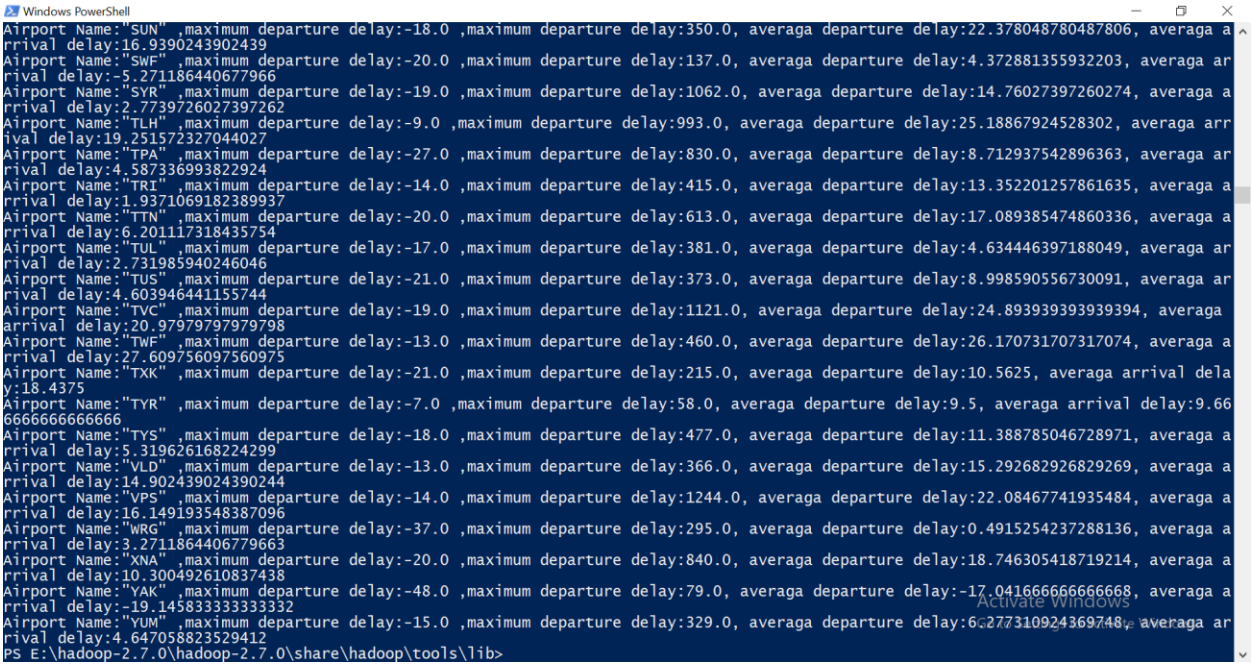
```
for k,v in final.items():
print("Airport Name:{} ,maximum departure delay:{} ,maximum departure delay:{},
averaga departure delay:{}, averaga arrival
delay:{}".format(v[0],v[1],v[2],v[3]/v[4],v[5]/v[4]))
```

Running commands:





Output: Only part of output shown



To calculates the list of top 10 airports by their average Arrival delay we divide total arrival delay with total occurrence of specific airport in dataset. Then find the list of top 10 airports by their average Arrival delay.

```
Arrival_Delay_Average={}
for k,v in final.items():
    Arrival_Delay_Average[v[0]]=v[5]/v[4]

d = Counter(Arrival_Delay_Average)
for k, v in d.most_common(10):
    print("Airport Name:{}, averaga arrival delay:{}".format(k,v))
```

```
PS E:\hadoop-2.7.0\hadoop-2.7.0\share\hadoop\tools\lib> hadoop dfs -cat /lab5/output5.txt/*
DEPRECATED: Use of this script to execute hdfs command is deprecated.
Instead use the hdfs command for it.
Airport Name:"LAW", averaga arrival delay:85.25925925925925
Airport Name:"GGG", averaga arrival delay:64.0
Airport Name:"EKO", averaga arrival delay:62.95918367346939
Airport Name:"BPT", averaga arrival delay:46.0
Airport Name:"LWS", averaga arrival delay:42.744680851063826
Airport Name:"ABR", averaga arrival delay:34.95
Airport Name:"ASE", averaga arrival delay:32.87064676616915
Airport Name:"HDN", averaga arrival delay:31.818181818181817
Airport Name:"JAC", averaga arrival delay:29.457575757575757
Airport Name:"ABI", averaga arrival delay:29.40740740740741
PS E:\hadoop-2.7.0\hadoop-2.7.0\share\hadoop\tools\lib>
```