```
import numpy as np;
import pandas as pd;
```

```
mass1= pd.Series((0.33, 4.07,  5.97, 0.642, 1090, 568, 86.0, 102, 0.0146, 0.000292),
                 index = ['mercury', 'venus', 'earth', 'mars', 'jupiter', 'saturn', 'u
dia1 = pd.Series((4079, 12104, 12756, 3475, 6792, 142904, 120536, 51110, 49528, 2370),
                 index = ['mercury', 'venus', 'earth', 'mars', 'jupiter', 'saturn', 'u
```

```
mass1
```

```
mercury       0.330000
venus         4.070000
earth         5.970000
mars          0.642000
jupiter    1090.000000
saturn      568.000000
uranus       86.000000
neptune     102.000000
pluto         0.014600
eris          0.000292
dtype: float64
```

```
dia1
```

```
mercury      4079
venus       12104
earth       12756
mars         3475
jupiter      6792
saturn     142904
uranus     120536
neptune     51110
pluto       49528
eris         2370
dtype: int64
```

```
df100 = pd.DataFrame(mass1,dia1)
df100
```

	0
4079	NaN
12104	NaN

```
df100 = pd.DataFrame({'mass' : mass1 ,'dia' : dia1})
df100
```

	mass	dia
mercury	0.330000	4079
venus	4.070000	12104
earth	5.970000	12756
mars	0.642000	3475
jupiter	1090.000000	6792
saturn	568.000000	142904
uranus	86.000000	120536
neptune	102.000000	51110
pluto	0.014600	49528
eris	0.000292	2370

```
df100['mass']
```

```
mercury       0.330000
venus         4.070000
earth         5.970000
mars          0.642000
jupiter    1090.000000
saturn      568.000000
uranus       86.000000
neptune     102.000000
pluto         0.014600
eris          0.000292
Name: mass, dtype: float64
```

```
df100['dia']
```

```
mercury      4079
venus       12104
earth       12756
mars         3475
jupiter      6792
saturn     142904
uranus     120536
neptune     51110
pluto       49528
eris         2370
Name: dia, dtype: int64
```

```
df100['mass']['earth']
```

```
5.97
```

```
df100.mass.earth
```

```
5.97
```

```
#adding new column to df
df100['Population']=0
```

```
df100
```

	mass	dia	Population
mercury	0.330000	4079	0
venus	4.070000	12104	0
earth	5.970000	12756	0
mars	0.642000	3475	0
jupiter	1090.000000	6792	0
saturn	568.000000	142904	0
uranus	86.000000	120536	0
neptune	102.000000	51110	0
pluto	0.014600	49528	0
eris	0.000292	2370	0

```
df100.Population.earth = 8000000000

```

```
/usr/local/lib/python3.7/dist-packages/pandas/core/generic.py:5516: SettingWithCopyWa
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/u
  self[name] = value
```

```
df100['Population']['mars'] = 1
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: SettingWithCopyWarnin
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/u
  """Entry point for launching an IPython kernel.
```

```
df100
```

	mass	dia	Population
mercury	0.330000	4079	0
venus	4.070000	12104	0
earth	5.970000	12756	8000000000
mars	0.642000	3475	1
jupiter	1090.000000	6792	0
saturn	568.000000	142904	0
uranus	86.000000	120536	0
neptune	102.000000	51110	0
pluto	0.014600	49528	0
eris	0.000292	2370	0

```
df100['mass'] is df100.mass
```

```
True
```

```
df100.loc['earth',:]
```

```
mass          5.970000e+00
dia           1.275600e+04
Population    8.000000e+09
Name: earth, dtype: float64
```

```
df100['MeanMass']=0
```

```
df100
```

	mass	dia	Population	MeanMass

```python
df100['MeanMass']=np.mean(df100.mass)
```

```python
df100
```

	mass	dia	Population	MeanMass
mercury	0.330000	4079	0	185.702689
venus	4.070000	12104	0	185.702689
earth	5.970000	12756	8000000000	185.702689
mars	0.642000	3475	1	185.702689
jupiter	1090.000000	6792	0	185.702689
saturn	568.000000	142904	0	185.702689
uranus	86.000000	120536	0	185.702689
neptune	102.000000	51110	0	185.702689
pluto	0.014600	49528	0	185.702689
eris	0.000292	2370	0	185.702689

```python
df100.drop('MeanMass',axis=1)
```

	mass	dia	Population
mercury	0.330000	4079	0
venus	4.070000	12104	0
earth	5.970000	12756	8000000000
mars	0.642000	3475	1
jupiter	1090.000000	6792	0
saturn	568.000000	142904	0
uranus	86.000000	120536	0
neptune	102.000000	51110	0
pluto	0.014600	49528	0
eris	0.000292	2370	0

```python
df100
```

	mass	dia	Population	MeanMass
mercury	0.330000	4079	0	185.702689
venus	4.070000	12104	0	185.702689
earth	5.970000	12756	8000000000	185.702689
mars	0.642000	3475	1	185.702689
jupiter	1090.000000	6792	0	185.702689
saturn	568.000000	142904	0	185.702689
uranus	86.000000	120536	0	185.702689
neptune	102.000000	51110	0	185.702689

```
df100.drop('MeanMass',axis=1,inplace=True)
```

```
df100
```

	mass	dia	Population
mercury	0.330000	4079	0
venus	4.070000	12104	0
earth	5.970000	12756	8000000000
mars	0.642000	3475	1
jupiter	1090.000000	6792	0
saturn	568.000000	142904	0
uranus	86.000000	120536	0
neptune	102.000000	51110	0
pluto	0.014600	49528	0
eris	0.000292	2370	0

```
df100.mean
```

```
<bound method NDFrame._add_numeric_operations.<locals>.mean of                 mass
mercury     0.330000    4079           0
venus       4.070000   12104           0
earth       5.970000   12756  8000000000
mars        0.642000    3475           1
jupiter  1090.000000    6792           0
saturn    568.000000  142904           0
uranus     86.000000  120536           0
neptune   102.000000   51110           0
pluto       0.014600   49528           0
eris        0.000292    2370           0>
```

```
df100.mean()
```

```
mass          1.857027e+02
dia           4.056540e+04
Population    8.000000e+08
dtype: float64
```

```
df100.median()
```

```
mass              5.02
dia           12430.00
Population        0.00
dtype: float64
```

```
df100.mean(axis=1)
```

```
mercury    1.359777e+03
venus      4.036023e+03
earth      2.666671e+09
mars       1.158881e+03
jupiter    2.627333e+03
saturn     4.782400e+04
uranus     4.020733e+04
neptune    1.707067e+04
pluto      1.650934e+04
eris       7.900001e+02
dtype: float64
```

```
df100.mean(axis=0)
```

```
mass          1.857027e+02
dia           4.056540e+04
Population    8.000000e+08
dtype: float64
```

```
df100.min()
```

```
mass             0.000292
dia           2370.000000
Population       0.000000
dtype: float64
```

```
df100.max()
```

```
mass          1.090000e+03
dia           1.429040e+05
Population    8.000000e+09
dtype: float64
```

```
  df100.quantile(0.25)
```

```
mass             0.408
dia           4757.250
```

```
Population       0.000
Name: 0.25, dtype: float64
```

```
df100.shape
```

```
(10, 3)
```

```
df100.size
```

```
30
```

```
df100.describe()
```

	mass	dia	Population
count	10.000000	10.000000	1.000000e+01
mean	185.702689	40565.400000	8.000000e+08
std	362.663272	51585.854011	2.529822e+09
min	0.000292	2370.000000	0.000000e+00
25%	0.408000	4757.250000	0.000000e+00
50%	5.020000	12430.000000	0.000000e+00
75%	98.000000	50714.500000	0.000000e+00
max	1090.000000	142904.000000	8.000000e+09

```
df100.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 10 entries, mercury to eris
Data columns (total 3 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   mass        10 non-null     float64
 1   dia         10 non-null     int64  
 2   Population  10 non-null     int64  
dtypes: float64(1), int64(2)
memory usage: 620.0+ bytes
```

```
import seaborn as sb
```

```
sb.load_dataset('planets')
#https://github.com/mwaskom/seaborn-data/blob/master/planets.csv
```

	method	number	orbital_period	mass	distance	year
0	Radial Velocity	1	269.300000	7.10	77.40	2006
1	Radial Velocity	1	874.774000	2.21	56.95	2008
2	Radial Velocity	1	763.000000	2.60	19.84	2011
3	Radial Velocity	1	326.030000	19.40	110.62	2007
4	Radial Velocity	1	516.220000	10.50	119.47	2009
...	...	...	...	...	...	...
1030	Transit	1	3.941507	NaN	172.00	2006
1031	Transit	1	2.615864	NaN	148.00	2007
1032	Transit	1	3.191524	NaN	174.00	2007

```python
df_planets=sb.load_dataset('planets')
df_planets
```

	method	number	orbital_period	mass	distance	year
0	Radial Velocity	1	269.300000	7.10	77.40	2006
1	Radial Velocity	1	874.774000	2.21	56.95	2008
2	Radial Velocity	1	763.000000	2.60	19.84	2011
3	Radial Velocity	1	326.030000	19.40	110.62	2007
4	Radial Velocity	1	516.220000	10.50	119.47	2009
...	...	...	...	...	...	...
1030	Transit	1	3.941507	NaN	172.00	2006
1031	Transit	1	2.615864	NaN	148.00	2007
1032	Transit	1	3.191524	NaN	174.00	2007
1033	Transit	1	4.125083	NaN	293.00	2008
1034	Transit	1	4.187757	NaN	260.00	2008

1035 rows × 6 columns

```python
df_planets.shape
```

```
(1035, 6)
```

```python
df_planets.size
```

```
6210
```

```python
df_planets.describe()
```

	number	orbital_period	mass	distance	year
count	1035.000000	992.000000	513.000000	808.000000	1035.000000
mean	1.785507	2002.917596	2.638161	264.069282	2009.070531
std	1.240976	26014.728304	3.818617	733.116493	3.972567
min	1.000000	0.090706	0.003600	1.350000	1989.000000
25%	1.000000	5.442540	0.229000	32.560000	2007.000000
50%	1.000000	39.979500	1.260000	55.250000	2010.000000
75%	2.000000	526.005000	3.040000	178.500000	2012.000000
max	7.000000	730000.000000	25.000000	8500.000000	2014.000000

```
df_planets.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1035 entries, 0 to 1034
Data columns (total 6 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   method          1035 non-null   object
 1   number          1035 non-null   int64
 2   orbital_period  992 non-null    float64
 3   mass            513 non-null    float64
 4   distance        808 non-null    float64
 5   year            1035 non-null   int64
dtypes: float64(3), int64(2), object(1)
memory usage: 48.6+ KB
```

```
for row in df_planets :
    for col in df_planets :
        if pd.isnull(df_planets.loc(row,col)) :
            df_planets.drop(row,inplace=True)
            break
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-45-90e78c5214a7> in <module>()
      1 for row in df_planets :
      2     for col in df_planets :
----> 3         if pd.isnull(df_planets.loc(row,col)) :
      4             df_planets.drop(row,inplace=True)
      5             break

TypeError: __call__() takes from 1 to 2 positional arguments but 3 were given
```

SEARCH STACK OVERFLOW

```
for row in df_planets.index :
    for col in df_planets.columns :
        if pd.isnull(df_planets.loc[row,col]) :
            df_planets.drop(row,inplace=True)
            break
```

```python
df_planets.describe()
```

	number	orbital_period	mass	distance	year
count	498.00000	498.000000	498.000000	498.000000	498.000000
mean	1.73494	835.778671	2.509320	52.068213	2007.377510
std	1.17572	1469.128259	3.636274	46.596041	4.167284
min	1.00000	1.328300	0.003600	1.350000	1989.000000
25%	1.00000	38.272250	0.212500	24.497500	2005.000000
50%	1.00000	357.000000	1.245000	39.940000	2009.000000
75%	2.00000	999.600000	2.867500	59.332500	2011.000000
max	6.00000	17337.500000	25.000000	354.000000	2014.000000

```python
df_planets.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 498 entries, 0 to 784
Data columns (total 6 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   method          498 non-null    object 
 1   number          498 non-null    int64  
 2   orbital_period  498 non-null    float64
 3   mass            498 non-null    float64
 4   distance        498 non-null    float64
 5   year            498 non-null    int64  
dtypes: float64(3), int64(2), object(1)
memory usage: 27.2+ KB
```

```python
for rows,columns in df_planets.iterrows() : #used for traversing instead of for loop
    print(rows)
    print(columns)
    break
```

```
0
method            Radial Velocity
number                          1
orbital_period              269.3
mass                          7.1
distance                     77.4
year                         2006
Name: 0, dtype: object
```

```python
for columns,rows in df_planets.iterrows() :
    #used for traversing instead of for loop
    if pd.isnull(rows).any() :
        df.planets.drop(rows,inplace=True)
    break
```

```
df_planets.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 498 entries, 0 to 784
Data columns (total 6 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   method          498 non-null    object 
 1   number          498 non-null    int64  
 2   orbital_period  498 non-null    float64
 3   mass            498 non-null    float64
 4   distance        498 non-null    float64
 5   year            498 non-null    int64  
dtypes: float64(3), int64(2), object(1)
memory usage: 27.2+ KB
```

```
df_planets.dropna(inplace=True)
```

```
df_planets.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 498 entries, 0 to 784
Data columns (total 6 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   method          498 non-null    object 
 1   number          498 non-null    int64  
 2   orbital_period  498 non-null    float64
 3   mass            498 non-null    float64
 4   distance        498 non-null    float64
 5   year            498 non-null    int64  
dtypes: float64(3), int64(2), object(1)
memory usage: 27.2+ KB
```

```
df_planets.describe()
```

	number	orbital_period	mass	distance	year
count	498.00000	498.000000	498.000000	498.000000	498.000000
mean	1.73494	835.778671	2.509320	52.068213	2007.377510
std	1.17572	1469.128259	3.636274	46.596041	4.167284
min	1.00000	1.328300	0.003600	1.350000	1989.000000
25%	1.00000	38.272250	0.212500	24.497500	2005.000000
50%	1.00000	357.000000	1.245000	39.940000	2009.000000
75%	2.00000	999.600000	2.867500	59.332500	2011.000000
max	6.00000	17337.500000	25.000000	354.000000	2014.000000

Sir way

```
df_2=df_planets.copy()
```

```
df_2.describe()
```

	number	orbital_period	mass	distance	year
count	498.00000	498.000000	498.000000	498.000000	498.000000
mean	1.73494	835.778671	2.509320	52.068213	2007.377510
std	1.17572	1469.128259	3.636274	46.596041	4.167284
min	1.00000	1.328300	0.003600	1.350000	1989.000000
25%	1.00000	38.272250	0.212500	24.497500	2005.000000
50%	1.00000	357.000000	1.245000	39.940000	2009.000000
75%	2.00000	999.600000	2.867500	59.332500	2011.000000
max	6.00000	17337.500000	25.000000	354.000000	2014.000000

```
p_75 = df_2.distance.quantile(.75)
for ind,row in df_2.iterrows() :
    if row['year']< 2010 :
        df_2.drop(ind,inplace=True)
        continue;
    if (row['method'] != 'Radial Velocity' and row['method']!= 'Transit') :
        df_2.drop(ind,inplace=True)
        continue;
    if(row['distance']<p_75) :
        df_2.drop(ind,inplace=True)
        continue;
```

```
df_2.describe()
```

	number	orbital_period	mass	distance	year
count	50.000000	50.000000	50.000000	50.000000	50.000000
mean	1.300000	763.904808	3.322740	133.142600	2011.360000
std	0.505076	966.789870	3.648002	70.378699	1.120496
min	1.000000	2.703390	0.770000	65.620000	2010.000000
25%	1.000000	255.555000	1.325000	80.205000	2011.000000
50%	1.000000	550.500000	1.875000	121.070000	2011.000000
75%	2.000000	873.625000	3.400000	150.097500	2012.000000
max	3.000000	5584.000000	20.600000	354.000000	2014.000000

My way

```
#filter rows for planets found in 2010s and method is radial velocity or in Transit ar
dis75=np.percentile(df_planets['distance'],75)
dis75
```

```
59.3325
```

```
#more efficient way
```

```
df3=df_planets.copy()
df3.describe()
```

	number	orbital_period	mass	distance	year
count	498.00000	498.000000	498.000000	498.000000	498.000000
mean	1.73494	835.778671	2.509320	52.068213	2007.377510
std	1.17572	1469.128259	3.636274	46.596041	4.167284
min	1.00000	1.328300	0.003600	1.350000	1989.000000
25%	1.00000	38.272250	0.212500	24.497500	2005.000000
50%	1.00000	357.000000	1.245000	39.940000	2009.000000
75%	2.00000	999.600000	2.867500	59.332500	2011.000000
max	6.00000	17337.500000	25.000000	354.000000	2014.000000

```
df4=df3[
    (df3['year']>=2010) &
    ((df3['method']=='Radial Velocity') | (df3['method']=='Transit')) &
    (df3['distance']>p_75)
    ]
df4.describe()
```

	number	orbital_period	mass	distance	year
count	50.000000	50.000000	50.000000	50.000000	50.000000
mean	1.300000	763.904808	3.322740	133.142600	2011.360000
std	0.505076	966.789870	3.648002	70.378699	1.120496
min	1.000000	2.703390	0.770000	65.620000	2010.000000
25%	1.000000	255.555000	1.325000	80.205000	2011.000000
50%	1.000000	550.500000	1.875000	121.070000	2011.000000
75%	2.000000	873.625000	3.400000	150.097500	2012.000000
max	3.000000	5584.000000	20.600000	354.000000	2014.000000

```
#3 task modify the method
```