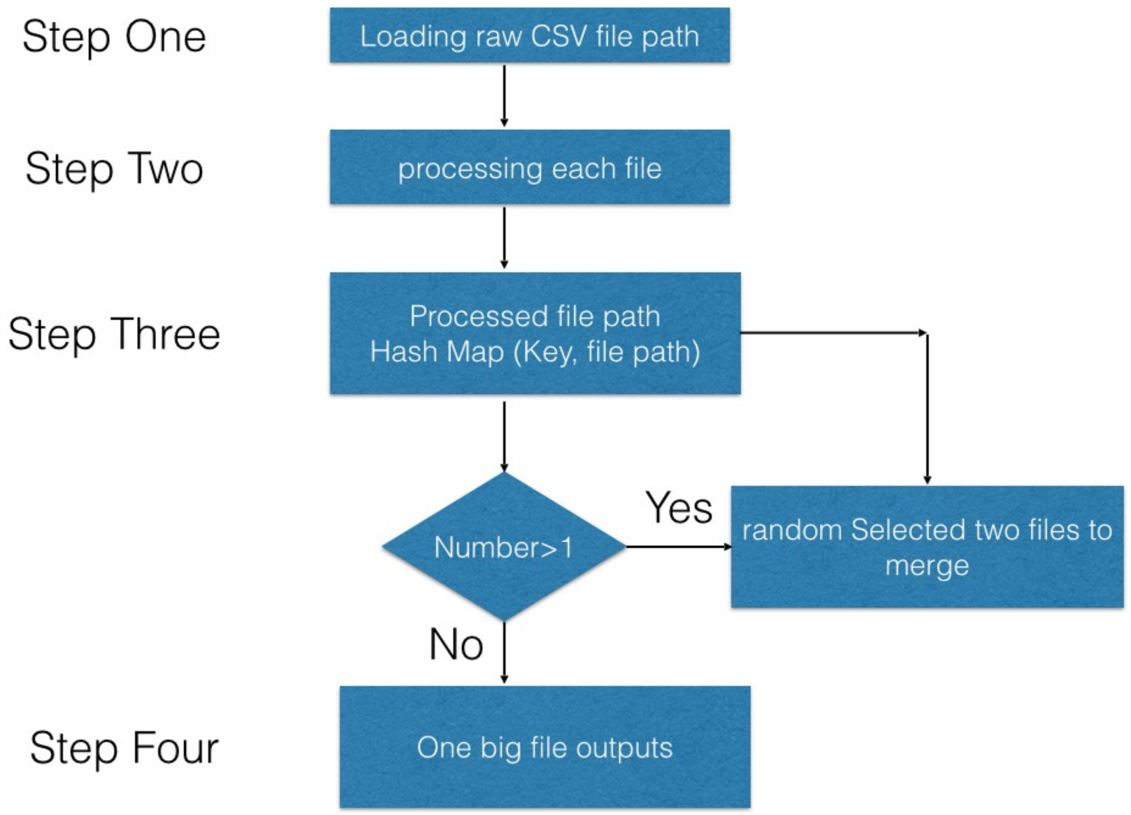
Project 2:

Data aggregation:

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

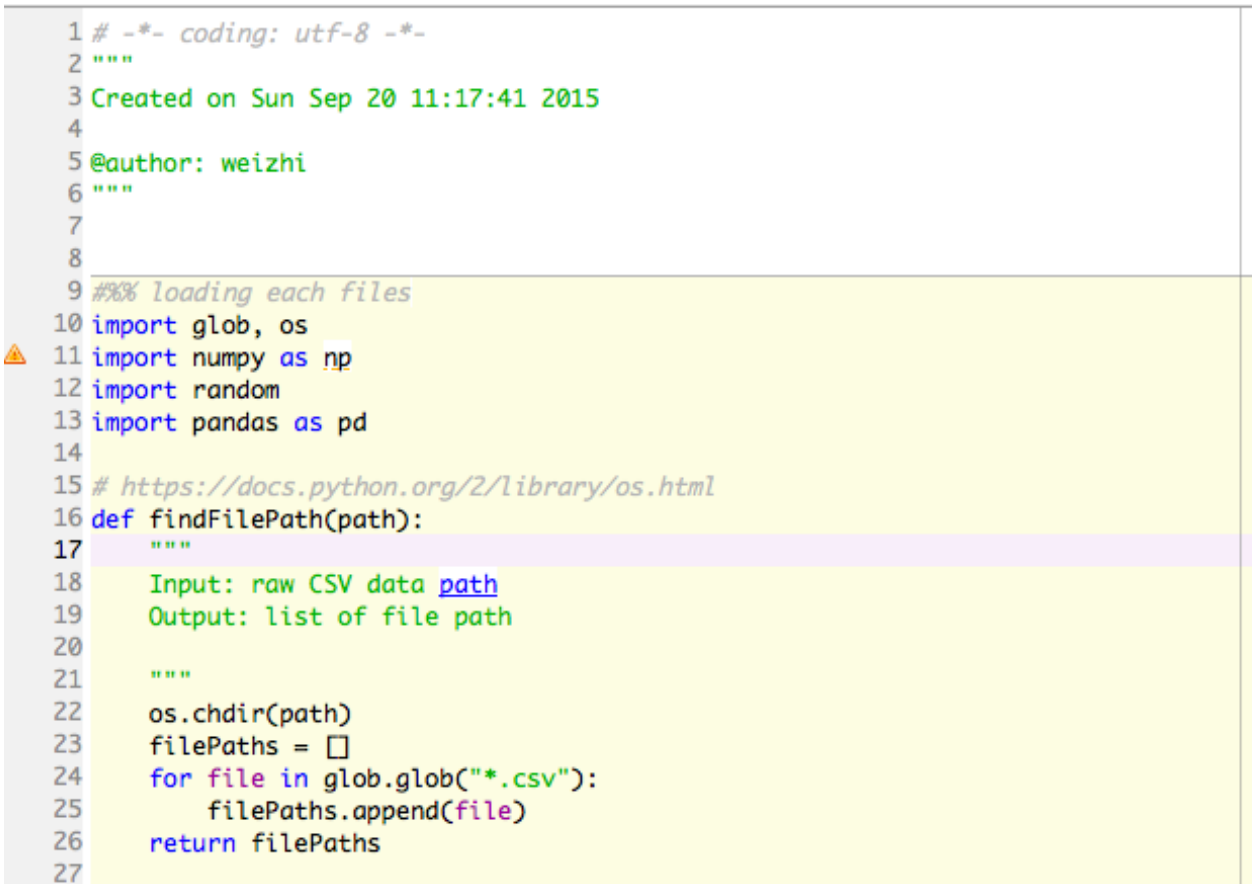


Step One: Loading raw CSV file path:

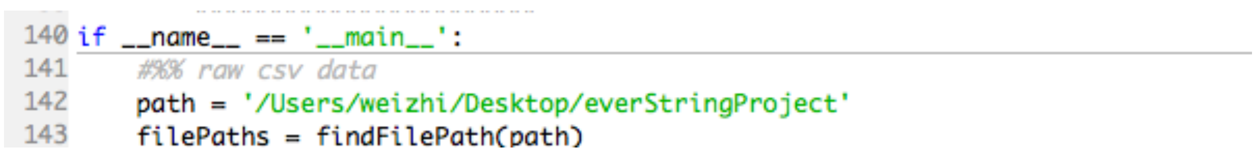
Input: raw CSV data file path, it may has multiply paths. In this code, we consider all CSV files are in one folder.

Outputs: lists of CSV data files

Code:

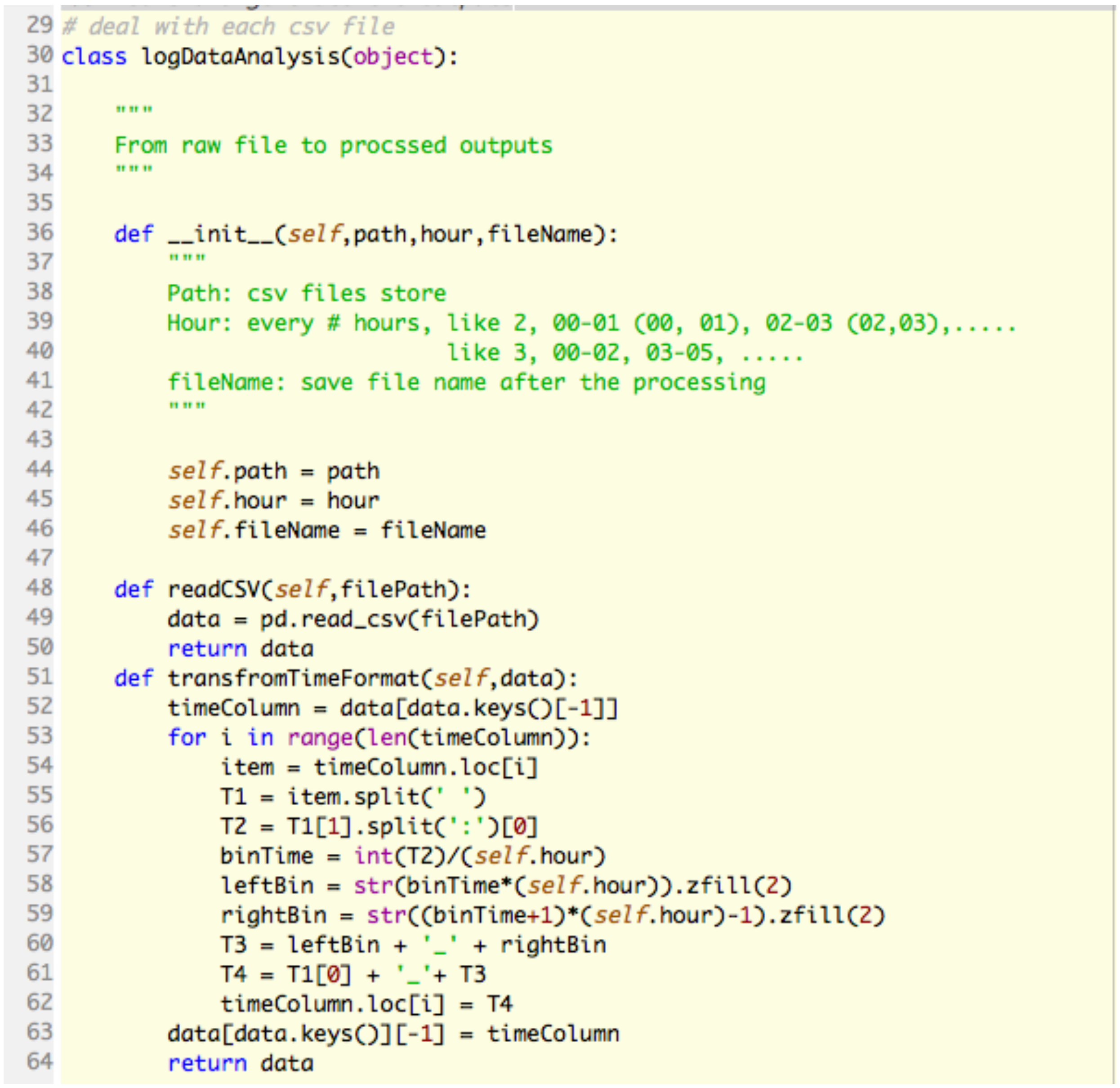


```python
# -*- coding: utf-8 -*-
"""
Created on Sun Sep 20 11:17:41 2015

@author: weizhi
"""


#%% loading each files
import glob, os
import numpy as np
import random
import pandas as pd

# https://docs.python.org/2/library/os.html
def findFilePath(path):
    """
    Input: raw CSV data path
    Output: list of file path

    """
    os.chdir(path)
    filePaths = []
    for file in glob.glob("*.csv"):
        filePaths.append(file)
    return filePaths

