Python for Data Science LAB Session-9

Working with Dataset using pandas

Q-1) Write a program to fill an intermittent time series so all missing dates show up with values of previous non-missing date.

In [1]:

```
import numpy as np
import pandas as pd
```

In [2]:

```
ser = pd.Series([1,10,3, np.nan], index=pd.to_datetime(['2000-05-20', '2000-05-23', '2000-05-26', '2000-05-28']))


print("This series will look like:\n",ser)


# Solution
print("\n\nThe Output of the code should be:\n")
ser.resample('D').ffill()  # fill with previous value
```

```
This series will look like:
 2000-05-20     1.0
2000-05-23    10.0
2000-05-26     3.0
2000-05-28     NaN
dtype: float64


The Output of the code should be:

```

Out[2]:

```
2000-05-20     1.0
2000-05-21     1.0
2000-05-22     1.0
2000-05-23    10.0
2000-05-24    10.0
2000-05-25    10.0
2000-05-26     3.0
2000-05-27     3.0
2000-05-28     NaN
Freq: D, dtype: float64
```

Q-2)Write a sample program to explain how to find missing values from a dataset, how to remove the missing values and how to fill the missing values using fillna() and Imputer.

In [3]:

```
df = pd.read_csv("heart.csv")
```

In [4]:

```
df # dataframe
```

Out[4]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63.0	1	3	145.0	233.0	1.0	0.0	150	0.0	2.3	0.0	0.0	1.0	1.0
1	NaN	1	2	130.0	250.0	0.0	1.0	187	0.0	3.5	0.0	0.0	2.0	NaN
2	41.0	0	1	130.0	NaN	NaN	0.0	172	0.0	NaN	2.0	0.0	2.0	1.0
3	56.0	1	1	120.0	236.0	0.0	1.0	178	0.0	0.8	2.0	0.0	2.0	1.0
4	57.0	0	0	NaN	354.0	0.0	1.0	163	1.0	0.6	2.0	NaN	2.0	1.0
5	NaN	1	0	140.0	192.0	0.0	NaN	148	0.0	0.4	NaN	0.0	1.0	1.0
6	56.0	0	1	140.0	294.0	0.0	0.0	153	0.0	1.3	1.0	0.0	NaN	1.0
7	44.0	1	1	120.0	NaN	0.0	1.0	173	NaN	0.0	2.0	0.0	3.0	1.0
8	52.0	1	2	172.0	199.0	1.0	1.0	162	0.0	0.5	2.0	0.0	3.0	NaN
9	57.0	1	2	150.0	168.0	0.0	1.0	174	0.0	1.6	2.0	0.0	2.0	1.0
10	54.0	1	0	140.0	239.0	0.0	1.0	160	0.0	1.2	2.0	0.0	2.0	1.0
11	48.0	0	2	130.0	275.0	0.0	1.0	139	0.0	0.2	2.0	0.0	2.0	1.0
12	49.0	1	1	130.0	266.0	0.0	1.0	171	0.0	0.6	2.0	0.0	2.0	1.0
13	64.0	1	3	110.0	211.0	0.0	0.0	144	1.0	1.8	1.0	0.0	2.0	1.0
14	58.0	0	3	150.0	283.0	1.0	0.0	162	0.0	1.0	2.0	0.0	2.0	1.0
15	50.0	0	2	120.0	219.0	0.0	1.0	158	0.0	1.6	1.0	0.0	2.0	1.0
16	58.0	0	2	120.0	340.0	0.0	1.0	172	0.0	0.0	2.0	0.0	2.0	1.0
17	66.0	0	3	150.0	226.0	0.0	1.0	114	0.0	2.6	0.0	0.0	2.0	1.0
18	43.0	1	0	150.0	247.0	0.0	1.0	171	0.0	1.5	2.0	0.0	2.0	1.0
19	69.0	0	3	140.0	239.0	0.0	1.0	151	0.0	1.8	2.0	2.0	2.0	1.0
20	59.0	1	0	135.0	234.0	0.0	1.0	161	0.0	0.5	1.0	0.0	3.0	1.0
21	44.0	1	2	130.0	233.0	0.0	1.0	179	1.0	0.4	2.0	0.0	2.0	1.0
22	42.0	1	0	140.0	226.0	0.0	1.0	178	0.0	0.0	2.0	0.0	2.0	1.0
23	61.0	1	2	150.0	243.0	1.0	1.0	137	1.0	1.0	1.0	0.0	2.0	1.0
24	40.0	1	3	140.0	199.0	0.0	1.0	178	1.0	1.4	2.0	0.0	3.0	1.0
25	71.0	0	1	160.0	302.0	0.0	1.0	162	0.0	0.4	2.0	2.0	2.0	1.0
26	59.0	1	2	150.0	212.0	1.0	1.0	157	0.0	1.6	2.0	0.0	2.0	1.0
27	51.0	1	2	110.0	175.0	0.0	1.0	123	0.0	0.6	2.0	0.0	2.0	1.0
28	65.0	0	2	140.0	417.0	1.0	0.0	157	0.0	0.8	2.0	1.0	2.0	1.0
29	53.0	1	2	130.0	197.0	1.0	0.0	152	0.0	1.2	0.0	0.0	2.0	1.0
30	41.0	0	1	105.0	198.0	0.0	1.0	168	0.0	0.0	2.0	1.0	2.0	1.0
31	65.0	1	0	120.0	177.0	0.0	1.0	140	0.0	0.4	2.0	0.0	3.0	1.0
32	44.0	1	1	130.0	219.0	0.0	0.0	188	0.0	0.0	2.0	0.0	2.0	1.0
33	54.0	1	2	125.0	273.0	0.0	0.0	152	0.0	0.5	0.0	1.0	2.0	1.0
34	51.0	1	3	125.0	213.0	0.0	0.0	125	1.0	1.4	2.0	1.0	2.0	1.0
35	46.0	0	2	142.0	177.0	0.0	0.0	160	1.0	1.4	0.0	0.0	2.0	1.0
36	54.0	0	2	135.0	304.0	1.0	1.0	170	0.0	0.0	2.0	0.0	2.0	1.0
37	54.0	1	2	150.0	232.0	0.0	0.0	165	0.0	1.6	2.0	0.0	3.0	1.0
38	65.0	0	2	155.0	269.0	0.0	1.0	148	0.0	0.8	2.0	0.0	2.0	1.0
39	65.0	0	2	160.0	360.0	0.0	0.0	151	0.0	0.8	2.0	0.0	2.0	1.0
40	51.0	0	2	140.0	308.0	0.0	0.0	142	0.0	1.5	2.0	1.0	2.0	1.0
41	48.0	1	1	130.0	245.0	0.0	0.0	180	0.0	0.2	1.0	0.0	2.0	1.0
42	45.0	1	0	104.0	208.0	0.0	0.0	148	1.0	3.0	1.0	0.0	2.0	1.0
43	53.0	0	0	130.0	264.0	0.0	0.0	143	0.0	0.4	1.0	0.0	2.0	1.0
44	39.0	1	2	140.0	321.0	0.0	0.0	182	0.0	0.0	2.0	0.0	2.0	1.0
45	52.0	1	1	120.0	325.0	0.0	1.0	172	0.0	0.2	2.0	0.0	2.0	1.0
46	44.0	1	2	140.0	235.0	0.0	0.0	180	0.0	0.0	2.0	0.0	2.0	1.0
47	47.0	1	2	138.0	257.0	0.0	0.0	156	0.0	0.0	2.0	0.0	2.0	1.0
48	53.0	0	2	128.0	216.0	0.0	0.0	115	0.0	0.0	2.0	0.0	0.0	1.0

