Shubhangi_Dhikale_35

1.Define the Pandas/Python pandas? ¶

In [ ]:
```
1)Pandas is an open source python package that is most widely used for
data science/data analysis and machine learning tasks.
2)Pandas is a python library used for working with data sets.
3)It has functions for analyzing,cleaning,exploring,and manipulating data.
4)Pndas allows us to analyze big data and make conclusions based on statistical t
5)Pndas can clean messy data sets,and make them readable and relevant.
```

2.What are the different types of Data Structures in Pandas?

In [ ]:
```
There are two types of data structures:
1) Series: one-dimensional labeled arrays pd.It is immutable.
2) DataFrame=: Two dimensional data structure with columns,much like a table.
```

3.Explain Series and DataFrame In Pandas

In [ ]:
```
1) Series: one-dimensional labeled arrays pd that is capable of storing various d
The row labela of series are called the index.
we can easily convert the list,tuple and dictionary into series using "seires" me
It has one parameter.

2) DataFrame=: Two dimensional data structure with columns,much like a table.
It is defined as a standard way to store data and has two different indexes,i.e,
It consists of the following properties:
It can be seen as a dictionary of series structure where both the rows and columr
It is denoted as "columns" in case of columns and "index" in case of rows.
The columns can be heterogeous types like int,bool,and so on.
```

4.How Can You Create An Empty DataFrame and series in Pandas?

In [2]:
```
import pandas as pd
#empty dataframe
df=pd.DataFrame()
df
```

Out[2]:

In [4]:
```
#empty data series
series=pd.Series()
series
```

```
C:\Users\Sunil\AppData\Local\Temp\ipykernel_4008\3931766751.py:2: FutureWarnin
g: The default dtype for empty Series will be 'object' instead of 'float64' in
a future version. Specify a dtype explicitly to silence this warning.
  series=pd.Series()
```

Out[4]:
```
Series([], dtype: float64)
```

In [6]:
```
series=pd.Series(dtype="O")
series
```

Out[6]:
```
Series([], dtype: object)
```

5.How to check an empty DataFrame?

In [ ]:
```
#df.empty
# df.size
# df.info()
# df.index
# df.shape
```

In [7]:
```
df=pd.DataFrame()
df
```

Out[7]:
```
_
```

In [8]:
```
df.empty
```

Out[8]:
```
True
```

In [9]:
```
df.size
```

Out[9]:
```
0
```

In [10]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 0 entries
Empty DataFrame
```

In [11]:
```
df.index
```

Out[11]:
```
Index([], dtype='object')
```

In [12]:
```
df.shape
```

Out[12]:
```
(0, 0)
```

6.What Are The Most Important Features Of The Pandas Library?

In [ ]:
```
Important Features of the pandas:
1) It has a fast and efficient Dataframe object with the default and customized i
2) Used for reshaping and pivoting of the data sets.
3) Group by data for aggregations and transformations.
4) It is used for data alignment and integration of the missing data.
5) Provide the functionality of the time series.
6) process of variety of data sets in different formats like matrix data,tabular
```

7.How Will You Explain Reindexing In Pandas?

In [19]:
```
# Reindexing along rows:
# one can reindex a single row or multiple rows by using reindex()method.
# Default values in the new index that are not present in the dataframe are assig

df=pd.DataFrame({"col1":[1,2,3,4,5],
                 "col2":[10,20,30,40,50]},index=["A","B","C","D","E"])
df
```

Out[19]:

	col1	col2
A	1	10
B	2	20
C	3	30
D	4	40
E	5	50

In [23]:
```
df.reindex(["B","C","D","A","E"])
```

Out[23]:

	col1	col2
B	2	20
C	3	30
D	4	40
A	1	10
E	5	50

In [24]:
```
# if index is not present,NaN values are populated in the dataframe
df.reindex(["B","C","D","A","E","R"])
```

Out[24]:

	col1	col2
B	2.0	20.0
C	3.0	30.0
D	4.0	40.0
A	1.0	10.0
E	5.0	50.0
R	NaN	NaN

In [33]:
```
# Reindexing along columns using axis keywords:
# one can reindex a single column or multiple columns by using reindex()method ar
# Default values in the new index that are not present in the dataframe are assig
```

In [50]:
```
import numpy as np
data=np.array([1,2,3,4,5,6])
df=pd.DataFrame(data,columns=["x"])
df
```

Out[50]:

	x
0	1
1	2
2	3
3	4
4	5
5	6

8.What are the different ways of creating DataFrame in pandas? Explain with examples.

