

What is Pandas

Pandas is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the Python programming language.

- Prudhvi Vardhan Notes



Pandas Series

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

Importing Pandas

```
In [4]: import numpy as np
import pandas as pd
```

Series using String

```
In [6]: # string
country = ['India', 'Pakistan', 'USA', 'Nepal', 'Srilanka']
pd.Series(country)
```

```
Out[6]: 0      India
1    Pakistan
2         USA
3       Nepal
4    Srilanka
dtype: object
```

```
In [7]: # integers
marks = [13, 24, 56, 78, 100]
pd.Series(marks)
```

```
Out[7]: 0      13
1      24
2      56
3      78
4     100
dtype: int64
```

```
In [8]: # custom index
marks = [67,57,89,100]
subjects = ['maths','english','science','hindi']

pd.Series(marks,index=subjects)
```

```
Out[8]: maths      67
english    57
science    89
hindi     100
dtype: int64
```

```
In [10]: # setting a name
marks = pd.Series(marks , index=subjects , name="Jack Marks")
marks
```

```
Out[10]: maths      67
english    57
science    89
hindi     100
Name: Jack Marks, dtype: int64
```

Series from dictionary

When a Pandas Series is converted to a dictionary using the `to_dict()` method, the resulting dictionary has the same keys and values as the Series. The index values of the Series become the keys in the dictionary, and the corresponding values become the values in the dictionary.

```
In [11]: marks = {
          'maths':67,
          'english':57,
          'science':89,
          'hindi':100
        }
marks_series = pd.Series(marks,name="jack Marks")
```

```
In [12]: marks_series
```

```
Out[12]: maths      67
english    57
science    89
hindi     100
Name: jack Marks, dtype: int64
```

Series Attributes

size: Returns the number of elements in the Series.

```
In [13]: # size
marks_series.size
```

```
Out[13]: 4
```

dtype: Returns the data type of the values in the Series.

```
In [14]: # dtype
marks_series.dtype
```

```
Out[14]: dtype('int64')
```

name: Returns the name of the Series.

```
In [15]: # name
marks_series.name
```

```
Out[15]: 'jack Marks'
```

unique is an attribute of a Pandas Series that returns an array of the unique values in the Series.

```
In [16]: # is_unique
marks_series.is_unique
```

```
Out[16]: True
```

```
In [17]: pd.Series([1,1,2,3,4,44,2]).is_unique #It gives false because of repetation
```

```
Out[17]: False
```

index: Returns the index labels of the Series.

```
In [18]: # index
marks_series.index
```

```
Out[18]: Index(['maths', 'english', 'science', 'hindi'], dtype='object')
```

values: Returns the data contained in the Series as a NumPy array.

```
In [19]: # values
marks_series.values
```

```
Out[19]: array([ 67,  57,  89, 100], dtype=int64)
```

```
In [20]: type(marks_series.values)
```

```
Out[20]: numpy.ndarray
```

Series using read_csv

```
In [21]: # with one col
sub = pd.read_csv("D:\\datascience\\Nitish isr\\Pandas\\subs.csv")
```

Pandas.read_csv

Automatically converts everything into data frames not in series.

```
In [23]: type(sub)
```

```
Out[23]: pandas.core.frame.DataFrame
```

```
In [30]: sub.head(5)
```

```
Out[30]:
```

	Subscribers gained
0	48
1	57
2	40
3	43
4	44

To convert data into series,

we have to apply a parameter called as "Squeeze" is equals to True.

```
In [31]: sub = pd.read_csv("subs.csv",squeeze=True)
```

```
In [32]: type(sub)
```

```
Out[32]: pandas.core.series.Series
```

```
In [33]: sub
```

```
Out[33]: 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 [56]: #With 2 col
kl=pd.read_csv("kohli_ip1.csv",index_col="match_no",squeeze=True)
```

In [57]:

kl

Out[57]:

```
match_no
1      1
2     23
3     13
4     12
5      1
..
211    0
212    20
213    73
214    25
215     7
Name: runs, Length: 215, dtype: int64
```

In [37]:

```
movies=pd.read_csv( "bollywood.csv", index_col="movie",squeeze=True)
```

In [38]:

movies

Out[38]:

```
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

head(n): Returns the first n elements of the Series.

In [40]:

```
# Head
sub.head()
```

Out[40]:

```
0      48
1      57
2      40
3      43
4      44
Name: Subscribers gained, dtype: int64
```

tail(n): Returns the last n elements of the Series.

```
In [41]: # tail
kl.tail()
```

```
Out[41]: match_no
211      0
212     20
213     73
214     25
215      7
Name: runs, dtype: int64
```

```
In [43]: # sample - Gives random data
movies.sample()
```

```
Out[43]: movie
Enemmy    Sunil Shetty
Name: lead, dtype: object
```

value_counts(): Returns a Series containing the counts of unique values in the Series.

```
In [44]: # Value Counts
movies.value_counts()
```

```
Out[44]: Akshay Kumar      48
Amitabh Bachchan    45
Ajay Devgn         38
Salman Khan        31
Sanjay Dutt        26
..
Diganth            1
Parveen Kaur       1
Seema Azmi         1
Akanksha Puri      1
Edwin Fernandes   1
Name: lead, Length: 566, dtype: int64
```

```
In [45]: #sort_values - temporary changes ##### sort_values(): Returns a sorted Series by the values
kl.sort_values()
```

```
Out[45]: match_no
87      0
211     0
207     0
206     0
91      0
...
164    100
120    100
123    108
126    109
128    113
Name: runs, Length: 215, dtype: int64
```

```
In [50]: # method chaining
kl.sort_values(ascending=False).head(1).values[0]
```

```
Out[50]: 113
```

```
In [55]: # For permanent Changes use Inplace
kl.sort_values(inplace=True)
kl
```