NA -> FOR NOT AVAILABLE NaN -> Not a Number None -> Python singleton object that is often used for missing data in Python code

```
In [6]: data = df.iloc[1:10] # fetch only 10 value
```

```
In [7]: data
```

Out[7]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
1	NaN	1	2	130.0	250.0	0.0	1.0	187	0.0	3.5	0.0	0.0	2.0	NaN
2	41.0	0	1	130.0	NaN	NaN	0.0	172	0.0	NaN	2.0	0.0	2.0	1.0
3	56.0	1	1	120.0	236.0	0.0	1.0	178	0.0	0.8	2.0	0.0	2.0	1.0
4	57.0	0	0	NaN	354.0	0.0	1.0	163	1.0	0.6	2.0	NaN	2.0	1.0
5	NaN	1	0	140.0	192.0	0.0	NaN	148	0.0	0.4	NaN	0.0	1.0	1.0
6	56.0	0	1	140.0	294.0	0.0	0.0	153	0.0	1.3	1.0	0.0	NaN	1.0
7	44.0	1	1	120.0	NaN	0.0	1.0	173	NaN	0.0	2.0	0.0	3.0	1.0
8	52.0	1	2	172.0	199.0	1.0	1.0	162	0.0	0.5	2.0	0.0	3.0	NaN
9	57.0	1	2	150.0	168.0	0.0	1.0	174	0.0	1.6	2.0	0.0	2.0	1.0

we use isnull() function this function return dataframe of Boolean values which are True for NaN values

```
In [8]: data.isnull()  # True return where missing value otherwise false
```

