In [1]:
```
# VIVEK-CHAUHAN-ADVANCED-DATA-ANALYTICS-PANDAS-2
```

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

In [3]:
```
df1 = pd.DataFrame(data = [[11,12,13],[14,15,16],[17,18,19]])
print(df1)
```

```
    0   1   2
0  11  12  13
1  14  15  16
2  17  18  19
```

In [5]:
```
df2 = pd.DataFrame(data = [[11,12,13],[14,15,16],[17,18,19]])
print(df2)
```

```
    0   1   2
0  11  12  13
1  14  15  16
2  17  18  19
```

In [19]:
```
# addition binary operation using various methods

print(df1 + df2) #vector operation will be perform
```

```
    0   1   2
0  22  24  26
1  28  30  32
2  34  36  38
```

In [21]:
```
# addition binary operation using various methods

print(df1.add(df2)) #vector operation will be perform
```

```
    0   1   2
0  22  24  26
1  28  30  32
2  34  36  38
```

In [25]:
```
# radd means the reverse addition will be perform like dataframe two will be add in

print(df1.radd(df2)) # output will be changed if there is nagative value in the dat
```

```
    0   1   2
0  22  24  26
1  28  30  32
2  34  36  38
```

In [32]:
```
# substraction binary operation using various methods

print(df2 - df1)
```

```
   0  1  2
0  0  0  0
1  0  0  0
2  0  0  0
```

In [34]:
```
# substraction binary operation using various methods

print(df2.sub(df1))
```

```
   0  1  2
0  0  0  0
1  0  0  0
2  0  0  0
```

In [38]:
```
# substraction binary operation using various methods

print(df1.sub(df2)) # rsub means reverse substraction will be perform in this meani
```

```
   0  1  2
0  0  0  0
1  0  0  0
2  0  0  0
```

In [42]:
```
# multiplication binary operation using various methods

print(df1 * df2)
```

```
     0    1    2
0  121  144  169
1  196  225  256
2  289  324  361
```

In [44]:
```
# multiplication binary operation using various methods

print(df1.mul(df2))
```

```
     0    1    2
0  121  144  169
1  196  225  256
2  289  324  361
```

In [48]:
```
# division binary operation using various methods

print(df1 / df2)
```

```
     0    1    2
0  1.0  1.0  1.0
1  1.0  1.0  1.0
2  1.0  1.0  1.0
```

In [50]:
```
# division binary operation using various methods

print(df1.div(df2))
```

```
     0    1    2
0  1.0  1.0  1.0
1  1.0  1.0  1.0
2  1.0  1.0  1.0
```

In [58]:
```
#statistical functions min and max is here.

school1 = pd.Series(data = [10,20,30],index = ["science","commerce","arts"])
school2 = pd.Series(data = [40,50,60],index = ["science","commerce","arts"])
```

```
total_students = school1 + school2

dataframe = pd.DataFrame(total_students)
print(dataframe)
```

```
           0
science   50
commerce  70
arts      90
```

In [64]:
```
#statistical functions min and max is here.

dataframe.min() # min value will be print
```

Out[64]:
```
0    50
dtype: int64
```

In [68]:
```
#statistical functions min and max is here.

dataframe.max() # max value will be print
```

Out[68]:
```
0    90
dtype: int64
```

In [13]:
```
# min-max values using axis.

print(df1.max(axis = 1)) # it will return each and every row wise column max number
```

```
0    13
1    16
2    19
dtype: int64
```

In [3]:
```
ages = {
    "age" : [22,11,25,26,27,22,25],
    "salary" : [25000,17000,26000,17000,25000,17000,11000]
}

dataframe = pd.DataFrame(ages)
print(dataframe)
```

```
   age  salary
0   22   25000
1   11   17000
2   25   26000
3   26   17000
4   27   25000
5   22   17000
6   25   11000
```

In [43]:
```
# mode() function.

print(dataframe.mode(axis = 0))
```

```
   age   salary
0   22  17000.0
1   25      NaN
```

In [45]:
```
# median() function.

print(dataframe.median(axis = 0))
```

```
age          25.0
salary    17000.0
dtype: float64
```

In [47]:
```
# mean() function

print(dataframe.mean(axis = 0))
```

```
age          22.571429
salary    19714.285714
dtype: float64
```

In [51]:
```
# count() and sum() functions

print(dataframe.count(axis = 1))
```

```
0    2
1    2
2    2
3    2
4    2
5    2
6    2
dtype: int64
```