```
Out[55]: match_no
87      0
211     0
207     0
206     0
91      0
...
164    100
120    100
123    108
126    109
128    113
Name: runs, Length: 215, dtype: int64
```

```
In [60]: # sort_index -> inplace -> movies

movies.sort_index()
```

```
Out[60]: movie
1920 (film)          Rajnesh Duggall
1920: London          Sharman Joshi
1920: The Evil Returns  Vicky Ahuja
1971 (2007 film)      Manoj Bajpayee
2 States (2014 film)   Arjun Kapoor
...
Zindagi 50-50          Veena Malik
Zindagi Na Milegi Dobara  Hrithik Roshan
Zindagi Tere Naam       Mithun Chakraborty
Zokkomon               Darsheel Safary
Zor Lagaa Ke...Haiya!    Meghan Jadhav
Name: lead, Length: 1500, dtype: object
```

```
In [61]: movies.sort_index(ascending=False)
```

```
Out[61]: movie
Zor Lagaa Ke...Haiya!    Meghan Jadhav
Zokkomon                 Darsheel Safary
Zindagi Tere Naam        Mithun Chakraborty
Zindagi Na Milegi Dobara  Hrithik Roshan
Zindagi 50-50            Veena Malik
...
2 States (2014 film)      Arjun Kapoor
1971 (2007 film)         Manoj Bajpayee
1920: The Evil Returns    Vicky Ahuja
1920: London              Sharman Joshi
1920 (film)               Rajnesh Duggall
Name: lead, Length: 1500, dtype: object
```

Series Maths Methods

Diffence between Count And Size

Count gives the total number of items present in the series. But only NON missing values but, if we have missing values ,it doesnt count them . But, size gives the total item including missing values

```
In [62]: # count
kl.count()
```

Out[62]: 215

sum(): Returns the sum of the values in the Series.

```
In [66]: # sum -> Product
sub.sum()
```

Out[66]: 49510

```
In [67]: sub.product() # Multiply the items
```

Out[67]: 0

Static Methods

mean(): Returns the mean value of the Series.

```
In [68]: # mean
```

```
sub.mean()
```

Out[68]: 135.64383561643837

median(): Returns the median value of the Series.

```
In [72]: # median
kl.median()
```

Out[72]: 24.0

mode(): The mode is the value that appears most frequently in the Series.

```
In [74]: # mode
print(movies.mode())
```

```
0    Akshay Kumar
dtype: object
```

std(): Returns the standard deviation of the values in the Series.

```
In [71]: # std -> Standard deviation
sub.std()
```

Out[71]: 62.67502303725269


```
In [75]: # var -> variance  
sub.var()
```

```
Out[75]: 3928.1585127201556
```

min(): Returns the minimum value of the Series.

```
In [76]: # min  
sub.min()
```

```
Out[76]: 33
```

max(): Returns the maximum value of the Series.

```
In [77]: # max  
sub.max()
```

```
Out[77]: 396
```

describe(): Generates descriptive statistics of the Series.

```
In [79]: # describe  
movies.describe()
```

```
Out[79]: count          1500  
unique           566  
top      Akshay Kumar  
freq           48  
Name: lead, dtype: object
```

```
In [80]: kl.describe()
```

```
Out[80]: count    215.000000  
mean      30.855814  
std       26.229801  
min        0.000000  
25%        9.000000  
50%       24.000000  
75%       48.000000  
max      113.000000  
Name: runs, dtype: float64
```

```
In [81]: sub.describe()
```

```
Out[81]: 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
```

Series Indexing

```
In [83]: # integer indexing
x = pd.Series([12,13,14,35,46,57,58,79,9])
x[1]
```

Out[83]: 13

```
In [85]: # negative indexing
movies[-1]
```

Out[85]: 'Akshay Kumar'

```
In [86]: movies[0]
```

Out[86]: 'Vicky Kaushal'

```
In [87]: sub[0]
```

Out[87]: 48

```
In [90]: # slicing
kl[4:10]
```

Out[90]: match_no
5 1
6 9
7 34
8 0
9 21
10 3
Name: runs, dtype: int64

```
In [95]: #Negative slicing
sub[-5:]
```

Out[95]: 360 231
361 226
362 155
363 144
364 172
Name: Subscribers gained, dtype: int64

```
In [96]: movies[-5:]
```

Out[96]: 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 [97]: movies[::2]
```

```
Out[97]: 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 [98]: # Fancy indexing
kl[[1,8,22,11,2]]
```

```
Out[98]: match_no
1      1
8      0
22     38
11     10
2      23
Name: runs, dtype: int64
```

```
In [99]: # Fancy indexing -> indexing with Labels
movies
```

```
Out[99]: 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 [100]: movies['Evening Shadows']
```

```
Out[100]: 'Mona Ambegaonkar'
```

Editing the series

```
In [101]: # using the index number
marks_series
```

```
Out[101]: maths      67
english    57
science    89
hindi      100
Name: jack Marks, dtype: int64
```

```
In [102]: marks_series[1]=88
marks_series
```

```
Out[102]: maths      67
english    88
science    89
hindi     100
Name: jack Marks, dtype: int64
```

```
In [103]: # we can add data , if it doesnt exist
marks_series['social']=90
marks_series
```

```
Out[103]: maths      67
english    88
science    89
hindi     100
social     90
Name: jack Marks, dtype: int64
```

```
In [111]: # using index label
movies
```

```
Out[111]: 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 [114]: movies['Hum Tumhare Hain Sanam'] = 'Jack'
```

```
In [115]: movies
```

```
Out[115]: 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           Jack
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 with Python Functionalities