Out[8]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
1	True	False	False	False	False	False	False	False	False	False	False	False	False	True
2	False	False	False	False	True	True	False	False	False	True	False	False	False	False
3	False	False	False	False	False	False	False	False	False	False	False	False	False	False
4	False	False	False	True	False	False	False	False	False	False	False	True	False	False
5	True	False	False	False	False	False	True	False	False	False	True	False	False	False
6	False	False	False	False	False	False	False	False	False	False	False	False	True	False
7	False	False	False	False	True	False	False	False	True	False	False	False	False	False
8	False	False	False	False	False	False	False	False	False	False	False	False	False	True
9	False	False	False	False	False	False	False	False	False	False	False	False	False	False

```
In [9]: data.notnull() # it retuen True where vaule exits otherwise False for missing vaule
```

Out[9]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
1	False	True	True	True	True	True	True	True	True	True	True	True	True	False
2	True	True	True	True	False	False	True	True	True	False	True	True	True	True
3	True	True	True	True	True	True	True	True	True	True	True	True	True	True
4	True	True	True	False	True	True	True	True	True	True	True	False	True	True
5	False	True	True	True	True	True	False	True	True	True	False	True	True	True
6	True	True	True	True	True	True	True	True	True	True	True	True	False	True
7	True	True	True	True	False	True	True	True	False	True	True	True	True	True
8	True	True	True	True	True	True	True	True	True	True	True	True	True	False
9	True	True	True	True	True	True	True	True	True	True	True	True	True	True

```
In [10]: data.fillna('Empty')   # fillna retuen Null vaule to fill EMPTY value
```

Out[10]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
1	Empty	1	2	130	250	0	1	187	0	3.5	0	0	2	Empty
2	41	0	1	130	Empty	Empty	0	172	0	Empty	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	Empty	354	0	1	163	1	0.6	2	Empty	2	1
5	Empty	1	0	140	192	0	Empty	148	0	0.4	Empty	0	1	1
6	56	0	1	140	294	0	0	153	0	1.3	1	0	Empty	1
7	44	1	1	120	Empty	0	1	173	Empty	0	2	0	3	1
8	52	1	2	172	199	1	1	162	0	0.5	2	0	3	Empty
9	57	1	2	150	168	0	1	174	0	1.6	2	0	2	1

```
In [11]: data.fillna(method ='pad') # it's return Null vaule to fill with previous vaule
```

Out[11]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
1	NaN	1	2	130.0	250.0	0.0	1.0	187	0.0	3.5	0.0	0.0	2.0	NaN
2	41.0	0	1	130.0	250.0	0.0	0.0	172	0.0	3.5	2.0	0.0	2.0	1.0
3	56.0	1	1	120.0	236.0	0.0	1.0	178	0.0	0.8	2.0	0.0	2.0	1.0
4	57.0	0	0	120.0	354.0	0.0	1.0	163	1.0	0.6	2.0	0.0	2.0	1.0
5	57.0	1	0	140.0	192.0	0.0	1.0	148	0.0	0.4	2.0	0.0	1.0	1.0
6	56.0	0	1	140.0	294.0	0.0	0.0	153	0.0	1.3	1.0	0.0	1.0	1.0
7	44.0	1	1	120.0	294.0	0.0	1.0	173	0.0	0.0	2.0	0.0	3.0	1.0
8	52.0	1	2	172.0	199.0	1.0	1.0	162	0.0	0.5	2.0	0.0	3.0	1.0
9	57.0	1	2	150.0	168.0	0.0	1.0	174	0.0	1.6	2.0	0.0	2.0	1.0

```
In [12]: data.fillna(method ='bfill')   # it's return Null vaule to fill with next vaule
```

Out[12]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
1	41.0	1	2	130.0	250.0	0.0	1.0	187	0.0	3.5	0.0	0.0	2.0	1.0
2	41.0	0	1	130.0	236.0	0.0	0.0	172	0.0	0.8	2.0	0.0	2.0	1.0
3	56.0	1	1	120.0	236.0	0.0	1.0	178	0.0	0.8	2.0	0.0	2.0	1.0
4	57.0	0	0	140.0	354.0	0.0	1.0	163	1.0	0.6	2.0	0.0	2.0	1.0
5	56.0	1	0	140.0	192.0	0.0	0.0	148	0.0	0.4	1.0	0.0	1.0	1.0
6	56.0	0	1	140.0	294.0	0.0	0.0	153	0.0	1.3	1.0	0.0	3.0	1.0
7	44.0	1	1	120.0	199.0	0.0	1.0	173	0.0	0.0	2.0	0.0	3.0	1.0
8	52.0	1	2	172.0	199.0	1.0	1.0	162	0.0	0.5	2.0	0.0	3.0	1.0
9	57.0	1	2	150.0	168.0	0.0	1.0	174	0.0	1.6	2.0	0.0	2.0	1.0

In [13]:
```
data["age"].fillna("No age ", inplace = True)
data   # just replace age column in null value as No age
```

```
c:\users\arjun vankani\appdata\local\programs\python\python37\lib\site-packages\pandas\core\series.py:4523: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  downcast=downcast,
```

Out[13]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
1	No age	1	2	130.0	250.0	0.0	1.0	187	0.0	3.5	0.0	0.0	2.0	NaN
2	41	0	1	130.0	NaN	NaN	0.0	172	0.0	NaN	2.0	0.0	2.0	1.0
3	56	1	1	120.0	236.0	0.0	1.0	178	0.0	0.8	2.0	0.0	2.0	1.0
4	57	0	0	NaN	354.0	0.0	1.0	163	1.0	0.6	2.0	NaN	2.0	1.0
5	No age	1	0	140.0	192.0	0.0	NaN	148	0.0	0.4	NaN	0.0	1.0	1.0
6	56	0	1	140.0	294.0	0.0	0.0	153	0.0	1.3	1.0	0.0	NaN	1.0
7	44	1	1	120.0	NaN	0.0	1.0	173	NaN	0.0	2.0	0.0	3.0	1.0
8	52	1	2	172.0	199.0	1.0	1.0	162	0.0	0.5	2.0	0.0	3.0	NaN
9	57	1	2	150.0	168.0	0.0	1.0	174	0.0	1.6	2.0	0.0	2.0	1.0

In [14]:
```
data.replace(to_replace = np.nan, value = '--')
```

Out[14]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
1	No age	1	2	130	250	0	1	187	0	3.5	0	0	2	--
2	41	0	1	130	--	--	0	172	0	--	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	--	354	0	1	163	1	0.6	2	--	2	1
5	No age	1	0	140	192	0	--	148	0	0.4	--	0	1	1
6	56	0	1	140	294	0	0	153	0	1.3	1	0	--	1
7	44	1	1	120	--	0	1	173	--	0	2	0	3	1
8	52	1	2	172	199	1	1	162	0	0.5	2	0	3	--
9	57	1	2	150	168	0	1	174	0	1.6	2	0	2	1

It uses various interpolation technique to fill the missing values rather than hard-coding the value, Linear method ignore the index and treat the values as equally spaced

In [15]:
```
data.interpolate(method ='linear', limit_direction ='forward')
```

Out[15]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
1	No age	1	2	130.0	250.0	0.0	1.0	187	0.0	3.50	0.0	0.0	2.0	NaN
2	41	0	1	130.0	243.0	0.0	0.0	172	0.0	2.15	2.0	0.0	2.0	1.0
3	56	1	1	120.0	236.0	0.0	1.0	178	0.0	0.80	2.0	0.0	2.0	1.0
4	57	0	0	130.0	354.0	0.0	1.0	163	1.0	0.60	2.0	0.0	2.0	1.0
5	No age	1	0	140.0	192.0	0.0	0.5	148	0.0	0.40	1.5	0.0	1.0	1.0
6	56	0	1	140.0	294.0	0.0	0.0	153	0.0	1.30	1.0	0.0	2.0	1.0
7	44	1	1	120.0	246.5	0.0	1.0	173	0.0	0.00	2.0	0.0	3.0	1.0
8	52	1	2	172.0	199.0	1.0	1.0	162	0.0	0.50	2.0	0.0	3.0	1.0
9	57	1	2	150.0	168.0	0.0	1.0	174	0.0	1.60	2.0	0.0	2.0	1.0

In [16]:
```
data.interpolate(method ='linear', limit_direction ='backward', limit = 1)
```

Out[16]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
1	No age	1	2	130.0	250.0	0.0	1.0	187	0.0	3.50	0.0	0.0	2.0	1.0
2	41	0	1	130.0	243.0	0.0	0.0	172	0.0	2.15	2.0	0.0	2.0	1.0
3	56	1	1	120.0	236.0	0.0	1.0	178	0.0	0.80	2.0	0.0	2.0	1.0
4	57	0	0	130.0	354.0	0.0	1.0	163	1.0	0.60	2.0	0.0	2.0	1.0
5	No age	1	0	140.0	192.0	0.0	0.5	148	0.0	0.40	1.5	0.0	1.0	1.0
6	56	0	1	140.0	294.0	0.0	0.0	153	0.0	1.30	1.0	0.0	2.0	1.0
7	44	1	1	120.0	246.5	0.0	1.0	173	0.0	0.00	2.0	0.0	3.0	1.0
8	52	1	2	172.0	199.0	1.0	1.0	162	0.0	0.50	2.0	0.0	3.0	1.0
9	57	1	2	150.0	168.0	0.0	1.0	174	0.0	1.60	2.0	0.0	2.0	1.0

In [17]:
```
data.dropna()  #only show fully filled row , another all are leave
```

Out[17]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
3	56	1	1	120.0	236.0	0.0	1.0	178	0.0	0.8	2.0	0.0	2.0	1.0
9	57	1	2	150.0	168.0	0.0	1.0	174	0.0	1.6	2.0	0.0	2.0	1.0

In [18]:
```
data.dropna(how = 'all')  # we drop a rows whose all data is missing or contain null value
```

Out[18]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
1	No age	1	2	130.0	250.0	0.0	1.0	187	0.0	3.5	0.0	0.0	2.0	NaN
2	41	0	1	130.0	NaN	NaN	0.0	172	0.0	NaN	2.0	0.0	2.0	1.0
3	56	1	1	120.0	236.0	0.0	1.0	178	0.0	0.8	2.0	0.0	2.0	1.0
4	57	0	0	NaN	354.0	0.0	1.0	163	1.0	0.6	2.0	NaN	2.0	1.0
5	No age	1	0	140.0	192.0	0.0	NaN	148	0.0	0.4	NaN	0.0	1.0	1.0
6	56	0	1	140.0	294.0	0.0	0.0	153	0.0	1.3	1.0	0.0	NaN	1.0
7	44	1	1	120.0	NaN	0.0	1.0	173	NaN	0.0	2.0	0.0	3.0	1.0
8	52	1	2	172.0	199.0	1.0	1.0	162	0.0	0.5	2.0	0.0	3.0	NaN
9	57	1	2	150.0	168.0	0.0	1.0	174	0.0	1.6	2.0	0.0	2.0	1.0

In [19]:
```
data.dropna(axis = 1)   # drop null column
```

Out[19]:

	age	sex	cp	thalach
1	No age	1	2	187
2	41	0	1	172
3	56	1	1	178
4	57	0	0	163
5	No age	1	0	148
6	56	0	1	153
7	44	1	1	173
8	52	1	2	162
9	57	1	2	174

In [20]:
```
data.dropna(axis = 0, how ='any')  # if only one vaule missing in any row then remove it's row
```

Out[20]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
3	56	1	1	120.0	236.0	0.0	1.0	178	0.0	0.8	2.0	0.0	2.0	1.0
9	57	1	2	150.0	168.0	0.0	1.0	174	0.0	1.6	2.0	0.0	2.0	1.0

Q-3) Write a sample program to explain working with categorical variables:

```
a. Define categorical variables with at least 5 categories. Take sample input that has values from the categories and some values which are out of the categories and observe
the output

b. Rename the categorical values to some new values and observer the output

c. Combine any two categories with each other (combining the levels) so that now there are only three categories. Apply the new categories to the dataset and observe the out
put.
```

In [64]:
```
m = pd.Categorical(['a', 'k', 'l', 'e', 'a', 'b','e'])
m
```

Out[64]:
```
['a', 'k', 'l', 'e', 'a', 'b', 'e']
Categories (5, object): ['a', 'b', 'e', 'k', 'l']
```

In [27]:
```
e = pd.Categorical(['a', 'b', 'c', 'e', 'a', 'b','e'])
e
```

Out[27]:
```
['a', 'b', 'c', 'e', 'a', 'b', 'e']
Categories (4, object): ['a', 'b', 'c', 'e']
```

In [87]:
```
from pandas.api.types import CategoricalDtype
e == CategoricalDtype(['b', 'c', 'a'])

# yes it return true bcz it belong to this catogary
```

Out[87]:
```
True
```

In [88]:
```
m == CategoricalDtype(['k', 'l', 'a'], ordered=True)
```

Out[88]:
```
array([False, False, False, False, False, False, False])
```

In [30]:
```
c.describe()
```

Out[30]:
```
count     5
unique    5
top       e
freq      1
dtype: object
```

In [29]:
```
e.describe()
```

Out[29]:

	counts	freqs
categories		
a	2	0.285714
b	2	0.285714
c	1	0.142857
e	2	0.285714

In [90]:
```
#renameing
```

In [94]:
```
d1 = pd.Series(["k","l","m","p","k"],dtype="category")
d1
```

Out[94]:
```
0    k
1    l
2    m
3    p
4    k
dtype: category
Categories (4, object): ['k', 'l', 'm', 'p']
```

In [95]:
```
d1.cat.categories = ["Group %s" % g for g in d1.cat.categories] # group
```

In [96]:
```
d1
```

Out[96]:
```
0    Group k
1    Group l
2    Group m
3    Group p
4    Group k
dtype: category
Categories (4, object): ['Group k', 'Group l', 'Group m', 'Group p']
```

In [101]:
```
d1 = d1.cat.rename_categories([1, 2, 3, 4])
d1  #rename
```

Out[101]:
```
0    1
1    2
2    3
3    4
4    1
dtype: category
Categories (4, int64): [1, 2, 3, 4]
```

In [ ]:
```

```

In [106]:
```
d1 = d1.cat.add_categories([5])
d1
```

Out[106]:
```
0    1
1    2
2    3
3    4
4    1
dtype: category
Categories (6, object): [1, 2, 3, 4, 'a', 5]
```

In [107]:
```
d1.cat.categories
```

Out[107]:
```
Index([1, 2, 3, 4, 'a', 5], dtype='object')
```

In [110]:
```
d1 = d1.cat.remove_categories([4])
d1
```

Out[110]:
```
0      1
1      2
2      3
3    NaN
4      1
dtype: category
Categories (5, object): [1, 2, 3, 'a', 5]
```

In [ ]:
```

```

In [115]:

```
from pandas.api.types import union_categoricals
ser1 = pd.Series(['k', 'l'], dtype='category')
ser2 = pd.Series(['m', 'b', 'a'], dtype='category')
print(ser1)
print(ser2)
pd.concat([ser1,ser2])
```

```
0    k
1    l
dtype: category
Categories (2, object): ['k', 'l']
0    m
1    b
2    a
dtype: category
Categories (3, object): ['a', 'b', 'm']
```

Out[115]:

```
0    k
1    l
0    m
1    b
2    a
dtype: object
```

In [123]:

```
ser3 = union_categoricals([ser1.array, ser2.array])
ser3
```

Out[123]:

```
['k', 'l', 'm', 'b', 'a']
Categories (5, object): ['k', 'l', 'a', 'b', 'm']
```

Q-4) Write a sample program to demonstrate how to sort and shuffle the data from a dataset.

In [126]:

```
data
```

Out[126]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
1	No age	1	2	130.0	250.0	0.0	1.0	187	0.0	3.5	0.0	0.0	2.0	NaN
2	41	0	1	130.0	NaN	NaN	0.0	172	0.0	NaN	2.0	0.0	2.0	1.0
3	56	1	1	120.0	236.0	0.0	1.0	178	0.0	0.8	2.0	0.0	2.0	1.0
4	57	0	0	NaN	354.0	0.0	1.0	163	1.0	0.6	2.0	NaN	2.0	1.0
5	No age	1	0	140.0	192.0	0.0	NaN	148	0.0	0.4	NaN	0.0	1.0	1.0
6	56	0	1	140.0	294.0	0.0	0.0	153	0.0	1.3	1.0	0.0	NaN	1.0
7	44	1	1	120.0	NaN	0.0	1.0	173	NaN	0.0	2.0	0.0	3.0	1.0
8	52	1	2	172.0	199.0	1.0	1.0	162	0.0	0.5	2.0	0.0	3.0	NaN
9	57	1	2	150.0	168.0	0.0	1.0	174	0.0	1.6	2.0	0.0	2.0	1.0

In [134]:

```
data.sort_values("trestbps", axis = 0, ascending = True, inplace = True)
data  # sort by perticular column
```

```
c:\users\arjun vankani\appdata\local\programs\python\python37\lib\site-packages\ipykernel_launcher.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  """Entry point for launching an IPython kernel.
```

Out[134]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
3	56	1	1	120.0	236.0	0.0	1.0	178	0.0	0.8	2.0	0.0	2.0	1.0
7	44	1	1	120.0	NaN	0.0	1.0	173	NaN	0.0	2.0	0.0	3.0	1.0
1	No age	1	2	130.0	250.0	0.0	1.0	187	0.0	3.5	0.0	0.0	2.0	NaN
2	41	0	1	130.0	NaN	NaN	0.0	172	0.0	NaN	2.0	0.0	2.0	1.0
5	No age	1	0	140.0	192.0	0.0	NaN	148	0.0	0.4	NaN	0.0	1.0	1.0
6	56	0	1	140.0	294.0	0.0	0.0	153	0.0	1.3	1.0	0.0	NaN	1.0
9	57	1	2	150.0	168.0	0.0	1.0	174	0.0	1.6	2.0	0.0	2.0	1.0
8	52	1	2	172.0	199.0	1.0	1.0	162	0.0	0.5	2.0	0.0	3.0	NaN
4	57	0	0	NaN	354.0	0.0	1.0	163	1.0	0.6	2.0	NaN	2.0	1.0

In [136]:

```
data.sort_values("thalach", axis = 0, ascending = True, inplace = True)
data
```

```
c:\users\arjun vankani\appdata\local\programs\python\python37\lib\site-packages\ipykernel_launcher.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  """Entry point for launching an IPython kernel.
```

Out[136]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
5	No age	1	0	140.0	192.0	0.0	NaN	148	0.0	0.4	NaN	0.0	1.0	1.0
6	56	0	1	140.0	294.0	0.0	0.0	153	0.0	1.3	1.0	0.0	NaN	1.0
8	52	1	2	172.0	199.0	1.0	1.0	162	0.0	0.5	2.0	0.0	3.0	NaN
4	57	0	0	NaN	354.0	0.0	1.0	163	1.0	0.6	2.0	NaN	2.0	1.0
2	41	0	1	130.0	NaN	NaN	0.0	172	0.0	NaN	2.0	0.0	2.0	1.0
7	44	1	1	120.0	NaN	0.0	1.0	173	NaN	0.0	2.0	0.0	3.0	1.0
9	57	1	2	150.0	168.0	0.0	1.0	174	0.0	1.6	2.0	0.0	2.0	1.0
3	56	1	1	120.0	236.0	0.0	1.0	178	0.0	0.8	2.0	0.0	2.0	1.0
1	No age	1	2	130.0	250.0	0.0	1.0	187	0.0	3.5	0.0	0.0	2.0	NaN

In [137]:

```
data.sample(frac=1).reset_index(drop=True) #shuffle frame
```

Out[137]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	No age	1	0	140.0	192.0	0.0	NaN	148	0.0	0.4	NaN	0.0	1.0	1.0
1	57	0	0	NaN	354.0	0.0	1.0	163	1.0	0.6	2.0	NaN	2.0	1.0
2	No age	1	2	130.0	250.0	0.0	1.0	187	0.0	3.5	0.0	0.0	2.0	NaN
3	57	1	2	150.0	168.0	0.0	1.0	174	0.0	1.6	2.0	0.0	2.0	1.0
4	52	1	2	172.0	199.0	1.0	1.0	162	0.0	0.5	2.0	0.0	3.0	NaN
5	41	0	1	130.0	NaN	NaN	0.0	172	0.0	NaN	2.0	0.0	2.0	1.0
6	56	1	1	120.0	236.0	0.0	1.0	178	0.0	0.8	2.0	0.0	2.0	1.0
7	56	0	1	140.0	294.0	0.0	0.0	153	0.0	1.3	1.0	0.0	NaN	1.0
8	44	1	1	120.0	NaN	0.0	1.0	173	NaN	0.0	2.0	0.0	3.0	1.0

In [142]:
```
data.sample(frac=1)
```

Out[142]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
4	57	0	0	NaN	354.0	0.0	1.0	163	1.0	0.6	2.0	NaN	2.0	1.0
8	52	1	2	172.0	199.0	1.0	1.0	162	0.0	0.5	2.0	0.0	3.0	NaN
5	No age	1	0	140.0	192.0	0.0	NaN	148	0.0	0.4	NaN	0.0	1.0	1.0
2	41	0	1	130.0	NaN	NaN	0.0	172	0.0	NaN	2.0	0.0	2.0	1.0
6	56	0	1	140.0	294.0	0.0	0.0	153	0.0	1.3	1.0	0.0	NaN	1.0
3	56	1	1	120.0	236.0	0.0	1.0	178	0.0	0.8	2.0	0.0	2.0	1.0
7	44	1	1	120.0	NaN	0.0	1.0	173	NaN	0.0	2.0	0.0	3.0	1.0
9	57	1	2	150.0	168.0	0.0	1.0	174	0.0	1.6	2.0	0.0	2.0	1.0
1	No age	1	2	130.0	250.0	0.0	1.0	187	0.0	3.5	0.0	0.0	2.0	NaN

