# Chap.-1 Python Pandas

- **import pandas as pd**
- **analyzing, cleaning, exploring, and manipulating data.**

## Series : list, tuple, str, dict

- **Set is not allowed in Series**

In [3]:

```python
import pandas as pd

l=[10,20,30]
s = pd.Series(l)
print(s)
```

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

In [113]:

```python
import pandas

dir(pandas)
dir(help(pandas))
```

```
Help on package pandas:

NAME
    pandas

DESCRIPTION
    pandas - a powerful data analysis and manipulation library for Python
    =====================================================================

    **pandas** is a Python package providing fast, flexible, and expressive data
    structures designed to make working with "relational" or "labeled" data both
    easy and intuitive. It aims to be the fundamental high-level building block for
    doing practical, **real world** data analysis in Python. Additionally, it has
    the broader goal of becoming **the most powerful and flexible open source data
    analysis / manipulation tool available in any language**. It is already well on
    its way toward this goal.

    Main Features
    -------------
```

In [6]:

```python
l=[10,20,'30']
s = pd.Series(l)
print(s)
```

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

- **dtype : int64 -> every var is int**
- **dtype : object -> any str form**

## Datatypes in pandas

1. int
2. object -> more than 1 data type
3. boolean
4. float -> when only int & float final will be float
5. time

In [7]:

```
l=(10,20,30)
s = pd.Series(l)
print(s)
```

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

In [9]:

```
l={10,20,30}
s = pd.Series(l)
print(s)

# TypeError: 'set' type is unordered
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-9-8b3620cd99e4> in <module>
      1 l={10,20,30}
----> 2 s = pd.Series(l)
      3 print(s)

C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\series.py in __init__(self, data, index, dtype, name, copy, fa
stpath)
    297                 pass
    298             elif isinstance(data, (set, frozenset)):
--> 299                 raise TypeError(f"'{type(data).__name__}' type is unordered")
    300             else:
    301                 data = com.maybe_iterable_to_list(data)

TypeError: 'set' type is unordered
```

In [10]:

```
l='Python'
s = pd.Series(l)
print(s)
```

```
0    Python
dtype: object
```

In [11]:

```
l={'A':10,'B':20}
s = pd.Series(l)
print(s)
```

```
A    10
B    20
dtype: int64
```

- ### Without Dict

In [12]:

```
l=[10,20,30]
s = pd.Series(l,index=['A','B','C'])
print(s)
```

```
A    10
B    20
C    30
dtype: int64
```

In [14]:

```
l=[10,20]
s = pd.Series(l,index=['A','B','C'])
print(s)

# ValueError: Length of passed values is 2, index implies 3.
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-14-d31ebb60a2aa> in <module>
      1 l=[10,20]
----> 2 s = pd.Series(l,index=['A','B','C'])
      3 print(s)
      4 
      5 # ValueError: Length of passed values is 2, index implies 3.

C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\series.py in __init__(self, data, index, dtype, name, copy, fa
stpath)
    311                 try:
    312                     if len(index) != len(data):
--> 313                         raise ValueError(
    314                             f"Length of passed values is {len(data)}, "
    315                             f"index implies {len(index)}."

ValueError: Length of passed values is 2, index implies 3.
```

In [3]:

```
import pandas as pd
l={'A':[10,20],'B':20,'C':'pd'}
s = pd.Series(l,index=['A','B','C'])
print(s)
```

```
A    [10, 20]
B          20
C          pd
dtype: object
```

In [2]:

```
import pandas as pd
l={'A':[10,20],'B':20,'C':'pd'}
s = pd.Series(l,index=['A','B','C','D'])
print(s)
```

```
A    [10, 20]
B          20
C          pd
D         NaN
dtype: object
```

In [16]:

```
l={'A':10,'B':20, 'C':1.1}
s = pd.Series(l)
print(s)
```

```
A    10.0
B    20.0
C     1.1
dtype: float64
```

In [17]:

```
l={'A':[10,20],'B':20,'C':True}
s = pd.Series(l,index=['A','B','C'])
print(s)
```

```
A    [10, 20]
B          20
C        True
dtype: object
```

- **Indexing, Slicing & Label are applicable in Series.**

In [15]:

```
l={'A':[10,15],'B':20,'C':True}
s = pd.Series(l,index=['A','B','C'])
print(s[1])
print(s['B'])
print(s[0][1])
print(s[2])
print("---------------")
print(s[0:2])
print("---------------")
print(s['A':'C'])
```

```
20
20
15
True
---------------
A    [10, 15]
B          20
dtype: object
---------------
A    [10, 15]
B          20
C        True
dtype: object
```

# DataFrame - It's a tabular Structure(2D)

1. List
2. Dict
3. Set
4. tuple
5. Array

In [24]:

```
l=[10,20,30]
df = pd.DataFrame(l)
print(df)
```

```
    0
0  10
1  20
2  30
```

In [23]:

```
# L=[10,20,30]
l = [[[10,50,60],[50,90,51]],[[10,50,60],[50,90,51]]]
df = pd.DataFrame(l)
print(df)
```

```
              0             1
0  [10, 50, 60]  [50, 90, 51]
1  [10, 50, 60]  [50, 90, 51]
```

In [20]:

```
s = {10,20,30,90,30,80}
sr = pd.DataFrame(s)
print(sr)
```

```
    0
0  10
1  80
2  20
3  90
4  30
```

In [26]:

```
l={'A':[10,20],'B':[100,200],'C':[1000,2000]}
df = pd.DataFrame(l)
print(df)
```

```
    A    B     C
0  10  100  1000
1  20  200  2000
```

In [24]:

```
l={'A':[10,20],'B':[100,200],'C':[1000,2000]}
s = pd.Series(l)
print(s)
```

```
A        [10, 20]
B      [100, 200]
C    [1000, 2000]
dtype: object
```

- ## Shape must be same

In [28]:

```
l={'A':[10,20],'B':[100],'C':[1000,2000]}
df = pd.DataFrame(l)
print(df)

# ValueError: arrays must all be same length
```

In [30]:

```
l={'A':[10,20],'B':[100,200],'C':[1000,2000]}
df = pd.DataFrame(l, index=['Day1','Day2'])
print(df)
```

```
       A    B     C
Day1  10  100  1000
Day2  20  200  2000
```

In [44]:

```
import pandas as pd

l = {'mpg':[18.0,15.0,17.0,27.0,31.0],
    'Cylinders':[8,8,8,4,4],
    'hp':[130,165,150,84,79],
    'weight':[3504,3693,3436,3433,2130],
    'mYear': [70,82,70,70,82],
    'Origin':[1,1,1,2,2]}
df = pd.DataFrame(l)
print(df)
```

```
    mpg  Cylinders   hp  weight  mYear  Origin
0  18.0          8  130    3504     70       1
1  15.0          8  165    3693     82       1
2  17.0          8  150    3436     70       1
3  27.0          4   84    3433     70       2
4  31.0          4   79    2130     82       2
```

## For Filteration

In [45]:

```
import pandas as pd

l = {'mpg':[18.0,15.0,17.0,27.0,31.0],
    'Cylinders':[8,8,8,4,4],
    'hp':[130,165,150,84,79],
    'weight':[3504,3693,3436,3433,2130],
    'mYear': [70,82,70,70,82],
    'Origin':[1,1,1,2,2]}
df = pd.DataFrame(l,columns=['Cylinders','hp'])
print(df)
```

```
   Cylinders   hp
0          8  130
1          8  165
2          8  150
3          4   84
4          4   79
```

In [51]:

```
l = [10,20,30,40]
s = pd.Series(l)
print(s)
print('-------------')
print('s[2] ->',s[2])
print('s[1] ->',s[1])

```

```
0    10
1    20
2    30
3    40
dtype: int64
-------------
s[2] -> 30
s[1] -> 20
```

In [72]:

```
l = [10,20,30,40]
df = pd.DataFrame(l)
print(df)

print('df[2] ->',df.[1])
```

```
  File "<ipython-input-72-56b162dd42cb>", line 5
    print('df[2] ->',df.[1])
                        ^
SyntaxError: invalid syntax
```

In [74]:

```
l = [10,20,30,40]
df = pd.DataFrame(l)
print(df)
print("-------------------------")
print('df[2] ->',df.loc[1])
```

```
    0
0  10
1  20
2  30
3  40
-------------------------
df[2] -> 0    20
Name: 1, dtype: int64
```

In [63]:

```
import pandas as pd

l = {'mpg':[18.0,15.0,17.0,27.0,31.0],
    'Cylinders':[8,8,8,4,4],
    'hp':[130,165,150,84,79],
    'weight':[3504,3693,3436,3433,2130],
    'mYear': [70,82,70,70,82],
    'Origin':[1,1,1,2,2]}
df = pd.DataFrame(l,columns=['Cylinders','hp'])
print(df)

print(df['mpg'])
# KeyError: 'mpg'
```

In [68]:

```python
import pandas as pd

l = {'mpg':[18.0,15.0,17.0,27.0,31.0],
    'Cylinders':[8,8,8,4,4],
    'hp':[130,165,150,84,79],
    'weight':[3504,3693,3436,3433,2130],
    'mYear': [70,82,70,70,82],
    'Origin':[1,1,1,2,2]}
df = pd.DataFrame(l,columns=['Cylinders','hp'])
print(df)
print('-----------------------')
print(df.loc[0])
```

```
   Cylinders   hp
0          8  130
1          8  165
2          8  150
3          4   84
4          4   79
-----------------------
Cylinders      8
hp           130
Name: 0, dtype: int64
```

In [88]:

```python
import pandas as pd

l = {'mpg':[18.0,15.0,17.0,27.0,31.0],
    'Cylinders':[8,8,8,4,4],
    'hp':[130,165,150,84,79],
    'weight':[3504,3693,3436,3433,2130],
    'mYear': [70,82,70,70,82],
    'Origin':[1,1,1,2,2]}
df = pd.DataFrame(l,columns=['Cylinders','hp'])
print(df)
print('-----------------------')
print(df.loc[0]['hp'])
```

```
   Cylinders   hp
0          8  130
1          8  165
2          8  150
3          4   84
4          4   79
-----------------------
130
```

# loc - label based

In [27]:

```python
import pandas as pd

l = {'mpg':[18.0,15.0,17.0,27.0,31.0],
    'Cylinders':[8,8,8,4,4],
    'hp':[130,165,150,84,79],
    'weight':[3504,3693,3436,3433,2130],
    'mYear': [70,82,70,70,82],
    'Origin':[1,1,1,2,2]}
df = pd.DataFrame(l)
print(df)
print('\n-------------------df.loc[0:3]---------------------\n')
print(df.loc[0:3])
```

```
    mpg  Cylinders   hp  weight  mYear  Origin
0  18.0          8  130    3504     70       1
1  15.0          8  165    3693     82       1
2  17.0          8  150    3436     70       1
3  27.0          4   84    3433     70       2
4  31.0          4   79    2130     82       2

-------------------df.loc[0:3]---------------------

    mpg  Cylinders   hp  weight  mYear  Origin
0  18.0          8  130    3504     70       1
1  15.0          8  165    3693     82       1
2  17.0          8  150    3436     70       1
3  27.0          4   84    3433     70       2
```

# iloc - that follows concept of simple indexing

## (Locates on base of index & slicing)

In [28]:

```
import pandas as pd

l = {'mpg':[18.0,15.0,17.0,27.0,31.0],
    'Cylinders':[8,8,8,4,4],
    'hp':[130,165,150,84,79],
    'weight':[3504,3693,3436,3433,2130],
    'mYear': [70,82,70,70,82],
    'Origin':[1,1,1,2,2]}
df = pd.DataFrame(l)
print(df)
print('\n--------------------df.iloc[0:3]-------------------\n')
print(df.iloc[0:3])
```

```
    mpg  Cylinders   hp  weight  mYear  Origin
0  18.0          8  130    3504     70       1
1  15.0          8  165    3693     82       1
2  17.0          8  150    3436     70       1
3  27.0          4   84    3433     70       2
4  31.0          4   79    2130     82       2

--------------------df.iloc[0:3]-------------------

    mpg  Cylinders   hp  weight  mYear  Origin
0  18.0          8  130    3504     70       1
1  15.0          8  165    3693     82       1
2  17.0          8  150    3436     70       1
```

In [107]:

```
import pandas as pd

l = {'mpg':[18.0,15.0,17.0,27.0,31.0],
    'Cylinders':[8,8,8,4,4],
    'hp':[130,165,150,84,79],
    'weight':[3504,3693,3436,3433,2130],
    'mYear': [70,82,70,70,82],
    'Origin':[1,1,1,2,2]}
df = pd.DataFrame(l)
print(df)
print('\n--------------------------------------------------\n')
print(df.iloc[0:3,['hp']])

# IndexError: .iloc requires numeric indexers, got ['hp']
```

In [30]:

```
import pandas as pd

l = {'mpg':[18.0,15.0,17.0,27.0,31.0],
    'Cylinders':[8,8,8,4,4],
    'hp':[130,165,150,84,79],
    'weight':[3504,3693,3436,3433,2130],
    'mYear': [70,82,70,70,82],
    'Origin':[1,1,1,2,2]}
df = pd.DataFrame(l)
print(df)
print("\n----------------df.iloc[0:3]['hp']-----------------\n")
print(df.iloc[0:3]['hp'])
```

```
    mpg  Cylinders   hp  weight  mYear  Origin
0  18.0          8  130    3504     70       1
1  15.0          8  165    3693     82       1
2  17.0          8  150    3436     70       1
3  27.0          4   84    3433     70       2
4  31.0          4   79    2130     82       2

----------------df.iloc[0:3]['hp']-----------------

0    130
1    165
2    150
Name: hp, dtype: int64
```

In [100]:

```
import pandas as pd

l = {'mpg':[18.0,15.0,17.0,27.0,31.0],
    'Cylinders':[8,8,8,4,4],
    'hp':[130,165,150,84,79],
    'weight':[3504,3693,3436,3433,2130],
    'mYear': [70,82,70,70,82],
    'Origin':[1,1,1,2,2]}
df = pd.DataFrame(l)
print(df)
print('\n--------------------------------------------------\n')
print(df.iloc[0:3,['hp','mYear']])

# KeyError: ('hp', 'mYear')
```

In [31]:

```
import pandas as pd

l = {'mpg':[18.0,15.0,17.0,27.0,31.0],
    'Cylinders':[8,8,8,4,4],
    'hp':[130,165,150,84,79],
    'weight':[3504,3693,3436,3433,2130],
    'mYear': [70,82,70,70,82],
    'Origin':[1,1,1,2,2]}
df = pd.DataFrame(l)
print(df)
print("\n-------------df.loc[0:2,['hp','mYear']]-------------\n")
print(df.loc[0:2,['hp','mYear']])
```

```
    mpg  Cylinders   hp  weight  mYear  Origin
0  18.0          8  130    3504     70       1
1  15.0          8  165    3693     82       1
2  17.0          8  150    3436     70       1
3  27.0          4   84    3433     70       2
4  31.0          4   79    2130     82       2

-------------df.loc[0:2,['hp','mYear']]-------------

    hp  mYear
0  130     70
1  165     82
2  150     70
```

In [137]:

```
import pandas as pd

l = {'mpg':[18.0,15.0,17.0,27.0,31.0],
    'Cylinders':[8,8,8,4,4],
    'hp':[130,165,150,84,79],
    'weight':[3504,3693,3436,3433,2130],
    'mYear': [70,82,70,70,82],
    'Origin':[1,1,1,2,2]}
df = pd.DataFrame(l,dtype=float)
print(df)
print('\n--------------------------------------------------\n')
print(df.iloc[0:4,1:5])
# row x col
```

```
    mpg  Cylinders     hp  weight  mYear  Origin
0  18.0        8.0  130.0  3504.0   70.0     1.0
1  15.0        8.0  165.0  3693.0   82.0     1.0
2  17.0        8.0  150.0  3436.0   70.0     1.0
3  27.0        4.0   84.0  3433.0   70.0     2.0
4  31.0        4.0   79.0  2130.0   82.0     2.0

--------------------------------------------------

   Cylinders     hp  weight  mYear
0        8.0  130.0  3504.0   70.0
1        8.0  165.0  3693.0   82.0
2        8.0  150.0  3436.0   70.0
3        4.0   84.0  3433.0   70.0
```

In [120]:

```
d = {'a':10,'b':20,'c':30,'d':40}
df = pd.DataFrame(d)
print(df)
# ValueError: If using all scalar values, you must pass an index
```

In [121]:

```
d = {'a':10,'b':20,'c':30,'d':40}
df = pd.DataFrame(d,index=['x','y'])
print(df)
```

```
    a   b   c   d
x  10  20  30  40
y  10  20  30  40
```

In [124]:

```python
import pandas as pd

l = {'mpg':[18.0,15.0,17.0,27.0,31.0],
    'Cylinders':[8,8,8,4,4],
    'hp':[130,165,150,84,79],
    'weight':[None,3693,3436,3433,2130],
    'mYear': [70,82,70,70,82],
    'Origin':[1,1,1,2,2]}
df = pd.DataFrame(l)
print(df)
print('\n--------------------------------------------------\n')
print(df.iloc[0:4,1:5:2])
```

```
    mpg  Cylinders   hp  weight  mYear  Origin
0  18.0          8  130     NaN     70       1
1  15.0          8  165  3693.0     82       1
2  17.0          8  150  3436.0     70       1
3  27.0          4   84  3433.0     70       2
4  31.0          4   79  2130.0     82       2

--------------------------------------------------

   Cylinders  weight
0          8     NaN
1          8  3693.0
2          8  3436.0
3          4  3433.0
```

## Whether NaN in col then whole converted into float

In [130]:

```python
import pandas as pd

l = {'mpg':[18.0,15.0,17.0,27.0,31.0],
    'Cylinders':[8,8,8,4,4],
    'hp':[130,165,150,84,79],
    'weight':[None,3693,3436,3433,2130],
    'mYear': [70,82,70,70,82],
    'Origin':[1,1,1,2,2]}
df = pd.DataFrame(l)
print(df)
print('\n--------------------------------------------------\n')
print(df['hp'])
```

```
    mpg  Cylinders   hp  weight  mYear  Origin
0  18.0          8  130     NaN     70       1
1  15.0          8  165  3693.0     82       1
2  17.0          8  150  3436.0     70       1
3  27.0          4   84  3433.0     70       2
4  31.0          4   79  2130.0     82       2

--------------------------------------------------

0    130
1    165
2    150
3     84
4     79
Name: hp, dtype: int64
```

In [134]:

```python
import pandas as pd

l = {'mpg':[18.0,15.0,17.0,27.0,31.0],
    'Cylinders':[8,8,8,4,4],
    'hp':[130,165,150,84,79],
    'weight':[None,3693,3436,3433,2130],
    'mYear': [70,82,70,70,82],
    'Origin':[1,1,1,2,2]}
df = pd.DataFrame(l,columns=['hp'])
print(df)
```

```
    hp
0  130
1  165
2  150
3   84
4   79
```

- ## Works with .csv file

In [36]:

```python
import pandas as pd

df = pd.read_csv('Datasets/auto-mpg.csv')
df
```

Out[36]:

| | 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 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 |

398 rows × 9 columns

In [50]:

```python
import pandas as pd

df = pd.read_csv('Datasets/auto-mpg.csv')
print(df)
print("----------------------df.shape-----------------------")
print(df.shape)
print("-------------------df.shape[1]-----------------------")
print(df.shape[1])
print("--------------------df.columns-----------------------")
print(df.columns)
```

```
      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]
----------------------df.shape-----------------------
(398, 9)
-------------------df.shape[1]-----------------------
9
--------------------df.columns-----------------------
Index(['mpg', 'cylinders', 'displacement', 'horsepower', 'weight',
       'acceleration', 'model year', 'origin', 'car name'],
      dtype='object')
```

In [51]:

```
import pandas as pd

df = pd.read_csv('Datasets/auto-mpg.csv')
print("--------------------df.head()--------------------------")
print(df.head())
print("--------------------df.head(10)--------------------------")
print(df.head(10))
print("--------------------df.tail(20)--------------------------")
print(df.tail(20))
```

```
--------------------df.head()-------------------------
    mpg  cylinders  displacement horsepower  weight  acceleration  model year  \
0  18.0          8         307.0        130    3504          12.0          70
1  15.0          8         350.0        165    3693          11.5          70
2  18.0          8         318.0        150    3436          11.0          70
3  16.0          8         304.0        150    3433          12.0          70
4  17.0          8         302.0        140    3449          10.5          70

   origin                   car name
0       1  chevrolet chevelle malibu
1       1          buick skylark 320
2       1         plymouth satellite
3       1              amc rebel sst
4       1                ford torino
--------------------df.head(10)-------------------------
    mpg  cylinders  displacement horsepower  weight  acceleration  model year  \
0  18.0          8         307.0        130    3504          12.0          70
1  15.0          8         350.0        165    3693          11.5          70
2  18.0          8         318.0        150    3436          11.0          70
3  16.0          8         304.0        150    3433          12.0          70
4  17.0          8         302.0        140    3449          10.5          70
5  15.0          8         429.0        198    4341          10.0          70
6  14.0          8         454.0        220    4354           9.0          70
7  14.0          8         440.0        215    4312           8.5          70
8  14.0          8         455.0        225    4425          10.0          70
9  15.0          8         390.0        190    3850           8.5          70

   origin                   car name
0       1  chevrolet chevelle malibu
1       1          buick skylark 320
2       1         plymouth satellite
3       1              amc rebel sst
4       1                ford torino
5       1           ford galaxie 500
6       1           chevrolet impala
7       1          plymouth fury iii
8       1           pontiac catalina
9       1         amc ambassador dpl
--------------------df.tail(20)-------------------------
      mpg  cylinders  displacement horsepower  weight  acceleration  \
378  38.0          4         105.0         63    2125          14.7
379  36.0          4          98.0         70    2125          17.3
380  36.0          4         120.0         88    2160          14.5
381  36.0          4         107.0         75    2205          14.5
382  34.0          4         108.0         70    2245          16.9
383  38.0          4          91.0         67    1965          15.0
384  32.0          4          91.0         67    1965          15.7
385  38.0          4          91.0         67    1995          16.2
386  25.0          6         181.0        110    2945          16.4
387  38.0          6         262.0         85    3015          17.0
388  26.0          4         156.0         92    2585          14.5
389  22.0          6         232.0        112    2835          14.7
390  32.0          4         144.0         96    2665          13.9
391  36.0          4         135.0         84    2370          13.0
392  27.0          4         151.0         90    2950          17.3
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
378          82       1             plymouth horizon miser
379          82       1                     mercury lynx l
380          82       3                   nissan stanza xe
381          82       3                       honda accord
382          82       3                     toyota corolla
383          82       3                        honda civic
384          82       3                 honda civic (auto)
385          82       3                      datsun 310 gx
386          82       1              buick century limited
387          82       1  oldsmobile cutlass ciera (diesel)
388          82       1         chrysler lebaron medallion
389          82       1                     ford granada l
390          82       3                   toyota celica gt
391          82       1                  dodge charger 2.2
392          82       1                   chevrolet camaro
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
```

In [55]:

```python
import pandas as pd

df = pd.read_csv('Datasets/auto-mpg.csv')
print(df.info())
# This is True
```

```
<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
None
```

In [54]:

```python
import pandas as pd

df = pd.read_csv('Datasets/auto-mpg.csv')
print(df.info)
```

```
<bound method DataFrame.info of       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 [58]:

```python
import pandas as pd

df = pd.read_csv('Datasets/auto-mpg.csv')
print(df.describe())
```

```
              mpg   cylinders  displacement       weight  acceleration  \
count  398.000000  398.000000    398.000000   398.000000    398.000000   
mean    23.514573    5.454774    193.425879  2970.424623     15.568090   
std      7.815984    1.701004    104.269838   846.841774      2.757689   
min      9.000000    3.000000     68.000000  1613.000000      8.000000   
25%     17.500000    4.000000    104.250000  2223.750000     13.825000   
50%     23.000000    4.000000    148.500000  2803.500000     15.500000   
75%     29.000000    8.000000    262.000000  3608.000000     17.175000   
max     46.600000    8.000000    455.000000  5140.000000     24.800000   

       model year      origin  
count  398.000000  398.000000  
mean    76.010050    1.572864  
std      3.697627    0.802055  
min     70.000000    1.000000  
25%     73.000000    1.000000  
50%     76.000000    1.000000  
75%     79.000000    2.000000  
max     82.000000    3.000000  
```

In [62]:

```python
import pandas as pd

df = pd.read_csv('Datasets/auto-mpg.csv')
print(df.describe(include=object))
```

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

In [63]:

```python
import pandas as pd

df = pd.read_csv('Datasets/auto-mpg.csv')
print(df.describe(percentiles=[.3,.6,.9]))
```

```
              mpg   cylinders  displacement       weight  acceleration  \
count  398.000000  398.000000    398.000000   398.000000    398.000000   
mean    23.514573    5.454774    193.425879  2970.424623     15.568090   
std      7.815984    1.701004    104.269838   846.841774      2.757689   
min      9.000000    3.000000     68.000000  1613.000000      8.000000   
30%     18.000000    4.000000    112.000000  2301.000000     14.200000   
50%     23.000000    4.000000    148.500000  2803.500000     15.500000   
60%     25.000000    6.000000    200.000000  3085.200000     16.000000   
90%     34.330000    8.000000    350.000000  4275.200000     19.000000   
max     46.600000    8.000000    455.000000  5140.000000     24.800000   

       model year      origin  
count  398.000000  398.000000  
mean    76.010050    1.572864  
std      3.697627    0.802055  
min     70.000000    1.000000  
30%     73.000000    1.000000  
50%     76.000000    1.000000  
60%     77.000000    1.000000  
90%     81.000000    3.000000  
max     82.000000    3.000000  
```

In [64]:

```python
import pandas as pd

df = pd.read_csv('Datasets/auto-mpg.csv')
print(df.loc[df['cylinders']==8])
```

```
      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   
..    ...        ...           ...        ...     ...           ...   
291  19.2          8         267.0        125    3605          15.0   
292  18.5          8         360.0        150    3940          13.0   
298  23.0          8         350.0        125    3900          17.4   
300  23.9          8         260.0         90    3420          22.2   
364  26.6          8         350.0        105    3725          19.0   

     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  
..          ...     ...                                 ...  
291          79       1          chevrolet malibu classic (sw)  
292          79       1  chrysler lebaron town @ country (sw)  
298          79       1                   cadillac eldorado  
300          79       1     oldsmobile cutlass salon brougham  
364          81       1                oldsmobile cutlass ls  

[103 rows x 9 columns]
```

In [71]:

```
df = pd.read_csv('Datasets/auto-mpg.csv')
print(df.loc[(df['displacement']>300) & (df['origin']==1)])
```

```
      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   
..    ...        ...           ...        ...     ...           ...   
289  16.9          8         350.0        155    4360          14.9   
290  15.5          8         351.0        142    4054          14.3   
292  18.5          8         360.0        150    3940          13.0   
298  23.0          8         350.0        125    3900          17.4   
364  26.6          8         350.0        105    3725          19.0   

     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  
..          ...     ...                                    ...  
289          79       1                buick estate wagon (sw)  
290          79       1               ford country squire (sw)  
292          79       1  chrysler lebaron town @ country (sw)  
298          79       1                      cadillac eldorado  
364          81       1                  oldsmobile cutlass ls  

[98 rows x 9 columns]
```

In [79]:

```
df = pd.read_csv('Datasets/auto-mpg.csv')
print(df.loc[(df['horsepower']=='150') |(( df['cylinders']==8)&(df['weight']>3500))])
```

```
      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   
5    15.0          8         429.0        198    4341          10.0   
..    ...        ...           ...        ...     ...           ...   
290  15.5          8         351.0        142    4054          14.3   
291  19.2          8         267.0        125    3605          15.0   
292  18.5          8         360.0        150    3940          13.0   
298  23.0          8         350.0        125    3900          17.4   
364  26.6          8         350.0        105    3725          19.0   

     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  
5            70       1                       ford galaxie 500  
..          ...     ...                                    ...  
290          79       1               ford country squire (sw)  
291          79       1           chevrolet malibu classic (sw)  
292          79       1  chrysler lebaron town @ country (sw)  
298          79       1                      cadillac eldorado  
364          81       1                  oldsmobile cutlass ls  

[95 rows x 9 columns]
```

In [80]:

```
df = pd.read_csv('Datasets/auto-mpg.csv')
print(df['cylinders'])
```

```
0      8
1      8
2      8
3      8
4      8
      ..
393    4
394    4
395    4
396    4
397    4
Name: cylinders, Length: 398, dtype: int64
```

In [81]:

```python
df = pd.read_csv('Datasets/Car details v3.csv')
df
```

Out[81]:

| | name | year | selling_price | km_driven | fuel | seller_type | transmission | owner | mileage | engine | max_power | torque | seats |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | Maruti Swift Dzire VDI | 2014 | 450000 | 145500 | Diesel | Individual | Manual | First Owner | 23.4 kmpl | 1248 CC | 74 bhp | 190Nm@ 2000rpm | 5.0 |
| **1** | Skoda Rapid 1.5 TDI Ambition | 2014 | 370000 | 120000 | Diesel | Individual | Manual | Second Owner | 21.14 kmpl | 1498 CC | 103.52 bhp | 250Nm@ 1500-2500rpm | 5.0 |
| **2** | Honda City 2017-2020 EXi | 2006 | 158000 | 140000 | Petrol | Individual | Manual | Third Owner | 17.7 kmpl | 1497 CC | 78 bhp | 12.7@ 2,700(kgm@ rpm) | 5.0 |
| **3** | Hyundai i20 Sportz Diesel | 2010 | 225000 | 127000 | Diesel | Individual | Manual | First Owner | 23.0 kmpl | 1396 CC | 90 bhp | 22.4 kgm at 1750-2750rpm | 5.0 |
| **4** | Maruti Swift VXI BSIII | 2007 | 130000 | 120000 | Petrol | Individual | Manual | First Owner | 16.1 kmpl | 1298 CC | 88.2 bhp | 11.5@ 4,500(kgm@ rpm) | 5.0 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **8123** | Hyundai i20 Magna | 2013 | 320000 | 110000 | Petrol | Individual | Manual | First Owner | 18.5 kmpl | 1197 CC | 82.85 bhp | 113.7Nm@ 4000rpm | 5.0 |
| **8124** | Hyundai Verna CRDi SX | 2007 | 135000 | 119000 | Diesel | Individual | Manual | Fourth & Above Owner | 16.8 kmpl | 1493 CC | 110 bhp | 24@ 1,900-2,750(kgm@ rpm) | 5.0 |
| **8125** | Maruti Swift Dzire ZDi | 2009 | 382000 | 120000 | Diesel | Individual | Manual | First Owner | 19.3 kmpl | 1248 CC | 73.9 bhp | 190Nm@ 2000rpm | 5.0 |
| **8126** | Tata Indigo CR4 | 2013 | 290000 | 25000 | Diesel | Individual | Manual | First Owner | 23.57 kmpl | 1396 CC | 70 bhp | 140Nm@ 1800-3000rpm | 5.0 |
| **8127** | Tata Indigo CR4 | 2013 | 290000 | 25000 | Diesel | Individual | Manual | First Owner | 23.57 kmpl | 1396 CC | 70 bhp | 140Nm@ 1800-3000rpm | 5.0 |

8128 rows × 13 columns

In [83]:

```python
df = pd.read_csv('Datasets/Car details v3.csv', usecols=['name'])
df
```

Out[83]:

| | name |
|---|---|
| **0** | Maruti Swift Dzire VDI |
| **1** | Skoda Rapid 1.5 TDI Ambition |
| **2** | Honda City 2017-2020 EXi |
| **3** | Hyundai i20 Sportz Diesel |
| **4** | Maruti Swift VXI BSIII |
| **...** | ... |
| **8123** | Hyundai i20 Magna |
| **8124** | Hyundai Verna CRDi SX |
| **8125** | Maruti Swift Dzire ZDi |
| **8126** | Tata Indigo CR4 |
| **8127** | Tata Indigo CR4 |

8128 rows × 1 columns

In [93]:

```python
df = pd.read_csv('Datasets/Car details v3.csv', skiprows=5)
df
```

Out[93]:

| | Maruti Swift VXI BSIII | 2007 | 130000 | 120000 | Petrol | Individual | Manual | First Owner | 16.1 kmpl | 1298 CC | 88.2 bhp | 11.5@ 4,500(kgm@ rpm) | 5 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | Hyundai Xcent 1.2 VTVT E Plus | 2017 | 440000 | 45000 | Petrol | Individual | Manual | First Owner | 20.14 kmpl | 1197 CC | 81.86 bhp | 113.75nm@ 4000rpm | 5.0 |
| **1** | Maruti Wagon R LXI DUO BSIII | 2007 | 96000 | 175000 | LPG | Individual | Manual | First Owner | 17.3 km/kg | 1061 CC | 57.5 bhp | 7.8@ 4,500(kgm@ rpm) | 5.0 |
| **2** | Maruti 800 DX BSII | 2001 | 45000 | 5000 | Petrol | Individual | Manual | Second Owner | 16.1 kmpl | 796 CC | 37 bhp | 59Nm@ 2500rpm | 4.0 |
| **3** | Toyota Etios VXD | 2011 | 350000 | 90000 | Diesel | Individual | Manual | First Owner | 23.59 kmpl | 1364 CC | 67.1 bhp | 170Nm@ 1800-2400rpm | 5.0 |
| **4** | Ford Figo Diesel Celebration Edition | 2013 | 200000 | 169000 | Diesel | Individual | Manual | First Owner | 20.0 kmpl | 1399 CC | 68.1 bhp | 160Nm@ 2000rpm | 5.0 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **8118** | Hyundai i20 Magna | 2013 | 320000 | 110000 | Petrol | Individual | Manual | First Owner | 18.5 kmpl | 1197 CC | 82.85 bhp | 113.7Nm@ 4000rpm | 5.0 |
| **8119** | Hyundai Verna CRDi SX | 2007 | 135000 | 119000 | Diesel | Individual | Manual | Fourth & Above Owner | 16.8 kmpl | 1493 CC | 110 bhp | 24@ 1,900-2,750(kgm@ rpm) | 5.0 |
| **8120** | Maruti Swift Dzire ZDi | 2009 | 382000 | 120000 | Diesel | Individual | Manual | First Owner | 19.3 kmpl | 1248 CC | 73.9 bhp | 190Nm@ 2000rpm | 5.0 |
| **8121** | Tata Indigo CR4 | 2013 | 290000 | 25000 | Diesel | Individual | Manual | First Owner | 23.57 kmpl | 1396 CC | 70 bhp | 140Nm@ 1800-3000rpm | 5.0 |
| **8122** | Tata Indigo CR4 | 2013 | 290000 | 25000 | Diesel | Individual | Manual | First Owner | 23.57 kmpl | 1396 CC | 70 bhp | 140Nm@ 1800-3000rpm | 5.0 |

8123 rows × 13 columns

# Handling missing values:

1. `isna() / isnull()`
2. `dropna()`

In [105]:

```python
import pandas as pd
df = pd.DataFrame({'A':[None,None,40,None,26],
                   'B':[20,None,None,25,None,],
                   'C':[None,None,None,None,None],
                   'D':[30,None,60,None,100],
                   'E':[None,None,90,None,None,]})
df
```

Out[105]:

| | A | B | C | D | E |
|---|---|---|---|---|---|
| **0** | NaN | 20.0 | None | 30.0 | NaN |
| **1** | NaN | NaN | None | NaN | NaN |
| **2** | 40.0 | NaN | None | 60.0 | 90.0 |
| **3** | NaN | 25.0 | None | NaN | NaN |
| **4** | 26.0 | NaN | None | 100.0 | NaN |

In [106]:

```python
df = pd.DataFrame({'A':[None,None,40,None,26],'B':[20,None,None,25,None,],'C':[None,None,None,None,None],
                   'D':[30,None,60,None,100],'E':[None,None,90,None,None,]})
print(df.isna())
df.isnull()
```

```
       A      B     C      D      E
0   True  False  True  False   True
1   True   True  True   True   True
2  False   True  True  False  False
3   True  False  True   True   True
4  False   True  True  False   True
```

Out[106]:

| | A | B | C | D | E |
|---|---|---|---|---|---|
| **0** | True | False | True | False | True |
| **1** | True | True | True | True | True |
| **2** | False | True | True | False | False |
| **3** | True | False | True | True | True |
| **4** | False | True | True | False | True |

In [107]:

```
df = pd.DataFrame({'A':[None,None,40,None,26],'B':[20,None,None,25,None,],'C':[None,None,None,None,None],
                   'D':[30,None,60,None,100],'E':[None,None,90,None,None,]})
print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5 entries, 0 to 4
Data columns (total 5 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   A       2 non-null      float64
 1   B       2 non-null      float64
 2   C       0 non-null      object 
 3   D       3 non-null      float64
 4   E       1 non-null      float64
dtypes: float64(4), object(1)
memory usage: 328.0+ bytes
None
```

In [111]:

```
df = pd.DataFrame({'A':[None,None,40,None,26],'B':[20,None,None,25,None,],'C':[None,None,None,None,None],
                   'D':[30,None,60,None,100],'E':[None,None,90,None,None,]})
print(df['D'].isna())
```

```
0    False
1     True
2    False
3     True
4    False
Name: D, dtype: bool
```

## dropna(how, axis, subset, thresh, inplace)

In [113]:

```
df = pd.DataFrame({'A':[None,None,40,None,26],'B':[20,None,15,25,None,],'C':[None,None,20,None,None],
                   'D':[30,None,60,None,100],'E':[None,None,90,None,None,]})
new_df = df.dropna()
print(new_df)
```

```
      A     B     C     D     E
2  40.0  15.0  20.0  60.0  90.0
```

In [122]:

```
df = pd.DataFrame({'A':[None,None,40,None,26],'B':[20,30,55,25,21],'C':[None,None,None,None,None],
                   'D':[30,None,60,None,100],'E':[None,None,90,None,None,]})
print(df)
print("--------------------------------")
new_df = df.dropna()
print(new_df)
```

```
      A   B     C      D     E
0   NaN  20  None   30.0   NaN
1   NaN  30  None    NaN   NaN
2  40.0  55  None   60.0  90.0
3   NaN  25  None    NaN   NaN
4  26.0  21  None  100.0   NaN
--------------------------------
Empty DataFrame
Columns: [A, B, C, D, E]
Index: []
```

In [121]:

```
df = pd.DataFrame({'A':[None,None,40,None,26],'B':[20,30,55,25,21],'C':[None,None,None,None,None],
                   'D':[30,None,60,None,100],'E':[None,None,90,None,None,]})
print(df)
print("--------------------------------")
new_df = df.dropna(axis=1) # for column
print(new_df)
```

```
      A   B     C      D     E
0   NaN  20  None   30.0   NaN
1   NaN  30  None    NaN   NaN
2  40.0  55  None   60.0  90.0
3   NaN  25  None    NaN   NaN
4  26.0  21  None  100.0   NaN
--------------------------------
    B
0  20
1  30
2  55
3  25
4  21
```

In [127]:

```
df = pd.DataFrame({'A':[None,None,40,None,26],'B':[20,None,None,25,None,],'C':[None,None,None,None,None],
                   'D':[30,None,60,None,100],'E':[None,None,90,None,None,]})
new_df = df.dropna(axis=1,how='all')
print(new_df)
```

```
      A     B      D     E
0   NaN  20.0   30.0   NaN
1   NaN   NaN    NaN   NaN
2  40.0   NaN   60.0  90.0
3   NaN  25.0    NaN   NaN
4  26.0   NaN  100.0   NaN
```

- ### It removes whole columns consists None

In [128]:

```
df = pd.DataFrame({'A':[None,None,None,None,None],'B':[20,None,None,25,None,],'C':[None,None,None,None,None],
                   'D':[30,None,60,None,100],'E':[None,None,90,None,None,]})
new_df = df.dropna(axis=1,how='all')
print(new_df)
```

```
      B      D     E
0  20.0   30.0   NaN
1   NaN    NaN   NaN
2   NaN   60.0  90.0
3  25.0    NaN   NaN
4   NaN  100.0   NaN
```

- ### It removes any columns consists any one None

In [130]:

```
df = pd.DataFrame({'A':[None,None,40,None,26],'B':[20,None,None,25,None,],'C':[None,None,None,None,None],
                   'D':[30,None,60,None,100],'E':[None,None,90,None,None,]})
new_df = df.dropna(axis=1,how='any')
print(new_df)
```

```
Empty DataFrame
Columns: []
Index: [0, 1, 2, 3, 4]
```

In [133]:

```
import pandas as pd

# Create a DataFrame with some missing values
df = pd.DataFrame({
    'A': [1, 2, None, 4, 5],
    'B': [None, 2, 3, 4, None],
    'C': [1, None, 3, 4, 5]
})

# Drop rows that contain any NaN values and reset the index
new_df = df.dropna(axis=0, how='any', ignore_index=True)

print("Original DataFrame:")
print(df)

print("\nNew DataFrame after dropping rows with NaN values:")
print(new_df)
```

- ### subset : to focus on particular data

In [4]:

```
import pandas as pd
df = pd.DataFrame({'A':[None,None,40,None,26],'B':[20,None,None,25,None,],'C':[None,None,None,None,None],
                   'D':[30,None,60,None,100],'E':[None,None,90,None,None,]})
new_df = df.dropna(subset=['B','D'])
print(new_df)
df
```

```
    A     B     C     D   E
0 NaN  20.0  None  30.0 NaN
```

Out[4]:

| | A | B | C | D | E |
|---|---|---|---|---|---|
| **0** | NaN | 20.0 | None | 30.0 | NaN |
| **1** | NaN | NaN | None | NaN | NaN |
| **2** | 40.0 | NaN | None | 60.0 | 90.0 |
| **3** | NaN | 25.0 | None | NaN | NaN |
| **4** | 26.0 | NaN | None | 100.0 | NaN |

In [10]:

```python
import pandas as pd
df = pd.DataFrame({'A':[None,None,40,None,26],'B':[20,None,None,25,60,],'C':[None,None,None,None,None],
                   'D':[30,None,60,None,100],'E':[None,None,90,None,None,]})
new_df = df.dropna(subset=['B','D'])
print(new_df)
df
```

```
      A     B     C      D   E
0   NaN  20.0  None   30.0 NaN
4  26.0  60.0  None  100.0 NaN
```

Out[10]:

|   | A | B | C | D | E |
|---|---|---|---|---|---|
| **0** | NaN | 20.0 | None | 30.0 | NaN |
| **1** | NaN | NaN | None | NaN | NaN |
| **2** | 40.0 | NaN | None | 60.0 | 90.0 |
| **3** | NaN | 25.0 | None | NaN | NaN |
| **4** | 26.0 | 60.0 | None | 100.0 | NaN |

In [137]:

```python
df = pd.DataFrame({'A':[None,None,40,None,26],'B':[20,None,None,25,None,],'C':[None,None,None,None,None],
                   'D':[30,None,60,None,100],'E':[None,None,90,None,None,]})
new_df = df.dropna(how='all', inplace=True)
print(new_df)
```

```
None
```

- **inplace - it returns nothing🤷‍♀️**

In [141]:

```python
df = pd.DataFrame({'A':[None,None,40,None,26],'B':[20,None,None,25,None,],'C':[None,None,None,None,None],
                   'D':[30,None,60,None,100],'E':[None,None,90,None,None,]})
print(df)
print("-------------------------------------")
df.dropna(how='all', inplace=True)
print(df)
```

```
      A     B     C      D     E
0   NaN  20.0  None   30.0   NaN
1   NaN   NaN  None    NaN   NaN
2  40.0   NaN  None   60.0  90.0
3   NaN  25.0  None    NaN   NaN
4  26.0   NaN  None  100.0   NaN
-------------------------------------
      A     B     C      D     E
0   NaN  20.0  None   30.0   NaN
2  40.0   NaN  None   60.0  90.0
3   NaN  25.0  None    NaN   NaN
4  26.0   NaN  None  100.0   NaN
```

- **thresh can be deifened as min no. of notnull values in a row or col else will be deleted.**

In [9]:

```python
df = pd.DataFrame({'A':[None,None,40,None,26],'B':[20,None,None,25,26],'C':[30,None,None,None,None],
                   'D':[30,None,60,None,100],'E':[None,None,90,None,None,]})
x = df.dropna(thresh=3)
print(x)
df
```

```
      A     B     C      D     E
0   NaN  20.0  30.0   30.0   NaN
2  40.0   NaN   NaN   60.0  90.0
4  26.0  26.0   NaN  100.0   NaN
```

Out[9]:

|   | A | B | C | D | E |
|---|---|---|---|---|---|
| **0** | NaN | 20.0 | 30.0 | 30.0 | NaN |
| **1** | NaN | NaN | NaN | NaN | NaN |
| **2** | 40.0 | NaN | NaN | 60.0 | 90.0 |
| **3** | NaN | 25.0 | NaN | NaN | NaN |
| **4** | 26.0 | 26.0 | NaN | 100.0 | NaN |

# Insert at NaN

In [14]:

```
df = pd.DataFrame({'A':[None,None,40,None,26],'B':[20,None,None,25,26],'C':[30,None,None,None,None],
                   'D':[30,None,60,None,100],'E':[None,None,90,None,None,]})
df.fillna(500,inplace=True)
df
```

Out[14]:

| | A | B | C | D | E |
|---|---|---|---|---|---|
| **0** | NaN | 20.0 | 30.0 | 30.0 | NaN |
| **1** | NaN | NaN | NaN | NaN | NaN |
| **2** | 40.0 | NaN | NaN | 60.0 | 90.0 |
| **3** | NaN | 25.0 | NaN | NaN | NaN |
| **4** | 26.0 | 26.0 | NaN | 100.0 | NaN |

In [15]:

```
df = pd.DataFrame({'A':[None,None,40,None,26],'B':[20,None,None,25,26],'C':[30,None,None,None,None],
                   'D':[30,None,60,None,100],'E':[None,None,90,None,None,]})
x = df.fillna(500,inplace=False)
x
```

Out[15]:

| | A | B | C | D | E |
|---|---|---|---|---|---|
| **0** | 500.0 | 20.0 | 30.0 | 30.0 | 500.0 |
| **1** | 500.0 | 500.0 | 500.0 | 500.0 | 500.0 |
| **2** | 40.0 | 500.0 | 500.0 | 60.0 | 90.0 |
| **3** | 500.0 | 25.0 | 500.0 | 500.0 | 500.0 |
| **4** | 26.0 | 26.0 | 500.0 | 100.0 | 500.0 |

- ## limit = It will Replace NaN by value Column wise.

In [22]:

```
df = pd.DataFrame({'A':[1,None,40,None,26],'B':[20,None,None,25,26],'C':[30,None,None,None,None],
                   'D':[30,None,60,None,100],'E':[None,None,90,None,None,]})
df.fillna(500,inplace=True, limit=2)
df
```

Out[22]:

| | A | B | C | D | E |
|---|---|---|---|---|---|
| **0** | 1.0 | 20.0 | 30.0 | 30.0 | 500.0 |
| **1** | 500.0 | 500.0 | 500.0 | 500.0 | 500.0 |
| **2** | 40.0 | 500.0 | 500.0 | 60.0 | 90.0 |
| **3** | 500.0 | 25.0 | NaN | 500.0 | NaN |
| **4** | 26.0 | 26.0 | NaN | 100.0 | NaN |

In [27]:

```
df = pd.DataFrame({'A':[None,None,40,None,26],'B':[20,None,None,25,26],'C':[30,None,None,None,None],
                   'D':[30,None,60,None,100],'E':[None,None,90,None,None,]})
v = {'A':500,'B':600,'C':700,'D':800,'E':900}
df.fillna(v,inplace=True)
df
```

Out[27]:

| | A | B | C | D | E |
|---|---|---|---|---|---|
| **0** | 500.0 | 20.0 | 30.0 | 30.0 | 900.0 |
| **1** | 500.0 | 600.0 | 700.0 | 800.0 | 900.0 |
| **2** | 40.0 | 600.0 | 700.0 | 60.0 | 90.0 |
| **3** | 500.0 | 25.0 | 700.0 | 800.0 | 900.0 |
| **4** | 26.0 | 26.0 | 700.0 | 100.0 | 900.0 |

In [32]:

```python
df = pd.DataFrame({'A':[40,None,40,None,26],'B':[20,None,None,25,26],'C':[30,None,None,None,None],
                   'D':[30,None,60,None,100],'E':[None,None,90,None,None,]})
v = {'A':500,'B':600,'C':700,'D':800,'E':900}
print(df)
df.fillna(method='ffill',inplace=True)
df
# ffill - forward fill
```

```
      A     B     C      D     E
0  40.0  20.0  30.0   30.0   NaN
1   NaN   NaN   NaN    NaN   NaN
2  40.0   NaN   NaN   60.0  90.0
3   NaN  25.0   NaN    NaN   NaN
4  26.0  26.0   NaN  100.0   NaN
```

Out[32]:

```
      A     B     C      D     E
0  40.0  20.0  30.0   30.0   NaN
1  40.0  20.0  30.0   30.0   NaN
2  40.0  20.0  30.0   60.0  90.0
3  40.0  25.0  30.0   60.0  90.0
4  26.0  26.0  30.0  100.0  90.0
```

In [31]:

```python
df = pd.DataFrame({'A':[40,None,40,None,26],'B':[20,None,None,25,26],'C':[30,None,None,None,None],
                   'D':[30,None,60,None,100],'E':[None,None,90,None,None,]})
v = {'A':500,'B':600,'C':700,'D':800,'E':900}
print(df)
df.fillna(method='bfill',inplace=True)
df
# bfill - backward fill
```

```
      A     B     C      D     E
0  40.0  20.0  30.0   30.0   NaN
1   NaN   NaN   NaN    NaN   NaN
2  40.0   NaN   NaN   60.0  90.0
3   NaN  25.0   NaN    NaN   NaN
4  26.0  26.0   NaN  100.0   NaN
```

Out[31]:

```
      A     B     C      D     E
0  40.0  20.0  30.0   30.0  90.0
1  40.0  25.0   NaN   60.0  90.0
2  40.0  25.0   NaN   60.0  90.0
3  26.0  25.0   NaN  100.0   NaN
4  26.0  26.0   NaN  100.0   NaN
```

In [35]:

```python
df = pd.DataFrame({'A':[40,None,40,None,26],'B':[20,None,None,25,26],'C':[30,None,None,None,None],
                   'D':[30,None,60,None,100],'E':[None,None,90,None,None,]})
v = {'A':500,'B':600,'C':700,'D':800,'E':900}
print(df)
df.fillna(method='ffill',inplace=True,axis=0,limit=2)
df
```

```
      A     B     C      D     E
0  40.0  20.0  30.0   30.0   NaN
1   NaN   NaN   NaN    NaN   NaN
2  40.0   NaN   NaN   60.0  90.0
3   NaN  25.0   NaN    NaN   NaN
4  26.0  26.0   NaN  100.0   NaN
```

Out[35]:

```
      A     B     C      D     E
0  40.0  20.0  30.0   30.0   NaN
1  40.0  20.0  30.0   30.0   NaN
2  40.0  20.0  30.0   60.0  90.0
3  40.0  25.0   NaN   60.0  90.0
4  26.0  26.0   NaN  100.0  90.0
```

In [33]:

```python
help(pd.DataFrame.fillna)
```

```
Help on function fillna in module pandas.core.frame:

fillna(self, value=None, method=None, axis=None, inplace=False, limit=None, downcast=None) -> Union[ForwardRef('DataF
rame'), NoneType]
    Fill NA/NaN values using the specified method.
    
    Parameters
    ----------
    value : scalar, dict, Series, or DataFrame
        Value to use to fill holes (e.g. 0), alternately a
        dict/Series/DataFrame of values specifying which value to use for
        each index (for a Series) or column (for a DataFrame).  Values not
        in the dict/Series/DataFrame will not be filled. This value cannot
        be a list.
    method : {'backfill', 'bfill', 'pad', 'ffill', None}, default None
        Method to use for filling holes in reindexed Series
        pad / ffill: propagate last valid observation forward to next valid
        backfill / bfill: use next valid observation to fill gap.
    axis : {0 or 'index', 1 or 'columns'}
        Axis along which to fill missing values.
    inplace : bool, default False
        If True, fill in-place. Note: this will modify any
        other views on this object (e.g., a no-copy slice for a column in a
        DataFrame).
    limit : int, default None
        If method is specified, this is the maximum number of consecutive
        NaN values to forward/backward fill. In other words, if there is
        a gap with more than this number of consecutive NaNs, it will only
        be partially filled. If method is not specified, this is the
        maximum number of entries along the entire axis where NaNs will be
        filled. Must be greater than 0 if not None.
    downcast : dict, default is None
        A dict of item->dtype of what to downcast if possible,
        or the string 'infer' which will try to downcast to an appropriate
        equal type (e.g. float64 to int64 if possible).
    
    Returns
    -------
    DataFrame or None
        Object with missing values filled or None if ``inplace=True``.
    
    See Also
    --------
    interpolate : Fill NaN values using interpolation.
    reindex : Conform object to new index.
    asfreq : Convert TimeSeries to specified frequency.
    
    Examples
    --------
    >>> df = pd.DataFrame([[np.nan, 2, np.nan, 0],
    ...                    [3, 4, np.nan, 1],
    ...                    [np.nan, np.nan, np.nan, 5],
    ...                    [np.nan, 3, np.nan, 4]],
    ...                   columns=list('ABCD'))
    >>> df
         A    B   C  D
    0  NaN  2.0 NaN  0
    1  3.0  4.0 NaN  1
    2  NaN  NaN NaN  5
    3  NaN  3.0 NaN  4
    
    Replace all NaN elements with 0s.
    
    >>> df.fillna(0)
        A   B   C   D
    0   0.0 2.0 0.0 0
    1   3.0 4.0 0.0 1
    2   0.0 0.0 0.0 5
    3   0.0 3.0 0.0 4
    
    We can also propagate non-null values forward or backward.
    
    >>> df.fillna(method='ffill')
        A   B   C   D
    0   NaN 2.0 NaN 0
    1   3.0 4.0 NaN 1
    2   3.0 4.0 NaN 5
    3   3.0 3.0 NaN 4
    
    Replace all NaN elements in column 'A', 'B', 'C', and 'D', with 0, 1,
    2, and 3 respectively.
    
    >>> values = {'A': 0, 'B': 1, 'C': 2, 'D': 3}
    >>> df.fillna(value=values)
        A   B   C   D
    0   0.0 2.0 2.0 0
    1   3.0 4.0 2.0 1
    2   0.0 1.0 2.0 5
```

```
3   0.0 3.0 2.0 4

Only replace the first NaN element.

>>> df.fillna(value=values, limit=1)
    A   B   C   D
0   0.0 2.0 2.0 0
1   3.0 4.0 NaN 1
2   NaN 1.0 NaN 5
3   NaN 3.0 NaN 4
```

In [39]:

```
df = pd.read_csv('CleaningBook.csv')
df
```

Out[39]:

| | Roll_Number | Maths | Phy | Chem | Eng | Com | Grade | Total | Percentage |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 1.0 | 22.0 | 23.0 | 14.0 | 22.0 | 20.0 | B | 101.0 | 81.0 |
| **1** | 2.0 | 23.0 | 22.0 | NaN | 23.0 | 20.0 | B | 103.0 | 82.0 |
| **2** | 6.0 | 21.0 | NaN | 15.0 | 17.0 | NaN | D | 92.0 | 73.0 |
| **3** | 7.0 | 15.0 | 14.0 | 16.0 | 21.0 | 18.0 | E | 84.0 | 67.0 |
| **4** | 8.0 | NaN | 17.0 | NaN | 20.0 | NaN | E | 86.0 | 69.0 |
| **5** | 11.0 | 19.0 | 19.0 | 18.0 | 18.0 | 18.0 | D | 92.0 | 74.0 |
| **6** | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| **7** | 12.0 | 20.0 | 17.0 | 14.0 | 18.0 | 19.0 | D | 88.0 | 70.0 |
| **8** | 13.0 | 21.0 | 18.0 | 15.0 | 21.0 | 19.0 | C | 94.0 | 75.0 |
| **9** | 14.0 | 25.0 | 19.0 | 21.0 | 20.0 | 18.0 | B | 103.0 | 82.0 |
| **10** | 21.0 | 15.0 | 11.0 | 15.0 | 19.0 | 18.0 | F | 78.0 | 62.0 |
| **11** | 26.0 | 1.0 | 2.0 | 1.0 | 7.0 | 25.0 | O | 36.0 | 29.0 |
| **12** | 27.0 | 24.0 | 13.0 | 18.0 | 19.0 | 18.0 | D | 92.0 | 74.0 |
| **13** | 28.0 | 5.0 | 7.0 | 6.0 | NaN | 19.0 | O | 46.0 | 36.0 |
| **14** | 29.0 | 15.0 | 17.0 | 16.0 | 16.0 | 18.0 | E | 82.0 | 66.0 |
| **15** | 30.0 | 19.0 | 12.0 | 14.0 | 19.0 | 19.0 | E | 83.0 | 66.0 |
| **16** | 3.0 | 24.0 | 25.0 | 14.0 | 24.0 | 20.0 | A | 107.0 | 86.0 |
| **17** | 13.0 | 21.0 | 18.0 | 15.0 | 21.0 | 19.0 | C | 94.0 | 75.0 |

In [41]:

```
df = pd.read_csv('CleaningBook.csv')
x = df.drop(labels=[3,5,7],axis=0)
x
```

Out[41]:

| | Roll_Number | Maths | Phy | Chem | Eng | Com | Grade | Total | Percentage |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 1.0 | 22.0 | 23.0 | 14.0 | 22.0 | 20.0 | B | 101.0 | 81.0 |
| **1** | 2.0 | 23.0 | 22.0 | NaN | 23.0 | 20.0 | B | 103.0 | 82.0 |
| **2** | 6.0 | 21.0 | NaN | 15.0 | 17.0 | NaN | D | 92.0 | 73.0 |
| **4** | 8.0 | NaN | 17.0 | NaN | 20.0 | NaN | E | 86.0 | 69.0 |
| **6** | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| **8** | 13.0 | 21.0 | 18.0 | 15.0 | 21.0 | 19.0 | C | 94.0 | 75.0 |
| **9** | 14.0 | 25.0 | 19.0 | 21.0 | 20.0 | 18.0 | B | 103.0 | 82.0 |
| **10** | 21.0 | 15.0 | 11.0 | 15.0 | 19.0 | 18.0 | F | 78.0 | 62.0 |
| **11** | 26.0 | 1.0 | 2.0 | 1.0 | 7.0 | 25.0 | O | 36.0 | 29.0 |
| **12** | 27.0 | 24.0 | 13.0 | 18.0 | 19.0 | 18.0 | D | 92.0 | 74.0 |
| **13** | 28.0 | 5.0 | 7.0 | 6.0 | NaN | 19.0 | O | 46.0 | 36.0 |
| **14** | 29.0 | 15.0 | 17.0 | 16.0 | 16.0 | 18.0 | E | 82.0 | 66.0 |
| **15** | 30.0 | 19.0 | 12.0 | 14.0 | 19.0 | 19.0 | E | 83.0 | 66.0 |
| **16** | 3.0 | 24.0 | 25.0 | 14.0 | 24.0 | 20.0 | A | 107.0 | 86.0 |
| **17** | 13.0 | 21.0 | 18.0 | 15.0 | 21.0 | 19.0 | C | 94.0 | 75.0 |

In [50]:

```
df = pd.read_csv('CleaningBook.csv')

```

In [52]:
```
df.drop(labels=[3,5,7],axis=0, inplace=True)
df
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-52-a24ba1ca78a4> in <module>
----> 1 df.drop(labels=[3,5,7],axis=0, inplace=True)
      2 df

C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\frame.py in drop(self, labels, axis, index, columns, level, in
place, errors)
   4161                 weight  1.0     0.8
   4162         """
-> 4163         return super().drop(
   4164             labels=labels,
   4165             axis=axis,

C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\generic.py in drop(self, labels, axis, index, columns, level,
inplace, errors)
   3885         for axis, labels in axes.items():
   3886             if labels is not None:
-> 3887                 obj = obj._drop_axis(labels, axis, level=level, errors=errors)
   3888 
   3889         if inplace:

C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\generic.py in _drop_axis(self, labels, axis, level, errors)
   3919                 new_axis = axis.drop(labels, level=level, errors=errors)
   3920             else:
-> 3921                 new_axis = axis.drop(labels, errors=errors)
   3922             result = self.reindex(**{axis_name: new_axis})
   3923 

C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\indexes\base.py in drop(self, labels, errors)
   5280         if mask.any():
   5281             if errors != "ignore":
-> 5282                 raise KeyError(f"{labels[mask]} not found in axis")
   5283         indexer = indexer[~mask]
   5284         return self.delete(indexer)

KeyError: '[3 5 7] not found in axis'
```

In [ ]:
```
df = pd.read_csv('CleaningBook.csv')

```

In [54]:
```
df.drop(labels=[3,5,7],axis=0, inplace=True, errors='ignore')
df
```

Out[54]:

| | Roll_Number | Maths | Phy | Chem | Eng | Com | Grade | Total | Percentage |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 1.0 | 22.0 | 23.0 | 14.0 | 22.0 | 20.0 | B | 101.0 | 81.0 |
| **1** | 2.0 | 23.0 | 22.0 | NaN | 23.0 | 20.0 | B | 103.0 | 82.0 |
| **2** | 6.0 | 21.0 | NaN | 15.0 | 17.0 | NaN | D | 92.0 | 73.0 |
| **4** | 8.0 | NaN | 17.0 | NaN | 20.0 | NaN | E | 86.0 | 69.0 |
| **6** | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| **8** | 13.0 | 21.0 | 18.0 | 15.0 | 21.0 | 19.0 | C | 94.0 | 75.0 |
| **9** | 14.0 | 25.0 | 19.0 | 21.0 | 20.0 | 18.0 | B | 103.0 | 82.0 |
| **10** | 21.0 | 15.0 | 11.0 | 15.0 | 19.0 | 18.0 | F | 78.0 | 62.0 |
| **11** | 26.0 | 1.0 | 2.0 | 1.0 | 7.0 | 25.0 | O | 36.0 | 29.0 |
| **12** | 27.0 | 24.0 | 13.0 | 18.0 | 19.0 | 18.0 | D | 92.0 | 74.0 |
| **13** | 28.0 | 5.0 | 7.0 | 6.0 | NaN | 19.0 | O | 46.0 | 36.0 |
| **14** | 29.0 | 15.0 | 17.0 | 16.0 | 16.0 | 18.0 | E | 82.0 | 66.0 |
| **15** | 30.0 | 19.0 | 12.0 | 14.0 | 19.0 | 19.0 | E | 83.0 | 66.0 |
| **16** | 3.0 | 24.0 | 25.0 | 14.0 | 24.0 | 20.0 | A | 107.0 | 86.0 |
| **17** | 13.0 | 21.0 | 18.0 | 15.0 | 21.0 | 19.0 | C | 94.0 | 75.0 |

In [55]:
```
df = pd.read_csv('CleaningBook.csv')

```

In [57]:

```
df.drop(labels=[3,5,7],axis=0, inplace=True, errors='raise')
df
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-57-c1549c2c2799> in <module>
----> 1 df.drop(labels=[3,5,7],axis=0, inplace=True, errors='raise')
      2 df

C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\frame.py in drop(self, labels, axis, index, columns, level, in
place, errors)
   4161                 weight  1.0     0.8
   4162         """
-> 4163         return super().drop(
   4164             labels=labels,
   4165             axis=axis,

C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\generic.py in drop(self, labels, axis, index, columns, level,
inplace, errors)
   3885         for axis, labels in axes.items():
   3886             if labels is not None:
-> 3887                 obj = obj._drop_axis(labels, axis, level=level, errors=errors)
   3888
   3889         if inplace:

C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\generic.py in _drop_axis(self, labels, axis, level, errors)
   3919                 new_axis = axis.drop(labels, level=level, errors=errors)
   3920             else:
-> 3921                 new_axis = axis.drop(labels, errors=errors)
   3922             result = self.reindex(**{axis_name: new_axis})
   3923

C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\indexes\base.py in drop(self, labels, errors)
   5280         if mask.any():
   5281             if errors != "ignore":
-> 5282                 raise KeyError(f"{labels[mask]} not found in axis")
   5283             indexer = indexer[~mask]
   5284         return self.delete(indexer)

KeyError: '[3 5 7] not found in axis'
```

In [61]:

```
df = pd.read_csv('CleaningBook.csv')
df.drop(index=[1,4,7], columns=['Phy'],axis=0, inplace=True, errors='ignore')
df
```

Out[61]:

| | Roll_Number | Maths | Chem | Eng | Com | Grade | Total | Percentage |
|---|---|---|---|---|---|---|---|---|
| **0** | 1.0 | 22.0 | 14.0 | 22.0 | 20.0 | B | 101.0 | 81.0 |
| **2** | 6.0 | 21.0 | 15.0 | 17.0 | NaN | D | 92.0 | 73.0 |
| **3** | 7.0 | 15.0 | 16.0 | 21.0 | 18.0 | E | 84.0 | 67.0 |
| **5** | 11.0 | 19.0 | 18.0 | 18.0 | 18.0 | D | 92.0 | 74.0 |
| **6** | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| **8** | 13.0 | 21.0 | 15.0 | 21.0 | 19.0 | C | 94.0 | 75.0 |
| **9** | 14.0 | 25.0 | 21.0 | 20.0 | 18.0 | B | 103.0 | 82.0 |
| **10** | 21.0 | 15.0 | 15.0 | 19.0 | 18.0 | F | 78.0 | 62.0 |
| **11** | 26.0 | 1.0 | 1.0 | 7.0 | 25.0 | O | 36.0 | 29.0 |
| **12** | 27.0 | 24.0 | 18.0 | 19.0 | 18.0 | D | 92.0 | 74.0 |
| **13** | 28.0 | 5.0 | 6.0 | NaN | 19.0 | O | 46.0 | 36.0 |
| **14** | 29.0 | 15.0 | 16.0 | 16.0 | 18.0 | E | 82.0 | 66.0 |
| **15** | 30.0 | 19.0 | 14.0 | 19.0 | 19.0 | E | 83.0 | 66.0 |
| **16** | 3.0 | 24.0 | 14.0 | 24.0 | 20.0 | A | 107.0 | 86.0 |
| **17** | 13.0 | 21.0 | 15.0 | 21.0 | 19.0 | C | 94.0 | 75.0 |

## ValueError: Cannot specify both 'labels' and 'index'/'columns'

In [62]:

```
df = pd.read_csv('CleaningBook.csv')
df.drop(labels=[1,4,7], columns=['Phy'],axis=0, inplace=True, errors='ignore')
df
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-62-78fb24ba440a> in <module>
      1 df = pd.read_csv('CleaningBook.csv')
----> 2 df.drop(labels=[1,4,7], columns=['Phy'],axis=0, inplace=True, errors='ignore')
      3 df

C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\frame.py in drop(self, labels, axis, index, columns, level, in
place, errors)
   4161                 weight  1.0     0.8
   4162         """
-> 4163         return super().drop(
   4164             labels=labels,
   4165             axis=axis,

C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\generic.py in drop(self, labels, axis, index, columns, level,
inplace, errors)
   3871         if labels is not None:
   3872             if index is not None or columns is not None:
-> 3873                 raise ValueError("Cannot specify both 'labels' and 'index'/'columns'")
   3874             axis_name = self._get_axis_name(axis)
   3875             axes = {axis_name: labels}

ValueError: Cannot specify both 'labels' and 'index'/'columns'
```

# Handling duplication

In [65]:

```python
df = pd.read_csv('CleaningBook.csv')
print(df)
df.drop_duplicates(inplace=True)
df
# 8 & 17 are same - 17 will be deleted
```

```
    Roll_Number  Maths   Phy  Chem   Eng   Com Grade  Total  Percentage
0           1.0   22.0  23.0  14.0  22.0  20.0     B  101.0        81.0
1           2.0   23.0  22.0   NaN  23.0  20.0     B  103.0        82.0
2           6.0   21.0   NaN  15.0  17.0   NaN     D   92.0        73.0
3           7.0   15.0  14.0  16.0  21.0  18.0     E   84.0        67.0
4           8.0    NaN  17.0   NaN  20.0   NaN     E   86.0        69.0
5          11.0   19.0  19.0  18.0  18.0  18.0     D   92.0        74.0
6           NaN    NaN   NaN   NaN   NaN   NaN   NaN    NaN         NaN
7          12.0   20.0  17.0  14.0  18.0  19.0     D   88.0        70.0
8          13.0   21.0  18.0  15.0  21.0  19.0     C   94.0        75.0
9          14.0   25.0  19.0  21.0  20.0  18.0     B  103.0        82.0
10         21.0   15.0  11.0  15.0  19.0  18.0     F   78.0        62.0
11         26.0    1.0   2.0   1.0   7.0  25.0     O   36.0        29.0
12         27.0   24.0  13.0  18.0  19.0  18.0     D   92.0        74.0
13         28.0    5.0   7.0   6.0   NaN  19.0     O   46.0        36.0
14         29.0   15.0  17.0  16.0  16.0  18.0     E   82.0        66.0
15         30.0   19.0  12.0  14.0  19.0  19.0     E   83.0        66.0
16          3.0   24.0  25.0  14.0  24.0  20.0     A  107.0        86.0
17         13.0   21.0  18.0  15.0  21.0  19.0     C   94.0        75.0
```

Out[65]:

| | Roll_Number | Maths | Phy | Chem | Eng | Com | Grade | Total | Percentage |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 1.0 | 22.0 | 23.0 | 14.0 | 22.0 | 20.0 | B | 101.0 | 81.0 |
| **1** | 2.0 | 23.0 | 22.0 | NaN | 23.0 | 20.0 | B | 103.0 | 82.0 |
| **2** | 6.0 | 21.0 | NaN | 15.0 | 17.0 | NaN | D | 92.0 | 73.0 |
| **3** | 7.0 | 15.0 | 14.0 | 16.0 | 21.0 | 18.0 | E | 84.0 | 67.0 |
| **4** | 8.0 | NaN | 17.0 | NaN | 20.0 | NaN | E | 86.0 | 69.0 |
| **5** | 11.0 | 19.0 | 19.0 | 18.0 | 18.0 | 18.0 | D | 92.0 | 74.0 |
| **6** | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| **7** | 12.0 | 20.0 | 17.0 | 14.0 | 18.0 | 19.0 | D | 88.0 | 70.0 |
| **8** | 13.0 | 21.0 | 18.0 | 15.0 | 21.0 | 19.0 | C | 94.0 | 75.0 |
| **9** | 14.0 | 25.0 | 19.0 | 21.0 | 20.0 | 18.0 | B | 103.0 | 82.0 |
| **10** | 21.0 | 15.0 | 11.0 | 15.0 | 19.0 | 18.0 | F | 78.0 | 62.0 |
| **11** | 26.0 | 1.0 | 2.0 | 1.0 | 7.0 | 25.0 | O | 36.0 | 29.0 |
| **12** | 27.0 | 24.0 | 13.0 | 18.0 | 19.0 | 18.0 | D | 92.0 | 74.0 |
| **13** | 28.0 | 5.0 | 7.0 | 6.0 | NaN | 19.0 | O | 46.0 | 36.0 |
| **14** | 29.0 | 15.0 | 17.0 | 16.0 | 16.0 | 18.0 | E | 82.0 | 66.0 |
| **15** | 30.0 | 19.0 | 12.0 | 14.0 | 19.0 | 19.0 | E | 83.0 | 66.0 |
| **16** | 3.0 | 24.0 | 25.0 | 14.0 | 24.0 | 20.0 | A | 107.0 | 86.0 |

In [69]:

```python
df = pd.read_csv('CleaningBook.csv')
df.drop_duplicates(inplace=True, keep='last')
df
# 8 will be deleted
```

Out[69]:

| | Roll_Number | Maths | Phy | Chem | Eng | Com | Grade | Total | Percentage |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 1.0 | 22.0 | 23.0 | 14.0 | 22.0 | 20.0 | B | 101.0 | 81.0 |
| **1** | 2.0 | 23.0 | 22.0 | NaN | 23.0 | 20.0 | B | 103.0 | 82.0 |
| **2** | 6.0 | 21.0 | NaN | 15.0 | 17.0 | NaN | D | 92.0 | 73.0 |
| **3** | 7.0 | 15.0 | 14.0 | 16.0 | 21.0 | 18.0 | E | 84.0 | 67.0 |
| **4** | 8.0 | NaN | 17.0 | NaN | 20.0 | NaN | E | 86.0 | 69.0 |
| **5** | 11.0 | 19.0 | 19.0 | 18.0 | 18.0 | 18.0 | D | 92.0 | 74.0 |
| **6** | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| **7** | 12.0 | 20.0 | 17.0 | 14.0 | 18.0 | 19.0 | D | 88.0 | 70.0 |
| **9** | 14.0 | 25.0 | 19.0 | 21.0 | 20.0 | 18.0 | B | 103.0 | 82.0 |
| **10** | 21.0 | 15.0 | 11.0 | 15.0 | 19.0 | 18.0 | F | 78.0 | 62.0 |
| **11** | 26.0 | 1.0 | 2.0 | 1.0 | 7.0 | 25.0 | O | 36.0 | 29.0 |
| **12** | 27.0 | 24.0 | 13.0 | 18.0 | 19.0 | 18.0 | D | 92.0 | 74.0 |
| **13** | 28.0 | 5.0 | 7.0 | 6.0 | NaN | 19.0 | O | 46.0 | 36.0 |
| **14** | 29.0 | 15.0 | 17.0 | 16.0 | 16.0 | 18.0 | E | 82.0 | 66.0 |
| **15** | 30.0 | 19.0 | 12.0 | 14.0 | 19.0 | 19.0 | E | 83.0 | 66.0 |
| **16** | 3.0 | 24.0 | 25.0 | 14.0 | 24.0 | 20.0 | A | 107.0 | 86.0 |
| **17** | 13.0 | 21.0 | 18.0 | 15.0 | 21.0 | 19.0 | C | 94.0 | 75.0 |

In [70]:

```
df = pd.read_csv('CleaningBook.csv')
df.drop_duplicates(inplace=True, keep='first')
df
# 17 will be deleted
```

Out[70]:

| | Roll_Number | Maths | Phy | Chem | Eng | Com | Grade | Total | Percentage |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 1.0 | 22.0 | 23.0 | 14.0 | 22.0 | 20.0 | B | 101.0 | 81.0 |
| **1** | 2.0 | 23.0 | 22.0 | NaN | 23.0 | 20.0 | B | 103.0 | 82.0 |
| **2** | 6.0 | 21.0 | NaN | 15.0 | 17.0 | NaN | D | 92.0 | 73.0 |
| **3** | 7.0 | 15.0 | 14.0 | 16.0 | 21.0 | 18.0 | E | 84.0 | 67.0 |
| **4** | 8.0 | NaN | 17.0 | NaN | 20.0 | NaN | E | 86.0 | 69.0 |
| **5** | 11.0 | 19.0 | 19.0 | 18.0 | 18.0 | 18.0 | D | 92.0 | 74.0 |
| **6** | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| **7** | 12.0 | 20.0 | 17.0 | 14.0 | 18.0 | 19.0 | D | 88.0 | 70.0 |
| **8** | 13.0 | 21.0 | 18.0 | 15.0 | 21.0 | 19.0 | C | 94.0 | 75.0 |
| **9** | 14.0 | 25.0 | 19.0 | 21.0 | 20.0 | 18.0 | B | 103.0 | 82.0 |
| **10** | 21.0 | 15.0 | 11.0 | 15.0 | 19.0 | 18.0 | F | 78.0 | 62.0 |
| **11** | 26.0 | 1.0 | 2.0 | 1.0 | 7.0 | 25.0 | O | 36.0 | 29.0 |
| **12** | 27.0 | 24.0 | 13.0 | 18.0 | 19.0 | 18.0 | D | 92.0 | 74.0 |
| **13** | 28.0 | 5.0 | 7.0 | 6.0 | NaN | 19.0 | O | 46.0 | 36.0 |
| **14** | 29.0 | 15.0 | 17.0 | 16.0 | 16.0 | 18.0 | E | 82.0 | 66.0 |
| **15** | 30.0 | 19.0 | 12.0 | 14.0 | 19.0 | 19.0 | E | 83.0 | 66.0 |
| **16** | 3.0 | 24.0 | 25.0 | 14.0 | 24.0 | 20.0 | A | 107.0 | 86.0 |

In [72]:

```
df = pd.read_csv('CleaningBook.csv')
df.drop_duplicates(inplace=True, keep=False)
df
# 8 & 17 both are deleted.
```

Out[72]:

| | Roll_Number | Maths | Phy | Chem | Eng | Com | Grade | Total | Percentage |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 1.0 | 22.0 | 23.0 | 14.0 | 22.0 | 20.0 | B | 101.0 | 81.0 |
| **1** | 2.0 | 23.0 | 22.0 | NaN | 23.0 | 20.0 | B | 103.0 | 82.0 |
| **2** | 6.0 | 21.0 | NaN | 15.0 | 17.0 | NaN | D | 92.0 | 73.0 |
| **3** | 7.0 | 15.0 | 14.0 | 16.0 | 21.0 | 18.0 | E | 84.0 | 67.0 |
| **4** | 8.0 | NaN | 17.0 | NaN | 20.0 | NaN | E | 86.0 | 69.0 |
| **5** | 11.0 | 19.0 | 19.0 | 18.0 | 18.0 | 18.0 | D | 92.0 | 74.0 |
| **6** | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| **7** | 12.0 | 20.0 | 17.0 | 14.0 | 18.0 | 19.0 | D | 88.0 | 70.0 |
| **9** | 14.0 | 25.0 | 19.0 | 21.0 | 20.0 | 18.0 | B | 103.0 | 82.0 |
| **10** | 21.0 | 15.0 | 11.0 | 15.0 | 19.0 | 18.0 | F | 78.0 | 62.0 |
| **11** | 26.0 | 1.0 | 2.0 | 1.0 | 7.0 | 25.0 | O | 36.0 | 29.0 |
| **12** | 27.0 | 24.0 | 13.0 | 18.0 | 19.0 | 18.0 | D | 92.0 | 74.0 |
| **13** | 28.0 | 5.0 | 7.0 | 6.0 | NaN | 19.0 | O | 46.0 | 36.0 |
| **14** | 29.0 | 15.0 | 17.0 | 16.0 | 16.0 | 18.0 | E | 82.0 | 66.0 |
| **15** | 30.0 | 19.0 | 12.0 | 14.0 | 19.0 | 19.0 | E | 83.0 | 66.0 |
| **16** | 3.0 | 24.0 | 25.0 | 14.0 | 24.0 | 20.0 | A | 107.0 | 86.0 |

In [78]:

```
df = pd.read_csv('CleaningBook.csv')
x = df.drop_duplicates(keep=False, subset=['Phy'])
x
```

Out[78]:

| | Roll_Number | Maths | Phy | Chem | Eng | Com | Grade | Total | Percentage |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 1.0 | 22.0 | 23.0 | 14.0 | 22.0 | 20.0 | B | 101.0 | 81.0 |
| **1** | 2.0 | 23.0 | 22.0 | NaN | 23.0 | 20.0 | B | 103.0 | 82.0 |
| **3** | 7.0 | 15.0 | 14.0 | 16.0 | 21.0 | 18.0 | E | 84.0 | 67.0 |
| **10** | 21.0 | 15.0 | 11.0 | 15.0 | 19.0 | 18.0 | F | 78.0 | 62.0 |
| **11** | 26.0 | 1.0 | 2.0 | 1.0 | 7.0 | 25.0 | O | 36.0 | 29.0 |
| **12** | 27.0 | 24.0 | 13.0 | 18.0 | 19.0 | 18.0 | D | 92.0 | 74.0 |
| **13** | 28.0 | 5.0 | 7.0 | 6.0 | NaN | 19.0 | O | 46.0 | 36.0 |
| **15** | 30.0 | 19.0 | 12.0 | 14.0 | 19.0 | 19.0 | E | 83.0 | 66.0 |
| **16** | 3.0 | 24.0 | 25.0 | 14.0 | 24.0 | 20.0 | A | 107.0 | 86.0 |

In [79]:

```python
df = pd.read_csv('CleaningBook.csv')
x = df.drop_duplicates(keep=False, subset=['Phy'])
print(x)
nx = x.reset_index(drop=True)
nx
```

```
    Roll_Number  Maths   Phy  Chem   Eng   Com Grade  Total  Percentage
0           1.0   22.0  23.0  14.0  22.0  20.0     B  101.0        81.0
1           2.0   23.0  22.0   NaN  23.0  20.0     B  103.0        82.0
3           7.0   15.0  14.0  16.0  21.0  18.0     E   84.0        67.0
10         21.0   15.0  11.0  15.0  19.0  18.0     F   78.0        62.0
11         26.0    1.0   2.0   1.0   7.0  25.0     O   36.0        29.0
12         27.0   24.0  13.0  18.0  19.0  18.0     D   92.0        74.0
13         28.0    5.0   7.0   6.0   NaN  19.0     O   46.0        36.0
15         30.0   19.0  12.0  14.0  19.0  19.0     E   83.0        66.0
16          3.0   24.0  25.0  14.0  24.0  20.0     A  107.0        86.0
```

Out[79]:

| | Roll_Number | Maths | Phy | Chem | Eng | Com | Grade | Total | Percentage |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 1.0 | 22.0 | 23.0 | 14.0 | 22.0 | 20.0 | B | 101.0 | 81.0 |
| **1** | 2.0 | 23.0 | 22.0 | NaN | 23.0 | 20.0 | B | 103.0 | 82.0 |
| **2** | 7.0 | 15.0 | 14.0 | 16.0 | 21.0 | 18.0 | E | 84.0 | 67.0 |
| **3** | 21.0 | 15.0 | 11.0 | 15.0 | 19.0 | 18.0 | F | 78.0 | 62.0 |
| **4** | 26.0 | 1.0 | 2.0 | 1.0 | 7.0 | 25.0 | O | 36.0 | 29.0 |
| **5** | 27.0 | 24.0 | 13.0 | 18.0 | 19.0 | 18.0 | D | 92.0 | 74.0 |
| **6** | 28.0 | 5.0 | 7.0 | 6.0 | NaN | 19.0 | O | 46.0 | 36.0 |
| **7** | 30.0 | 19.0 | 12.0 | 14.0 | 19.0 | 19.0 | E | 83.0 | 66.0 |
| **8** | 3.0 | 24.0 | 25.0 | 14.0 | 24.0 | 20.0 | A | 107.0 | 86.0 |

# P.b. -36

In [85]:
```
# que.1
df = pd.read_csv('Datasets/movies.csv')
df.loc[df['year_of_release']==2019]
```

Out[85]:

| | title_x | imdb_id | poster_path | wiki_link | title_y | original_title | is_a |
|---|---|---|---|---|---|---|---|
| **0** | Uri: The Surgical Strike | tt8291224 | https://upload.wikimedia.org/wikipedia/en/thum... | https://en.wikipedia.org/wiki/Uri:_The_Surgica... | Uri: The Surgical Strike | Uri: The Surgical Strike | |
| **1** | Battalion 609 | tt9472208 | NaN | https://en.wikipedia.org/wiki/Battalion_609 | Battalion 609 | Battalion 609 | |
| **2** | The Accidental Prime Minister (film) | tt6986710 | https://upload.wikimedia.org/wikipedia/en/thum... | https://en.wikipedia.org/wiki/The_Accidental_P... | The Accidental Prime Minister | The Accidental Prime Minister | |
| **3** | Why Cheat India | tt8108208 | https://upload.wikimedia.org/wikipedia/en/thum... | https://en.wikipedia.org/wiki/Why_Cheat_India | Why Cheat India | Why Cheat India | |
| **6** | Fraud Saiyaan | tt5013008 | https://upload.wikimedia.org/wikipedia/en/thum... | https://en.wikipedia.org/wiki/Fraud_Saiyaan | Fraud Saiyaan | Fraud Saiyyan | |
| **...** | ... | ... | ... | ... | ... | ... | |
| **76** | Commando 3 (film) | tt8983168 | https://upload.wikimedia.org/wikipedia/en/thum... | https://en.wikipedia.org/wiki/Commando_3_(film) | Commando 3 | Commando 3 | |
| **77** | Mardaani 2 | tt5668770 | https://upload.wikimedia.org/wikipedia/en/thum... | https://en.wikipedia.org/wiki/Mardaani_2 | Mardaani 2 | Mardaani 2 | |
| **78** | Dabangg 3 | tt7059844 | https://upload.wikimedia.org/wikipedia/en/thum... | https://en.wikipedia.org/wiki/Dabangg_3 | Dabangg 3 | Dabangg 3 | |
| **79** | Good Newwz | tt8504014 | NaN | https://en.wikipedia.org/wiki/Good_Newwz | Good Newwz | Good Newwz | |
| **1627** | Daaka | tt10833860 | https://upload.wikimedia.org/wikipedia/en/thum... | https://en.wikipedia.org/wiki/Daaka | Daaka | Daaka | |

75 rows × 18 columns

In [100]:
```
# que.2
x = df.loc[df['imdb_rating']>7]
len(x)

# or
# x = df.loc[df['imdb_rating']>7].shape[0]
# x
```

Out[100]:
```
331
```

In [111]:

```python
# que.3
x = df.loc[df['imdb_votes']>20000]
nx = x[['title_y', 'story']]
nx
```

Out[111]:

| | title_y | story |
|---|---|---|
| **0** | Uri: The Surgical Strike | Divided over five chapters the film chronicle... |
| **11** | Gully Boy | Gully Boy is a film about a 22-year-old boy "M... |
| **36** | Kabir Singh | This Sandeep Vanga directorial is a remake of ... |
| **74** | Dil Bechara | A love story about two cancer patients. |
| **88** | Padmaavat | This fictional story is set in 13th century me... |
| **...** | ... | ... |
| **1490** | Devdas | Devdas Mukherji is black-listed by his multi-m... |
| **1565** | Kabhi Khushi Kabhie Gham... | Yashvardhan Raichand lives a very wealthy life... |
| **1567** | Lagaan: Once Upon a Time in India | This is the story about the resilience shown b... |
| **1568** | Lagaan: Once Upon a Time in India | This is the story about the resilience shown b... |
| **1571** | Dil Chahta Hai | Three young men Akash Sameer and Siddharth a... |

105 rows × 2 columns

In [114]:

```python
# que.4
x = df.loc[df['year_of_release']==2018]
nx = x[['title_y', 'year_of_release']]
nx
```

Out[114]:

| | title_y | year_of_release |
|---|---|---|
| **4** | Evening Shadows | 2018 |
| **5** | Soni | 2018 |
| **16** | Mard Ko Dard Nahin Hota | 2018 |
| **17** | Hamid | 2018 |
| **20** | Mere Pyare Prime Minister | 2018 |
| **...** | ... | ... |
| **156** | Rajma Chawal | 2018 |
| **157** | Zero | 2018 |
| **158** | Simmba | 2018 |
| **166** | Thugs of Hindostan | 2018 |
| **1626** | Sabse Bada Sukh | 2018 |

79 rows × 2 columns

In [116]:

```python
# que.5
df[['title_y','wiki_link']]
```

Out[116]:

| | title_y | wiki_link |
|---|---|---|
| **0** | Uri: The Surgical Strike | https://en.wikipedia.org/wiki/Uri:_The_Surgica... |
| **1** | Battalion 609 | https://en.wikipedia.org/wiki/Battalion_609 |
| **2** | The Accidental Prime Minister | https://en.wikipedia.org/wiki/The_Accidental_P... |
| **3** | Why Cheat India | https://en.wikipedia.org/wiki/Why_Cheat_India |
| **4** | Evening Shadows | https://en.wikipedia.org/wiki/Evening_Shadows |
| **...** | ... | ... |
| **1624** | Tera Mera Saath Rahen | https://en.wikipedia.org/wiki/Tera_Mera_Saath_... |
| **1625** | Yeh Zindagi Ka Safar | https://en.wikipedia.org/wiki/Yeh_Zindagi_Ka_S... |
| **1626** | Sabse Bada Sukh | https://en.wikipedia.org/wiki/Sabse_Bada_Sukh |
| **1627** | Daaka | https://en.wikipedia.org/wiki/Daaka |
| **1628** | Humsafar | https://en.wikipedia.org/wiki/Humsafar |

1629 rows × 2 columns

# Statistical Analysis

1. `corr()`
2. `scatter_matrix`
3. `parallel_coordinates`
4. `describe()`

In [1]:

```python
import pandas as pd
df = pd.read_csv('Datasets/auto-mpg.csv')
df
```

Out[1]:

| | 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 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 |

398 rows × 9 columns

-> The .corr() function in Pandas calculates the pairwise correlation of columns in a DataFrame, providing a statistical measure of the strength and direction of linear relationships between variables.

In [3]:

```python
df = pd.read_csv('Datasets/auto-mpg.csv')
df.corr()
```

Out[3]:

| | 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 [4]:

```python
df = pd.read_csv('Datasets/auto-mpg.csv')
df.describe()
```

Out[4]:

| | 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 [7]:

```python
df = pd.read_csv('Datasets/auto-mpg.csv')
df.describe(include=['O'])
```

Out[7]:

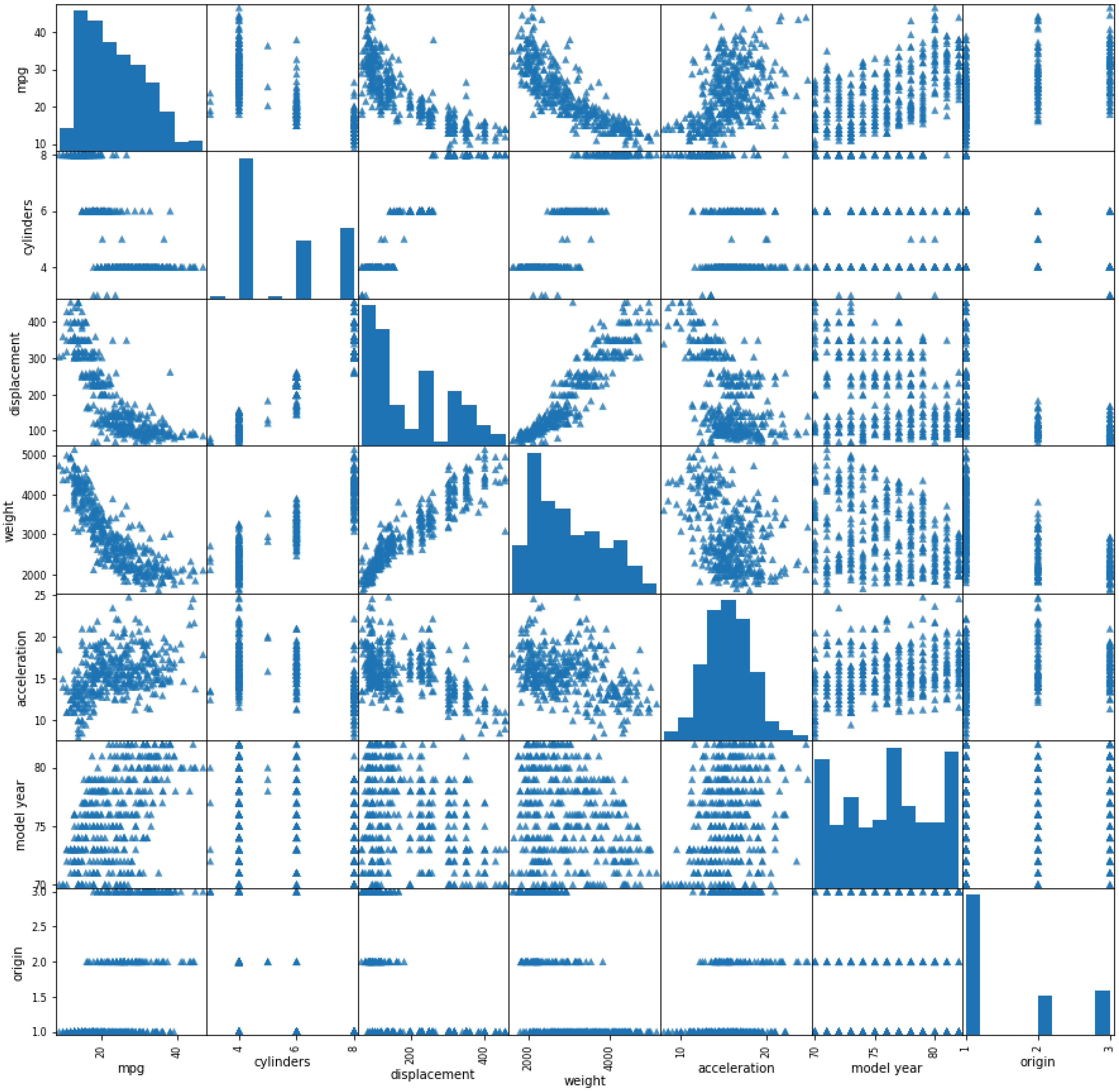
| | horsepower | car name |
|---|---|---|
| **count** | 398 | 398 |
| **unique** | 94 | 305 |
| **top** | 150 | ford pinto |
| **freq** | 22 | 6 |

In [9]:

```
df = pd.read_csv('Datasets/auto-mpg.csv')
pd.plotting.scatter_matrix(df, figsize=[15,15], marker='^', alpha=0.8)
```

Out[9]:

```
array([[<AxesSubplot:xlabel='mpg', ylabel='mpg'>,
        <AxesSubplot:xlabel='cylinders', ylabel='mpg'>,
        <AxesSubplot:xlabel='displacement', ylabel='mpg'>,
        <AxesSubplot:xlabel='weight', ylabel='mpg'>,
        <AxesSubplot:xlabel='acceleration', ylabel='mpg'>,
        <AxesSubplot:xlabel='model year', ylabel='mpg'>,
        <AxesSubplot:xlabel='origin', ylabel='mpg'>],
       [<AxesSubplot:xlabel='mpg', ylabel='cylinders'>,
        <AxesSubplot:xlabel='cylinders', ylabel='cylinders'>,
        <AxesSubplot:xlabel='displacement', ylabel='cylinders'>,
        <AxesSubplot:xlabel='weight', ylabel='cylinders'>,
        <AxesSubplot:xlabel='acceleration', ylabel='cylinders'>,
        <AxesSubplot:xlabel='model year', ylabel='cylinders'>,
        <AxesSubplot:xlabel='origin', ylabel='cylinders'>],
       [<AxesSubplot:xlabel='mpg', ylabel='displacement'>,
        <AxesSubplot:xlabel='cylinders', ylabel='displacement'>,
        <AxesSubplot:xlabel='displacement', ylabel='displacement'>,
        <AxesSubplot:xlabel='weight', ylabel='displacement'>,
        <AxesSubplot:xlabel='acceleration', ylabel='displacement'>,
        <AxesSubplot:xlabel='model year', ylabel='displacement'>,
        <AxesSubplot:xlabel='origin', ylabel='displacement'>],
       [<AxesSubplot:xlabel='mpg', ylabel='weight'>,
        <AxesSubplot:xlabel='cylinders', ylabel='weight'>,
        <AxesSubplot:xlabel='displacement', ylabel='weight'>,
        <AxesSubplot:xlabel='weight', ylabel='weight'>,
        <AxesSubplot:xlabel='acceleration', ylabel='weight'>,
        <AxesSubplot:xlabel='model year', ylabel='weight'>,
        <AxesSubplot:xlabel='origin', ylabel='weight'>],
       [<AxesSubplot:xlabel='mpg', ylabel='acceleration'>,
        <AxesSubplot:xlabel='cylinders', ylabel='acceleration'>,
        <AxesSubplot:xlabel='displacement', ylabel='acceleration'>,
        <AxesSubplot:xlabel='weight', ylabel='acceleration'>,
        <AxesSubplot:xlabel='acceleration', ylabel='acceleration'>,
        <AxesSubplot:xlabel='model year', ylabel='acceleration'>,
        <AxesSubplot:xlabel='origin', ylabel='acceleration'>],
       [<AxesSubplot:xlabel='mpg', ylabel='model year'>,
        <AxesSubplot:xlabel='cylinders', ylabel='model year'>,
        <AxesSubplot:xlabel='displacement', ylabel='model year'>,
        <AxesSubplot:xlabel='weight', ylabel='model year'>,
        <AxesSubplot:xlabel='acceleration', ylabel='model year'>,
        <AxesSubplot:xlabel='model year', ylabel='model year'>,
        <AxesSubplot:xlabel='origin', ylabel='model year'>],
       [<AxesSubplot:xlabel='mpg', ylabel='origin'>,
        <AxesSubplot:xlabel='cylinders', ylabel='origin'>,
        <AxesSubplot:xlabel='displacement', ylabel='origin'>,
        <AxesSubplot:xlabel='weight', ylabel='origin'>,
        <AxesSubplot:xlabel='acceleration', ylabel='origin'>,
        <AxesSubplot:xlabel='model year', ylabel='origin'>,
        <AxesSubplot:xlabel='origin', ylabel='origin'>]], dtype=object)
```


mpg
cylinders
displacement
weight
acceleration
model year
origin
10
20
30
40
4
6
8
100
200
300
400
2000
3000
4000
5000
15
25
70
75
80
1.0
1.5
2.0
2.5
3.0
1
2
3

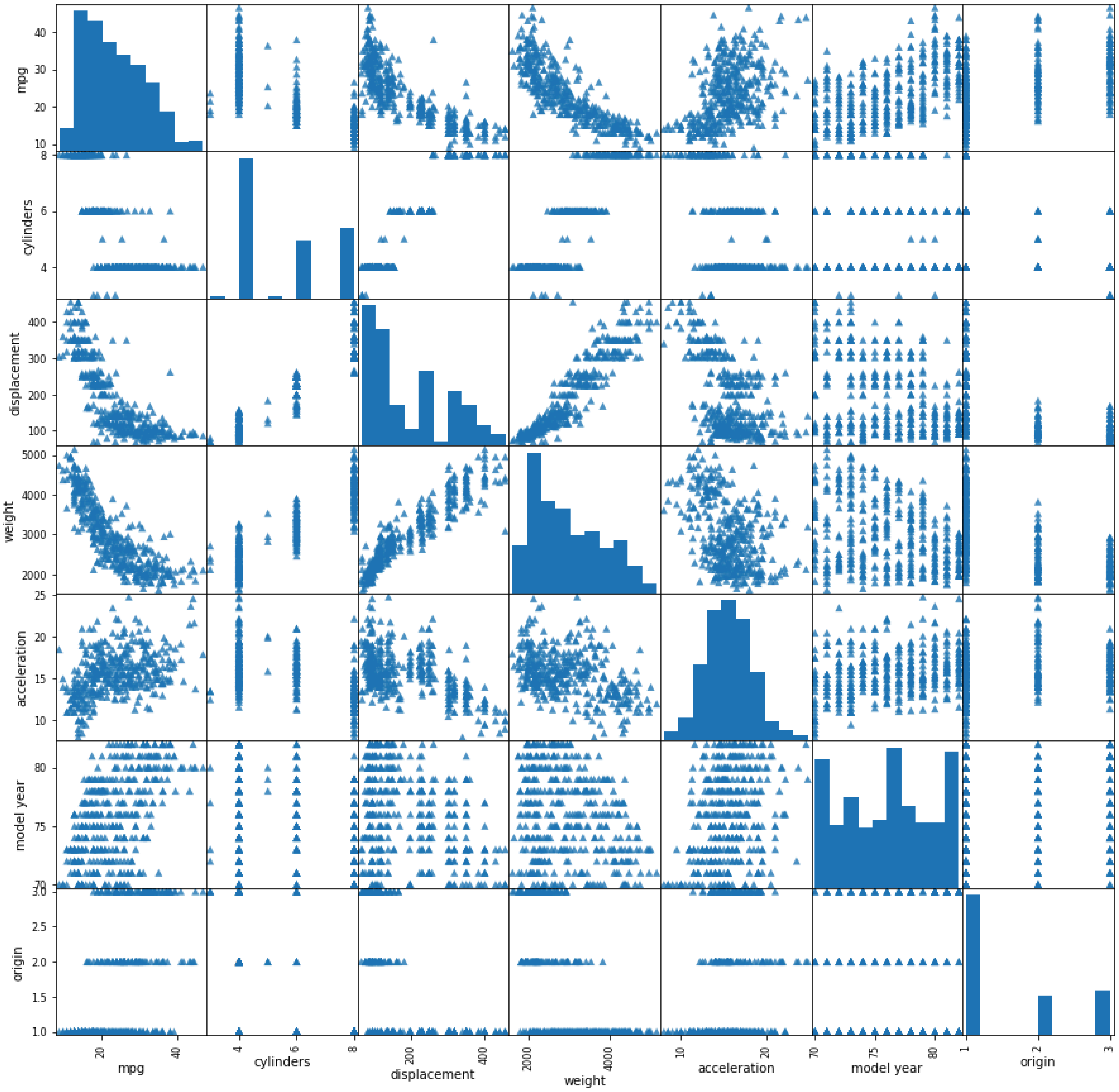
In [10]:

```
df = pd.read_csv('Datasets/auto-mpg.csv')
pd.plotting.scatter_matrix(df, figsize=[15,15], marker='^', alpha=0.8)
df
```

Out[10]:

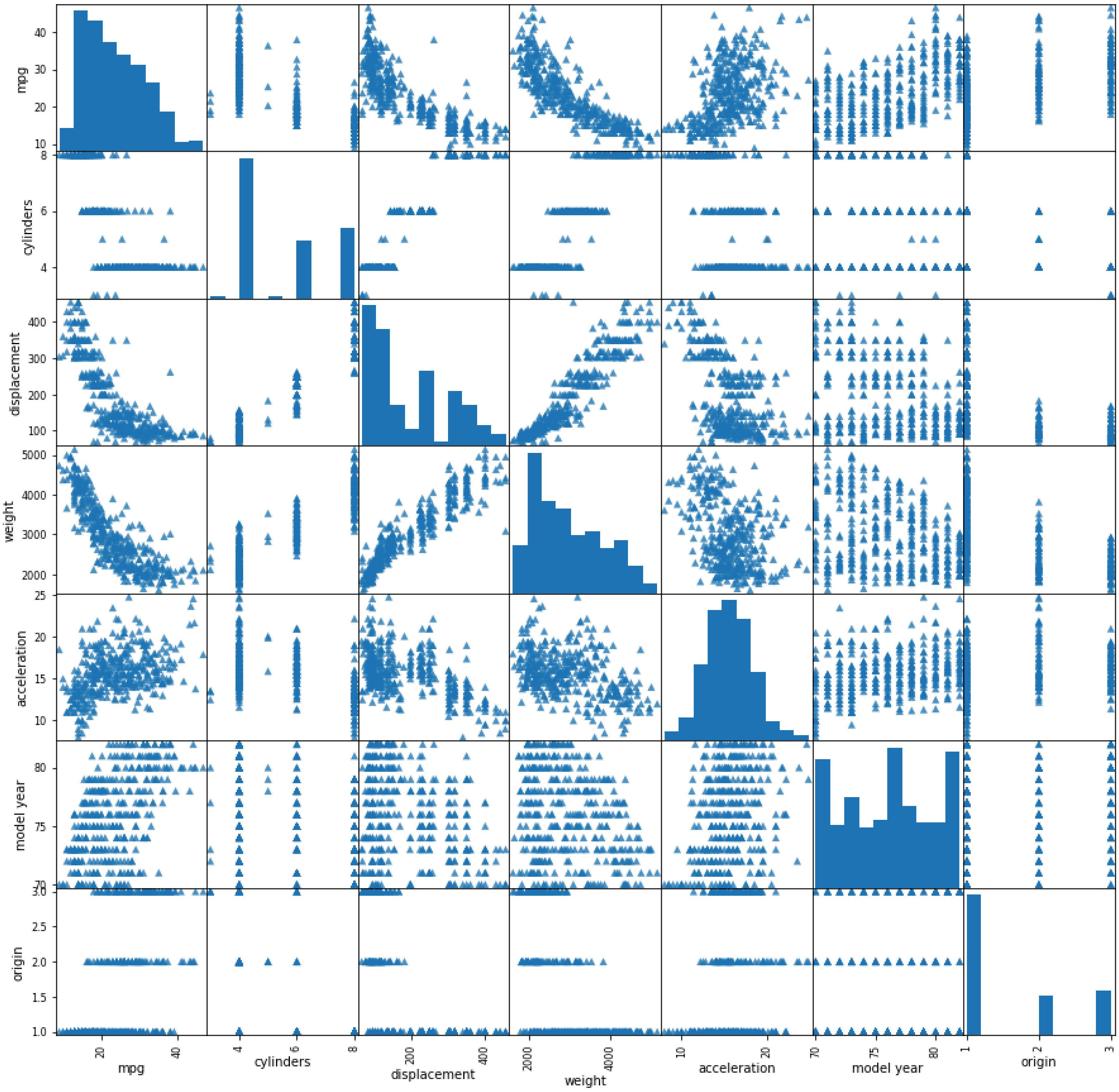
| | mpg | cylinders | displacement | horsepower | weight | acceleration | model year | origin | car name |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 18.0 | 8 | 307.0 | 130 | 3504 | 12.0 | 70 | 1 | chevrolet chevelle malibu |
| **1** | 15.0 | 8 | 350.0 | 165 | 3693 | 11.5 | 70 | 1 | buick skylark 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 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 |

398 rows × 9 columns



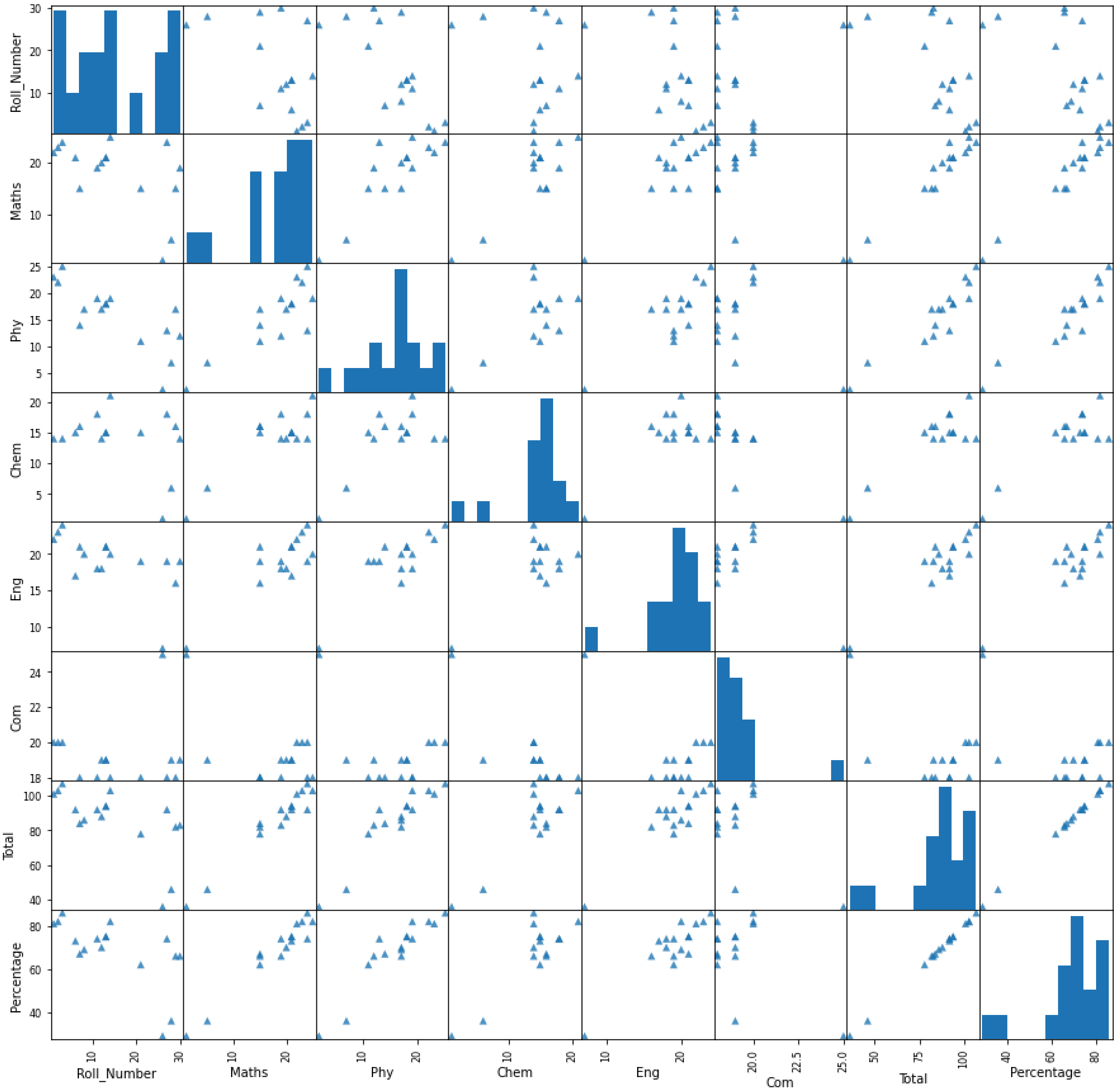
In [15]:

```
import matplotlib.pyplot as plt
df = pd.read_csv('Datasets/auto-mpg.csv')
pd.plotting.scatter_matrix(df, figsize=[15,15], marker='^', alpha=0.8)
plt.show()
```



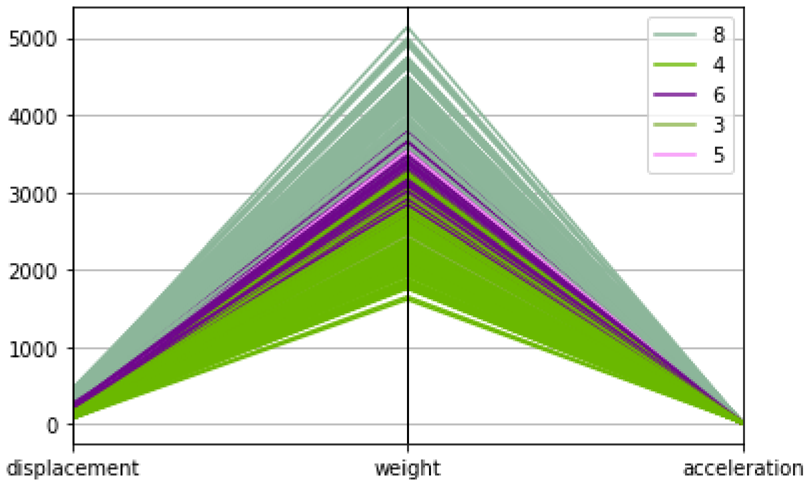
In [16]:

```python
import matplotlib.pyplot as plt
df = pd.read_csv('CleaningBook.csv')
pd.plotting.scatter_matrix(df, figsize=[15,15], marker='^', alpha=0.8)
plt.show()
```



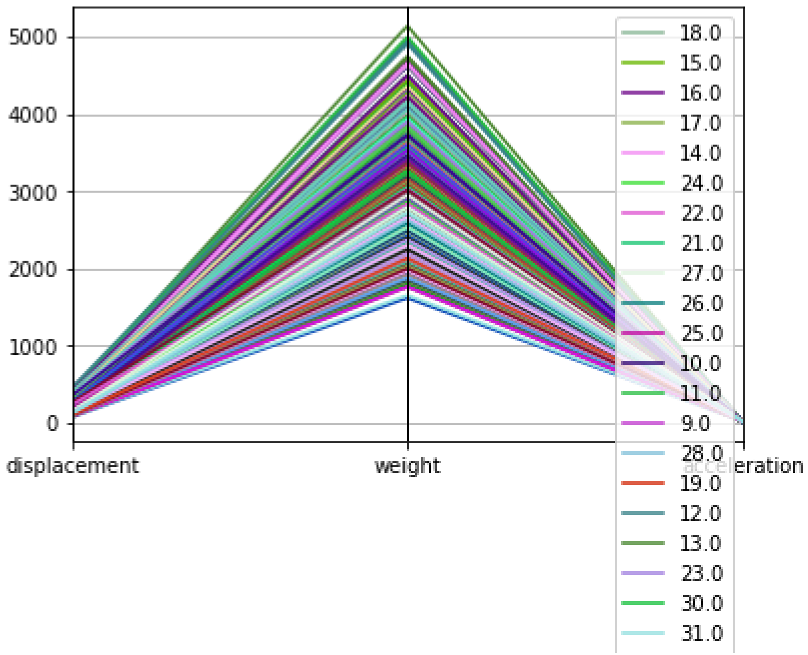
In [20]:

```python
import matplotlib.pyplot as plt
df = pd.read_csv('Datasets/auto-mpg.csv')
pd.plotting.parallel_coordinates(df, class_column='cylinders',cols=['displacement','weight','acceleration'])
plt.show()
```



In [23]:

```python
import matplotlib.pyplot as plt
df = pd.read_csv('Datasets/auto-mpg.csv')
pd.plotting.parallel_coordinates(df, class_column='mpg',cols=['displacement','weight','acceleration'])
plt.show()
```



- ## difference between qualitative and quantitative data

```
-> Qualitative data describes qualities or characteristics (e.g., colors, opinions)
-> Quantitative data is numerical and measurable (e.g., height, weight).
```

- ## qualitative

```
-> use methods like interviews, partipant obesravtion, focus on a grouping to gain collective informaiton
-> Data format is textual data Datasheets may content audios, video recordings or notes.
-> Explains the questions like why and how(talks about the experience and quality)
-> Data is Analysed by grouping into different categories
-> qualitative are subjective & can be further open for interpretesion
```

- ## quantitative

```
-> uses methods as questionnaire as surveys and structral observation to gain collective information
-> Data format is Numerical
-> Explains the questions like how much, how many(talks about quantity)
-> Data is analyshed by statistical method
-> quantitative data are fixed & Universal.
```

In [31]:

```
import pandas as pd

df = pd.read_csv('Info.csv')
df
```

Out[31]:

|  | Cid | Bill | Tip | Gender | Day | Time | Smoker |
|---|---|---|---|---|---|---|---|
| **0** | 1 | 1500 | 10 | M | Sun | Dinner | Y |
| **1** | 2 | 2500 | 20 | F | Mon | Dinner | Y |
| **2** | 3 | 1850 | 10 | M | Sun | Dinner | N |
| **3** | 4 | 1259 | 20 | F | Sun | Dinner | N |
| **4** | 5 | 5698 | 100 | M | Tue | Dinner | Y |
| **5** | 6 | 2568 | 50 | M | Sun | Dinner | Y |
| **6** | 7 | 4587 | 150 | M | Tue | Lunch | N |
| **7** | 8 | 1258 | 20 | F | Wed | Lunch | N |
| **8** | 9 | 1200 | 20 | F | Wed | Lunch | N |
| **9** | 10 | 1700 | 10 | F | Tue | Lunch | Y |
| **10** | 11 | 2000 | 20 | M | Tue | Dinner | Y |
| **11** | 12 | 2593 | 50 | F | Wed | Dinner | N |
| **12** | 13 | 1569 | 100 | M | Wed | Lunch | N |
| **13** | 14 | 1200 | 10 | F | Tue | Lunch | N |
| **14** | 15 | 1600 | 60 | F | Sun | Dinner | Y |

In [28]:

```
import pandas as pd

df = pd.read_csv('Info.csv')
x = pd.crosstab(df['Gender'],df['Smoker'])
x
```

Out[28]:

| **Smoker** | **N** | **Y** |
|---|---|---|
| **Gender** |  |  |
| **F** | 5 | 3 |
| **M** | 3 | 4 |

In [30]:

```
import pandas as pd

df = pd.read_csv('Info.csv')
x = pd.crosstab(df['Gender'],df['Smoker'])
print(x)
```

```
Smoker   N  Y
Gender
F        5  3
M        3  4
```

In [47]:

```python
import pandas as pd
import numpy as np
df=pd.DataFrame([[0,1,2,np.nan,5],[2,0,1,5,np.nan],[5,0,1,np.nan,5],[2,0,1,np.nan,np.nan]])
print(df)
print("----------------------------------")
df=df.drop_duplicates(subset=[1,2])
print(df)
print("----------------------------------")
df=df.drop_duplicates(subset=[4])
print(df)
print("----------------------------------")
df.dropna(thresh=2,axis=1)
# print(df.shape) # (2, 5)

# inplace is not available so it's not store.
```

```
   0  1  2    3    4
0  0  1  2  NaN  5.0
1  2  0  1  5.0  NaN
2  5  0  1  NaN  5.0
3  2  0  1  NaN  NaN
----------------------------------
   0  1  2    3    4
0  0  1  2  NaN  5.0
1  2  0  1  5.0  NaN
----------------------------------
   0  1  2    3    4
0  0  1  2  NaN  5.0
1  2  0  1  5.0  NaN
----------------------------------
```

Out[47]:

| | 0 | 1 | 2 |
|---|---|---|---|
| **0** | 0 | 1 | 2 |
| **1** | 2 | 0 | 1 |

In [48]:

```python
import pandas as pd
import numpy as np
df=pd.DataFrame({"a":[1,2,np.nan,3,4],"b":[1,5,np.nan,2,1]})
df
# df=df.drop_duplicates(subset="b")
# df.dropna()
# df.fillna(20,inplace=True)
# print(df.shape[0])

```

Out[48]:

| | a | b |
|---|---|---|
| **0** | 1.0 | 1.0 |
| **1** | 2.0 | 5.0 |
| **2** | NaN | NaN |
| **3** | 3.0 | 2.0 |
| **4** | 4.0 | 1.0 |

In [52]:

```python
# que.23
import pandas as pd
import numpy as np
df=pd.DataFrame([[1,2,3,4,5],[2,1,3,4,5],[np.nan,np.nan,np.nan,np.nan,np.nan]])
print(df)
df.dropna(thresh=3,axis=1,inplace=True)
df
# print(df.shape[1])

```

```
     0    1    2    3    4
0  1.0  2.0  3.0  4.0  5.0
1  2.0  1.0  3.0  4.0  5.0
2  NaN  NaN  NaN  NaN  NaN
```

Out[52]:

| |
|---|
| **0** |
| **1** |
| **2** |

In [61]:

```
# que.29
data = {
 "A": ["TeamA", "TeamB", "TeamB", "TeamC", "TeamA"],
 "B": [50, 40, 40, 30, 50],
 "C": [True, False, False, False, True]
}
df = pd.DataFrame(data)
print(df)
df = df.drop_duplicates()
df = df.reset_index(drop=True)
df
```

```
       A   B      C
0  TeamA  50   True
1  TeamB  40  False
2  TeamB  40  False
3  TeamC  30  False
4  TeamA  50   True
```

Out[61]:

| | A | B | C |
|---|---|---|---|
| **0** | TeamA | 50 | True |
| **1** | TeamB | 40 | False |
| **2** | TeamC | 30 | False |

# P.b. 32

In [63]:

```
import pandas as pd

df = pd.read_csv('Datasets/ipl-matches.csv')
df
```

Out[63]:

| | ID | City | Date | Season | MatchNumber | Team1 | Team2 | Venue | TossWinner | TossDecision | SuperOver | WinningTea |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1312200 | Ahmedabad | 2022-05-29 | 2022 | Final | Rajasthan Royals | Gujarat Titans | Narendra Modi Stadium, Ahmedabad | Rajasthan Royals | bat | N | Gujarat Titai |
| **1** | 1312199 | Ahmedabad | 2022-05-27 | 2022 | Qualifier 2 | Royal Challengers Bangalore | Rajasthan Royals | Narendra Modi Stadium, Ahmedabad | Rajasthan Royals | field | N | Rajastha Roya |
| **2** | 1312198 | Kolkata | 2022-05-25 | 2022 | Eliminator | Royal Challengers Bangalore | Lucknow Super Giants | Eden Gardens, Kolkata | Lucknow Super Giants | field | N | Roy Challenge Bangalo |
| **3** | 1312197 | Kolkata | 2022-05-24 | 2022 | Qualifier 1 | Rajasthan Royals | Gujarat Titans | Eden Gardens, Kolkata | Gujarat Titans | field | N | Gujarat Titai |
| **4** | 1304116 | Mumbai | 2022-05-22 | 2022 | 70 | Sunrisers Hyderabad | Punjab Kings | Wankhede Stadium, Mumbai | Sunrisers Hyderabad | bat | N | Punjab Kin |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | |
| **945** | 335986 | Kolkata | 2008-04-20 | 2007/08 | 4 | Kolkata Knight Riders | Deccan Chargers | Eden Gardens | Deccan Chargers | bat | N | Kolkata Knig Ride |
| **946** | 335985 | Mumbai | 2008-04-20 | 2007/08 | 5 | Mumbai Indians | Royal Challengers Bangalore | Wankhede Stadium | Mumbai Indians | bat | N | Roy Challenge Bangalo |
| **947** | 335984 | Delhi | 2008-04-19 | 2007/08 | 3 | Delhi Daredevils | Rajasthan Royals | Feroz Shah Kotla | Rajasthan Royals | bat | N | De Daredev |
| **948** | 335983 | Chandigarh | 2008-04-19 | 2007/08 | 2 | Kings XI Punjab | Chennai Super Kings | Punjab Cricket Association Stadium, Mohali | Chennai Super Kings | bat | N | Chenr Super Kin |
| **949** | 335982 | Bangalore | 2008-04-18 | 2007/08 | 1 | Royal Challengers Bangalore | Kolkata Knight Riders | M Chinnaswamy Stadium | Royal Challengers Bangalore | field | N | Kolkata Knig Ride |

950 rows × 20 columns

In [67]:

```
# 1.
df = pd.read_csv('Datasets/ipl-matches.csv')
df.loc[df['SuperOver']=='Y']
```

Out[67]:

| | ID | City | Date | Season | MatchNumber | Team1 | Team2 | Venue | TossWinner | TossDecision | SuperOver | WinningTea |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **114** | 1254077 | Chennai | 2021-04-25 | 2021 | 20 | Delhi Capitals | Sunrisers Hyderabad | MA Chidambaram Stadium, Chepauk, Chennai | Delhi Capitals | bat | Y | Delhi Capita |
| **158** | 1216512 | Abu Dhabi | 2020-10-18 | 2020/21 | 35 | Kolkata Knight Riders | Sunrisers Hyderabad | Sheikh Zayed Stadium | Sunrisers Hyderabad | field | Y | Kolkata Knig Ride |
| **159** | 1216517 | NaN | 2020-10-18 | 2020/21 | 36 | Mumbai Indians | Kings XI Punjab | Dubai International Cricket Stadium | Mumbai Indians | bat | Y | Kings Punj |
| **184** | 1216547 | NaN | 2020-09-28 | 2020/21 | 10 | Royal Challengers Bangalore | Mumbai Indians | Dubai International Cricket Stadium | Mumbai Indians | field | Y | Roy Challenge Bangalo |
| **192** | 1216493 | NaN | 2020-09-20 | 2020/21 | 2 | Delhi Capitals | Kings XI Punjab | Dubai International Cricket Stadium | Kings XI Punjab | field | Y | Delhi Capita |
| **203** | 1178426 | Mumbai | 2019-05-02 | 2019 | 51 | Mumbai Indians | Sunrisers Hyderabad | Wankhede Stadium | Mumbai Indians | bat | Y | Mumb India |
| **244** | 1175365 | Delhi | 2019-03-30 | 2019 | 10 | Kolkata Knight Riders | Delhi Capitals | Arun Jaitley Stadium | Delhi Capitals | field | Y | Delhi Capita |
| **339** | 1082625 | Rajkot | 2017-04-29 | 2017 | 35 | Gujarat Lions | Mumbai Indians | Saurashtra Cricket Association Stadium | Gujarat Lions | bat | Y | Mumb India |
| **474** | 829741 | Ahmedabad | 2015-04-21 | 2015 | 18 | Rajasthan Royals | Kings XI Punjab | Sardar Patel Stadium, Motera | Kings XI Punjab | field | Y | Kings Punj |
| **533** | 729315 | Abu Dhabi | 2014-04-29 | 2014 | 19 | Kolkata Knight Riders | Rajasthan Royals | Sheikh Zayed Stadium | Rajasthan Royals | bat | Y | Rajasth Roya |
| **608** | 598017 | Bangalore | 2013-04-16 | 2013 | 21 | Royal Challengers Bangalore | Delhi Daredevils | M Chinnaswamy Stadium | Royal Challengers Bangalore | field | Y | Roy Challenge Bangalo |
| **621** | 598004 | Hyderabad | 2013-04-07 | 2013 | 7 | Sunrisers Hyderabad | Royal Challengers Bangalore | Rajiv Gandhi International Stadium, Uppal | Royal Challengers Bangalore | bat | Y | Sunrise Hyderab |
| **819** | 419121 | Chennai | 2010-03-21 | 2009/10 | 16 | Chennai Super Kings | Kings XI Punjab | MA Chidambaram Stadium, Chepauk | Chennai Super Kings | field | Y | Kings Punj |
| **883** | 392190 | Cape Town | 2009-04-23 | 2009 | 10 | Kolkata Knight Riders | Rajasthan Royals | Newlands | Kolkata Knight Riders | field | Y | Rajasth Roya |

In [68]:

```
# 2.
df.loc[(df['WinningTeam']=='Chennai Super Kings') & (df['City']=='Kolkata')]
```

Out[68]:

| | ID | City | Date | Season | MatchNumber | Team1 | Team2 | Venue | TossWinner | TossDecision | SuperOver | WinningTeam | WonBy | Mar |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **224** | 1178404 | Kolkata | 2019-04-14 | 2019 | 29 | Kolkata Knight Riders | Chennai Super Kings | Eden Gardens | Chennai Super Kings | field | N | Chennai Super Kings | Wickets | |
| **602** | 598022 | Kolkata | 2013-04-20 | 2013 | 26 | Kolkata Knight Riders | Chennai Super Kings | Eden Gardens | Kolkata Knight Riders | bat | N | Chennai Super Kings | Wickets | |
| **641** | 548368 | Kolkata | 2012-05-14 | 2012 | 63 | Kolkata Knight Riders | Chennai Super Kings | Eden Gardens | Chennai Super Kings | field | N | Chennai Super Kings | Wickets | |
| **827** | 419113 | Kolkata | 2010-03-16 | 2009/10 | 8 | Kolkata Knight Riders | Chennai Super Kings | Eden Gardens | Chennai Super Kings | bat | N | Chennai Super Kings | Runs | 5 |
| **908** | 336025 | Kolkata | 2008-05-18 | 2007/08 | 41 | Kolkata Knight Riders | Chennai Super Kings | Eden Gardens | Kolkata Knight Riders | bat | N | Chennai Super Kings | Runs | |

In [70]:

```
# 3.
df.loc[(df['Team1']=='Chennai Super Kings') & (df['Team2']=='Mumbai Indians') & (df['Player_of_Match']=='MS Dhoni'
```

Out[70]:

| | ID | City | Date | Season | MatchNumber | Team1 | Team2 | Venue | TossWinner | TossDecision | SuperOver | WinningTeam | WonB |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **630** | 548379 | Bangalore | 2012-05-23 | 2012 | Elimination Final | Chennai Super Kings | Mumbai Indians | M Chinnaswamy Stadium | Mumbai Indians | field | N | Chennai Super Kings | Run |

In [ ]:

```
# 4.

```

In [71]:

```
# 5.
df.loc[df['WinningTeam']=='Gujarat Titans']
```

Out[71]:

| | ID | City | Date | Season | MatchNumber | Team1 | Team2 | Venue | TossWinner | TossDecision | SuperOver | WinningTeam |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1312200 | Ahmedabad | 2022-05-29 | 2022 | Final | Rajasthan Royals | Gujarat Titans | Narendra Modi Stadium, Ahmedabad | Rajasthan Royals | bat | N | Gujarat Titans |
| **3** | 1312197 | Kolkata | 2022-05-24 | 2022 | Qualifier 1 | Rajasthan Royals | Gujarat Titans | Eden Gardens, Kolkata | Gujarat Titans | field | N | Gujarat Titans |
| **12** | 1304108 | Mumbai | 2022-05-15 | 2022 | 62 | Chennai Super Kings | Gujarat Titans | Wankhede Stadium, Mumbai | Chennai Super Kings | bat | N | Gujarat Titans |
| **17** | 1304103 | Pune | 2022-05-10 | 2022 | 57 | Gujarat Titans | Lucknow Super Giants | Maharashtra Cricket Association Stadium, Pune | Gujarat Titans | bat | N | Gujarat Titans |
| **31** | 1304089 | Mumbai | 2022-04-30 | 2022 | 43 | Royal Challengers Bangalore | Gujarat Titans | Brabourne Stadium, Mumbai | Royal Challengers Bangalore | bat | N | Gujarat Titans |
| **34** | 1304086 | Mumbai | 2022-04-27 | 2022 | 40 | Sunrisers Hyderabad | Gujarat Titans | Wankhede Stadium, Mumbai | Gujarat Titans | field | N | Gujarat Titans |
| **39** | 1304081 | Navi Mumbai | 2022-04-23 | 2022 | 35 | Gujarat Titans | Kolkata Knight Riders | Dr DY Patil Sports Academy, Mumbai | Gujarat Titans | bat | N | Gujarat Titans |
| **45** | 1304075 | Pune | 2022-04-17 | 2022 | 29 | Chennai Super Kings | Gujarat Titans | Maharashtra Cricket Association Stadium, Pune | Gujarat Titans | field | N | Gujarat Titans |
| **50** | 1304070 | Mumbai | 2022-04-14 | 2022 | 24 | Gujarat Titans | Rajasthan Royals | Dr DY Patil Sports Academy, Mumbai | Rajasthan Royals | field | N | Gujarat Titans |
| **58** | 1304062 | Mumbai | 2022-04-08 | 2022 | 16 | Punjab Kings | Gujarat Titans | Brabourne Stadium, Mumbai | Gujarat Titans | field | N | Gujarat Titans |
| **64** | 1304056 | Pune | 2022-04-02 | 2022 | 10 | Gujarat Titans | Delhi Capitals | Maharashtra Cricket Association Stadium, Pune | Delhi Capitals | field | N | Gujarat Titans |
| **70** | 1304050 | Mumbai | 2022-03-28 | 2022 | 4 | Lucknow Super Giants | Gujarat Titans | Wankhede Stadium, Mumbai | Gujarat Titans | field | N | Gujarat Titans |