```
In [117]: # len/type/dir/sorted/max/min
print(len(sub))
print(type(sub))
```

```
365 <class 'pandas.core.series.Series'>
```

```
In [122]: print(dir(sub))
          print(sorted(sub))
```

```

ipate', '_validate_dtype', '_values', '_where', 'abs', 'add', 'add_prefix', 'add_suff
ix', 'agg', 'aggregate', 'align', 'all', 'any', 'append', 'apply', 'argmax', 'argmin',
'argsort', 'array', 'asfreq', 'asof', 'astype', 'at', 'at_time', 'attrs', 'autocorr',
'axes', 'backfill', 'between', 'between_time', 'bfill', 'bool', 'clip', 'combine', 'co
mbine_first', 'compare', 'convert_dtypes', 'copy', 'corr', 'count', 'cov', 'cummax',
'cummin', 'cumprod', 'cumsum', 'describe', 'diff', 'div', 'divide', 'divmod', 'dot',
'drop', 'drop_duplicates', 'droplevel', 'dropna', 'dtype', 'dtypes', 'duplicated', 'em
pty', 'eq', 'equals', 'ewm', 'expanding', 'explode', 'factorize', 'ffill', 'fillna',
'filter', 'first', 'first_valid_index', 'flags', 'floordiv', 'ge', 'get', 'groupby',
'gt', 'hasnans', 'head', 'hist', 'iat', 'idxmax', 'idxmin', 'iloc', 'index', 'infer_ob
jects', 'interpolate', 'is_monotonic', 'is_monotonic_decreasing', 'is_monotonic_increa
sing', 'is_unique', 'isin', 'isna', 'isnull', 'item', 'items', 'iteritems', 'keys', 'k
urt', 'kurtosis', 'last', 'last_valid_index', 'le', 'loc', 'lt', 'mad', 'map', 'mask',
'max', 'mean', 'median', 'memory_usage', 'min', 'mod', 'mode', 'mul', 'multiply', 'nam
e', 'nbytes', 'ndim', 'ne', 'nlargest', 'notna', 'notnull', 'nsmallest', 'nunique', 'p
ad', 'pct_change', 'pipe', 'plot', 'pop', 'pow', 'prod', 'product', 'quantile', 'rad
d', 'rank', 'ravel', 'rdiv', 'rdivmod', 'reindex', 'reindex_like', 'rename', 'rename_a
xis', 'reorder_levels', 'repeat', 'replace', 'resample', 'reset_index', 'rfloordiv',
'rmod', 'rmul', 'rolling', 'round', 'rpow', 'rsub', 'rtruediv', 'sample', 'searchsorte
d', 'sem', 'set_axis', 'set_flags', 'shape', 'shift', 'size', 'skew', 'slice_shift',

```

```
In [123]: print(min(sub))
          print(max(sub))
```

33
396

```
In [125]: # type conversion
          list(marks_series)
```

Out[125]: [67, 88, 89, 100, 90]

```
In [126]: dict(marks_series)
```

```
Out[126]: {'maths': 67, 'english': 88, 'science': 89, 'hindi': 100, 'social': 90}
```

```
In [129]: # membership operator
          'Hum Tumhare Hain Sanam' in movies # In operator only searches in index values
```

Out[129]: True

```
In [133]: "Jack" in movies.values
```

Out[133]: True

```
In [138]: # Looping
for i in movies:
    print(i)
```

Vicky Kaushal
Vicky Ahuja
Anupam Kher
Emraan Hashmi
Mona Ambegaonkar
Geetika Vidya Ohlyan
Arshad Warsi
Radhika Apte
Kangana Ranaut
Nawazuddin Siddiqui
Ali Asgar
Ranveer Singh
Prit Kamani
Ajay Devgn
Sushant Singh Rajput
Amitabh Bachchan
Abhimanyu Dasani
Talha Arshad Reshi
Nawazuddin Siddiqui
~ . . .

```
In [139]: 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 [140]: # Arithmetic Operators (Broadcasting)
100-marks_series
```

```
Out[140]: maths      33
english    12
science    11
hindi      0
social     10
Name: jack Marks, dtype: int64
```

```
In [141]: 100*marks_series
```

```
Out[141]: maths      167
english    188
science    189
hindi      200
social     190
Name: jack Marks, dtype: int64
```

```
In [143]: # Relational operators
kl>=50
```

```
Out[143]: match_no
1      False
2      False
3      False
4      False
5      False
...
211    False
212    False
213     True
214    False
215    False
Name: runs, Length: 215, dtype: bool
```

Boolean Indexing on Series

```
In [146]: # Find no of 50's and 100's scored by kohli
kl[kl>=50].size
```

```
Out[146]: 50
```

```
In [148]: # find number of ducks
kl[kl == 0].size
```

```
Out[148]: 9
```

```
In [149]: # Count number of day when I had more than 200 subs a day
sub[sub>=200].size
```

```
Out[149]: 59
```

```
In [159]: # find actors who have done more than 20 movies
num_mov=movies.value_counts()
num_mov[num_mov>=20]
```

```
Out[159]: Akshay Kumar      48
Amitabh Bachchan    45
Ajay Devgn          38
Salman Khan         31
Sanjay Dutt         26
Shah Rukh Khan      21
Emraan Hashmi       21
Name: lead, dtype: int64
```

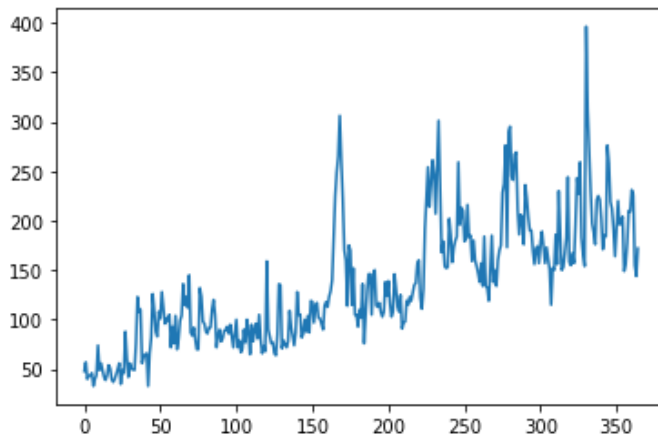
```
In [160]: num_mov[num_mov>=20].size
```

```
Out[160]: 7
```

Plotting Graphs on Series

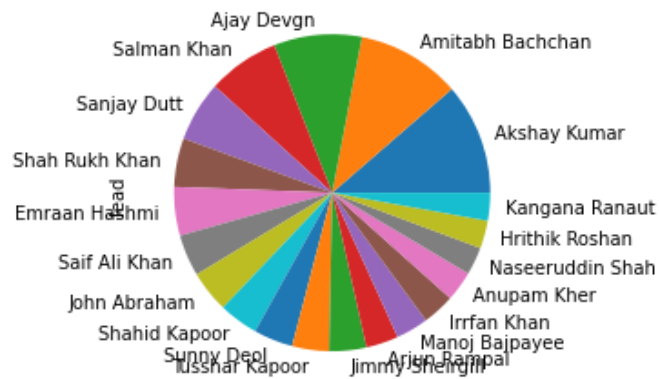
```
In [162]: sub.plot()
```

```
Out[162]: <AxesSubplot:>
```



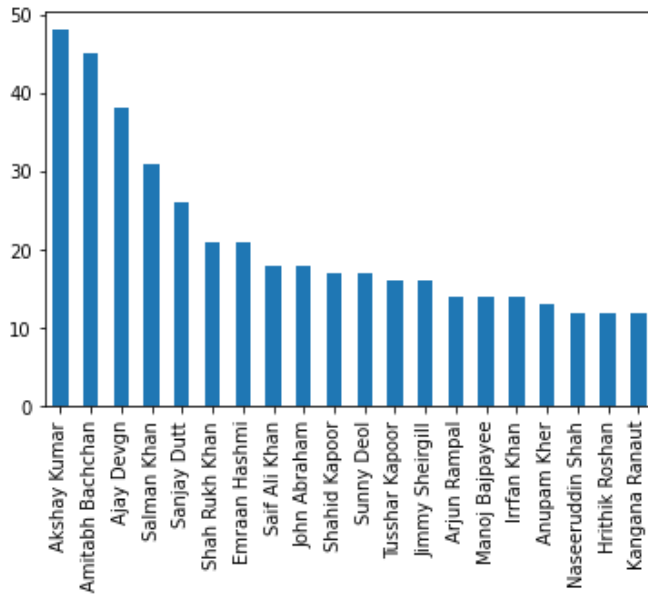
```
In [164]: movies.value_counts().head(20).plot(kind="pie")
```

```
Out[164]: <AxesSubplot:ylabel='lead'>
```




```
In [165]: movies.value_counts().head(20).plot(kind="bar")
```

```
Out[165]: <AxesSubplot:>
```



Some Important Series Methods

```
In [166]: # astype
# between
# clip
# drop_duplicates
# isnull
# dropna
# fillna
# isin
# apply
# copy
```

```
In [175]: # astype
import sys
sys.getsizeof(k1)
```

```
Out[175]: 11752
```

```
In [176]: k1
```

```
Out[176]: match_no
1         1
2        23
3        13
4        12
5         1
..
211        0
212        0
213        73
214        25
215         7
Name: runs, Length: 215, dtype: int64
```

```
In [177]: (kl.astype("int16"))
```

```
Out[177]: match_no
1         1
2        23
3        13
4        12
5         1
..
211        0
212        0
213        7
214        2
215         7
Name: runs, Length: 215, dtype: int16
```

```
In [178]: sys.getsizeof(kl.astype("int16"))
```

```
Out[178]: 10462
```

```
In [181]: # between
kl[kl.between(50,60)]
```

```
Out[181]: match_no
15        50
34        58
44        56
57        57
71        51
73        58
80        57
85        56
103       51
122       52
129       54
131       54
137       55
141       58
144       57
182       50
197       51
198       53
209       58
Name: runs, dtype: int64
```

```
In [182]: kl[kl.between(50,60)].size
```

```
Out[182]: 19
```

```
In [183]: # clip  
sub.clip(100,200)
```

```
Out[183]: 0      100  
          1      100  
          2      100  
          3      100  
          4      100  
          ...  
          360    200  
          361    200  
          362    155  
          363    144  
          364    172  
          Name: Subscribers gained, Length: 365, dtype: int64
```

```
In [186]: # drop duplicates #### drop_duplicates(): Returns a Series with duplicates removed.  
  
dele = pd.Series([1,2,33,3,3,3,1,23,33,22,33,11])  
dele
```

```
Out[186]: 0      1  
          1      2  
          2     33  
          3      3  
          4      3  
          5      3  
          6      1  
          7     23  
          8     33  
          9     22  
         10     33  
         11     11  
          dtype: int64
```

```
In [188]: dele.drop_duplicates()
```

```
Out[188]: 0      1  
          1      2  
          2     33  
          3      3  
          7     23  
          9     22  
         11     11  
          dtype: int64
```

```
In [189]: dele.drop_duplicates(keep='last')
```

```
Out[189]: 1      2  
          5      3  
          6      1  
          7     23  
          9     22  
         10     33  
         11     11  
          dtype: int64
```

```
In [190]: movies.drop_duplicates()
```

```
Out[190]: 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
...
Rules: Pyaar Ka Superhit Formula      Tanuja
Right Here Right Now (film)          Ankit
Talaash: The Hunt Begins...          Rakhee Gulzar
The Pink Mirror                    Edwin Fernandes
Hum Tumhare Hain Sanam              Jack
Name: lead, Length: 567, dtype: object
```

```
In [191]: dele.duplicated().sum()
```

```
Out[191]: 5
```

```
In [193]: kl.duplicated().sum()
```

```
Out[193]: 137
```

```
In [194]: dele.count()
```

```
Out[194]: 12
```

isin(values): Returns a boolean Series indicating whether each element in the Series is in the provided values

```
In [198]: # isnull
```

```
kl.isnull().sum()
```

```
Out[198]: 0
```

```
In [199]: dele.isnull().sum()
```

```
Out[199]: 0
```

```
In [200]: # dropna
```

```
dele.dropna()
```

```
Out[200]: 0      1
1      2
2     33
3      3
4      3
5      3
6      1
7     23
8     33
9     22
10    33
11    11
dtype: int64
```

In [202]: `# fillna`

```
dele.fillna(0)
dele.fillna(dele.mean())
```

Out[202]:

0	1
1	2
2	33
3	3
4	3
5	3
6	1
7	23
8	33
9	22
10	33
11	11

dtype: int64

In [205]: `# isin`

kl

Out[205]:

match_no
1
2
3
4
5
..
211
212
213
214
215

Name: runs, Length: 215, dtype: int64

In [207]: `kl[(kl==49) | (kl==99)]`

Out[207]:

match_no
82
86

Name: runs, dtype: int64

In [209]: `kl[kl.isin([49,99])]`

Out[209]:

match_no
82
86

Name: runs, dtype: int64

In [210]: `# apply`

movies

```
Out[210]: 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           Jack
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 [212]: `movies.apply(lambda x:x.split()) # split name in to two using Lambda function`

```
Out[212]: 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           [Jack]
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 [213]: `movies.apply(lambda x:x.split()[0]) # select first word`

```
Out[213]: movie
Uri: The Surgical Strike          Vicky
Battalion 609                    Vicky
The Accidental Prime Minister (film)  Anupam
Why Cheat India                  Emraan
Evening Shadows                  Mona
...
Hum Tumhare Hain Sanam           Jack
Aankhen (2002 film)              Amitabh
Saathiya (film)                 Vivek
Company (film)                  Ajay
Awara Paagal Deewana            Akshay
Name: lead, Length: 1500, dtype: object
```

```
In [214]: movies.apply(lambda x:x.split()[0].upper()) # Upper case
```

```
Out[214]: movie
Uri: The Surgical Strike          VICKY
Battalion 609                    VICKY
The Accidental Prime Minister (film) ANUPAM
Why Cheat India                  EMRAAN
Evening Shadows                  MONA
...
Hum Tumhare Hain Sanam          JACK
Aankhen (2002 film)             AMITABH
Saathiya (film)                 VIVEK
Company (film)                  AJAY
Awara Paagal Deewana            AKSHAY
Name: lead, Length: 1500, dtype: object
```

```
In [215]: sub
```

```
Out[215]: 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 [216]: sub.mean()
```

```
Out[216]: 135.64383561643837
```

```
In [217]: sub.apply(lambda x:'good day' if x > sub.mean() else 'bad day')
```

```
Out[217]: 0      bad day
1      bad day
2      bad day
3      bad day
4      bad day
...
360    good day
361    good day
362    good day
363    good day
364    good day
Name: Subscribers gained, Length: 365, dtype: object
```

```
In [229]: # Copy
```

```
k1
```

```
Out[229]: match_no
1         1
2        23
3        13
4        12
5         1
..
211        0
212       20
213       73
214       25
215        7
Name: runs, Length: 215, dtype: int64
```

```
In [230]: new = k1.head()
```

```
In [231]: new[1]=100
```

```
In [232]: new
```

```
Out[232]: match_no
1        100
2         23
3         13
4         12
5          1
Name: runs, dtype: int64
```

```
In [233]: k1
```

```
Out[233]: match_no
1         100
2          23
3          13
4          12
5           1
...
211         0
212        20
213        73
214        25
215         7
Name: runs, Length: 215, dtype: int64
```

```
In [240]: new = k1.head(5).copy()
```

```
In [241]: new[1]=20
```


In [242]: new

```
Out[242]: match_no
1      20
2      23
3      13
4      12
5       1
Name: runs, dtype: int64
```

In [250]: k1

```
Out[250]: match_no
1      100
2       23
3       13
4       12
5        1
...
211     0
212    20
213    73
214    25
215     7
Name: runs, Length: 215, dtype: int64
```

In []:

In []:

```
In [1]: import numpy as np
import pandas as pd
```

Creating DataFrame

```
In [2]: # Using the Lists
student_data = [
    [100,80,10],
    [90,70,7],
    [120,100,14],
    [80,50,2]
]

pd.DataFrame(student_data,columns=['iq','marks','package'])
```

```
Out[2]:
```

	iq	marks	package
0	100	80	10
1	90	70	7
2	120	100	14
3	80	50	2

```
In [3]: # using dicts
student_dict = {
    'name':['peter','saint','noeum','parle','samme','dave'],
    'iq':[100,90,120,80,13,90],
    'marks':[80,70,100,50,11,80],
    'package':[10,7,14,2,15,100]
}
students=pd.DataFrame(student_dict)
students
```

```
Out[3]:
```

	name	iq	marks	package
0	peter	100	80	10
1	saint	90	70	7
2	noeum	120	100	14
3	parle	80	50	2
4	samme	13	11	15
5	dave	90	80	100

```
In [4]: students.set_index('name',inplace=True)
students
```

```
Out[4]:
```

	iq	marks	package
name			
peter	100	80	10
saint	90	70	7
noeum	120	100	14
parle	80	50	2
samme	13	11	15
dave	90	80	100

In [5]:

```
# Read csv
movies = pd.read_csv("movies.csv")
movies.head()
```

Out[5]:

	title_x	imdb_id	poster_path	wiki_link	title_y	original_title	is_adult	year_
0	Uri: The Surgical Strike	tt8291224	https://upload.wikimedia.org/wikipedia/en/thum...	https://en.wikipedia.org/wiki/Uri:_The_Surgica...	Uri: The Surgikal Strike	Uri: The Surgical Strike	0	
1	Battalion 609	tt9472208	NaN	https://en.wikipedia.org/wiki/Battalion_609	Battalion 609	Battalion 609	0	
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	0	
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	0	
4	Evening Shadows	tt6028796	NaN	https://en.wikipedia.org/wiki/Evening_Shadows	Evening Shadows	Evening Shadows	0	

In [6]:

```
ipl = pd.read_csv("ipl-matches.csv")
ipl.head()
```

Out[6]:

	ID	City	Date	Season	MatchNumber	Team1	Team2	Venue	TossWinner	TossDecision	SuperOver	WinningTeam	W
0	1312200	Ahmedabad	2022-05-29	2022	Final	Rajasthan Royals	Gujarat Titans	Narendra Modi Stadium, Ahmedabad	Rajasthan Royals	bat	N	Gujarat Titans	Wi
1	1312199	Ahmedabad	2022-05-27	2022	Qualifier 2	Royal Challengers Bangalore	Rajasthan Royals	Narendra Modi Stadium, Ahmedabad	Rajasthan Royals	field	N	Rajasthan Royals	Wi
2	1312198	Kolkata	2022-05-25	2022	Eliminator	Royal Challengers Bangalore	Lucknow Super Giants	Eden Gardens, Kolkata	Lucknow Super Giants	field	N	Royal Challengers Bangalore	
3	1312197	Kolkata	2022-05-24	2022	Qualifier 1	Rajasthan Rovals	Gujarat Titans	Eden Gardens, Kolkata	Gujarat Titans	field	N	Gujarat Titans	Wi

DataFrame Attributes and Methods

In [7]:

```
# shape
movies.shape
```

Out[7]:

(1629, 18)

In [8]:

```
ipl.shape
```

Out[8]:

(950, 20)

```
In [9]: # dtype
movies.dtypes
```

```
Out[9]: title_x          object
imdb_id                object
poster_path           object
wiki_link             object
title_y               object
original_title         object
is_adult               int64
year_of_release        int64
runtime               object
genres                 object
imdb_rating            float64
imdb_votes             int64
story                 object
summary               object
tagline                object
actors                object
wins_nominations       object
release_date           object
dtype: object
```

```
In [10]: ipl.dtypes
```

```
Out[10]: ID              int64
City                  object
Date                  object
Season                object
MatchNumber           object
Team1                 object
Team2                 object
Venue                 object
TossWinner            object
TossDecision          object
SuperOver             object
WinningTeam           object
WonBy                 object
Margin                float64
method                object
Player_of_Match        object
Team1Players           object
Team2Players           object
Umpire1                object
Umpire2                object
dtype: object
```

```
In [11]: # index
movies.index
```

```
Out[11]: RangeIndex(start=0, stop=1629, step=1)
```

```
In [12]: ipl.index
```

```
Out[12]: RangeIndex(start=0, stop=950, step=1)
```

```
In [13]: # Columns
movies.columns
```

```
Out[13]: Index(['title_x', 'imdb_id', 'poster_path', 'wiki_link', 'title_y',
               'original_title', 'is_adult', 'year_of_release', 'runtime', 'genres',
               'imdb_rating', 'imdb_votes', 'story', 'summary', 'tagline', 'actors',
               'wins_nominations', 'release_date'],
               dtype='object')
```

```
In [14]: ipl.columns
```

```
Out[14]: Index(['ID', 'City', 'Date', 'Season', 'MatchNumber', 'Team1', 'Team2',
               'Venue', 'TossWinner', 'TossDecision', 'SuperOver', 'WinningTeam',
               'WonBy', 'Margin', 'method', 'Player_of_Match', 'Team1Players',
               'Team2Players', 'Umpire1', 'Umpire2'],
               dtype='object')
```

```
In [15]: # Values
students.values

Out[15]: array([[100, 80, 10],
                [ 90, 70, 7],
                [120, 100, 14],
                [ 80, 50, 2],
                [ 13, 11, 15],
                [ 90, 80, 100]], dtype=int64)

In [16]: ipl.values

Out[16]: array([[1312200, 'Ahmedabad', '2022-05-29', ...,
                "['WP Saha', 'Shubman Gill', 'MS Wade', 'HH Pandya', 'DA Miller', 'R Tewatia', 'Rashid Khan', 'R Sai Kishore',
                'LH Ferguson', 'Yash Dayal', 'Mohammed Shami']]",
                'CB Gaffaney', 'Nitin Menon'],
                [1312199, 'Ahmedabad', '2022-05-27', ...,
                "['YBK Jaiswal', 'JC Buttler', 'SV Samson', 'D Padikkal', 'SO Hetmyer', 'R Parag', 'R Ashwin', 'TA Boult', 'YS C
                hahal', 'M Prasidh Krishna', 'OC McCoy']]",
                'CB Gaffaney', 'Nitin Menon'],
                [1312198, 'Kolkata', '2022-05-25', ...,
                "['Q de Kock', 'KL Rahul', 'M Vohra', 'DJ Hooda', 'MP Stoinis', 'E Lewis', 'KH Pandya', 'PVD Chameera', 'Mohsin
                Khan', 'Avesh Khan', 'Ravi Bishnoi']]",
                'J Madanagopal', 'MA Gough'],
                ...,
                [335984, 'Delhi', '2008-04-19', ...,
                "['T Kohli', 'YK Pathan', 'SR Watson', 'M Kaif', 'DS Lehmann', 'RA Jadeja', 'M Rawat', 'D Salunkhe', 'SK Warne',
                'SK Trivedi', 'MM Patel']]",
                'Aleem Dar', 'GA Pratapkumar'],
                [335983, 'Chandigarh', '2008-04-19', ...,
                "['PA Patel', 'ML Hayden', 'MEK Hussey', 'MS Dhoni', 'SK Raina', 'JDP Oram', 'S Badrinath', 'Joginder Sharma',
                'P Amarnath', 'MS Gony', 'M Muralitharan']]",
                'MR Benson', 'SL Shastri'],
                [335982, 'Bangalore', '2008-04-18', ...,
                "['SC Ganguly', 'BB McCullum', 'RT Ponting', 'DJ Hussey', 'Mohammad Hafeez', 'LR Shukla', 'WP Saha', 'AB Agarka
                r', 'AB Dinda', 'M Kartik', 'I Sharma']]",
                'Asad Rauf', 'RE Koertzen']]], dtype=object)

