**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

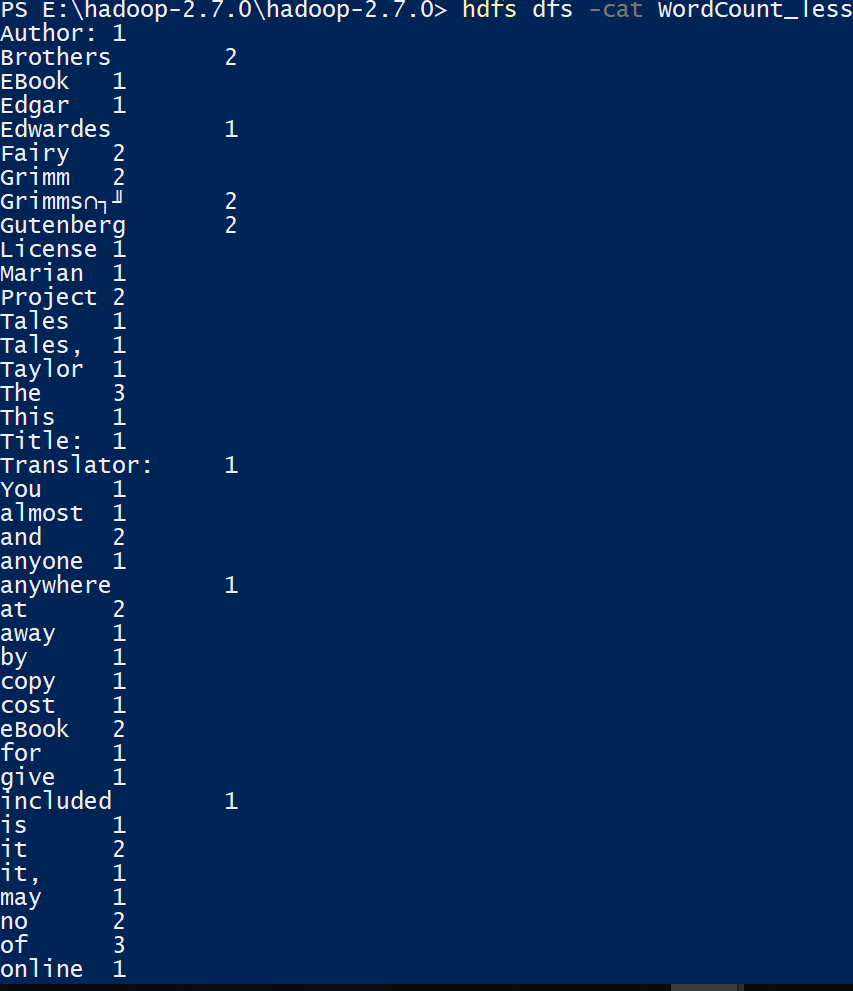
E:\DDA\exercise_5\2.PNG

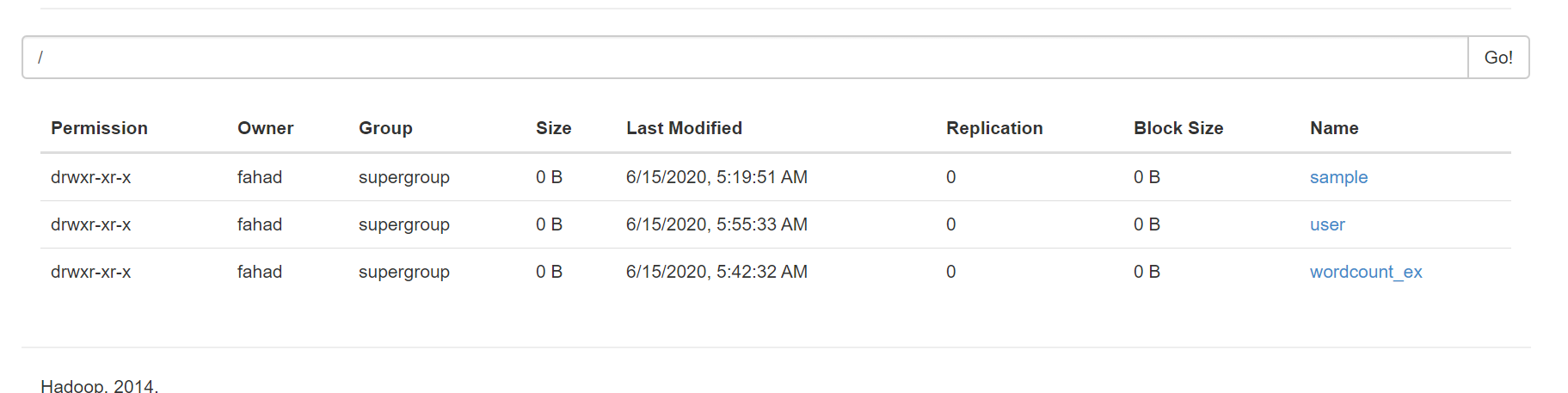
Copying text file to HDFS

E:\DDA\exercise_5\3.PNG

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

E:\DDA\exercise_5\4.PNG





**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.

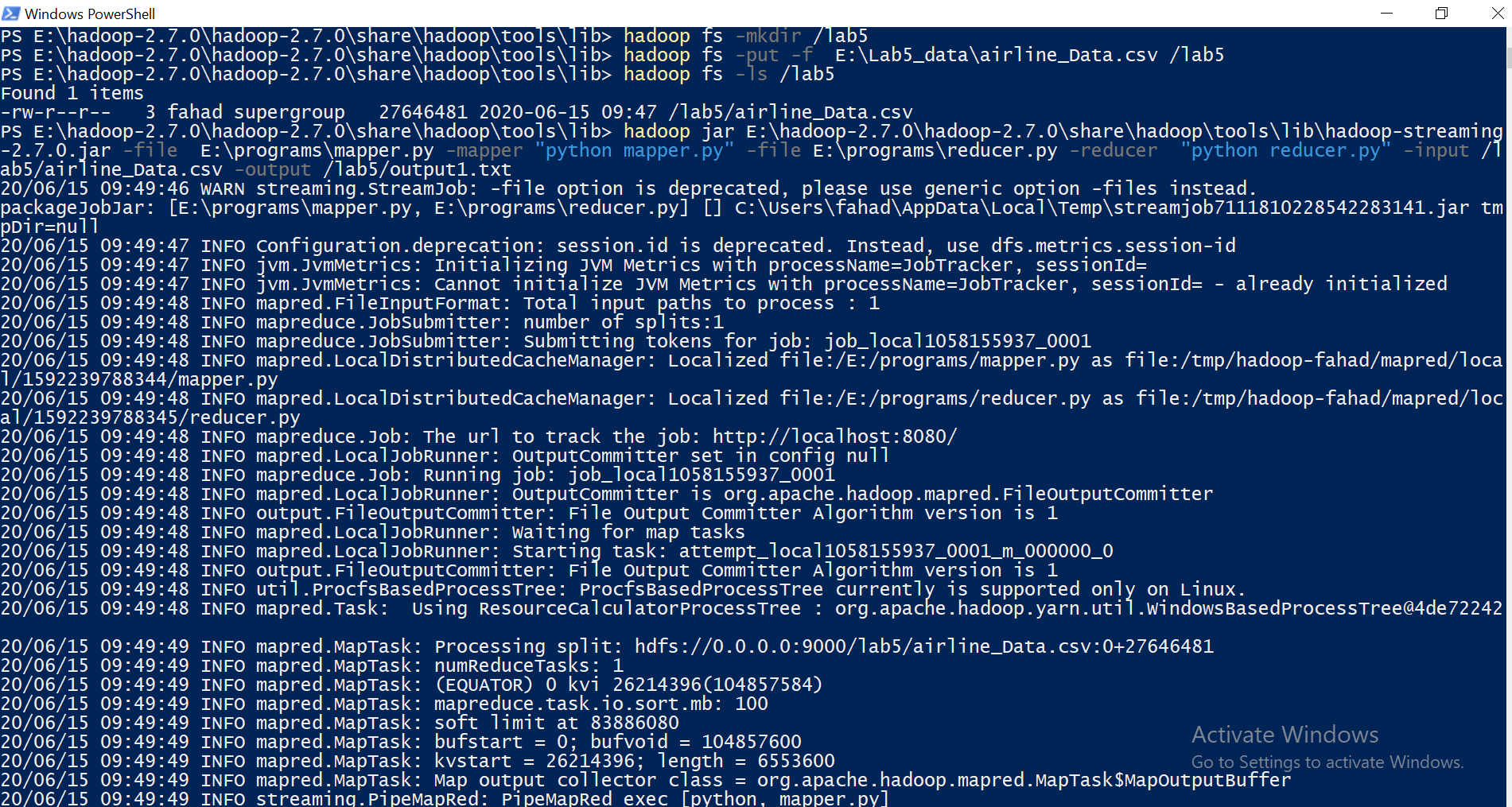
In “final” dictionary, key is airport names and values are tuple. In each tuple first column represent airport name, 2nd represent minimum, 3rd represent maximum, 4rth represent count of specific airport in dataset and 5th represent total arrival delay for specific airport.

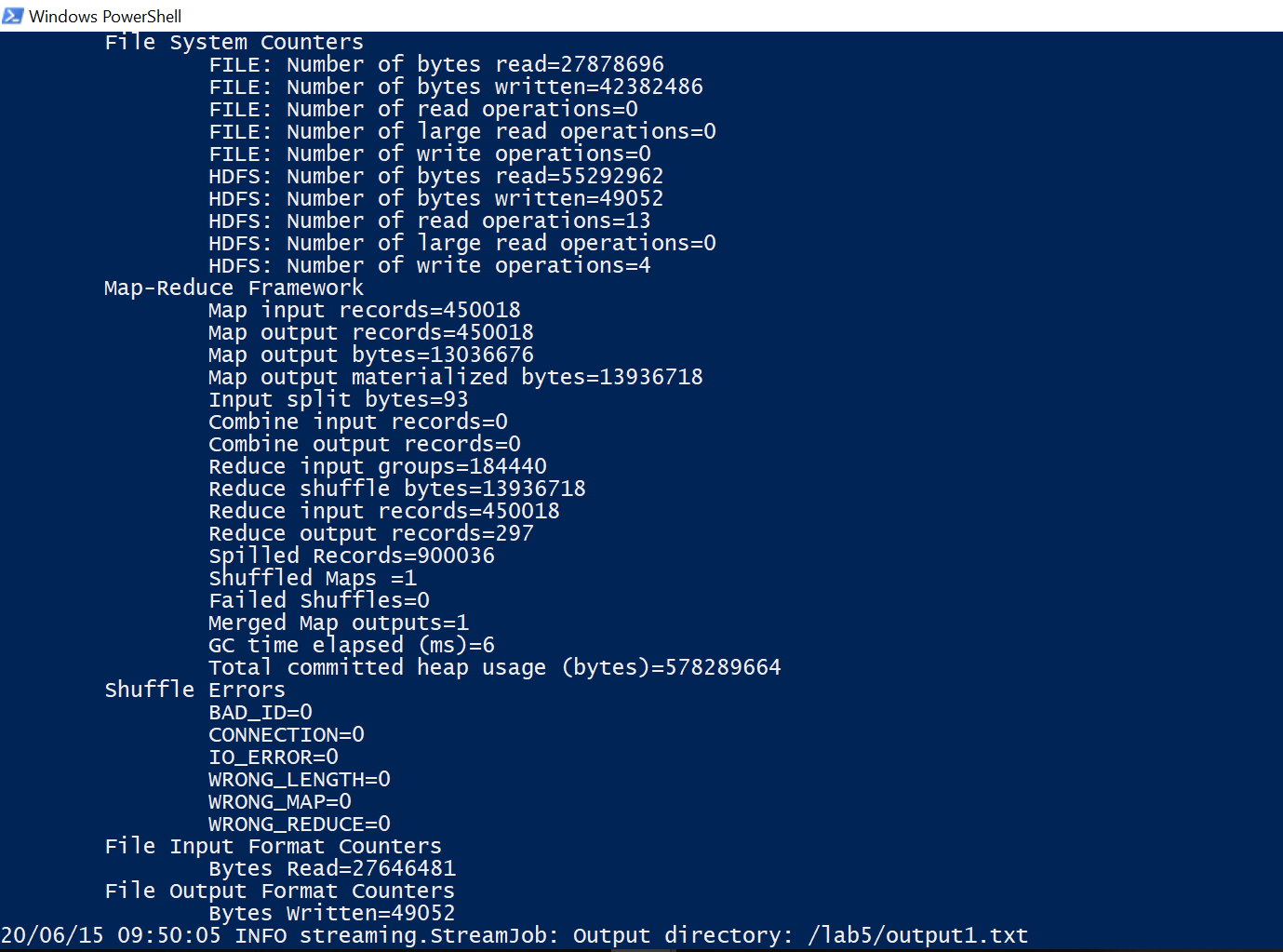
if line:  
 if line*[*0*]* in final.keys*()*:  
 old\_val = final*[*line*[*0*]]* airport = old\_val*[*0*]* low = old\_val*[*1*]* high = old\_val*[*2*]* total = old\_val*[*3*]* + line*[*1*]* count = old\_val*[*4*]* + 1  
 total\_arrival\_delay = old\_val*[*5*]* + line*[*2*]* if line*[*1*]* < old\_val*[*1*]*:  
 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:**

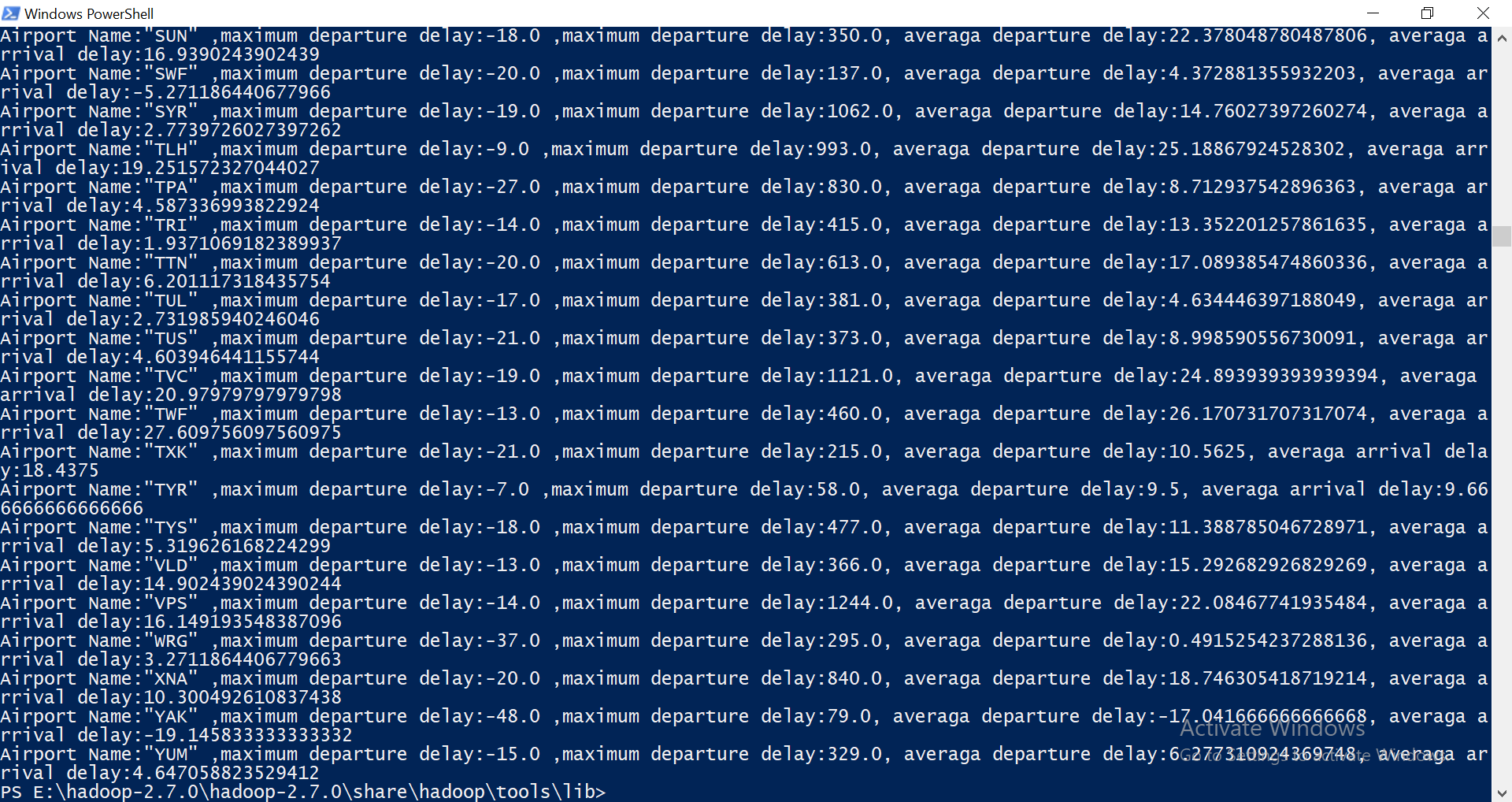
for k,v in final.items*()*:  
print*(*"Airport Name:{} ,maximum departure delay:{} ,maximum departure delay:{}, averaga departure delay:{}, averaga arrival delay:{}".format*(*v*[*0*]*,v*[*1*]*,v*[*2*]*,v*[*3*]*/v*[*4*]*,v*[*5*]*/v*[*4*]))*

**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*))*

