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

```

In [2]:
```
df = pd.DataFrame(europe_dict)
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]:
```
# type your data here
```

In [4]:
```
df[ 'FamousFor'] = ['bullfights','Eiffel Tower', 'Architecture', 'Lofoten Is
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 'Captial' by specifying the column number.

Access column 'Population' by specifying the column name.

Access a country information

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

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

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

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

In [7]:
```
# 3. Access a country information
# Lets say we want access information for France
df[df['Country'] == 'France']
# 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]:
```
# import the data set
# type your data here
df_example = pd.read_csv('example.csv')
print(df_example)

```

```
  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]:
```
# check dimension/shape of the dataframe
print(df_example.shape)
# 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]:
```
# type your data here
df_Students = pd.DataFrame({"Name" : ["A", "B", "C"], "Height" : [162,177,17
df_Students['BMI calculted'] = df_Students['Weight']/(df_Students['Height']/
df_Students

```

Out[10]:

	Name	Height	Weight	BMI calculted
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]:
```
# type your data here
import numpy as np
```

In [12]:
```
ds = pd.Series({"d":4,"b":7, "a":-5, "c":3 , "e":np.nan})
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]:
```
# the minimum of all values
ds.min()
# type your data here
```

Out[13]:
```
-5.0
```

In [14]:
```
# the maximum of all value
ds.max()
# 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]:
```
# sorting the values in ascending order
ds.sort_values(ascending=True,na_position='last')
# type your data here
```

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

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

```

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

In [ ]:
```

```

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]:

```
htmlTable ='''
<table>
<tr>
<th>Name</th>
<th>Salary</th>
<th>City</th>
</tr>
<tr>
<td>John</td>
<td>3400</td>
<td>Sydeny</td>
</tr>
<tr>
<td>Robert</td>
<td>3000</td>
<td>Chicago</td>
</tr>
<tr>
<td>Aadi</td>
<td>1600</td>
<td>New York</td>
</tr>
<tr>
<td>Robert</td>
<td>3000</td>
<td>Chicago</td>
</tr>
<tr>
<td>Robert</td>
<td>3000</td>
<td>Chicago</td>
</tr>
<tr>
<td>Robert</td>
<td>3000</td>
<td>Texas</td>
</tr>
<tr>
<td>Aadi</td>
<td>4000</td>
<td>London</td>
</tr>
<tr>
<td>Sachin</td>
<td>3000</td>
<td>Chicago</td>
</tr>
</table>
'''
```

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

```
htmlTable2 = '''
<table>
<tr>
<th>Month</th>
<th>Sales</th>
<th>Seasons</th>
</tr>
<tr>
<td>Jan</td>
<td>22000</td>
<td>Winter</td>
</tr>
<tr>
<td>Feb</td>
<td>27000</td>
<td>Winter</td>
</tr>
<tr>
<td>Mar</td>
<td>25000</td>
<td>Spring</td>
</tr>
<tr>
<td>Apr</td>
<td>29000</td>
<td>Spring</td>
</tr>
<tr>
<td>May</td>
<td>35000</td>
<td>Spring</td>
</tr>
<tr>
<td>June</td>
<td>67000</td>
<td>Summer</td>
</tr>
<tr>
<td>July</td>
<td>78000</td>
<td>Summer</td>
</tr>
<tr>
<td>Aug</td>
<td>67000</td>
<td>Summer</td>
</tr>
<tr>
<td>Sep</td>
<td>56000</td>
<td>Fall</td>
</tr>
<tr>
<td>Oct</td>
<td>56000</td>
<td>Fall</td>
```

```
</tr>
<tr>
<td>Nov</td>
<td>56000</td>
<td>Fall</td>
</tr>
<tr>
<td>Dec</td>
<td>60000</td>
<td>Winter</td>
</tr>

</table>
'''
```

In [21]:

```
df_Table2 = pd.read_html(htmlTable2)
df2 = df_Table2[0]
df2.groupby('Seasons')['Sales'].describe()
```

Out[21]:

Seasons	count	mean	std	min	25%	50%	75%	max
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]:

```
table = '''
<table>
<tr>
<th>Name</th>
<th>Maths</th>
<th>Science</th>
<th>English</th>
</tr>
<tr>
<td>Emma</td>
<td>56</td>
<td>89</td>
<td>89</td>
</tr>
<tr>
<td>Mia</td>
<td>78</td>
<td>87</td>
<td>89</td>
</tr>
<tr>
<td>Sophia</td>
<td>78</td>
<td>78</td>
<td>76</td>
</tr>
<tr>
<td>James</td>
<td>67</td>
<td>89</td>
<td>78</td>
</tr>
<tr>
<td>John</td>
<td>88</td>
<td>78</td>
<td>87</td>
</tr>

</table>
'''
```

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

```
table_a = '''
<table>
<tr>
<th>ID</th>
<th>Name</th>
<th>Subject</th>
</tr>
<tr>
<td>101</td>
<td>Alex</td>
<td>Maths</td>
</tr>
<tr>
<td>102</td>
<td>Amy</td>
<td>English</td>
</tr>
<tr>
<td>103</td>
<td>Allen</td>
<td>Science</td>
</tr>
<tr>
<td>104</td>
<td>Alice</td>
<td>German</td>
</tr>
<tr>
<td>105</td>
<td>Ayoung</td>
<td>History</td>
</tr>

</table>
'''

table_b = '''
<table>
<tr>
<th>ID</th>
<th>Name</th>
<th>Subject</th>
</tr>
<tr>
<td>101</td>
<td>Billy</td>
<td>English</td>
</tr>
<tr>
<td>102</td>
<td>Brian</td>
<td>Science</td>
</tr>
<tr>
<td>103</td>
<td>Bran</td>
```

```
<td>Social Science</td>
</tr>
<tr>
<td>104</td>
<td>Bryce</td>
<td>German</td>
</tr>
<tr>
<td>105</td>
<td>Betty</td>
<td>History</td>
</tr>

</table>
'''
```

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

```

In [26]:
```
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 [ ]:
```

```