In [17]: # Sample -> to reduce bias
ipl.sample(2)
```

Out[17]:

	ID	City	Date	Season	MatchNumber	Team1	Team2	Venue	TossWinner	TossDecision	SuperOver	WinningTeam	WonBy	Ma
226	1178402	Mumbai	2019-04-13	2019	27	Mumbai Indians	Rajasthan Royals	Wankhede Stadium	Rajasthan Royals	field	N	Rajasthan Royals	Wickets	
63	1304057	Mumbai	2022-04-03	2022	11	Punjab Kings	Chennai Super Kings	Brabourne Stadium, Mumbai	Chennai Super Kings	field	N	Punjab Kings	Runs	:

In [18]: `# info`
`movies.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 [19]: `ipl.info()`

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 950 entries, 0 to 949
Data columns (total 20 columns):
#   Column                Non-Null Count  Dtype
---  -
0   ID                    950 non-null    int64
1   City                  899 non-null    object
2   Date                  950 non-null    object
3   Season                950 non-null    object
4   MatchNumber           950 non-null    object
5   Team1                 950 non-null    object
6   Team2                 950 non-null    object
7   Venue                 950 non-null    object
8   TossWinner            950 non-null    object
9   TossDecision          950 non-null    object
10  SuperOver             946 non-null    object
11  WinningTeam           946 non-null    object
12  WonBy                 950 non-null    object
13  Margin                932 non-null    float64
14  method                19 non-null     object
15  Player_of_Match       946 non-null    object
16  Team1Players           950 non-null    object
17  Team2Players           950 non-null    object
18  Umpire1               950 non-null    object
19  Umpire2               950 non-null    object
dtypes: float64(1), int64(1), object(18)
memory usage: 148.6+ KB
```

In [20]: `# describe`
`movies.describe()`

Out[20]:

	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 [21]: ipl.describe()
```

Out[21]:

	ID	Margin
count	9.500000e+02	932.000000
mean	8.304852e+05	17.056867
std	3.375678e+05	21.633109
min	3.359820e+05	1.000000
25%	5.012612e+05	6.000000
50%	8.297380e+05	8.000000
75%	1.175372e+06	19.000000
max	1.312200e+06	146.000000

```
In [22]: # isnull
movies.isnull()
```

Out[22]:

	title_x	imdb_id	poster_path	wiki_link	title_y	original_title	is_adult	year_of_release	runtime	genres	imdb_rating	imdb_votes	story	summary
0	False	False	False	False	False	False	False	False	False	False	False	False	False	False
1	False	False	True	False	False	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False	False	False	False	False	False
4	False	False	True	False	False	False	False	False	False	False	False	False	False	False
...
1624	False	False	False	False	False	False	False	False	False	False	False	False	False	False
1625	False	False	False	False	False	False	False	False	False	False	False	False	False	False
1626	False	False	True	False	False	False	False	False	False	False	False	False	False	False
1627	False	False	False	False	False	False	False	False	False	False	False	False	False	False
1628	False	False	False	False	False	False	False	False	False	False	False	False	False	False

1629 rows × 18 columns

```
In [23]: movies.isnull().sum()
```

Out[23]:

title_x	0
imdb_id	0
poster_path	103
wiki_link	0
title_y	0
original_title	0
is_adult	0
year_of_release	0
runtime	0
genres	0
imdb_rating	0
imdb_votes	0
story	20
summary	0
tagline	1072
actors	5
wins_nominations	922
release_date	107
dtype:	int64

```
In [24]: # duplicated
movies.duplicated().sum()
```

Out[24]: 0

```
In [25]: # rename
students
```

```
Out[25]:
```

	iq	marks	package
name			
peter	100	80	10
saint	90	70	7
noeum	120	100	14
parle	80	50	2
samme	13	11	15
dave	90	80	100

```
In [26]: students.rename(columns={'marks':'percent','package':'lpa'},inplace=True)
```

```
In [ ]: students.drop(columns='name',inplace=True)
```

Maths Method

```
In [28]: # sum -> Axis Argument
students.sum(axis=1)
```

```
Out[28]: name
peter    190
saint    167
noeum    234
parle    132
samme     39
dave    270
dtype: int64
```

```
In [29]: # mean
students.mean()
```

```
Out[29]: iq          82.166667
percent    65.166667
lpa        24.666667
dtype: float64
```

```
In [30]: students.min(axis=1)
```

```
Out[30]: name
peter     10
saint      7
noeum     14
parle      2
samme     11
dave      80
dtype: int64
```

```
In [31]: students.var()
```

```
Out[31]: iq          1332.166667
percent    968.166667
lpa        1384.666667
dtype: float64
```


Selecting cols from a DataFrame

