pandas_basic_NCERT, Cse_study_50_Basic_Questions

In [1]:

```
# creating series
import pandas as pd
series1 =pd.Series([10,20,30])
display(series1)
```

```
0    10
1    20
2    30
dtype: int64
```

In [2]:

```
series2 = pd.Series(["ravi","saif","john"],index = [3,2,1])
```

In [3]:

```
series2
```

Out[3]:

```
3    ravi
2    saif
1    john
dtype: object
```

In [4]:

```
# creating series from numpy arrays
import numpy as np
array1 =np.arange(6)
array1
```

Out[4]:

```
array([0, 1, 2, 3, 4, 5])
```

In [5]:

```
series3  =pd.Series(array1,index = [1,2,3,4,5,6])
series3
```

Out[5]:

```
1    0
2    1
3    2
4    3
5    4
6    5
dtype: int32
```

In [6]:

```
# creation of series from dictionary
dict1 = {'country':'capital','India': 'NewDelhi', 'UK': 'London', 'Japan': 'Tokyo'}
series4 = pd.Series(dict1)
display(series4)
```

```
country     capital
India      NewDelhi
UK           London
Japan         Tokyo
dtype: object
```

In [7]:

```
# accessing Elements of series
seriesNum = pd.Series([10,20,30])
seriesNum[1]
```

Out[7]:

```
20
```

In [8]:

```
seriesMnths = pd.Series([2,3,4],index=["Feb","Mar","Apr"])
seriesMnths["Feb"]
```

Out[8]:

```
2
```

In [9]:

```
seriesCapCntry = pd.Series(['NewDelhi', 'WashingtonDC', 'London', 'Paris'], index=['India', 'USA', 'UK', 'France'])
```

In [10]:

```
seriesCapCntry
```

Out[10]:

```
India         NewDelhi
USA       WashingtonDC
UK              London
France           Paris
dtype: object
```

Attributes of Series

In [11]:

```
seriesCapCntry.name = 'Capitals'   # assigning  name to the series
```

In [12]:

```
seriesCapCntry
```

Out[12]:

```
India         NewDelhi
USA       WashingtonDC
UK              London
France           Paris
Name: Capitals, dtype: object
```

In [13]:

```
seriesCapCntry.index.name = "countires"
```

In [14]:

```
seriesCapCntry
```

Out[14]:

```
countires
India         NewDelhi
USA       WashingtonDC
UK              London
France           Paris
Name: Capitals, dtype: object
```

In [15]:

```
# seriesCapCntry.values.name = "capital"
```

method of series

In [16]:

```
# method of series
seriesTenTwenty =pd.Series(np.arange(10,30,2))
```

In [17]:

```
seriesTenTwenty
```

Out[17]:

```
0    10
1    12
2    14
3    16
4    18
5    20
6    22
7    24
8    26
9    28
dtype: int32
```

In [18]:

```
seriesTenTwenty.head()
```

Out[18]:

```
0    10
1    12
2    14
3    16
4    18
dtype: int32
```

In [19]:

```
seriesTenTwenty.tail()
```

Out[19]:

```
5    20
6    22
7    24
8    26
9    28
dtype: int32
```

In [20]:

```
seriesTenTwenty.count()
```

Out[20]:

```
10
```

In [21]:

```
seriesA = pd.Series([1,2,3,4,5], index = ['a', 'b', 'c', 'd', 'e'])
```

In [22]:

```
seriesA
```

Out[22]:

```
a    1
b    2
c    3
d    4
e    5
dtype: int64
```

In [23]:

```
seriesB = pd.Series([10,20,-10,-50,100], index = ['z', 'y', 'a', 'c', 'e'])
```

In [24]:

```
seriesB
```

Out[24]:

```
z     10
y     20
a    -10
c    -50
e    100
dtype: int64
```

normal addition

In [25]:

```
seriesA + seriesB
```

Out[25]:

```
a     -9.0
b      NaN
c    -47.0
d      NaN
e    105.0
y      NaN
z      NaN
dtype: float64
```

In [26]:

```
# using fill_value =0 means wherever in series A  the element will be null that  that wil
l get replace with "0"
seriesA.add(seriesB,fill_value=0)
```

Out[26]:

```
a     -9.0
b      2.0
c    -47.0
d      4.0
e    105.0
y     20.0
z     10.0
dtype: float64
```

In [27]:

```
seriesA.sub(seriesB, fill_value=1000)
```

Out[27]:

```
a     11.0
b   -998.0
c     53.0
d   -996.0
e    -95.0
y    980.0
z    990.0
```

```
dtype: float64
```

DataFrame

In [28]:

```
import numpy as np
array1  =np.array([10,20,30])
array2 =np.array([100,200,300])
array3=np.array([-10,-20,-30,-40])
df4 = pd.DataFrame([array1,array2,array3],columns = ["A","B","C","D"])
```

In [29]:

```
df4
```

Out[29]:

	A	B	C	D
0	10	20	30	NaN
1	100	200	300	NaN
2	-10	-20	-30	-40.0

In [30]:

```
# Create list of dictionaries
listDict = [{'a':10, 'b':20}, {'a':5, 'b':10, 'c':20}]
df5 =pd.DataFrame(listDict)
display(df5)
```

	a	b	c
0	10	20	NaN
1	5	10	20.0

In [31]:

```
# Creation of DataFrame from Dictionary of Lists
dictForest = {'State': ['Assam', 'Delhi', 'Kerala'],'GArea': [78438, 1483, 38852] ,'VDF'
: [2797, 6.72,1663]}
```

In [32]:

```
df6=pd.DataFrame(dictForest)
display(df6)
```

	State	GArea	VDF
0	Assam	78438	2797.00
1	Delhi	1483	6.72
2	Kerala	38852	1663.00

In [33]:

```
# Creation of DataFrame from Series
seriesA = pd.Series([1,2,3,4,5],index = ['a', 'b', 'c', 'd', 'e'])
seriesB = pd.Series ([1000,2000,-1000,-5000,1000],index = ['a', 'b', 'c', 'd', 'e'])
seriesB = pd.Series([10,20,-10,-50,100],index = ['z', 'y', 'a', 'c', 'e'])
```

In [34]:

```
df7 = pd.DataFrame([seriesA,seriesB,seriesB])
```

```
display(df7)
```

	a	b	c	d	e	z	y
0	1.0	2.0	3.0	4.0	5.0	NaN	NaN
1	-10.0	NaN	-50.0	NaN	100.0	10.0	20.0
2	-10.0	NaN	-50.0	NaN	100.0	10.0	20.0

In [35]:

```
df8 = pd.DataFrame([seriesA, seriesB])
df8
```

Out[35]:

	a	b	c	d	e	z	y
0	1.0	2.0	3.0	4.0	5.0	NaN	NaN
1	-10.0	NaN	-50.0	NaN	100.0	10.0	20.0

In [36]:

```
ResultSheet={'Arnab': pd.Series([90, 91, 97],index=['Maths','Science','Hindi']),'Ramit':
pd.Series([92, 81, 96],index=['Maths','Science','Hindi']),'Samridhi': pd.Series([89, 91,
88],index=['Maths','Science','Hindi']),'Riya': pd.Series([81, 71, 67],index=['Maths','Sc
ience','Hindi']),'Mallika': pd.Series([94, 95, 99],index=['Maths','Science','Hindi'])}
```

In [37]:

```
ResultSheet={'Arnab': pd.Series([90, 91, 97],index=['Maths','Science','Hindi']),'Ramit':
pd.Series([92, 81, 96],index=['Maths','Science','Hindi']),'Samridhi': pd.Series([89, 91,
88],index=['Maths','Science','Hindi']),'Riya': pd.Series([81, 71, 67],index=['Maths','Sc
ience','Hindi']),'Mallika': pd.Series([94, 95, 99],index=['Maths','Science','Hindi'])}
```

In [38]:

```
ResultSheetDF = pd.DataFrame(ResultSheet)
```

In [39]:

```
display(ResultSheetDF)
```

	Arnab	Ramit	Samridhi	Riya	Mallika
Maths	90	92	89	81	94
Science	91	81	91	71	95
Hindi	97	96	88	67	99

In [40]:

```
ResultSheetDF["preeti"] = [89,90,96]
```

In [41]:

```
display(ResultSheetDF)
```

	Arnab	Ramit	Samridhi	Riya	Mallika	preeti
Maths	90	92	89	81	94	89
Science	91	81	91	71	95	90
Hindi	97	96	88	67	99	96

In [42]:

```
# addition of new row
ResultSheetDF.loc["English"] = np.random.randint(85, 89, size=6, dtype=int)
```

In [43]:

```
display(ResultSheetDF)
```

	Arnab	Ramit	Samridhi	Riya	Mallika	preeti
Maths	90	92	89	81	94	89
Science	91	81	91	71	95	90
Hindi	97	96	88	67	99	96
English	87	85	85	88	86	86

In [44]:

```
# changing rows values
# lets consider maths value  = 99
ResultSheetDF.loc["Maths"] =99
```

In [45]:

```
ResultSheetDF
```

Out[45]:

	Arnab	Ramit	Samridhi	Riya	Mallika	preeti
Maths	99	99	99	99	99	99
Science	91	81	91	71	95	90
Hindi	97	96	88	67	99	96
English	87	85	85	88	86	86

In [46]:

```
# Deleting Rows or Columns from a DataFrame
```

In [47]:

```
ResultSheetDF = ResultSheetDF.drop("Science",axis =0)   # here axis =0 i.e its operating rows by rows
```

In [48]:

```
ResultSheetDF
```

Out[48]:

	Arnab	Ramit	Samridhi	Riya	Mallika	preeti
Maths	99	99	99	99	99	99
Hindi	97	96	88	67	99	96
English	87	85	85	88	86	86

In [49]:

```
ResultSheetDF.drop(["Arnab","Ramit"],axis =1) # here axis = 1 means columns by columns
```

Out[49]:

	Samridhi	Riya	Mallika	preeti
Maths	99	99	99	99
Hindi	88	67	99	96

English	**Samridhi 85**	**Riya 88**	**Mallika 86**	**preeti 86**

In [50]:

```
ResultSheetDF
```

Out[50]:

	Arnab	Ramit	Samridhi	Riya	Mallika	preeti
Maths	99	99	99	99	99	99
Hindi	97	96	88	67	99	96
English	87	85	85	88	86	86

In [51]:

```
# Renaming Row Labels of a DataFrame
ResultSheetDF.rename({"Maths":"sub1","Hindi":"sub2","English":"sub3"},axis = "index")
```

Out[51]:

	Arnab	Ramit	Samridhi	Riya	Mallika	preeti
sub1	99	99	99	99	99	99
sub2	97	96	88	67	99	96
sub3	87	85	85	88	86	86

In [52]:

```
ResultSheetDF
```

Out[52]:

	Arnab	Ramit	Samridhi	Riya	Mallika	preeti
Maths	99	99	99	99	99	99
Hindi	97	96	88	67	99	96
English	87	85	85	88	86	86

```
                                   Accessing DataFrames Element through  Indexing
```

In [53]:

```
ResultSheetDF.loc['English']
```

Out[53]:

```
Arnab       87
Ramit       85
Samridhi    85
Riya        88
Mallika     86
preeti      86
Name: English, dtype: int64
```

In [54]:

```
ResultSheet={'Arnab': pd.Series([90, 91, 97],index=['Maths','Science','Hindi']),'Ramit':
pd.Series([92, 81, 96],index=['Maths','Science','Hindi']),'Samridhi': pd.Series([89, 91,
88],index=['Maths','Science','Hindi']),'Riya': pd.Series([81, 71, 67],index=['Maths','Sc
ience','Hindi']),'Mallika': pd.Series([94, 95, 99],index=['Maths','Science','Hindi'])}
```

In [55]:

```
ResultDF = pd.DataFrame(ResultSheet)
```

In [56]:

```
ResultDF
```

Out[56]:

	Arnab	Ramit	Samridhi	Riya	Mallika
Maths	90	92	89	81	94
Science	91	81	91	71	95
Hindi	97	96	88	67	99

In [57]:

```
ResultDF.iloc[0:2,0:4]
# When a single column label is passed, it returns the column
# as a Series.
#  ResultDF.loc[:,'Arnab']
```

Out[57]:

	Arnab	Ramit	Samridhi	Riya
Maths	90	92	89	81
Science	91	81	91	71

In [58]:

```
ResultDF.loc['Maths': 'Science', "Arnab":"Samridhi"]
```

Out[58]:

	Arnab	Ramit	Samridhi
Maths	90	92	89
Science	91	81	91

In [ ]:

In [59]:

```
ResultDF.loc['Science']
```

Out[59]:

```
Arnab       91
Ramit       81
Samridhi    91
Riya        71
Mallika     95
Name: Science, dtype: int64
```

In [60]:

```
ResultDF.loc['Maths'] > 90
```

Out[60]:

```
Arnab       False
Ramit        True
Samridhi    False
Riya        False
Mallika      True
Name: Maths, dtype: bool
```

In [61]:

```
ResultDF.loc[:,'Arnab']>90
```

Out[61]:

```
Maths      False
Science     True
Hindi       True
Name: Arnab, dtype: bool
```

joining dataframe

In [62]:

```
dFrame1=pd.DataFrame([[1, 2, 3], [4, 5], [6]], columns=['C1', 'C2', 'C3'], index=['R1',
'R2', 'R3'])
```

In [63]:

```
dFrame1
```

Out[63]:

	C1	C2	C3
R1	1	2.0	3.0
R2	4	5.0	NaN
R3	6	NaN	NaN

In [64]:

```
dFrame2=pd.DataFrame([[10, 20], [30], [40, 50]], columns=['C2', 'C5'], index=['R4', 'R2'
, 'R5'])
```

In [65]:

```
dFrame1=dFrame1.append(dFrame2)
```

```
C:\Users\mrpam\AppData\Local\Temp\ipykernel_7452\2125746391.py:1: FutureWarning: The fram
e.append method is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  dFrame1=dFrame1.append(dFrame2)
```

In [66]:

```
dFrame1
```

Out[66]:

	C1	C2	C3	C5
R1	1.0	2.0	3.0	NaN
R2	4.0	5.0	NaN	NaN
R3	6.0	NaN	NaN	NaN
R4	NaN	10.0	NaN	20.0
R2	NaN	30.0	NaN	NaN
R5	NaN	40.0	NaN	50.0

-----casestudy_NCERT--------

In [67]:

In [67]:

```
name = ["madhu","kusum","kinshuk","ankit","shruti"]
y_2014 = np.random.randint(100,40000,size = 5)
y_2015 = np.random.randint(12000,45000,size = 5)
y_2016 = np.random.randint(20000,125000,size = 5)
y_2017 = np.random.randint(5000,90000,size = 5)
```

In [68]:

```
df_sales = pd.DataFrame({"name":name,"y_2014":y_2014,"y_2015":y_2015,"y_2016":y_2016,"y_2
017":y_2017})
```

In [69]:

```
df_sales.set_index("name")
```

Out[69]:

	y_2014	y_2015	y_2016	y_2017
name				
madhu	25405	33280	106867	11627
kusum	30278	17755	27783	25399
kinshuk	15744	18040	31150	5923
ankit	4673	29118	121970	22681
shruti	9755	27285	70748	7912

In [70]:

```
df_sales.loc[0]
```

Out[70]:

```
name         madhu
y_2014       25405
y_2015       33280
y_2016      106867
y_2017       11627
Name: 0, dtype: object
```

In [71]:

```
df_sales.loc[0:2]
```

Out[71]:

	name	y_2014	y_2015	y_2016	y_2017
0	madhu	25405	33280	106867	11627
1	kusum	30278	17755	27783	25399
2	kinshuk	15744	18040	31150	5923

In [72]:

```
df_sales.loc[0:4:2]
```

Out[72]:

	name	y_2014	y_2015	y_2016	y_2017
0	madhu	25405	33280	106867	11627
2	kinshuk	15744	18040	31150	5923
4	shruti	9755	27285	70748	7912

In [73]:

In [73]:

```
import pandas as pd
```

In [74]:

```
marksUT= {'Name':['Raman','Raman','Raman','Zuhaire','Zuhaire','Zuhaire', 'Ashravy','Ashravy','Ashravy','Mishti','Mishti','Mishti'],'UT':[1,2,3,1,2,3,1,2,3,1,2,3],'Maths':[22,21,14,20,23,22,23,24,12,15,18,17],'Science':[21,20,19,17,15,18,19,22,25,22,21,18],'S.St':[18,17,15,22,21,19,20,24,19,25,25,20],'Hindi':[20,22,24,24,25,23,15,17,21,22,24,25],'Eng':[21,24,23,19,15,13,22,21,23,22,23,20]}
```

In [75]:

```
marksUT
```

Out[75]:

```
{'Name': ['Raman',
  'Raman',
  'Raman',
  'Zuhaire',
  'Zuhaire',
  'Zuhaire',
  'Ashravy',
  'Ashravy',
  'Ashravy',
  'Mishti',
  'Mishti',
  'Mishti'],
 'UT': [1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3],
 'Maths': [22, 21, 14, 20, 23, 22, 23, 24, 12, 15, 18, 17],
 'Science': [21, 20, 19, 17, 15, 18, 19, 22, 25, 22, 21, 18],
 'S.St': [18, 17, 15, 22, 21, 19, 20, 24, 19, 25, 25, 20],
 'Hindi': [20, 22, 24, 24, 25, 23, 15, 17, 21, 22, 24, 25],
 'Eng': [21, 24, 23, 19, 15, 13, 22, 21, 23, 22, 23, 20]}
```

In [76]:

```
df = pd.DataFrame(marksUT)
display(df)
```

	Name	UT	Maths	Science	S.St	Hindi	Eng
0	Raman	1	22	21	18	20	21
1	Raman	2	21	20	17	22	24
2	Raman	3	14	19	15	24	23
3	Zuhaire	1	20	17	22	24	19
4	Zuhaire	2	23	15	21	25	15
5	Zuhaire	3	22	18	19	23	13
6	Ashravy	1	23	19	20	15	22
7	Ashravy	2	24	22	24	17	21
8	Ashravy	3	12	25	19	21	23
9	Mishti	1	15	22	25	22	22
10	Mishti	2	18	21	25	24	23
11	Mishti	3	17	18	20	25	20

--------------Descriptive Statistics-------------

Descriptive Statistics are used to summarise the given data. In other words, they refer to the methods which are used to get some basic idea about the data.

In [77]:

```
# Calculating Maximum Values
df.max()
```

Out[77]:

```
Name       Zuhaire
UT               3
Maths           24
Science         25
S.St            25
Hindi           25
Eng             24
dtype: object
```

In [78]:

```
print(df.max(numeric_only = True)) # set numeric_only  =True
```

```
UT          3
Maths      24
Science    25
S.St       25
Hindi      25
Eng        24
dtype: int64
```

In [79]:

```
# Write the statements to output the  maximum marks obtained in each subject in Unit Test
2
df
```

Out[79]:

	Name	UT	Maths	Science	S.St	Hindi	Eng
0	Raman	1	22	21	18	20	21
1	Raman	2	21	20	17	22	24
2	Raman	3	14	19	15	24	23
3	Zuhaire	1	20	17	22	24	19
4	Zuhaire	2	23	15	21	25	15
5	Zuhaire	3	22	18	19	23	13
6	Ashravy	1	23	19	20	15	22
7	Ashravy	2	24	22	24	17	21
8	Ashravy	3	12	25	19	21	23
9	Mishti	1	15	22	25	22	22
10	Mishti	2	18	21	25	24	23
11	Mishti	3	17	18	20	25	20

In [80]:

```
dfUT2 = df[df.UT ==2]
dfUT2[["Maths","Science","S.St","Hindi",'Eng']].max(numeric_only=True)
```

Out[80]:

```
Maths      24
Science    22
S.St       25
Hindi      25
Eng        24
dtype: int64
```

In [81]:

```
dfUT2.max(numeric_only=True) # here by defualt axis =0   means columns wise operation
```

Out[81]:

```
UT          2
Maths      24
Science    22
S.St       25
Hindi      25
Eng        24
dtype: int64
```

In [82]:

```
df.max(axis= 1)
```

```
C:\Users\mrpam\AppData\Local\Temp\ipykernel_7452\652354474.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 valid columns before calling the
reduction.
  df.max(axis= 1)
```

Out[82]:

```
0     22
1     24
2     24
3     24
4     25
5     23
6     23
7     24
8     25
9     25
10    25
11    25
dtype: int64
```

In [83]:

```
df.min()
```

Out[83]:

```
Name       Ashravy
UT               1
Maths           12
Science         15
S.St            15
Hindi           15
Eng             13
dtype: object
```

In [84]:

```
# Write the statements to display the minimum marks obtained by a particular
# student 'Mishti' in all the unit tests for each subject.
```

In [85]:

```
df
```

Out[85]:

	Name	UT	Maths	Science	S.St	Hindi	Eng
0	Raman	1	22	21	18	20	21
1	Raman	2	21	20	17	22	24
2	Raman	3	14	19	15	24	23

	Name	UT	Maths	Science	S.St	Hindi	Eng
3	Zuhaire	1	20	17	22	24	19
4	Zuhaire	2	23	15	21	25	15
5	Zuhaire	3	22	18	19	23	13
6	Ashravy	1	23	19	20	15	22
7	Ashravy	2	24	22	24	17	21
8	Ashravy	3	12	25	19	21	23
9	Mishti	1	15	22	25	22	22
10	Mishti	2	18	21	25	24	23
11	Mishti	3	17	18	20	25	20

In [86]:

```
dfMishti = df[df.Name =="Mishti"]
```

In [87]:

```
dfMishti = df.loc[df.Name == 'Mishti']
```

In [88]:

```
dfMishti
```

Out[88]:

	Name	UT	Maths	Science	S.St	Hindi	Eng
9	Mishti	1	15	22	25	22	22
10	Mishti	2	18	21	25	24	23
11	Mishti	3	17	18	20	25	20

In [89]:

```
dfMishti[["Maths","Science","S.St","Hindi","Eng"]].min()
```

Out[89]:

```
Maths      15
Science    18
S.St       20
Hindi      22
Eng        20
dtype: int64
```

In [90]:

```
# calculating sum values
df.sum()
```

Out[90]:

```
Name       RamanRamanRamanZuhaireZuhaireZuhaireAshravyAsh...
UT                                                        24
Maths                                                    231
Science                                                  237
S.St                                                     245
Hindi                                                    262
Eng                                                      246
dtype: object
```

In [91]:

```
df["Maths"].sum()
```

Out[91]:

```
231
```

Write the python statement to print the total marks secured by raman in each subject.

In [92]:

```
dfRaman =df.loc[df.Name =="Raman"]
```

In [93]:

```
dfRaman[["Maths","Science","S.St","Hindi","Eng"]].sum()
```

Out[93]:

```
Maths      57
Science    60
S.St       50
Hindi      66
Eng        68
dtype: int64
```

In [94]:

```
dfRaman[["Maths","Science","S.St","Hindi","Eng"]].sum()
```

Out[94]:

```
Maths      57
Science    60
S.St       50
Hindi      66
Eng        68
dtype: int64
```

In [95]:

```
# to print marks scored by raman in all subject in each columns
dfRaman[["Maths","Science","S.St","Hindi","Eng"]].sum(axis =1)
```

Out[95]:

```
0    102
1    104
2     95
dtype: int64
```

```
   ------------Calculating Number of Values--------------
```

In [96]:

```
df.count()
```

Out[96]:

```
Name       12
UT         12
Maths      12
Science    12
S.St       12
Hindi      12
Eng        12
dtype: int64
```

In [97]:

```
#row wise
df.count(axis = 1)
```

Out[97]:

Out[97]:

```
0      7
1      7
2      7
3      7
4      7
5      7
6      7
7      7
8      7
9      7
10     7
11     7
dtype: int64
```

```
          ----------Calculating Mean--------------
```

In [98]:

```
df.mean()
```

```
C:\Users\mrpam\AppData\Local\Temp\ipykernel_7452\3698961737.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 valid columns before calling the
reduction.
  df.mean()
```

Out[98]:

```
UT           2.000000
Maths       19.250000
Science     19.750000
S.St        20.416667
Hindi       21.833333
Eng         20.500000
dtype: float64
```

In [99]:

```
df_Zuhaire = df[df.Name =="Zuhaire"]
```

In [100]:

```
df_Zuhaire
```

Out[100]:

	Name	UT	Maths	Science	S.St	Hindi	Eng
3	Zuhaire	1	20	17	22	24	19
4	Zuhaire	2	23	15	21	25	15
5	Zuhaire	3	22	18	19	23	13

In [102]:

```
df_Zuhaire.loc[:,"Maths":"Eng"]
# here we can see, : single columns passing to select columns and return columns as serie
s
# selecting to show marks of subject so ranging columns
```

Out[102]:

	Maths	Science	S.St	Hindi	Eng
3	20	17	22	24	19
4	23	15	21	25	15
5	22	18	19	23	13

	Maths	Science	S.St	Hindi	Eng

In [103]:

```
# When a single column label is passed, it returns the column as a Series.
df.loc[:,'Maths']
```

Out[103]:

```
0      22
1      21
2      14
3      20
4      23
5      22
6      23
7      24
8      12
9      15
10     18
11     17
Name: Maths, dtype: int64
```

In [104]:

```
# we can aslo customize columns and rows using loc
df.loc[3:5,"Maths":"Hindi"]
```

Out[104]:

	Maths	Science	S.St	Hindi
3	20	17	22	24
4	23	15	21	25
5	22	18	19	23

In [105]:

```
df
```

Out[105]:

	Name	UT	Maths	Science	S.St	Hindi	Eng
0	Raman	1	22	21	18	20	21
1	Raman	2	21	20	17	22	24
2	Raman	3	14	19	15	24	23
3	Zuhaire	1	20	17	22	24	19
4	Zuhaire	2	23	15	21	25	15
5	Zuhaire	3	22	18	19	23	13
6	Ashravy	1	23	19	20	15	22
7	Ashravy	2	24	22	24	17	21
8	Ashravy	3	12	25	19	21	23
9	Mishti	1	15	22	25	22	22
10	Mishti	2	18	21	25	24	23
11	Mishti	3	17	18	20	25	20

Write the statements to get an average of marks obtained by Zuhaire in all the Unit Tests.

In [106]:

```
df[df["Name" ] == "Zuhaire"]
```

Out[106]:

Out[106]:

	Name	UT	Maths	Science	S.St	Hindi	Eng
3	Zuhaire	1	20	17	22	24	19
4	Zuhaire	2	23	15	21	25	15
5	Zuhaire	3	22	18	19	23	13

In [107]:

```
dfZuhaire = df.loc[3:5,"Maths":"Eng"]
```

In [108]:

```
dfZuhaire
```

Out[108]:

	Maths	Science	S.St	Hindi	Eng
3	20	17	22	24	19
4	23	15	21	25	15
5	22	18	19	23	13

In [109]:

```
dfZuhaire.mean(axis =1)
```

Out[109]:

```
3    20.4
4    19.8
5    19.0
dtype: float64
```

```
       ------------Calculating Median---------
```

In [110]:

```
df.median()
```

```
C:\Users\mrpam\AppData\Local\Temp\ipykernel_7452\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 valid columns before calling the
reduction.
  df.median()
```

Out[110]:

```
UT          2.0
Maths      20.5
Science    19.5
S.St       20.0
Hindi      22.5
Eng        21.5
dtype: float64
```

In [111]:

```
df["Maths"]
```

Out[111]:

```
0     22
1     21
2     14
3     20
4     23
```

```
5      22
6      23
7      24
8      12
9      15
10     18
11     17
Name: Maths, dtype: int64
```

In [112]:

```
df[df.UT ==1]["Maths"].median()
```

Out[112]:

```
21.0
```

In [113]:

```
# Calculating Mode
df.Hindi.mode()
```

Out[113]:

```
0    24
Name: Hindi, dtype: int64
```

---------------Calculating Quartile

In [114]:

```
df.quantile
```

Out[114]:

```
<bound method DataFrame.quantile of         Name  UT  Maths  Science  S.St  Hindi  Eng
0      Raman   1     22       21    18     20   21
1      Raman   2     21       20    17     22   24
2      Raman   3     14       19    15     24   23
3    Zuhaire   1     20       17    22     24   19
4    Zuhaire   2     23       15    21     25   15
5    Zuhaire   3     22       18    19     23   13
6    Ashravy   1     23       19    20     15   22
7    Ashravy   2     24       22    24     17   21
8    Ashravy   3     12       25    19     21   23
9     Mishti   1     15       22    25     22   22
10    Mishti   2     18       21    25     24   23
11    Mishti   3     17       18    20     25   20>
```

In [115]:

```
df.quantile()
```

Out[115]:

```
UT          2.0
Maths      20.5
Science    19.5
S.St       20.0
Hindi      22.5
Eng        21.5
Name: 0.5, dtype: float64
```

In [116]:

```
df.quantile(q =.75)
```

Out[116]:

```
UT          3.00
Maths      22.25
Science    21.25
```

```
S.St       22.50
Hindi      24.00
Eng        23.00
Name: 0.75, dtype: float64
```

In [ ]:

```

```

Write the statement to display the first and third quartiles of all subjects.

In [117]:

```
dfSubject=df[['Maths','Science','S.St','Hindi','Eng']]
```

In [118]:

```
dfSubject
```

Out[118]:

	Maths	Science	S.St	Hindi	Eng
0	22	21	18	20	21
1	21	20	17	22	24
2	14	19	15	24	23
3	20	17	22	24	19
4	23	15	21	25	15
5	22	18	19	23	13
6	23	19	20	15	22
7	24	22	24	17	21
8	12	25	19	21	23
9	15	22	25	22	22
10	18	21	25	24	23
11	17	18	20	25	20

In [119]:

```
dfSubject.Maths.quantile(q=.25)
```

Out[119]:

```
16.5
```

In [120]:

```
dfSubject.quantile([.25,.75])
```

Out[120]:

	Maths	Science	S.St	Hindi	Eng
0.25	16.50	18.00	18.75	20.75	19.75
0.75	22.25	21.25	22.50	24.00	23.00

In [121]:

```
# Calculating Variance
df[['Maths','Science','S.St','Hindi','Eng']].var()
```

Out[121]:

```
Maths      15.840909
Science     7.113636
Science      7.113636
S.St         9.901515
Hindi        9.969697
Eng         11.363636
dtype: float64
```

In [ ]:

```
# Calculating Standard Deviation
```

In [122]:

```
df[['Maths','Science','S.St','Hindi','Eng']].std()
```

Out[122]:

```
Maths      3.980064
Science    2.667140
S.St       3.146667
Hindi      3.157483
Eng        3.370999
dtype: float64
```

----DATA AGGREGATIONS----------

In [123]:

```
df
```

Out[123]:

	Name	UT	Maths	Science	S.St	Hindi	Eng
0	Raman	1	22	21	18	20	21
1	Raman	2	21	20	17	22	24
2	Raman	3	14	19	15	24	23
3	Zuhaire	1	20	17	22	24	19
4	Zuhaire	2	23	15	21	25	15
5	Zuhaire	3	22	18	19	23	13
6	Ashravy	1	23	19	20	15	22
7	Ashravy	2	24	22	24	17	21
8	Ashravy	3	12	25	19	21	23
9	Mishti	1	15	22	25	22	22
10	Mishti	2	18	21	25	24	23
11	Mishti	3	17	18	20	25	20

In [ ]:

```

```

In [124]:

```
df.aggregate("max")
```

Out[124]:

```
Name       Zuhaire
UT               3
Maths           24
Science         25
S.St            25
Hindi           25
Eng             24
dtype: object
```

```
dtype: object
```

In [125]:

```
# £to use mutiple aggregate function
df.aggregate(["max","count","min"])
```

Out[125]:

	Name	UT	Maths	Science	S.St	Hindi	Eng
max	Zuhaire	3	24	25	25	25	24
count	12	12	12	12	12	12	12
min	Ashravy	1	12	15	15	15	13

In [126]:

```
df['Maths'].aggregate(["max","min"])
```

Out[126]:

```
max     24
min     12
Name: Maths, dtype: int64
```

We can also use the parameter axis with aggregate function. By default, the value of axis is zero, means columns

In [127]:

```
df[['Maths','Science']].aggregate('sum',axis=1)
```

Out[127]:

```
0      43
1      41
2      33
3      37
4      38
5      40
6      42
7      46
8      37
9      37
10     39
11     35
dtype: int64
```

Sorting a DataFrame

Dataframe.sort_values(by,axis =0,ascending = True)

In [128]:

```
df.sort_values(by=["Name"],axis =0,ascending = True)
```

Out[128]:

	Name	UT	Maths	Science	S.St	Hindi	Eng
6	Ashravy	1	23	19	20	15	22
7	Ashravy	2	24	22	24	17	21
8	Ashravy	3	12	25	19	21	23
9	Mishti	1	15	22	25	22	22
10	Mishti	2	18	21	25	24	23
11	Mishti	3	17	18	20	25	20
0	Raman	1	22	21	18	20	21

	Name	UT	Maths	Science	S.St	Hindi	Eng
1	Raman	2	21	20	17	22	24
2	Raman	3	14	19	15	24	23
3	Zuhaire	1	20	17	22	24	19
4	Zuhaire	2	23	15	21	25	15
5	Zuhaire	3	22	18	19	23	13

In [129]:

```
# to obtain sorted list of marks scored by all
# students in Science in Unit Test 2 can be used:
DFut2 =df[df.UT ==2]
```

In [130]:

```
DFut2
```

Out[130]:

	Name	UT	Maths	Science	S.St	Hindi	Eng
1	Raman	2	21	20	17	22	24
4	Zuhaire	2	23	15	21	25	15
7	Ashravy	2	24	22	24	17	21
10	Mishti	2	18	21	25	24	23

In [131]:

```
df.sort_values(by =["Hindi","Science"],axis =0,ascending =True)
```

Out[131]:

	Name	UT	Maths	Science	S.St	Hindi	Eng
6	Ashravy	1	23	19	20	15	22
7	Ashravy	2	24	22	24	17	21
0	Raman	1	22	21	18	20	21
8	Ashravy	3	12	25	19	21	23
1	Raman	2	21	20	17	22	24
9	Mishti	1	15	22	25	22	22
5	Zuhaire	3	22	18	19	23	13
3	Zuhaire	1	20	17	22	24	19
2	Raman	3	14	19	15	24	23
10	Mishti	2	18	21	25	24	23
4	Zuhaire	2	23	15	21	25	15
11	Mishti	3	17	18	20	25	20

In [132]:

```
df.sort_values(by=["UT"],axis =0,ascending = True)
```

Out[132]:

	Name	UT	Maths	Science	S.St	Hindi	Eng
0	Raman	1	22	21	18	20	21
3	Zuhaire	1	20	17	22	24	19
6	Ashravy	1	23	19	20	15	22

	Name	UT	Maths	Science	S.St	Hindi	Eng
9	Mishti	1	15	22	25	22	22
1	Raman	2	21	20	17	22	24
4	Zuhaire	2	23	15	21	25	15
7	Ashravy	2	24	22	24	17	21
10	Mishti	2	18	21	25	24	23
2	Raman	3	14	19	15	24	23
5	Zuhaire	3	22	18	19	23	13
8	Ashravy	3	12	25	19	21	23
11	Mishti	3	17	18	20	25	20

In [133]:

```
# Write the statement which will sort the
# marks in English in the DataFrame df
# based on Unit Test 3, in descending order.
dfUT3 =df[df.UT ==3]
```

In [134]:

```
dfUT3.sort_values(by=["Eng","Science"],ascending = False ,axis =0)
```

Out[134]:

	Name	UT	Maths	Science	S.St	Hindi	Eng
8	Ashravy	3	12	25	19	21	23
2	Raman	3	14	19	15	24	23
11	Mishti	3	17	18	20	25	20
5	Zuhaire	3	22	18	19	23	13

```
                    -------GROUP BY Functions-------------------
        Split the data into groups by creating a GROUP BY object from the original Dat
    aFrame
        Apply the required function.
```

In [135]:

```
#Create a GROUP BY Name of the student from
# DataFrame df
```

In [136]:

```
# g1 = df.GROUP BY("Name")
```

```
  Input In [136]
    g1 = df.GROUP BY("Name")
                  ^
SyntaxError: invalid syntax
```

In [137]:

```
df
```

Out[137]:

	Name	UT	Maths	Science	S.St	Hindi	Eng
0	Raman	1	22	21	18	20	21
1	Raman	2	21	20	17	22	24
2	Raman	3	14	19	15	24	23

	Name	UT	Maths	Science	S.St	Hindi	Eng
3	Zuhaire	1	20	17	22	24	19
4	Zuhaire	2	23	15	21	25	15
5	Zuhaire	3	22	18	19	23	13
6	Ashravy	1	23	19	20	15	22
7	Ashravy	2	24	22	24	17	21
8	Ashravy	3	12	25	19	21	23
9	Mishti	1	15	22	25	22	22
10	Mishti	2	18	21	25	24	23
11	Mishti	3	17	18	20	25	20

In [138]:

```
g1=df.groupby('Name')
```

In [139]:

```
g1.first()
```

Out[139]:

	UT	Maths	Science	S.St	Hindi	Eng
Name						
Ashravy	1	23	19	20	15	22
Mishti	1	15	22	25	22	22
Raman	1	22	21	18	20	21
Zuhaire	1	20	17	22	24	19

In [140]:

```
g1.size()
```

Out[140]:

```
Name
Ashravy    3
Mishti     3
Raman      3
Zuhaire    3
dtype: int64
```

In [141]:

```
g1.groups
```

Out[141]:

```
{'Ashravy': [6, 7, 8], 'Mishti': [9, 10, 11], 'Raman': [0, 1, 2], 'Zuhaire': [3, 4, 5]}
```

In [142]:

```
g1.get_group('Raman')
```

Out[142]:

	Name	UT	Maths	Science	S.St	Hindi	Eng
0	Raman	1	22	21	18	20	21
1	Raman	2	21	20	17	22	24
2	Raman	3	14	19	15	24	23

In [143]:

```
df.groupby("Name")["Maths"].mean()
```

Out[143]:

```
Name
Ashravy    19.666667
Mishti     16.666667
Raman      19.000000
Zuhaire    21.666667
Name: Maths, dtype: float64
```

In [144]:

```
df.groupby(["Name","UT"])["Maths"].mean()
```

Out[144]:

```
Name     UT
Ashravy  1     23.0
         2     24.0
         3     12.0
Mishti   1     15.0
         2     18.0
         3     17.0
Raman    1     22.0
         2     21.0
         3     14.0
Zuhaire  1     20.0
         2     23.0
         3     22.0
Name: Maths, dtype: float64
```

In [ ]:

```

```

In [ ]:

```

```

In [145]:

```
#Calculating average marks scored by all students in each subject for each UT
df.groupby(['UT']).aggregate('mean')
```

Out[145]:

	Maths	Science	S.St	Hindi	Eng
UT					
1	20.00	19.75	21.25	20.25	21.00
2	21.50	19.50	21.75	22.00	20.75
3	16.25	20.00	18.25	23.25	19.75

Write the python statements to print the mean, variance, standard deviation and quartile of the marks scored in Mathematics by each student across the UTs

In [146]:

```
df.groupby(by='Name')['Maths'].agg(['mean','var','std','quantile'])
```

Out[146]:

	mean	var	std	quantile
Name				
Ashravy	19.666667	44.333333	6.658328	23.0

	mean	var	std	quantile
Name				
Mishti	16.666667	2.333333	1.527525	17.0
Raman	19.000000	19.000000	4.358899	21.0
Zuhaire	21.666667	2.333333	1.527525	22.0

Altering the Index

In [147]:

```
df
```

Out[147]:

	Name	UT	Maths	Science	S.St	Hindi	Eng
0	Raman	1	22	21	18	20	21
1	Raman	2	21	20	17	22	24
2	Raman	3	14	19	15	24	23
3	Zuhaire	1	20	17	22	24	19
4	Zuhaire	2	23	15	21	25	15
5	Zuhaire	3	22	18	19	23	13
6	Ashravy	1	23	19	20	15	22
7	Ashravy	2	24	22	24	17	21
8	Ashravy	3	12	25	19	21	23
9	Mishti	1	15	22	25	22	22
10	Mishti	2	18	21	25	24	23
11	Mishti	3	17	18	20	25	20

In [148]:

```
dfUT1 = df[df.UT ==1]
```

In [149]:

```
dfUT1
```

Out[149]:

	Name	UT	Maths	Science	S.St	Hindi	Eng
0	Raman	1	22	21	18	20	21
3	Zuhaire	1	20	17	22	24	19
6	Ashravy	1	23	19	20	15	22
9	Mishti	1	15	22	25	22	22

In [150]:

```
dfUT1.reset_index(inplace =True)
```

In [151]:

```
dfUT1
```

Out[151]:

	index	Name	UT	Maths	Science	S.St	Hindi	Eng
0	0	Raman	1	22	21	18	20	21
1	3	Zuhaire	1	20	17	22	24	19
2	6	Ashravy	1	23	19	20	15	22

	index	Name	UT	Maths	Science	S.St	Hindi	Eng
3	9	Mishti	1	15	22	25	22	22

In [152]:

```
dfUT1.drop(columns="index",inplace =True)
```

```
C:\Users\mrpam\AppData\Local\Temp\ipykernel_7452\268329024.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_g
uide/indexing.html#returning-a-view-versus-a-copy
  dfUT1.drop(columns="index",inplace =True)
```

In [153]:

```
dfUT1
```

Out[153]:

	Name	UT	Maths	Science	S.St	Hindi	Eng
0	Raman	1	22	21	18	20	21
1	Zuhaire	1	20	17	22	24	19
2	Ashravy	1	23	19	20	15	22
3	Mishti	1	15	22	25	22	22

In [154]:

```
dfUT1.set_index("Name")
```

Out[154]:

	UT	Maths	Science	S.St	Hindi	Eng
Name						
Raman	1	22	21	18	20	21
Zuhaire	1	20	17	22	24	19
Ashravy	1	23	19	20	15	22
Mishti	1	15	22	25	22	22

In [160]:

```
# dfUT1.reset_index("Name",inplace = True)
```

In [ ]:

```
------Other DataFrame Operations--------
```

Reshaping Data PiVOT

In [156]:

```
import pandas as pd
data={'Store':['S1','S4','S3','S1','S2','S3','S1','S2','S3'], 'Year':[2016,2016,2016,2017
,2017,2017,2018,2018,2018],'Total_sales(Rs)':[12000,330000,420000, 20000,10000,450000,30
000, 11000,89000],'Total_profit(Rs)':[1100,5500,21000,32000,9000,45000,3000, 1900,23000]}
```

In [157]:

```
df = pd.DataFrame(data)
```

In [158]:

```
df
```

Out[158]:

	Store	Year	Total_sales(Rs)	Total_profit(Rs)
0	S1	2016	12000	1100
1	S4	2016	330000	5500
2	S3	2016	420000	21000
3	S1	2017	20000	32000
4	S2	2017	10000	9000
5	S3	2017	450000	45000
6	S1	2018	30000	3000
7	S2	2018	11000	1900
8	S3	2018	89000	23000

In [159]:

```
# 1) What was the total sale of store S1 in all the years?
df.groupby("Store")["Total_sales(Rs)"].sum()
```

Out[159]:

```
Store
S1     62000
S2     21000
S3    959000
S4    330000
Name: Total_sales(Rs), dtype: int64
```

In [ ]:

```

```

In [161]:

```
S1df = df[df.Store =="S1"]
```

In [162]:

```
S1df["Total_sales(Rs)"].sum()
```

Out[162]:

```
62000
```

In [ ]:

```
# Which store had the maximum total sale in all the years?
```

In [163]:

```
S1df = df[df.Store=='S1']
S2df=df[df.Store == 'S2']
S3df = df[df.Store=='S3']
S4df = df[df.Store=='S4']
S1total = S1df['Total_sales(Rs)'].sum()
S2total = S2df['Total_sales(Rs)'].sum()
S3total = S3df['Total_sales(Rs)'].sum()
S4total = S4df['Total_sales(Rs)'].sum()
max(S1total,S2total,S3total,S4total)
```

Out[163]:

```
959000
```

In [164]:

In [164]:

```
df
```

Out[164]:

	Store	Year	Total_sales(Rs)	Total_profit(Rs)
0	S1	2016	12000	1100
1	S4	2016	330000	5500
2	S3	2016	420000	21000
3	S1	2017	20000	32000
4	S2	2017	10000	9000
5	S3	2017	450000	45000
6	S1	2018	30000	3000
7	S2	2018	11000	1900
8	S3	2018	89000	23000

In [ ]:

```
df = pd.DataFrame(marksUT)
display(df)
```

In [165]:

```
marksUT = {'Name':['Raman','Raman','Raman','Raman','Zuhaire','Zuhaire','Zuhaire','Zuhair
e','Ashravy','Ashravy','Ashravy','Ashravy','Mishti','Mishti','Mishti','Mishti'],
'UT':[1,2,3,4,1,2,3,4,1,2,3,4,1,2,3,4],'Maths':[22,21,14,np.NaN,20,23,22,19,23,24,12,15
,15,18,17,14], 'Science':[21,20,19,np.NaN,17,15,18,20,19,22,25,20,22,21,18,20], 'S.St':
[18,17,15,19,22,21,19,17,20,24,19,20,25,25,20,19], 'Hindi':[20,22,24,18,24,25,23,21, 15
,17,21,20,22,24,25,20], 'Eng':[21,24,23,np.NaN,19,15,13,16,22,21,23,17,22,23,20,18]}
```

In [166]:

```
df =pd.DataFrame(marksUT)
```

In [167]:

```
df
```

Out[167]:

	Name	UT	Maths	Science	S.St	Hindi	Eng
0	Raman	1	22.0	21.0	18	20	21.0
1	Raman	2	21.0	20.0	17	22	24.0
2	Raman	3	14.0	19.0	15	24	23.0
3	Raman	4	NaN	NaN	19	18	NaN
4	Zuhaire	1	20.0	17.0	22	24	19.0
5	Zuhaire	2	23.0	15.0	21	25	15.0
6	Zuhaire	3	22.0	18.0	19	23	13.0
7	Zuhaire	4	19.0	20.0	17	21	16.0
8	Ashravy	1	23.0	19.0	20	15	22.0
9	Ashravy	2	24.0	22.0	24	17	21.0
10	Ashravy	3	12.0	25.0	19	21	23.0
11	Ashravy	4	15.0	20.0	20	20	17.0
12	Mishti	1	15.0	22.0	25	22	22.0
13	Mishti	2	18.0	21.0	25	24	23.0
14	Mishti	3	17.0	18.0	20	25	20.0

14	Mishti	3	17.0	18.0	20	25	20.0
	Name	UT	Maths	Science	S.St	Hindi	Eng
15	Mishti	4	14.0	20.0	19	20	18.0

In [168]:

```
df.isnull().sum()
```

Out[168]:

```
Name       0
UT         0
Maths      1
Science    1
S.St       0
Hindi      0
Eng        1
dtype: int64
```

In [169]:

```
# check each attributes
df["Science"].isnull().sum()
```

Out[169]:

```
1
```

In [170]:

```
#  any is used to return to entire data
df.isnull().any().sum()
```

Out[170]:

```
3
```

In [171]:

```
print(df['Science'].isnull().any())
```

```
True
```

In [172]:

```
print(df['Hindi'].isnull().any())
```

```
False
```

In [173]:

```
df.isnull().sum()
```

Out[173]:

```
Name       0
UT         0
Maths      1
Science    1
S.St       0
Hindi      0
Eng        1
dtype: int64
```

In [ ]:

```

```

In [ ]:

```

```

In [174]:

```
# to find total no of sum
df.isnull().sum().sum()
```

Out[174]:

```
3
```

```
                  ----- EDA FOR SIMPLE DATA ------
```

In [175]:

```
# Write a program to find the percentage of marks scored by Raman in hindi
dfRaman =df[df["Name"]=="Raman"]
```

In [ ]:

```

```

In [176]:

```
dfHindi = dfRaman["Hindi"]
row = len(dfHindi)
```

In [177]:

```
print("percentage by rAMAN IN HINDI",(dfRaman["Hindi"].sum()*100)/(25*row),"%")
```

```
percentage by rAMAN IN HINDI 84.0 %
```

In [178]:

```
# Write a python program to find the percentage of marks obtained by Raman in Maths subje
ct.
```

In [179]:

```
dfRaman = df[df["Name"]=="Raman"]
```

In [180]:

```
dfMaths=dfRaman["Maths"]
```

In [181]:

```
row = len(dfMaths)
```

In [183]:

```
print("percentage by rAMAN IN HINDI",(dfRaman["Maths"].sum()*100)/(25*row),"%")
```

```
percentage by rAMAN IN HINDI 57.0 %
```

In [182]:

```
df
```

Out[182]:

	Name	UT	Maths	Science	S.St	Hindi	Eng
0	Raman	1	22.0	21.0	18	20	21.0
1	Raman	2	21.0	20.0	17	22	24.0
2	Raman	3	14.0	19.0	15	24	23.0
3	Raman	4	NaN	NaN	19	18	NaN
4	Zuhaire	1	20.0	17.0	22	24	19.0

	Name	UT	Maths	Science	S.St	Hindi	Eng
5	Zuhaire	2	23.0	15.0	21	25	15.0
6	Zuhaire	3	22.0	18.0	19	23	13.0
7	Zuhaire	4	19.0	20.0	17	21	16.0
8	Ashravy	1	23.0	19.0	20	15	22.0
9	Ashravy	2	24.0	22.0	24	17	21.0
10	Ashravy	3	12.0	25.0	19	21	23.0
11	Ashravy	4	15.0	20.0	20	20	17.0
12	Mishti	1	15.0	22.0	25	22	22.0
13	Mishti	2	18.0	21.0	25	24	23.0
14	Mishti	3	17.0	18.0	20	25	20.0
15	Mishti	4	14.0	20.0	19	20	18.0

```
           -------Dropping Missing Values--------------
```

In [184]:

```
# dropna()
```

In [185]:

```
df1 =df.dropna()
```

In [186]:

```
df1
```

Out[186]:

	Name	UT	Maths	Science	S.St	Hindi	Eng
0	Raman	1	22.0	21.0	18	20	21.0
1	Raman	2	21.0	20.0	17	22	24.0
2	Raman	3	14.0	19.0	15	24	23.0
4	Zuhaire	1	20.0	17.0	22	24	19.0
5	Zuhaire	2	23.0	15.0	21	25	15.0
6	Zuhaire	3	22.0	18.0	19	23	13.0
7	Zuhaire	4	19.0	20.0	17	21	16.0
8	Ashravy	1	23.0	19.0	20	15	22.0
9	Ashravy	2	24.0	22.0	24	17	21.0
10	Ashravy	3	12.0	25.0	19	21	23.0
11	Ashravy	4	15.0	20.0	20	20	17.0
12	Mishti	1	15.0	22.0	25	22	22.0
13	Mishti	2	18.0	21.0	25	24	23.0
14	Mishti	3	17.0	18.0	20	25	20.0
15	Mishti	4	14.0	20.0	19	20	18.0

In [187]:

```
# marks obtained by Raman in all the unit tests
dfRaman = df[df.Name == "Raman"]
```

In [188]:

```
dfRaman.dropna(inplace =True,how = "any")
```

```
C:\Users\mrpam\AppData\Local\Temp\ipykernel_7452\3021977047.py:1: SettingWithCopyWarning:

A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_g
uide/indexing.html#returning-a-view-versus-a-copy
  dfRaman.dropna(inplace =True,how = "any")
```

In [189]:

```
dfMaths = dfRaman["Maths"]
```

In [190]:

```
dfMaths
```

Out[190]:

```
0    22.0
1    21.0
2    14.0
Name: Maths, dtype: float64
```

In [ ]:

```

```

In [191]:

```
row = len(dfMaths)
```

In [192]:

```
print(dfMaths.sum()*100/(25*row),"%")
```

```
76.0 %
```

------------Estimating Missing Values----------------------

In [ ]:

```
# Missing values can be filled by using estimations or approximations
# The fillna(num) function can be used to replace
# missing value(s) by the value specified in num. For
# example, fillna(0) replaces missing value by 0. Similarly
# fillna(1) replaces missing value by 1
```

In [193]:

```
#Marks Scored by Raman in all the subjects across the tests
dfRaman = df.loc[df["Name"] =="Raman"]
```

In [194]:

```
(row,col) = dfRaman.shape
```

In [195]:

```
(row,col)
```

Out[195]:

```
(4, 7)
```

In [196]:

```
dfScience = dfRaman.loc[:,"Science"]
```

In [197]:

```
dfScience
```

Out[197]:

```
0    21.0
1    20.0
2    19.0
3     NaN
Name: Science, dtype: float64
```

In [198]:

```
dfFillZeroScience=dfScience.fillna(0)
```

In [199]:

```
print("percentage of marks by Raman", dfFillZeroScience.sum()*100/(row*25),"%")
```

```
percentage of marks by Raman 60.0 %
```

In [ ]:

```
# df.fillna(method='pad') replaces the missing value by the value before the missing valu
e while
# df.fillna(method='bfill') replaces the missing value by the value after the missing val
ue
```

In [200]:

```
dfEng = dfRaman.loc[:,'Eng']
```

In [201]:

```
dfEng
```

Out[201]:

```
0    21.0
1    24.0
2    23.0
3     NaN
Name: Eng, dtype: float64
```

In [202]:

```
dfFillPadEng = dfEng.fillna(method='pad')
```

In [203]:

```
dfFillPadEng
```

Out[203]:

```
0    21.0
1    24.0
2    23.0
3    23.0
Name: Eng, dtype: float64
```

In [204]:

```
print(dfFillPadEng.sum()*100/(25*row),"%")
```

```
91.0 %
```