Shubhangi_Dhikale_35

1.Write a Pandas program to replace all the NaN values with mean in a column of a DataFrame

In [1]:
```
import pandas as pd
```

In [2]:
```
df=pd.read_csv("Emp_Records.csv")
df
```

Out[2]:

	Emp ID	First Name	Age in Yrs	Weight in Kgs	Age in Company	Salary	City
0	677509	Lois	36.36	60.0	13.68	168251.0	Denver
1	940761	Brenda	47.02	60.0	9.01	51063.0	Stonewall
2	428945	Joe	54.15	68.0	NaN	50155.0	Michigantown
3	408351	Diane	NaN	51.0	NaN	NaN	Hydetown
4	193819	Benjamin	40.31	NaN	4.01	NaN	Fremont
...	...	...	...	...	...	...	...
95	639892	Jose	22.82	89.0	1.05	129774.0	Biloxi
96	704709	Harold	32.61	77.0	5.93	156194.0	Carol Stream
97	461593	Nicole	52.66	60.0	28.53	95673.0	Detroit
98	392491	Theresa	29.60	57.0	6.99	51015.0	Mc Grath
99	495141	Tammy	38.38	55.0	2.26	93650.0	Alma

100 rows × 7 columns

In [4]:
```
df["Age in Yrs"]=df["Age in Yrs"].fillna(df["Age in Yrs"].mean())
df["Weight in Kgs"]=df["Weight in Kgs"].fillna(df["Weight in Kgs"].mean())
df["Age in Company"]=df["Age in Company"].fillna(df["Age in Company"].mean())
df["Salary"]=df["Salary"].fillna(df["Salary"].mean())
df
```

Out[4]:

	Emp ID	First Name	Age in Yrs	Weight in Kgs	Age in Company	Salary	City
0	677509	Lois	36.360000	60.000000	13.68000	168251.000000	Denver
1	940761	Brenda	47.020000	60.000000	9.01000	51063.000000	Stonewall
2	428945	Joe	54.150000	68.000000	8.75625	50155.000000	Michigantown
3	408351	Diane	39.508557	51.000000	8.75625	119624.412371	Hydetown
4	193819	Benjamin	40.310000	57.653061	4.01000	119624.412371	Fremont
...	...	...	...	...	...	...	...
95	639892	Jose	22.820000	89.000000	1.05000	129774.000000	Biloxi
96	704709	Harold	32.610000	77.000000	5.93000	156194.000000	Carol Stream
97	461593	Nicole	52.660000	60.000000	28.53000	95673.000000	Detroit
98	392491	Theresa	29.600000	57.000000	6.99000	51015.000000	Mc Grath
99	495141	Tammy	38.380000	55.000000	2.26000	93650.000000	Alma

100 rows × 7 columns

2.Write a Pandas program to create and display a DataFrame from a specified dictionary data which has the index labels

In [10]:
```
import numpy as np
data={"ones":np.ones(10,dtype=int),
      "zeros":np.zeros(10,dtype=int),
      "arange":np.arange(1,11),
      "linspace":np.linspace(1,50,num=10),
      "list":[i**2 for i in range(1,11)]}
data
df=pd.DataFrame(data,index=list("ABCDEFGHIJ"))
df
```

Out[10]:

	ones	zeros	arange	linspace	list
A	1	0	1	1.000000	1
B	1	0	2	6.444444	4
C	1	0	3	11.888889	9
D	1	0	4	17.333333	16
E	1	0	5	22.777778	25
F	1	0	6	28.222222	36
G	1	0	7	33.666667	49
H	1	0	8	39.111111	64
I	1	0	9	44.555556	81
J	1	0	10	50.000000	100

3.Write a Pandas program to get the first 3 rows of a given DataFrame.

In [11]:
```
import numpy as np
data={"ones":np.ones(10,dtype=int),
      "zeros":np.zeros(10,dtype=int),
      "arange":np.arange(1,11),
      "linspace":np.linspace(1,50,num=10),
      "list":[i**2 for i in range(1,11)]}
data
df=pd.DataFrame(data,index=list("ABCDEFGHIJ"))
df
```

Out[11]:

	ones	zeros	arange	linspace	list
A	1	0	1	1.000000	1
B	1	0	2	6.444444	4
C	1	0	3	11.888889	9
D	1	0	4	17.333333	16
E	1	0	5	22.777778	25
F	1	0	6	28.222222	36
G	1	0	7	33.666667	49
H	1	0	8	39.111111	64
I	1	0	9	44.555556	81
J	1	0	10	50.000000	100

In [13]:
```
df.iloc[0:3]
```

Out[13]:

	ones	zeros	arange	linspace	list
A	1	0	1	1.000000	1
B	1	0	2	6.444444	4
C	1	0	3	11.888889	9

In [14]:
```
df[0:3]
```

Out[14]:

	ones	zeros	arange	linspace	list
A	1	0	1	1.000000	1
B	1	0	2	6.444444	4
C	1	0	3	11.888889	9

In [15]:
```
df.loc["A":"C"]
```

Out[15]:

	ones	zeros	arange	linspace	list
A	1	0	1	1.000000	1
B	1	0	2	6.444444	4
C	1	0	3	11.888889	9

In [16]:
```
df.head(3)
```

Out[16]:

	ones	zeros	arange	linspace	list
A	1	0	1	1.000000	1
B	1	0	2	6.444444	4
C	1	0	3	11.888889	9

4.Write a Pandas program to select the first 2 rows, 2 columns, and specific two columns

In [17]:
```
import numpy as np
data={"ones":np.ones(10,dtype=int),
      "zeros":np.zeros(10,dtype=int),
      "arange":np.arange(1,11),
      "linspace":np.linspace(1,50,num=10),
      "list":[i**2 for i in range(1,11)]}
data
df=pd.DataFrame(data,index=list("ABCDEFGHIJ"))
df
```

Out[17]:

	ones	zeros	arange	linspace	list
A	1	0	1	1.000000	1
B	1	0	2	6.444444	4
C	1	0	3	11.888889	9
D	1	0	4	17.333333	16
E	1	0	5	22.777778	25
F	1	0	6	28.222222	36
G	1	0	7	33.666667	49
H	1	0	8	39.111111	64
I	1	0	9	44.555556	81
J	1	0	10	50.000000	100

In [18]:
```
df[0:2]     #first 2 rows
```

Out[18]:

	ones	zeros	arange	linspace	list
A	1	0	1	1.000000	1
B	1	0	2	6.444444	4

In [19]:
```
df.iloc[0:2]
```

Out[19]:

	ones	zeros	arange	linspace	list
A	1	0	1	1.000000	1
B	1	0	2	6.444444	4

In [27]:
```
df.iloc[0:9,0:2]
```

Out[27]:

	ones	zeros
A	1	0
B	1	0
C	1	0
D	1	0
E	1	0
F	1	0
G	1	0
H	1	0
I	1	0

In [28]:
```
df.loc[:,"ones":"zeros"]
```

Out[28]:

	ones	zeros
A	1	0
B	1	0
C	1	0
D	1	0
E	1	0
F	1	0
G	1	0
H	1	0
I	1	0
J	1	0

In [29]:
```
df[["arange","list"]]    #two specific columns
```

Out[29]:

	arange	list
A	1	1
B	2	4
C	3	9
D	4	16
E	5	25
F	6	36
G	7	49
H	8	64
I	9	81
J	10	100

In [31]:
```
df.loc[:,["arange","list"]]
```

Out[31]:

	arange	list
A	1	1
B	2	4
C	3	9
D	4	16
E	5	25
F	6	36
G	7	49
H	8	64
I	9	81
J	10	100

5.Write a Pandas program to select the specified columns and rows from a given DataFrame

In [32]:
```
import numpy as np
data={"ones":np.ones(10,dtype=int),
      "zeros":np.zeros(10,dtype=int),
      "arange":np.arange(1,11),
      "linspace":np.linspace(1,50,num=10),
      "list":[i**2 for i in range(1,11)]}
data
df=pd.DataFrame(data,index=list("ABCDEFGHIJ"))
df
```

Out[32]:

	ones	zeros	arange	linspace	list
A	1	0	1	1.000000	1
B	1	0	2	6.444444	4
C	1	0	3	11.888889	9
D	1	0	4	17.333333	16
E	1	0	5	22.777778	25
F	1	0	6	28.222222	36
G	1	0	7	33.666667	49
H	1	0	8	39.111111	64
I	1	0	9	44.555556	81
J	1	0	10	50.000000	100

In [37]:
```
df.iloc[0:5,0:3]
```

Out[37]:

	ones	zeros	arange
A	1	0	1
B	1	0	2
C	1	0	3
D	1	0	4
E	1	0	5

In [38]:
```
df.loc["A":"G","zeros":"linspace"]
```

Out[38]:

	zeros	arange	linspace
A	0	1	1.000000
B	0	2	6.444444
C	0	3	11.888889
D	0	4	17.333333
E	0	5	22.777778
F	0	6	28.222222
G	0	7	33.666667

6.Write a Pandas program to detect missing values of a given DataFrame. Display True or False

In [40]:
```
df=pd.read_csv("Emp_Records.csv")
df
```

Out[40]:

	Emp ID	First Name	Age in Yrs	Weight in Kgs	Age in Company	Salary	City
0	677509	Lois	36.36	60.0	13.68	168251.0	Denver
1	940761	Brenda	47.02	60.0	9.01	51063.0	Stonewall
2	428945	Joe	54.15	68.0	NaN	50155.0	Michigantown
3	408351	Diane	NaN	51.0	NaN	NaN	Hydetown
4	193819	Benjamin	40.31	NaN	4.01	NaN	Fremont
...	...	...	...	...	...	...	...
95	639892	Jose	22.82	89.0	1.05	129774.0	Biloxi
96	704709	Harold	32.61	77.0	5.93	156194.0	Carol Stream
97	461593	Nicole	52.66	60.0	28.53	95673.0	Detroit
98	392491	Theresa	29.60	57.0	6.99	51015.0	Mc Grath
99	495141	Tammy	38.38	55.0	2.26	93650.0	Alma

100 rows × 7 columns

In [41]:
```
df.isnull()
```

Out[41]:

	Emp ID	First Name	Age in Yrs	Weight in Kgs	Age in Company	Salary	City
0	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False
2	False	False	False	False	True	False	False
3	False	False	True	False	True	True	False
4	False	False	False	True	False	True	False
...	...	...	...	...	...	...	...
95	False	False	False	False	False	False	False
96	False	False	False	False	False	False	False
97	False	False	False	False	False	False	False
98	False	False	False	False	False	False	False
99	False	False	False	False	False	False	False

100 rows × 7 columns

7.Write a Pandas program to count the number of missing values in each column of a given DataFrame

In [42]:
```
df=pd.read_csv("Emp_Records.csv")
df
```

Out[42]:

	Emp ID	First Name	Age in Yrs	Weight in Kgs	Age in Company	Salary	City
0	677509	Lois	36.36	60.0	13.68	168251.0	Denver
1	940761	Brenda	47.02	60.0	9.01	51063.0	Stonewall
2	428945	Joe	54.15	68.0	NaN	50155.0	Michigantown
3	408351	Diane	NaN	51.0	NaN	NaN	Hydetown
4	193819	Benjamin	40.31	NaN	4.01	NaN	Fremont
...	...	...	...	...	...	...	...
95	639892	Jose	22.82	89.0	1.05	129774.0	Biloxi
96	704709	Harold	32.61	77.0	5.93	156194.0	Carol Stream
97	461593	Nicole	52.66	60.0	28.53	95673.0	Detroit
98	392491	Theresa	29.60	57.0	6.99	51015.0	Mc Grath
99	495141	Tammy	38.38	55.0	2.26	93650.0	Alma

100 rows × 7 columns

In [43]:
```
df.isna().sum()
```

Out[43]:
```
Emp ID            0
First Name        0
Age in Yrs        3
Weight in Kgs     2
Age in Company    4
Salary            3
City              0
dtype: int64
```

In [44]:
```
df.isnull().sum()
```

Out[44]:
```
Emp ID            0
First Name        0
Age in Yrs        3
Weight in Kgs     2
Age in Company    4
Salary            3
City              0
dtype: int64
```

8.Write a Pandas program to find and replace the missing values in a given DataFrame which do not have any valuable information.

○ Example: ○ Missing values: ?, -- ○ Replace those values with NaN

In [ ]:

9.Write a Pandas program to drop the rows where at least one element is missing in a given DataFrame

In [45]:
```
df=pd.read_csv("Emp_Records.csv")
df
```

Out[45]:

	Emp ID	First Name	Age in Yrs	Weight in Kgs	Age in Company	Salary	City
0	677509	Lois	36.36	60.0	13.68	168251.0	Denver
1	940761	Brenda	47.02	60.0	9.01	51063.0	Stonewall
2	428945	Joe	54.15	68.0	NaN	50155.0	Michigantown
3	408351	Diane	NaN	51.0	NaN	NaN	Hydetown
4	193819	Benjamin	40.31	NaN	4.01	NaN	Fremont
...	...	...	...	...	...	...	...
95	639892	Jose	22.82	89.0	1.05	129774.0	Biloxi
96	704709	Harold	32.61	77.0	5.93	156194.0	Carol Stream
97	461593	Nicole	52.66	60.0	28.53	95673.0	Detroit
98	392491	Theresa	29.60	57.0	6.99	51015.0	Mc Grath
99	495141	Tammy	38.38	55.0	2.26	93650.0	Alma

100 rows × 7 columns

In [46]:
```
df.dropna()      #at least one value is missing
```

Out[46]:

	Emp ID	First Name	Age in Yrs	Weight in Kgs	Age in Company	Salary	City
0	677509	Lois	36.36	60.0	13.68	168251.0	Denver
1	940761	Brenda	47.02	60.0	9.01	51063.0	Stonewall
8	477616	Frances	58.18	42.0	23.27	121587.0	Delmita
10	231469	Ralph	42.50	80.0	8.29	118457.0	Sabetha
11	153989	Jack	22.21	61.0	0.56	82965.0	Las Vegas
...	...	...	...	...	...	...	...
95	639892	Jose	22.82	89.0	1.05	129774.0	Biloxi
96	704709	Harold	32.61	77.0	5.93	156194.0	Carol Stream
97	461593	Nicole	52.66	60.0	28.53	95673.0	Detroit
98	392491	Theresa	29.60	57.0	6.99	51015.0	Mc Grath
99	495141	Tammy	38.38	55.0	2.26	93650.0	Alma

93 rows × 7 columns

In [47]:
```
df.dropna(thresh=6)
```

Out[47]:

	Emp ID	First Name	Age in Yrs	Weight in Kgs	Age in Company	Salary	City
0	677509	Lois	36.36	60.0	13.68	168251.0	Denver
1	940761	Brenda	47.02	60.0	9.01	51063.0	Stonewall
2	428945	Joe	54.15	68.0	NaN	50155.0	Michigantown
5	499687	Patrick	34.86	58.0	12.02	NaN	Macksburg
6	539712	Nancy	NaN	50.0	0.87	98189.0	Atlanta
...	...	...	...	...	...	...	...
95	639892	Jose	22.82	89.0	1.05	129774.0	Biloxi
96	704709	Harold	32.61	77.0	5.93	156194.0	Carol Stream
97	461593	Nicole	52.66	60.0	28.53	95673.0	Detroit
98	392491	Theresa	29.60	57.0	6.99	51015.0	Mc Grath
99	495141	Tammy	38.38	55.0	2.26	93650.0	Alma

97 rows × 7 columns

10.Write a Pandas program to drop the rows where all elements are missing in a given DataFrame

In [48]:
```
df=pd.read_csv("Emp_Records.csv")
df
```

Out[48]:

	Emp ID	First Name	Age in Yrs	Weight in Kgs	Age in Company	Salary	City
0	677509	Lois	36.36	60.0	13.68	168251.0	Denver
1	940761	Brenda	47.02	60.0	9.01	51063.0	Stonewall
2	428945	Joe	54.15	68.0	NaN	50155.0	Michigantown
3	408351	Diane	NaN	51.0	NaN	NaN	Hydetown
4	193819	Benjamin	40.31	NaN	4.01	NaN	Fremont
...	...	...	...	...	...	...	...
95	639892	Jose	22.82	89.0	1.05	129774.0	Biloxi
96	704709	Harold	32.61	77.0	5.93	156194.0	Carol Stream
97	461593	Nicole	52.66	60.0	28.53	95673.0	Detroit
98	392491	Theresa	29.60	57.0	6.99	51015.0	Mc Grath
99	495141	Tammy	38.38	55.0	2.26	93650.0	Alma

100 rows × 7 columns

In [49]:
```
df.dropna(axis=0,how="all")
```

Out[49]:

	Emp ID	First Name	Age in Yrs	Weight in Kgs	Age in Company	Salary	City
0	677509	Lois	36.36	60.0	13.68	168251.0	Denver
1	940761	Brenda	47.02	60.0	9.01	51063.0	Stonewall
2	428945	Joe	54.15	68.0	NaN	50155.0	Michigantown
3	408351	Diane	NaN	51.0	NaN	NaN	Hydetown
4	193819	Benjamin	40.31	NaN	4.01	NaN	Fremont
...	...	...	...	...	...	...	...
95	639892	Jose	22.82	89.0	1.05	129774.0	Biloxi
96	704709	Harold	32.61	77.0	5.93	156194.0	Carol Stream
97	461593	Nicole	52.66	60.0	28.53	95673.0	Detroit
98	392491	Theresa	29.60	57.0	6.99	51015.0	Mc Grath
99	495141	Tammy	38.38	55.0	2.26	93650.0	Alma

100 rows × 7 columns

In [54]:
```
df=pd.DataFrame({"list1":[1,2,3,4,5],
                 "ones":np.nan,
                 "names":["X","Y","Z","W","R"]})
df
```

Out[54]:

	list1	ones	names
0	1	NaN	X
1	2	NaN	Y
2	3	NaN	Z
3	4	NaN	W
4	5	NaN	R

In [55]:
```
df.dropna(axis=1,how="all")
```

Out[55]:

	list1	names
0	1	X
1	2	Y
2	3	Z
3	4	W
4	5	R

11.Write a Pandas program to keep the rows with at least 2 NaN values in a given DataFrame

In [56]:
```
df=pd.read_csv("Emp_Records.csv")
df
```

Out[56]:

	Emp ID	First Name	Age in Yrs	Weight in Kgs	Age in Company	Salary	City
0	677509	Lois	36.36	60.0	13.68	168251.0	Denver
1	940761	Brenda	47.02	60.0	9.01	51063.0	Stonewall
2	428945	Joe	54.15	68.0	NaN	50155.0	Michigantown
3	408351	Diane	NaN	51.0	NaN	NaN	Hydetown
4	193819	Benjamin	40.31	NaN	4.01	NaN	Fremont
...	...	...	...	...	...	...	...
95	639892	Jose	22.82	89.0	1.05	129774.0	Biloxi
96	704709	Harold	32.61	77.0	5.93	156194.0	Carol Stream
97	461593	Nicole	52.66	60.0	28.53	95673.0	Detroit
98	392491	Theresa	29.60	57.0	6.99	51015.0	Mc Grath
99	495141	Tammy	38.38	55.0	2.26	93650.0	Alma

100 rows × 7 columns

In [59]:
```
df.dropna(thresh=5)
```

Out[59]:

	Emp ID	First Name	Age in Yrs	Weight in Kgs	Age in Company	Salary	City
0	677509	Lois	36.36	60.0	13.68	168251.0	Denver
1	940761	Brenda	47.02	60.0	9.01	51063.0	Stonewall
2	428945	Joe	54.15	68.0	NaN	50155.0	Michigantown
4	193819	Benjamin	40.31	NaN	4.01	NaN	Fremont
5	499687	Patrick	34.86	58.0	12.02	NaN	Macksburg
...	...	...	...	...	...	...	...
95	639892	Jose	22.82	89.0	1.05	129774.0	Biloxi
96	704709	Harold	32.61	77.0	5.93	156194.0	Carol Stream
97	461593	Nicole	52.66	60.0	28.53	95673.0	Detroit
98	392491	Theresa	29.60	57.0	6.99	51015.0	Mc Grath
99	495141	Tammy	38.38	55.0	2.26	93650.0	Alma

98 rows × 7 columns

12.Write a Pandas program to keep the valid entries of a given DataFrame.

In [60]:
```
df
```

Out[60]:

	Emp ID	First Name	Age in Yrs	Weight in Kgs	Age in Company	Salary	City
0	677509	Lois	36.36	60.0	13.68	168251.0	Denver
1	940761	Brenda	47.02	60.0	9.01	51063.0	Stonewall
2	428945	Joe	54.15	68.0	NaN	50155.0	Michigantown
3	408351	Diane	NaN	51.0	NaN	NaN	Hydetown
4	193819	Benjamin	40.31	NaN	4.01	NaN	Fremont
...	...	...	...	...	...	...	...
95	639892	Jose	22.82	89.0	1.05	129774.0	Biloxi
96	704709	Harold	32.61	77.0	5.93	156194.0	Carol Stream
97	461593	Nicole	52.66	60.0	28.53	95673.0	Detroit
98	392491	Theresa	29.60	57.0	6.99	51015.0	Mc Grath
99	495141	Tammy	38.38	55.0	2.26	93650.0	Alma

100 rows × 7 columns

In [61]:
```
df.dropna()
```

Out[61]:

	Emp ID	First Name	Age in Yrs	Weight in Kgs	Age in Company	Salary	City
0	677509	Lois	36.36	60.0	13.68	168251.0	Denver
1	940761	Brenda	47.02	60.0	9.01	51063.0	Stonewall
8	477616	Frances	58.18	42.0	23.27	121587.0	Delmita
10	231469	Ralph	42.50	80.0	8.29	118457.0	Sabetha
11	153989	Jack	22.21	61.0	0.56	82965.0	Las Vegas
...	...	...	...	...	...	...	...
95	639892	Jose	22.82	89.0	1.05	129774.0	Biloxi
96	704709	Harold	32.61	77.0	5.93	156194.0	Carol Stream
97	461593	Nicole	52.66	60.0	28.53	95673.0	Detroit
98	392491	Theresa	29.60	57.0	6.99	51015.0	Mc Grath
99	495141	Tammy	38.38	55.0	2.26	93650.0	Alma

93 rows × 7 columns

13.Write a Pandas program to calculate the total number of missing values in a DataFrame

In [62]:
```
df=pd.read_csv("Emp_Records.csv")
df
```

Out[62]:

	Emp ID	First Name	Age in Yrs	Weight in Kgs	Age in Company	Salary	City
0	677509	Lois	36.36	60.0	13.68	168251.0	Denver
1	940761	Brenda	47.02	60.0	9.01	51063.0	Stonewall
2	428945	Joe	54.15	68.0	NaN	50155.0	Michigantown
3	408351	Diane	NaN	51.0	NaN	NaN	Hydetown
4	193819	Benjamin	40.31	NaN	4.01	NaN	Fremont
...	...	...	...	...	...	...	...
95	639892	Jose	22.82	89.0	1.05	129774.0	Biloxi
96	704709	Harold	32.61	77.0	5.93	156194.0	Carol Stream
97	461593	Nicole	52.66	60.0	28.53	95673.0	Detroit
98	392491	Theresa	29.60	57.0	6.99	51015.0	Mc Grath
99	495141	Tammy	38.38	55.0	2.26	93650.0	Alma

100 rows × 7 columns

In [63]:
```
df.isnull().sum()
```

Out[63]:
```
Emp ID            0
First Name        0
Age in Yrs        3
Weight in Kgs     2
Age in Company    4
Salary            3
City              0
dtype: int64
```

In [64]:
```
df.isna().values.sum()
```

Out[64]:
```
12
```

14.Write a Pandas program to replace NaNs with a single constant value in specified columns in a DataFrame

In [65]:
```
df
```

Out[65]:

	Emp ID	First Name	Age in Yrs	Weight in Kgs	Age in Company	Salary	City
0	677509	Lois	36.36	60.0	13.68	168251.0	Denver
1	940761	Brenda	47.02	60.0	9.01	51063.0	Stonewall
2	428945	Joe	54.15	68.0	NaN	50155.0	Michigantown
3	408351	Diane	NaN	51.0	NaN	NaN	Hydetown
4	193819	Benjamin	40.31	NaN	4.01	NaN	Fremont
...	...	...	...	...	...	...	...
95	639892	Jose	22.82	89.0	1.05	129774.0	Biloxi
96	704709	Harold	32.61	77.0	5.93	156194.0	Carol Stream
97	461593	Nicole	52.66	60.0	28.53	95673.0	Detroit
98	392491	Theresa	29.60	57.0	6.99	51015.0	Mc Grath
99	495141	Tammy	38.38	55.0	2.26	93650.0	Alma

100 rows × 7 columns

In [67]:
```
df["Salary"]=df["Salary"].fillna(1000)
df
```

Out[67]:

	Emp ID	First Name	Age in Yrs	Weight in Kgs	Age in Company	Salary	City
0	677509	Lois	36.36	60.0	13.68	168251.0	Denver
1	940761	Brenda	47.02	60.0	9.01	51063.0	Stonewall
2	428945	Joe	54.15	68.0	NaN	50155.0	Michigantown
3	408351	Diane	NaN	51.0	NaN	1000.0	Hydetown
4	193819	Benjamin	40.31	NaN	4.01	1000.0	Fremont
...	...	...	...	...	...	...	...
95	639892	Jose	22.82	89.0	1.05	129774.0	Biloxi
96	704709	Harold	32.61	77.0	5.93	156194.0	Carol Stream
97	461593	Nicole	52.66	60.0	28.53	95673.0	Detroit
98	392491	Theresa	29.60	57.0	6.99	51015.0	Mc Grath
99	495141	Tammy	38.38	55.0	2.26	93650.0	Alma

100 rows × 7 columns

In [69]:
```
df["Weight in Kgs"]=df["Weight in Kgs"].fillna(56)
df
```

Out[69]:

	Emp ID	First Name	Age in Yrs	Weight in Kgs	Age in Company	Salary	City
0	677509	Lois	36.36	60.0	13.68	168251.0	Denver
1	940761	Brenda	47.02	60.0	9.01	51063.0	Stonewall
2	428945	Joe	54.15	68.0	NaN	50155.0	Michigantown
3	408351	Diane	NaN	51.0	NaN	1000.0	Hydetown
4	193819	Benjamin	40.31	56.0	4.01	1000.0	Fremont
...	...	...	...	...	...	...	...
95	639892	Jose	22.82	89.0	1.05	129774.0	Biloxi
96	704709	Harold	32.61	77.0	5.93	156194.0	Carol Stream
97	461593	Nicole	52.66	60.0	28.53	95673.0	Detroit
98	392491	Theresa	29.60	57.0	6.99	51015.0	Mc Grath
99	495141	Tammy	38.38	55.0	2.26	93650.0	Alma

100 rows × 7 columns

15.Write a Pandas program to replace NaNs with the median or mean of the specified columns in a given DataFrame

In [70]:
```
df=pd.read_csv("Emp_Records.csv")
df
```

Out[70]:

	Emp ID	First Name	Age in Yrs	Weight in Kgs	Age in Company	Salary	City
0	677509	Lois	36.36	60.0	13.68	168251.0	Denver
1	940761	Brenda	47.02	60.0	9.01	51063.0	Stonewall
2	428945	Joe	54.15	68.0	NaN	50155.0	Michigantown
3	408351	Diane	NaN	51.0	NaN	NaN	Hydetown
4	193819	Benjamin	40.31	NaN	4.01	NaN	Fremont
...	...	...	...	...	...	...	...
95	639892	Jose	22.82	89.0	1.05	129774.0	Biloxi
96	704709	Harold	32.61	77.0	5.93	156194.0	Carol Stream
97	461593	Nicole	52.66	60.0	28.53	95673.0	Detroit
98	392491	Theresa	29.60	57.0	6.99	51015.0	Mc Grath
99	495141	Tammy	38.38	55.0	2.26	93650.0	Alma

100 rows × 7 columns

In [72]:
```
df["Salary"]=df["Salary"].fillna(df["Salary"].mean())
df
```

Out[72]:

	Emp ID	First Name	Age in Yrs	Weight in Kgs	Age in Company	Salary	City
0	677509	Lois	36.36	60.0	13.68	168251.000000	Denver
1	940761	Brenda	47.02	60.0	9.01	51063.000000	Stonewall
2	428945	Joe	54.15	68.0	NaN	50155.000000	Michigantown
3	408351	Diane	NaN	51.0	NaN	119624.412371	Hydetown
4	193819	Benjamin	40.31	NaN	4.01	119624.412371	Fremont
...	...	...	...	...	...	...	...
95	639892	Jose	22.82	89.0	1.05	129774.000000	Biloxi
96	704709	Harold	32.61	77.0	5.93	156194.000000	Carol Stream
97	461593	Nicole	52.66	60.0	28.53	95673.000000	Detroit
98	392491	Theresa	29.60	57.0	6.99	51015.000000	Mc Grath
99	495141	Tammy	38.38	55.0	2.26	93650.000000	Alma

100 rows × 7 columns

In [73]:
```
df["Salary"].mean()
```

Out[73]:
```
119624.412371134
```

In [74]:
```
df["Age in Company"]=df["Age in Company"].fillna(df["Age in Company"].median())
df
```

Out[74]:

	Emp ID	First Name	Age in Yrs	Weight in Kgs	Age in Company	Salary	City
0	677509	Lois	36.36	60.0	13.680	168251.000000	Denver
1	940761	Brenda	47.02	60.0	9.010	51063.000000	Stonewall
2	428945	Joe	54.15	68.0	6.435	50155.000000	Michigantown
3	408351	Diane	NaN	51.0	6.435	119624.412371	Hydetown
4	193819	Benjamin	40.31	NaN	4.010	119624.412371	Fremont
...	...	...	...	...	...	...	...
95	639892	Jose	22.82	89.0	1.050	129774.000000	Biloxi
96	704709	Harold	32.61	77.0	5.930	156194.000000	Carol Stream
97	461593	Nicole	52.66	60.0	28.530	95673.000000	Detroit
98	392491	Theresa	29.60	57.0	6.990	51015.000000	Mc Grath
99	495141	Tammy	38.38	55.0	2.260	93650.000000	Alma

100 rows × 7 columns

In [75]:
```
df["Age in Company"].median()
```

Out[75]:
```
6.4350000000000005
```

16.Write a Pandas program to find the Indexes of missing values in a given DataFrame

In [76]:
```
df=pd.read_csv("Emp_Records.csv")
df
```

Out[76]:

	Emp ID	First Name	Age in Yrs	Weight in Kgs	Age in Company	Salary	City
0	677509	Lois	36.36	60.0	13.68	168251.0	Denver
1	940761	Brenda	47.02	60.0	9.01	51063.0	Stonewall
2	428945	Joe	54.15	68.0	NaN	50155.0	Michigantown
3	408351	Diane	NaN	51.0	NaN	NaN	Hydetown
4	193819	Benjamin	40.31	NaN	4.01	NaN	Fremont
...	...	...	...	...	...	...	...
95	639892	Jose	22.82	89.0	1.05	129774.0	Biloxi
96	704709	Harold	32.61	77.0	5.93	156194.0	Carol Stream
97	461593	Nicole	52.66	60.0	28.53	95673.0	Detroit
98	392491	Theresa	29.60	57.0	6.99	51015.0	Mc Grath
99	495141	Tammy	38.38	55.0	2.26	93650.0	Alma

100 rows × 7 columns

In [77]:
```
df.isnull().sum()
```

Out[77]:
```
Emp ID            0
First Name        0
Age in Yrs        3
Weight in Kgs     2
Age in Company    4
Salary            3
City              0
dtype: int64
```

In [78]:
```
df["Salary"].isnull().to_numpy().nonzero()   #this shows missing value is in 3,4,
```

Out[78]:
```
(array([3, 4, 5], dtype=int64),)
```

In [79]:
```
df['Age in Yrs'].isnull().to_numpy().nonzero()
```

Out[79]:
```
(array([3, 6, 9], dtype=int64),)
```

17.Write a Pandas program to select rows from a given DataFrame based on values in some columns

In [81]:
```
df=pd.read_csv("movies_data.csv")
df
```

Out[81]:

	star_rating	title	content_rating	genre	duration	actors_list
0	9.3	The Shawshank Redemption	R	Crime	142	[u'Tim Robbins', u'Morgan Freeman', u'Bob Gunt...
1	9.2	The Godfather	R	Crime	175	[u'Marlon Brando', u'Al Pacino', u'James Caan']
2	9.1	The Godfather: Part II	R	Crime	200	[u'Al Pacino', u'Robert De Niro', u'Robert Duv...
3	9.0	The Dark Knight	PG-13	Action	152	[u'Christian Bale', u'Heath Ledger', u'Aaron E...
4	8.9	Pulp Fiction	R	Crime	154	[u'John Travolta', u'Uma Thurman', u'Samuel L....
...	...	...	...	...	...	...
974	7.4	Tootsie	PG	Comedy	116	[u'Dustin Hoffman', u'Jessica Lange', u'Teri G...
975	7.4	Back to the Future Part III	PG	Adventure	118	[u'Michael J. Fox', u'Christopher Lloyd', u'Ma...
976	7.4	Master and Commander: The Far Side of the World	PG-13	Action	138	[u'Russell Crowe', u'Paul Bettany', u'Billy Bo...
977	7.4	Poltergeist	PG	Horror	114	[u'JoBeth Williams', u"Heather O'Rourke", u'Cr...
978	7.4	Wall Street	R	Crime	126	[u'Charlie Sheen', u'Michael Douglas', u'Tamar...

979 rows × 6 columns

In [82]:
```
df.loc[df["content_rating"]=="PG-13"]
```

Out[82]:

	star_rating	title	content_rating	genre	duration	actors_list
3	9.0	The Dark Knight	PG-13	Action	152	[u'Christian Bale', u'Heath Ledger', u'Aaron E...
7	8.9	The Lord of the Rings: The Return of the King	PG-13	Adventure	201	[u'Elijah Wood', u'Viggo Mortensen', u'Ian McK...
10	8.8	The Lord of the Rings: The Fellowship of the Ring	PG-13	Adventure	178	[u'Elijah Wood', u'Ian McKellen', u'Orlando Bl...
11	8.8	Inception	PG-13	Action	148	[u'Leonardo DiCaprio', u'Joseph Gordon-Levitt'...
13	8.8	Forrest Gump	PG-13	Drama	142	[u'Tom Hanks', u'Robin Wright', u'Gary Sinise']
...	...	...	...	...	...	...
964	7.4	Lincoln	PG-13	Biography	150	[u'Daniel Day-Lewis', u'Sally Field', u'David ...
965	7.4	Limitless	PG-13	Mystery	105	[u'Bradley Cooper', u'Anna Friel', u'Abbie Cor...
966	7.4	The Simpsons Movie	PG-13	Animation	87	[u'Dan Castellaneta', u'Julie Kavner', u'Nancy...
973	7.4	The Cider House Rules	PG-13	Drama	126	[u'Tobey Maguire', u'Charlize Theron', u'Micha...
976	7.4	Master and Commander: The Far Side of the World	PG-13	Action	138	[u'Russell Crowe', u'Paul Bettany', u'Billy Bo...

189 rows × 6 columns

In [83]:
```
df.loc[df["genre"]=="Action"]
```

Out[83]:

	star_rating	title	content_rating	genre	duration	actors_list
3	9.0	The Dark Knight	PG-13	Action	152	[u'Christian Bale', u'Heath Ledger', u'Aaron E...
11	8.8	Inception	PG-13	Action	148	[u'Leonardo DiCaprio', u'Joseph Gordon-Levitt'...
12	8.8	Star Wars: Episode V - The Empire Strikes Back	PG	Action	124	[u'Mark Hamill', u'Harrison Ford', u'Carrie Fi...
19	8.7	Star Wars	PG	Action	121	[u'Mark Hamill', u'Harrison Ford', u'Carrie Fi...
20	8.7	The Matrix	R	Action	136	[u'Keanu Reeves', u'Laurence Fishburne', u'Car...
...	...	...	...	...	...	...
918	7.5	Running Scared	R	Action	122	[u'Paul Walker', u'Cameron Bright', u'Chazz Pa...
954	7.4	X-Men	PG-13	Action	104	[u'Patrick Stewart', u'Hugh Jackman', u'Ian Mc...
963	7.4	La Femme Nikita	R	Action	118	[u'Anne Parillaud', u'Marc Duret', u'Patrick F...
967	7.4	The Rock	R	Action	136	[u'Sean Connery', u'Nicolas Cage', u'Ed Harris']
976	7.4	Master and Commander: The Far Side of the World	PG-13	Action	138	[u'Russell Crowe', u'Paul Bettany', u'Billy Bo...

136 rows × 6 columns

18.Write a Pandas program to change the order of a DataFrame columns

In [84]:
```
df=pd.read_csv("movies_data.csv")
df
```

Out[84]:

	star_rating	title	content_rating	genre	duration	actors_list
0	9.3	The Shawshank Redemption	R	Crime	142	[u'Tim Robbins', u'Morgan Freeman', u'Bob Gunt...
1	9.2	The Godfather	R	Crime	175	[u'Marlon Brando', u'Al Pacino', u'James Caan']
2	9.1	The Godfather: Part II	R	Crime	200	[u'Al Pacino', u'Robert De Niro', u'Robert Duv...
3	9.0	The Dark Knight	PG-13	Action	152	[u'Christian Bale', u'Heath Ledger', u'Aaron E...
4	8.9	Pulp Fiction	R	Crime	154	[u'John Travolta', u'Uma Thurman', u'Samuel L....
...	...	...	...	...	...	...
974	7.4	Tootsie	PG	Comedy	116	[u'Dustin Hoffman', u'Jessica Lange', u'Teri G...
975	7.4	Back to the Future Part III	PG	Adventure	118	[u'Michael J. Fox', u'Christopher Lloyd', u'Ma...
976	7.4	Master and Commander: The Far Side of the World	PG-13	Action	138	[u'Russell Crowe', u'Paul Bettany', u'Billy Bo...
977	7.4	Poltergeist	PG	Horror	114	[u'JoBeth Williams', u"Heather O'Rourke", u'Cr...
978	7.4	Wall Street	R	Crime	126	[u'Charlie Sheen', u'Michael Douglas', u'Tamar...

979 rows × 6 columns

In [85]:
```
df.sort_index(axis=1)
```

Out[85]:

	actors_list	content_rating	duration	genre	star_rating	title
0	[u'Tim Robbins', u'Morgan Freeman', u'Bob Gunt...	R	142	Crime	9.3	The Shawshank Redemption
1	[u'Marlon Brando', u'Al Pacino', u'James Caan']	R	175	Crime	9.2	The Godfather
2	[u'Al Pacino', u'Robert De Niro', u'Robert Duv...	R	200	Crime	9.1	The Godfather: Part II
3	[u'Christian Bale', u'Heath Ledger', u'Aaron E...	PG-13	152	Action	9.0	The Dark Knight
4	[u'John Travolta', u'Uma Thurman', u'Samuel L....	R	154	Crime	8.9	Pulp Fiction
...	...	...	...	...	...	...
974	[u'Dustin Hoffman', u'Jessica Lange', u'Teri G...	PG	116	Comedy	7.4	Tootsie
975	[u'Michael J. Fox', u'Christopher Lloyd', u'Ma...	PG	118	Adventure	7.4	Back to the Future Part III
976	[u'Russell Crowe', u'Paul Bettany', u'Billy Bo...	PG-13	138	Action	7.4	Master and Commander: The Far Side of the World
977	[u'JoBeth Williams', u"Heather O'Rourke", u'Cr...	PG	114	Horror	7.4	Poltergeist
978	[u'Charlie Sheen', u'Michael Douglas', u'Tamar...	R	126	Crime	7.4	Wall Street

979 rows × 6 columns

19.Write a Pandas program to add one row in an existing DataFrame

In [87]:
```
df=pd.DataFrame({"list1":[2,4,6,8,10],
                 "list2":["A","B","C","D","E"]})
df
```

Out[87]:

	list1	list2
0	2	A
1	4	B
2	6	C
3	8	D
4	10	E

In [90]:
```
df.loc[5]=[12,"F"]
df
```

Out[90]:

	list1	list2
0	2	A
1	4	B
2	6	C
3	8	D
4	10	E
5	12	F

20.Write a Pandas program to delete DataFrame row(s) based on a given column value

In [92]:
```
dict1={"list1":[i for i in range(1,11)],
       "random":np.random.randint(1,20,size=10),
       "lnspace":np.linspace(1,50,num=10),
       "zeros":np.zeros(10,dtype=int)}
dict1
df=pd.DataFrame(dict1)
df
```

Out[92]:

	list1	random	lnspace	zeros
0	1	17	1.000000	0
1	2	16	6.444444	0
2	3	3	11.888889	0
3	4	11	17.333333	0
4	5	1	22.777778	0
5	6	10	28.222222	0
6	7	16	33.666667	0
7	8	3	39.111111	0
8	9	15	44.555556	0
9	10	1	50.000000	0

In [93]:
```
df=df[df["random"]!=1]
df
```

Out[93]:

	list1	random	lnspace	zeros
0	1	17	1.000000	0
1	2	16	6.444444	0
2	3	3	11.888889	0
3	4	11	17.333333	0
5	6	10	28.222222	0
6	7	16	33.666667	0
7	8	3	39.111111	0
8	9	15	44.555556	0

In [96]:
```
df=df[df["lnspace"]>30]
df
```

Out[96]:

	list1	random	lnspace	zeros
6	7	16	33.666667	0
7	8	3	39.111111	0
8	9	15	44.555556	0

21.Write a Pandas program to select a row of series/DataFrame by given integer index

In [97]:
```
df=pd.read_csv("titanic.csv")
df
```

Out[97]:

	PassengerId	Survived	Pclass	Name	Gender	Age	SibSp	Parch	Ticket	Fare	C
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	
...	...	...	...	...	...	...	...	...	...	...	
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.4500	
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500	

891 rows × 12 columns

In [98]:
```
df.iloc[2:4]
```

Out[98]:

	PassengerId	Survived	Pclass	Name	Gender	Age	SibSp	Parch	Ticket	Fare	Cabi
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.925	Nal
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.100	C12

In [99]:
```
df.loc[3:6,"Name":"Gender"]
```

Out[99]:

	Name	Gender
3	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female
4	Allen, Mr. William Henry	male
5	Moran, Mr. James	male
6	McCarthy, Mr. Timothy J	NaN

In [100]:
```
df[1:8]
```

Out[100]:

	PassengerId	Survived	Pclass	Name	Gender	Age	SibSp	Parch	Ticket	Fare	Cal
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	N
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C1
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	N
5	6	0	3	Moran, Mr. James	male	NaN	0	0	330877	8.4583	N
6	7	0	1	McCarthy, Mr. Timothy J	NaN	54.0	0	0	17463	51.8625	E
7	8	0	3	Palsson, Master. Gosta Leonard	NaN	2.0	3	1	349909	21.0750	N

22.Write a Pandas program to get the length of the string present in a given column in a DataFrame

In [101]:
```
df=pd.read_csv("movies_data.csv")
df
```

Out[101]:

	star_rating	title	content_rating	genre	duration	actors_list
0	9.3	The Shawshank Redemption	R	Crime	142	[u'Tim Robbins', u'Morgan Freeman', u'Bob Gunt...
1	9.2	The Godfather	R	Crime	175	[u'Marlon Brando', u'Al Pacino', u'James Caan']
2	9.1	The Godfather: Part II	R	Crime	200	[u'Al Pacino', u'Robert De Niro', u'Robert Duv...
3	9.0	The Dark Knight	PG-13	Action	152	[u'Christian Bale', u'Heath Ledger', u'Aaron E...
4	8.9	Pulp Fiction	R	Crime	154	[u'John Travolta', u'Uma Thurman', u'Samuel L....
...	...	...	...	...	...	...
974	7.4	Tootsie	PG	Comedy	116	[u'Dustin Hoffman', u'Jessica Lange', u'Teri G...
975	7.4	Back to the Future Part III	PG	Adventure	118	[u'Michael J. Fox', u'Christopher Lloyd', u'Ma...
976	7.4	Master and Commander: The Far Side of the World	PG-13	Action	138	[u'Russell Crowe', u'Paul Bettany', u'Billy Bo...
977	7.4	Poltergeist	PG	Horror	114	[u'JoBeth Williams', u"Heather O'Rourke", u'Cr...
978	7.4	Wall Street	R	Crime	126	[u'Charlie Sheen', u'Michael Douglas', u'Tamar...

979 rows × 6 columns

In [103]:
```
df["title_length"]=df["title"].apply(len)
df
```

Out[103]:

	star_rating	title	content_rating	genre	duration	actors_list	title_length
0	9.3	The Shawshank Redemption	R	Crime	142	[u'Tim Robbins', u'Morgan Freeman', u'Bob Gunt...	24
1	9.2	The Godfather	R	Crime	175	[u'Marlon Brando', u'Al Pacino', u'James Caan']	13
2	9.1	The Godfather: Part II	R	Crime	200	[u'Al Pacino', u'Robert De Niro', u'Robert Duv...	22
3	9.0	The Dark Knight	PG-13	Action	152	[u'Christian Bale', u'Heath Ledger', u'Aaron E...	15
4	8.9	Pulp Fiction	R	Crime	154	[u'John Travolta', u'Uma Thurman', u'Samuel L....	12
...	...	...	...	...	...	...	...
974	7.4	Tootsie	PG	Comedy	116	[u'Dustin Hoffman', u'Jessica Lange', u'Teri G...	7
975	7.4	Back to the Future Part III	PG	Adventure	118	[u'Michael J. Fox', u'Christopher Lloyd', u'Ma...	27
976	7.4	Master and Commander: The Far Side of the World	PG-13	Action	138	[u'Russell Crowe', u'Paul Bettany', u'Billy Bo...	47
977	7.4	Poltergeist	PG	Horror	114	[u'JoBeth Williams', u"Heather O'Rourke", u'Cr...	11
978	7.4	Wall Street	R	Crime	126	[u'Charlie Sheen', u'Michael Douglas', u'Tamar...	11

979 rows × 7 columns

In [104]:
```
df[["title","title_length"]]
```

Out[104]:

	title	title_length
0	The Shawshank Redemption	24
1	The Godfather	13
2	The Godfather: Part II	22
3	The Dark Knight	15
4	Pulp Fiction	12
...	...	...
974	Tootsie	7
975	Back to the Future Part III	27
976	Master and Commander: The Far Side of the World	47
977	Poltergeist	11
978	Wall Street	11

979 rows × 2 columns

23.Write a Pandas program to swap the cases of a specified character column in a given DataFrame

In [105]:
```
df=pd.read_csv("movies_data.csv")
df
```

Out[105]:

	star_rating	title	content_rating	genre	duration	actors_list
0	9.3	The Shawshank Redemption	R	Crime	142	[u'Tim Robbins', u'Morgan Freeman', u'Bob Gunt...
1	9.2	The Godfather	R	Crime	175	[u'Marlon Brando', u'Al Pacino', u'James Caan']
2	9.1	The Godfather: Part II	R	Crime	200	[u'Al Pacino', u'Robert De Niro', u'Robert Duv...
3	9.0	The Dark Knight	PG-13	Action	152	[u'Christian Bale', u'Heath Ledger', u'Aaron E...
4	8.9	Pulp Fiction	R	Crime	154	[u'John Travolta', u'Uma Thurman', u'Samuel L....
...	...	...	...	...	...	...
974	7.4	Tootsie	PG	Comedy	116	[u'Dustin Hoffman', u'Jessica Lange', u'Teri G...
975	7.4	Back to the Future Part III	PG	Adventure	118	[u'Michael J. Fox', u'Christopher Lloyd', u'Ma...
976	7.4	Master and Commander: The Far Side of the World	PG-13	Action	138	[u'Russell Crowe', u'Paul Bettany', u'Billy Bo...
977	7.4	Poltergeist	PG	Horror	114	[u'JoBeth Williams', u"Heather O'Rourke", u'Cr...
978	7.4	Wall Street	R	Crime	126	[u'Charlie Sheen', u'Michael Douglas', u'Tamar...

979 rows × 6 columns

In [106]:
```
df["title"]=df["title"].apply(lambda x:x.swapcase())
df
```

Out[106]:

	star_rating	title	content_rating	genre	duration	actors_list
0	9.3	tHE sHAWSHANK rEDEMPTION	R	Crime	142	[u'Tim Robbins', u'Morgan Freeman', u'Bob Gunt...
1	9.2	tHE gODFATHER	R	Crime	175	[u'Marlon Brando', u'Al Pacino', u'James Caan']
2	9.1	tHE gODFATHER: pART ii	R	Crime	200	[u'Al Pacino', u'Robert De Niro', u'Robert Duv...
3	9.0	tHE dARK kNIGHT	PG-13	Action	152	[u'Christian Bale', u'Heath Ledger', u'Aaron E...
4	8.9	pULP fICTION	R	Crime	154	[u'John Travolta', u'Uma Thurman', u'Samuel L....
...	...	...	...	...	...	...
974	7.4	tOOTSIE	PG	Comedy	116	[u'Dustin Hoffman', u'Jessica Lange', u'Teri G...
975	7.4	bACK TO THE fUTURE pART iii	PG	Adventure	118	[u'Michael J. Fox', u'Christopher Lloyd', u'Ma...
976	7.4	mASTER AND cOMMANDER: tHE fAR sIDE OF THE wORLD	PG-13	Action	138	[u'Russell Crowe', u'Paul Bettany', u'Billy Bo...
977	7.4	pOLTERGEIST	PG	Horror	114	[u'JoBeth Williams', u"Heather O'Rourke", u'Cr...
978	7.4	wALL sTREET	R	Crime	126	[u'Charlie Sheen', u'Michael Douglas', u'Tamar...

979 rows × 6 columns

24.Write a Pandas program to convert a specified character column in upper/lower cases in a given DataFrame.

In [107]:
```
df=pd.read_csv("movies_data.csv")
df
```

Out[107]:

	star_rating	title	content_rating	genre	duration	actors_list
0	9.3	The Shawshank Redemption	R	Crime	142	[u'Tim Robbins', u'Morgan Freeman', u'Bob Gunt...
1	9.2	The Godfather	R	Crime	175	[u'Marlon Brando', u'Al Pacino', u'James Caan']
2	9.1	The Godfather: Part II	R	Crime	200	[u'Al Pacino', u'Robert De Niro', u'Robert Duv...
3	9.0	The Dark Knight	PG-13	Action	152	[u'Christian Bale', u'Heath Ledger', u'Aaron E...
4	8.9	Pulp Fiction	R	Crime	154	[u'John Travolta', u'Uma Thurman', u'Samuel L....
...	...	...	...	...	...	...
974	7.4	Tootsie	PG	Comedy	116	[u'Dustin Hoffman', u'Jessica Lange', u'Teri G...
975	7.4	Back to the Future Part III	PG	Adventure	118	[u'Michael J. Fox', u'Christopher Lloyd', u'Ma...
976	7.4	Master and Commander: The Far Side of the World	PG-13	Action	138	[u'Russell Crowe', u'Paul Bettany', u'Billy Bo...
977	7.4	Poltergeist	PG	Horror	114	[u'JoBeth Williams', u"Heather O'Rourke", u'Cr...
978	7.4	Wall Street	R	Crime	126	[u'Charlie Sheen', u'Michael Douglas', u'Tamar...

979 rows × 6 columns

In [108]:
```
df["title"]=df["title"].apply(lambda x:x.lower())
df
```

Out[108]:

	star_rating	title	content_rating	genre	duration	actors_list
0	9.3	the shawshank redemption	R	Crime	142	[u'Tim Robbins', u'Morgan Freeman', u'Bob Gunt...
1	9.2	the godfather	R	Crime	175	[u'Marlon Brando', u'Al Pacino', u'James Caan']
2	9.1	the godfather: part ii	R	Crime	200	[u'Al Pacino', u'Robert De Niro', u'Robert Duv...
3	9.0	the dark knight	PG-13	Action	152	[u'Christian Bale', u'Heath Ledger', u'Aaron E...
4	8.9	pulp fiction	R	Crime	154	[u'John Travolta', u'Uma Thurman', u'Samuel L....
...	...	...	...	...	...	...
974	7.4	tootsie	PG	Comedy	116	[u'Dustin Hoffman', u'Jessica Lange', u'Teri G...
975	7.4	back to the future part iii	PG	Adventure	118	[u'Michael J. Fox', u'Christopher Lloyd', u'Ma...
976	7.4	master and commander: the far side of the world	PG-13	Action	138	[u'Russell Crowe', u'Paul Bettany', u'Billy Bo...
977	7.4	poltergeist	PG	Horror	114	[u'JoBeth Williams', u"Heather O'Rourke", u'Cr...
978	7.4	wall street	R	Crime	126	[u'Charlie Sheen', u'Michael Douglas', u'Tamar...

979 rows × 6 columns

In [109]:
```
df["genre"]=df["genre"].apply(lambda x:x.upper())
df
```

Out[109]:

	star_rating	title	content_rating	genre	duration	actors_list
0	9.3	the shawshank redemption	R	CRIME	142	[u'Tim Robbins', u'Morgan Freeman', u'Bob Gunt...
1	9.2	the godfather	R	CRIME	175	[u'Marlon Brando', u'Al Pacino', u'James Caan']
2	9.1	the godfather: part ii	R	CRIME	200	[u'Al Pacino', u'Robert De Niro', u'Robert Duv...
3	9.0	the dark knight	PG-13	ACTION	152	[u'Christian Bale', u'Heath Ledger', u'Aaron E...
4	8.9	pulp fiction	R	CRIME	154	[u'John Travolta', u'Uma Thurman', u'Samuel L....
...	...	...	...	...	...	...
974	7.4	tootsie	PG	COMEDY	116	[u'Dustin Hoffman', u'Jessica Lange', u'Teri G...
975	7.4	back to the future part iii	PG	ADVENTURE	118	[u'Michael J. Fox', u'Christopher Lloyd', u'Ma...
976	7.4	master and commander: the far side of the world	PG-13	ACTION	138	[u'Russell Crowe', u'Paul Bettany', u'Billy Bo...
977	7.4	poltergeist	PG	HORROR	114	[u'JoBeth Williams', u"Heather O'Rourke", u'Cr...
978	7.4	wall street	R	CRIME	126	[u'Charlie Sheen', u'Michael Douglas', u'Tamar...

979 rows × 6 columns

25.Write a Pandas program to remove whitespaces, left-sided whitespaces,and right-sided whitespaces of the string values of a given pandas series

In [114]:
```
s1=pd.Series([" Shubhangi "," Mansi "," Rutika "," Samrudhi "])
print(list(s1))
import re
def whitespace(string):
    return re.sub("\s+","",string)
s2=s1.apply(whitespace)
print("after removing whitespaces:",list(s2))
```

```
[' Shubhangi ', ' Mansi ', ' Rutika ', ' Samrudhi ']
after removing whitespaces: ['Shubhangi', 'Mansi', 'Rutika', 'Samrudhi']
```

In [117]:
```
s1=pd.Series([" Shubhangi "," Mansi "," Rutika "," Samrudhi "])
print(list(s1))
print("after removing left sided whitespaces:")
print(list(s1.apply(lambda x:x.lstrip())))
```

```
[' Shubhangi ', ' Mansi ', ' Rutika ', ' Samrudhi ']
after removing left sided whitespaces:
['Shubhangi ', 'Mansi ', 'Rutika ', 'Samrudhi ']
```

In [119]:
```
s1=pd.Series([" Shubhangi "," Mansi "," Rutika "," Samrudhi "])
print(list(s1))
print("after removing right sided whitespaces:")
print(list(s1.apply(lambda x:x.rstrip())))
```

```
[' Shubhangi ', ' Mansi ', ' Rutika ', ' Samrudhi ']
after removing right sided whitespaces:
[' Shubhangi', ' Mansi', ' Rutika', ' Samrudhi']
```

26. Write a Pandas program to extract years between 1800 to 2200 from the specified column of a given DataFrame.

In [137]:
```
df = pd.DataFrame({'company_code': ['c0001','c0002','c0003', 'c0003', 'c0004'],
                   'year': [1800,2000,2300,1900,2200]})
df
```

Out[137]:

	company_code	year
0	c0001	1800
1	c0002	2000
2	c0003	2300
3	c0003	1900
4	c0004	2200

In [138]:
```
df.loc[(df["year"]>1800) & (df["year"]<2200)]
```

Out[138]:

	company_code	year
1	c0002	2000
3	c0003	1900

27.Write a Pandas program to join the two given dataframes along rows

In [145]:
```
df1=pd.DataFrame({"ASCII VALUE":[i for i in range(65,70)],
                  "char":[chr(i) for i in range(65,70)]})
print(df1)
print()
df2=pd.DataFrame({"ASCII VALUE": [i for i in range(70,75)],
                  "char":[chr(i) for i in range(70,75)]})
print(df2)
```

```
   ASCII VALUE char
0           65    A
1           66    B
2           67    C
3           68    D
4           69    E

   ASCII VALUE char
0           70    F
1           71    G
2           72    H
3           73    I
4           74    J
```

In [147]:
```
df1.append(df2,ignore_index=True)
```

```
C:\Users\Sunil\AppData\Local\Temp\ipykernel_12416\2717680053.py:1: FutureWarnin
g: The frame.append method is deprecated and will be removed from pandas in a f
uture version. Use pandas.concat instead.
  df1.append(df2,ignore_index=True)
```

Out[147]:

	ASCII VALUE	char
0	65	A
1	66	B
2	67	C
3	68	D
4	69	E
5	70	F
6	71	G
7	72	H
8	73	I
9	74	J

In [151]:
```
pd.concat([df1,df2],axis=0,ignore_index=True)
```

Out[151]:

	ASCII VALUE	char
0	65	A
1	66	B
2	67	C
3	68	D
4	69	E
5	70	F
6	71	G
7	72	H
8	73	I
9	74	J

28.Write a Pandas program to join the two given dataframes along with columns

In [152]:
```
df1=pd.DataFrame({"ASCII VALUE":[i for i in range(65,70)],
                  "char":[chr(i) for i in range(65,70)]})
print(df1)
print()
df2=pd.DataFrame({"ASCII VALUE": [i for i in range(70,75)],
                  "char":[chr(i) for i in range(70,75)]})
print(df2)
```

```
   ASCII VALUE char
0           65    A
1           66    B
2           67    C
3           68    D
4           69    E

   ASCII VALUE char
0           70    F
1           71    G
2           72    H
3           73    I
4           74    J
```

In [153]:
```
pd.concat([df1,df2],axis=1)
```

Out[153]:

	ASCII VALUE	char	ASCII VALUE	char
0	65	A	70	F
1	66	B	71	G
2	67	C	72	H
3	68	D	73	I
4	69	E	74	J

In [154]:
```
pd.concat([df1,df2],axis=1,ignore_index=True)
```

Out[154]:

	0	1	2	3
0	65	A	70	F
1	66	B	71	G
2	67	C	72	H
3	68	D	73	I
4	69	E	74	J

29.Write a Pandas program to join the two given dataframes along rows and merge with another dataframe along with the common column id.

In [ ]:

30.Write a Pandas program to join the two dataframes using the common column of both dataframes

In [163]:
```
df1=pd.DataFrame({"class":["c1","c2","c3","c4","c5"],
                  "Name":["sanket","vaibhav","shubham","mayur","jaydeep"]})
display(df1)
print()
df2=pd.DataFrame({"class":["c1","c2","c3","c4","c5"],
                  "Name":["shubhangi","vaibhav","shubham","rutuja","Pooja"]})
display(df2)
pd.merge(df1,df2)
```

	class	Name
0	c1	sanket
1	c2	vaibhav
2	c3	shubham
3	c4	mayur
4	c5	jaydeep

	class	Name
0	c1	shubhangi
1	c2	vaibhav
2	c3	shubham
3	c4	rutuja
4	c5	Pooja

Out[163]:

	class	Name
0	c2	vaibhav
1	c3	shubham

31.Write a Pandas program to join (left join) the two dataframes using keys from the left dataframe only

In [165]:
```
df1=pd.DataFrame({"A":["A0","A1","A2"],
                  "B":["B0","B1","B2"]},index=["R0","R1","R2"])
display(df1)
df2=pd.DataFrame({"C":["C0","C1","C2"],
                  "D":["D0","D1","D2"]},index=["R0","R1","R3"])
display(df2)
```

	A	B
R0	A0	B0
R1	A1	B1
R2	A2	B2

	C	D
R0	C0	D0
R1	C1	D1
R3	C2	D2

In [166]:
```
df1.join(df2,how="left")
```

Out[166]:

	A	B	C	D
R0	A0	B0	C0	D0
R1	A1	B1	C1	D1
R2	A2	B2	NaN	NaN

32.Write a Pandas program to join two dataframes using keys from the right dataframe only.

In [167]:
```
df1=pd.DataFrame({"A":["A0","A1","A2"],
                  "B":["B0","B1","B2"]},index=["R0","R1","R2"])
display(df1)
df2=pd.DataFrame({"C":["C0","C1","C2"],
                  "D":["D0","D1","D2"]},index=["R0","R1","R3"])
display(df2)
df1.join(df2,how="right")
```

	A	B
R0	A0	B0
R1	A1	B1
R2	A2	B2

	C	D
R0	C0	D0
R1	C1	D1
R3	C2	D2

Out[167]:

	A	B	C	D
R0	A0	B0	C0	D0
R1	A1	B1	C1	D1
R3	NaN	NaN	C2	D2

33.Write a Pandas program to merge two given datasets using multiple join keys

In [176]:
```
df1=pd.DataFrame({"ASCII VALUE":[i for i in range(65,70)],
                  "Char":[chr(i) for i in range(65,70)]})
display(df1)
df2=pd.DataFrame({"ASCII VALUE":[i for i in range(67,72)],
                  "Char":[chr(i) for i in range(67,72)]})
display(df2)
```

	ASCII VALUE	Char
0	65	A
1	66	B
2	67	C
3	68	D
4	69	E

	ASCII VALUE	Char
0	67	C
1	68	D
2	69	E
3	70	F
4	71	G

In [177]:
```
pd.merge(df1,df2,on="ASCII VALUE",how="inner")
```

Out[177]:

	ASCII VALUE	Char_x	Char_y
0	67	C	C
1	68	D	D
2	69	E	E

In [178]:
```
pd.merge(df1,df2,on="ASCII VALUE",how="outer")
```

Out[178]:

	ASCII VALUE	Char_x	Char_y
0	65	A	NaN
1	66	B	NaN
2	67	C	C
3	68	D	D
4	69	E	E
5	70	NaN	F
6	71	NaN	G

34.Write a Pandas program to merge two given dataframes with

different columns

In [181]:
```
dict1={"Name":["Shreya","Megha","Kartik","Vaibhav"],
       "Age":[23,27,19,24],
       "College":["KTHM","SP","DYP","MIT"],
       "Marks":[90,92,98,94]}
df1=pd.DataFrame(dict1)
display(df1)
dict2={"Name":["Gaytri","Pooja","Mansi","Vedika"],
       "Age":[19,25,20,21],
       "College":["SINHGAD","SYMBOYSIS","RYK","MIT"],
       "Marks":[98,78,70,93]}
df2=pd.DataFrame(dict2)
display(df2)
```

	Name	Age	College	Marks
0	Shreya	23	KTHM	90
1	Megha	27	SP	92
2	Kartik	19	DYP	98
3	Vaibhav	24	MIT	94

	Name	Age	College	Marks
0	Gaytri	19	SINHGAD	98
1	Pooja	25	SYMBOYSIS	78
2	Mansi	20	RYK	70
3	Vedika	21	MIT	93

In [182]:
```
pd.concat([df1,df2],axis=0,ignore_index=True)
```

Out[182]:

	Name	Age	College	Marks
0	Shreya	23	KTHM	90
1	Megha	27	SP	92
2	Kartik	19	DYP	98
3	Vaibhav	24	MIT	94
4	Gaytri	19	SINHGAD	98
5	Pooja	25	SYMBOYSIS	78
6	Mansi	20	RYK	70
7	Vedika	21	MIT	93

In [184]:
```
pd.concat([df1,df2],axis=1,ignore_index=False)
```

Out[184]:

	Name	Age	College	Marks	Name	Age	College	Marks
0	Shreya	23	KTHM	90	Gaytri	19	SINHGAD	98
1	Megha	27	SP	92	Pooja	25	SYMBOYSIS	78
2	Kartik	19	DYP	98	Mansi	20	RYK	70
3	Vaibhav	24	MIT	94	Vedika	21	MIT	93

35.Write a Pandas program to sort movies on runtime in descending order(Use movies dataset)

In [185]:
```
df=pd.read_csv("movies_data.csv")
df
```

Out[185]:

	star_rating	title	content_rating	genre	duration	actors_list
0	9.3	The Shawshank Redemption	R	Crime	142	[u'Tim Robbins', u'Morgan Freeman', u'Bob Gunt...
1	9.2	The Godfather	R	Crime	175	[u'Marlon Brando', u'Al Pacino', u'James Caan']
2	9.1	The Godfather: Part II	R	Crime	200	[u'Al Pacino', u'Robert De Niro', u'Robert Duv...
3	9.0	The Dark Knight	PG-13	Action	152	[u'Christian Bale', u'Heath Ledger', u'Aaron E...
4	8.9	Pulp Fiction	R	Crime	154	[u'John Travolta', u'Uma Thurman', u'Samuel L....
...	...	...	...	...	...	...
974	7.4	Tootsie	PG	Comedy	116	[u'Dustin Hoffman', u'Jessica Lange', u'Teri G...
975	7.4	Back to the Future Part III	PG	Adventure	118	[u'Michael J. Fox', u'Christopher Lloyd', u'Ma...
976	7.4	Master and Commander: The Far Side of the World	PG-13	Action	138	[u'Russell Crowe', u'Paul Bettany', u'Billy Bo...
977	7.4	Poltergeist	PG	Horror	114	[u'JoBeth Williams', u"Heather O'Rourke", u'Cr...
978	7.4	Wall Street	R	Crime	126	[u'Charlie Sheen', u'Michael Douglas', u'Tamar...

979 rows × 6 columns

In [187]:
```
df.sort_values(["genre"],ascending=False)
```

Out[187]:

	star_rating	title	content_rating	genre	duration	actors_list
107	8.3	For a Few Dollars More	APPROVED	Western	132	[u'Clint Eastwood', u'Lee Van Cleef', u'Gian M...
421	7.9	The Outlaw Josey Wales	PG	Western	135	[u'Clint Eastwood', u'Sondra Locke', u'Chief D...
704	7.6	High Plains Drifter	R	Western	105	[u'Clint Eastwood', u'Verna Bloom', u'Marianna...
6	8.9	The Good, the Bad and the Ugly	NOT RATED	Western	161	[u'Clint Eastwood', u'Eli Wallach', u'Lee Van ...
26	8.6	Once Upon a Time in the West	PG-13	Western	175	[u'Henry Fonda', u'Charles Bronson', u'Claudia...
...	...	...	...	...	...	...
118	8.3	Indiana Jones and the Last Crusade	PG-13	Action	127	[u'Harrison Ford', u'Sean Connery', u'Alison D...
586	7.7	Star Trek II: The Wrath of Khan	PG	Action	113	[u'William Shatner', u'Leonard Nimoy', u'DeFor...
581	7.8	Man on Fire	R	Action	146	[u'Denzel Washington', u'Christopher Walken', ...
113	8.3	Batman Begins	PG-13	Action	140	[u'Christian Bale', u'Michael Caine', u'Ken Wa...
420	7.9	The Man Who Would Be King	PG	Action	129	[u'Sean Connery', u'Michael Caine', u'Christop...

979 rows × 6 columns

In [189]:
```
df.sort_values(["duration"],ascending=False)
```

Out[189]:

	star_rating	title	content_rating	genre	duration	actors_list
476	7.8	Hamlet	PG-13	Drama	242	[u'Kenneth Branagh', u'Julie Christie', u'Dere...
157	8.2	Gone with the Wind	G	Drama	238	[u'Clark Gable', u'Vivien Leigh', u'Thomas Mit...
78	8.4	Once Upon a Time in America	R	Crime	229	[u'Robert De Niro', u'James Woods', u'Elizabet...
142	8.3	Lagaan: Once Upon a Time in India	PG	Adventure	224	[u'Aamir Khan', u'Gracy Singh', u'Rachel Shell...
445	7.9	The Ten Commandments	APPROVED	Adventure	220	[u'Charlton Heston', u'Yul Brynner', u'Anne Ba...
...	...	...	...	...	...	...
293	8.1	Duck Soup	PASSED	Comedy	68	[u'Groucho Marx', u'Harpo Marx', u'Chico Marx']
88	8.4	The Kid	NOT RATED	Comedy	68	[u'Charles Chaplin', u'Edna Purviance', u'Jack...
258	8.1	The Cabinet of Dr. Caligari	UNRATED	Crime	67	[u'Werner Krauss', u'Conrad Veidt', u'Friedric...
338	8.0	Battleship Potemkin	UNRATED	History	66	[u'Aleksandr Antonov', u'Vladimir Barsky', u'G...
389	8.0	Freaks	UNRATED	Drama	64	[u'Wallace Ford', u'Leila Hyams', u'Olga Bacla...

979 rows × 6 columns

36.Write a Pandas program to get the longest runtime and shortest runtime

In [190]:
```
df=pd.read_csv("movies_data.csv")
df
```

Out[190]:

	star_rating	title	content_rating	genre	duration	actors_list
0	9.3	The Shawshank Redemption	R	Crime	142	[u'Tim Robbins', u'Morgan Freeman', u'Bob Gunt...
1	9.2	The Godfather	R	Crime	175	[u'Marlon Brando', u'Al Pacino', u'James Caan']
2	9.1	The Godfather: Part II	R	Crime	200	[u'Al Pacino', u'Robert De Niro', u'Robert Duv...
3	9.0	The Dark Knight	PG-13	Action	152	[u'Christian Bale', u'Heath Ledger', u'Aaron E...
4	8.9	Pulp Fiction	R	Crime	154	[u'John Travolta', u'Uma Thurman', u'Samuel L....
...	...	...	...	...	...	...
974	7.4	Tootsie	PG	Comedy	116	[u'Dustin Hoffman', u'Jessica Lange', u'Teri G...
975	7.4	Back to the Future Part III	PG	Adventure	118	[u'Michael J. Fox', u'Christopher Lloyd', u'Ma...
976	7.4	Master and Commander: The Far Side of the World	PG-13	Action	138	[u'Russell Crowe', u'Paul Bettany', u'Billy Bo...
977	7.4	Poltergeist	PG	Horror	114	[u'JoBeth Williams', u"Heather O'Rourke", u'Cr...
978	7.4	Wall Street	R	Crime	126	[u'Charlie Sheen', u'Michael Douglas', u'Tamar...

979 rows × 6 columns

In [194]:
```
df.loc[(df["duration"]==df["duration"].max())]
```

Out[194]:

	star_rating	title	content_rating	genre	duration	actors_list
476	7.8	Hamlet	PG-13	Drama	242	[u'Kenneth Branagh', u'Julie Christie', u'Dere...

In [195]:
```
df.loc[(df["duration"]==df.duration.max())]
```

Out[195]:

	star_rating	title	content_rating	genre	duration	actors_list
476	7.8	Hamlet	PG-13	Drama	242	[u'Kenneth Branagh', u'Julie Christie', u'Dere...

In [196]:
```
df.loc[(df["duration"]==df["duration"].min())]
```

Out[196]:

	star_rating	title	content_rating	genre	duration	actors_list
389	8.0	Freaks	UNRATED	Drama	64	[u'Wallace Ford', u'Leila Hyams', u'Olga Bacla...

In [197]:
```
print("longest runtime for movie in movies dataset in minutes:")
print(df.duration.max())
print()
print("shortest runtime for movie in movies dataset in minutes:")
print(df.duration.min())
```

```
longest runtime for movie in movies dataset in minutes:
242

shortest runtime for movie in movies dataset in minutes:
64
```

37.Write a Pandas program to get Action and crime movies

In [198]:
```
df=pd.read_csv("movies_data.csv")
df
```

Out[198]:

	star_rating	title	content_rating	genre	duration	actors_list
0	9.3	The Shawshank Redemption	R	Crime	142	[u'Tim Robbins', u'Morgan Freeman', u'Bob Gunt...
1	9.2	The Godfather	R	Crime	175	[u'Marlon Brando', u'Al Pacino', u'James Caan']
2	9.1	The Godfather: Part II	R	Crime	200	[u'Al Pacino', u'Robert De Niro', u'Robert Duv...
3	9.0	The Dark Knight	PG-13	Action	152	[u'Christian Bale', u'Heath Ledger', u'Aaron E...
4	8.9	Pulp Fiction	R	Crime	154	[u'John Travolta', u'Uma Thurman', u'Samuel L....
...	...	...	...	...	...	...
974	7.4	Tootsie	PG	Comedy	116	[u'Dustin Hoffman', u'Jessica Lange', u'Teri G...
975	7.4	Back to the Future Part III	PG	Adventure	118	[u'Michael J. Fox', u'Christopher Lloyd', u'Ma...
976	7.4	Master and Commander: The Far Side of the World	PG-13	Action	138	[u'Russell Crowe', u'Paul Bettany', u'Billy Bo...
977	7.4	Poltergeist	PG	Horror	114	[u'JoBeth Williams', u"Heather O'Rourke", u'Cr...
978	7.4	Wall Street	R	Crime	126	[u'Charlie Sheen', u'Michael Douglas', u'Tamar...

979 rows × 6 columns

In [201]:
```
df.loc[(df["genre"]=="Action")|(df["genre"]=="Crime")]
```

Out[201]:

	star_rating	title	content_rating	genre	duration	actors_list
0	9.3	The Shawshank Redemption	R	Crime	142	[u'Tim Robbins', u'Morgan Freeman', u'Bob Gunt...
1	9.2	The Godfather	R	Crime	175	[u'Marlon Brando', u'Al Pacino', u'James Caan']
2	9.1	The Godfather: Part II	R	Crime	200	[u'Al Pacino', u'Robert De Niro', u'Robert Duv...
3	9.0	The Dark Knight	PG-13	Action	152	[u'Christian Bale', u'Heath Ledger', u'Aaron E...
4	8.9	Pulp Fiction	R	Crime	154	[u'John Travolta', u'Uma Thurman', u'Samuel L....
...	...	...	...	...	...	...
963	7.4	La Femme Nikita	R	Action	118	[u'Anne Parillaud', u'Marc Duret', u'Patrick F...
967	7.4	The Rock	R	Action	136	[u'Sean Connery', u'Nicolas Cage', u'Ed Harris']
969	7.4	Law Abiding Citizen	R	Crime	109	[u'Gerard Butler', u'Jamie Foxx', u'Leslie Bibb']
976	7.4	Master and Commander: The Far Side of the World	PG-13	Action	138	[u'Russell Crowe', u'Paul Bettany', u'Billy Bo...
978	7.4	Wall Street	R	Crime	126	[u'Charlie Sheen', u'Michael Douglas', u'Tamar...

260 rows × 6 columns

In [202]:
```
df[(df["genre"]=="Action")|(df["genre"]=="Crime")]
```

Out[202]:

	star_rating	title	content_rating	genre	duration	actors_list
0	9.3	The Shawshank Redemption	R	Crime	142	[u'Tim Robbins', u'Morgan Freeman', u'Bob Gunt...
1	9.2	The Godfather	R	Crime	175	[u'Marlon Brando', u'Al Pacino', u'James Caan']
2	9.1	The Godfather: Part II	R	Crime	200	[u'Al Pacino', u'Robert De Niro', u'Robert Duv...
3	9.0	The Dark Knight	PG-13	Action	152	[u'Christian Bale', u'Heath Ledger', u'Aaron E...
4	8.9	Pulp Fiction	R	Crime	154	[u'John Travolta', u'Uma Thurman', u'Samuel L....
...	...	...	...	...	...	...
963	7.4	La Femme Nikita	R	Action	118	[u'Anne Parillaud', u'Marc Duret', u'Patrick F...
967	7.4	The Rock	R	Action	136	[u'Sean Connery', u'Nicolas Cage', u'Ed Harris']
969	7.4	Law Abiding Citizen	R	Crime	109	[u'Gerard Butler', u'Jamie Foxx', u'Leslie Bibb']
976	7.4	Master and Commander: The Far Side of the World	PG-13	Action	138	[u'Russell Crowe', u'Paul Bettany', u'Billy Bo...
978	7.4	Wall Street	R	Crime	126	[u'Charlie Sheen', u'Michael Douglas', u'Tamar...

260 rows × 6 columns

38.Write a Pandas program to count the city-wise number of people from a given data

set (city, name of the person) Sample data: City Number of people 0 California 4 1 Georgia 2 2 Los Angeles 4

In [203]:
```
df=pd.DataFrame({"city":["California","Georgia","Los Angeles","California","Georg
                 "name":["Bulter","Cage","Marc","Jamie","Billy","Michael"]})
df
```

Out[203]:

	city	name
0	California	Bulter
1	Georgia	Cage
2	Los Angeles	Marc
3	California	Jamie
4	Georgia	Billy
5	California	Michael

In [210]:
```
df1=df.groupby(["city"]).size().reset_index(name="Number of people")
df1
```

Out[210]:

	city	Number of people
0	California	3
1	Georgia	2
2	Los Angeles	1

39.Write a Pandas program to replace all the NaN values with Zero's in a column of a DataFrame

In [211]:
```
df=pd.read_csv("Emp_Records.csv")
df
```

Out[211]:

	Emp ID	First Name	Age in Yrs	Weight in Kgs	Age in Company	Salary	City
0	677509	Lois	36.36	60.0	13.68	168251.0	Denver
1	940761	Brenda	47.02	60.0	9.01	51063.0	Stonewall
2	428945	Joe	54.15	68.0	NaN	50155.0	Michigantown
3	408351	Diane	NaN	51.0	NaN	NaN	Hydetown
4	193819	Benjamin	40.31	NaN	4.01	NaN	Fremont
...	...	...	...	...	...	...	...
95	639892	Jose	22.82	89.0	1.05	129774.0	Biloxi
96	704709	Harold	32.61	77.0	5.93	156194.0	Carol Stream
97	461593	Nicole	52.66	60.0	28.53	95673.0	Detroit
98	392491	Theresa	29.60	57.0	6.99	51015.0	Mc Grath
99	495141	Tammy	38.38	55.0	2.26	93650.0	Alma

100 rows × 7 columns

In [217]:
```
df.fillna(0)
```

Out[217]:

	Emp ID	First Name	Age in Yrs	Weight in Kgs	Age in Company	Salary	City
0	677509	Lois	36.36	60.0	13.68	168251.0	Denver
1	940761	Brenda	47.02	60.0	9.01	51063.0	Stonewall
2	428945	Joe	54.15	68.0	0.00	50155.0	Michigantown
3	408351	Diane	0.00	51.0	0.00	0.0	Hydetown
4	193819	Benjamin	40.31	0.0	4.01	0.0	Fremont
...	...	...	...	...	...	...	...
95	639892	Jose	22.82	89.0	1.05	129774.0	Biloxi
96	704709	Harold	32.61	77.0	5.93	156194.0	Carol Stream
97	461593	Nicole	52.66	60.0	28.53	95673.0	Detroit
98	392491	Theresa	29.60	57.0	6.99	51015.0	Mc Grath
99	495141	Tammy	38.38	55.0	2.26	93650.0	Alma

100 rows × 7 columns

In [219]:
```
df=df.replace({np.nan:0})
df
```

Out[219]:

	Emp ID	First Name	Age in Yrs	Weight in Kgs	Age in Company	Salary	City
0	677509	Lois	36.36	60.0	13.68	168251.0	Denver
1	940761	Brenda	47.02	60.0	9.01	51063.0	Stonewall
2	428945	Joe	54.15	68.0	0.00	50155.0	Michigantown
3	408351	Diane	0.00	51.0	0.00	0.0	Hydetown
4	193819	Benjamin	40.31	0.0	4.01	0.0	Fremont
...	...	...	...	...	...	...	...
95	639892	Jose	22.82	89.0	1.05	129774.0	Biloxi
96	704709	Harold	32.61	77.0	5.93	156194.0	Carol Stream
97	461593	Nicole	52.66	60.0	28.53	95673.0	Detroit
98	392491	Theresa	29.60	57.0	6.99	51015.0	Mc Grath
99	495141	Tammy	38.38	55.0	2.26	93650.0	Alma

100 rows × 7 columns

40.Write a Pandas program to drop a list of rows from a specified DataFrame

Sample data: Original DataFrame col1 col2 col3 0 1 4 7 1 4 5 8 2 3 6 9 3 4 7 0 4 5 8 1

In [233]:
```
d1={"col1":[1,4,3,4,5],"col2":[4,5,6,7,8],"col3":[7,8,9,0,1]}
df=pd.DataFrame(d1)
df
```

Out[233]:

	col1	col2	col3
0	1	4	7
1	4	5	8
2	3	6	9
3	4	7	0
4	5	8	1

In [234]:
```
df.drop(0,axis=0)
```

Out[234]:

	col1	col2	col3
1	4	5	8
2	3	6	9
3	4	7	0
4	5	8	1

In [235]:
```
df.drop(2,axis=0)
```

Out[235]:

	col1	col2	col3
0	1	4	7
1	4	5	8
3	4	7	0
4	5	8	1

In [236]:
```
df.drop([3,4])
```

Out[236]:

	col1	col2	col3
0	1	4	7
1	4	5	8
2	3	6	9

In [237]:
```
df.drop([1,2],axis=0,inplace=True)
df
```

Out[237]:

	col1	col2	col3
0	1	4	7
3	4	7	0
4	5	8	1