Take-Home (Day 2)

Let's begin with some hands-on practice exercises

1. Create a dictionary named 'europe_dict' which has names of countries in europe, their capital and their population.

Country	Capital	Population
Spain	Madrid	46.77
France	Paris	66.03
Germany	Berlin	80.62
Norway	Oslo	5.084

In [8]:

```python
# type your code here
import pandas as pd
import numpy as np

europe_dict = {'Country':['Spain','France','Germany','Norway'],
               'Capital':['Madrid','Paris','Berlin','Oslo'],
               'Population':[46.77,66.03,80.62,5.084]
              }
pd1 = pd.DataFrame(europe_dict)
print('The data frame from the given data is: \n')
pd1
```

```
The data frame from the given data is:
```

Out[8]:

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

Country	FamousFor
Spain	Football
France	Eiffel Tower
Germany	Cars

Norway Midnight sun

2. In the data in question 1, add a new column 'FamousFor'. The column tells what each country is famous for.

In [10]:
```
# type your code here
famousfor_dict = {'Spain':'Football','France':'Eiffel Tower','Germany':'Cars','N
pd1['FamousFor']=pd1.Country.map(famousfor_dict)
print('The data frame after adding a new column:\n')
pd1
```

```
The data frame after adding a new column:
```

Out[10]:

	Country	Capital	Population	FamousFor
0	Spain	Madrid	46.770	Football
1	France	Paris	66.030	Eiffel Tower
2	Germany	Berlin	80.620	Cars
3	Norway	Oslo	5.084	Midnight sun

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

1. **Access column 'Captial' by specifying the column number.**
2. **Access column 'Population' by specifying the column name.**
3. **Access a country information**

In [18]:
```
# 1. Access column 'Captial' by specifying the column number.

# type your code here
pd1.iloc[:,1:2]
```

Out[18]:

	Capital
0	Madrid
1	Paris
2	Berlin
3	Oslo

In [19]:
```
# 2. Access column 'Population' by specifying the column name.

# type your code here
pd1.loc[:,'Capital']
```

Out[19]:
```
0    Madrid
1     Paris
2    Berlin
3      Oslo
Name: Capital, dtype: object
```

4. Read a csv file "products.csv", print it and also check its dimensions

In [29]:

```
# read the file

# type your code here
pd4 = pd.read_csv('products.csv')
print(pd4)
print('\nThe dimensions of the .csv files is : ',pd4.ndim)
```

```
  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

The dimensions of the .csv files is :  2
```

In [30]:

```
# check dimension/shape of the dataframe

# type your code here

print('The shape of the list is: ',pd4.shape)
```

```
The shape of the list is:  (10, 3)
```

5. Create a new column Profit, calculate the profit using selling price and the cost price.

In [38]:

```
# type your code here
pd4.rename(columns={'Selling Price':'Selling','Cost Price':'Cost'},inplace=True)
pd4['Profits']=pd4.Selling-pd4.Cost
pd4.rename(columns={'Selling':'Selling Price','Cost':'Cost Price'},inplace=True)
print(pd4)
```

```
  Product ID  Cost Price  Selling Price  Profits
0       45SD          60            135       75
1       12PO          43            121       78
2       54PL          78            150       72
3       26PL          65            121       56
4       68HG          50            132       82
5       21ER         150            152        2
6       10FG         132            165       33
7       57HB         134            161       27
8       75VB         109            124       15
9      32HJH         121            152       31
```

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

In [81]:
```
## type your code here
num = [4,7,-5,3,np.nan]
index_lst = ['d','b','a','c','e']
pd6 = pd.DataFrame(num,index=index_lst,columns=['value'])
pd6
```

Out[81]:

	value
d	4.0
b	7.0
a	-5.0
c	3.0
e	NaN

7. Using the series in question 6, find:
1. the minimum of all values
2. the maximum of all value

In [82]:
```
# the minimum of all values

# type your code here
print('The minimum value in the list is: ',pd6.agg(min))
```

```
The minimum value in the list is:  value   -5.0
dtype: float64
```

In [83]:
```
# the maximum of all value

# type your code here
print('The maximum value in the list is: ',pd6.agg(max))
```

```
The maximum value in the list is:  value    7.0
dtype: float64
```

8. Using the series in question 6, sort:
1. the values in ascending order
2. the values in decending order

In [85]:

```
# sorting the values in ascending order

# type your code here
pd6.sort_values(by='value')
```

Out[85]:

```
   value
a   -5.0
c    3.0
d    4.0
b    7.0
e    NaN
```

In [86]:

```
# sorting the values in decending order

# type your code here
pd6.sort_values(by='value',ascending=False)
```

Out[86]:

```
   value
b    7.0
d    4.0
c    3.0
a   -5.0
e    NaN
```

9. Import dataset 'flights' from library seaborn. Print the first 10 rows and last 20 rows of the data set.

In [89]:

```
# import the dataset

# type your code here
import seaborn as sb
df9 = sb.load_dataset('flights')
```

In [90]:

```
# print the first 10 rows

# type your code here
df9.head(10)
```

Out[90]:

	year	month	passengers
0	1949	January	112
1	1949	February	118
2	1949	March	132
3	1949	April	129
4	1949	May	121
5	1949	June	135
6	1949	July	148
7	1949	August	148
8	1949	September	136
9	1949	October	119

In [91]:

```
# print the last 20 rows

# type your code here
df9.tail(20)
```

Out[91]:

	year	month	passengers
124	1959	May	420
125	1959	June	472
126	1959	July	548
127	1959	August	559
128	1959	September	463
129	1959	October	407
130	1959	November	362
131	1959	December	405
132	1960	January	417
133	1960	February	391
134	1960	March	419
135	1960	April	461
136	1960	May	472
137	1960	June	535
138	1960	July	622
139	1960	August	606
140	1960	September	508
141	1960	October	461
142	1960	November	390
143	1960	December	432

10. Import dataset 'iris' from library seaborn. Check for datatypes of all variable.

In [92]:

```
# load the iris data set as iris

# type your code here
df9b = sb.load_dataset('iris')
```

In [93]:

```
# check for data types

# type your code here
df9b.dtypes
```

Out[93]:

```
sepal_length    float64
sepal_width     float64
petal_length    float64
petal_width     float64
species          object
dtype: object
```