4/20/20

Dominic Zecchino

I pledge my honor that I have abided by the Stevens Honor System.

I would like to preface this report by saying that I am grateful for this class and I have learned a great deal whilst watching the lectures and completing the assignments. That being said, I would like to mention that I have spent a great deal of time and put quite a large amount of effort into this assignment. Unfortunately, this homework proved quite difficult to me and it is not going to be complete by the deadline despite my efforts. I am not asking for any leniency I am simply saying that I hope whatever grade I am given is somewhat reflective of the amount of time I have put into the assignment, which to be quite frank was an extremely large amount of time. I would like to extend my gratitude to both the TA and the professor for all of the effort you have put forth for the students.

Part 1: Card Transaction Data

This portion of the assignment was dedicated to projecting spending habits among different individuals, which was detailed within the .csv file that was provided. The first section of this part was to determine the total amount spent of all individuals. This proved to be rather difficult due to the fact that while the majority of the detailed amounts were numbers, certain cells within the .csv file contained unwanted symbols and/or letters. Due to the fact that it is impossible to add a cell with a numeric value with a cell that contained a dollar sign and/or some other symbol, for example one of the cells contained ($29.99), the data had to first be cleaned. Cleaning the ‘Amount’ column was completed by utilizing a mapping function that called a function that I wrote labeled ‘clean’. The clean function analyzed the contents of each cell, taking only the numeric values and the decimal point, and converted everything to a string (if it

wasn't already). Those strings were then converted to floating point variables so that their values can be held, which included the decimal places. A simple addition equation was then created that took the value from each row of the 'Amount' column, and added them all together. The process was quite similar for the other three sections which looked to see the spending at specific companies. The difference, however, was that it was required that we make sure to only add the values that correspond to a respective company. To accomplish this, two different labels were set to the string in the 'Vendor' column that corresponded with the vendor that was desired (WW Grainger and WM Supercenter), and one label was set to the string in the 'Merchant Category Code (MCC)' column that corresponded with the MCC that was desired (Grocery Stores). Every vendor was then compared with the both of the vendor labels, and if there was a match, then the respective 'Amount' value for that row would be added to the total of the respective vendor. The same thought was applied for the MCC section, where every MCC that had Grocery Store then had their respective amounts totaled.

Part 2:

This part proved even more challenging than the last, as I will detail the specific area(s) in which I had trouble, which prevented me from moving further within the assignment. After quite a lot of manipulation with my code, I got some minor bugs fixed, but ultimately I am unsure of how to fix my underlying issues. Currently I have spent countless hours attempting to fix my dataset for the second part of this question but my issue arises when looking at the column length. I was looking to filter out all unnecessary columns, which were those that only had characters(AKA non-numeric values). I had tested a few different methods, but ultimately it all boiled down to the same issue. I set up a for loop with the goal of setting my frame1 = frame1.drop(frame1.columns[i]. The idea was to test all the columns based on their

data types and make a fully new dataset that includes everything from the original dataset except the columns that are not of type int or float. The problem was that as I set the new frame1 after each iteration of the loop, the length of the loop would decrease, which means that no matter what method I employed, the index would eventually be out of range, resulting in an error. Now on to how this affected the rest of my assignment. Considering I was unable to come up with a method to get the new dataset with the dropped values, that means that the third section was out of the question. An error would result every time if I were to attempt to replace all None and NaN values considering many of the columns contain only letters. This also throws off the ability to output my own .csv file simply based off of the fact that I was unable to drop the desired columns in section 2. Given more time, although I am not requesting an extension, I believe I would’ve been able to figure out the remainder of the assignment. I also feel I have learned a lot about Pandas and its uses despite not completing the assignment.

This is my code for the first part of Homework 4 Part 1

```
from pandas import Series, DataFrame
import pandas as pd
import numpy as np



#class CSVFile:


data = pd.read_csv (r'/home/dominic/Documents/Python/FE
520/HW4/Homework4_Dataset/res_purchase_2014.csv')
table = pd.DataFrame(data, columns= ['Year-Month','Agency Number','Agency
Name','Cardholder Last Name','Cardholder First
Initial','Description','Amount','Vendor','Transaction Date','Posted Date','Merchant Category Code
(MCC)'])


def clean(s):
    res = ""
    for i in str(s):
        if (i is "-"):
            res += i
        if (i is "."):
            res += i
        if (i.isdigit()):
            res += i
```

```
    return res
table['Amount'] = table['Amount'].map(clean)
#table['Amount'] = table['Amount'].str.extract('^(\d*\.*\d*)', expand=False)                #looking to
map instead
#table['Amount'] = table['Amount']str.replace(r"[a-zA-Z],'')
#table['Amount'] = table['Amount'].map(lambda x: ''.join([i for i in x if i.isdigit()]))
#table['Amount'] = table['Amount'].astype(float)

#print(table['Amount'])
#def WWGrainger():

    #print (table)                        #This shows that I am properly reading the data
    #print (table.iloc[0]['Description'])          #Testing to see if the columns are properly set
    #print(len(table))                    #Shows me that I do not need to search for the number of rows
specifically, and that it is automatic

    #frame = pd.DataFrame(table)
    #print(frame)

    #print(table.iloc[0]['Vendor'])        #Testing to retrieve specific values from a given row of a
data set

    #print(table.iloc[50]['Vendor'])       #Index 50 is the first known location of WW Grainer
```

```
totalSpent = 0

SpentWWG = 0

SpentGS = 0

SpentWMS = 0

'''

temp = ''

sum = 0.0

for i in range (len(table)):

  amount = table.iloc[i]['Amount']

  count = 0

  print(i)

  for j in range (len(table.iloc[i]['Amount'])):

    temp = 0

    if(amount[j] != 1 or amount[j] != 2 or amount[j] != 3 or amount[j] != 4 or amount[j] != 5 or

amount[j] != 6 or amount[j] != 7 or amount[j] != 8 or amount[j] != 9 or amount[j] != 0 or i != '.'):

        temp = 0

    else:

        temp = float(amount[i])

    sum+= temp



print(sum)


'''
```

```
#x=table.iloc[50]['Vendor']
#y=table.iloc[265463]['Vendor']
#z= table.iloc[265543]['Merchant Category Code (MCC)']
#print(x,y,z)
x = "WW GRAINGER"
y = "WM SUPERCENTER"
z = "GROCERY STORES,AND SUPERMARKETS"
print("Please be patient, the code isn't optimized so it takes a few minutes.")
for i in range (len(table)):
#       '''======================The code below is for total
spending========================'''
    #print(i)
    #table.iloc[i]['Amount'] = table.iloc[i]['Amount'].str.extract('(\d+)', expand=False)
#determined there is probably a better way to do this
    amount = float(table.iloc[i]['Amount'])
    #amount = round(amount, 2)                    #not useful anymore
    #print('The amount is===== ', amount)
    totalSpent = round(totalSpent + amount, 2)



#    '''======================The code below is for WW
Grainger========================'''
                                    #Used to test the total amount
```

```
    #if (table.iloc[i]['Vendor'] == "WW GRAINER")                    #First attempt at getting the
correct amounts
    if (x in table.iloc[i]['Vendor']):          #For now i will resort to this method until I can come up
with a better one
        value = table.iloc[i]['Amount']
        #print(value)
        #value = int(value)                                  #This is the wrong typecast, it should be
float
        SpentWWG = SpentWWG + float(value)                      #This should be the proper
formula
        SpentWWG = round(SpentWWG, 2)                           #Without this rounding the
answer doesnt come out to 2 decimal places for some reason
        #print(f'The concatenated spending value is {SpentWWG:.2f}')
        #SpentWWG = {SpentWWG:.2f}                              #not the right method
        #print('The current value being examined is: ',value)          #Print statement used to test
        #print('The total amount spent at WW GRAINGER is: ', SpentWWG)              #Print
statement used to test

#print('''=================================================================
=================================================================
====================
        #
=================================================================
```

```
==========================================================================
===============
      #
==========================================================================
==========================================================================
===============
      #    ''')
   #print(i)                                    #Print statement used to test


#   ====================== The code below is for WM Supercenter
====================
   if (y in table.iloc[i]['Vendor']):          #For now i will resort to this method until I can come up
with a better one
      #print(i)
      value = table.iloc[i]['Amount']
     #value = int(value)                                  #This is the wrong typecast, it should be float
      SpentWMS = SpentWMS + float(value)                        #This should be the proper
formula
      SpentWMS = round(SpentWMS, 2)



#   ====================== The code below is for Grocery Stores
====================
```

```
    if (z in table.iloc[i]['Merchant Category Code (MCC)']):          #For now i will resort to this
method until I can come up with a better one
        #print(i)
        value = table.iloc[i]['Amount']
      #value = int(value)                                  #This is the wrong typecast, it should be float
        SpentGS = SpentGS + float(value)                        #This should be the proper formula
        SpentGS = round(SpentGS, 2)
#   ====================== The code below outputs our desired values
====================
print('The total amount spent overall is: ',totalSpent)
print('The amount spent at WW GRAINGER is: ', SpentWWG)
print('The amount spent at WM Supercenter is: ', SpentWMS)
print('The amount spent at the Grocery Stores is: ', SpentGS)



#print(table.iloc[265446]['Vendor'])                          #used for testing
```

This is my code for Homework 4 Part 2

```
from pandas import Series, DataFrame, ExcelWriter, ExcelFile
import pandas as pd
import numpy as np
import math

#df = pd.read_excel(open('/home/dominic/Documents/Python/FE
520/HW4/Homework4_Dataset/Energy.xlsx', 'rb'), sheetname = 'Sheet1')        #didnt work
BalanceSheet = pd.read_excel (r'/home/dominic/Documents/Python/FE
520/HW4/Homework4_Dataset/Energy.xlsx')                                    #works
Ratings = pd.read_excel (r'/home/dominic/Documents/Python/FE
520/HW4/Homework4_Dataset/EnergyRating.xlsx')                              #works
frame1  = pd.DataFrame(BalanceSheet)
frame1copy = frame1

rows = len(frame1)
column = len(frame1.columns)
temp = range(column)

#print(frame1.dtypes)

for i in temp:
    if (not(frame1.dtypes[i] == np.int64 or frame1.dtypes[i] == np.float64)):
        frame1copy = frame1.drop(frame1.columns[i], 1)
```

```
        #temp = range(len(frame1.columns))                    #don work


print(frame1copy)



#frame1 = frame1.T                    #probably not the way i should do this
#print (data)
#print (data2)


count = 0
for j in range(column):
    frame2 = frame1.columns[j]
    print(frame2)
    print()
    #=============This is for the first column that needs testing


    for i in range(rows):
        x = frame1.iloc[i][frame2]
        if(x == None or float(x) == 0.0 ):
            count += 1
            print(count)


        if ((count/rows) > 0.9 ):
            frame1 = frame1.drop(frame2, axis=1)
```

```
print(frame1)


#    for i in range(rows):
#        #x = float(frame1[i+1])
#        if(is_number(frame1[i])):#or frame1[i] == None or int(frame1[i]) == 0 ):
#            count += 1
#        if ((count/rows) > 0.9 ):
#            frame1copy.drop(frame1.columns[j])
#            frame1copy = frame1copy.drop(frame1copy.columns[j])



print('count is: ',count)
#print(len(frame1))                  #test Code
#print(frame1)                       #test Code
#print(frame1['Global Company Key'])    #test Code
#print(frame1.columns[1])             #test code
#print(len(frame1.columns))           #test code
```