```

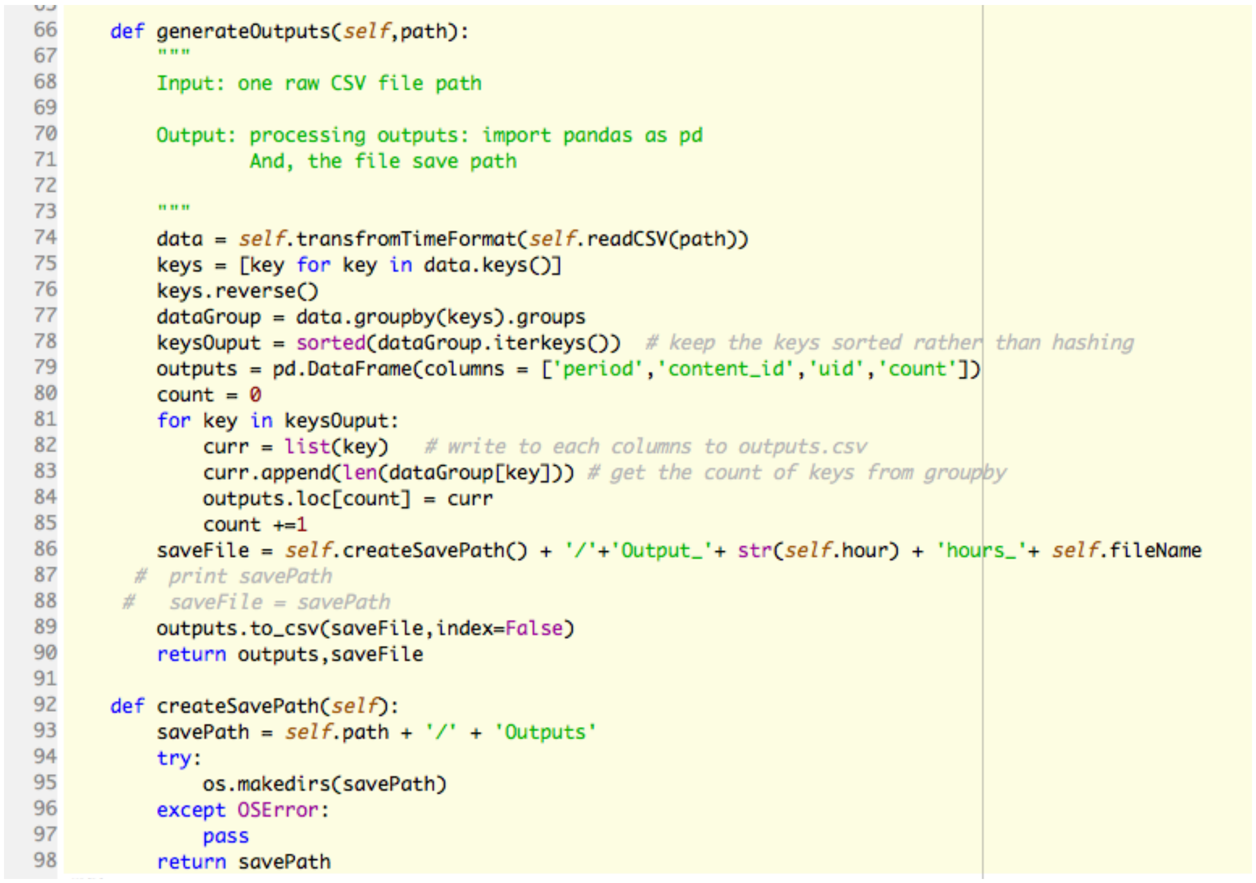


```python
if __name__ == '__main__':
    #%% raw csv data
    path = '/Users/weizhi/Desktop/everStringProject'
    filePaths = findFilePath(path)
```

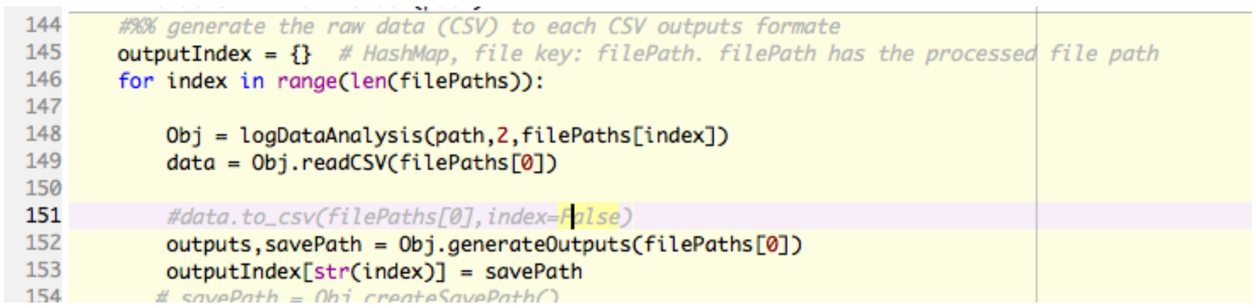
Step 2: Process each file



```python
# deal with each csv file
class logDataAnalysis(object):

    """
    From raw file to procssed outputs
    """

    def __init__(self,path,hour,fileName):
        """
        Path: csv files store
        Hour: every # hours, like 2, 00-01 (00, 01), 02-03 (02,03),.....
                             like 3, 00-02, 03-05, .....
        fileName: save file name after the processing
        """

        self.path = path
        self.hour = hour
        self.fileName = fileName

    def readCSV(self,filePath):
        data = pd.read_csv(filePath)
        return data
    def transfromTimeFormat(self,data):
        timeColumn = data[data.keys()[-1]]
        for i in range(len(timeColumn)):
            item = timeColumn.loc[i]
            T1 = item.split(' ')
            T2 = T1[1].split(':')[0]
            binTime = int(T2)/(self.hour)
            leftBin = str(binTime*(self.hour)).zfill(2)
            rightBin = str((binTime+1)*(self.hour)-1).zfill(2)
            T3 = leftBin + '_' + rightBin
            T4 = T1[0] + '_'+ T3
            timeColumn.loc[i] = T4
        data[data.keys()][-1] = timeColumn
        return data
```

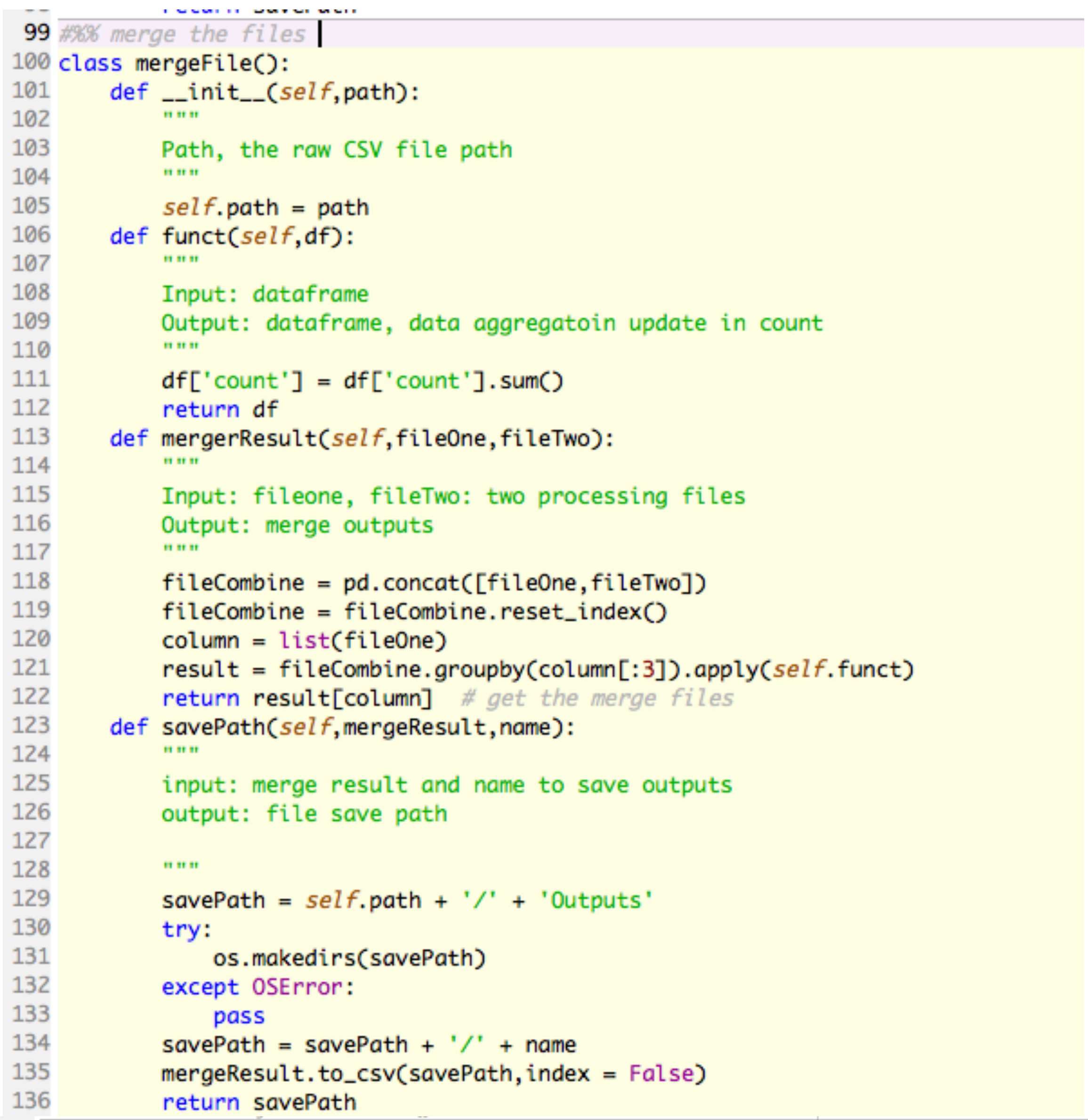


```python
    def generateOutputs(self,path):
        """
        Input: one raw CSV file path

        Output: processing outputs: import pandas as pd
                And, the file save path

        """
        data = self.transfromTimeFormat(self.readCSV(path))
        keys = [key for key in data.keys()]
        keys.reverse()
        dataGroup = data.groupby(keys).groups
        keysOuput = sorted(dataGroup.iterkeys())  # keep the keys sorted rather than hashing
        outputs = pd.DataFrame(columns = ['period','content_id','uid','count'])
        count = 0
        for key in keysOuput:
            curr = list(key)   # write to each columns to outputs.csv
            curr.append(len(dataGroup[key])) # get the count of keys from groupby
            outputs.loc[count] = curr
            count +=1
        saveFile = self.createSavePath() + '/'+'Output_'+ str(self.hour) + 'hours_'+ self.fileName
      #  print savePath
     #   saveFile = savePath
        outputs.to_csv(saveFile,index=False)
        return outputs,saveFile

    def createSavePath(self):
        savePath = self.path + '/' + 'Outputs'
        try:
            os.makedirs(savePath)
        except OSError:
            pass
        return savePath
```

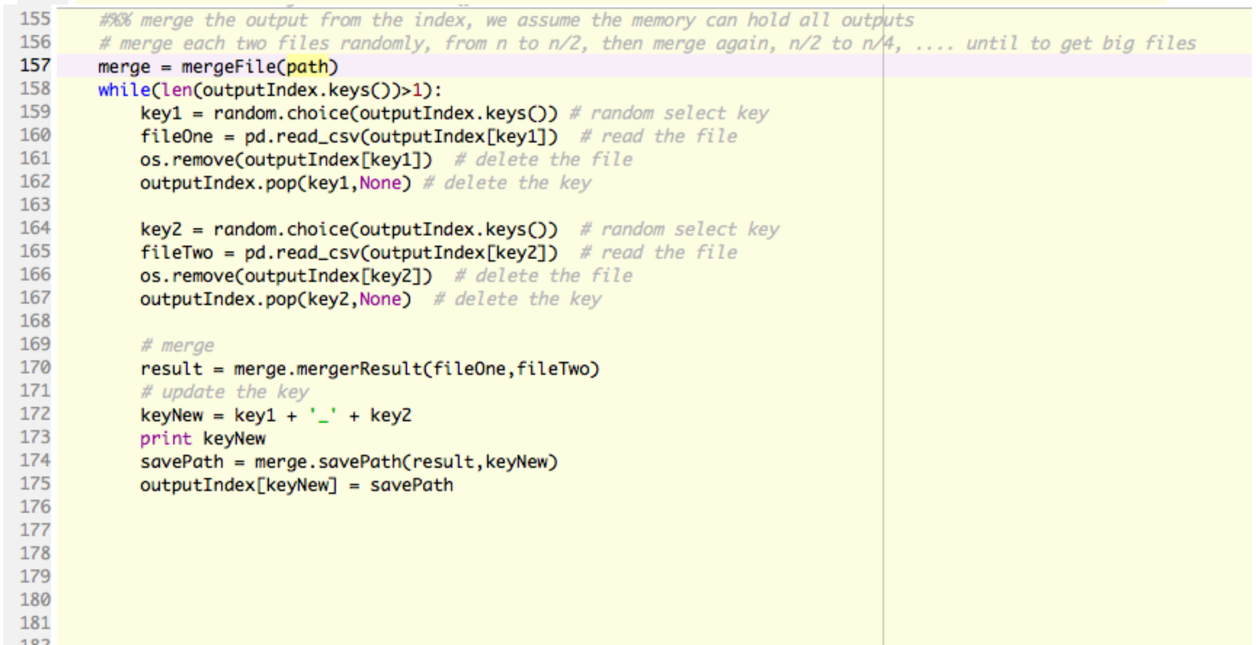


```python
#%% generate the raw data (CSV) to each CSV outputs formate
outputIndex = {}  # HashMap, file key: filePath. filePath has the processed file path
for index in range(len(filePaths)):

    Obj = logDataAnalysis(path,2,filePaths[index])
    data = Obj.readCSV(filePaths[0])

    #data.to_csv(filePaths[0],index=False)
    outputs,savePath = Obj.generateOutputs(filePaths[0])
    outputIndex[str(index)] = savePath
   # savePath = Obj.createSavePath()
```

Step 3: Merge each processed files and generate the output



```python
#%% merge the files
class mergeFile():
    def __init__(self,path):
        """
        Path, the raw CSV file path
        """
        self.path = path
    def funct(self,df):
        """
        Input: dataframe
        Output: dataframe, data aggregatoin update in count
        """
        df['count'] = df['count'].sum()
        return df
    def mergerResult(self,fileOne,fileTwo):
        """
        Input: fileone, fileTwo: two processing files
        Output: merge outputs
        """
        fileCombine = pd.concat([fileOne,fileTwo])
        fileCombine = fileCombine.reset_index()
        column = list(fileOne)
        result = fileCombine.groupby(column[:3]).apply(self.funct)
        return result[column]  # get the merge files
    def savePath(self,mergeResult,name):
        """
        input: merge result and name to save outputs
        output: file save path

        """
        savePath = self.path + '/' + 'Outputs'
        try:
            os.makedirs(savePath)
        except OSError:
            pass
        savePath = savePath + '/' + name
        mergeResult.to_csv(savePath,index = False)
        return savePath
```

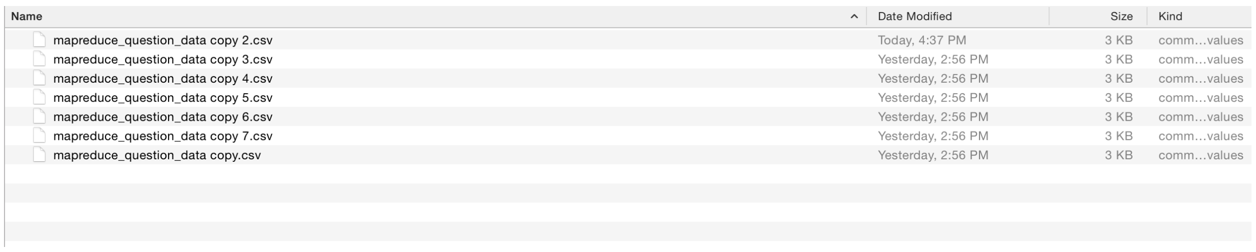


```python
    #%% merge the output from the index, we assume the memory can hold all outputs
    # merge each two files randomly, from n to n/2, then merge again, n/2 to n/4, .... until to get big files
    merge = mergeFile(path)
    while(len(outputIndex.keys())>1):
        key1 = random.choice(outputIndex.keys()) # random select key
        fileOne = pd.read_csv(outputIndex[key1])  # read the file
        os.remove(outputIndex[key1])  # delete the file
        outputIndex.pop(key1,None) # delete the key

        key2 = random.choice(outputIndex.keys())  # random select key
        fileTwo = pd.read_csv(outputIndex[key2])  # read the file
        os.remove(outputIndex[key2])  # delete the file
        outputIndex.pop(key2,None)  # delete the key

        # merge
        result = merge.mergerResult(fileOne,fileTwo)
        # update the key
        keyNew = key1 + '_' + key2
        print keyNew
        savePath = merge.savePath(result,keyNew)
        outputIndex[keyNew] = savePath
```

Conclusion:

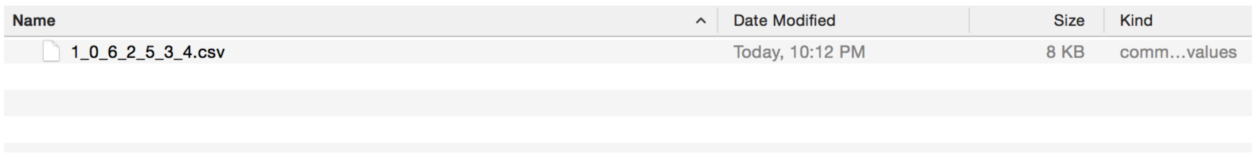
I have duplicated your attached seven times, and run the code above, I have got the the merge files.

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

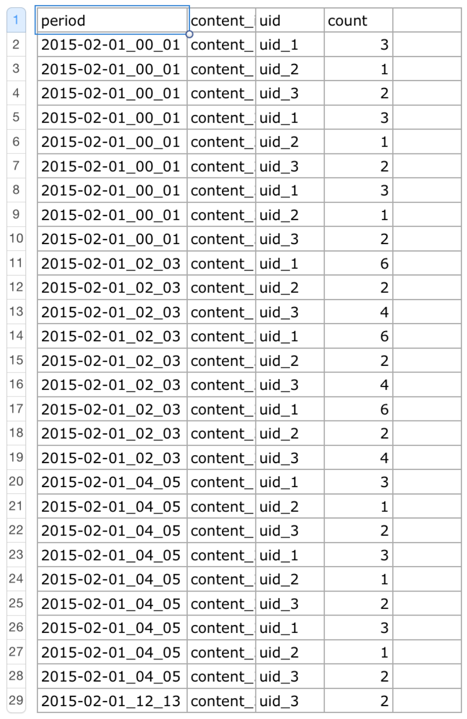
Outputs:



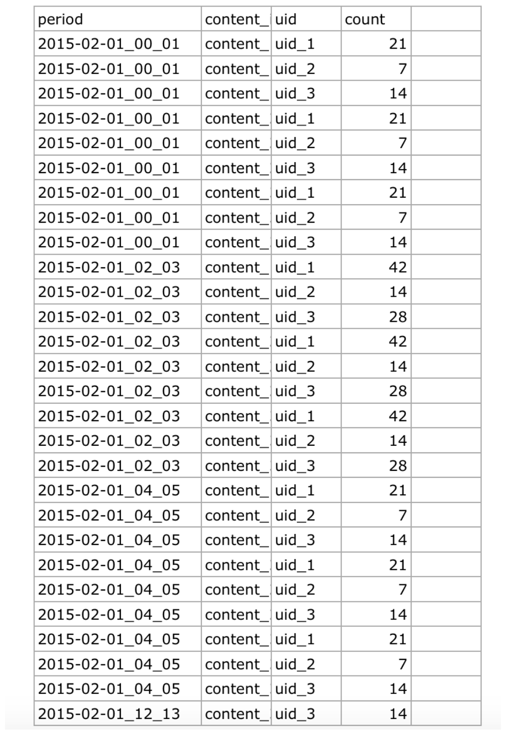
Name	Date Modified	Size	Kind
1_0_6_2_5_3_4.csv	Today, 10:12 PM	8 KB	comm…values

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
2015-02-01_00_01	content_	uid_1	3
2015-02-01_00_01	content_	uid_2	1
2015-02-01_00_01	content_	uid_3	2
2015-02-01_00_01	content_	uid_1	3
2015-02-01_00_01	content_	uid_2	1
2015-02-01_00_01	content_	uid_3	2
2015-02-01_00_01	content_	uid_1	3
2015-02-01_00_01	content_	uid_2	1
2015-02-01_00_01	content_	uid_3	2
2015-02-01_02_03	content_	uid_1	6
2015-02-01_02_03	content_	uid_2	2
2015-02-01_02_03	content_	uid_3	4
2015-02-01_02_03	content_	uid_1	6
2015-02-01_02_03	content_	uid_2	2
2015-02-01_02_03	content_	uid_3	4
2015-02-01_02_03	content_	uid_1	6
2015-02-01_02_03	content_	uid_2	2
2015-02-01_02_03	content_	uid_3	4
2015-02-01_04_05	content_	uid_1	3
2015-02-01_04_05	content_	uid_2	1
2015-02-01_04_05	content_	uid_3	2
2015-02-01_04_05	content_	uid_1	3
2015-02-01_04_05	content_	uid_2	1
2015-02-01_04_05	content_	uid_3	2
2015-02-01_04_05	content_	uid_1	3
2015-02-01_04_05	content_	uid_2	1
2015-02-01_04_05	content_	uid_3	2
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