```
import pandas as pd
df=pd.DataFrame([[1,2,3],
                 [4,5,6],
                 [7,8,9]],columns=['x','y','z'],index=['a','b','c'])
```

all code sections are connected ,the data frame df connected above is can be used below,kindof storesthe statemeans,if wenot runabove one tghen there isno change below after change in above

```
df
# df.tail(1)
# df.head(1)
```

```
   x  y  z
a  1  2  3
b  4  5  6
c  7  8  9
```

df.head(n) is used to view the head of the table i.e, n starting rows,where as df.tail(n) to see tail of the table i.e, last n rows

```
# df.columns
df.index
```

```
Index(['a', 'b', 'c'], dtype='object')
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 3 entries, a to c
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   x       3 non-null      int64
 1   y       3 non-null      int64
 2   z       3 non-null      int64
dtypes: int64(3)
memory usage: 96.0+ bytes
```

```
df.describe()
```

```
         x    y    z
count  3.0  3.0  3.0
mean   4.0  5.0  6.0
std    3.0  3.0  3.0
min    1.0  2.0  3.0
25%    2.5  3.5  4.5
50%    4.0  5.0  6.0
75%    5.5  6.5  7.5
max    7.0  8.0  9.0
```

```
df.nunique()

x    3
y    3
z    3
dtype: int64

df['y']

a    2
b    5
c    8
Name: y, dtype: int64

df.shape

(3, 3)
```

this above are used for information about dataframe table,and indexing works column wise

loading in DAtaFrames From files

```
crop=pd.read_csv('Crop_recommendation.csv')
crop

        N   P   K  temperature   humidity        ph    rainfall
label
0      90  42  43    20.879744  82.002744  6.502985  202.935536
rice
1      85  58  41    21.770462  80.319644  7.038096  226.655537
rice
2      60  55  44    23.004459  82.320763  7.840207  263.964248
rice
3      74  35  40    26.491096  80.158363  6.980401  242.864034
rice
4      78  42  42    20.130175  81.604873  7.628473  262.717340
rice
...   ...  ..  ..          ...        ...       ...         ...     ..
.
2195  107  34  32    26.774637  66.413269  6.780064  177.774507
coffee
2196   99  15  27    27.417112  56.636362  6.086922  127.924610
coffee
2197  118  33  30    24.131797  67.225123  6.362608  173.322839
coffee
2198  117  32  34    26.272418  52.127394  6.758793  127.175293
coffee
2199  104  18  30    23.603016  60.396475  6.779833  140.937041
coffee

[2200 rows x 8 columns]
```

Accesing Data with Pandas

```
crop.describe()
```

```
                 N            P            K  temperature     humidity
\
count  2200.000000  2200.000000  2200.000000  2200.000000  2200.000000

mean     50.551818    53.362727    48.149091    25.616244    71.481779

std      36.917334    32.985883    50.647931     5.063749    22.263812

min       0.000000     5.000000     5.000000     8.825675    14.258040

25%      21.000000    28.000000    20.000000    22.769375    60.261953

50%      37.000000    51.000000    32.000000    25.598693    80.473146

75%      84.250000    68.000000    49.000000    28.561654    89.948771

max     140.000000   145.000000   205.000000    43.675493    99.981876


                ph     rainfall
count  2200.000000  2200.000000
mean      6.469480   103.463655
std       0.773938    54.958389
min       3.504752    20.211267
25%       5.971693    64.551686
50%       6.425045    94.867624
75%       6.923643   124.267508
max       9.935091   298.560117
```

```
crop.sample(10,random_state=1)
```

```
        N    P    K  temperature   humidity        ph    rainfall
label
1276   25  129  195    17.986678  81.177121  5.777271   72.371277
grapes
1446  106   20   51    29.730197  90.970157  6.342573   20.490356
muskmelon
335    33   59   22    22.642369  21.593961  5.947000  122.388601
kidneybeans
1458   89    9   47    29.471563  90.770696  6.668383   28.752261
muskmelon
2038   62   49   37    24.217446  82.852840  7.479248  166.136589
jute
1314  104   17   46    25.713143  80.229728  6.190016   43.089618
watermelon
389     9   69   20    19.306073  23.963628  5.591561  129.344933
kidneybeans
```

```
1639    1   17    6    10.786898  91.384119  6.819827  117.529345
orange
2004   96   41   40    23.584193  72.004608  6.090060  190.424216
jute
403    27   57   24    27.335349  43.357960  6.091863  142.330368
pigeonpeas
```

loc allows us to filter by rows or columns,used as crop.loc[rows,columns]

```
crop.loc[10]
```

```
N                     91
P                     53
K                     40
temperature    26.527235
humidity       81.417538
ph              5.386168
rainfall       264.61487
label               rice
Name: 10, dtype: object
```

if you want multiple rows ,put it as a array or slice(inclusive end and start)

```
# crop.loc[[0,1,2]]
crop.loc[0:20:2]
```

```
     N   P   K  temperature   humidity        ph    rainfall label
0   90  42  43    20.879744  82.002744  6.502985  202.935536  rice
2   60  55  44    23.004459  82.320763  7.840207  263.964248  rice
4   78  42  42    20.130175  81.604873  7.628473  262.717340  rice
6   69  55  38    22.708838  82.639414  5.700806  271.324860  rice
8   89  54  38    24.515881  83.535216  6.685346  230.446236  rice
10  91  53  40    26.527235  81.417538  5.386168  264.614870  rice
12  78  58  44    26.800796  80.886848  5.108682  284.436457  rice
14  94  50  37    25.665852  80.663850  6.948020  209.586971  rice
16  85  38  41    21.587118  82.788371  6.249051  276.655246  rice
18  77  38  36    21.865252  80.192301  5.953933  224.555017  rice
20  89  45  36    21.325042  80.474764  6.442475  185.497473  rice
```

for column use name of the column and slicing also works here

```
crop.loc[0:5,"K":"ph":1]
```

```
    K  temperature   humidity        ph
0  43    20.879744  82.002744  6.502985
1  41    21.770462  80.319644  7.038096
2  44    23.004459  82.320763  7.840207
3  40    26.491096  80.158363  6.980401
```

```
4  42    20.130175  81.604873  7.628473
5  42    23.058049  83.370118  7.073454
```

crop.iloc[rws,cloumns] uses index for both rows and columns only

```
crop.iloc[0:5,0:4]
```

```
    N   P   K  temperature
0  90  42  43    20.879744
1  85  58  41    21.770462
2  60  55  44    23.004459
3  74  35  40    26.491096
4  78  42  42    20.130175
```

Modifying values by assining operator

```
crop.loc[0:5,"N"]=99
crop.head(6)
```

```
    N   P   K  temperature   humidity        ph    rainfall label
0  99  42  43    20.879744  82.002744  6.502985  202.935536  rice
1  99  58  41    21.770462  80.319644  7.038096  226.655537  rice
2  99  55  44    23.004459  82.320763  7.840207  263.964248  rice
3  99  35  40    26.491096  80.158363  6.980401  242.864034  rice
4  99  42  42    20.130175  81.604873  7.628473  262.717340  rice
5  99  37  42    23.058049  83.370118  7.073454  251.055000  rice
```

optimized way to get specfic value is using at or iat,at is similar as loc and iat is similiar as iat

```
crop.at[1,"N"]
```

```
np.int64(99)
```

```
crop.iat[1,1]
```

```
np.int64(58)
```

Another way to access data ,only colum acces is allow

```
crop["N"]
```

```
0        99
1        99
2        99
3        99
4        99
       ...
2195    107
2196     99
```

```
2197    118
2198    117
2199    104
Name: N, Length: 2200, dtype: int64
```

sorting of data,sort_values sort with respect to specific column label,for multiple column use array and ,for descending use ascending=False or ascending=0 or can be for 2 columns ascending=[0,1]

```
crop.sort_values(["N","ph"],ascending=[1,1])
```

```
        N    P    K  temperature   humidity        ph    rainfall
label
517     0   55   25    28.174894  43.667230  4.524172   45.781728
mothbeans
477     0   70   21    36.300497  56.030213  4.672437  101.607399
pigeonpeas
1184    0   36   26    34.130722  51.257862  5.101206   96.388080
mango
1588    0  145  205    21.225034  90.098778  5.520783  113.976046
apple
1874    0   26   31    25.070725  95.021568  5.547933  192.903631
coconut
...   ...  ...  ...          ...        ...       ...         ...
...
1974  136   36   24    22.744470  80.411985  7.597820   90.073266
cotton
1928  139   35   15    25.248679  83.463015  5.898293   86.555178
cotton
1978  140   45   15    25.530827  80.046628  5.801048   99.395572
cotton
1950  140   40   17    22.727672  77.075981  6.006086   77.551763
cotton
1912  140   38   15    24.147295  75.882986  6.021440   69.915635
cotton

[2200 rows x 8 columns]
```

ANother way ,using for loop

```
for index,rows in crop.iterrows():
    print(index)
    print(rows["N"])
    print("\n\n\n")
```

```
0
99
```

1
99

2
99

3
99

4
99

5
99

6
69

7
94

8
89

9
68

10
91

11
90

12
78

13
93

14
94

15
60

16
85

17
91

18

```
77



19
88



20
89



21
76



22
67



23
83



24
98



25
66



26
97
```

2180
80

2181
101

2182
103

2183
93

2184
104

2185
116

2186
107

2187
101

2188

```
107



2189
99



2190
103



2191
118



2192
106



2193
116



2194
97



2195
107



2196
99
```

```
2197
118



2198
117



2199
104
```

Filtering Data

```
crop.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2200 entries, 0 to 2199
Data columns (total 8 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   N            2200 non-null   int64  
 1   P            2200 non-null   int64  
 2   K            2200 non-null   int64  
 3   temperature  2200 non-null   float64
 4   humidity     2200 non-null   float64
 5   ph           2200 non-null   float64
 6   rainfall     2200 non-null   float64
 7   label        2200 non-null   object 
dtypes: float64(4), int64(3), object(1)
memory usage: 137.6+ KB
```

filtering data using loc ,with multiple conditions data specifification

```
crop.loc[(crop['rainfall']>100) & (crop['label']=='apple'),
['temperature','humidity','rainfall']].head(10)
```

	temperature	humidity	rainfall
1500	22.750888	90.694892	110.431786
1501	23.849401	94.348150	114.051249
1502	22.608010	94.589006	116.039659
1503	21.186674	91.134357	122.233323
1504	23.410447	91.699133	116.077793

```
1505    22.860066  93.128599  117.729673
1506    22.484030  93.408192  105.547363
1507    22.027754  92.961295  121.134918
1508    21.911913  91.687481  117.076128
1509    23.710591  93.273924  112.667659
```

another method,column can be obtained using nested indexing aur chain indexing

```
crop[crop['N']>90]['N']
```

```
0        99
1        99
2        99
3        99
4        99
       ...
2195    107
2196     99
2197    118
2198    117
2199    104
Name: N, Length: 458, dtype: int64
```

string filtering of data

```
crop[crop['label'].str.contains('co|app',case=False)]
```

```
         N    P    K  temperature   humidity        ph    rainfall
label
1500   24  128  196    22.750888  90.694892  5.521467  110.431786
apple
1501    7  144  197    23.849401  94.348150  6.133221  114.051249
apple
1502   14  128  205    22.608010  94.589006  6.226290  116.039659
apple
1503    8  120  201    21.186674  91.134357  6.321152  122.233323
apple
1504   20  129  201    23.410447  91.699133  5.587906  116.077793
apple
...   ...  ...  ...          ...        ...       ...         ...
...
2195  107   34   32    26.774637  66.413269  6.780064  177.774507
coffee
2196   99   15   27    27.417112  56.636362  6.086922  127.924610
coffee
2197  118   33   30    24.131797  67.225123  6.362608  173.322839
coffee
2198  117   32   34    26.272418  52.127394  6.758793  127.175293
coffee
```

```
2199  104   18   30    23.603016  60.396475  6.779833  140.937041
coffee

[400 rows x 8 columns]
```

using query function

```
crop.query('label == "cotton" or label == "coconut"')
```

```
        N   P   K  temperature   humidity        ph    rainfall
label
1800   18  30  29    26.762749  92.860569  6.420019  224.590366
coconut
1801   37  23  28    25.612944  94.313884  5.740055  224.320676
coconut
1802   13  28  33    28.130115  95.648076  5.686973  151.076190
coconut
1803    2  21  35    25.028872  91.537209  6.293662  179.824894
coconut
1804   10  18  35    27.797977  99.645730  6.381975  181.694228
coconut
...   ...  ..  ..          ...        ...       ...         ...       .
..
1995  113  38  20    22.107190  78.583201  6.364730   74.941366
cotton
1996  102  53  21    23.038140  76.110215  6.913679   91.496975
cotton
1997  110  39  18    24.547953  75.397527  7.766260   63.880799
cotton
1998  107  58  15    23.738680  75.775038  7.556064   76.636692
cotton
1999  120  60  15    22.318719  83.861300  7.288377   65.357470
cotton

[200 rows x 8 columns]
```

ADDING /REMOVING Columns

```
product=pd.read_csv('ProductBasedData - Sheet1.csv')
product.head()
```

```
    Name                Email Address Product Name  Quantity
0  Sagar              sagar@gmail.com      Shampoo         3
1  Sagar              sagar@gmail.com      Shampoo         3
2  Sagar  pushkarkhattri111@gmail.com        Razor         3
3  sagar  pushkarkhattri111@gmail.com         ssss        22
```

Adding columns, you can add directely through indexing ,by give new column name as index label and assign value by assignment operator which is assigned to whole column,each row

```
product['Confirmation']='true'
product.head()
```

```
    Name                Email Address Product Name  Quantity
Confirmation
0  Sagar              sagar@gmail.com      Shampoo         3
true
1  Sagar              sagar@gmail.com      Shampoo         3
true
2  Sagar  pushkarkhattri111@gmail.com        Razor         3
true
3  sagar  pushkarkhattri111@gmail.com         ssss        22
true
```

dataset is also referenced here to a single dataset like array

```
import numpy as np
product['validity']=np.where(product['Product
Name']=='Shampoo','yes','No')
product.head()
```

```
    Name                Email Address Product Name  Quantity
Confirmation  \
0  Sagar              sagar@gmail.com      Shampoo         3
true
1  Sagar              sagar@gmail.com      Shampoo         3
true
2  Sagar  pushkarkhattri111@gmail.com        Razor         3
true
3  sagar  pushkarkhattri111@gmail.com         ssss        22
true

  validity
0      yes
1      yes
2       No
3       No
```

to drop a column in a data set

```
product.drop(columns=['Name'])
```

```
                  Email Address Product Name  Quantity Confirmation
validity
0              sagar@gmail.com      Shampoo         3         true
yes
1              sagar@gmail.com      Shampoo         3         true
yes
2  pushkarkhattri111@gmail.com        Razor         3         true
```

```
No
3  pushkarkhattri111@gmail.com         ssss        22         true
No
```

more specific columns ,can be also used to drop out this column

```
product=product[['Email Address','Product Name','Quantity']]
product['num']=3
product.head()
```

	Email Address	Product Name	Quantity	num
0	sagar@gmail.com	Shampoo	3	3
1	sagar@gmail.com	Shampoo	3	3
2	pushkarkhattri111@gmail.com	Razor	3	3
3	pushkarkhattri111@gmail.com	ssss	22	3

Operation btw columns

```
product['rev']=product['Quantity']*product['num']
product.head()
```

	Email Address	Product Name	Quantity	num	rev
0	sagar@gmail.com	Shampoo	3	3	9
1	sagar@gmail.com	Shampoo	3	3	9
2	pushkarkhattri111@gmail.com	Razor	3	3	9
3	pushkarkhattri111@gmail.com	ssss	22	3	66

Rename of column,but you have to store this also

```
product=product.rename(columns={'num':'val'})
product.head()
```

	Email Address	Product Name	Quantity	val	rev
0	sagar@gmail.com	Shampoo	3	3	9
1	sagar@gmail.com	Shampoo	3	3	9
2	pushkarkhattri111@gmail.com	Razor	3	3	9
3	pushkarkhattri111@gmail.com	ssss	22	3	66

Extracting specific stuff from dataset,we need to use str to use string methods

```
product_new=product.copy()

product_new['code']=product_new['Product Name'].str.slice(0,3)
product_new.head()
```

	Email Address	Product Name	Quantity	val	rev	code
0	sagar@gmail.com	Shampoo	3	3	9	Sha
1	sagar@gmail.com	Shampoo	3	3	9	Sha

```
2  pushkarkhattri111@gmail.com        Razor         3    3    9  Raz
3  pushkarkhattri111@gmail.com         ssss        22    3   66  sss
```

converting column type

```
product_new['date']=pd.to_datetime(product_new['val'])

product_new.head()
```

```
                  Email Address Product Name  Quantity  val  rev
code  \
0              sagar@gmail.com      Shampoo         3    3    9  Sha

1              sagar@gmail.com      Shampoo         3    3    9  Sha

2  pushkarkhattri111@gmail.com        Razor         3    3    9  Raz

3  pushkarkhattri111@gmail.com         ssss        22    3   66  sss


                           date
0 1970-01-01 00:00:00.000000003
1 1970-01-01 00:00:00.000000003
2 1970-01-01 00:00:00.000000003
3 1970-01-01 00:00:00.000000003
```

```
product_new['yr']=product_new['date'].dt.year
product_new.head()
```

```
                  Email Address Product Name  Quantity  val  rev
code  \
0              sagar@gmail.com      Shampoo         3    3    9  Sha

1              sagar@gmail.com      Shampoo         3    3    9  Sha

2  pushkarkhattri111@gmail.com        Razor         3    3    9  Raz

3  pushkarkhattri111@gmail.com         ssss        22    3   66  sss


                           date    yr
0 1970-01-01 00:00:00.000000003  1970
1 1970-01-01 00:00:00.000000003  1970
2 1970-01-01 00:00:00.000000003  1970
3 1970-01-01 00:00:00.000000003  1970
```

conversions

df to csv,excel,parquet

```
df.head()
```

```
   x  y  z
a  1  2  3
b  4  5  6
c  7  8  9
```

```
df.to_csv('data/dam.csv',index=False)
data=pd.read_csv('data/dam.csv')
```

Now using custom functions and lambda functions

```
data['k']=data['x'].apply(lambda x: x*2 if x<3 else x**2)
data.head()
```

```
   x  y  z   k
0  1  2  3   2
1  4  5  6  16
2  7  8  9  49
```

only used for row,only row is passed

```
def mod(row):
  sum=0
  for i in row:
    sum=sum+i
  return sum

data['colsum']=data.apply(mod,axis=1)

data.head()
```

```
   x  y  z   k  colsum
0  1  2  3   2       8
1  4  5  6  16      31
2  7  8  9  49      73
```

Merging & Concatenating DATA

```
bios=pd.read_csv('data/bios.csv')
noc=pd.read_csv('data/noc_regions.csv')
bios.head()
```

```
   athlete_id                    name   born_date    born_city  \
0           1  Jean-François Blanchy  1886-12-12     Bordeaux
1           2         Arnaud Boetsch  1969-04-01       Meulan
2           3           Jean Borotra  1898-08-13     Biarritz
3           4        Jacques Brugnon  1895-05-11  Paris VIIIe
4           5           Albert Canet  1878-04-17   Wandsworth
```

```
            born_region born_country     NOC  height_cm  weight_kg
died_date
0              Gironde          FRA  France        NaN        NaN
1960-10-02
1             Yvelines          FRA  France      183.0       76.0
NaN
2  Pyrénées-Atlantiques          FRA  France      183.0       76.0
1994-07-17
3                 Paris          FRA  France      168.0       64.0
1978-03-20
4               England          GBR  France        NaN        NaN
1930-07-25

noc.head()

   NOC       region                   notes
0  AFG  Afghanistan                     NaN
1  AHO      Curacao  Netherlands Antilles
2  ALB      Albania                     NaN
3  ALG      Algeria                     NaN
4  AND      Andorra                     NaN
```

it can imagine as vein diagram ,as left is left circle and right is right circle ,now we do setoperation,

```
bios_new=pd.merge(bios,noc,left_on='born_country',right_on='NOC',how='
inner')
```

the merge() function is used to combine two DataFrames, similar to SQL joins. The two DataFrames are passed as the left and right arguments. If both DataFrames share the same column name for joining, you can use the on parameter. However, if the column names differ, you should use left_on and right_on to specify the respective columns from each DataFrame. The how parameter defines the type of join: 'inner' keeps only matching rows, 'left' keeps all rows from the left DataFrame and matches from the right, 'right' keeps all rows from the right and matches from the left, and 'outer' keeps all rows from both, filling missing values with NaN. When both DataFrames have columns with the same name (other than the key), the suffixes parameter helps differentiate them. You can also merge using indices by setting left_index=True and right_index=True. Overall, merge() provides a powerful and flexible way to join DataFrames, while join() is a simpler method for index-based joins and concat() is mainly for stacking DataFrames without join logic.

```
bios_new.head()

   athlete_id                    name   born_date    born_city  \
0           1  Jean-François Blanchy  1886-12-12     Bordeaux
1           2         Arnaud Boetsch  1969-04-01       Meulan
2           3           Jean Borotra  1898-08-13     Biarritz
3           4        Jacques Brugnon  1895-05-11  Paris VIIIe
4           5           Albert Canet  1878-04-17   Wandsworth
```

```
            born_region born_country   NOC_x  height_cm  weight_kg  \
0              Gironde          FRA  France        NaN        NaN
1             Yvelines          FRA  France      183.0       76.0
2  Pyrénées-Atlantiques          FRA  France      183.0       76.0
3                 Paris          FRA  France      168.0       64.0
4               England          GBR  France        NaN        NaN

    died_date NOC_y  region notes
0  1960-10-02   FRA  France   NaN
1         NaN   FRA  France   NaN
2  1994-07-17   FRA  France   NaN
3  1978-03-20   FRA  France   NaN
4  1930-07-25   GBR      UK   NaN
```

we here copy all born country usa separate

```
usa=bios_new[bios_new['born_country']=='USA'].copy()
gbr=bios_new[bios_new['born_country']=='GBR'].copy()

usa
```

```
        athlete_id               name   born_date         born_city  \
48              55      Monique Javer  1967-07-22        Burlingame
697            964   Xóchitl Escobedo  1968-09-17       West Covina
698            965  Angélica Gavaldón  1973-10-03         El Centro
904           1238     Bert Schneider  1897-07-01         Cleveland
968           1352         Laura Berg  1975-01-06  Santa Fe Springs
...            ...                ...         ...               ...
110599      149168     Kristen Santos  1994-11-02         Fairfield
110600      149169   Corinne Stoddard  2001-08-15           Seattle
110606      149180      Anna Hoffmann  2000-03-28           Madison
110609      149183     Alix Wilkinson  2000-08-02     Mammoth Lakes
110620      149195  Justin Abdelkader  1987-02-25          Muskegon

        born_region born_country          NOC_x  height_cm  weight_kg
\
48       California          USA  Great Britain      177.0       64.0

697      California          USA         Mexico      170.0       60.0

698      California          USA         Mexico      160.0       54.0

904            Ohio          USA         Canada        NaN        NaN

968      California          USA  United States      168.0       61.0

...             ...          ...            ...        ...        ...

110599  Connecticut          USA  United States        NaN        NaN
```

```
110600   Washington          USA  United States        NaN        NaN

110606    Wisconsin          USA  United States        NaN        NaN

110609   California          USA  United States        NaN        NaN

110620     Michigan          USA  United States      187.0       97.0


         died_date NOC_y region notes
48             NaN   USA    USA   NaN
697            NaN   USA    USA   NaN
698            NaN   USA    USA   NaN
904     1986-02-20   USA    USA   NaN
968            NaN   USA    USA   NaN
...            ...   ...    ...   ...
110599         NaN   USA    USA   NaN
110600         NaN   USA    USA   NaN
110606         NaN   USA    USA   NaN
110609         NaN   USA    USA   NaN
110620         NaN   USA    USA   NaN

[9641 rows x 13 columns]

new_bio=pd.concat([usa,gbr])
new_bio

        athlete_id                name   born_date
born_city  \
48              55       Monique Javer  1967-07-22        Burlingame

697            964    Xóchitl Escobedo  1968-09-17       West Covina

698            965   Angélica Gavaldón  1973-10-03         El Centro

904           1238      Bert Schneider  1897-07-01         Cleveland

968           1352          Laura Berg  1975-01-06  Santa Fe Springs

...            ...                 ...         ...               ...

110103      148512  Benjamin Alexander  1983-05-08            London

110104      148517       Ashley Watson  1993-10-28      Peterborough

110225      148716     Peder Kongshaug  2001-08-13         Wimbledon

110482      149041          Axel Brown  1992-04-02         Harrogate

110545      149111      Jean-Luc Baker  1993-10-07           Burnley
```

```
       born_region born_country                NOC_x  height_cm
weight_kg  \
48      California          USA        Great Britain      177.0
64.0
697     California          USA               Mexico      170.0
60.0
698     California          USA               Mexico      160.0
54.0
904           Ohio          USA               Canada        NaN
NaN
968     California          USA        United States      168.0
61.0
...            ...          ...                  ...        ...
...
110103     England          GBR              Jamaica        NaN
NaN
110104     England          GBR              Jamaica        NaN
NaN
110225     England          GBR               Norway      184.0
86.0
110482     England          GBR  Trinidad and Tobago        NaN
NaN
110545     England          GBR        United States        NaN
NaN

         died_date NOC_y region notes
48             NaN   USA    USA   NaN
697            NaN   USA    USA   NaN
698            NaN   USA    USA   NaN
904     1986-02-20   USA    USA   NaN
968            NaN   USA    USA   NaN
...            ...   ...    ...   ...
110103         NaN   GBR     UK   NaN
110104         NaN   GBR     UK   NaN
110225         NaN   GBR     UK   NaN
110482         NaN   GBR     UK   NaN
110545         NaN   GBR     UK   NaN

[15433 rows x 13 columns]
```

by deafault left to right is top to down data concatenation

Handling Null Values

```
data_new=data.copy()
data_new['k']=1
data_new
```

```
   x  y  z  k  colsum
0  1  2  3  1       8
1  4  5  6  1      31
2  7  8  9  1      73
```

```
data_new.loc[[2],'k']=np.nan
data_new
```

```
   x  y  z    k  colsum
0  1  2  3  1.0       8
1  4  5  6  1.0      31
2  7  8  9  NaN      73
```

```
data_new[data_new['k'].isna()]
```

```
   x  y  z   k  colsum
2  7  8  9 NaN      73
```

```
data_new[data_new['k'].notna()]
```

```
   x  y  z    k  colsum
0  1  2  3  1.0       8
1  4  5  6  1.0      31
```

use fillna to handle nan,give some value,mean,interpolate but use dropna carefully as it removes whole row of nan

interpolate only works for date and numerice and start and end should not be nan,fix it

```
data_new.fillna(1)
```

```
   x  y  z    k  colsum
0  1  2  3  1.0       8
1  4  5  6  1.0      31
2  7  8  9  1.0      73
```

Aggregating Data

```
bios.head()
```

```
   athlete_id                    name   born_date    born_city  \
0           1  Jean-François Blanchy  1886-12-12     Bordeaux
1           2         Arnaud Boetsch  1969-04-01       Meulan
2           3           Jean Borotra  1898-08-13     Biarritz
3           4        Jacques Brugnon  1895-05-11  Paris VIIIe
4           5           Albert Canet  1878-04-17   Wandsworth

            born_region born_country     NOC  height_cm  weight_kg
died_date
0              Gironde          FRA  France        NaN        NaN
1960-10-02
```

```
1              Yvelines          FRA  France      183.0       76.0
NaN
2  Pyrénées-Atlantiques          FRA  France      183.0       76.0
1994-07-17
3                 Paris          FRA  France      168.0       64.0
1978-03-20
4               England          GBR  France        NaN        NaN
1930-07-25
```

reption of same thing

```
bios['born_city'].value_counts()
```

```
born_city
Budapest           1378
Moskva (Moscow)     883
Oslo                708
Stockholm           629
Praha (Prague)      600
                   ...
Kirovgrad             1
Pereiaslav            1
Podgornyy             1
Kudepsta              1
Furmanov              1
Name: count, Length: 22368, dtype: int64
```

```
data_new['x'].value_counts()
```

```
x
1    1
4    1
7    1
Name: count, dtype: int64
```

we first choose rows with USA born country then born country usa's born regions and then their value count ,then tail to get last elements ,it si showing chaining

```
bios[bios['born_country']=='USA']['born_region'].value_counts().tail()
```

```
born_region
South Dakota     27
West Virginia    24
Delaware         22
North Dakota     16
Wyoming          14
Name: count, dtype: int64
```

```
data_new[data_new['x']>=4][data_new['y']>5]['z'].value_counts()
```

```
C:\Users\ACER\AppData\Local\Temp\ipykernel_8328\3984545577.py:1:
UserWarning: Boolean Series key will be reindexed to match DataFrame
index.
  data_new[data_new['x']>=4][data_new['y']>5]['z'].value_counts()

z
9    1
Name: count, dtype: int64
```

some chaining practice ,for reference

```
bios[bios['height_cm']>=183.0]['weight_kg'].mean()

np.float64(85.867534831618)
```

using groupby

```
bios.groupby(['born_country'])['born_region'].value_counts()

born_country  born_region
AFG           Kabul                  24
              Kandahar                1
AGU           Anguilla                2
ALB           Tirana                 11
              Shkodër                 9
                                     ..
ZIM           Mashonaland West        3
              Mashonaland East        2
              Masvingo                2
              Mashonaland Central     1
              Matabeleland South      1
Name: count, Length: 2616, dtype: int64
```

it basicaly group born region on the bsasis of same born country

```
bios.groupby(bios['born_country']=='USA')
['born_region'].value_counts()

born_country  born_region
False         England             4824
              Ontario             1710
              Budapest            1447
              New South Wales     1120
              Bayern              1095
                                  ...
True          South Dakota          27
              West Virginia         24
              Delaware              22
              North Dakota          16
```

```
              Wyoming               14
Name: count, Length: 2556, dtype: int64
```

```
bios.groupby(['weight_kg'])['height_cm'].mean()
```

```
weight_kg
25.0     135.000000
28.0     140.000000
30.0     143.857143
31.0     137.000000
32.0     145.222222
            ...
178.0    187.000000
180.0    187.000000
182.0    196.000000
190.0    200.000000
198.0    200.000000
Name: height_cm, Length: 141, dtype: float64
```

means that for a specifc weight ,what is the mean heigth among many

works like chainig top level is born region then weights works like

->->->

```
bios.groupby(['born_region','weight_kg']).agg({'height_cm':'mean','athlete_id':'sum'})
```

```
                       height_cm  athlete_id
born_region weight_kg
Aargau      50.0           168.0       75746
            51.0           173.0       68724
            53.0           167.0       82987
            54.0           167.0      126385
            56.0           162.0      106792
...                          ...         ...
Žirovnica   61.0           167.0      110903
            65.0           179.0       79408
            72.0           177.0       87457
Ștefan Vodă 55.0           160.0      200473
            96.0           186.0      117118

[33618 rows x 2 columns]
```

creating pivots

```
data_new
```

```
   x  y  z    k  colsum
0  1  2  3  1.0       8
```

```
1  4  5  6  1.0      31
2  7  8  9  NaN      73
```

```
pivot=data_new.pivot(columns='x',index='y',values='k')
```

```
pivot
```

```
x    1    4   7
y
2  1.0  NaN NaN
5  NaN  1.0 NaN
8  NaN  NaN NaN
```

```
pivot1=data_new.pivot(columns='x',index='y',values='z')
pivot1
```

```
x    1    4    7
y
2  3.0  NaN  NaN
5  NaN  6.0  NaN
8  NaN  NaN  9.0
```

```
bioss=bios.copy()
```

```
bioss
```

```
        athlete_id                    name   born_date        born_city
\
0                1  Jean-François Blanchy  1886-12-12         Bordeaux

1                2         Arnaud Boetsch  1969-04-01           Meulan

2                3           Jean Borotra  1898-08-13         Biarritz

3                4        Jacques Brugnon  1895-05-11      Paris VIIIe

4                5           Albert Canet  1878-04-17       Wandsworth

...            ...                    ...         ...              ...

145495      149222      Polina Luchnikova  2002-01-30            Serov

145496      149223    Valeriya Merkusheva  1999-09-20  Moskva (Moscow)

145497      149224        Yuliya Smirnova  1998-05-08           Kotlas

145498      149225         André Foussard  1899-05-19            Niort

145499      149814          Bill Phillips  1913-07-15     Dulwich Hill


                 born_region born_country        NOC  height_cm
```

```
weight_kg  \
0                     Gironde          FRA     France         NaN
NaN
1                    Yvelines          FRA     France       183.0
76.0
2       Pyrénées-Atlantiques           FRA     France       183.0
76.0
3                       Paris          FRA     France       168.0
64.0
4                     England          GBR     France         NaN
NaN
...                       ...          ...        ...         ...
...
145495              Sverdlovsk         RUS         ROC       167.0
61.0
145496                  Moskva         RUS         ROC       168.0
65.0
145497             Arkhangelsk         RUS         ROC       163.0
55.0
145498             Deux-Sèvres         FRA     France       166.0
NaN
145499       New South Wales           AUS  Australia         NaN
NaN

          died_date
0       1960-10-02
1              NaN
2       1994-07-17
3       1978-03-20
4       1930-07-25
...            ...
145495         NaN
145496         NaN
145497         NaN
145498  1986-03-18
145499  2003-10-20

[145500 rows x 10 columns]
```

```
bioss=bioss.drop_duplicates(subset='height_cm',keep='first')
```

```
bioss
```

```
        athlete_id                    name   born_date       born_city
\
0                1  Jean-François Blanchy  1886-12-12        Bordeaux

1                2         Arnaud Boetsch  1969-04-01          Meulan

3                4        Jacques Brugnon  1895-05-11     Paris VIIIe
```

```
5                6      Nicolas Chatelain  1970-01-13          Amiens
6                7          Patrick Chila  1969-11-27     Ris-Orangis
...            ...                    ...         ...             ...
28760        28974          Choe Jong-Sil  1966-06-23             NaN
29054        29268      Dominique Moceanu  1981-09-30             NaN
48662        49016          Nadia Fezzani         NaN             NaN
89070        89782              Yao Ming  1980-09-12  Xuhui District
107408      108533       Peter John Ramos  1985-05-23         Fajardo

        born_region born_country
NOC  \
0           Gironde          FRA
France
1          Yvelines          FRA
France
3             Paris          FRA
France
5             Somme          FRA
France
6           Essonne          FRA
France
...             ...          ...                                      ..
.
28760           NaN          NaN  Democratic People's Republic of
Korea
29054           NaN          NaN                            United
States
48662           NaN          NaN
Libya
89070      Shanghai          CHN              People's Republic of
China
107408  Puerto Rico          PUR                             Puerto
Rico

        height_cm  weight_kg   died_date
0             NaN        NaN  1960-10-02
1           183.0       76.0         NaN
3           168.0       64.0  1978-03-20
5           181.0       70.0         NaN
6           180.0       73.0         NaN
...           ...        ...         ...
```

```
28760       141.0       30.0         NaN
29054       139.0       34.0         NaN
48662       131.0       41.0         NaN
89070       226.0      141.0         NaN
107408      219.0      113.0         NaN

[96 rows x 10 columns]
```

```
bioss=bioss.drop_duplicates(subset='weight_kg',keep='first')

bioss
```

```
        athlete_id                     name   born_date
born_city  \
0                1    Jean-François Blanchy  1886-12-12
Bordeaux
1                2           Arnaud Boetsch  1969-04-01
Meulan
3                4          Jacques Brugnon  1895-05-11          Paris
VIIIe
5                6        Nicolas Chatelain  1970-01-13
Amiens
6                7            Patrick Chila  1969-11-27          Ris-
Orangis
14              15              Damien Éloi  1969-07-04
Vire
16              17               Guy Forget  1965-01-04
Casablanca
25              26            Henri Leconte  1963-07-04
Lillers
26              27        Christophe Legoût  1973-08-06
Montbéliard
41              42            Gillian Clark  1961-09-02
Baghdad
47              48               Sara Gomer  1964-05-13
Torquay
49              50           Gillian Gowers  1964-04-09
NaN
55              56               Valda Lake  1968-10-11
Torquay
69              70              Helen Troke  1964-11-07
Southampton
79              80               Neil Broad  1966-11-20
Cape Town
100            101               Chris Hunt  1968-12-01
Bolton
128            129               Olga Nemes  1968-06-09          Târgu
Mureș
134            135          Karen Stechmann  1971-09-15
Stade
```

```
175             176  Konstantinos Efremoglou  1962-12-04      Athina
(Athens)
186             187         Kenneth Erichsen  1972-12-28
NaN
192             193              Hui So Hung  1958-12-02
NaN
212             213           Krisztina Tóth  1974-05-29
Miskolc
223             224            Károly Németh  1970-08-31
Budapest
417             418      Henrik Kromann Toft  1968-07-11
Aalborg
420             421        Montserrat Martín  1966-07-26
Montblanc
596             598             Chan Siu Yuk  1955-12-21
NaN
701             705            Lyanne Kosaka  1974-02-06  São José dos
Campos
998            1002         Michiel Schapers  1959-10-11
Rotterdam
1080           1087           Bozhil Lozanov  1934-08-16
NaN
1195           1202             Lennox Lewis  1965-09-02
West Ham
1465           1472            Sten Lindberg  1973-11-09
Sydney
1646           1653            Raúl González  1967-06-05
NaN
1673           1680         Alexis Rubalcaba  1972-08-09     Pedro
Betancourt
1674           1681              Félix Savón  1967-09-22          San
Vicente
2647           2659            Milan Šrejber  1963-12-30       Praha
(Prague)
2826           2838          Mirza Delibašić  1954-01-09
Tuzla
2827           2839       Nik'oloz Deriugini  1959-04-30
Kutaisi
2832           2844            Elşad Qadaşev  1968-05-01
Bakı
4084           4098         Gaetano Curcetti  1947-06-29
Foggia
5089           5108       Viktor Pankrashkin  1957-06-19      Moskva
(Moscow)
5092           5111        Zvonko Petričević  1940-07-26
Prizren
5583           5606     Paulinho Villas Boas  1963-01-26            São
Paulo
5599           5622        Vladimir Andreyev  1945-06-14
```

```
Astrakhan
5609          5632          Franjo Arapović  1965-06-02
NaN
5673          5696           Gunther Behnke  1963-01-19
Leverkusen
5683          5706     Aleksandr Belostenny  1959-02-24
Odesa
5781          5804           Tommy Burleson  1952-02-24
Crossnore
27862        28071              Jenny Smith  1980-03-31
Perth
28019        28229             Jessica Tudos  1969-04-04
Toronto
28105        28315                Ana Manso  1966-03-07
Tarragona
28760        28974             Choe Jong-Sil  1966-06-23
NaN
29054        29268        Dominique Moceanu  1981-09-30
NaN
48662        49016             Nadia Fezzani         NaN
NaN
89070        89782                 Yao Ming  1980-09-12       Xuhui
District
107408      108533         Peter John Ramos  1985-05-23
Fajardo

                  born_region born_country  \
0                    Gironde          FRA
1                   Yvelines          FRA
3                      Paris          FRA
5                      Somme          FRA
6                    Essonne          FRA
14                  Calvados          FRA
16         Casablanca-Settat          MAR
25             Pas-de-Calais          FRA
26                     Doubs          FRA
41                   Baghdad          IRQ
47                   England          GBR
49                       NaN          NaN
55                   England          GBR
69                   England          GBR
79              Western Cape          RSA
100                  England          GBR
128                    Mureș          ROU
134            Niedersachsen          GER
175                   Attiki          GRE
186                      NaN          NaN
192                      NaN          NaN
212     Borsod-Abaúj-Zemplén          HUN
```

223	Budapest	HUN
417	Nordjylland	DEN
420	Tarragona	ESP
596	NaN	NaN
701	São Paulo	BRA
998	Zuid-Holland	NED
1080	NaN	NaN
1195	England	GBR
1465	New South Wales	AUS
1646	NaN	NaN
1673	Matanzas	CUB
1674	Guantánamo	CUB
2647	Hlavní město Praha	CZE
2826	Tuzlanski kanton	BIH
2827	Imereti	GEO
2832	Bakı	AZE
4084	Foggia	ITA
5089	Moskva	RUS
5092	Prizren	KOS
5583	São Paulo	BRA
5599	Astrakhan	RUS
5609	NaN	NaN
5673	Nordrhein-Westfalen	GER
5683	Odesa	UKR
5781	North Carolina	USA
27862	Western Australia	AUS
28019	Ontario	CAN
28105	Tarragona	ESP
28760	NaN	NaN
29054	NaN	NaN
48662	NaN	NaN
89070	Shanghai	CHN
107408	Puerto Rico	PUR

	NOC	height_cm	weight_kg	\
0	France	NaN	NaN	
1	France	183.0	76.0	
3	France	168.0	64.0	
5	France	181.0	70.0	
6	France	180.0	73.0	
14	France	165.0	58.0	
16	France	189.0	79.0	
25	France	184.0	78.0	
26	France	177.0	75.0	
41	Great Britain	176.0	68.0	
47	Great Britain	190.0	85.0	
49	Great Britain	157.0	55.0	
55	Great Britain	173.0	61.0	
69	Great Britain	172.0	63.0	

79	Great Britain	191.0	87.0
100	Great Britain	182.0	88.0
128	Germany West Germany	163.0	53.0
134	Germany	169.0	59.0
175	Greece	187.0	74.0
186	Guatemala	179.0	72.0
192	Hong Kong, China	160.0	57.0
212	Hungary	164.0	62.0
223	Hungary	167.0	67.0
417	Denmark	195.0	80.0
420	Spain	156.0	56.0
596	Hong Kong, China	152.0	47.0
701	Brazil	158.0	54.0
998	Netherlands	200.0	83.0
1080	Bulgaria	208.0	133.0
1195	Canada	194.0	100.0
1465	Australia	199.0	97.0
1646	Cuba	153.0	51.0
1673	Cuba	204.0	95.0
1674	Cuba	198.0	91.0
2647	Czechoslovakia	203.0	98.0
2826	Bosnia and Herzegovina Yugoslavia	197.0	86.0
2827	Soviet Union	206.0	105.0
2832	Unified Team	205.0	110.0
4084	Italy	151.0	48.0
5089	Soviet Union	220.0	112.0
5092	Yugoslavia	210.0	130.0
5583	Brazil	217.0	106.0
5599	Soviet Union	215.0	90.0
5609	Croatia Yugoslavia	211.0	120.0
5673	Germany	221.0	114.0
5683	Soviet Union Unified Team	214.0	117.0
5781	United States	223.0	102.0
27862	Australia	149.0	43.0
28019	Canada	137.0	35.0
28105	Spain	132.0	31.0
28760	Democratic People's Republic of Korea	141.0	30.0
29054	United States	139.0	34.0
48662	Libya	131.0	41.0
89070	People's Republic of China	226.0	141.0
107408	Puerto Rico	219.0	113.0

	died_date
0	1960-10-02
1	NaN
3	1978-03-20
5	NaN
6	NaN
14	NaN

```
16              NaN
25              NaN
26              NaN
41              NaN
47              NaN
49              NaN
55              NaN
69              NaN
79              NaN
100             NaN
128             NaN
134             NaN
175             NaN
186             NaN
192             NaN
212             NaN
223             NaN
417             NaN
420             NaN
596             NaN
701             NaN
998             NaN
1080            NaN
1195            NaN
1465            NaN
1646            NaN
1673            NaN
1674            NaN
2647            NaN
2826    2001-12-08
2827            NaN
2832            NaN
4084            NaN
5089    1993-07-24
5092    2009-01-20
5583            NaN
5599            NaN
5609            NaN
5673            NaN
5683    2010-05-24
5781            NaN
27862           NaN
28019           NaN
28105           NaN
28760           NaN
29054           NaN
48662           NaN
89070           NaN
107408          NaN
```

```
pivot_new=bioss.pivot(columns='born_region',index='weight_kg',values='
height_cm')

pivot_new
```

```
born_region    NaN  Astrakhan  Attiki  Baghdad   Bakı  Borsod-Abaúj-
Zemplén  \
weight_kg

NaN             NaN         NaN     NaN      NaN    NaN
NaN
30.0          141.0         NaN     NaN      NaN    NaN
NaN
31.0            NaN         NaN     NaN      NaN    NaN
NaN
34.0          139.0         NaN     NaN      NaN    NaN
NaN
35.0            NaN         NaN     NaN      NaN    NaN
NaN
41.0          131.0         NaN     NaN      NaN    NaN
NaN
43.0            NaN         NaN     NaN      NaN    NaN
NaN
47.0          152.0         NaN     NaN      NaN    NaN
NaN
48.0            NaN         NaN     NaN      NaN    NaN
NaN
51.0          153.0         NaN     NaN      NaN    NaN
NaN
53.0            NaN         NaN     NaN      NaN    NaN
NaN
54.0            NaN         NaN     NaN      NaN    NaN
NaN
55.0          157.0         NaN     NaN      NaN    NaN
NaN
56.0            NaN         NaN     NaN      NaN    NaN
NaN
57.0          160.0         NaN     NaN      NaN    NaN
NaN
58.0            NaN         NaN     NaN      NaN    NaN
NaN
59.0            NaN         NaN     NaN      NaN    NaN
NaN
61.0            NaN         NaN     NaN      NaN    NaN
NaN
62.0            NaN         NaN     NaN      NaN    NaN
164.0
63.0            NaN         NaN     NaN      NaN    NaN
NaN
64.0            NaN         NaN     NaN      NaN    NaN
```

```
NaN
67.0            NaN         NaN      NaN       NaN     NaN
NaN
68.0            NaN         NaN      NaN     176.0     NaN
NaN
70.0            NaN         NaN      NaN       NaN     NaN
NaN
72.0          179.0         NaN      NaN       NaN     NaN
NaN
73.0            NaN         NaN      NaN       NaN     NaN
NaN
74.0            NaN         NaN    187.0       NaN     NaN
NaN
75.0            NaN         NaN      NaN       NaN     NaN
NaN
76.0            NaN         NaN      NaN       NaN     NaN
NaN
78.0            NaN         NaN      NaN       NaN     NaN
NaN
79.0            NaN         NaN      NaN       NaN     NaN
NaN
80.0            NaN         NaN      NaN       NaN     NaN
NaN
83.0            NaN         NaN      NaN       NaN     NaN
NaN
85.0            NaN         NaN      NaN       NaN     NaN
NaN
86.0            NaN         NaN      NaN       NaN     NaN
NaN
87.0            NaN         NaN      NaN       NaN     NaN
NaN
88.0            NaN         NaN      NaN       NaN     NaN
NaN
90.0            NaN       215.0      NaN       NaN     NaN
NaN
91.0            NaN         NaN      NaN       NaN     NaN
NaN
95.0            NaN         NaN      NaN       NaN     NaN
NaN
97.0            NaN         NaN      NaN       NaN     NaN
NaN
98.0            NaN         NaN      NaN       NaN     NaN
NaN
100.0           NaN         NaN      NaN       NaN     NaN
NaN
102.0           NaN         NaN      NaN       NaN     NaN
NaN
105.0           NaN         NaN      NaN       NaN     NaN
NaN
```

```
106.0           NaN         NaN     NaN       NaN    NaN
NaN
110.0           NaN         NaN     NaN       NaN  205.0
NaN
112.0           NaN         NaN     NaN       NaN    NaN
NaN
113.0           NaN         NaN     NaN       NaN    NaN
NaN
114.0           NaN         NaN     NaN       NaN    NaN
NaN
117.0           NaN         NaN     NaN       NaN    NaN
NaN
120.0         211.0         NaN     NaN       NaN    NaN
NaN
130.0           NaN         NaN     NaN       NaN    NaN
NaN
133.0         208.0         NaN     NaN       NaN    NaN
NaN
141.0           NaN         NaN     NaN       NaN    NaN
NaN

born_region  Budapest  Calvados  Casablanca-Settat  Doubs  ...  Puerto
Rico  \
weight_kg                                                  ...

NaN                NaN        NaN                NaN    NaN  ...
NaN
30.0               NaN        NaN                NaN    NaN  ...
NaN
31.0               NaN        NaN                NaN    NaN  ...
NaN
34.0               NaN        NaN                NaN    NaN  ...
NaN
35.0               NaN        NaN                NaN    NaN  ...
NaN
41.0               NaN        NaN                NaN    NaN  ...
NaN
43.0               NaN        NaN                NaN    NaN  ...
NaN
47.0               NaN        NaN                NaN    NaN  ...
NaN
48.0               NaN        NaN                NaN    NaN  ...
NaN
51.0               NaN        NaN                NaN    NaN  ...
NaN
53.0               NaN        NaN                NaN    NaN  ...
NaN
54.0               NaN        NaN                NaN    NaN  ...
NaN
```

```
55.0                NaN        NaN                  NaN    NaN  ...
NaN
56.0                NaN        NaN                  NaN    NaN  ...
NaN
57.0                NaN        NaN                  NaN    NaN  ...
NaN
58.0                NaN      165.0                  NaN    NaN  ...
NaN
59.0                NaN        NaN                  NaN    NaN  ...
NaN
61.0                NaN        NaN                  NaN    NaN  ...
NaN
62.0                NaN        NaN                  NaN    NaN  ...
NaN
63.0                NaN        NaN                  NaN    NaN  ...
NaN
64.0                NaN        NaN                  NaN    NaN  ...
NaN
67.0              167.0        NaN                  NaN    NaN  ...
NaN
68.0                NaN        NaN                  NaN    NaN  ...
NaN
70.0                NaN        NaN                  NaN    NaN  ...
NaN
72.0                NaN        NaN                  NaN    NaN  ...
NaN
73.0                NaN        NaN                  NaN    NaN  ...
NaN
74.0                NaN        NaN                  NaN    NaN  ...
NaN
75.0                NaN        NaN                  NaN  177.0  ...
NaN
76.0                NaN        NaN                  NaN    NaN  ...
NaN
78.0                NaN        NaN                  NaN    NaN  ...
NaN
79.0                NaN        NaN                189.0    NaN  ...
NaN
80.0                NaN        NaN                  NaN    NaN  ...
NaN
83.0                NaN        NaN                  NaN    NaN  ...
NaN
85.0                NaN        NaN                  NaN    NaN  ...
NaN
86.0                NaN        NaN                  NaN    NaN  ...
NaN
87.0                NaN        NaN                  NaN    NaN  ...
NaN
88.0                NaN        NaN                  NaN    NaN  ...
```

```
NaN
90.0                NaN        NaN                 NaN     NaN  ...
NaN
91.0                NaN        NaN                 NaN     NaN  ...
NaN
95.0                NaN        NaN                 NaN     NaN  ...
NaN
97.0                NaN        NaN                 NaN     NaN  ...
NaN
98.0                NaN        NaN                 NaN     NaN  ...
NaN
100.0               NaN        NaN                 NaN     NaN  ...
NaN
102.0               NaN        NaN                 NaN     NaN  ...
NaN
105.0               NaN        NaN                 NaN     NaN  ...
NaN
106.0               NaN        NaN                 NaN     NaN  ...
NaN
110.0               NaN        NaN                 NaN     NaN  ...
NaN
112.0               NaN        NaN                 NaN     NaN  ...
NaN
113.0               NaN        NaN                 NaN     NaN  ...
219.0
114.0               NaN        NaN                 NaN     NaN  ...
NaN
117.0               NaN        NaN                 NaN     NaN  ...
NaN
120.0               NaN        NaN                 NaN     NaN  ...
NaN
130.0               NaN        NaN                 NaN     NaN  ...
NaN
133.0               NaN        NaN                 NaN     NaN  ...
NaN
141.0               NaN        NaN                 NaN     NaN  ...
NaN

born_region  Shanghai  Somme  São Paulo  Tarragona  Tuzlanski
kanton  \
weight_kg

NaN                NaN    NaN        NaN        NaN                NaN

30.0               NaN    NaN        NaN        NaN                NaN

31.0               NaN    NaN        NaN      132.0                NaN

34.0               NaN    NaN        NaN        NaN                NaN
```

35.0	NaN	NaN	NaN	NaN	NaN
41.0	NaN	NaN	NaN	NaN	NaN
43.0	NaN	NaN	NaN	NaN	NaN
47.0	NaN	NaN	NaN	NaN	NaN
48.0	NaN	NaN	NaN	NaN	NaN
51.0	NaN	NaN	NaN	NaN	NaN
53.0	NaN	NaN	NaN	NaN	NaN
54.0	NaN	NaN	158.0	NaN	NaN
55.0	NaN	NaN	NaN	NaN	NaN
56.0	NaN	NaN	NaN	156.0	NaN
57.0	NaN	NaN	NaN	NaN	NaN
58.0	NaN	NaN	NaN	NaN	NaN
59.0	NaN	NaN	NaN	NaN	NaN
61.0	NaN	NaN	NaN	NaN	NaN
62.0	NaN	NaN	NaN	NaN	NaN
63.0	NaN	NaN	NaN	NaN	NaN
64.0	NaN	NaN	NaN	NaN	NaN
67.0	NaN	NaN	NaN	NaN	NaN
68.0	NaN	NaN	NaN	NaN	NaN
70.0	NaN	181.0	NaN	NaN	NaN
72.0	NaN	NaN	NaN	NaN	NaN
73.0	NaN	NaN	NaN	NaN	NaN
74.0	NaN	NaN	NaN	NaN	NaN
75.0	NaN	NaN	NaN	NaN	NaN
76.0	NaN	NaN	NaN	NaN	NaN
78.0	NaN	NaN	NaN	NaN	NaN

79.0	NaN	NaN	NaN	NaN	NaN
80.0	NaN	NaN	NaN	NaN	NaN
83.0	NaN	NaN	NaN	NaN	NaN
85.0	NaN	NaN	NaN	NaN	NaN
86.0	NaN	NaN	NaN	NaN	197.0
87.0	NaN	NaN	NaN	NaN	NaN
88.0	NaN	NaN	NaN	NaN	NaN
90.0	NaN	NaN	NaN	NaN	NaN
91.0	NaN	NaN	NaN	NaN	NaN
95.0	NaN	NaN	NaN	NaN	NaN
97.0	NaN	NaN	NaN	NaN	NaN
98.0	NaN	NaN	NaN	NaN	NaN
100.0	NaN	NaN	NaN	NaN	NaN
102.0	NaN	NaN	NaN	NaN	NaN
105.0	NaN	NaN	NaN	NaN	NaN
106.0	NaN	NaN	217.0	NaN	NaN
110.0	NaN	NaN	NaN	NaN	NaN
112.0	NaN	NaN	NaN	NaN	NaN
113.0	NaN	NaN	NaN	NaN	NaN
114.0	NaN	NaN	NaN	NaN	NaN
117.0	NaN	NaN	NaN	NaN	NaN
120.0	NaN	NaN	NaN	NaN	NaN
130.0	NaN	NaN	NaN	NaN	NaN
133.0	NaN	NaN	NaN	NaN	NaN
141.0	226.0	NaN	NaN	NaN	NaN

born_region	Western Australia	Western Cape	Yvelines	Zuid-Holland

weight_kg				
NaN	NaN	NaN	NaN	NaN
30.0	NaN	NaN	NaN	NaN
31.0	NaN	NaN	NaN	NaN
34.0	NaN	NaN	NaN	NaN
35.0	NaN	NaN	NaN	NaN
41.0	NaN	NaN	NaN	NaN
43.0	149.0	NaN	NaN	NaN
47.0	NaN	NaN	NaN	NaN
48.0	NaN	NaN	NaN	NaN
51.0	NaN	NaN	NaN	NaN
53.0	NaN	NaN	NaN	NaN
54.0	NaN	NaN	NaN	NaN
55.0	NaN	NaN	NaN	NaN
56.0	NaN	NaN	NaN	NaN
57.0	NaN	NaN	NaN	NaN
58.0	NaN	NaN	NaN	NaN
59.0	NaN	NaN	NaN	NaN
61.0	NaN	NaN	NaN	NaN
62.0	NaN	NaN	NaN	NaN
63.0	NaN	NaN	NaN	NaN
64.0	NaN	NaN	NaN	NaN
67.0	NaN	NaN	NaN	NaN
68.0	NaN	NaN	NaN	NaN
70.0	NaN	NaN	NaN	NaN
72.0	NaN	NaN	NaN	NaN
73.0	NaN	NaN	NaN	NaN
74.0	NaN	NaN	NaN	NaN
75.0	NaN	NaN	NaN	NaN
76.0	NaN	NaN	183.0	NaN
78.0	NaN	NaN	NaN	NaN
79.0	NaN	NaN	NaN	NaN
80.0	NaN	NaN	NaN	NaN
83.0	NaN	NaN	NaN	200.0
85.0	NaN	NaN	NaN	NaN
86.0	NaN	NaN	NaN	NaN
87.0	NaN	191.0	NaN	NaN
88.0	NaN	NaN	NaN	NaN
90.0	NaN	NaN	NaN	NaN
91.0	NaN	NaN	NaN	NaN
95.0	NaN	NaN	NaN	NaN
97.0	NaN	NaN	NaN	NaN
98.0	NaN	NaN	NaN	NaN
100.0	NaN	NaN	NaN	NaN
102.0	NaN	NaN	NaN	NaN
105.0	NaN	NaN	NaN	NaN
106.0	NaN	NaN	NaN	NaN
110.0	NaN	NaN	NaN	NaN
112.0	NaN	NaN	NaN	NaN

```
113.0                       NaN           NaN       NaN           NaN
114.0                       NaN           NaN       NaN           NaN
117.0                       NaN           NaN       NaN           NaN
120.0                       NaN           NaN       NaN           NaN
130.0                       NaN           NaN       NaN           NaN
133.0                       NaN           NaN       NaN           NaN
141.0                       NaN           NaN       NaN           NaN

[55 rows x 40 columns]
```

Advance Functionality

.shift(),.rank(),.rolling(),.cumsum()

```
data
```

```
   x  y  z   k  colsum
0  1  2  3   2       8
1  4  5  6  16      31
2  7  8  9  49      73
```

```
data['pre']=data['colsum'].shift(1)
```

```
data
```

```
   x  y  z   k  colsum   pre
0  1  2  3   2       8   NaN
1  4  5  6  16      31   8.0
2  7  8  9  49      73  31.0
```

shift is used to shift the whole data column wise by one by defaults, or you can increase the shift rate,you can shift backward by using negative values

```
bios
```

```
        athlete_id                    name   born_date        born_city
\
0                1  Jean-François Blanchy  1886-12-12         Bordeaux

1                2         Arnaud Boetsch  1969-04-01           Meulan

2                3           Jean Borotra  1898-08-13         Biarritz

3                4        Jacques Brugnon  1895-05-11      Paris VIIIe

4                5           Albert Canet  1878-04-17       Wandsworth

...            ...                    ...         ...              ...

145495      149222      Polina Luchnikova  2002-01-30            Serov
```

```
145496      149223    Valeriya Merkusheva  1999-09-20  Moskva (Moscow)
145497      149224        Yuliya Smirnova  1998-05-08           Kotlas
145498      149225         André Foussard  1899-05-19            Niort
145499      149814           Bill Phillips  1913-07-15     Dulwich Hill

                  born_region born_country        NOC  height_cm
weight_kg  \
0                    Gironde          FRA     France        NaN
NaN
1                   Yvelines          FRA     France      183.0
76.0
2       Pyrénées-Atlantiques          FRA     France      183.0
76.0
3                      Paris          FRA     France      168.0
64.0
4                    England          GBR     France        NaN
NaN
...                      ...          ...        ...        ...
...
145495            Sverdlovsk          RUS        ROC      167.0
61.0
145496                Moskva          RUS        ROC      168.0
65.0
145497            Arkhangelsk          RUS        ROC      163.0
55.0
145498            Deux-Sèvres          FRA     France      166.0
NaN
145499       New South Wales          AUS  Australia        NaN
NaN

         died_date
0       1960-10-02
1              NaN
2       1994-07-17
3       1978-03-20
4       1930-07-25
...            ...
145495         NaN
145496         NaN
145497         NaN
145498  1986-03-18
145499  2003-10-20

[145500 rows x 10 columns]
```

```
bios['height_rank']=bios['height_cm'].rank(ascending=False)

bios.sort_values(['height_rank']).interpolate()
```

```
C:\Users\ACER\AppData\Local\Temp\ipykernel_8328\2823149012.py:1:
FutureWarning: DataFrame.interpolate with object dtype is deprecated
and will raise in a future version. Call obj.infer_objects(copy=False)
before interpolating instead.
  bios.sort_values(['height_rank']).interpolate()
```

```
        athlete_id                    name   born_date
born_city  \
89070        89782                Yao Ming  1980-09-12      Xuhui
District
6978          7013         Arvydas Sabonis  1964-12-19
Kaunas
5781          5804          Tommy Burleson  1952-02-24
Crossnore
5673          5696          Gunther Behnke  1963-01-19
Leverkusen
89075        89787          Roberto Dueñas  1975-11-01
Madrid
...            ...                     ...         ...
...
145490      149217             Sin Ye-Chan  1995-06-13
NaN
145491      149218           Matthew Wepke  1989-12-05
NaN
145492      149219  Carlos García-Ordóñez  1927-04-24  La Habana
(Havana)
145493      149220     Landysh Falyakhova  1998-08-31    Dva Polya
Artash
145499      149814           Bill Phillips  1913-07-15        Dulwich
Hill

                 born_region born_country                          NOC
\
89070               Shanghai          CHN  People's Republic of China

6978                  Kaunas          LTU      Lithuania Soviet Union

5781           North Carolina          USA                United States

5673     Nordrhein-Westfalen          GER                      Germany

89075                 Madrid          ESP                        Spain

...                      ...          ...                          ...

145490                  NaN          NaN           Republic of Korea
```

```
145491                   NaN          NaN                      Jamaica

145492   Ciudad de La Habana          CUB                         Cuba

145493  Respublika Tatarstan          RUS                          ROC

145499       New South Wales          AUS                    Australia


        height_cm  weight_kg   died_date  height_rank
89070       226.0      141.0         NaN          1.0
6978        223.0      122.0         NaN          2.5
5781        223.0      102.0         NaN          2.5
5673        221.0      114.0         NaN          5.0
89075       221.0      137.0         NaN          5.0
...           ...        ...         ...          ...
145490      127.0       62.0         NaN     106650.5
145491      127.0       62.0         NaN     106650.5
145492      127.0       62.0  2019-11-24     106650.5
145493      127.0       62.0         NaN     106650.5
145499      127.0       62.0  2003-10-20     106650.5

[145500 rows x 11 columns]
```

give rating or ranking to a data column

```
bios

        athlete_id                    name   born_date        born_city
\
0                1  Jean-François Blanchy  1886-12-12         Bordeaux

1                2         Arnaud Boetsch  1969-04-01           Meulan

2                3           Jean Borotra  1898-08-13         Biarritz

3                4        Jacques Brugnon  1895-05-11      Paris VIIIe

4                5           Albert Canet  1878-04-17       Wandsworth

...            ...                    ...         ...              ...

145495      149222      Polina Luchnikova  2002-01-30           Serov

145496      149223    Valeriya Merkusheva  1999-09-20  Moskva (Moscow)

145497      149224        Yuliya Smirnova  1998-05-08          Kotlas

145498      149225         André Foussard  1899-05-19           Niort

145499      149814          Bill Phillips  1913-07-15     Dulwich Hill
```

```
                  born_region born_country        NOC  height_cm
weight_kg  \
0                    Gironde          FRA     France        NaN
NaN
1                   Yvelines          FRA     France      183.0
76.0
2       Pyrénées-Atlantiques          FRA     France      183.0
76.0
3                      Paris          FRA     France      168.0
64.0
4                    England          GBR     France        NaN
NaN
...                      ...          ...        ...        ...
...
145495             Sverdlovsk          RUS        ROC      167.0
61.0
145496                 Moskva          RUS        ROC      168.0
65.0
145497             Arkhangelsk          RUS        ROC      163.0
55.0
145498             Deux-Sèvres          FRA     France      166.0
NaN
145499       New South Wales          AUS  Australia        NaN
NaN

         died_date  height_rank
0       1960-10-02          NaN
1              NaN      79054.5
2       1994-07-17      79054.5
3       1978-03-20      22677.0
4       1930-07-25          NaN
...            ...          ...
145495         NaN      19463.5
145496         NaN      22677.0
145497         NaN      10346.5
145498  1986-03-18      17377.5
145499  2003-10-20          NaN

[145500 rows x 11 columns]
```

data

```
   x  y  z   k  colsum   pre
0  1  2  3   2       8   NaN
1  4  5  6  16      31   8.0
2  7  8  9  49      73  31.0
```

```
data['summer']=data['colsum'].cumsum()
```

used to find cummulative sum

```
data
```

```
   x  y  z   k  colsum   pre  summer
0  1  2  3   2       8   NaN       8
1  4  5  6  16      31   8.0      39
2  7  8  9  49      73  31.0     112
```

the .rolling() function is used to create a moving window view over a sequence of data basically used to select specific number of elemnts one time and perform operations on them

```
data['3d_cumm']=data['colsum'].rolling(3).sum()
data
```

```
   x  y  z   k  colsum   pre  summer  3d_cumm
0  1  2  3   2       8   NaN       8      NaN
1  4  5  6  16      31   8.0      39      NaN
2  7  8  9  49      73  31.0     112    112.0
```

new functionallity

```
pd.__version__
```

```
'2.3.2'
```

pyarrow

PyArrow is a Python library for reading and writing Apache Arrow data, which is a cross-language development platform for in-memory data. It enables fast data interchange between systems and supports efficient columnar data structures, making it useful for big data, analytics, and interoperability with tools like pandas, Parquet, and more. PyArrow is commonly used for high-performance data processing, serialization, and file formats such as Parquet and Feather.

```
dam_numpy=pd.read_csv('data/dam.csv')
dam_arrow=pd.read_csv('data/dam.csv',engine='pyarrow',dtype_backend='pyarrow')

dam_numpy
```

```
   x  y  z
0  1  2  3
1  4  5  6
2  7  8  9
```

```
dam_numpy.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3 entries, 0 to 2
Data columns (total 3 columns):
```

```
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   x       3 non-null      int64
 1   y       3 non-null      int64
 2   z       3 non-null      int64
dtypes: int64(3)
memory usage: 204.0 bytes
```

```
dam_arrow['val']=["ss","aa","kk"]
dam_arrow
```

```
   x  y  z val
0  1  2  3  ss
1  4  5  6  aa
2  7  8  9  kk
```

```
dam_arrow.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3 entries, 0 to 2
Data columns (total 4 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   x       3 non-null      int64[pyarrow]
 1   y       3 non-null      int64[pyarrow]
 2   z       3 non-null      int64[pyarrow]
 3   val     3 non-null      object
dtypes: int64[pyarrow](3), object(1)
memory usage: 228.0+ bytes
```

```
# 1. **Faster Coding**: Copilot suggests code snippets and functions as you type, speeding up data analysis tasks.
# 2. **Error Reduction**: It helps avoid common mistakes by providing context-aware suggestions and correcting syntax.
# 3. **Learning Aid**: Copilot can show best practices and new pandas features, helping users learn more efficiently.
# 4. **Automation**: Routine tasks like data cleaning, transformation, and visualization can be automated with smart code completions.
# 5. **Documentation**: Copilot can generate comments and explanations for complex pandas operations, improving code readability.
```

```
dam_numpy['rowsum'] = dam_numpy.sum(axis=1)
dam_numpy
```

```
   x  y  z  rowsum
0  1  2  3       6
1  4  5  6      15
2  7  8  9      24
```