Pandas Introduction

What is Pandas?

Pandas is a Python library used for working with data sets. It has functions for analyzing, cleaning, exploring, and manipulating data. The name "Pandas" has a reference to both "Panel Data", and "Python Data Analysis" and was created by Wes McKinney in 2008.

Why Use Pandas? Pandas allows us to analyze big data and make conclusions based on statistical theories. Pandas can clean messy data sets, and make them readable and relevant. Relevant data is very important in data science.

Data Science: is a branch of computer science where we study how to store, use and analyze data for deriving information from it.

What Can Pandas Do? Pandas gives you answers about the data. Like:

Is there a correlation between two or more columns? What is average value? Max value? Min value? Pandas are also able to delete rows that are not relevant, or contains wrong values, like empty or NULL values. This is called cleaning the data.

In [1]:

```
#importing pandas

import numpy as np
import pandas as pd
```

panda Series

A Pandas Series is like a column in a table. It is a 1-D array holding data of any type.

In [31]:

```
cities=["ahmedabad","surat","baroda","mumbai","delhi"]
p=pd.Series(cities)
print(p)
```

```
0    ahmedabad
1        surat
2       baroda
3       mumbai
4        delhi
dtype: object
```

In [32]:

```
marks=[50,60,70,80,90]
pd.Series(marks)
```

Out[32]:

```
0    50
1    60
2    70
3    80
4    90
dtype: int64
```

In [33]:

```
student_name=["ram","shyam","radha","geeta","seeta"]
marks=[50,60,70,80,90]
pd.Series(marks,index=student_name)
```

Out[33]:

```
ram       50
shyam     60
radha     70
geeta     80
seeta     90
dtype: int64
```

In [34]:

```
student_name=["ram","shyam","radha","geeta","seeta"]
marks=[50,60,70,80,90]
pd.Series(marks,index=student_name,name="student result")
```

Out[34]:

```
ram       50
shyam     60
radha     70
geeta     80
seeta     90
Name: student result, dtype: int64
```

In [3]:

```
marks={"ram":55,"shyam":60,"radha":80}
marks_series=pd.Series(marks,name="student result")
print(marks_series)
```

```
ram       55
shyam     60
radha     80
Name: student result, dtype: int64
```

series attributes

In [36]:

```
marks_series.size
```

Out[36]:

```
3
```

In [37]:

```
marks_series.dtype
```

Out[37]:

```
dtype('int64')
```

In [38]:

```
marks_series.name
```

Out[38]:

```
'student result'
```

In [39]:

```
marks_series.index
```

Out[39]:

```
Index(['ram', 'shyam', 'radha'], dtype='object')
```

In [40]:

```
print(marks_series.values)
print(type(marks_series.values))
```

```
[55 60 80]
<class 'numpy.ndarray'>
```

In [4]:
```
print(marks_series.is_unique)

print(pd.Series([55,55,22,33,44,55]).is_unique)
```

```
True
False
```

In [42]:
```
# with one col
subs = pd.read_csv('subs.csv')#,squeeze=True)
subs
```

Out[42]:

	Subscribers gained
0	48
1	57
2	40
3	43
4	44
...	...
360	231
361	226
362	155
363	144
364	172

365 rows × 1 columns

In [43]:
```
type(subs)
```

Out[43]:
```
pandas.core.frame.DataFrame
```

In [3]:
```
subs = pd.read_csv(('subs.csv'),squeeze=True) # squeeze is used to converdt
subs
```

Out[3]:
```
0       48
1       57
2       40
3       43
4       44
      ... 
360    231
361    226
362    155
363    144
364    172
Name: Subscribers gained, Length: 365, dtype: int64
```

In [45]:
```
type(subs)
```

Out[45]:
```
pandas.core.series.Series
```

In [6]:
```
movies = pd.read_csv('bollywood.csv',index_col='movie',squeeze=True)
movies
```

Out[6]:
```
movie
Uri: The Surgical Strike                                   Vicky Kaushal
Battalion 609                                                Vicky Ahuja
The Accidental Prime Minister (film)                         Anupam Kher
Why Cheat India                                            Emraan Hashmi
Evening Shadows                                         Mona Ambegaonkar
                                                              ...
Hum Tumhare Hain Sanam                                    Shah Rukh Khan
Aankhen (2002 film)                                     Amitabh Bachchan
Saathiya (film)                                             Vivek Oberoi
Company (film)                                                Ajay Devgn
Awara Paagal Deewana                                        Akshay Kumar
Name: lead, Length: 1500, dtype: object
```

series methods

In [47]:
```
movies.head()
```

Out[47]:
```
movie
Uri: The Surgical Strike                                   Vicky Kaushal
Battalion 609                                                Vicky Ahuja
The Accidental Prime Minister (film)                         Anupam Kher
Why Cheat India                                            Emraan Hashmi
Evening Shadows                                         Mona Ambegaonkar
Name: lead, dtype: object
```

In [48]:
```
movies.tail()
```

Out[48]:
```
movie
Hum Tumhare Hain Sanam      Shah Rukh Khan
Aankhen (2002 film)       Amitabh Bachchan
Saathiya (film)               Vivek Oberoi
Company (film)                  Ajay Devgn
Awara Paagal Deewana          Akshay Kumar
Name: lead, dtype: object
```

In [49]:
```
movies.head(3)
```

Out[49]:
```
movie
Uri: The Surgical Strike                Vicky Kaushal
Battalion 609                             Vicky Ahuja
The Accidental Prime Minister (film)      Anupam Kher
Name: lead, dtype: object
```

In [50]:
```
movies.tail(3)
```

Out[50]:
```
movie
Saathiya (film)         Vivek Oberoi
Company (film)            Ajay Devgn
Awara Paagal Deewana    Akshay Kumar
Name: lead, dtype: object
```

```
In [8]:  subs = pd.read_csv(('subs.csv'),squeeze=True) # squeeze is used to converdt
         subs.describe()

Out[8]:  count    365.000000
         mean     135.643836
         std       62.675023
         min       33.000000
         25%       88.000000
         50%      123.000000
         75%      177.000000
         max      396.000000
         Name: Subscribers gained, dtype: float64
```

```
In [52]: subs.min()

Out[52]: 33
```

```
In [53]: subs.max()

Out[53]: 396
```

```
In [54]: subs.median()

Out[54]: 123.0
```

```
In [55]: subs.sum()

Out[55]: 49510
```

```
In [9]:  subs.mean()

Out[9]:  135.64383561643837
```

series indexing

```
In [56]: x = pd.Series([12,13,14,35,46,57,58,79,9])
         x

Out[56]: 0    12
         1    13
         2    14
         3    35
         4    46
         5    57
         6    58
         7    79
         8     9
         dtype: int64
```

```
In [57]: x[2]

Out[57]: 14
```

In [58]:
```
x[0:5]
```

Out[58]:
```
0    12
1    13
2    14
3    35
4    46
dtype: int64
```

In [59]:
```
x[::-1]
```

Out[59]:
```
8     9
7    79
6    58
5    57
4    46
3    35
2    14
1    13
0    12
dtype: int64
```

```
In [60]: x[-1] # if indexing is integer or number than it will give error and if it i
```

```
---------------------------------------------------------------------------
-
ValueError                                Traceback (most recent call las
t)
C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\indexes\range.py in
get_loc(self, key, method, tolerance)
    354                 try:
--> 355                     return self._range.index(new_key)
    356                 except ValueError as err:

ValueError: -1 is not in range

The above exception was the direct cause of the following exception:

KeyError                                  Traceback (most recent call las
t)
<ipython-input-60-1987a1c571db> in <module>
----> 1 x[-1] # if indexing is integer or number than it will give error a
nd if it is string than it will give values

C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\series.py in __geti
tem__(self, key)
    880
    881         elif key_is_scalar:
--> 882             return self._get_value(key)
    883
    884         if is_hashable(key):

C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\series.py in _get_v
alue(self, label, takeable)
    987
    988         # Similar to Index.get_value, but we do not fall back to p
ositional
--> 989         loc = self.index.get_loc(label)
    990         return self.index._get_values_for_loc(self, loc, label)
    991

C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\indexes\range.py in
get_loc(self, key, method, tolerance)
    355                     return self._range.index(new_key)
    356                 except ValueError as err:
--> 357                     raise KeyError(key) from err
    358             raise KeyError(key)
    359         return super().get_loc(key, method=method, tolerance=toler
ance)

KeyError: -1
```

```
In [61]: marks_series[-1]

Out[61]: 80
```

In [62]:
```
movies
```

Out[62]:
```
movie
Uri: The Surgical Strike                   Vicky Kaushal
Battalion 609                                Vicky Ahuja
The Accidental Prime Minister (film)         Anupam Kher
Why Cheat India                            Emraan Hashmi
Evening Shadows                         Mona Ambegaonkar
                                                     ...
Hum Tumhare Hain Sanam                    Shah Rukh Khan
Aankhen (2002 film)                     Amitabh Bachchan
Saathiya (film)                             Vivek Oberoi
Company (film)                                Ajay Devgn
Awara Paagal Deewana                        Akshay Kumar
Name: lead, Length: 1500, dtype: object
```

In [63]:
```
movies[0]
```

Out[63]:
```
'Vicky Kaushal'
```

In [64]:
```
movies['Uri: The Surgical Strike']
```

Out[64]:
```
'Vicky Kaushal'
```

In [65]:
```
movies[-5:]
```

Out[65]:
```
movie
Hum Tumhare Hain Sanam      Shah Rukh Khan
Aankhen (2002 film)       Amitabh Bachchan
Saathiya (film)               Vivek Oberoi
Company (film)                  Ajay Devgn
Awara Paagal Deewana          Akshay Kumar
Name: lead, dtype: object
```

In [66]:
```
movies[::2]
```

Out[66]:
```
movie
Uri: The Surgical Strike                   Vicky Kaushal
The Accidental Prime Minister (film)         Anupam Kher
Evening Shadows                         Mona Ambegaonkar
Fraud Saiyaan                               Arshad Warsi
Manikarnika: The Queen of Jhansi          Kangana Ranaut
                                                     ...
Raaz (2002 film)                              Dino Morea
Waisa Bhi Hota Hai Part II                  Arshad Warsi
Kaante                                  Amitabh Bachchan
Aankhen (2002 film)                     Amitabh Bachchan
Company (film)                                Ajay Devgn
Name: lead, Length: 750, dtype: object
```

In [67]:
```
movies['2 States (2014 film)'] #fancy indexing
```

Out[67]:
```
'Arjun Kapoor'
```

```
In [68]: marks_series[1] = 100
         marks_series
```

```
Out[68]: ram       55
         shyam    100
         radha     80
         Name: student result, dtype: int64
```

```
In [69]: movies[[0,1,3,4,5]]
```

```
Out[69]: movie
         Uri: The Surgical Strike           Vicky Kaushal
         Battalion 609                        Vicky Ahuja
         Why Cheat India                    Emraan Hashmi
         Evening Shadows                 Mona Ambegaonkar
         Soni (film)                  Geetika Vidya Ohlyan
         Name: lead, dtype: object
```

```
In [14]: print(movies.iloc[[1,6]])
```

```
         movie
         Battalion 609     Vicky Ahuja
         Fraud Saiyaan    Arshad Warsi
         Name: lead, dtype: object
```

Series with Python Functionalities

```
In [71]: list(marks_series)
```

```
Out[71]: [55, 100, 80]
```

```
In [72]: dict(marks_series)
```

```
Out[72]: {'ram': 55, 'shyam': 100, 'radha': 80}
```

```
In [73]: '2 States (2014 film)' in movies
```

```
Out[73]: True
```

```
In [74]: 'Alia Bhatt' in movies.values
```

```
Out[74]: True
```

In [75]:
```
for i in movies.index:
  print(i)
```

```
Uri: The Surgical Strike
Battalion 609
The Accidental Prime Minister (film)
Why Cheat India
Evening Shadows
Soni (film)
Fraud Saiyaan
Bombairiya
Manikarnika: The Queen of Jhansi
Thackeray (film)
Amavas
Gully Boy
Hum Chaar
Total Dhamaal
Sonchiriya
Badla (2019 film)
Mard Ko Dard Nahi Hota
Hamid (film)
Photograph (film)
```

In [76]:
```
movies.index
```

Out[76]:
```
Index(['Uri: The Surgical Strike', 'Battalion 609',
       'The Accidental Prime Minister (film)', 'Why Cheat India',
       'Evening Shadows', 'Soni (film)', 'Fraud Saiyaan', 'Bombairiya',
       'Manikarnika: The Queen of Jhansi', 'Thackeray (film)',
       ...
       'Raaz (2002 film)', 'Zameen (2003 film)', 'Waisa Bhi Hota Hai Part
II',
       'Devdas (2002 Hindi film)', 'Kaante', 'Hum Tumhare Hain Sanam',
       'Aankhen (2002 film)', 'Saathiya (film)', 'Company (film)',
       'Awara Paagal Deewana'],
      dtype='object', name='movie', length=1500)
```

In [77]:
```
100 + marks_series
```

Out[77]:
```
ram      155
shyam    200
radha    180
Name: student result, dtype: int64
```

In [78]:
```
marks_series >= 100
```

Out[78]:
```
ram      False
shyam     True
radha    False
Name: student result, dtype: bool
```

In [79]:

```
# find actors who have done more than 20 movies
num_movies = movies.value_counts()
num_movies[num_movies > 20]
```

Out[79]:

```
Akshay Kumar        48
Amitabh Bachchan    45
Ajay Devgn          38
Salman Khan         31
Sanjay Dutt         26
Shah Rukh Khan      22
Emraan Hashmi       21
Name: lead, dtype: int64
```

In [80]:

```
movies.value_counts()
```

Out[80]:

```
Akshay Kumar        48
Amitabh Bachchan    45
Ajay Devgn          38
Salman Khan         31
Sanjay Dutt         26
                    ..
Gulshan Grover       1
Juhi Babbar          1
Satish Kaushik       1
Naman Jain           1
Parzaan Dastur       1
Name: lead, Length: 566, dtype: int64
```

In [81]:

```
# Count number of day when I had more than 200 subs a day
subs[subs > 200].size
```

Out[81]:

```
59
```

```
In [82]: #Write a Pandas program to add, subtract, multiple and divide two Pandas Ser

         a = pd.Series([2, 4, 6, 8, 10])
         b = pd.Series([1, 3, 5, 7, 10])

         print(a+b)
         print(a-b)
         print(a*b)
         print(a/b)
```

```
0     3
1     7
2    11
3    15
4    20
dtype: int64
0    1
1    1
2    1
3    1
4    0
dtype: int64
0      2
1     12
2     30
3     56
4    100
dtype: int64
0    2.000000
1    1.333333
2    1.200000
3    1.142857
4    1.000000
dtype: float64
```

In [4]:

```
#Write a Pandas program to compare the elements of the two Pandas Series.
#Sample Series: [2, 4, 6, 8, 10], [1, 3, 5, 7, 10]
# code here
a = pd.Series([2, 4, 6, 8, 10])
b = pd.Series([1, 3, 5, 7, 10])

print(a==b)
print(a<b)
print(a>b)
print(a&b)
print(a|b)
```

```
0    False
1    False
2    False
3    False
4     True
dtype: bool
0    False
1    False
2    False
3    False
4    False
dtype: bool
0     True
1     True
2     True
3     True
4    False
dtype: bool
0     0
1     0
2     4
3     0
4    10
dtype: int64
0     3
1     7
2     7
3    15
4    10
dtype: int64
```

In [6]:

```
#To select only some of the items in the dictionary, use the index argument
#only the items you want to include in the Series.
import pandas as pd

calories = {"day1": 420, "day2": 380, "day3": 390}

myvar = pd.Series(calories, index = ["day1", "day2"])

print(myvar)
```

```
day1    420
day2    380
dtype: int64
```

What is a DataFrame?

A Pandas DataFrame is a 2 dimensional data structure, like a 2 dimensional array, or a table with rows and columns.

In [2]:

```
import pandas as pd

data = {
  "calories": [420, 380, 390],
  "duration": [50, 40, 45]
}

df = pd.DataFrame(data)

print(df)
```

```
   calories  duration
0       420        50
1       380        40
2       390        45
```

In [3]:

```
print(df.iloc[1]) #integer location in dataframe
```

```
calories    380
duration     40
Name: 1, dtype: int64
```

Locate Row

As you can see from the result above, the DataFrame is like a table with rows and columns.

Pandas use the loc attribute to return one or more specified row(s)

In [5]:

```
print(df.loc[0])
```

```
calories    420
duration     50
Name: 0, dtype: int64
```

In [21]:

```
#use a list of indexes:
print(df.iloc[[0, 1]])
```

```
      calories  duration
day1       420        50
day2       380        40
```

In [16]:

```
#With the index argument, you can name your own indexes.
import pandas as pd

data = {
  "calories": [420, 380, 390],
  "duration": [50, 40, 45]
}

df = pd.DataFrame(data, index = ["day1", "day2", "day3"])

print(df)
```

```
      calories  duration
day1       420        50
day2       380        40
day3       390        45
```

In [89]:

```
print(df.loc["day2"])
```

```
calories    380
duration     40
Name: day2, dtype: int64
```

In [6]:

```
print(df.iloc["day2"])
```

```
---------------------------------------------------------------------------
-
TypeError                                 Traceback (most recent call las
t)
<ipython-input-6-cb17086948c3> in <module>
----> 1 print(df.iloc["day2"])

C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\indexing.py in __ge
titem__(self, key)
    877 
    878             maybe_callable = com.apply_if_callable(key, self.obj)
--> 879             return self._getitem_axis(maybe_callable, axis=axis)
    880 
    881     def _is_scalar_access(self, key: Tuple):

C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\indexing.py in _get
item_axis(self, key, axis)
   1491             key = item_from_zerodim(key)
   1492             if not is_integer(key):
-> 1493                 raise TypeError("Cannot index by location index wi
th a non-integer key")
   1494 
   1495             # validate the location

TypeError: Cannot index by location index with a non-integer key
```

In [11]:
```
df[::2] # all type of indexes we can apply in dataframe same as series
```

Out[11]:

	calories	duration
0	420	50
2	390	45

If your data sets are stored in a file, Pandas can load them into a DataFrame.

In [2]:
```
import pandas as pd
dataset = pd.read_csv('diabetes.csv')

print(dataset)
```

```
     Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0              6      148             72             35        0  33.6   
1              1       85             66             29        0  26.6   
2              8      183             64              0        0  23.3   
3              1       89             66             23       94  28.1   
4              0      137             40             35      168  43.1   
..           ...      ...            ...            ...      ...   ...   
763           10      101             76             48      180  32.9   
764            2      122             70             27        0  36.8   
765            5      121             72             23      112  26.2   
766            1      126             60              0        0  30.1   
767            1       93             70             31        0  30.4   

     DiabetesPedigreeFunction  Age  Outcome  
0                       0.627   50        1  
1                       0.351   31        0  
2                       0.672   32        1  
3                       0.167   21        0  
4                       2.288   33        1  
..                        ...  ...      ...  
763                     0.171   63        0  
764                     0.340   27        0  
765                     0.245   30        0  
766                     0.349   47        1  
767                     0.315   23        0  

[768 rows x 9 columns]
```

In [4]:
```
print(dataset.to_string()) #  to see complete dataframe
```

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

In [91]:
```
import pandas as pd
dataset = pd.read_csv('movies.csv')
print(dataset)
```

```
                                   title_x     imdb_id  \
0                 Uri: The Surgical Strike   tt8291224
1                            Battalion 609   tt9472208
2      The Accidental Prime Minister (film)   tt6986710
3                          Why Cheat India   tt8108208
4                          Evening Shadows   tt6028796
...                                    ...         ...
1624                 Tera Mera Saath Rahen   tt0301250
1625                  Yeh Zindagi Ka Safar   tt0298607
1626                       Sabse Bada Sukh   tt0069204
1627                                 Daaka  tt10833860
1628                              Humsafar   tt2403201

                                            poster_path  \
0     https://upload.wikimedia.org/wikipedia/en/thum... (https://upload.
wikimedia.org/wikipedia/en/thum...)
1                                                   NaN
2     https://upload.wikimedia.org/wikipedia/en/thum... (https://upload.
wikimedia.org/wikipedia/en/thum...)
```

In [92]:

```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1629 entries, 0 to 1628
Data columns (total 18 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   title_x           1629 non-null   object 
 1   imdb_id           1629 non-null   object 
 2   poster_path       1526 non-null   object 
 3   wiki_link         1629 non-null   object 
 4   title_y           1629 non-null   object 
 5   original_title    1629 non-null   object 
 6   is_adult          1629 non-null   int64  
 7   year_of_release   1629 non-null   int64  
 8   runtime           1629 non-null   object 
 9   genres            1629 non-null   object 
 10  imdb_rating       1629 non-null   float64
 11  imdb_votes        1629 non-null   int64  
 12  story             1609 non-null   object 
 13  summary           1629 non-null   object 
 14  tagline           557 non-null    object 
 15  actors            1624 non-null   object 
 16  wins_nominations  707 non-null    object 
 17  release_date      1522 non-null   object 
dtypes: float64(1), int64(3), object(14)
memory usage: 229.2+ KB
```

In [93]:

```
dataset.tail()
```

Out[93]:

	title_x	imdb_id	poster_path	
1624	Tera Mera Saath Rahen	tt0301250	https://upload.wikimedia.org/wikipedia/en/2/2b...	https://en.wikipedia.org/v
1625	Yeh Zindagi Ka Safar	tt0298607	https://upload.wikimedia.org/wikipedia/en/thum...	https://en.wikipedia.org/v
1626	Sabse Bada Sukh	tt0069204	NaN	https://en.wikipedia.org
1627	Daaka	tt10833860	https://upload.wikimedia.org/wikipedia/en/thum...	https://en.
1628	Humsafar	tt2403201	https://upload.wikimedia.org/wikipedia/en/thum...	https://en.wik

In [94]:
```
dataset.tail(7)
```

Out[94]:

	title_x	imdb_id	poster_path	
1622	Yeh Teraa Ghar Yeh Meraa Ghar	tt0298606	https://upload.wikimedia.org/wikipedia/en/thum...	https://en.wikipedia.org/v
1623	Zubeidaa	tt0255713	https://upload.wikimedia.org/wikipedia/en/thum...	https://en.wik
1624	Tera Mera Saath Rahen	tt0301250	https://upload.wikimedia.org/wikipedia/en/2/2b...	https://en.wikipedia.org/v
1625	Yeh Zindagi Ka Safar	tt0298607	https://upload.wikimedia.org/wikipedia/en/thum...	https://en.wikipedia.org/v
1626	Sabse Bada Sukh	tt0069204	NaN	https://en.wikipedia.org
1627	Daaka	tt10833860	https://upload.wikimedia.org/wikipedia/en/thum...	https://en.
1628	Humsafar	tt2403201	https://upload.wikimedia.org/wikipedia/en/thum...	https://en.wik

In [95]:

```
dataset.head()
```

Out[95]:

	title_x	imdb_id	poster_path	
0	Uri: The Surgical Strike	tt8291224	https://upload.wikimedia.org/wikipedia/en/thum...	https://en.wikipedia.org/wiki/U
1	Battalion 609	tt9472208	NaN	https://en.wikipedia.org
2	The Accidental Prime Minister (film)	tt6986710	https://upload.wikimedia.org/wikipedia/en/thum...	https://en.wikipedia.org/wiki/T
3	Why Cheat India	tt8108208	https://upload.wikimedia.org/wikipedia/en/thum...	https://en.wikipedia.org/wik
4	Evening Shadows	tt6028796	NaN	https://en.wikipedia.org/wiki/

In [96]:
```
dataset.head(7)
```

Out[96]:

	title_x	imdb_id	poster_path	
0	Uri: The Surgical Strike	tt8291224	https://upload.wikimedia.org/wikipedia/en/thum...	https://en.wikipedia.org/wiki/U
1	Battalion 609	tt9472208	NaN	https://en.wikipedia.org
2	The Accidental Prime Minister (film)	tt6986710	https://upload.wikimedia.org/wikipedia/en/thum...	https://en.wikipedia.org/wiki/T
3	Why Cheat India	tt8108208	https://upload.wikimedia.org/wikipedia/en/thum...	https://en.wikipedia.org/wik
4	Evening Shadows	tt6028796	NaN	https://en.wikipedia.org/wiki/
5	Soni (film)	tt6078866	https://upload.wikimedia.org/wikipedia/en/thum...	https://en.wikipedia.
6	Fraud Saiyaan	tt5013008	https://upload.wikimedia.org/wikipedia/en/thum...	https://en.wikipedia.org/v

In [97]:
```
dataset.shape
```

Out[97]:
```
(1629, 18)
```

In [98]:
```
dataset.describe()
```

Out[98]:

	is_adult	year_of_release	imdb_rating	imdb_votes
count	1629.0	1629.000000	1629.000000	1629.000000
mean	0.0	2010.263966	5.557459	5384.263352
std	0.0	5.381542	1.567609	14552.103231
min	0.0	2001.000000	0.000000	0.000000
25%	0.0	2005.000000	4.400000	233.000000
50%	0.0	2011.000000	5.600000	1000.000000
75%	0.0	2015.000000	6.800000	4287.000000
max	0.0	2019.000000	9.400000	310481.000000

In [99]:
```
dataset.describe(include='all')
```

Out[99]:

	title_x	imdb_id	poster_path	
count	1629	1629	1526	
unique	1625	1623	1517	
top	Lagaan	tt0346507	https://upload.wikimedia.org/wikipedia/en/thum...	https://en.wikipedia.org/wi
freq	2	2	4	
mean	NaN	NaN	NaN	
std	NaN	NaN	NaN	
min	NaN	NaN	NaN	
25%	NaN	NaN	NaN	
50%	NaN	NaN	NaN	
75%	NaN	NaN	NaN	
max	NaN	NaN	NaN	

A great aspect of the Pandas module is the corr() method.

The corr() method calculates the relationship between each column in your data set.

In [100]:
```
dataset.corr()
```

Out[100]:

	is_adult	year_of_release	imdb_rating	imdb_votes
is_adult	NaN	NaN	NaN	NaN
year_of_release	NaN	1.000000	0.105161	0.057019
imdb_rating	NaN	0.105161	1.000000	0.338362
imdb_votes	NaN	0.057019	0.338362	1.000000

Perfect Correlation: We can see that "mpg" and "mpg" got the number 1.000000, which makes sense, each column always has a perfect relationship with itself.

Good Correlation: "cylinders" and "displacement" got a 0.950721 correlation, which is a very good correlation, and we can predict that more cylinders means more displacement.

Bad Correlation: "model year" and "acceleration" got a 0.288137 correlation, which is a very bad correlation, meaning that we can not predict the max pulse by just looking at the duration of the work out, and vice versa.

Scatter Matrix/Pair Plots

Returns a numpy.ndarray

By default, alpha=1. If you would like to form the graph plot more transparent, then you'll make alpha but 1, such as 0.5 or 0.25.

If you would like to form the graph plot less transparent, then you'll make alpha greater than 1. This solidifies the graph plot, making it less transparent and more thick and dense, so to talk .

In [2]:

```
import pandas as pd
dataset = pd.read_csv('https://raw.githubusercontent.com/Jovita7/Data-Analys
print(dataset)
print(dataset.to_string())
print(dataset.info())
```

```
      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  
```

In [21]:
```
dataset.describe(include="all")
```

Out[21]:

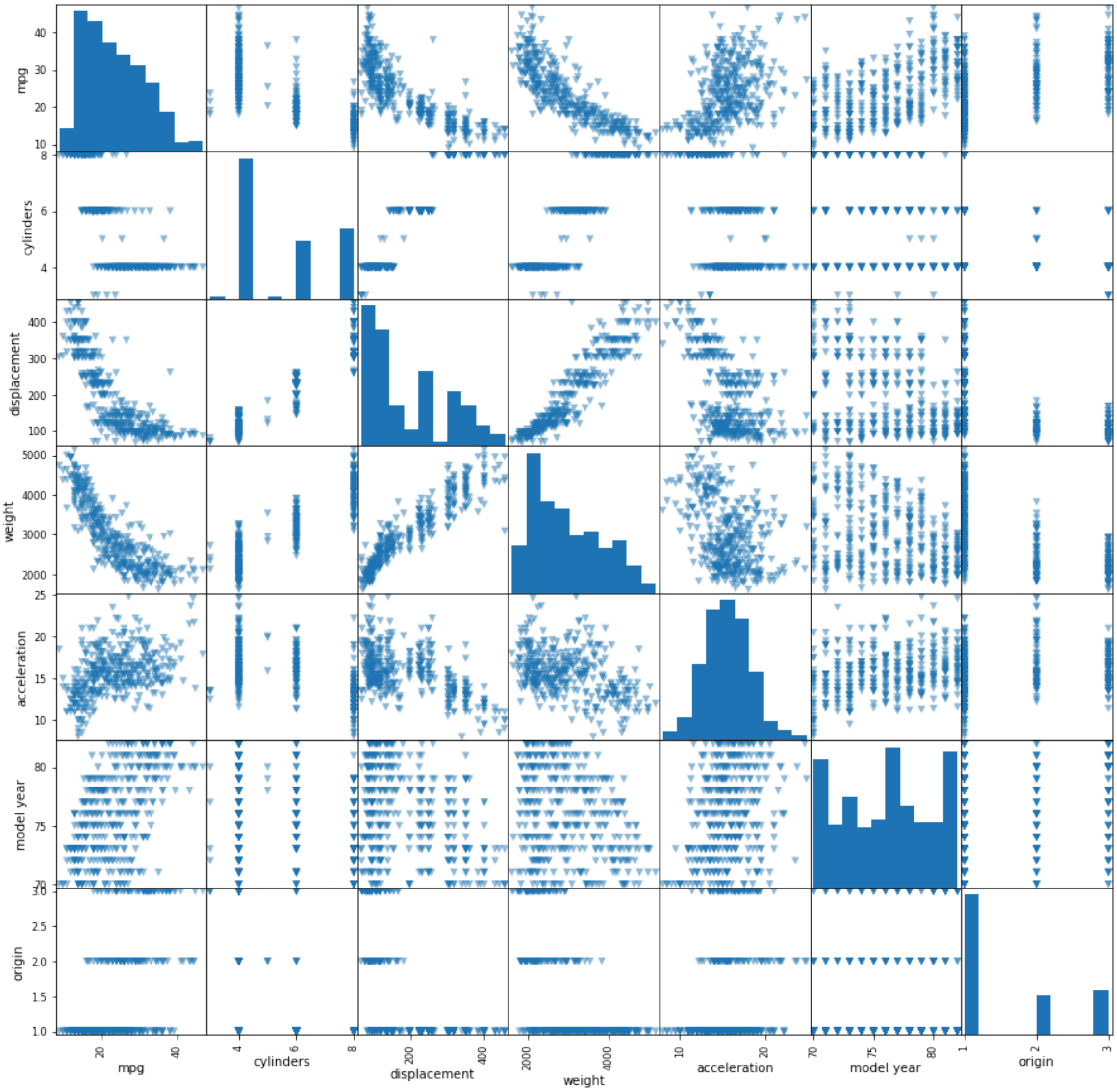
	mpg	cylinders	displacement	horsepower	weight	acceleration	model y
count	398.000000	398.000000	398.000000	398	398.000000	398.000000	398.000
unique	NaN	NaN	NaN	94	NaN	NaN	
top	NaN	NaN	NaN	150	NaN	NaN	
freq	NaN	NaN	NaN	22	NaN	NaN	
mean	23.514573	5.454774	193.425879	NaN	2970.424623	15.568090	76.010
std	7.815984	1.701004	104.269838	NaN	846.841774	2.757689	3.697
min	9.000000	3.000000	68.000000	NaN	1613.000000	8.000000	70.000
25%	17.500000	4.000000	104.250000	NaN	2223.750000	13.825000	73.000
50%	23.000000	4.000000	148.500000	NaN	2803.500000	15.500000	76.000
75%	29.000000	8.000000	262.000000	NaN	3608.000000	17.175000	79.000
max	46.600000	8.000000	455.000000	NaN	5140.000000	24.800000	82.000

In [105]:
```
dataset.corr()
```

Out[105]:

	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 [106]:
```
import matplotlib.pyplot as plt
pd.plotting.scatter_matrix(dataset, figsize  = [15, 15], marker = 'v', alpha
plt.show()
```



Qualitative Data vs Quantitative Data

Quantitative data relates to information about the quantity of an object – hence it can be measured. For example, if we consider the attribute ‘marks’, it can be measured using a scale of measurement. Quantitative data is also termed as numeric data.

Qualitative data provides information about the quality of an object or information which cannot be measured. For example, if we consider the quality of performance of students in terms of ‘Good’, ‘Average’, and ‘Poor’, it falls under the category of qualitative data. Also, name or roll number of students are information that cannot be measured using some scale of measurement. So they would fall under qualitative data. Qualitative data is also called categorical data.

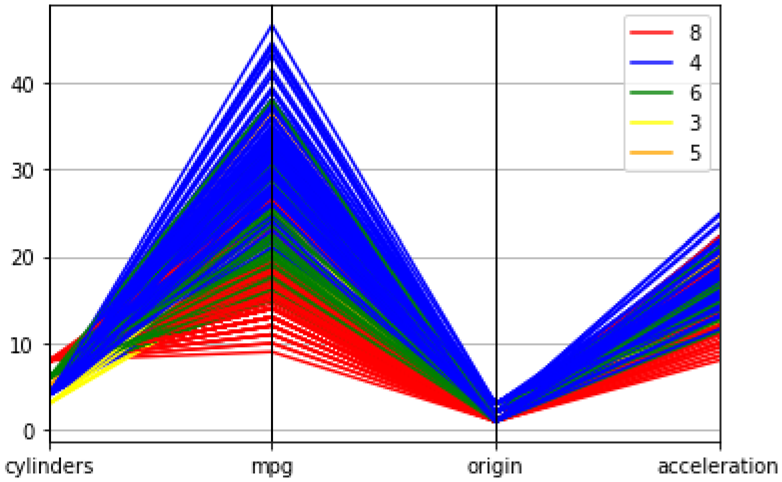
Quantitative Data can be analyzed by measures like mean, median, mode.

For qualitative data, we can use parallel coordinates and cross tabulation.

Parallel coordinates

Parallel coordinates charts are commonly used to visualize and analyze high dimensional multivariate data. It represents each data sample as polyline connecting parallel lines where each parallel line represents an attribute of that data sample.

In [3]:
```
import matplotlib.pyplot as plt
from pandas.plotting import parallel_coordinates
pll = parallel_coordinates(dataset, 'cylinders', cols=['cylinders','mpg', 'c
                           color=('red', 'blue', 'green', 'yellow', 'orange'
plt.show()
```



##Cross Tabulation

In [108]:
```
pd.crosstab(dataset['cylinders'], dataset['model year'], rownames=['cylinder
```

Out[108]:

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

Data Cleaning

Missing Values

In [4]:

```
import numpy as np
import pandas as pd
sales_data = pd.DataFrame({"name":["William","Emma","Sofia","Markus","Edwarc
,"region":[np.nan,"North","East",np.nan,"West","West","South",np.nan,"West",
,"sales":[50000,52000,np.nan,np.nan,42000,72000,49000,np.nan,67000,65000,67(
,"expenses":[42000,43000,np.nan,np.nan,38000,39000,42000,np.nan,39000,50000,
print(sales_data)
```

```
       name region    sales  expenses
0   William    NaN  50000.0   42000.0
1      Emma  North  52000.0   43000.0
2     Sofia   East      NaN       NaN
3    Markus    NaN      NaN       NaN
4    Edward   West  42000.0   38000.0
5    Thomas   West  72000.0   39000.0
6     Ethan  South  49000.0   42000.0
7       NaN    NaN      NaN       NaN
8      Arun   West  67000.0   39000.0
9    Anika   East  65000.0   50000.0
10    Paulo  South  67000.0   45000.0
```

In [19]:

```
sales_data.isna().sum()
```

Out[19]:

```
name        1
region      3
sales       3
expenses    3
dtype: int64
```

In [8]:

```
sales_data.isna()
```

Out[8]:

	name	region	sales	expenses
1	False	False	False	False
4	False	False	False	False
5	False	False	False	False
6	False	False	False	False
8	False	False	False	False
9	False	False	False	False
10	False	False	False	False

In [9]:

```
sales_data.dropna()
```

Out[9]:

	name	region	sales	expenses
1	Emma	North	52000.0	43000.0
4	Edward	West	42000.0	38000.0
5	Thomas	West	72000.0	39000.0
6	Ethan	South	49000.0	42000.0
8	Arun	West	67000.0	39000.0
9	Anika	East	65000.0	50000.0
10	Paulo	South	67000.0	45000.0

##thresh specifies quantity of valid data so thresh = 2 means remove if there aren't atleast two cells with valid data (not null data)

In [22]:

```
sales_data.dropna(thresh=2)
```

Out[22]:

	name	region	sales	expenses
0	William	NaN	50000.0	42000.0
1	Emma	North	52000.0	43000.0
2	Sofia	East	NaN	NaN
4	Edward	West	42000.0	38000.0
5	Thomas	West	72000.0	39000.0
6	Ethan	South	49000.0	42000.0
8	Arun	West	67000.0	39000.0
9	Anika	East	65000.0	50000.0
10	Paulo	South	67000.0	45000.0

In [23]:

```
sales_data.dropna(how='all')##remove only if all values are null
```

Out[23]:

	name	region	sales	expenses
0	William	NaN	50000.0	42000.0
1	Emma	North	52000.0	43000.0
2	Sofia	East	NaN	NaN
3	Markus	NaN	NaN	NaN
4	Edward	West	42000.0	38000.0
5	Thomas	West	72000.0	39000.0
6	Ethan	South	49000.0	42000.0
8	Arun	West	67000.0	39000.0
9	Anika	East	65000.0	50000.0
10	Paulo	South	67000.0	45000.0

In [24]:
```
#drop only if sales or expenses are null
sales_data.dropna(subset = ['sales', 'expenses'])
```

Out[24]:
```
       name  region    sales  expenses
 0  William     NaN  50000.0   42000.0
 1    Emma    North  52000.0   43000.0
 4   Edward    West  42000.0   38000.0
 5   Thomas    West  72000.0   39000.0
 6    Ethan   South  49000.0   42000.0
 8     Arun    West  67000.0   39000.0
 9    Anika    East  65000.0   50000.0
10    Paulo   South  67000.0   45000.0
```

In [25]:
```
sales_data.dropna(axis=0)
```

Out[25]:
```
       name  region    sales  expenses
 1     Emma   North  52000.0   43000.0
 4   Edward    West  42000.0   38000.0
 5   Thomas    West  72000.0   39000.0
 6    Ethan   South  49000.0   42000.0
 8     Arun    West  67000.0   39000.0
 9    Anika    East  65000.0   50000.0
10    Paulo   South  67000.0   45000.0
```

In [26]:
```
sales_data.dropna(axis=1)
```

Out[26]:
```
 0
 1
 2
 3
 4
 5
 6
 7
 8
 9
10
```

In [10]:
```
sales_data.dropna(inplace=True)
```

In [15]:
```
import numpy as np
import pandas as pd
sales_data = pd.DataFrame({"name":["William","Emma","Sofia","Markus","Edwarc
,"region":[np.nan,"North","East",np.nan,"West","West","South",np.nan,"West",
,"sales":[50000,52000,np.nan,np.nan,42000,72000,49000,np.nan,67000,65000,670
,"expenses":[42000,43000,np.nan,np.nan,38000,39000,42000,np.nan,39000,50000,
print(sales_data)
sales_data['sales'].fillna(0)
```

```
       name region    sales  expenses
0   William    NaN  50000.0   42000.0
1      Emma  North  52000.0   43000.0
2     Sofia   East      NaN       NaN
3    Markus    NaN      NaN       NaN
4    Edward   West  42000.0   38000.0
5    Thomas   West  72000.0   39000.0
6     Ethan  South  49000.0   42000.0
7       NaN    NaN      NaN       NaN
8      Arun   West  67000.0   39000.0
9     Anika   East  65000.0   50000.0
10    Paulo  South  67000.0   45000.0
```

Out[15]:
```
0     50000.0
1     52000.0
2         0.0
3         0.0
4     42000.0
5     72000.0
6     49000.0
7         0.0
8     67000.0
9     65000.0
10    67000.0
Name: sales, dtype: float64
```

In [118]:
```
sales_data['sales'].fillna(sales_data['sales'].mean())
```

Out[118]:
```
0     50000.0
1     52000.0
2     58000.0
3     58000.0
4     42000.0
5     72000.0
6     49000.0
7     58000.0
8     67000.0
9     65000.0
10    67000.0
Name: sales, dtype: float64
```

In [17]:

```
sales_data.fillna(0)
```

Out[17]:

	name	region	sales	expenses
0	William	0	50000.0	42000.0
1	Emma	North	52000.0	43000.0
2	Sofia	East	0.0	0.0
3	Markus	0	0.0	0.0
4	Edward	West	42000.0	38000.0
5	Thomas	West	72000.0	39000.0
6	Ethan	South	49000.0	42000.0
7	0	0	0.0	0.0
8	Arun	West	67000.0	39000.0
9	Anika	East	65000.0	50000.0
10	Paulo	South	67000.0	45000.0

In [119]:

```
sales_data['sales'].fillna(sales_data['sales'].median())
```

Out[119]:

```
0     50000.0
1     52000.0
2     58500.0
3     58500.0
4     42000.0
5     72000.0
6     49000.0
7     58500.0
8     67000.0
9     65000.0
10    67000.0
Name: sales, dtype: float64
```

In [1]:
```
import pandas as pd
dataset = pd.read_csv('https://raw.githubusercontent.com/Jovita7/Data-Analys
print(dataset)
```

```
      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 [2]:
```
dataset[dataset['horsepower']=='?']
```

Out[2]:

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

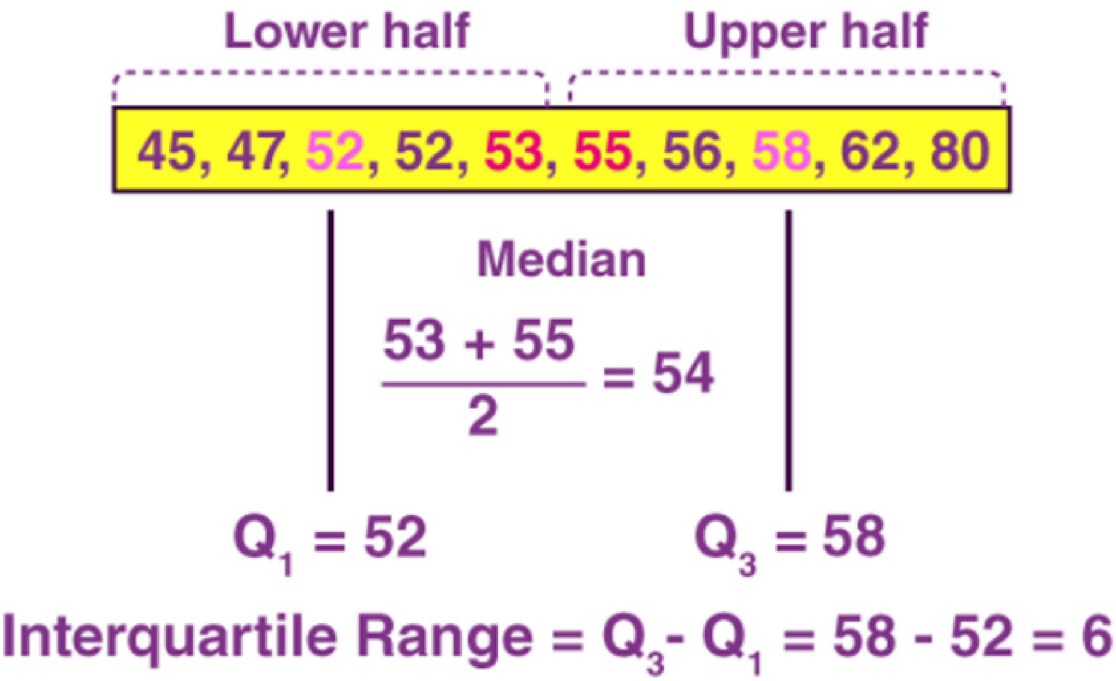
```
In [19]: dataset = dataset[dataset['horsepower']!='?']
         print(dataset)
```

```
      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  

[392 rows x 9 columns]
```

find and remove outliers



In [1]:

```
import pandas as pd
#finding outliers in 'mpg'
def find_outliers(ds, col):
  quart1 = ds[col].quantile(0.25)
  quart3 = ds[col].quantile(0.75)
  IQR = quart3 - quart1 #Inter-quartile range
  low_val = quart1 - 1.5*IQR
  high_val = quart3 + 1.5*IQR
  ds = ds.loc[(ds[col] < low_val) | (ds[col] > high_val)]
  return ds
dataset = pd.read_csv('https://raw.githubusercontent.com/Jovita7/Data-Analys
find_outliers(dataset, 'acceleration')
```

Out[1]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	model year	origin	car na
7	14.0	8	440.0	215	4312	8.5	70	1	plymo fur
9	15.0	8	390.0	190	3850	8.5	70	1	a ambassa
11	14.0	8	340.0	160	3609	8.0	70	1	plymo 'cuda :
59	23.0	4	97.0	54	2254	23.5	72	2	volkswa typ
195	29.0	4	85.0	52	2035	22.2	76	1	chevr chev
299	27.2	4	141.0	71	3190	24.8	79	2	peug :
300	23.9	8	260.0	90	3420	22.2	79	1	oldsmo cutl sa brough
326	43.4	4	90.0	48	2335	23.7	80	2	vw das (die
394	44.0	4	97.0	52	2130	24.6	82	2	vw picl

In [20]:

```
def remove_outliers(ds, col):
  quart1 = ds[col].quantile(0.25)
  quart3 = ds[col].quantile(0.75)
  IQR = quart3 - quart1 #Interquartile range
  low_val = quart1 - 1.5*IQR
  print(low_val)
  high_val = quart3 + 1.5*IQR
  print(high_val)
  df_out = ds.loc[(ds[col] >= low_val) & (ds[col] <= high_val)]
  return df_out
new_data = remove_outliers(dataset, 'acceleration')
```

```
8.900000000000007
21.89999999999999
```

In [21]:
```
print(new_data)
```

```
      mpg  cylinders  displacement horsepower  weight  acceleration  \
0    18.0          8         307.0        130    3504          12.0   
1    15.0          8         350.0        165    3693          11.5   
2    18.0          8         318.0        150    3436          11.0   
3    16.0          8         304.0        150    3433          12.0   
4    17.0          8         302.0        140    3449          10.5   
..    ...        ...           ...        ...     ...           ...   
392  27.0          4         151.0         90    2950          17.3   
393  27.0          4         140.0         86    2790          15.6   
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  
..          ...     ...                        ...  
392          82       1           chevrolet camaro  
393          82       1            ford mustang gl  
395          82       1              dodge rampage  
396          82       1                ford ranger  
397          82       1                 chevy s-10  

[381 rows x 9 columns]
```

In [126]:
```
#7 outliers removed from new_data
new_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 389 entries, 0 to 397
Data columns (total 9 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   mpg           389 non-null    float64
 1   cylinders     389 non-null    int64  
 2   displacement  389 non-null    float64
 3   horsepower    389 non-null    object 
 4   weight        389 non-null    int64  
 5   acceleration  389 non-null    float64
 6   model year    389 non-null    int64  
 7   origin        389 non-null    int64  
 8   car name      389 non-null    object 
dtypes: float64(3), int64(4), object(2)
memory usage: 30.4+ KB
```

Drop or remove Duplicates

In [22]:
```
dataset.drop('mpg', axis = 1)
```

Out[22]:

	cylinders	displacement	horsepower	weight	acceleration	model year	origin	car name
0	8	307.0	130	3504	12.0	70	1	chevrolet chevelle malibu
1	8	350.0	165	3693	11.5	70	1	buick skylark 320
2	8	318.0	150	3436	11.0	70	1	plymouth satellite
3	8	304.0	150	3433	12.0	70	1	amc rebel sst
4	8	302.0	140	3449	10.5	70	1	ford torino
...	...	...	...	...	...	...	...	...
393	4	140.0	86	2790	15.6	82	1	ford mustang gl
394	4	97.0	52	2130	24.6	82	2	vw pickup
395	4	135.0	84	2295	11.6	82	1	dodge rampage
396	4	120.0	79	2625	18.6	82	1	ford ranger
397	4	119.0	82	2720	19.4	82	1	chevy s-10

392 rows × 8 columns

In [23]:
```
print(dataset.shape)
```

```
(392, 9)
```

In [129]:
```
#drop duplicates

import pandas as pd

data = {
    "A": ["TeamA", "TeamB", "TeamB", "TeamC", "TeamA"],
    "B": [50, 40, 40, 30, 50],
    "C": [True, False, False, False, True]
}

df = pd.DataFrame(data)

dups = df.duplicated()
```

In [130]:

```
print(dups)
```

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

In [132]:

```
df = df.drop_duplicates()
print(df)
```

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

In [133]:

```
df = df.reset_index(drop=True)
print(df)
```

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