In [10]:

```python
import pandas as pd

a=[1,7,2]
myvar=pd.Series(a,index=["x","y","z"])
print(myvar)
print(type(myvar))
print(myvar[1])
print(myvar["z"])
print(myvar["y"])
```

```
x    1
y    7
z    2
dtype: int64
<class 'pandas.core.series.Series'>
7
2
7
```

In [13]:

```python
import pandas as pd

a=[1.4,"2",3,4]
myvar=pd.Series(a,index=["x","y","z","e"])
print(myvar)
print(type(myvar))
print(myvar[1])
print(myvar["z"])
print(myvar["y"])
```

```
x    1.4
y      2
z      3
e      4
dtype: object
<class 'pandas.core.series.Series'>
2
3
2
```

In [22]:

```python
import pandas as pd

student_name=["Ram","Shyam","Radha","Geeta","Seeta"]
marks=[50,60,60,80,90]
myvar=pd.Series(marks,index=student_name,name="Students Result")
print(myvar)
```

```
Ram      50
Shyam    60
Radha    60
Geeta    80
Seeta    90
Name: Students Result, dtype: int64
```

In [25]:

```python
import pandas as pd

Calories={'day1':400,'day2':350,'day3':380}
myvar=pd.Series(Calories,index=['day1','day2'])
print(myvar)
```

```
day1    400
day2    350
dtype: int64
```

In [40]:

```python
import pandas as pd

marks={"Ram":55,"Shyam":80,"Radha":80}
```

```
marks_series=pd.Series(marks,name="student result")
print(marks_series)
marks_series.size
marks_series.dtype
marks_series.name
marks_series.index
marks_series.values
marks_series.is_unique
```

```
Ram      55
Shyam    80
Radha    80
Name: student result, dtype: int64
```

Out[40]:
```
False
```

In [67]:
```
import pandas as pd

subs=pd.read_csv("subs.csv",squeeze=True)
print(subs)
print(type(subs))
subs.describe()
subs.min()
subs.max()
subs.median()
subs.sum()
subs[subs<200].size - subs[subs>100].size
```

```
0       48
1       57
2       40
3       43
4       44
      ...
360    231
361    226
362    155
363    144
364    172
Name: Subscribers gained, Length: 365, dtype: int64
<class 'pandas.core.series.Series'>
```

Out[67]:
```
66
```

In [90]:
```
import pandas as pd

mv=pd.read_csv("bollywood.csv",index_col="movie",squeeze=True)
print(mv)
mv.head(10)
mv.tail()
mv[100:200]
mv[-5:]
mv[::2]
mv[::-1]
mv[0]
mv["Uri: The Surgical Strike"]
```

```
movie
Uri: The Surgical Strike                   Vicky Kaushal
Battalion 609                                Vicky Ahuja
The Accidental Prime Minister (film)         Anupam Kher
Why Cheat India                            Emraan Hashmi
Evening Shadows                         Mona Ambegaonkar
                                              ...
Hum Tumhare Hain Sanam                    Shah Rukh Khan
Aankhen (2002 film)                     Amitabh Bachchan
Saathiya (film)                             Vivek Oberoi
Company (film)                                Ajay Devgn
```

```
Awara Paagal Deewana                          Akshay Kumar
Name: lead, Length: 1500, dtype: object
```

Out[90]:

```
'Vicky Kaushal'
```

In [9]:

```python
import pandas as pd

mv=pd.read_csv("bollywood.csv",index_col="movie",squeeze=True)
print(mv)
print("2 States (2014 film)" in mv)
print("Alia Bhatt" in mv.values)
mv.index
```

```
movie
Uri: The Surgical Strike                       Vicky Kaushal
Battalion 609                                    Vicky Ahuja
The Accidental Prime Minister (film)             Anupam Kher
Why Cheat India                                Emraan Hashmi
Evening Shadows                            Mona Ambegaonkar
                                                 ...
Hum Tumhare Hain Sanam                        Shah Rukh Khan
Aankhen (2002 film)                        Amitabh Bachchan
Saathiya (film)                                 Vivek Oberoi
Company (film)                                    Ajay Devgn
Awara Paagal Deewana                          Akshay Kumar
Name: lead, Length: 1500, dtype: object
True
True
```

Out[9]:

```
Index(['Uri: The Surgical Strike', 'Battalion 609',
       'The Accidental Prime Minister (film)', 'Why Cheat India',
       'Evening Shadows', 'Soni (film)', 'Fraud Saiyaan', 'Bombairiya',
       'Manikarnika: The Queen of Jhansi', 'Thackeray (film)',
       ...
       'Raaz (2002 film)', 'Zameen (2003 film)', 'Waisa Bhi Hota Hai Part II',
       'Devdas (2002 Hindi film)', 'Kaante', 'Hum Tumhare Hain Sanam',
       'Aankhen (2002 film)', 'Saathiya (film)', 'Company (film)',
       'Awara Paagal Deewana'],
      dtype='object', name='movie', length=1500)
```

In [19]:

```python
import pandas as pd

mv=pd.read_csv("bollywood.csv",index_col="movie",squeeze=True)
print(mv)
print("2 States (2014 film)" in mv)
print("Alia Bhatt" in mv.values)
mv.value_counts()

num=mv.value_counts()
num[num>20]
```

```
movie
Uri: The Surgical Strike                       Vicky Kaushal
Battalion 609                                    Vicky Ahuja
The Accidental Prime Minister (film)             Anupam Kher
Why Cheat India                                Emraan Hashmi
Evening Shadows                            Mona Ambegaonkar
                                                 ...
Hum Tumhare Hain Sanam                        Shah Rukh Khan
Aankhen (2002 film)                        Amitabh Bachchan
Saathiya (film)                                 Vivek Oberoi
Company (film)                                    Ajay Devgn
Awara Paagal Deewana                          Akshay Kumar
Name: lead, Length: 1500, dtype: object
True
True
```

Out[19]:

```
Akshay Kumar        48
Amitabh Bachchan    45
Ajay Devgn          38
```

```
Salman Khan         31
Sanjay Dutt         26
Shah Rukh Khan      22
Emraan Hashmi       21
Name: lead, dtype: int64
```

In [33]:

```
import pandas as pd
marks={"Ram":55,"Shyam":60,"Radha":70}
m=pd.Series(marks,name='result')
print(m)
```

```
Ram      55
Shyam    60
Radha    70
Name: result, dtype: int64
```

In [35]:

```
dict(m)
```

Out[35]:

```
{'Ram': 55, 'Shyam': 100, 'Radha': 70}
```

In [26]:

```
list(m)
```

Out[26]:

```
[55, 80, 80]
```

In [34]:

```
# m[1]=100
# print(m)
```

```
Ram       55
Shyam    100
Radha     70
Name: result, dtype: int64
```

In [52]:

```
# m=100+m
# print(m)
100+m
m
```

Out[52]:

```
Ram      255
Shyam    300
Radha    270
Name: result, dtype: int64
```

In [54]:

```
m>=100
```

Out[54]:

```
Ram      True
Shyam    True
Radha    True
Name: result, dtype: bool
```

In [61]:

```
import pandas as pd
a=pd.Series([2,4,6,8,10])
b=pd.Series([1,3,5,7,10])
print(a+b)
```

```
0     3
1     7
2    11
3    15
4    20
dtype: int64
```

In [57]:

```
print(a-b)
```

```
0    1
1    1
2    1
3    1
```

```
4    0
dtype: int64
```

In [58]:
```
print(a*b)
```

```
0      2
1     12
2     30
3     56
4    100
dtype: int64
```

In [59]:
```
print(a/b)
```

```
0    2.000000
1    1.333333
2    1.200000
3    1.142857
4    1.000000
dtype: float64
```

In [62]:
```
print(a==b)
```

```
0    False
1    False
2    False
3    False
4     True
dtype: bool
```

In [63]:
```
print(a<b)
```

```
0    False
1    False
2    False
3    False
4    False
dtype: bool
```

In [64]:
```
print(a>b)
```

```
0     True
1     True
2     True
3     True
4    False
dtype: bool
```

DataFrame

In [76]:
```
import pandas as pd
data={'calories':[400,350,380],'duration':[50,40,45]}
df=pd.DataFrame(data,index=['day1','day2','day3'])
print(df)
```

```
      calories  duration
day1       400        50
day2       350        40
day3       380        45
```

In [87]:
```
df.loc['day1']
```

Out[87]:
```
calories    400
duration     50
Name: day1, dtype: int64
```

In [86]:
```
df.loc['day2']
```

Out[86]:

```
calories    350
duration     40
Name: day2, dtype: int64
```

In [88]:

```
df.iloc[0]#for the intergers only
```

Out[88]:

```
calories    400
duration     50
Name: day1, dtype: int64
```

In [90]:

```
# df['calories']
df.calories
```

Out[90]:

```
day1    400
day2    350
day3    380
Name: calories, dtype: int64
```

In [91]:

```
df['duration']
```

Out[91]:

```
day1    50
day2    40
day3    45
Name: duration, dtype: int64
```

In [92]:

```
df.loc[['day1','day2']]
```

Out[92]:

	calories	duration
day1	400	50
day2	350	40

In [94]:

```
import pandas as pd

data=pd.read_csv('auto-mpg.csv')
print(data)
```

```
      mpg  cylinders  displacement horsepower  weight  acceleration  \
0    18.0          8         307.0        130    3504          12.0   
1    15.0          8         350.0        165    3693          11.5   
2    18.0          8         318.0        150    3436          11.0   
3    16.0          8         304.0        150    3433          12.0   
4    17.0          8         302.0        140    3449          10.5   
..    ...        ...           ...        ...     ...           ...   
393  27.0          4         140.0         86    2790          15.6   
394  44.0          4          97.0         52    2130          24.6   
395  32.0          4         135.0         84    2295          11.6   
396  28.0          4         120.0         79    2625          18.6   
397  31.0          4         119.0         82    2720          19.4   

     model year  origin                   car name  
0            70       1  chevrolet chevelle malibu  
1            70       1          buick skylark 320  
2            70       1         plymouth satellite  
3            70       1              amc rebel sst  
4            70       1                ford torino  
..          ...     ...                        ...  
393          82       1            ford mustang gl  
394          82       2                  vw pickup  
395          82       1              dodge rampage  
396          82       1                ford ranger  
397          82       1                 chevy s-10  

[398 rows x 9 columns]
```

In [96]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 398 entries, 0 to 397
Data columns (total 9 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   mpg           398 non-null    float64
 1   cylinders     398 non-null    int64  
 2   displacement  398 non-null    float64
 3   horsepower    398 non-null    object 
 4   weight        398 non-null    int64  
 5   acceleration  398 non-null    float64
 6   model year    398 non-null    int64  
 7   origin        398 non-null    int64  
 8   car name      398 non-null    object 
dtypes: float64(3), int64(4), object(2)
memory usage: 28.1+ KB
```

In [101…

```
data.head(10)
```

Out[101…

	mpg	cylinders	displacement	horsepower	weight	acceleration	model year	origin	car name
0	18.0	8	307.0	130	3504	12.0	70	1	chevrolet chevelle malibu
1	15.0	8	350.0	165	3693	11.5	70	1	buick skylark 320
2	18.0	8	318.0	150	3436	11.0	70	1	plymouth satellite
3	16.0	8	304.0	150	3433	12.0	70	1	amc rebel sst
4	17.0	8	302.0	140	3449	10.5	70	1	ford torino
5	15.0	8	429.0	198	4341	10.0	70	1	ford galaxie 500
6	14.0	8	454.0	220	4354	9.0	70	1	chevrolet impala
7	14.0	8	440.0	215	4312	8.5	70	1	plymouth fury iii
8	14.0	8	455.0	225	4425	10.0	70	1	pontiac catalina
9	15.0	8	390.0	190	3850	8.5	70	1	amc ambassador dpl

In [102…

```
data.loc[15]
```

Out[102…

```
mpg                           22
cylinders                      6
displacement                 198
horsepower                    95
weight                      2833
acceleration                15.5
model year                    70
origin                         1
car name         plymouth duster
Name: 15, dtype: object
```

In [103…

```
data.loc[5:15]
```

Out[103…

	mpg	cylinders	displacement	horsepower	weight	acceleration	model year	origin	car name
5	15.0	8	429.0	198	4341	10.0	70	1	ford galaxie 500
6	14.0	8	454.0	220	4354	9.0	70	1	chevrolet impala
7	14.0	8	440.0	215	4312	8.5	70	1	plymouth fury iii
8	14.0	8	455.0	225	4425	10.0	70	1	pontiac catalina
9	15.0	8	390.0	190	3850	8.5	70	1	amc ambassador dpl
10	15.0	8	383.0	170	3563	10.0	70	1	dodge challenger se
11	14.0	8	340.0	160	3609	8.0	70	1	plymouth 'cuda 340
12	15.0	8	400.0	150	3761	9.5	70	1	chevrolet monte carlo
13	14.0	8	455.0	225	3086	10.0	70	1	buick estate wagon (sw)
14	24.0	4	113.0	95	2372	15.0	70	3	toyota corona mark ii
15	22.0	6	198.0	95	2833	15.5	70	1	plymouth duster

In [104…

```
data.iloc[5:15]
```

Out[104…

	mpg	cylinders	displacement	horsepower	weight	acceleration	model year	origin	car name
5	15.0	8	429.0	198	4341	10.0	70	1	ford galaxie 500
6	14.0	8	454.0	220	4354	9.0	70	1	chevrolet impala
7	14.0	8	440.0	215	4312	8.5	70	1	plymouth fury iii
8	14.0	8	455.0	225	4425	10.0	70	1	pontiac catalina
9	15.0	8	390.0	190	3850	8.5	70	1	amc ambassador dpl
10	15.0	8	383.0	170	3563	10.0	70	1	dodge challenger se

	mpg	cylinders	displacement	horsepower	weight	acceleration	model year	origin	car name
11	14.0	8	340.0	160	3609	8.0	70	1	plymouth 'cuda 340
12	15.0	8	400.0	150	3761	9.5	70	1	chevrolet monte carlo
13	14.0	8	455.0	225	3086	10.0	70	1	buick estate wagon (sw)
14	24.0	4	113.0	95	2372	15.0	70	3	toyota corona mark ii

In [107…

```
data['mpg'].loc[4]
```

Out[107…

```
17.0
```

In [114…

```
    data.shape[1]
```

Out[114…

```
9
```

In [115…

```
data.describe()
```

Out[115…

	mpg	cylinders	displacement	weight	acceleration	model year	origin
count	398.000000	398.000000	398.000000	398.000000	398.000000	398.000000	398.000000
mean	23.514573	5.454774	193.425879	2970.424623	15.568090	76.010050	1.572864
std	7.815984	1.701004	104.269838	846.841774	2.757689	3.697627	0.802055
min	9.000000	3.000000	68.000000	1613.000000	8.000000	70.000000	1.000000
25%	17.500000	4.000000	104.250000	2223.750000	13.825000	73.000000	1.000000
50%	23.000000	4.000000	148.500000	2803.500000	15.500000	76.000000	1.000000
75%	29.000000	8.000000	262.000000	3608.000000	17.175000	79.000000	2.000000
max	46.600000	8.000000	455.000000	5140.000000	24.800000	82.000000	3.000000

In [118…

```
# data.describe(include='all')
import numpy as np
data.describe(include=[np.number])
```

Out[118…

	mpg	cylinders	displacement	weight	acceleration	model year	origin
count	398.000000	398.000000	398.000000	398.000000	398.000000	398.000000	398.000000
mean	23.514573	5.454774	193.425879	2970.424623	15.568090	76.010050	1.572864
std	7.815984	1.701004	104.269838	846.841774	2.757689	3.697627	0.802055
min	9.000000	3.000000	68.000000	1613.000000	8.000000	70.000000	1.000000
25%	17.500000	4.000000	104.250000	2223.750000	13.825000	73.000000	1.000000
50%	23.000000	4.000000	148.500000	2803.500000	15.500000	76.000000	1.000000
75%	29.000000	8.000000	262.000000	3608.000000	17.175000	79.000000	2.000000
max	46.600000	8.000000	455.000000	5140.000000	24.800000	82.000000	3.000000

In [124…

```
data.describe(exclude=[np.number])
```

Out[124…

	horsepower	car name
count	398	398
unique	94	305
top	150	ford pinto
freq	22	6

In [129…

```
data.describe(percentiles=[.30,.45,.60])
```

Out[129…

	mpg	cylinders	displacement	weight	acceleration	model year	origin
count	398.000000	398.000000	398.000000	398.000000	398.000000	398.000000	398.000000
mean	23.514573	5.454774	193.425879	2970.424623	15.568090	76.010050	1.572864
std	7.815984	1.701004	104.269838	846.841774	2.757689	3.697627	0.802055
min	9.000000	3.000000	68.000000	1613.000000	8.000000	70.000000	1.000000
30%	18.000000	4.000000	112.000000	2301.000000	14.200000	73.000000	1.000000
45%	21.065000	4.000000	140.000000	2670.650000	15.000000	75.000000	1.000000
50%	23.000000	4.000000	148.500000	2803.500000	15.500000	76.000000	1.000000
60%	25.000000	6.000000	200.000000	3085.200000	16.000000	77.000000	1.000000
max	46.600000	8.000000	455.000000	5140.000000	24.800000	82.000000	3.000000

In [134…

```
data[['mpg','cylinders']].describe()
```

Out[134…

	mpg	cylinders
count	398.000000	398.000000
mean	23.514573	5.454774
std	7.815984	1.701004
min	9.000000	3.000000
25%	17.500000	4.000000
50%	23.000000	4.000000
75%	29.000000	8.000000
max	46.600000	8.000000

In [135…

```
data[0:2].describe()
```

Out[135…

	mpg	cylinders	displacement	weight	acceleration	model year	origin
count	2.00000	2.0	2.000000	2.000000	2.000000	2.0	2.0
mean	16.50000	8.0	328.500000	3598.500000	11.750000	70.0	1.0
std	2.12132	0.0	30.405592	133.643182	0.353553	0.0	0.0
min	15.00000	8.0	307.000000	3504.000000	11.500000	70.0	1.0

	mpg	cylinders	displacement	weight	acceleration	model year	origin
25%	15.75000	8.0	317.750000	3551.250000	11.625000	70.0	1.0
50%	16.50000	8.0	328.500000	3598.500000	11.750000	70.0	1.0
75%	17.25000	8.0	339.250000	3645.750000	11.875000	70.0	1.0
max	18.00000	8.0	350.000000	3693.000000	12.000000	70.0	1.0

In [136…

```
data.loc[0:2].describe()
```

Out[136…

	mpg	cylinders	displacement	weight	acceleration	model year	origin
count	3.000000	3.0	3.000000	3.000000	3.00	3.0	3.0
mean	17.000000	8.0	325.000000	3544.333333	11.50	70.0	1.0
std	1.732051	0.0	22.338308	133.162808	0.50	0.0	0.0
min	15.000000	8.0	307.000000	3436.000000	11.00	70.0	1.0
25%	16.500000	8.0	312.500000	3470.000000	11.25	70.0	1.0
50%	18.000000	8.0	318.000000	3504.000000	11.50	70.0	1.0
75%	18.000000	8.0	334.000000	3598.500000	11.75	70.0	1.0
max	18.000000	8.0	350.000000	3693.000000	12.00	70.0	1.0

In [137…

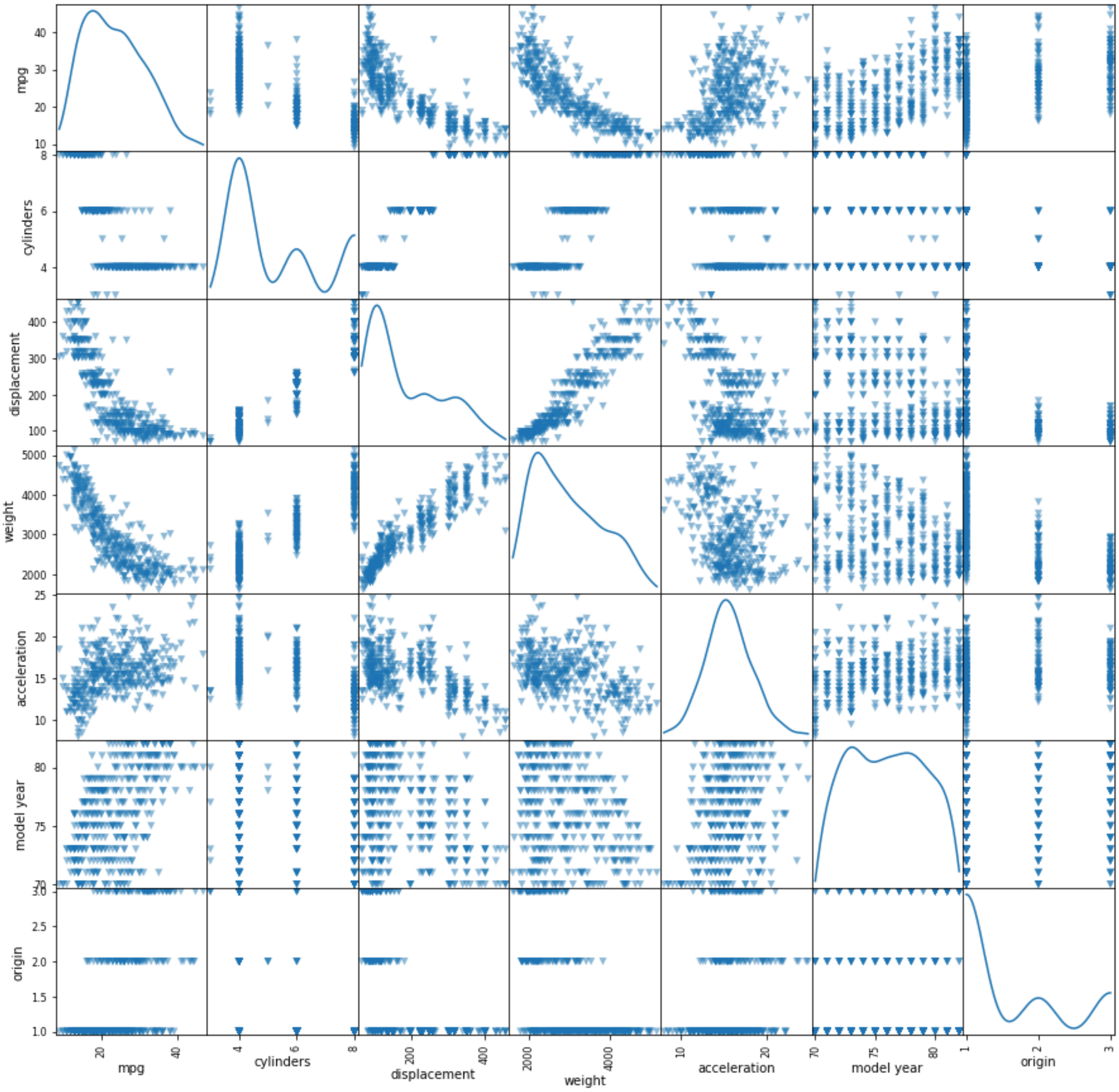
```
data.corr()
```

Out[137…

	mpg	cylinders	displacement	weight	acceleration	model year	origin
mpg	1.000000	-0.775396	-0.804203	-0.831741	0.420289	0.579267	0.563450
cylinders	-0.775396	1.000000	0.950721	0.896017	-0.505419	-0.348746	-0.562543
displacement	-0.804203	0.950721	1.000000	0.932824	-0.543684	-0.370164	-0.609409
weight	-0.831741	0.896017	0.932824	1.000000	-0.417457	-0.306564	-0.581024
acceleration	0.420289	-0.505419	-0.543684	-0.417457	1.000000	0.288137	0.205873
model year	0.579267	-0.348746	-0.370164	-0.306564	0.288137	1.000000	0.180662
origin	0.563450	-0.562543	-0.609409	-0.581024	0.205873	0.180662	1.000000

In [147…

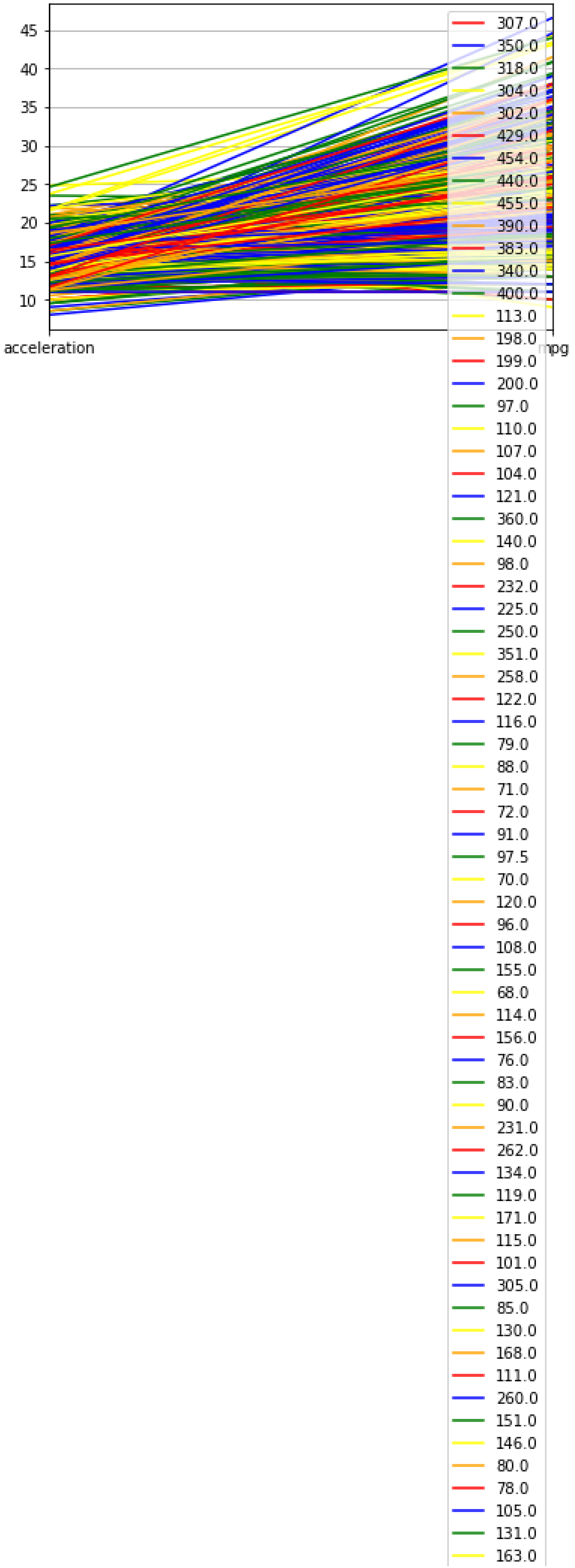
```
import pandas as pd
import matplotlib.pyplot as plt
pd.plotting.scatter_matrix(data,figsize=[15,15],marker='v',alpha=0.5,diagonal='kde')
plt.show()
```

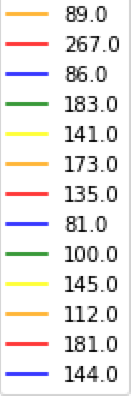


In [158…

```
import pandas as pd
from pandas.plotting import parallel_coordinates

pll=parallel_coordinates(data,'displacement',cols=['acceleration','mpg'],color=['red
```

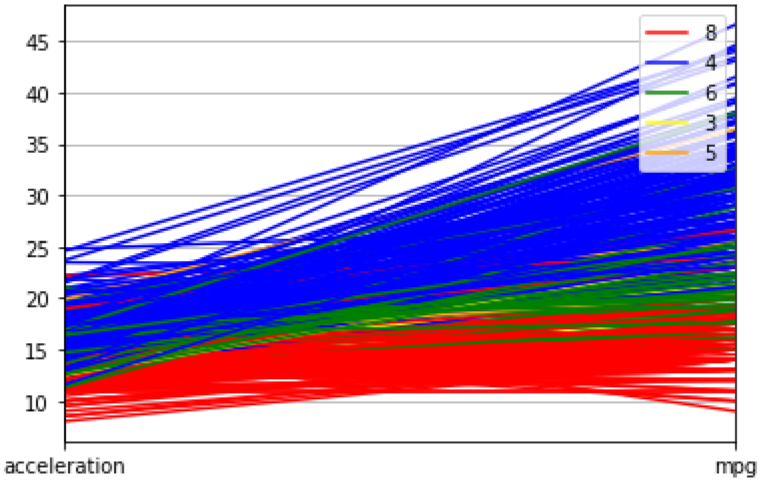

45
40
35
30
25
20
15
10
acceleration
mpg
307.0
350.0
318.0
304.0
302.0
429.0
454.0
440.0
455.0
390.0
383.0
340.0
400.0
113.0
198.0
199.0
200.0
97.0
110.0
107.0
104.0
121.0
360.0
140.0
98.0
232.0
225.0
250.0
351.0
258.0
122.0
116.0
79.0
88.0
71.0
72.0
91.0
97.5
70.0
120.0
96.0
108.0
155.0
68.0
114.0
156.0
76.0
83.0
90.0
231.0
262.0
134.0
119.0
171.0
115.0
101.0
305.0
85.0
130.0
168.0
111.0
260.0
151.0
146.0
80.0
78.0
105.0
131.0
163.0



In [159…

```
import pandas as pd
from pandas.plotting import parallel_coordinates

pll=parallel_coordinates(data,'cylinders',cols=['acceleration','mpg'],color=['red','
```



cross tabulation

In [6]:

```
    import pandas as pd
    data=pd.read_csv('auto-mpg.csv')
    pd.crosstab(data['cylinders'],data['model year'],rownames=['cylinders'],colnames
```

Out[6]:

model year	70	71	72	73	74	75	76	77	78	79	80	81	82
cylinders													
3	0	0	1	1	0	0	0	1	0	0	1	0	0
4	7	13	14	11	15	12	15	14	17	12	25	21	28
5	0	0	0	0	0	0	0	0	1	1	1	0	0
6	4	8	0	8	7	12	10	5	12	6	2	7	3
8	18	7	13	20	5	6	9	8	6	10	0	1	0

Data Cleaning

In [13]:

```
import numpy as np
import pandas as pd
sale_data=pd.DataFrame({"Name":["klillian","Emma","Sofia","Markus","Edward","Thomas"
                      "region":[np.nan,"North","East",np.nan,"West","West","South",n
```

```
                             "Sales":[5000,52000,np.nan,np.nan,42000,72000,49000,np.nan,670
                             "Expenses":[42000,43000,np.nan,np.nan,38000,39000,42000,np.nan
print(sale_data)
```

```
        Name region    Sales  Expenses
0   klillian    NaN   5000.0   42000.0
1       Emma  North  52000.0   43000.0
2      Sofia   East      NaN       NaN
3     Markus    NaN      NaN       NaN
4     Edward   West  42000.0   38000.0
5     Thomas   West  72000.0   39000.0
6      Etham  South  49000.0   42000.0
7        NaN    NaN      NaN       NaN
8       Arun   West  67000.0   39000.0
9     Anika   East  66000.0   50000.0
10       Paw  south  67000.0   45000.0
```

Out[13]:

```
Name        1
region      3
Sales       3
Expenses    3
dtype: int64
```

In [14]:

```
sale_data.isna().sum()
```

Out[14]:

```
Name        1
region      3
Sales       3
Expenses    3
dtype: int64
```

In [15]:

```
sale_data.dropna()
```

Out[15]:

	Name	region	Sales	Expenses
1	Emma	North	52000.0	43000.0
4	Edward	West	42000.0	38000.0
5	Thomas	West	72000.0	39000.0
6	Etham	South	49000.0	42000.0
8	Arun	West	67000.0	39000.0
9	Anika	East	66000.0	50000.0
10	Paw	south	67000.0	45000.0

In [17]:

```
sale_data
```

Out[17]:

	Name	region	Sales	Expenses
0	klillian	NaN	5000.0	42000.0
1	Emma	North	52000.0	43000.0
2	Sofia	East	NaN	NaN
3	Markus	NaN	NaN	NaN
4	Edward	West	42000.0	38000.0
5	Thomas	West	72000.0	39000.0
6	Etham	South	49000.0	42000.0
7	NaN	NaN	NaN	NaN

```
      Name  region    Sales  Expenses
 8    Arun    West  67000.0   39000.0
 9   Anika    East  66000.0   50000.0
10     Paw   south  67000.0   45000.0
```

```
In [20]: sale_data.dropna(thresh=1)
```

```
Out[20]:
        Name  region    Sales  Expenses
 0  klillian     NaN   5000.0   42000.0
 1      Emma   North  52000.0   43000.0
 2     Sofia    East      NaN       NaN
 3    Markus     NaN      NaN       NaN
 4    Edward    West  42000.0   38000.0
 5    Thomas    West  72000.0   39000.0
 6    Etham    South  49000.0   42000.0
 8      Arun    West  67000.0   39000.0
 9     Anika    East  66000.0   50000.0
10       Paw   south  67000.0   45000.0
```

```
In [21]: sale_data.dropna(how='any')
```

```
Out[21]:
        Name  region    Sales  Expenses
 1      Emma   North  52000.0   43000.0
 4    Edward    West  42000.0   38000.0
 5    Thomas    West  72000.0   39000.0
 6     Etham   South  49000.0   42000.0
 8      Arun    West  67000.0   39000.0
 9     Anika    East  66000.0   50000.0
10       Paw   south  67000.0   45000.0
```

```
In [22]: sale_data.dropna(how='all')
```

```
Out[22]:
        Name  region    Sales  Expenses
 0  klillian     NaN   5000.0   42000.0
 1      Emma   North  52000.0   43000.0
 2     Sofia    East      NaN       NaN
 3    Markus     NaN      NaN       NaN
 4    Edward    West  42000.0   38000.0
 5    Thomas    West  72000.0   39000.0
 6     Etham   South  49000.0   42000.0
```

	Name	region	Sales	Expenses
8	Arun	West	67000.0	39000.0
9	Anika	East	66000.0	50000.0
10	Paw	south	67000.0	45000.0

In [28]:
```
sale_data.dropna(subset['Sales','Expenses'])
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-28-0e96d5ecb300> in <module>
----> 1 sale_data.dropna(subset['Sales','Expenses'])

NameError: name 'subset' is not defined
```

In [29]:
```
sale_data.dropna(axis=1)
```

Out[29]:

0
1
2
3
4
5
6
7
8
9
10

In [30]:
```
sale_data
```

Out[30]:

	Name	region	Sales	Expenses
0	klillian	NaN	5000.0	42000.0
1	Emma	North	52000.0	43000.0
2	Sofia	East	NaN	NaN
3	Markus	NaN	NaN	NaN
4	Edward	West	42000.0	38000.0
5	Thomas	West	72000.0	39000.0
6	Etham	South	49000.0	42000.0
7	NaN	NaN	NaN	NaN
8	Arun	West	67000.0	39000.0
9	Anika	East	66000.0	50000.0
10	Paw	south	67000.0	45000.0

In [35]:

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

Out[35]:

	Name	region	Sales	Expenses
1	Emma	North	52000.0	43000.0
4	Edward	West	42000.0	38000.0
5	Thomas	West	72000.0	39000.0
6	Etham	South	49000.0	42000.0
8	Arun	West	67000.0	39000.0
9	Anika	East	66000.0	50000.0
10	Paw	south	67000.0	45000.0

In [36]:

```
sale_data.dropna(thresh=1,inplace=True)
sale_data
```

Out[36]:

	Name	region	Sales	Expenses
1	Emma	North	52000.0	43000.0
4	Edward	West	42000.0	38000.0
5	Thomas	West	72000.0	39000.0
6	Etham	South	49000.0	42000.0
8	Arun	West	67000.0	39000.0
9	Anika	East	66000.0	50000.0
10	Paw	south	67000.0	45000.0

In [37]:

```
import numpy as np
import pandas as pd
sale_data=pd.DataFrame({"Name":["klillian","Emma","Sofia","Markus","Edward","Thomas"
                      "region":[np.nan,"North","East",np.nan,"West","West","South",n
                      "Sales":[5000,52000,np.nan,np.nan,42000,72000,49000,np.nan,670
                      "Expenses":[42000,43000,np.nan,np.nan,38000,39000,42000,np.nan
print(sale_data)
```

```
        Name region    Sales  Expenses
0   klillian    NaN   5000.0   42000.0
1       Emma  North  52000.0   43000.0
2      Sofia   East      NaN       NaN
3     Markus    NaN      NaN       NaN
4     Edward   West  42000.0   38000.0
5     Thomas   West  72000.0   39000.0
6      Etham  South  49000.0   42000.0
7        NaN    NaN      NaN       NaN
8       Arun   West  67000.0   39000.0
9      Anika   East  66000.0   50000.0
10       Paw  south  67000.0   45000.0
```

In [40]:

```
sale_data.fillna(0)
```

Out[40]:

	Name	region	Sales	Expenses
0	klillian	0	5000.0	42000.0
1	Emma	North	52000.0	43000.0

	Name	region	Sales	Expenses
2	Sofia	East	0.0	0.0
3	Markus	0	0.0	0.0
4	Edward	West	42000.0	38000.0
5	Thomas	West	72000.0	39000.0
6	Etham	South	49000.0	42000.0
7	0	0	0.0	0.0
8	Arun	West	67000.0	39000.0
9	Anika	East	66000.0	50000.0
10	Paw	south	67000.0	45000.0

In [42]:

```
sale_data['Sales'].fillna(4000)
```

Out[42]:

```
0      5000.0
1     52000.0
2      4000.0
3      4000.0
4     42000.0
5     72000.0
6     49000.0
7      4000.0
8     67000.0
9     66000.0
10    67000.0
Name: Sales, dtype: float64
```

In [45]:

```
sale_data['Sales'].fillna(sale_data['Sales'].mean())
```

Out[45]:

```
0      5000.0
1     52000.0
2     52500.0
3     52500.0
4     42000.0
5     72000.0
6     49000.0
7     52500.0
8     67000.0
9     66000.0
10    67000.0
Name: Sales, dtype: float64
```

In [46]:

```
sale_data['Sales'].fillna(sale_data['Sales'].median())
```

Out[46]:

```
0      5000.0
1     52000.0
2     59000.0
3     59000.0
4     42000.0
5     72000.0
6     49000.0
7     59000.0
8     67000.0
9     66000.0
10    67000.0
Name: Sales, dtype: float64
```

In [51]:

```
import pandas as pd
data=pd.read_csv('auto-mpg.csv')
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 398 entries, 0 to 397
Data columns (total 9 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   mpg           398 non-null    float64
 1   cylinders     398 non-null    int64  
 2   displacement  398 non-null    float64
 3   horsepower    398 non-null    object 
 4   weight        398 non-null    int64  
 5   acceleration  398 non-null    float64
 6   model year    398 non-null    int64  
 7   origin        398 non-null    int64  
 8   car name      398 non-null    object 
dtypes: float64(3), int64(4), object(2)
memory usage: 28.1+ KB
```

In [53]:

```
data[data['horsepower']=='?']
```

Out[53]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	model year	origin	car name
32	25.0	4	98.0	?	2046	19.0	71	1	ford pinto
126	21.0	6	200.0	?	2875	17.0	74	1	ford maverick
330	40.9	4	85.0	?	1835	17.3	80	2	renault lecar deluxe
336	23.6	4	140.0	?	2905	14.3	80	1	ford mustang cobra
354	34.5	4	100.0	?	2320	15.8	81	2	renault 18i
374	23.0	4	151.0	?	3035	20.5	82	1	amc concord dl

In [58]:

```
data[data['horsepower']!='?']
```

Out[58]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	model year	origin	car name
0	18.0	8	307.0	130	3504	12.0	70	1	chevrolet chevelle malibu
1	15.0	8	350.0	165	3693	11.5	70	1	buick skylark 320
2	18.0	8	318.0	150	3436	11.0	70	1	plymouth satellite
3	16.0	8	304.0	150	3433	12.0	70	1	amc rebel sst
4	17.0	8	302.0	140	3449	10.5	70	1	ford torino
...	...	...	...	...	...	...	...	...	...
393	27.0	4	140.0	86	2790	15.6	82	1	ford mustang

	mpg	cylinders	displacement	horsepower	weight	acceleration	model year	origin	car name
									gl
394	44.0	4	97.0	52	2130	24.6	82	2	vw pickup
395	32.0	4	135.0	84	2295	11.6	82	1	dodge rampage
396	28.0	4	120.0	79	2625	18.6	82	1	ford ranger
397	31.0	4	119.0	82	2720	19.4	82	1	chevy s-10

392 rows × 9 columns

In [57]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 398 entries, 0 to 397
Data columns (total 9 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   mpg           398 non-null    float64
 1   cylinders     398 non-null    int64  
 2   displacement  398 non-null    float64
 3   horsepower    398 non-null    object 
 4   weight        398 non-null    int64  
 5   acceleration  398 non-null    float64
 6   model year    398 non-null    int64  
 7   origin        398 non-null    int64  
 8   car name      398 non-null    object 
dtypes: float64(3), int64(4), object(2)
memory usage: 28.1+ KB
```

In [52]:

```
import pandas as pd
data=pd.read_csv('auto-mpg.csv')
data.tail()
```

Out[52]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	model year	origin	car name
393	27.0	4	140.0	86	2790	15.6	82	1	ford mustang gl
394	44.0	4	97.0	52	2130	24.6	82	2	vw pickup
395	32.0	4	135.0	84	2295	11.6	82	1	dodge rampage
396	28.0	4	120.0	79	2625	18.6	82	1	ford ranger
397	31.0	4	119.0	82	2720	19.4	82	1	chevy s-10

In [ ]:

```
data.drop('mpg',axis=1)
```

In [ ]:

```
import pandas as pd
dataset=pd.read_csv('auto-mpg.csv')
def find_outliers(ds,col):
    quart1=ds[col].quabtile(0.25)
    quart2=
```