```
There are multiple ways how we can create a dataframe:
    1.List
    2.Dict
    3.Array
    4.Mangodb database
    5.sql database
    6.copy from clipboard
    7.excel
    8.csv
    9.json
```

In [54]:
```
#1. creating dataframe by using list
l1=["shubhangi","Rutuja","Mansi","Samrudhi"]
df=pd.DataFrame(l1,columns=["Names"])
df
```

Out[54]:

	Names
0	shubhangi
1	Rutuja
2	Mansi
3	Samrudhi

In [55]:
```
#2. creating dataframe by using dict
dict1={"name":["shubhangi","Rutuja","Mansi","Samrudhi"],
       "marks":[90,95,87,89]}
df=pd.DataFrame(dict1)
df
```

Out[55]:

	name	marks
0	shubhangi	90
1	Rutuja	95
2	Mansi	87
3	Samrudhi	89

In [56]:
```
#3.creatinf dataframe by using array
import numpy as np
arr1=np.array([10,20,30,40,50])
df=pd.DataFrame(arr1,columns=["number"])
df
```

Out[56]:

	number
0	10
1	20
2	30
3	40
4	50

In [58]:
```
#4.creating dataframe by using xlsx
df=pd.read_excel(r"C:\Users\Sunil\Downloads\Book1.xlsx")
df
```

Out[58]:

	Names	marks
0	Shubhangi	89
1	Rutuja	80
2	Mansi	78
3	Samrudhi	95
4	vedika	90

In [59]:
```
#5. creating dataframe by using csv
df=pd.read_csv("Iris.csv")
df
```

Out[59]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa
...	...	...	...	...	...	...
145	146	6.7	3.0	5.2	2.3	Iris-virginica
146	147	6.3	2.5	5.0	1.9	Iris-virginica
147	148	6.5	3.0	5.2	2.0	Iris-virginica
148	149	6.2	3.4	5.4	2.3	Iris-virginica
149	150	5.9	3.0	5.1	1.8	Iris-virginica

150 rows × 6 columns

9.Create a DataFrame using List

In [60]:
```
list1=["python","machine learning","powerBI","R","C++"]
df=pd.DataFrame(list1)
df
```

Out[60]:

	0
0	python
1	machine learning
2	powerBI
3	R
4	C++

10.Create a DataFrame using Numpy Functions.

In [65]:
```
import numpy as np
data=np.random.randint(1,31,size=10)
df=pd.DataFrame(data,columns=["col"],index=list("abcdefghij"))
df
```

Out[65]:

	col
a	13
b	15
c	22
d	18
e	13
f	20
g	8
h	19
i	8
j	4

11.How to convert a NumPy array to a DataFrame of a given shape?

In [68]:
```
arr1=np.array([[5,6,8],[1,2,3],[5,4,3]])
df=pd.DataFrame(arr1,columns=["c1","c2","c3"],index=["R1","R2","R3"])
df
```

Out[68]:

	c1	c2	c3
R1	5	6	8
R2	1	2	3
R3	5	4	3

12.Create a DataFrame using Dictionary with a list and arrays

In [73]:
```
data={"list1":[10,20,30,40,50],
      "arr1":np.array([1,2,3,4,5]),
      "arange":np.arange(21,26),
      "zeros":np.zeros(5,dtype=int)}
df=pd.DataFrame(data)
df
```

Out[73]:

	list1	arr1	arange	zeros
0	10	1	21	0
1	20	2	22	0
2	30	3	23	0
3	40	4	24	0
4	50	5	25	0

13.How To Create A Copy Of The Series and DataFrame in Pandas?

In [74]:
```
list1=[10,20,30,40]
series=pd.Series(list1)
series
```

Out[74]:
```
0    10
1    20
2    30
3    40
dtype: int64
```

In [76]:
```
#creating copy of series
series1=series.copy()
series1
```

Out[76]:
```
0    10
1    20
2    30
3    40
dtype: int64
```

In [77]:
```
list2=pd.DataFrame([2,4,6,8,10],columns=["evenNo"])
list2
```

Out[77]:

	evenNo
0	2
1	4
2	6
3	8
4	10

In [78]:
```
#Creating copy dataframe
dataframe=list2.copy()
dataframe
```

Out[78]:

	evenNo
0	2
1	4
2	6
3	8
4	10

14.How Will You Add An Index, Row, Or Column To A DataFrame In Pandas?

In [79]:
```
df=pd.DataFrame({"Name":["Shubhangi","shivani","Rutuja","Sanket","Vaibhav"],
                 "Surname":["Dhikale","sharma","gupta","jadhav","gawali"]})
df
```

Out[79]:

	Name	Surname
0	Shubhangi	Dhikale
1	shivani	sharma
2	Rutuja	gupta
3	Sanket	jadhav
4	Vaibhav	gawali

In [81]:
```
# add columns in data
df["marks"]=[20,30,40,60,50]
df
```

Out[81]:

	Name	Surname	marks
0	Shubhangi	Dhikale	20
1	shivani	sharma	30
2	Rutuja	gupta	40
3	Sanket	jadhav	60
4	Vaibhav	gawali	50

In [84]:
```
# add rows in data
df.loc[5]=["Nisha","Deshmukh",80]
df
```

Out[84]:

	Name	Surname	marks
0	Shubhangi	Dhikale	20
1	shivani	sharma	30
2	Rutuja	gupta	40
3	Sanket	jadhav	60
4	Vaibhav	gawali	50
5	Nisha	Deshmukh	80

15.What Method Will You Use To Rename The Index Or Columns Of Pandas DataFrame?

In [96]:
```
df=pd.DataFrame({"ones":np.ones(5),
                 "zeros":np.zeros(5),
                 "list1":[i for i in range(1,6)]})

df.rename(columns={"ones":"X","zeros":"Y"})
```

Out[96]:

	X	Y	list1
0	1.0	0.0	1
1	1.0	0.0	2
2	1.0	0.0	3
3	1.0	0.0	4
4	1.0	0.0	5

In [100]:
```
df.rename(index={0:"a",1:"b",2:"c",3:"d",4:"e"})
```

Out[100]:

	ones	zeros	list1
a	1.0	0.0	1
b	1.0	0.0	2
c	1.0	0.0	3
d	1.0	0.0	4
e	1.0	0.0	5

16.How Can You Iterate Over DataFrame In Pandas?

In [101]:
```
data={"Name":["shubhangi","sanket","shubham","mansi","rutuja"],
      "Age":[24,19,21,24,23],
      "Education":["MSC","Animation","B.A","MSC","MSC"]}
df=pd.DataFrame(data)
df
```

Out[101]:

	Name	Age	Education
0	shubhangi	24	MSC
1	sanket	19	Animation
2	shubham	21	B.A
3	mansi	24	MSC
4	rutuja	23	MSC

In [102]:
```
for i in range(len(df)):
    print(df.loc[i,"Name"])
```

```
shubhangi
sanket
shubham
mansi
rutuja
```

In [103]:
```
for i in range(len(df)):
    print(df.loc[i,"Age"])
```

```
24
19
21
24
23
```

In [104]:
```
for i in range(len(df)):
    print(df.loc[i,"Education"])
```

```
MSC
Animation
B.A
MSC
MSC
```

In [106]:
```
for i in range(len(df)):
    print(df.iloc[i,0])
```

```
shubhangi
sanket
shubham
mansi
rutuja
```

In [108]:
```
for i in range(len(df)):
    print(df.iloc[i,1])
```

```
24
19
21
24
23
```

In [109]:
```
for i in range(len(df)):
    print(df.iloc[i,2])
```

```
MSC
Animation
B.A
MSC
MSC
```

17.How to create an array from DataFrame

In [111]:
```
df={"list":[10,20,30,40],
   "zeros":np.zeros(4,dtype=int),
   "ones":np.ones(4,dtype=int),
   "arange":np.arange(1,5)}
df=pd.DataFrame(df)
df
```

Out[111]:

	list	zeros	ones	arange
0	10	0	1	1
1	20	0	1	2
2	30	0	1	3
3	40	0	1	4

In [112]:
```
arr1=df.to_numpy()
print(arr1)
```

```
[[10  0  1  1]
 [20  0  1  2]
 [30  0  1  3]
 [40  0  1  4]]
```

In [113]:
```
print(type(arr1))
```

```
<class 'numpy.ndarray'>
```

In [114]:
```
arr2=df.values
print(arr2)
```

```
[[10  0  1  1]
 [20  0  1  2]
 [30  0  1  3]
 [40  0  1  4]]
```

18.How to create a list from DataFrame

In [115]:
```
df={"list":[10,20,30,40],
   "zeros":np.zeros(4,dtype=int),
   "ones":np.ones(4,dtype=int),
   "arange":np.arange(1,5)}
df=pd.DataFrame(df)
df
```

Out[115]:

	list	zeros	ones	arange
0	10	0	1	1
1	20	0	1	2
2	30	0	1	3
3	40	0	1	4

In [121]:
```
list1=df.values.tolist()
print(list1)
```

```
[[10, 0, 1, 1], [20, 0, 1, 2], [30, 0, 1, 3], [40, 0, 1, 4]]
```

In [117]:
```
print(type(list1))
```

```
<class 'list'>
```

19.How to Reset the dataframes index?

In [128]:
```
df=pd.read_csv("Emp_Records.csv",index_col="Emp ID")
df
```

Out[128]:

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

100 rows × 6 columns

In [130]:
```
df1=df.reset_index()
df1
```

Out[130]:

	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 [131]:
```
df.reset_index(drop=True)
```

Out[131]:

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

100 rows × 6 columns

In [132]:
```
df.set_index("Age in Yrs")
```

Out[132]:

Age in Yrs	First Name	Weight in Kgs	Age in Company	Salary	City
36.36	Lois	60.0	13.68	168251.0	Denver
47.02	Brenda	60.0	9.01	51063.0	Stonewall
54.15	Joe	68.0	NaN	50155.0	Michigantown
NaN	Diane	51.0	NaN	NaN	Hydetown
40.31	Benjamin	NaN	4.01	NaN	Fremont
...	...	...	...	...	...
22.82	Jose	89.0	1.05	129774.0	Biloxi
32.61	Harold	77.0	5.93	156194.0	Carol Stream
52.66	Nicole	60.0	28.53	95673.0	Detroit
29.60	Theresa	57.0	6.99	51015.0	Mc Grath
38.38	Tammy	55.0	2.26	93650.0	Alma

100 rows × 5 columns

In [133]:
```
df.reset_index()
```

Out[133]:

	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

20.Write a Pandas program to count the number of rows and columns of a DataFrame

In [134]:
```
df={"list":[10,20,30,40],
   "zeros":np.zeros(4,dtype=int),
   "ones":np.ones(4,dtype=int),
   "arange":np.arange(1,5)}
df=pd.DataFrame(df)
df
```

Out[134]:

	list	zeros	ones	arange
0	10	0	1	1
1	20	0	1	2
2	30	0	1	3
3	40	0	1	4

In [136]:
```
print("number of row values:",df.index)
```

```
number of row values: RangeIndex(start=0, stop=4, step=1)
```

In [138]:
```
print("number of column values:",df.columns)
```

```
number of column values: Index(['list', 'zeros', 'ones', 'arange'], dtype='obje
ct')
```

In [139]:
```
print("number of row and column values:",df.axes)
```