In [143]:
```
data.sort_index()
```

Out[143]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
1	No age	1	2	130.0	250.0	0.0	1.0	187	0.0	3.5	0.0	0.0	2.0	NaN
2	41	0	1	130.0	NaN	NaN	0.0	172	0.0	NaN	2.0	0.0	2.0	1.0
3	56	1	1	120.0	236.0	0.0	1.0	178	0.0	0.8	2.0	0.0	2.0	1.0
4	57	0	0	NaN	354.0	0.0	1.0	163	1.0	0.6	2.0	NaN	2.0	1.0
5	No age	1	0	140.0	192.0	0.0	NaN	148	0.0	0.4	NaN	0.0	1.0	1.0
6	56	0	1	140.0	294.0	0.0	0.0	153	0.0	1.3	1.0	0.0	NaN	1.0
7	44	1	1	120.0	NaN	0.0	1.0	173	NaN	0.0	2.0	0.0	3.0	1.0
8	52	1	2	172.0	199.0	1.0	1.0	162	0.0	0.5	2.0	0.0	3.0	NaN
9	57	1	2	150.0	168.0	0.0	1.0	174	0.0	1.6	2.0	0.0	2.0	1.0

In [145]:
```
data.sort_index(ascending=False)
```

Out[145]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
9	57	1	2	150.0	168.0	0.0	1.0	174	0.0	1.6	2.0	0.0	2.0	1.0
8	52	1	2	172.0	199.0	1.0	1.0	162	0.0	0.5	2.0	0.0	3.0	NaN
7	44	1	1	120.0	NaN	0.0	1.0	173	NaN	0.0	2.0	0.0	3.0	1.0
6	56	0	1	140.0	294.0	0.0	0.0	153	0.0	1.3	1.0	0.0	NaN	1.0
5	No age	1	0	140.0	192.0	0.0	NaN	148	0.0	0.4	NaN	0.0	1.0	1.0
4	57	0	0	NaN	354.0	0.0	1.0	163	1.0	0.6	2.0	NaN	2.0	1.0
3	56	1	1	120.0	236.0	0.0	1.0	178	0.0	0.8	2.0	0.0	2.0	1.0
2	41	0	1	130.0	NaN	NaN	0.0	172	0.0	NaN	2.0	0.0	2.0	1.0
1	No age	1	2	130.0	250.0	0.0	1.0	187	0.0	3.5	0.0	0.0	2.0	NaN

In [146]:
```
data.sort_index(axis=1)
```

Out[146]:

	age	ca	chol	cp	exang	fbs	oldpeak	restecg	sex	slope	target	thal	thalach	trestbps
5	No age	0.0	192.0	0	0.0	0.0	0.4	NaN	1	NaN	1.0	1.0	148	140.0
6	56	0.0	294.0	1	0.0	0.0	1.3	0.0	0	1.0	1.0	NaN	153	140.0
8	52	0.0	199.0	2	0.0	1.0	0.5	1.0	1	2.0	NaN	3.0	162	172.0
4	57	NaN	354.0	0	1.0	0.0	0.6	1.0	0	2.0	1.0	2.0	163	NaN
2	41	0.0	NaN	1	0.0	NaN	NaN	0.0	0	2.0	1.0	2.0	172	130.0
7	44	0.0	NaN	1	NaN	0.0	0.0	1.0	1	2.0	1.0	3.0	173	120.0
9	57	0.0	168.0	2	0.0	0.0	1.6	1.0	1	2.0	1.0	2.0	174	150.0
3	56	0.0	236.0	1	0.0	0.0	0.8	1.0	1	2.0	1.0	2.0	178	120.0
1	No age	0.0	250.0	2	0.0	0.0	3.5	1.0	1	0.0	NaN	2.0	187	130.0

In [150]:
```
data.sort_values(by='trestbps',kind='mergesort')
```

Out[150]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
7	44	1	1	120.0	NaN	0.0	1.0	173	NaN	0.0	2.0	0.0	3.0	1.0
3	56	1	1	120.0	236.0	0.0	1.0	178	0.0	0.8	2.0	0.0	2.0	1.0
2	41	0	1	130.0	NaN	NaN	0.0	172	0.0	NaN	2.0	0.0	2.0	1.0
1	No age	1	2	130.0	250.0	0.0	1.0	187	0.0	3.5	0.0	0.0	2.0	NaN
5	No age	1	0	140.0	192.0	0.0	NaN	148	0.0	0.4	NaN	0.0	1.0	1.0
6	56	0	1	140.0	294.0	0.0	0.0	153	0.0	1.3	1.0	0.0	NaN	1.0
9	57	1	2	150.0	168.0	0.0	1.0	174	0.0	1.6	2.0	0.0	2.0	1.0
8	52	1	2	172.0	199.0	1.0	1.0	162	0.0	0.5	2.0	0.0	3.0	NaN
4	57	0	0	NaN	354.0	0.0	1.0	163	1.0	0.6	2.0	NaN	2.0	1.0

In [ ]: