Small Project

Q1.

a.

Java environment used by hadoop:

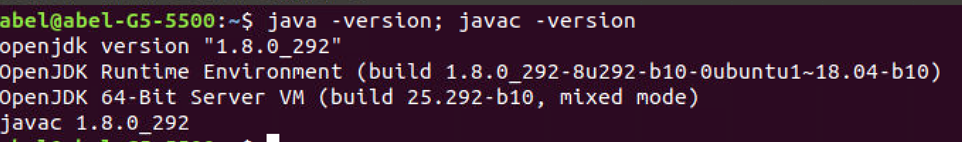


Figure 1: CMD for Java version

Configuration files that I’ve modified to setup the pseudo-distributed mode:

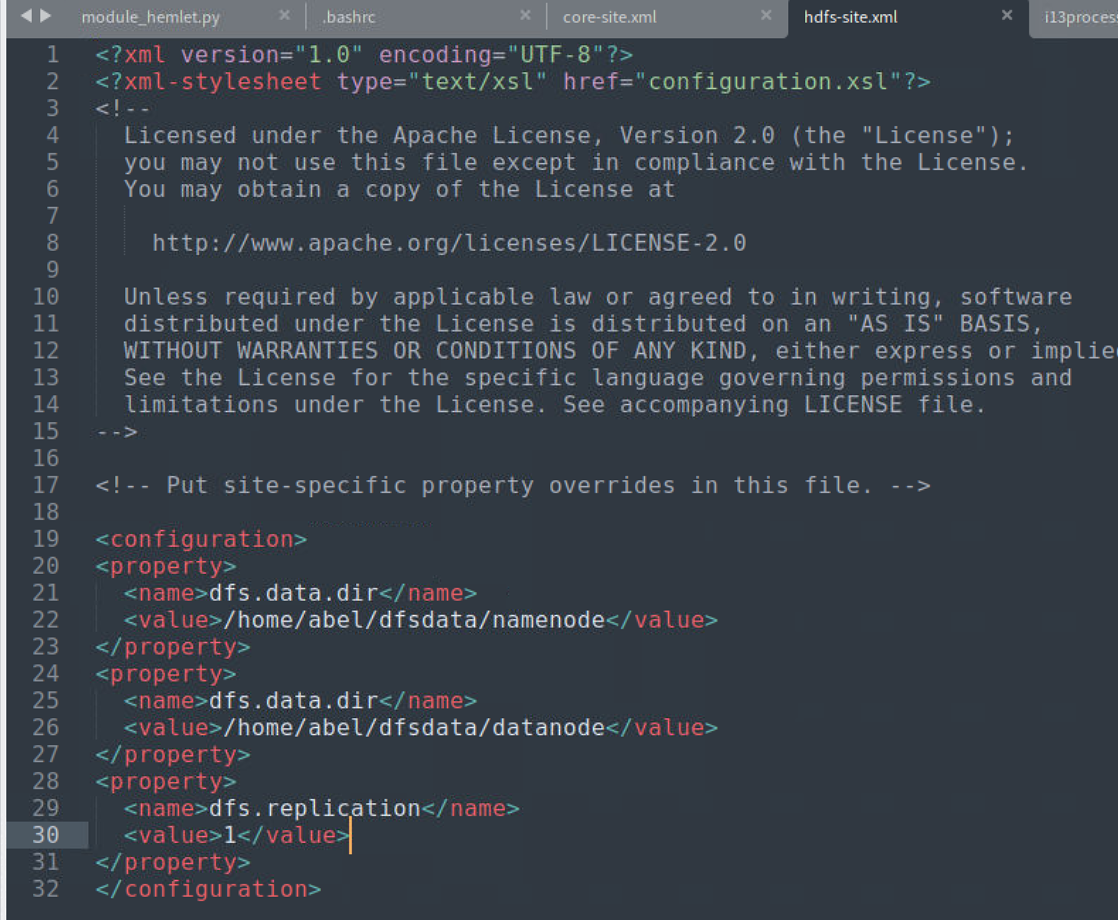


Figure 2: XML for hdfs-site

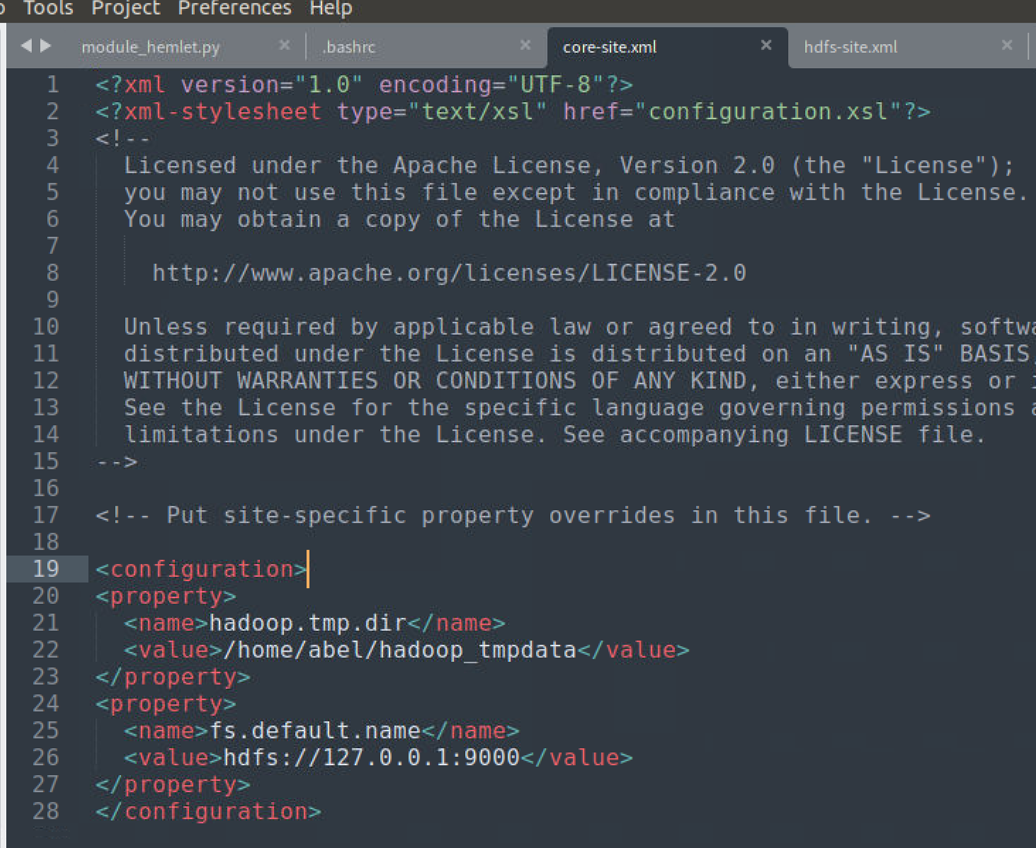
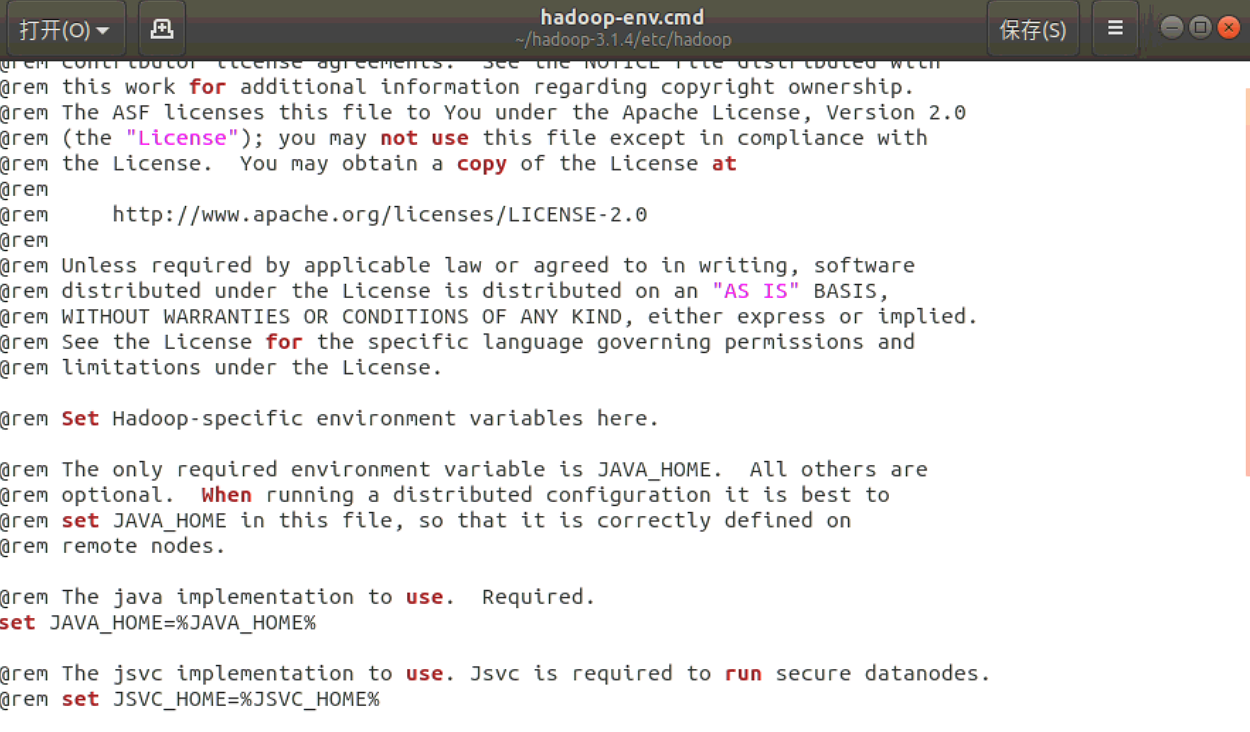
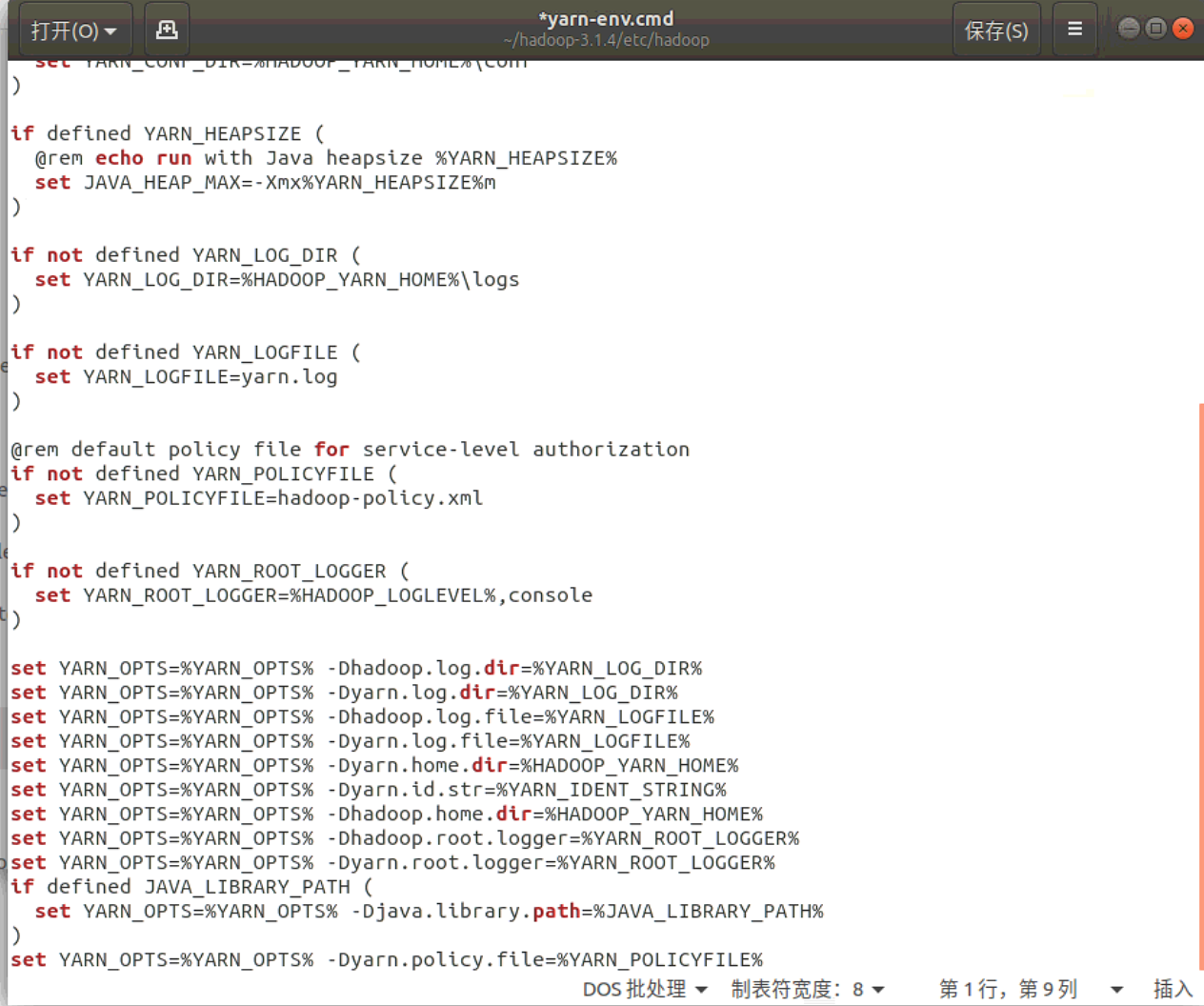


Figure 3: XML for core-site





b.

Firstly, use the command “hdfs dfs -mkdir /user/Jiang\_Ruizhao” to create a folder named Jiang-Ruizhao under the user folder. Then, write “Name: Jiang ruizhao, ID number: 510122199904210065” to the txt file called DMT1709166. Lastly, use hdfs dfs language to copy the txt file into the folder /user/Jiang\_Ruizhao.

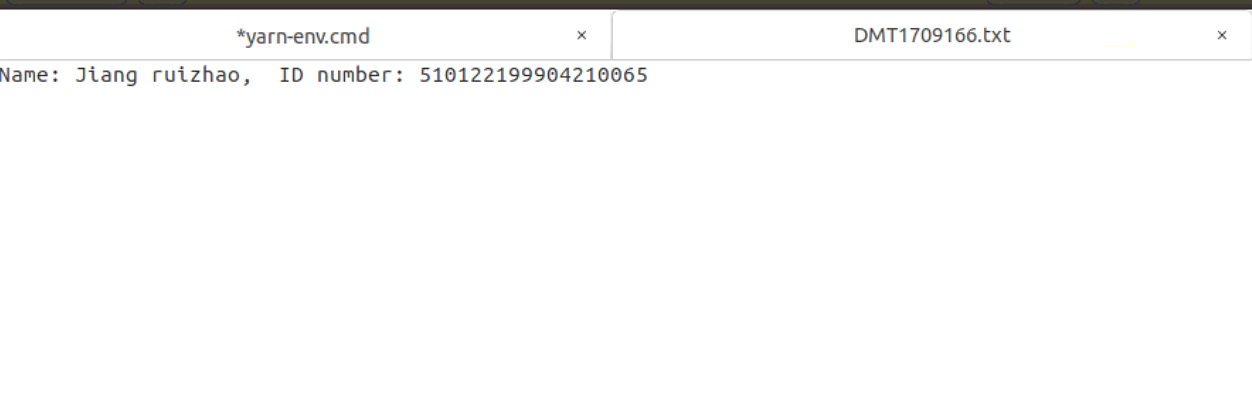


Figure 4-0: content of DMT1709166.txt

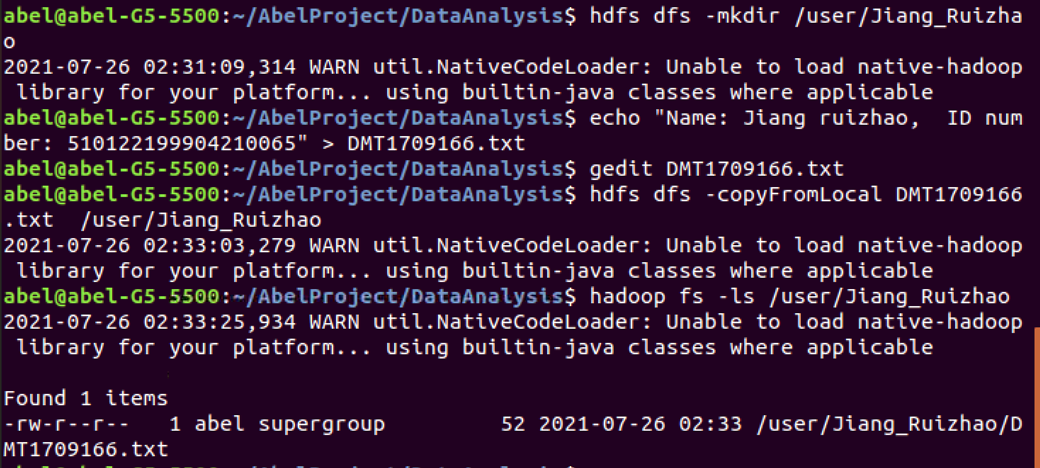


Figure 4-1: Implementation of writting DMT1709166.txt

c.

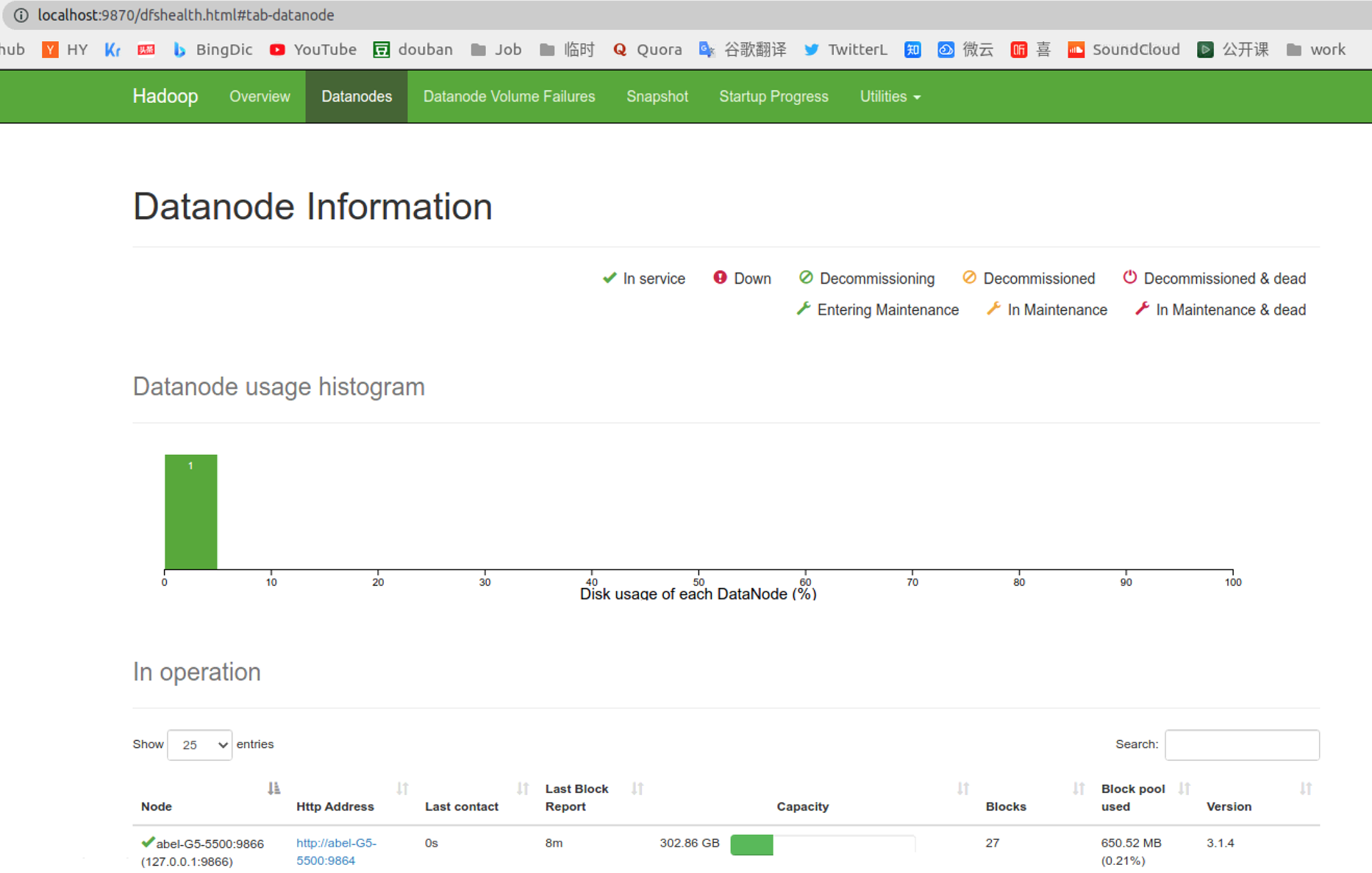


Figure 5: Datanode information web UI

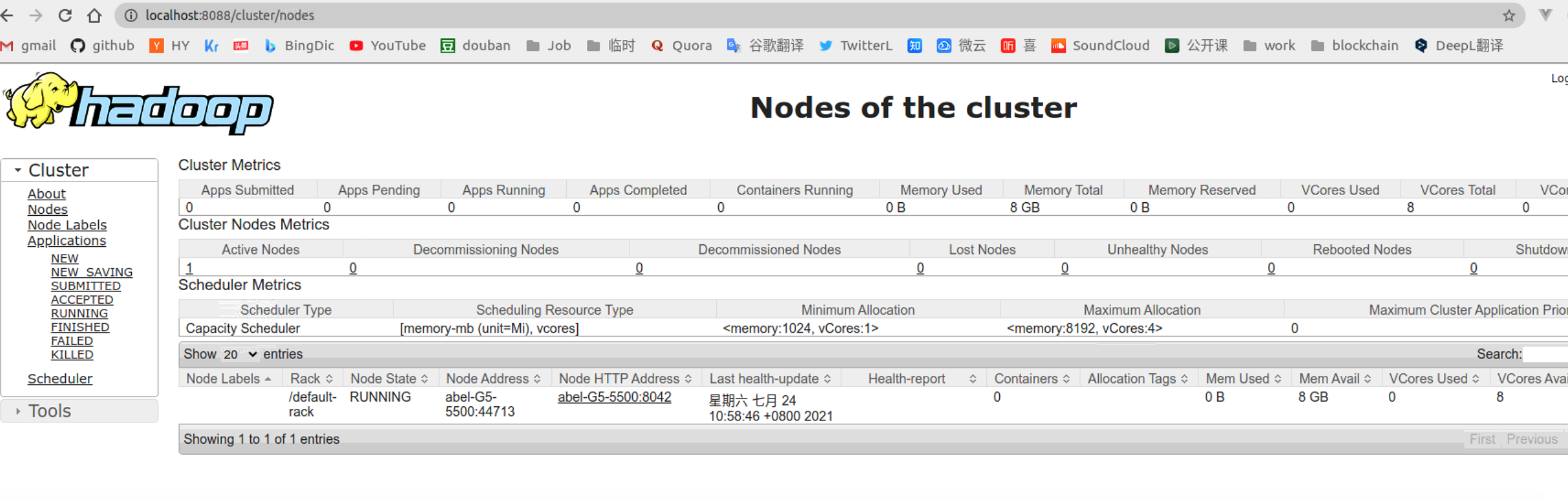


Figure 6: Nodes of the cluster in web UI

**Q2**

2.1.a

For the first map-reduce computing of task (1):

I use mapping function like:

(itemx in x) (itemy in y) -> (itemx, itemy) in (x,y)

At the same time, the reduce function is as follows:

Many itemx+itemy -> min number of the sum(itemx+itemy)

2.1.b

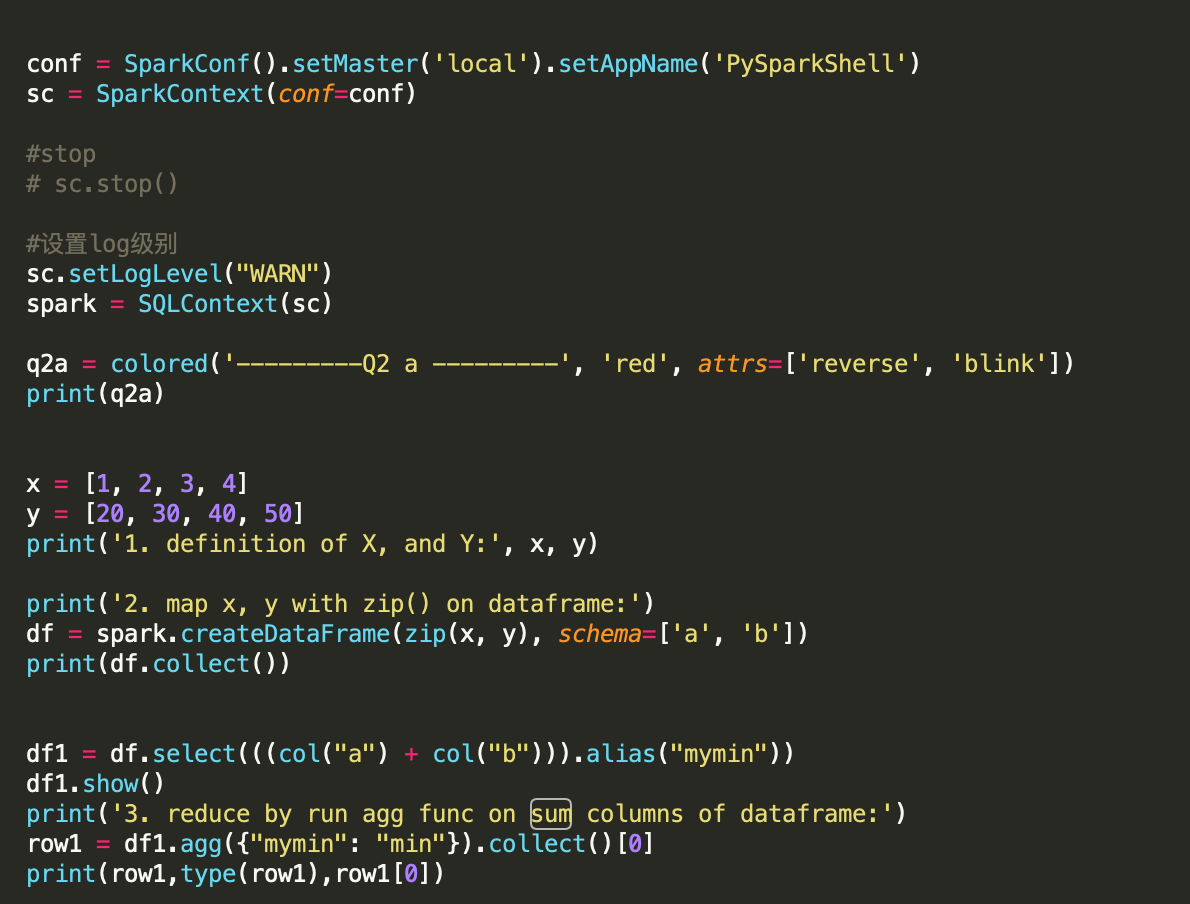


Figure 7: Codes to implement the task

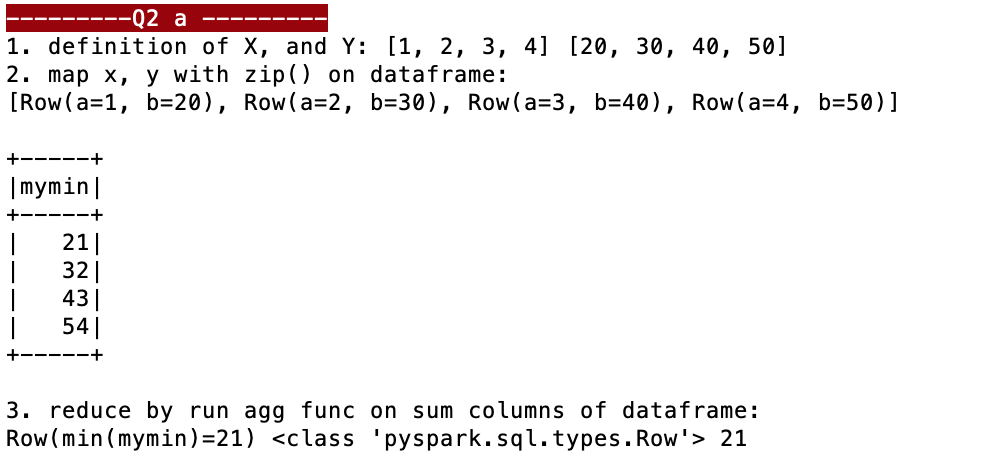


Figure 8: Results of the task

2.2.a

For the second map-reduce computing of task (2):

I use mapping function like:

(itemx in x) (columns\_of\_y in y) -> (itemx, columns\_of\_y) in (x,y)

At the same time, the reduce function is as follows:

Many itemx+columns\_of\_y -> min number of the sum(itemx+columns\_of\_y)

2.2.b



Figure 9: Codes to implement the task

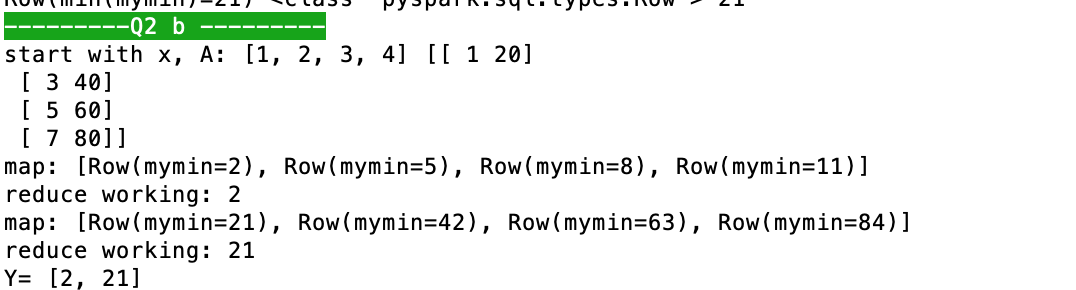


Figure 10: Results of the task

**Q3**

First, import corresponding module and load the file “flights2008.csv” into dataframe.

from pyspark import SparkContext, SparkConf

from pyspark.sql import HiveContext, Row

from pyspark.sql.types import IntegerType

import json

import sys

inputFile = 'flights2008.csv'

inputFile = '/Users/abel/Downloads/spare\_time/working/b13590\_1.8k\_spark\_7月28日/flights2008.csv'

conf = SparkConf().setAppName("SparkSQLAirTransit")

SparkConf().set("spark.sql.legacy.timeParserPolicy","LEGACY")

sc = SparkContext.getOrCreate(conf=conf)

hiveCtx = HiveContext(sc)

print("Loading flights from " + inputFile)

input = hiveCtx.read.option("header",True).csv(inputFile,inferSchema =True)

input.registerTempTable("air\_transit")

myair\_transits = hiveCtx.sql("SELECT \* FROM air\_transit LIMIT 3")

print('myair\_transits:' )

for item in myair\_transits.collect():

print(item, '\n')

**(a)**

print('1. column=',len(myair\_transits.columns))

myair\_transits = hiveCtx.sql("SELECT count(\*) FROM air\_transit ")

print('1. rows : ', myair\_transits.collect())

There are 7,009,728 rows and 13 columns in the “flights”.

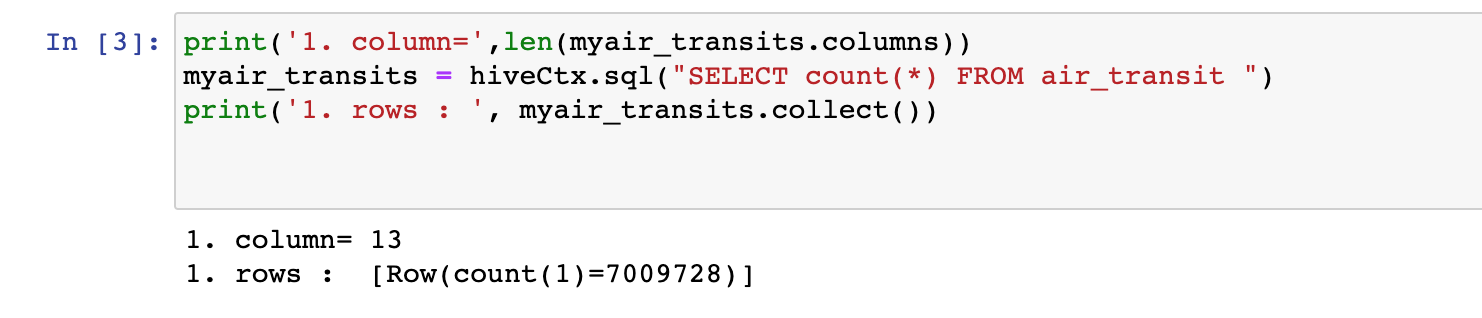


Figure 11: Results of the task (a)

There are 7,009,728 rows and 13 columns in the “flights”.

**(b)**

print('2. print out the schema of this dataframe')

input.printSchema()

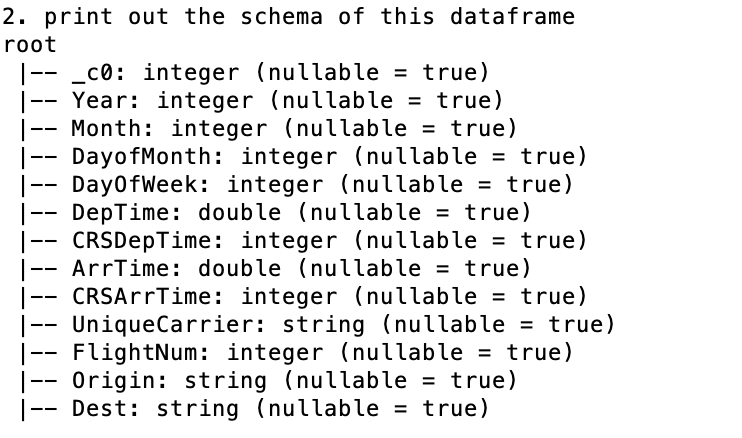


Figure 12: Results of the task (b)

The schema of the dataframe “flight” is shown above.

**(c)**

print('3. List the distinct carriers of in the dataframe:')

myair\_transits = hiveCtx.sql("SELECT distinct UniqueCarrier FROM air\_transit ")

print(myair\_transits.collect())

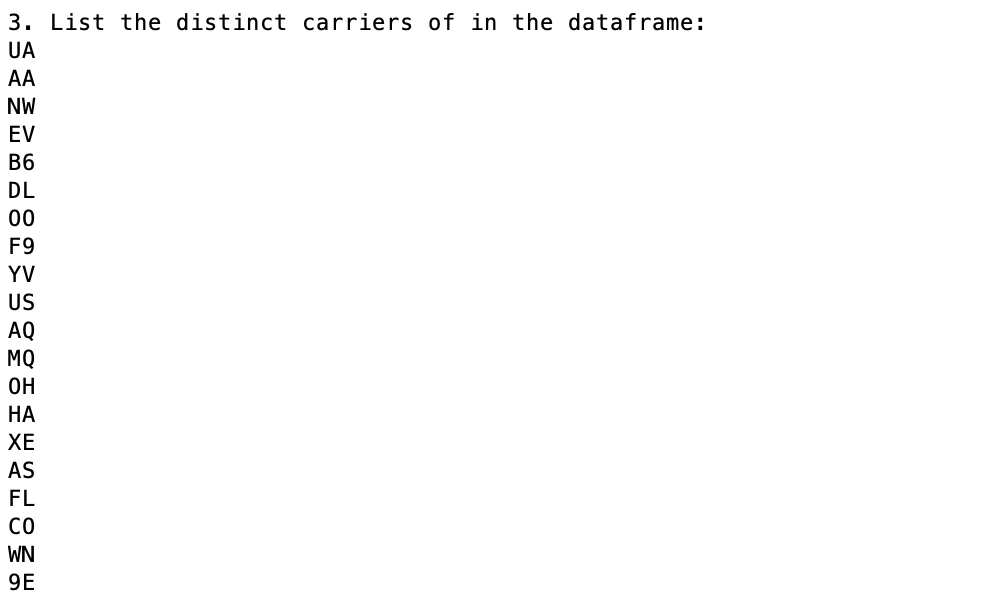


Figure 13: Results of the task (c)

Select the column of “UniqueCarrier”, and then list the distinct carriers in the dataframe.

**(d)**

print('4. How many flights in total are there in January:')

myair\_transits = hiveCtx

.sql("SELECT count(\*) FROM air\_transit where Month=1 ")

print(myair\_transits.collect())

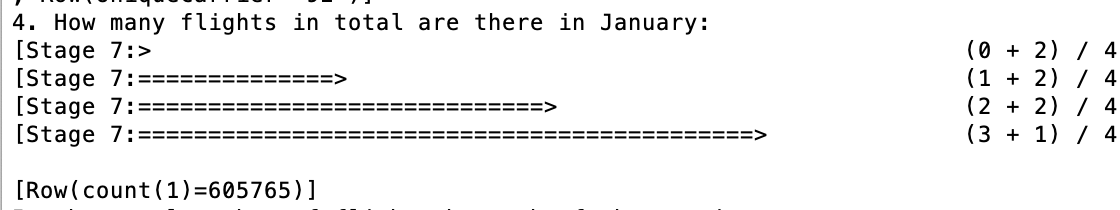


Figure 14: Results of the task (d)

**(e)**

print('5. the total number of flights by each of the carriers')

myair\_transits = hiveCtx.sql("SELECT UniqueCarrier, count(\*) FROM air\_transit group by UniqueCarrier ")

print('UniqueCarrier', ' count(\*)')

for row in myair\_transits.collect():

print(row[0], row[1])

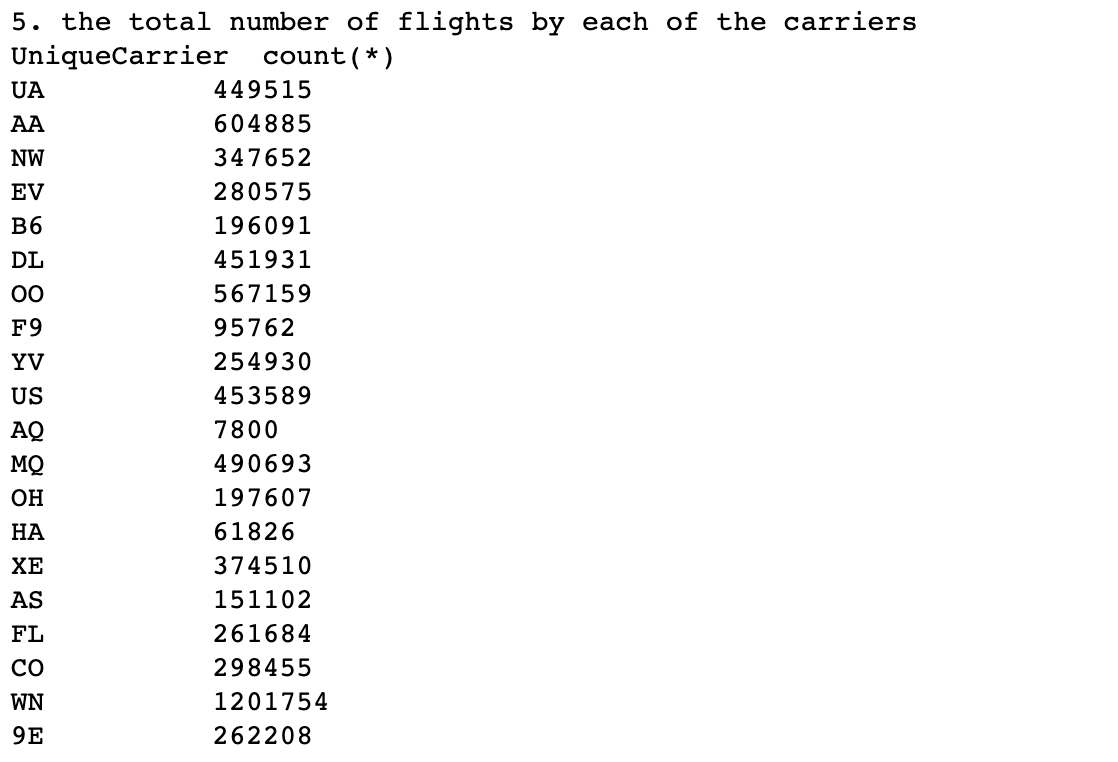


Figure 15: Results of the task (e)

**(f)**

print('6. Count the total number of flights in the first half year (month 1-6) by each of the carriers.')

myair\_transits = hiveCtx.sql("SELECT UniqueCarrier, count(\*) FROM air\_transit where Month <= 6 group by UniqueCarrier ")

print('UniqueCarrier', ' count(\*)')

for row in myair\_transits.collect():

print(row[0], row[1])

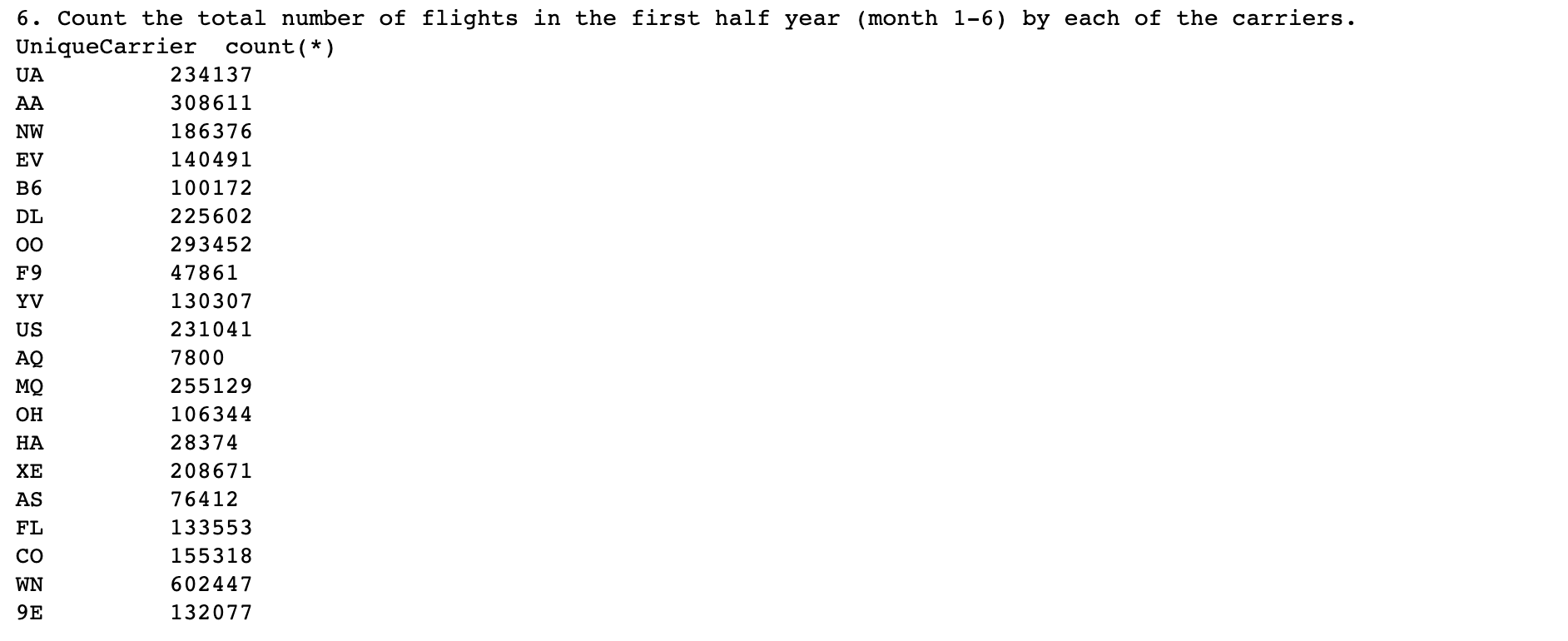


Figure 16: Results of the task (f)