```
number of row and column values: [RangeIndex(start=0, stop=4, step=1), Index
(['list', 'zeros', 'ones', 'arange'], dtype='object')]
```

In [140]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4 entries, 0 to 3
Data columns (total 4 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   list    4 non-null      int64
 1   zeros   4 non-null      int32
 2   ones    4 non-null      int32
 3   arange  4 non-null      int32
dtypes: int32(3), int64(1)
memory usage: 208.0 bytes
```

In [141]:
```
df.shape
```

Out[141]: (4, 4)

In [142]:
```
print("number of rows are:",df.shape[0])
print("number of columns are:",df.shape[1])
```

```
number of rows are: 4
number of columns are: 4
```

21.Write a Pandas program to add, subtract, multiple, and divide two Pandas Series

In [153]:
```
import numpy as np
s1=pd.Series(np.array([10,20,30,40]))
s2=pd.Series(np.array([1,2,3,4]))
print(s1)
print(s2)
print()
print("Addition of two series:\n",s1+s2)
print("Subtraction of two series:\n",s1-s2)
print("multiplication of two series:\n",s1*s2)
print("division of two series:\n",s1/s2)
```

```
0    10
1    20
2    30
3    40
dtype: int32
0    1
1    2
2    3
3    4
dtype: int32

Addition of two series:
 0    11
1    22
2    33
3    44
dtype: int32
Subtraction of two series:
 0     9
1    18
2    27
3    36
dtype: int32
multiplication of two series:
 0     10
1     40
2     90
3    160
dtype: int32
division of two series:
 0    10.0
1    10.0
2    10.0
3    10.0
dtype: float64
```

22.Write a Pandas program to import excel data into a Pandas dataframe

In [155]:
```
df=pd.read_excel(r"C:\Users\Sunil\Downloads\Book1.xlsx")
df
```

Out[155]:

	Names	marks
0	Shubhangi	89
1	Rutuja	80
2	Mansi	78
3	Samrudhi	95
4	vedika	90

23.Write a Pandas program to read specific columns from a given excel file

In [156]:
```
df=pd.read_excel(r"C:\Users\Sunil\Downloads\Book1.xlsx")
df
```

Out[156]:

	Names	marks
0	Shubhangi	89
1	Rutuja	80
2	Mansi	78
3	Samrudhi	95
4	vedika	90

In [158]:
```
df[["Names"]]
```

Out[158]:

	Names
0	Shubhangi
1	Rutuja
2	Mansi
3	Samrudhi
4	vedika

24.Write a Pandas program to find the sum, mean, max, min value of dataframe.

In [159]:
```
df=pd.read_csv("titanic.csv")
df
```

Out[159]:

	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 [160]:
```
df["Pclass"].sum()
```

Out[160]:
```
2057
```

In [161]:
```
df["Age"].mean()
```

Out[161]: 29.69911764705882

In [162]:
```
df["Age"].max()
```

Out[162]: 80.0

In [163]:
```
df["Age"].min()
```

Out[163]: 0.42

25.How Can A DataFrame Be Converted To An Excel File and CSV file?

In [165]:
```
data={"Name":["Shubhangi","Mansi","rutuja","nikita"],
      "Age":[20,22,30,21],
      "marks":[90,91,92,93]}
df=pd.DataFrame(data)
df
```

Out[165]:

	Name	Age	marks
0	Shubhangi	20	90
1	Mansi	22	91
2	rutuja	30	92
3	nikita	21	93

In [166]:
```
df.to_csv("Info.csv")
```

In [167]:
```
df=pd.read_csv("Info.csv")
df
```

Out[167]:

	Unnamed: 0	Name	Age	marks
0	0	Shubhangi	20	90
1	1	Mansi	22	91
2	2	rutuja	30	92
3	3	nikita	21	93

In [168]:
```
df.to_excel("GroupA.xlsx")
```

In [169]:
```
df=pd.read_excel("GroupA.xlsx")
df
```

Out[169]:

	Unnamed: 0.1	Unnamed: 0	Name	Age	marks
0	0	0	Shubhangi	20	90
1	1	1	Mansi	22	91
2	2	2	rutuja	30	92
3	3	3	nikita	21	93

26.What is Groupby Function In Pandas? Explain with example?

```
groupby() is a powerful and versatile in python.
It allows you split your data into seperate groups to perform computations for
better analysis.
groupby() and pass the name of the columns you want to group on,which is
"state".
then,you use ["last_name"] to specify the columns on which you want to perform
the actual aggregation.
You can pass a lot more than just a single column name to .groupby() as the
first argument.
```

In [170]:
```
df=pd.read_csv("movies_data.csv")
df
```

Out[170]:

	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 [172]:
```
df1=df.groupby("content_rating").get_group("PG-13")
df1
```

Out[172]:

	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 [173]:
```
df1=df.groupby(["genre","content_rating"]).get_group(("Action","PG-13"))
df1
```

Out[173]:

	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'...
43	8.5	The Dark Knight Rises	PG-13	Action	165	[u'Christian Bale', u'Tom Hardy', u'Anne Hatha...
113	8.3	Batman Begins	PG-13	Action	140	[u'Christian Bale', u'Michael Caine', u'Ken Wa...
118	8.3	Indiana Jones and the Last Crusade	PG-13	Action	127	[u'Harrison Ford', u'Sean Connery', u'Alison D...
177	8.2	The Avengers	PG-13	Action	143	[u'Robert Downey Jr.', u'Chris Evans', u'Scarl...
196	8.1	Guardians of the Galaxy	PG-13	Action	121	[u'Chris Pratt', u'Vin Diesel', u'Bradley Coop...
240	8.1	The Bourne Ultimatum	PG-13	Action	115	[u'Matt Damon', u'\xc9dgar Ram\xedrez', u'Joan...
248	8.1	X-Men: Days of Future Past	PG-13	Action	131	[u'Patrick Stewart', u'Ian McKellen', u'Hugh J...
301	8.0	Furious 7	PG-13	Action	137	[u'Vin Diesel', u'Paul Walker', u'Dwayne Johns...
312	8.0	Star Trek	PG-13	Action	127	[u'Chris Pine', u'Zachary Quinto', u'Simon Pegg']
366	8.0	Serenity	PG-13	Action	119	[u'Nathan Fillion', u'Gina Torres', u'Chiwetel...
380	8.0	Casino Royale	PG-13	Action	144	[u'Daniel Craig', u'Eva Green', u'Judi Dench']
385	8.0	Spartacus	PG-13	Action	197	[u'Kirk Douglas', u'Laurence Olivier', u'Jean ...
391	8.0	Edge of Tomorrow	PG-13	Action	113	[u'Tom Cruise', u'Emily Blunt', u'Bill Paxton']
403	7.9	Ying xiong	PG-13	Action	99	[u'Jet Li', u'Tony Chiu Wai Leung', u'Maggie C...

	star_rating	title	content_rating	genre	duration	actors_list
411	7.9	The Bourne Identity	PG-13	Action	119	[u'Franka Potente', u'Matt Damon', u'Chris Coo...
433	7.9	Avatar	PG-13	Action	162	[u'Sam Worthington', u'Zoe Saldana', u'Sigourn...
434	7.9	Iron Man	PG-13	Action	126	[u'Robert Downey Jr.', u'Gwyneth Paltrow', u'T...
437	7.9	Crouching Tiger, Hidden Dragon	PG-13	Action	120	[u'Yun-Fat Chow', u'Michelle Yeoh', u'Ziyi Zha...
455	7.9	Taken	PG-13	Action	93	[u'Liam Neeson', u'Maggie Grace', u'Famke Jans...
488	7.8	Star Trek Into Darkness	PG-13	Action	132	[u'Chris Pine', u'Zachary Quinto', u'Zoe Salda...
517	7.8	Captain America: The Winter Soldier	PG-13	Action	136	[u'Chris Evans', u'Samuel L. Jackson', u'Scarl...
532	7.8	X-Men: First Class	PG-13	Action	132	[u'James McAvoy', u'Michael Fassbender', u'Jen...
539	7.8	The Fugitive	PG-13	Action	130	[u'Harrison Ford', u'Tommy Lee Jones', u'Sela ...
567	7.8	The Bourne Supremacy	PG-13	Action	108	[u'Matt Damon', u'Franka Potente', u'Joan Allen']
568	7.8	Skyfall	PG-13	Action	143	[u'Daniel Craig', u'Javier Bardem', u'Naomie H...
591	7.7	The Count of Monte Cristo	PG-13	Action	131	[u'Jim Caviezel', u'Guy Pearce', u'Richard Har...
623	7.7	Dawn of the Planet of the Apes	PG-13	Action	130	[u'Gary Oldman', u'Keri Russell', u'Andy Serkis']
653	7.7	Fearless	PG-13	Action	104	[u'Jet Li', u'Li Sun', u'Yong Dong']
661	7.7	Star Wars: Episode III - Revenge of the Sith	PG-13	Action	140	[u'Hayden Christensen', u'Natalie Portman', u'...
684	7.7	The Big Blue	PG-13	Action	168	[u'Jean-Marc Barr', u'Jean Reno', u'Rosanna Ar...
686	7.7	Minority Report	PG-13	Action	145	[u'Tom Cruise', u'Colin Farrell', u'Samantha M...

	star_rating	title	content_rating	genre	duration	actors_list
699	7.6	The Fifth Element	PG-13	Action	126	[u'Bruce Willis', u'Milla Jovovich', u'Gary Ol...
726	7.6	Sherlock Holmes	PG-13	Action	128	[u'Robert Downey Jr.', u'Jude Law', u'Rachel M...
787	7.6	Rise of the Planet of the Apes	PG-13	Action	105	[u'James Franco', u'Andy Serkis', u'Freida Pin...
801	7.6	Batman	PG-13	Action	126	[u'Michael Keaton', u'Jack Nicholson', u'Kim B...
809	7.6	House of Flying Daggers	PG-13	Action	119	[u'Ziyi Zhang', u'Takeshi Kaneshiro', u'Andy L...
812	7.6	Star Trek: First Contact	PG-13	Action	111	[u'Patrick Stewart', u'Jonathan Frakes', u'Bre...
856	7.5	Scott Pilgrim vs. the World	PG-13	Action	112	[u'Michael Cera', u'Mary Elizabeth Winstead', ...
858	7.5	Sherlock Holmes: A Game of Shadows	PG-13	Action	129	[u'Robert Downey Jr.', u'Jude Law', u'Jared Ha...
871	7.5	X2	PG-13	Action	134	[u'Patrick Stewart', u'Hugh Jackman', u'Halle ...
954	7.4	X-Men	PG-13	Action	104	[u'Patrick Stewart', u'Hugh Jackman', u'Ian Mc...
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...

27.What is the use of apply function in pandas?

```
The apply() allows you to apply a function along one of the axis of the
DataFrame,
default 0,which is the index(row)axis.
```

In [174]:
```
data=np.random.randint(1,20,size=(5,6))
df=pd.DataFrame(data,columns=list("abcdef"))
df
```

Out[174]:

	a	b	c	d	e	f
0	18	13	19	8	16	3
1	12	18	19	19	7	1
2	11	3	5	9	11	18
3	7	15	1	1	4	1
4	17	14	13	19	1	12

In [175]:
```
def sqaure(n):
    return n**2
df.apply(sqaure)
```

Out[175]:

	a	b	c	d	e	f
0	324	169	361	64	256	9
1	144	324	361	361	49	1
2	121	9	25	81	121	324
3	49	225	1	1	16	1
4	289	196	169	361	1	144

28. How to use apply() with lambda function?

In [177]:
```
df=pd.DataFrame({"Country":["India","UK","USA"],
                 "Capitals":["Delhi","London","Washington"]})
df
```

Out[177]:

	Country	Capitals
0	India	Delhi
1	UK	London
2	USA	Washington

In [178]:
```
df["Capitals"]=df["Capitals"].apply(lambda x:x.upper())
df
```

Out[178]:

	Country	Capitals
0	India	DELHI
1	UK	LONDON
2	USA	WASHINGTON

29.Explain is the use of info, describe, head, head, and tail functions

In [179]:
```
df=pd.read_csv("Iris.csv")
df
```

Out[179]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa
...	...	...	...	...	...	...
145	146	6.7	3.0	5.2	2.3	Iris-virginica
146	147	6.3	2.5	5.0	1.9	Iris-virginica
147	148	6.5	3.0	5.2	2.0	Iris-virginica
148	149	6.2	3.4	5.4	2.3	Iris-virginica
149	150	5.9	3.0	5.1	1.8	Iris-virginica

150 rows × 6 columns

```
1.df.info():
The info() method prints information about the DataFrame.
The information contains the number of columns,column labels,column data
types,memory usage,
range index,and the number of cells in each column(non=null values).
```

In [180]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Id             150 non-null    int64  
 1   SepalLengthCm  150 non-null    float64
 2   SepalWidthCm   150 non-null    float64
 3   PetalLengthCm  150 non-null    float64
 4   PetalWidthCm   150 non-null    float64
 5   Species        150 non-null    object 
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
```

```
df.describe():
The describe() method returns description of the data in the DataFrame.
If the DataFrame contains numerical data,the description contains these
information for each column.
```

In [181]:
```
df.describe()
```

Out[181]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
count	150.000000	150.000000	150.000000	150.000000	150.000000
mean	75.500000	5.843333	3.054000	3.758667	1.198667
std	43.445368	0.828066	0.433594	1.764420	0.763161
min	1.000000	4.300000	2.000000	1.000000	0.100000
25%	38.250000	5.100000	2.800000	1.600000	0.300000
50%	75.500000	5.800000	3.000000	4.350000	1.300000
75%	112.750000	6.400000	3.300000	5.100000	1.800000
max	150.000000	7.900000	4.400000	6.900000	2.500000

```
df.head():
The head() function is used to get the first n rows.
This function returns the first n row for the object based on position.
It is useful for quickly testing if your object has the right type of data in
it.
If no value passed as argument,it gives first 5 rows in output.
```

In [182]:
```
df.head()
```

Out[182]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa

In [183]:
```
df.head(3)
```

Out[183]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa

```
df.tail():
The head() function is used to get the last n rows.
This function returns the last n row for the object based on position.
It is useful for quickly verifying data,for example,after sorting or appending
rows.
If no value passed as argument,it gives last 5 rows in output.
```

In [184]:
```
df.tail()
```

Out[184]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
145	146	6.7	3.0	5.2	2.3	Iris-virginica
146	147	6.3	2.5	5.0	1.9	Iris-virginica
147	148	6.5	3.0	5.2	2.0	Iris-virginica
148	149	6.2	3.4	5.4	2.3	Iris-virginica
149	150	5.9	3.0	5.1	1.8	Iris-virginica

30. Write a Pandas program to check whether only a title case is present in a given column of a DataFrame.

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 [186]:
```
df["title"]=df["title"].apply(lambda x:x.istitle())
df
```

Out[186]:

	star_rating	title	content_rating	genre	duration	actors_list
0	9.3	True	R	Crime	142	[u'Tim Robbins', u'Morgan Freeman', u'Bob Gunt...
1	9.2	True	R	Crime	175	[u'Marlon Brando', u'Al Pacino', u'James Caan']
2	9.1	False	R	Crime	200	[u'Al Pacino', u'Robert De Niro', u'Robert Duv...
3	9.0	True	PG-13	Action	152	[u'Christian Bale', u'Heath Ledger', u'Aaron E...
4	8.9	True	R	Crime	154	[u'John Travolta', u'Uma Thurman', u'Samuel L....
...	...	...	...	...	...	...
974	7.4	True	PG	Comedy	116	[u'Dustin Hoffman', u'Jessica Lange', u'Teri G...
975	7.4	False	PG	Adventure	118	[u'Michael J. Fox', u'Christopher Lloyd', u'Ma...
976	7.4	False	PG-13	Action	138	[u'Russell Crowe', u'Paul Bettany', u'Billy Bo...
977	7.4	True	PG	Horror	114	[u'JoBeth Williams', u"Heather O'Rourke", u'Cr...
978	7.4	True	R	Crime	126	[u'Charlie Sheen', u'Michael Douglas', u'Tamar...

979 rows × 6 columns

31.What is the map Function In Pandas?

