

PDS Mini Project 2

Give your Team Details

Group Number: 4

Team members:

1. Dinesh Subramanian
2. S.Aishwarya
3. Vinod A

Add comments and inferences about the solutions.

1. Create a dataframe named 'europe_df' which has names of countries in europe, their capital and their population. (2 marks)

```
europe_dict = { "Country":["Spain","France","Germany","Norway"], "Capital":  
["Madrid","Paris","Berlin","Oslo"], "Population":[46.77,66.03,80.62,5.084] }
```

```
In [1]: 1 import pandas as pd  
2 europe_dict = {  
3     "Country":["Spain","France","Germany","Norway"],  
4     "Capital":["Madrid","Paris","Berlin","Oslo"],  
5     "Population":[46.77,66.03,80.62,5.084]  
6 }  
7
```

```
In [2]: 1 df = pd.DataFrame(europe_dict)  
2 df
```

Out[2]:

	Country	Capital	Population
0	Spain	Madrid	46.770
1	France	Paris	66.030
2	Germany	Berlin	80.620
3	Norway	Oslo	5.084

2. Use the data from question 1 and add a new column 'FamousFor'. The column should indicate what each country is famous for. (2

marks)

```
In [3]: 1 # type your data here
```

```
In [4]: 1 df[ 'FamousFor' ] = [ 'bullfights', 'Eiffel Tower', 'Architecture', 'Lofoten Is
2 df
```

```
Out[4]:
```

	Country	Capital	Population	FamousFor
0	Spain	Madrid	46.770	bullfights
1	France	Paris	66.030	Eiffel Tower
2	Germany	Berlin	80.620	Architecture
3	Norway	Oslo	5.084	Lofoten Islands

Note: Assigning a column that doesn't exist will create a new column.

3. Use the data created in question 1, to do the following: (3 marks)

Access column 'Capital' by specifying the column number.

Access column 'Population' by specifying the column name.

Access a country information

```
In [5]: 1 # 1. Access column 'Capital' by specifying the column number.
2 df.iloc[0:,1]
3 # type your data here
```

```
Out[5]: 0    Madrid
1     Paris
2    Berlin
3     Oslo
Name: Capital, dtype: object
```

```
In [6]: 1 # 2. Access column 'Population' by specifying the column name.
2 df['Population']
3 # type your data here
```

```
Out[6]: 0    46.770
1    66.030
2    80.620
3     5.084
Name: Population, dtype: float64
```

```
In [7]: 1 # 3. Access a country information
        2 # Lets say we want access information for France
        3 df[df['Country'] == 'France']
        4 # type your data here
```

```
Out[7]:
```

	Country	Capital	Population	FamousFor
1	France	Paris	66.03	Eiffel Tower

4. Read a csv file "example.csv", print it and also check its dimensions (2marks)

```
In [8]: 1 # import the data set
        2 # type your data here
        3 df_example = pd.read_csv('example.csv')
        4 print(df_example)
        5
```

	Product ID	Cost Price	Selling Price
0	45SD	60	135
1	12PO	43	121
2	54PL	78	150
3	26PL	65	121
4	68HG	50	132
5	21ER	150	152
6	10FG	132	165
7	57HB	134	161
8	75VB	109	124
9	32HJH	121	152

```
In [9]: 1 # check dimension/shape of the dataframe
        2 print(df_example.shape)
        3 # type your data here
```

```
(10, 3)
```

5. Create a new column BMI, calculate the BMI of persons based on their weight and height. (2marks)

```
In [10]: 1 # type your data here
2 df_Students = pd.DataFrame({"Name" : ["A", "B", "C"], "Height" : [162,177,179], "Weight" : [69.5, 70.2, 66.3]})
3 df_Students['BMI calculated'] = df_Students['Weight']/(df_Students['Height']/100**2)
4 df_Students
5
```

Out[10]:

	Name	Height	Weight	BMI calculated
0	A	162	69.5	26.482244
1	B	177	70.2	22.407354
2	C	179	66.3	20.692238

6. Create a pandas series having values 4, 7, -5, 3, NAN and their index as d, b, a, c, e (2 marks)

```
In [11]: 1 # type your data here
2 import numpy as np
```

```
In [12]: 1 ds = pd.Series({"d":4, "b":7, "a":-5, "c":3 , "e":np.nan})
2 ds
```

Out[12]:

```
d    4.0
b    7.0
a   -5.0
c    3.0
e    NaN
dtype: float64
```

7. Using the series in question 6, find: (2 marks)

Find the minimum of all values

Find the maximum of all value

```
In [13]: 1 # the minimum of all values
2 ds.min()
3 # type your data here
```

Out[13]: -5.0

```
In [14]: 1 # the maximum of all value
2 ds.max()
3 # type your data here
```

Out[14]: 7.0

8. Using the series in question 6, sort: (2 marks)

the values in ascending order

the values in decending order

```
In [15]: 1 # sorting the values in ascending order
        2 ds.sort_values(ascending=True,na_position='last')
        3 # type your data here
```

```
Out[15]: a    -5.0
        c     3.0
        d     4.0
        b     7.0
        e     NaN
        dtype: float64
```

```
In [16]: 1 ds.sort_values(ascending=False,na_position='last')
        2
```

```
Out[16]: b     7.0
        d     4.0
        c     3.0
        a    -5.0
        e     NaN
        dtype: float64
```

```
In [ ]: 1
```

9. Find duplicate rows based on selected column ('Name') (2 marks)

Use the data frame given below:

Name	Salary	City
John	3400	Sydeny
Robert	3000	Chicago
Aadi	1600	New York
Robert	3000	Chicago
Robert	3000	Chicago
Robert	3000	Texas
Aadi	4000	London
Sachin	3000	Chicago

In [17]:

```
1  htmlTable = ''
2  <table>
3  <tr>
4  <th>Name</th>
5  <th>Salary</th>
6  <th>City</th>
7  </tr>
8  <tr>
9  <td>John</td>
10 <td>3400</td>
11 <td>Sydeny</td>
12 </tr>
13 <tr>
14 <td>Robert</td>
15 <td>3000</td>
16 <td>Chicago</td>
17 </tr>
18 <tr>
19 <td>Aadi</td>
20 <td>1600</td>
21 <td>New York</td>
22 </tr>
23 <tr>
24 <td>Robert</td>
25 <td>3000</td>
26 <td>Chicago</td>
27 </tr>
28 <tr>
29 <td>Robert</td>
30 <td>3000</td>
31 <td>Chicago</td>
32 </tr>
33 <tr>
34 <td>Robert</td>
35 <td>3000</td>
36 <td>Texas</td>
37 </tr>
38 <tr>
39 <td>Aadi</td>
40 <td>4000</td>
41 <td>London</td>
42 </tr>
43 <tr>
44 <td>Sachin</td>
45 <td>3000</td>
46 <td>Chicago</td>
47 </tr>
48 </table>
49 ''
```

```
In [18]: 1 df_Table1 = pd.read_html(htmlTable)
2 df1 = df_Table1[0]
3 df1
```

Out[18]:

	Name	Salary	City
0	John	3400	Sydeny
1	Robert	3000	Chicago
2	Aadi	1600	New York
3	Robert	3000	Chicago
4	Robert	3000	Chicago
5	Robert	3000	Texas
6	Aadi	4000	London
7	Sachin	3000	Chicago

```
In [19]: 1 df1[df1['Name'].duplicated()]
```

Out[19]:

	Name	Salary	City
3	Robert	3000	Chicago
4	Robert	3000	Chicago
5	Robert	3000	Texas
6	Aadi	4000	London

10. Get the descriptive statistics of the sales for each season (2marks)

Use the data frame given below:

Month	Sales	Seasons
Jan	22000	Winter
Feb	27000	Winter
Mar	25000	Spring
Apr	29000	Spring
May	35000	Spring
June	67000	Summer
July	78000	Summer
Aug	67000	Summer
Sep	56000	Fall

Oct	56000	Fall
Nov	56000	Fall
Dec	60000	Winter

In [20]:

```
1 htmlTable2 = '''
2 <table>
3 <tr>
4 <th>Month</th>
5 <th>Sales</th>
6 <th>Seasons</th>
7 </tr>
8 <tr>
9 <td>Jan</td>
10 <td>22000</td>
11 <td>Winter</td>
12 </tr>
13 <tr>
14 <td>Feb</td>
15 <td>27000</td>
16 <td>Winter</td>
17 </tr>
18 <tr>
19 <td>Mar</td>
20 <td>25000</td>
21 <td>Spring</td>
22 </tr>
23 <tr>
24 <td>Apr</td>
25 <td>29000</td>
26 <td>Spring</td>
27 </tr>
28 <tr>
29 <td>May</td>
30 <td>35000</td>
31 <td>Spring</td>
32 </tr>
33 <tr>
34 <td>June</td>
35 <td>67000</td>
36 <td>Summer</td>
37 </tr>
38 <tr>
39 <td>July</td>
40 <td>78000</td>
41 <td>Summer</td>
42 </tr>
43 <tr>
44 <td>Aug</td>
45 <td>67000</td>
46 <td>Summer</td>
47 </tr>
48 <tr>
49 <td>Sep</td>
50 <td>56000</td>
51 <td>Fall</td>
52 </tr>
53 <tr>
54 <td>Oct</td>
55 <td>56000</td>
56 <td>Fall</td>
```

```

57 </tr>
58 <tr>
59 <td>Nov</td>
60 <td>56000</td>
61 <td>Fall</td>
62 </tr>
63 <tr>
64 <td>Dec</td>
65 <td>60000</td>
66 <td>Winter</td>
67 </tr>
68
69 </table>
70 '''

```

```

In [21]: 1 df_Table2 = pd.read_html(htmlTable2)
          2 df2 = df_Table2[0]
          3 df2.groupby('Seasons')['Sales'].describe()

```

```

Out[21]:

```

	count	mean	std	min	25%	50%	75%	max
Seasons								
Fall	3.0	56000.000000	0.000000	56000.0	56000.0	56000.0	56000.0	56000.0
Spring	3.0	29666.666667	5033.222957	25000.0	27000.0	29000.0	32000.0	35000.0
Summer	3.0	70666.666667	6350.852961	67000.0	67000.0	67000.0	72500.0	78000.0
Winter	3.0	36333.333333	20647.840888	22000.0	24500.0	27000.0	43500.0	60000.0

11. Combine the new column age in the below data frame (2 marks)

Use the data frame given below:

Name	Maths	Science	English
Emma	56	89	89
Mia	78	87	89
Sophia	78	78	76
James	67	89	78
John	88	78	87

In [22]:

```
1 table = '''
2 <table>
3 <tr>
4 <th>Name</th>
5 <th>Maths</th>
6 <th>Science</th>
7 <th>English</th>
8 </tr>
9 <tr>
10 <td>Emma</td>
11 <td>56</td>
12 <td>89</td>
13 <td>89</td>
14 </tr>
15 <tr>
16 <td>Mia</td>
17 <td>78</td>
18 <td>87</td>
19 <td>89</td>
20 </tr>
21 <tr>
22 <td>Sophia</td>
23 <td>78</td>
24 <td>78</td>
25 <td>76</td>
26 </tr>
27 <tr>
28 <td>James</td>
29 <td>67</td>
30 <td>89</td>
31 <td>78</td>
32 </tr>
33 <tr>
34 <td>John</td>
35 <td>88</td>
36 <td>78</td>
37 <td>87</td>
38 </tr>
39
40 </table>
41 '''
```

```
In [23]: 1 df = pd.read_html(table)
2 df = df[0]
3 df['Age'] = np.nan
4 df
```

Out[23]:

	Name	Maths	Science	English	Age
0	Emma	56	89	89	NaN
1	Mia	78	87	89	NaN
2	Sophia	78	78	76	NaN
3	James	67	89	78	NaN
4	John	88	78	87	NaN

12.Concatenate two data frames along with the columns

Use the data frame given below:

ID	Name	Subject
101	Alex	Maths
102	Amy	English
103	Allen	Science
104	Alice	German
105	Ayoung	History

ID	Name	Subject
101	Billy	English
102	Brian	Science
103	Bran	Social Science
104	Bryce	German
105	Betty	History

In [24]:

```
1 table_a = '''
2 <table>
3 <tr>
4 <th>ID</th>
5 <th>Name</th>
6 <th>Subject</th>
7 </tr>
8 <tr>
9 <td>101</td>
10 <td>Alex</td>
11 <td>Maths</td>
12 </tr>
13 <tr>
14 <td>102</td>
15 <td>Amy</td>
16 <td>English</td>
17 </tr>
18 <tr>
19 <td>103</td>
20 <td>Allen</td>
21 <td>Science</td>
22 </tr>
23 <tr>
24 <td>104</td>
25 <td>Alice</td>
26 <td>German</td>
27 </tr>
28 <tr>
29 <td>105</td>
30 <td>Ayoung</td>
31 <td>History</td>
32 </tr>
33
34 </table>
35 '''
36
37 table_b = '''
38 <table>
39 <tr>
40 <th>ID</th>
41 <th>Name</th>
42 <th>Subject</th>
43 </tr>
44 <tr>
45 <td>101</td>
46 <td>Billy</td>
47 <td>English</td>
48 </tr>
49 <tr>
50 <td>102</td>
51 <td>Brian</td>
52 <td>Science</td>
53 </tr>
54 <tr>
55 <td>103</td>
56 <td>Bran</td>
```

```

57 <td>Social Science</td>
58 </tr>
59 <tr>
60 <td>104</td>
61 <td>Bryce</td>
62 <td>German</td>
63 </tr>
64 <tr>
65 <td>105</td>
66 <td>Betty</td>
67 <td>History</td>
68 </tr>
69
70 </table>
71 '''

```

```

In [25]: 1 dfA = pd.read_html(table_a)
          2 dfB = pd.read_html(table_b)
          3 dfC = pd.concat([dfA[0],dfB[0]], ignore_index=True)
          4

```

```

In [26]: 1 dfC

```

Out[26]:

	ID	Name	Subject
0	101	Alex	Maths
1	102	Amy	English
2	103	Allen	Science
3	104	Alice	German
4	105	Ayoung	History
5	101	Billy	English
6	102	Brian	Science
7	103	Bran	Social Science
8	104	Bryce	German
9	105	Betty	History

```

In [ ]: 1

```