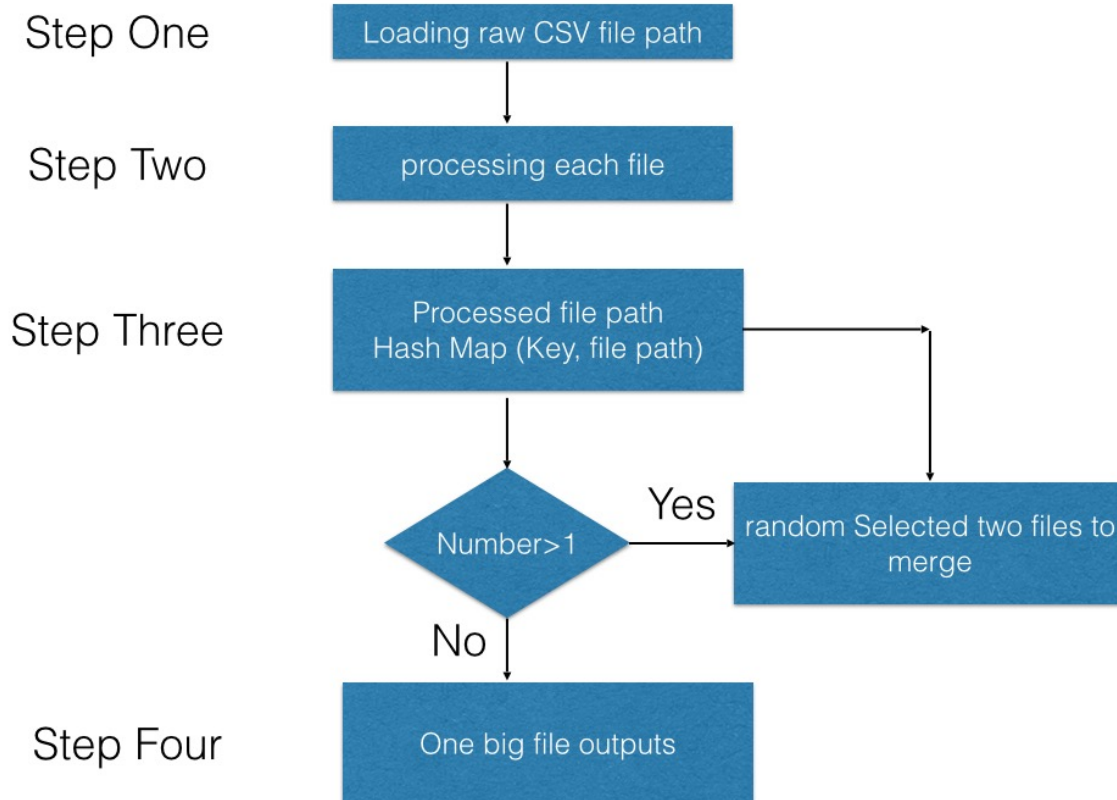


Project 2:

Data aggregation:

The code flow chart is below (No VIVO in my personal computer, sorry for that):



Step One: Loading raw CSV file path:

Input: raw CSV data file path, it may has multiply paths. In this code, we consider all CSV files are in one folder.

Outputs: lists of CSV data files

Code:

```
1 # -*- coding: utf-8 -*-
2 """
3 Created on Sun Sep 20 11:17:41 2015
4
5 @author: weizhi
6 """
7
8
9 ### loading each files
10 import glob, os
11 import numpy as np
12 import random
13 import pandas as pd
14
15 # https://docs.python.org/2/library/os.html
16 def findFilePath(path):
17     """
18     Input: raw CSV data path
19     Output: list of file path
20
21     """
22     os.chdir(path)
23     filePaths = []
24     for file in glob.glob("*.csv"):
25         filePaths.append(file)
26     return filePaths
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140 if __name__ == '__main__':
141     ### raw csv data
142     path = '/Users/weizhi/Desktop/everStringProject'
143     filePaths = findFilePath(path)
```

Step 2: Process each file

```
29 # deal with each csv file
30 class logDataAnalysis(object):
31     """
32     From raw file to proccsed outputs
33     """
34
35     def __init__(self, path, hour, fileName):
36         """
37         Path: csv files store
38         Hour: every # hours, like 2, 00-01 (00, 01), 02-03 (02,03),.....
39             like 3, 00-02, 03-05, .....
40         fileName: save file name after the processing
41         """
42
43         self.path = path
44         self.hour = hour
45         self.fileName = fileName
46
47     def readCSV(self, filePath):
48         data = pd.read_csv(filePath)
49         return data
50
51     def transfromTimeFormat(self, data):
52         timeColumn = data[data.keys()[-1]]
53         for i in range(len(timeColumn)):
54             item = timeColumn.loc[i]
55             T1 = item.split(' ')
56             T2 = T1[1].split(':')[0]
57             binTime = int(T2)/(self.hour)
58             leftBin = str(binTime*(self.hour)).zfill(2)
59             rightBin = str((binTime+1)*(self.hour)-1).zfill(2)
60             T3 = leftBin + '_' + rightBin
61             T4 = T1[0] + '_' + T3
62             timeColumn.loc[i] = T4
63         data[data.keys()[-1]] = timeColumn
64         return data
```

```

66 def generateOutputs(self,path):
67     """
68     Input: one raw CSV file path
69
70     Output: processing outputs: import pandas as pd
71           And, the file save path
72
73     """
74     data = self.transfromTimeFormat(self.readCSV(path))
75     keys = [key for key in data.keys()]
76     keys.reverse()
77     dataGroup = data.groupby(keys).groups
78     keysOuput = sorted(dataGroup.iterkeys()) # keep the keys sorted rather than hashing
79     outputs = pd.DataFrame(columns = ['period','content_id','uid','count'])
80     count = 0
81     for key in keysOuput:
82         curr = list(key) # write to each columns to outputs.csv
83         curr.append(len(dataGroup[key])) # get the count of keys from groupby
84         outputs.loc[count] = curr
85         count +=1
86     saveFile = self.createSavePath() + '/'+'Output_'+ str(self.hour) + 'hours_'+ self.fileName
87     # print savePath
88     # saveFile = savePath
89     outputs.to_csv(saveFile,index=False)
90     return outputs,saveFile
91
92 def createSavePath(self):
93     savePath = self.path + '/' + 'Outputs'
94     try:
95         os.makedirs(savePath)
96     except OSError:
97         pass
98     return savePath

```

```

144 #%% generate the raw data (CSV) to each CSV outputs formate
145 outputIndex = {} # HashMap, file key: filePath. filePath has the processed file path
146 for index in range(len(filePaths)):
147
148     Obj = logDataAnalysis(path,2,filePaths[index])
149     data = Obj.readCSV(filePaths[0])
150
151     #data.to_csv(filePaths[0],index=False)
152     outputs,savePath = Obj.generateOutputs(filePaths[0])
153     outputIndex[str(index)] = savePath
154     # savePath = Obj.createSavePath()

```

Step 3: Merge each processed files and generate the output


```
99 """ merge the files """
100 class mergeFile():
101     def __init__(self,path):
102         """
103         Path, the raw CSV file path
104         """
105         self.path = path
106     def funct(self,df):
107         """
108         Input: dataframe
109         Output: dataframe, data aggregatoin update in count
110         """
111         df['count'] = df['count'].sum()
112         return df
113     def mergerResult(self,fileOne,fileTwo):
114         """
115         Input: fileone, fileTwo: two processing files
116         Output: merge outputs
117         """
118         fileCombine = pd.concat([fileOne,fileTwo])
119         fileCombine = fileCombine.reset_index()
120         column = list(fileOne)
121         result = fileCombine.groupby(column[:3]).apply(self.funct)
122         return result[column] # get the merge files
123     def savePath(self,mergeResult,name):
124         """
125         input: merge result and name to save outputs
126         output: file save path
127         """
128         savePath = self.path + '/' + 'Outputs'
129         try:
130             os.makedirs(savePath)
131         except OSError:
132             pass
133         savePath = savePath + '/' + name
134         mergeResult.to_csv(savePath,index = False)
135         return savePath
136
137 """ merge the output from the index, we assume the memory can hold all outputs
138 # merge each two files randomly, from n to n/2, then merge again, n/2 to n/4, .... until to get big files
139 """
140 merge = mergeFile(path)
141 while(len(outputIndex.keys())>1):
142     key1 = random.choice(outputIndex.keys()) # random select key
143     fileOne = pd.read_csv(outputIndex[key1]) # read the file
144     os.remove(outputIndex[key1]) # delete the file
145     outputIndex.pop(key1,None) # delete the key
146
147     key2 = random.choice(outputIndex.keys()) # random select key
148     fileTwo = pd.read_csv(outputIndex[key2]) # read the file
149     os.remove(outputIndex[key2]) # delete the file
150     outputIndex.pop(key2,None) # delete the key
151
152     # merge
153     result = merge.mergerResult(fileOne,fileTwo)
154     # update the key
155     keyNew = key1 + '_' + key2
156     print keyNew
157     savePath = merge.savePath(result,keyNew)
158     outputIndex[keyNew] = savePath
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
```

I have duplicated your attached seven times, and run the code above, I have got the the merge files.

Input:

Name	Date Modified	Size	Kind
 mapreduce_question_data copy 2.csv	Today, 4:37 PM	3 KB	comm...values
 mapreduce_question_data copy 3.csv	Yesterday, 2:56 PM	3 KB	comm...values
 mapreduce_question_data copy 4.csv	Yesterday, 2:56 PM	3 KB	comm...values
 mapreduce_question_data copy 5.csv	Yesterday, 2:56 PM	3 KB	comm...values
 mapreduce_question_data copy 6.csv	Yesterday, 2:56 PM	3 KB	comm...values
 mapreduce_question_data copy 7.csv	Yesterday, 2:56 PM	3 KB	comm...values
 mapreduce_question_data copy.csv	Yesterday, 2:56 PM	3 KB	comm...values

Outputs:

Name	Date Modified	Size	Kind
 1_0_6_2_5_3_4.csv	Today, 10:12 PM	8 KB	comm...values

I have verified it one file (Right) and duplicate 7 times (Left). It show me correct answer:

One file

Duplicate 7 times

	period	content	uid	count	
2	2015-02-01_00_01	content	uid_1	3	
3	2015-02-01_00_01	content	uid_2	1	
4	2015-02-01_00_01	content	uid_3	2	
5	2015-02-01_00_01	content	uid_1	3	
6	2015-02-01_00_01	content	uid_2	1	
7	2015-02-01_00_01	content	uid_3	2	
8	2015-02-01_00_01	content	uid_1	3	
9	2015-02-01_00_01	content	uid_2	1	
10	2015-02-01_00_01	content	uid_3	2	
11	2015-02-01_02_03	content	uid_1	6	
12	2015-02-01_02_03	content	uid_2	2	
13	2015-02-01_02_03	content	uid_3	4	
14	2015-02-01_02_03	content	uid_1	6	
15	2015-02-01_02_03	content	uid_2	2	
16	2015-02-01_02_03	content	uid_3	4	
17	2015-02-01_02_03	content	uid_1	6	
18	2015-02-01_02_03	content	uid_2	2	
19	2015-02-01_02_03	content	uid_3	4	
20	2015-02-01_04_05	content	uid_1	3	
21	2015-02-01_04_05	content	uid_2	1	
22	2015-02-01_04_05	content	uid_3	2	
23	2015-02-01_04_05	content	uid_1	3	
24	2015-02-01_04_05	content	uid_2	1	
25	2015-02-01_04_05	content	uid_3	2	
26	2015-02-01_04_05	content	uid_1	3	
27	2015-02-01_04_05	content	uid_2	1	
28	2015-02-01_04_05	content	uid_3	2	
29	2015-02-01_12_13	content	uid_3	2	

period	content	uid	count	
2015-02-01_00_01	content	uid_1	21	
2015-02-01_00_01	content	uid_2	7	
2015-02-01_00_01	content	uid_3	14	
2015-02-01_00_01	content	uid_1	21	
2015-02-01_00_01	content	uid_2	7	
2015-02-01_00_01	content	uid_3	14	
2015-02-01_00_01	content	uid_1	21	
2015-02-01_00_01	content	uid_2	7	
2015-02-01_00_01	content	uid_3	14	
2015-02-01_02_03	content	uid_1	42	
2015-02-01_02_03	content	uid_2	14	
2015-02-01_02_03	content	uid_3	28	
2015-02-01_02_03	content	uid_1	42	
2015-02-01_02_03	content	uid_2	14	
2015-02-01_02_03	content	uid_3	28	
2015-02-01_02_03	content	uid_1	42	
2015-02-01_02_03	content	uid_2	14	
2015-02-01_02_03	content	uid_3	28	
2015-02-01_04_05	content	uid_1	21	
2015-02-01_04_05	content	uid_2	7	
2015-02-01_04_05	content	uid_3	14	
2015-02-01_04_05	content	uid_1	21	
2015-02-01_04_05	content	uid_2	7	
2015-02-01_04_05	content	uid_3	14	
2015-02-01_04_05	content	uid_1	21	
2015-02-01_04_05	content	uid_2	7	
2015-02-01_04_05	content	uid_3	14	
2015-02-01_12_13	content	uid_3	14	