```
The map() function is used to map values of series according to input
correspondance.
Used for substituting each value in a series with another value,that may be
derived from a function,a dict or a
series.This function works only with series
syntax: Series.map(self,arg,na_action=None)
map accepts a dict or a series. values that are not found in the dict are
convert to NaN,unles dict has a defualt value
```

In [187]:
```
s1=pd.Series(["shubhangi","mansi","gauri","rutuja"])
print(s1)
```

```
0    shubhangi
1        mansi
2        gauri
3       rutuja
dtype: object
```

In [189]:
```
s1.map({"shubhangi":"mansi","gauri":"rutuja"})
```

Out[189]:
```
0     mansi
1       NaN
2    rutuja
3       NaN
dtype: object
```

32.How will you add a column to a pandas DataFrame?

In [192]:
```
df=pd.DataFrame({"list1":[10,20,40,60,80],
                 "arr1":np.array([1,2,3,4,5]),
                 "empty":np.empty(5)})
df
```

Out[192]:

	list1	arr1	empty
0	10	1	1.018558e-312
1	20	2	1.018558e-312
2	40	3	1.018558e-312
3	60	4	1.018558e-312
4	80	5	2.121996e-314

In [194]:
```
df["ones"]=np.ones(5,dtype=int)
df["arange"]=np.arange(1,6)
df
```

Out[194]:

	list1	arr1	empty	ones	arange
0	10	1	1.018558e-312	1	1
1	20	2	1.018558e-312	1	2
2	40	3	1.018558e-312	1	3
3	60	4	1.018558e-312	1	4
4	80	5	2.121996e-314	1	5

In [195]:
```
df.insert(5,"marks",[20,70,78,56,34])
df
```

Out[195]:

	list1	arr1	empty	ones	arange	marks
0	10	1	1.018558e-312	1	1	20
1	20	2	1.018558e-312	1	2	70
2	40	3	1.018558e-312	1	3	78
3	60	4	1.018558e-312	1	4	56
4	80	5	2.121996e-314	1	5	34

33.How will you add a column at a specific index to a pandas DataFrame?

In [196]:
```
df=pd.DataFrame({"list1":[10,20,40,60,80],
                 "arr1":np.array([1,2,3,4,5]),
                 "empty":np.empty(5)})
df
```

Out[196]:

	list1	arr1	empty
0	10	1	1.081059e-311
1	20	2	1.081059e-311
2	40	3	1.081059e-311
3	60	4	1.081059e-311
4	80	5	1.081059e-311

In [197]:
```
df.insert(3,"ones",np.ones(5))
df
```

Out[197]:

	list1	arr1	empty	ones
0	10	1	1.081059e-311	1.0
1	20	2	1.081059e-311	1.0
2	40	3	1.081059e-311	1.0
3	60	4	1.081059e-311	1.0
4	80	5	1.081059e-311	1.0

34.How to Delete Indices, Rows, or Columns From a Pandas DataFrame?

In [198]:
```
data={"Name":["Shubhangi","Mansi","rutuja","nikita"],
      "Age":[20,22,30,21],
      "marks":[90,91,92,93],
      "sr.No":np.arange(1,5)}
df=pd.DataFrame(data)
df
```

Out[198]:

	Name	Age	marks	sr.No
0	Shubhangi	20	90	1
1	Mansi	22	91	2
2	rutuja	30	92	3
3	nikita	21	93	4

In [199]:
```
#deleting rows from a dataframe
df.drop([2],axis=0)
```

Out[199]:

	Name	Age	marks	sr.No
0	Shubhangi	20	90	1
1	Mansi	22	91	2
3	nikita	21	93	4

In [200]:
```
#deleting column from dataframe
df.drop(["Age"],axis=1)
```

Out[200]:

	Name	marks	sr.No
0	Shubhangi	90	1
1	Mansi	91	2
2	rutuja	92	3
3	nikita	93	4

35.How to get the items which are not common to both series A and series B?

In [202]:
```
s1=pd.Series([1,5,7,4,2,6])
s2=pd.Series([1,2,9,8,4,3])
s3=set(s1).symmetric_difference(set(s2))
print("not common to both series A and series B:",s3)
```

```
not common to both series A and series B: {3, 5, 6, 7, 8, 9}
```

36.How to get the minimum, 25th percentile, median, 75th, and max of a numeric series?

In [210]:
```
df=pd.read_csv("Emp_Records.csv")
df
```

Out[210]:

	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 [212]:
```
df1=df.describe()
df1
```

Out[212]:

	Emp ID	Age in Yrs	Weight in Kgs	Age in Company	Salary
count	100.00000	97.000000	98.000000	96.000000	97.000000
mean	547652.10000	39.508557	57.653061	8.756250	119624.412371
std	257664.16679	12.086416	12.008676	8.323324	46239.001949
min	134841.00000	21.100000	40.000000	0.020000	42005.000000
25%	328643.75000	28.230000	50.000000	2.152500	84318.000000
50%	497414.00000	37.620000	56.000000	6.435000	118457.000000
75%	766040.00000	50.050000	61.000000	13.477500	162159.000000
max	979607.00000	59.470000	89.000000	34.320000	197537.000000

In [213]:
```
df1[3:]
```

Out[213]:

	Emp ID	Age in Yrs	Weight in Kgs	Age in Company	Salary
min	134841.00	21.10	40.0	0.0200	42005.0
25%	328643.75	28.23	50.0	2.1525	84318.0
50%	497414.00	37.62	56.0	6.4350	118457.0
75%	766040.00	50.05	61.0	13.4775	162159.0
max	979607.00	59.47	89.0	34.3200	197537.0

In [214]:
```
df.median()
```

```
C:\Users\Sunil\AppData\Local\Temp\ipykernel_4008\530051474.py:1: FutureWarning:
Dropping of nuisance columns in DataFrame reductions (with 'numeric_only=None')
is deprecated; in a future version this will raise TypeError.  Select only vali
d columns before calling the reduction.
  df.median()
```

Out[214]:
```
Emp ID            497414.000
Age in Yrs            37.620
Weight in Kgs         56.000
Age in Company         6.435
Salary            118457.000
dtype: float64
```

37.How can we sort the DataFrame?

In [215]:
```
df=pd.read_csv("Emp_Records.csv")
df
```

Out[215]:

	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 [216]:
```
df.sort_index(axis=1)       #sorting along columns
```

Out[216]:

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

100 rows × 7 columns

In [217]:
```
df.sort_index(axis=0,ascending=True)       #sorting along rows
```

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	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 [218]:
```
df.sort_index(axis=0,ascending=False)
```

Out[218]:

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

100 rows × 7 columns

In [219]:
```
df.sort_values(["First Name"])    #sorting along particular columns
```

Out[219]:

	Emp ID	First Name	Age in Yrs	Weight in Kgs	Age in Company	Salary	City
37	134841	Aaron	33.89	57.0	4.03	129836.0	Lima
38	726264	Aaron	43.63	50.0	10.14	162159.0	Wright
36	218791	Aaron	48.49	56.0	20.08	54402.0	Eckerty
43	247137	Alan	38.65	80.0	11.15	154810.0	Knoxville
41	227922	Amanda	35.02	40.0	10.28	114257.0	Lake Charles
...	...	...	...	...	...	...	...
17	214352	Theresa	24.66	59.0	2.52	197537.0	Toeterville
19	622406	Thomas	49.85	73.0	19.15	73862.0	Dutchtown
64	917937	Todd	25.93	74.0	1.22	163560.0	Randallstown
13	301576	Wayne	21.10	87.0	0.02	92758.0	Maida
78	806955	William	50.05	50.0	17.36	73734.0	Mary Esther

100 rows × 7 columns

38.How to drop duplicate rows from DataFrame?

In [221]:
```
df=pd.DataFrame({"A":[1,5,1,4],
                 "B":[100,300,100,300],
                 "C":[3,3,3,4]})
df
```

Out[221]:

	A	B	C
0	1	100	3
1	5	300	3
2	1	100	3
3	4	300	4

In [223]:
```
df.duplicated()
```

Out[223]:
```
0    False
1    False
2     True
3    False
dtype: bool
```

In [224]:
```
df[~(df.duplicated())]
```

Out[224]:

	A	B	C
0	1	100	3
1	5	300	3
3	4	300	4

39. How to drop duplicate columns from DataFrame?

In [225]:
```
df=pd.DataFrame({"A":[1,5,1,4],
                 "B":[100,300,100,300],
                 "C":[3,3,3,4],
                 "D":[100,300,100,300]})
df
```

Out[225]:

	A	B	C	D
0	1	100	3	100
1	5	300	3	300
2	1	100	3	100
3	4	300	4	300

In [226]:
```
Y=df.T
Y
```

Out[226]:

	0	1	2	3
A	1	5	1	4
B	100	300	100	300
C	3	3	3	4
D	100	300	100	300

In [227]:
```
X=Y[~(Y.duplicated())]
X
```

Out[227]:

	0	1	2	3
A	1	5	1	4
B	100	300	100	300
C	3	3	3	4

In [229]:
```
df=X.T
df
```

Out[229]:

	A	B	C
0	1	100	3
1	5	300	3
2	1	100	3
3	4	300	4

40.Write a Pandas program to split a string of a column of a given DataFrame into multiple columns(Split Name and Surname

In [230]:
```
df=pd.DataFrame({"name":["SHUBHANGI DHIKALE","MANSI GAWALI","RUTUJA DESHMUKH","KR
                 "Age":[24,23,20,19],
                 "location":["pune","mumbai","nashik","delhi"]})
df
```

Out[230]:

	name	Age	location
0	SHUBHANGI DHIKALE	24	pune
1	MANSI GAWALI	23	mumbai
2	RUTUJA DESHMUKH	20	nashik
3	KR KUMAR	19	delhi

In [231]:
```
df[["first_name","last name"]]=df["name"].str.split(" ",expand=True)
df
```

Out[231]:

	name	Age	location	first_name	last name
0	SHUBHANGI DHIKALE	24	pune	SHUBHANGI	DHIKALE
1	MANSI GAWALI	23	mumbai	MANSI	GAWALI
2	RUTUJA DESHMUKH	20	nashik	RUTUJA	DESHMUKH
3	KR KUMAR	19	delhi	KR	KUMAR

In [ ]: