Dec 2021

PLEASE NOTE THAT THERE ARE MORE THAN ONE WAY FOR THE QUESTIONS ASKED.

THE FOLLOWING IS JUST ONE POSSIBLE SOLUTION

```
import pandas as pd
import numpy as np
#import scipy.stats
import seaborn as sns
import matplotlib.pyplot as plt #  plotting library for the Python programming language
from sklearn import preprocessing #sklearn provides machine learning tools
```

Final paper DAV 2021

Q 1 Given a list of strings where strings may be duplicated

List1=['Good Morning', 'Good Evening', 'Hello', 'Good afternoon', 'Greetings', 'Good Morning', 'Nice to see you']. Do the following.

a) Use list comprehension to store unique strings with multi-words of List1 to another list named Newlist. Also, write an anonymous function to sort the given list List1 on the last character of the strings 3+3

```
List1=['Good Morning', 'Good Evening', 'Hello', 'Good afternoon', 'Greetings', 'Good Morning', 'Nice to
```

```
Newlist=[ v for v in set(List1) if len(v.split(' '))>1]
Newlist
```

```
['Good afternoon', 'Good Morning', 'Nice to see you', 'Good Evening']
```

```
List1.sort(key= lambda x: x[-1])
List1
```

```
['Good Morning',
 'Good Evening',
 'Good Morning',
 'Good afternoon',
 'Hello',
```

```
 'Greetings',
 'Nice to see you']
```

1 b: Using ‘List1’, generate the following dictionary ‘Anydict’ where key is the count of words in a string and value is the list of strings having that count.

Anydict={1:['Greetings','Hello'], 2: ['Good Morning', ' Good Evening', 'Good afternoon', 'Good Morning'], 4: ['Nice to see you']}. Create a data series ‘Ds1’ using the created dictionary ‘Anydict’ 4+2

```
Anydict={}
for i,v in enumerate(List1):
    l=len(v.split(' '))
    if l not in Anydict:
        Anydict[l]=[v]
    else:
        Anydict[l].append(v)
Anydict
s = pd.Series(Anydict)
```

```
{1: ['Hello', 'Greetings'],
 2: ['Good Morning', 'Good Evening', 'Good Morning', 'Good afternoon'],
 4: ['Nice to see you']}
```

Q 1 c c) Draw a bar plot to compare frequency of strings with equal word counts of List1 where frequency is the count of such strings. E.g. Frequency of strings with two words in the given list List1 is 4. Give proper names to both axes.(5+1.75)

Putting dictionary in series

```
ser = pd.Series(Anydict)
ser
```

```
2    [Good Morning, Good Evening, Good Morning, Goo...
1                                  [Hello, Greetings]
4                                   [Nice to see you]
dtype: object
```

converting Data series to Data frame for simplicity. But series can also be used

```
df1=pd.DataFrame(ser,columns=['list1'])
df1
```

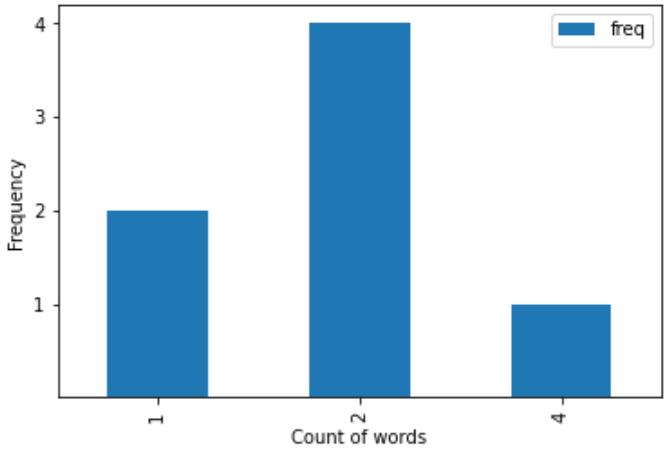
	list1
2	[Good Morning, Good Evening, Good Morning, Goo...
1	[Hello, Greetings]
4	[Nice to see you]

```
df1['freq']=df1['list1'].str.len()
df1.index.name='Numberofwords'
df1.sort_index(inplace=True)
```

```
df1
```

	list1	freq
Numberofwords		
1	[Hello, Greetings]	2
2	[Good Morning, Good Evening, Good Morning, Goo...	4
4	[Nice to see you]	1

```
ax=df1.plot(kind='bar',xlabel="Count of words",ylabel='Frequency',yticks=[1,2,3,4])
#ax.set_xlabel="Count of words"
#ax.set_ylabel='Frequency'
#plt.plot()
```




```
df1.index.values
```

```
array([1, 2, 4], dtype=int64)
```

Q 2 a) Create a new data frame SELECTED having a hierarchical index on columns “Name” and “Diet”. Then, find maximum pulse rate for each individual in the SELECTED Dataframe. (1 FOR CREATION , 2 FOR INDEXING, 4 FOR FINDING MAX PULSE)

```
from google.colab import files;
uploaded = files.upload()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
Saving examdata1.csv to examdata1.csv

```
DF1=pd.read_csv('/content/examdata1.csv')
DF1
# PLEASE NOTE THAT EXPLICIT USE OF PANDAS I.E READING VIA DATAFRAME COMMAND
# Student should have create a dataframe and not read from csv file. This statement was emnployed for ou
```

	ID	Name	diet	pulse	time	kind
0	0	A	low fat	85	40	walking
1	1	A	low fat	85	45	walking
2	2	A	no fat	88	30	running
3	3	B	no fat	90	10	walking
4	4	B	no fat	92	15	rest
5	5	B	low fat	93	30	rest
6	6	C	low fat	97	15	rest
7	7	C	low fat	97	15	rest
8	8	C	low fat	94	30	walking
9	9	D	low fat	80	10	walking
10	10	D	low fat	82	15	rest
11	11	D	low fat	83	30	rest
12	12	E	no fat	91	10	rest
13	13	E	low fat	92	15	running
14	14	E	low fat	91	30	running

Setting Hierarchical index

```
DF2=DF1.set_index(['Name','diet'],drop=True)
DF2
```

Name	diet	ID	pulse	time	kind
A	low fat	0	85	40	walking
	low fat	1	85	45	walking
	no fat	2	88	30	running
B	no fat	3	90	10	walking
	no fat	4	92	15	rest
	low fat	5	93	30	rest
C	low fat	6	97	15	rest
	low fat	7	97	15	rest
	low fat	8	94	30	walking
D	low fat	9	80	10	walking
	low fat	10	82	15	rest
	low fat	11	83	30	rest
E	no fat	12	91	10	rest
	low fat	13	92	15	running
	low fat	14	91	30	running

```
DF2.max(level=['Name'])['pulse']
```

```
Name
A    88
B    93
C    97
D    83
E    92
Name: pulse, dtype: int64
```

2 : b) Count total number of records of individuals having names ‘A’ or ‘B’ and are following low fat diet plan from the the data frame SELECTED created in part (a)

(2+4)

```
len(DF2.loc[[('B','low fat'),('A','low fat')]])
```

```
3
```

```
DF2.sort_index(level=0,ascending=False)
```

Name	diet	ID	pulse	time	kind
E	**no fat**	12	91	10	rest
	low fat	14	91	30	running
	low fat	13	92	15	running
D	**low fat**	11	83	30	rest
	low fat	10	82	15	rest
	low fat	9	80	10	walking
C	**low fat**	8	94	30	walking
	low fat	7	97	15	rest
	low fat	6	97	15	rest
B	**no fat**	4	92	15	rest
	no fat	3	90	10	walking
	low fat	5	93	30	rest
A	**no fat**	2	88	30	running
	low fat	1	85	45	walking
	low fat	0	85	40	walking

2(c). c) Using data frame EXERCISE , create a figure with two subplots and save the figure with name ‘exerciseplot.jpeg’. Set title of the figure as ‘EXERCISE’. First subplot compares the average pulse rate of individuals and the second subplot shows relationship between variables ‘pulse’ and ‘time’. Do color encoding using variable ‘kind’.

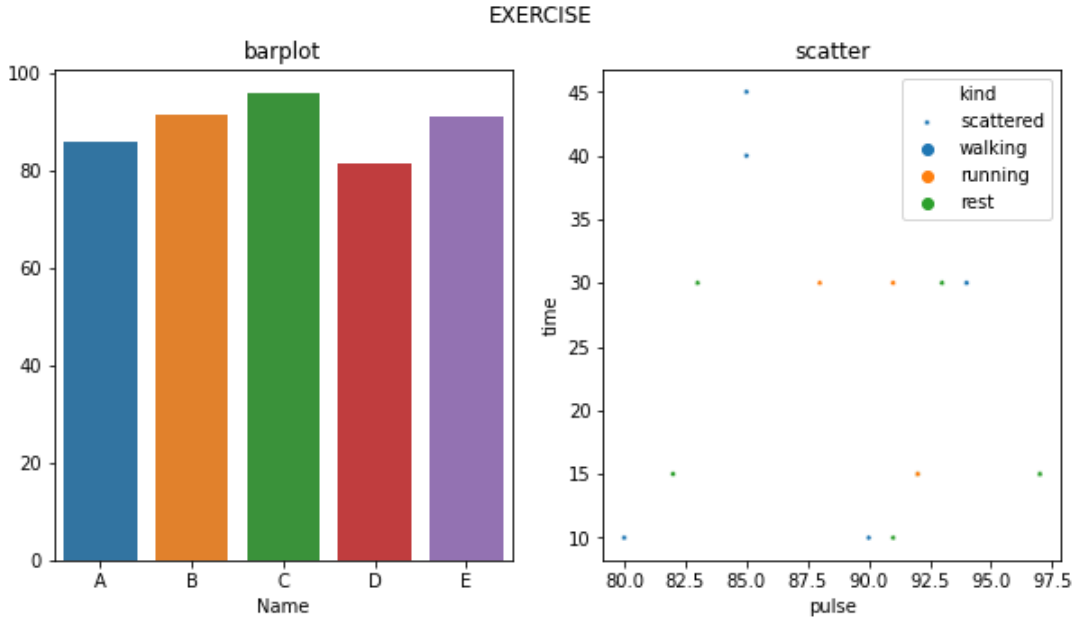
SAVING FIGURE NAME AND TITLE 2 MARKS PLOTTING SCATTER AND BAR PLOT 4.75(1 MARK SHOULD BE CONSIDERED INTO SUBPLOTS)

```
DF3=DF2.mean(level=['Name'])['pulse']
DF3
```

```
Name
A    86.000000
B    91.666667
C    96.000000
D    81.666667
```

```
E    91.333333
Name: pulse, dtype: float64
```

```
fig1 = plt.figure(facecolor='w',edgecolor='k')
fig1.set_figheight(5)
fig1.set_figwidth(10)
fig1.suptitle("EXERCISE")
ax1 = fig1.add_subplot(1, 2, 1,title='barplot')
ax2 = fig1.add_subplot(1, 2, 2,title='scatter')
sns.scatterplot(ax=ax2,x=DF2.pulse, y=DF2.time, hue=DF2.kind,marker='.',  label='scattered')
sns.barplot(ax=ax1, x=DF3.index, y=DF3.values,label='barplot')
fig1.savefig("exerciseplot.jpeg",dpi=400)
```



if sns library is not used them simple pandas plt functionality may be used

```
plt.scatter(DF2.pulse, DF2.time, marker='.', hue='kind', label='scattered')
```

```
sns.scatterplot(x='pulse', y='time',  hue='kind', data=DF2)
```

```
<AxesSubplot:xlabel='pulse', ylabel='time'>
```

Q3 A) Given the following commands to create series sr sr = pd.Series(['Madhuri','AjaySh@rma', 'R@ni', 'Radha',np.nan,'Smita','3567']) Write separate commands to compute the length of each string in the series, replace @ with 'a' in all strings in the series, count the occurrences of 'a' in each string, change the case of all letters, find all strings with pattern 'adh' in them and find all strings that end with letter 'i'. (1 MARK FOR FIRST FIVE COMMAND AND 1.75 FOR LAST COMMAND)

```
#Q3 a)
import pandas as pd
import numpy as np
sr = pd.Series(['Madhuri',' Ajay Sh@rma', 'R@ni', 'Radha',np.nan, 'Smita','3567'])
print (sr)
print (sr.str.len())
print (sr.str.replace('@','a'))
print (sr.str.count('a'))
print (sr.str.swapcase())
print (sr.str.find('adh'))
print (sr.str.endswith('i'))
```

Q3B)Create a DataFrame of 7 rows and 7 columns containing random integers in the range of 1 to 100. Compute the correlation of each row with the preceding row.(3+3)

Double-click (or enter) to edit

```
#Q3b
df = pd.DataFrame(np.random.randint(1,100,size=(7,7)))
print (df)
df.rolling(2).corr(pairwise = 'True')
```

Q3C)Write Numpy code to generate a random list of 100 integers (range of 55 to 150) and identify the index of the largest element and smallest element. Change this list into a 10 x 10 matrix and replace all diagonal elements with 1. (2+2+2.75)

```
#Q3c
X = np.random.randint(55,150,size=100)
print(X)
#display(np.argmin(X)) # will display only first instance
display(np.where(X == np.amin(X))) # will display all such indexes
display(np.where(X == np.amax(X)))
arr = X.reshape(10,10)
display(arr)
np.fill_diagonal(arr,1)
print(arr)
```

Q4 Using the data frame EXERCISE provided in Q2 , attempt the following questions a) What is a map function?

```
#Map() returns map object (iteratorof result after applying a given function to each item of given itera
EXERCISE['DIET'] = list(map(laMbda x:x.upper(),EXERCISE['DIET']))
```

Q4B) Assuming the data is stored in a csv file “Exercise.csv”, give appropriate commands to read this file, indexed on ‘Name’ and ‘Diet’ into a dataframe named EXERCISE. Modify this command to read only the first 5 rows of the file. If the file contains millions of records then give the command to read the file in small pieces of uniform size. (2+2+2)

```
EXERCISE_idx = pd.read_csv("/content/Exercise.csv",index_col = ['Name','diet'])
EXERCISE_rows = pd.read_csv("/content/Exercise.csv",index_col = ['Name','diet'],nrows = 5)
chunks = pd.read_csv("/content/Exercise.csv",chunksize = 8) #,index_col = ['Name','diet'],
RESULT = pd.concat(chunks)
```

Q4C)Differentiate between qcut and cut methods. Use the appropriate method to create 4 bins on the ‘Pulse’ attribute. Store the corresponding bin value of ‘Pulse’ attribute as a new attribute ‘Pbin’ in the original DataFrame. Display the count of values of each bin.(1+2+2+1.75)

```
EXERCISE['Pbin']= pd.qcut(EXERCISE['Pulse'], q=4)
print(EXERCISE)
EXERCISE['Pbin'].value_counts()
```

Q5 a) Consider the following DataFrame ADM containing data of freshly admitted students in a college during various rounds of admission. The DataFrame consists of the student’s name, cut off list in which he/she has taken admission, date of admission, his/her % of marks, course code and gender. A) Set the first column ‘Sid’ as the row index of the given DataFrame ADM. Create a pivot table of the DataFrame to display the total number of admissions as per ‘Course Code’ and ‘Gender’. (2+4)

```
#Q5a
import pandas as pd
# DATAFRAME CREATION COMMAND IS NOT REQD. FOR EXAMINER FACILITY ONLY

ADM = pd.DataFrame({
 'Sid': ['S1', 'S2', 'S3', 'S4', 'S5', 'S6', 'S7', 'S8', 'S9', 'S10'],
 'Name': ['Amit Jaiswal','Pradeep Dubey','Rinky Arora','Sonia Shah','Sushil Negi','Neeraj Gaur','Preeti
 'List': ['I', 'II', 'I', 'IV', 'III', 'II','IV','III', 'II','I'],
 'DateAdm': ['01-07-2021','09-07-2021','04-07-2021', '01-08-2021','21-07-2021', '01-07-2021',  '03-08-20
 'Marks%': [97,95,90,96,96.5,94.5,89,95.75,93.5,88.5],
 'CourseCode': ['C001','C009','C112','C001','C001','C009','C112','C001','C009','C112'],  'Gender': ['Mal
ADM
ADM.set_index('Sid')
ADM
pivot = ADM.pivot_table(index =['CourseCode','Gender'],
                        values='Sid',
                        aggfunc ='count')
```

```
pivot = ADM.pivot_table(index =['CourseCode','Gender'],
                        values='Sid',
                        aggfunc ='count').rename(columns={'Sid': 'Total Admissions'})

pivot
```

Q5 b) For each 'List', find the total number of admissions, minimum 'Marks%' and maximum 'Marks%' in each course.(6 MARKS)

```
#5b
result = ADM.groupby(['List','CourseCode']).agg({'Marks%': ['count','min', 'max']})
result
```

Q5 c) Calculate and display the average 'Marks%' of all Female students of course 'C112' (6.75)

```
#Q5 c
Temp=ADM[(ADM['Gender']=="Female") & (ADM['CourseCode']=='C112')]
print(Temp)
ADM[(ADM['Gender']=="Female") & (ADM['CourseCode']=='C112')].agg('Marks%').mean()
```

Q6a) Give Pandas statements to create two data series of random floating-point numbers where the first data series has a datetime index of all second Tuesdays of every month of 2021 and the second data series has a datetime index of 20 continuous dates ending at 31/01/2021(3+3)

```
#Q6 a)
import pandas as pd
import numpy as np
#Create a series of random floating-point numbers with a datetime index of
#all second Tuesdays of every month of 2021
r=pd.date_range('2021-01-01', '2021-12-31', freq='WOM-2TUE')
s=pd.Series(np.random.randn(12),index=r)
print(s)
print(s.index)
#Create a series of random floating-point numbers with a datetime index of
#20 continuous dates ending at 31/01/2021
r=pd.date_range(end='31/1/2021', periods=20, freq='D')
s=pd.Series(np.random.randn(20),index=r)
print(s)
print(s.index)
```

Q6 b) What is resampling? Write python code depicting the usage of resample method.(2+4)

Q6c) Create a DataFrame DS with two columns 'Dates' and 'Sale' containing all dates of January 2021 and 31 random integers between 500 and 1000 respectively. Add another column 'Moving Avg' to DS containing the rolling average of 5 consecutive values in the 'Sale' column. Plot simple line plots between 'Dates' and 'Sale' as

well as ‘Dates’ and ‘Moving Avg’. Explain the utility of the rolling method with respect to these plots. (1+2+2+1.75)

```
#Q6 c
data1=pd.date_range('1/1/2021','31/1/2021')
data2=np.random.randint(500,1000,size=31)
d=list(zip(data1,data2))
DS=pd.DataFrame(d, columns = ['Dates','Sale'])
DS
DS['Moving Avg']=DS['Sale'].rolling(window=5).mean()
print(DS)
DS.plot.line(x='Dates', y='Sale')
DS.plot.line(x='Dates', y='Moving Avg')
DS.plot.line(x='Dates', y=['Sale','Moving Avg'])
```

	Dates	Sale	Moving Avg
0	2021-01-01	512	NaN
1	2021-01-02	802	NaN
2	2021-01-03	561	NaN
3	2021-01-04	609	NaN
4	2021-01-05	551	607.0
5	2021-01-06	555	615.6
6	2021-01-07	538	562.8
7	2021-01-08	681	586.8
8	2021-01-09	549	574.8
9	2021-01-10	926	649.8
10	2021-01-11	527	644.2
11	2021-01-12	953	727.2
12	2021-01-13	874	765.8
13	2021-01-14	790	814.0
14	2021-01-15	764	781.6
15	2021-01-16	699	816.0
16	2021-01-17	665	758.4
17	2021-01-18	998	783.2
18	2021-01-19	504	726.0
19	2021-01-20	987	770.6
20	2021-01-21	544	739.6
21	2021-01-22	616	729.8
22	2021-01-23	631	656.4
23	2021-01-24	886	732.8
24	2021-01-25	772	689.8
25	2021-01-26	974	775.8
26	2021-01-27	901	832.8
27	2021-01-28	616	829.8
28	2021-01-29	698	792.2
29	2021-01-30	538	745.4
30	2021-01-31	855	721.6

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f7e6e11e490>
```