In [55]:
```
# count() and sum() functions

print(dataframe.count(axis = 0))
```

```
age       7
salary    7
dtype: int64
```

In [59]:
```
# count() and sum() functions

print(dataframe.sum(axis = 0))
```

```
age          158
salary    138000
dtype: int64
```

In [61]:
```
# count() and sum() functions

print(dataframe.sum(axis = 1))
```

```
0    25022
1    17011
2    26025
3    17026
4    25027
5    17022
6    11025
dtype: int64
```

In [73]:
```
# quantile() function

print(dataframe.quantile(q = 0.25)) # here q is the range of default quantile ratio
```

```
age          22.0
salary    17000.0
Name: 0.25, dtype: float64
```

In [75]:
```
# std() function is standard deviation function

print(dataframe.std())
```

```
age          5.442338
salary    5677.860093
dtype: float64
```

In [79]:
```
# std() function is standard deviation function

print(dataframe.std(axis = 1))
```

```
0    17662.113180
1    12013.037106
2    18367.098641
3    12002.430504
4    17658.577647
5    12005.258931
6     7760.496924
dtype: float64
```

In [81]:
```
# var() function variance

print(dataframe.var())
```

```
age       2.961905e+01
salary    3.223810e+07
dtype: float64
```

In [87]:
```
# describe() function

print(dataframe.describe()) # it will calculate all the statistics in this function
```

```
              age        salary
count    7.000000      7.000000
mean    22.571429  19714.285714
std      5.442338   5677.860093
min     11.000000  11000.000000
25%     22.000000  17000.000000
50%     25.000000  17000.000000
75%     25.500000  25000.000000
max     27.000000  26000.000000
```

In [89]:
```
# info() function

print(dataframe.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7 entries, 0 to 6
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   age     7 non-null      int64
 1   salary  7 non-null      int64
dtypes: int64(2)
memory usage: 244.0 bytes
None
```

In [97]:
```
# head and tail functions

print(dataframe.head(2))
```

```
   age  salary
0   22   25000
1   11   17000
```

In [99]:
```
# head and tail functions

print(dataframe.tail(2))
```

```
   age  salary
5   22   17000
6   25   11000
```

In [101…
```
# cumulative calculation functions

print(dataframe.cumsum())
```

```
   age  salary
0   22   25000
1   33   42000
2   58   68000
3   84   85000
4  111  110000
5  133  127000
6  158  138000
```

In [111…
```
# cumulative calculation functions

print(dataframe.cumsum(axis = 'columns'))
```

```
   age  salary
0   22   25022
1   11   17011
2   25   26025
3   26   17026
4   27   25027
5   22   17022
6   25   11025
```

In [113…
```
# cumprod,cummax,cummin functions

print(dataframe.cumprod())
```

```
         age               salary
0         22                25000
1        242            425000000
2       6050       11050000000000
3     157300   187850000000000000
4    4247100 -7669738795935662080
5   93436200 -3972417927144538112
6 2335905000  3739512028008546304
```

In [115…
```
# cumprod,cummax,cummin functions

print(dataframe.cummax())
```

```
   age  salary
0   22   25000
1   22   25000
2   25   26000
3   26   26000
4   27   26000
5   27   26000
6   27   26000
```

In [117…
```
# cumprod,cummax,cummin functions

print(dataframe.cummin())
```

```
   age  salary
0   22   25000
1   11   17000
2   11   17000
3   11   17000
4   11   17000
5   11   17000
6   11   11000
```

In [235…
```
# dataframe some missing data

students = {
    "student_1": {"name": "Alice Johnson", "age": 20, "grade": "A", "email": None},
    "student_2": {"name": "Bob Smith", "age": 21, "grade": "B", "email": "bob.smith
    "student_3": {"name": None, "age": 22, "grade": "A", "email": "carla.jones@emai
    "student_4": {"name": "David Lee", "age": None, "grade": "C", "email": "david.l
    "student_5": {"name": "Eva Green", "age": 19, "grade": None, "email": "eva.gree
    "student_6": {"name": "Frank Miller", "age": 23, "grade": "B", "email": "frank.
```

```
}

df = pd.DataFrame(students,index = ['name', 'age', 'grade', 'email'])
print(df)
```

```
          student_1            student_2              student_3  \
name   Alice Johnson            Bob Smith                   None
age               20                   21                     22
grade              A                    B                      A
email           None  bob.smith@email.com  carla.jones@email.com

                student_4            student_5              student_6
name            David Lee            Eva Green           Frank Miller
age                  None                   19                     23
grade                   C                 None                      B
email  david.lee@email.com  eva.green@email.com  frank.miller@email.com
```

In [127…
```
# detecting/filtering missing data

print(df.isnull()) # it will return the boolean values
```

```
       student_1  student_2  student_3  student_4  student_5  student_6
name       False      False       True      False      False      False
age        False      False      False       True      False      False
grade      False      False      False      False       True      False
email       True      False      False      False      False      False
```

In [197…
```
# dropping the missing data

df.dropna()
print(df)
```

```
          student_1            student_2              student_3  \
name   Alice Johnson            Bob Smith                   None
age               20                   21                     22
grade              A                    B                      A
email           None  bob.smith@email.com  carla.jones@email.com

                student_4            student_5              student_6
name            David Lee            Eva Green           Frank Miller
age                  None                   19                     23
grade                   C                 None                      B
email  david.lee@email.com  eva.green@email.com  frank.miller@email.com
```

In [199…
```
# dropping the missing data

dropna = df.dropna(axis = 1) # it will drop the whole column whose have nan,na,none
print(dropna)
```

```
                student_2              student_6
name            Bob Smith           Frank Miller
age                    21                     23
grade                   B                      B
email  bob.smith@email.com  frank.miller@email.com
```

In [201…
```
# filling missing values
```

```
print(df.fillna(0))
```

```
          student_1            student_2              student_3  \
name   Alice Johnson            Bob Smith                      0
age              20                   21                     22
grade             A                    B                      A
email             0  bob.smith@email.com  carla.jones@email.com

                student_4            student_5              student_6
name            David Lee            Eva Green           Frank Miller
age                     0                   19                     23
grade                   C                    0                      B
email  david.lee@email.com  eva.green@email.com  frank.miller@email.com
```

In [203…
```
# filling missing values

print(df.fillna("gmail.com"))
```

```
          student_1            student_2              student_3  \
name   Alice Johnson            Bob Smith              gmail.com
age              20                   21                     22
grade             A                    B                      A
email     gmail.com  bob.smith@email.com  carla.jones@email.com

                student_4            student_5              student_6
name            David Lee            Eva Green           Frank Miller
age          gmail.com                   19                     23
grade                   C           gmail.com                      B
email  david.lee@email.com  eva.green@email.com  frank.miller@email.com
```

In [3]:
```
# group by function

classes = {
    "classes" : [28,36,41,51,20],
    "country" : ['usa','uk','japan','brazil','usa'],
    "quarter" : ["tahira","jacob","tahira","jacob","tahira"],
}

df2 = pd.DataFrame(classes)
print(df2)
```

```
   classes country quarter
0       28     usa  tahira
1       36      uk   jacob
2       41   japan  tahira
3       51  brazil   jacob
4       20     usa  tahira
```

In [285…
```
# group by function

gdf = df2.groupby('quarter') # it will groupby on the basis of the classes but not
print(gdf.groups) # lists the groups created. in the list is the index numbers
```

```
{'jacob': [1, 3], 'tahira': [0, 2, 4]}
```

In [291…
```
# to print the result of the group by function is .get_group(criteria)
```

```
print(gdf.get_group("jacob"))
```

```
   classes country quarter
1       36      uk   jacob
3       51  brazil   jacob
```

In [295…
```
print(gdf.size()) # size of the groups created
```

```
quarter
jacob     2
tahira    3
dtype: int64
```

In [301…
```
print(gdf.count()) # count the groups values.
```

```
         classes  country
quarter
jacob          2        2
tahira         3        3
```

In [17]:
```
print(df2["quarter"].head()) # return the head values of given column from the give
```

```
0    tahira
1     jacob
2    tahira
3     jacob
4    tahira
Name: quarter, dtype: object
```

In [26]:
```
print(df2["quarter"].sum()) # it will return the sum values if value is not integer
```

```
tahirajacobtahirajacobtahira
```

In [28]:
```
print(df2["classes"].sum()) # it will return the sum values if value is not integer
```

```
176
```

In [30]:
```
print(df2["classes"]) # it will return the mean values of given specified column
```

```
0    28
1    36
2    41
3    51
4    20
Name: classes, dtype: int64
```

In [31]:
```
# q7

print(df2.sum())
print(df2["classes"].mean())
print(df2.sum(axis = 0))
```

```
classes                             176
country             usaukjapanbrazilusa
quarter    tahirajacobtahirajacobtahira
dtype: object
35.2
classes                             176
country             usaukjapanbrazilusa
quarter    tahirajacobtahirajacobtahira
dtype: object
```

In [ ]: