

Consider the data set [NYC airbnb](#)

(<https://www.kaggle.com/code/dgomonov/data-exploration-on-nyc-airbnb/data>)

- Write hadoop java program to do:

ทำการ pre-process data โดยการตัดบรรทัดแรกออกเนื่องจาก hadoop มองเป็น plain text ทำให้เกิด error และตัด column ที่ไม่จำเป็นออกเนื่องจากเกิดปัญหาบรรทัดเลื่อนจากอักขระบางตัวได้ผลลัพธ์ตาราง data ออกมาดังนี้

	A	B	C	D	E
1	2539	2787	Brooklyn	149	6
2	2595	2845	Manhattan	225	2
3	3647	4632	Manhattan	150	1
4	3831	4869	Brooklyn	89	1
5	5022	7192	Manhattan	80	1
6	5099	7322	Manhattan	200	1
7	5121	7356	Brooklyn	60	1
8	5178	8967	Manhattan	79	1
9	5203	7490	Manhattan	79	1
10	5238	7549	Manhattan	150	4
11	5295	7702	Manhattan	135	1
12	5441	7989	Manhattan	85	1
13	5803	9744	Brooklyn	89	3

ตามไฟล์ AB_NYC_2019.csv

1) What is the average price in dollar? What is the maximum price in dollar?

The screenshot shows a code editor with the file `AveragePrice.java` and a terminal window below it. The code is a Java class that uses Hadoop MapReduce to calculate the average price from a dataset. The terminal output shows the execution progress, including shuffle errors, file input/output counters, and the final result: Average price 152.72069.

```

AveragePrice.java
dft > AveragePrice.java
14
15 public class AveragePrice {
16
17     //Driver Class
18     public static void main(String[] args) throws Exception {
19         Configuration c = new Configuration();
20         String[] files = new GenericOptionsParser(c, args).getRemainingArgs();
21         Path input = new Path(files[0]);
22         Path output = new Path(files[1]);
23         Job job = new Job(c, "average price");
24         job.setJarByClass(AveragePrice.class);
25         job.setMapperClass(MapForAverage.class);
26         job.setReducerClass(ReduceForAverage.class);
27         job.setOutputKeyClass(Text.class);
28         job.setOutputValueClass(FloatWritable.class);
29
30         //get input paths from arguments
31         FileInputFormat.addInputPath(job, input);
32         FileOutputFormat.setOutputPath(job, output);
33
34         job.waitForCompletion(true);
35         System.exit(0);
36     }
37
TERMINAL    PORTS    PROBLEMS    OUTPUT    DEBUG CONSOLE
Peak Reduce Virtual memory (bytes)=2548400128
Shuffle Errors
BAD_ID=0
CONNECTION=0
IO_ERROR=0
WRONG_LENGTH=0
WRONG_MAP=0
WRONG_REDUCE=0
File Input Format Counters
Bytes Read=1633592
File Output Format Counters
Bytes Written=24
2022-07-21 23:19:29,832 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-
2022-07-21 23:19:30,759 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localhostTrusted = false, remoteHostTr
Average price 152.72069
hadoop@BigData:~/dft$

```

Average price in dollar = 152.72069

```

maxPrice.java
dft_1 > maxPrice.java
28     job.setOutputValueClass(FloatWritable.class);
29
30     //get input paths from arguments
31     FileInputFormat.addInputPath(job, input);
32     FileOutputFormat.setOutputPath(job, output);
33
34     job.waitForCompletion(true);
35     System.exit(0);
36 }
37
38
39 //Mapper
40 public static class MapFormax extends Mapper<LongWritable, Text, Text, FloatWritable> {
41
42     public void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException {
43         String line = value.toString();
44         String[] str = line.split(",");
45         Text outputKey = new Text("max price");
46         FloatWritable outputValue = new FloatWritable(Float.parseFloat(str[3].trim()));
47         // System.out.println(outputValue);
48         context.write(outputKey, outputValue);
49     }
50 }

```

TERMINAL PORTS PROBLEMS OUTPUT DEBUG CONSOLE

```

Peak Reduce Virtual memory (bytes)=2548477952
Shuffle Errors
BAD_ID=0
CONNECTION=0
IO_ERROR=0
WRONG_LENGTH=0
WRONG_MAP=0
WRONG_REDUCE=0
File Input Format Counters
  Bytes Read=1633592
File Output Format Counters
  Bytes Written=18
2022-07-21 23:20:53,937 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java cl
2022-07-21 23:20:54,831 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localhostTrusted = false, remoteHostTrusted
max price      10000.0
hadoop@BigData:~/dft_1$

```

Max price in dollar is 10000

2) Calculate the frequency of each neighbor group.

```
freqneighbor.java X
dft_2 > freqneighbor.java
18
19 private final static IntWritable one = new IntWritable(1);
20 private Text word = new Text();
21
22 public void map(Object key, Text value, Context context
23                 ) throws IOException, InterruptedException {
24     String line = value.toString();
25     String[] str = line.split(",");
26     if(str[2].contains("Staten Island") == true){
27         // str[2] = "Staten-Island";
28         StringTokenizer itr = new StringTokenizer("Staten-Island");
29         while (itr.hasMoreTokens()) {
30             word.set(itr.nextToken());
31             context.write(word, one);
32         }
33     }else{
34         StringTokenizer itr = new StringTokenizer(str[2]);
35         while (itr.hasMoreTokens()) {
36             word.set(itr.nextToken());
37             context.write(word, one);
38         }
39     }
40     // while (itr.hasMoreTokens()) {
```

TERMINAL PORTS PROBLEMS OUTPUT DEBUG CONSOLE

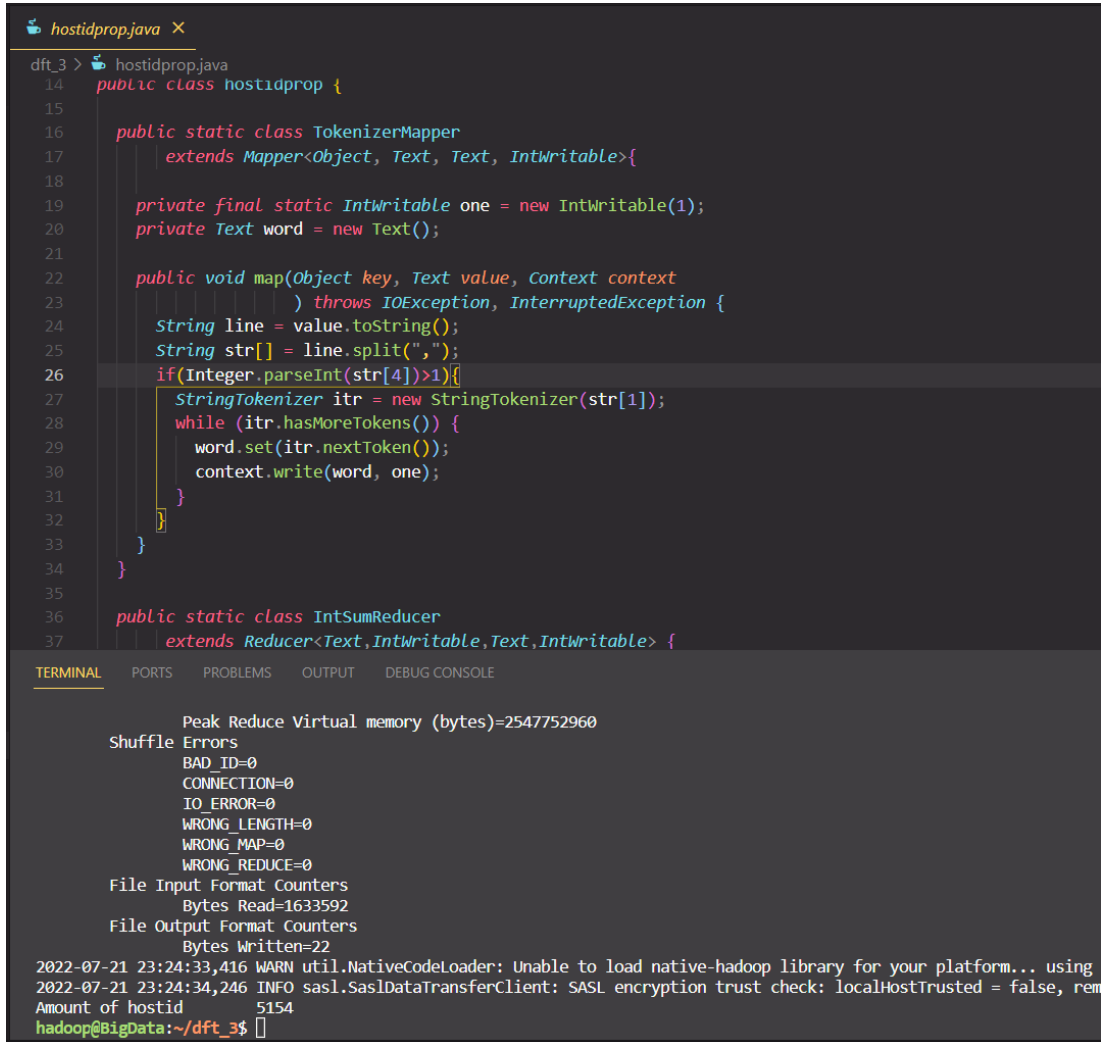
```

IO_ERROR=0
WRONG_LENGTH=0
WRONG_MAP=0
WRONG_REDUCE=0
File Input Format Counters
  Bytes Read=1633592
File Output Format Counters
  Bytes Written=72
2022-07-22 00:20:55,983 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform...
2022-07-22 00:20:56,731 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localhostTrusted = fa
Bronx      1091
Brooklyn   20104
Manhattan  21661
Queens     5666
Staten-Island 373
hadoop@BigData:~/dft_2$
```

Frequency of each neighbor group is

Bronx	1091
Brooklyn	20104
Island-Staten	373
Manhattan	21661
Queens	5666

3) How many hostid that own more than one property?



```

hostidprop.java X
dft_3 > hostidprop.java
14 public class hostidprop {
15
16     public static class TokenizerMapper
17         extends Mapper<Object, Text, Text, IntWritable>{
18
19         private final static IntWritable one = new IntWritable(1);
20         private Text word = new Text();
21
22         public void map(Object key, Text value, Context context
23             ) throws IOException, InterruptedException {
24             String line = value.toString();
25             String str[] = line.split(",");
26             if(Integer.parseInt(str[4])>1){
27                 StringTokenizer itr = new StringTokenizer(str[1]);
28                 while (itr.hasMoreTokens()) {
29                     word.set(itr.nextToken());
30                     context.write(word, one);
31                 }
32             }
33         }
34     }
35
36     public static class IntSumReducer
37         extends Reducer<Text,IntWritable,Text,IntWritable> {

```

TERMINAL

```

Peak Reduce Virtual memory (bytes)=2547752960
Shuffle Errors
BAD_ID=0
CONNECTION=0
IO_ERROR=0
WRONG_LENGTH=0
WRONG_MAP=0
WRONG_REDUCE=0
File Input Format Counters
Bytes Read=1633592
File Output Format Counters
Bytes Written=22
2022-07-21 23:24:33,416 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using
2022-07-21 23:24:34,246 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localhostTrusted = false, rem
Amount of hostid 5154
hadoop@BigData:~/dft_3$

```

Amount of hostid that own more than one property is 5154

You may verify your answer using excel counting.

```
df["price"].describe()
```

count	48895.000000
mean	152.720687
std	240.154170
min	0.000000
25%	69.000000
50%	106.000000
75%	175.000000
max	10000.000000

Name: price, dtype: float64

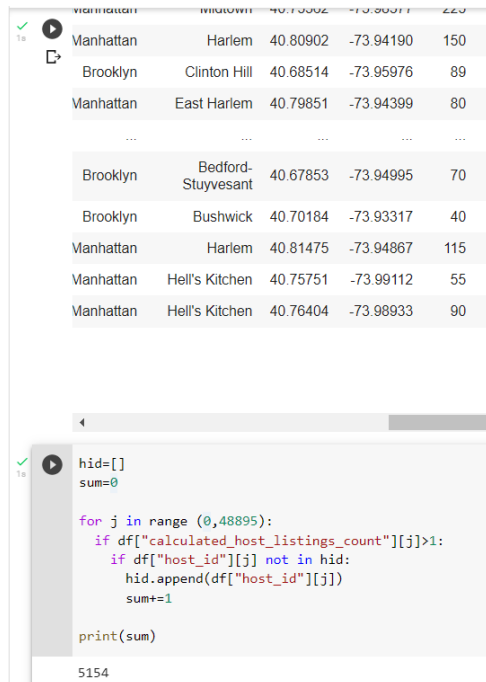
Verify max_price = 152.720687 and avg_price = 10000

```
df["neighbourhood_group"].value_counts()
```

Manhattan	21661
Brooklyn	20104
Queens	5666
Bronx	1091
Staten Island	373

Name: neighbourhood_group, dtype: int64

Verify frequency of neighbor_group



The screenshot shows a Jupyter Notebook interface. The top part displays a table with columns for neighborhood, sub-neighborhood, longitude, latitude, and a count. The bottom part shows a Python code cell that iterates through a DataFrame to count unique host IDs where the calculated host listings count is greater than 1.

Neighborhood	Sub-Neighborhood	Longitude	Latitude	Count
Manhattan	Harlem	40.80902	-73.94190	150
Brooklyn	Clinton Hill	40.68514	-73.95976	89
Manhattan	East Harlem	40.79851	-73.94399	80
...
Brooklyn	Bedford-Stuyvesant	40.67853	-73.94995	70
Brooklyn	Bushwick	40.70184	-73.93317	40
Manhattan	Harlem	40.81475	-73.94867	115
Manhattan	Hell's Kitchen	40.75751	-73.99112	55
Manhattan	Hell's Kitchen	40.76404	-73.98933	90

```
hid=[]
sum=0

for j in range(0,48895):
    if df["calculated_host_listings_count"][j]>1:
        if df["host_id"][j] not in hid:
            hid.append(df["host_id"][j])
            sum+=1

print(sum)
```

5154

Verify in Amount of host id is 5154

Submit a link in in zip containing

1. program java
2. google doc link that answers the above question by putting the screen shot on each question.
3. Names of each member. (You may do in a group of two at most).