```
In [32]: # single cols
movies['title_x']
```

```
Out[32]: 0          Uri: The Surgical Strike
1          Battalion 609
2    The Accidental Prime Minister (film)
3          Why Cheat India
4          Evening Shadows
...
1624         Tera Mera Saath Rahen
1625         Yeh Zindagi Ka Safar
1626         Sabse Bada Sukh
1627          Daaka
1628         Humsafar
Name: title_x, Length: 1629, dtype: object
```

```
In [33]: type(movies['title_x'])
```

```
Out[33]: pandas.core.series.Series
```

```
In [137]: # multiple columns
movies[['year_of_release', 'actors', 'title_x']].head(2)
```

```
In [35]: type(movies[['year_of_release', 'actors', 'title_x']].head(2))
```

```
Out[35]: pandas.core.frame.DataFrame
```

```
In [36]: ipl[['City', 'Team1', 'Team2' ]]
```

```
Out[36]:
```

	City	Team1	Team2
0	Ahmedabad	Rajasthan Royals	Gujarat Titans
1	Ahmedabad	Royal Challengers Bangalore	Rajasthan Royals
2	Kolkata	Royal Challengers Bangalore	Lucknow Super Giants
3	Kolkata	Rajasthan Royals	Gujarat Titans
4	Mumbai	Sunrisers Hyderabad	Punjab Kings
...
945	Kolkata	Kolkata Knight Riders	Deccan Chargers
946	Mumbai	Mumbai Indians	Royal Challengers Bangalore
947	Delhi	Delhi Daredevils	Rajasthan Royals
948	Chandigarh	Kings XI Punjab	Chennai Super Kings
949	Bangalore	Royal Challengers Bangalore	Kolkata Knight Riders

950 rows × 3 columns

```
In [37]: student_dict = {
    'name': ['peter', 'saint', 'noeum', 'parle', 'samme', 'dave'],
    'iq': [100, 90, 120, 80, 13, 90],
    'marks': [80, 70, 100, 50, 11, 80],
    'package': [10, 7, 14, 2, 15, 100]
}
students = pd.DataFrame(student_dict)
students.set_index('name', inplace=True)
```

```
In [38]: students
```

```
Out[38]:
```

	iq	marks	package
name			
peter	100	80	10
saint	90	70	7
noeum	120	100	14
parle	80	50	2
samme	13	11	15
dave	90	80	100

Selecting rows from a DataFrame

- **iloc** - searches using index positions
- **loc** - searches using index labels

```
In [39]: # single_row
movies.iloc[1]
```

Out[39]:

title_x	Battalion 609
imdb_id	tt9472208
poster_path	NaN
wiki_link	https://en.wikipedia.org/wiki/Battalion_609 (https://en.wikipedia.org/wiki/Battalion_609)
title_y	Battalion 609
original_title	Battalion 609
is_adult	0
year_of_release	2019
runtime	131
genres	War
imdb_rating	4.1
imdb_votes	73
story	The story revolves around a cricket match betw...
summary	The story of Battalion 609 revolves around a c...
tagline	NaN
actors	Vicky Ahuja Shoaib Ibrahim Shrikant Kamat Elen...
wins_nominations	NaN
release_date	11 January 2019 (India)

Name: 1, dtype: object

```
In [40]: # Multiple rows
movies.iloc[5:10]
```

Out[40]:

	title_x	imdb_id	poster_path	wiki_link	title_y	original_title	is_adult	y
5	Soni (film)	tt6078866	https://upload.wikimedia.org/wikipedia/en/thum...	https://en.wikipedia.org/wiki/Soni_(film)	Soni	Soni	0	
6	Fraud Saiyaan	tt5013008	https://upload.wikimedia.org/wikipedia/en/thum...	https://en.wikipedia.org/wiki/Fraud_Saiyaan	Fraud Saiyaan	Fraud Saiyyan	0	
7	Bombairiya	tt4971258	https://upload.wikimedia.org/wikipedia/en/thum...	https://en.wikipedia.org/wiki/Bombairiya	Bombairiya	Bombairiya	0	
8	Manikarnika: The Queen of Jhansi	tt6903440	https://upload.wikimedia.org/wikipedia/en/thum...	https://en.wikipedia.org/wiki/Manikarnika:_The...	Manikarnika: The Queen of Jhansi	Manikarnika: The Queen of Jhansi	0	
9	Thackeray (film)	tt7777196	https://upload.wikimedia.org/wikipedia/en/thum...	https://en.wikipedia.org/wiki/Thackeray_(film)	Thackeray	Thackeray	0	

In [41]:

movies.iloc[5:12:2]

Out[41]:

	title_x	imdb_id	poster_path	wiki_link	title_y	original_title	is_adult	year_
5	Soni (film)	tt6078866	https://upload.wikimedia.org/wikipedia/en/thum...	https://en.wikipedia.org/wiki/Soni_(film)	Soni	Soni	0	
7	Bombairiya	tt4971258	https://upload.wikimedia.org/wikipedia/en/thum...	https://en.wikipedia.org/wiki/Bombairiya	Bombairiya	Bombairiya	0	
9	Thackeray (film)	tt7777196	https://upload.wikimedia.org/wikipedia/en/thum...	https://en.wikipedia.org/wiki/Thackeray_(film)	Thackeray	Thackeray	0	
11	Gully Boy	tt2395469	https://upload.wikimedia.org/wikipedia/en/thum...	https://en.wikipedia.org/wiki/Gully_Boy	Gully Boy	Gully Boy	0	

In [42]:

Fancy indexing
ipl.iloc[[0,4,5]]

Out[42]:

	ID	City	Date	Season	MatchNumber	Team1	Team2	Venue	TossWinner	TossDecision	SuperOver	WinningTeam	WonBy
0	1312200	Ahmedabad	2022-05-29	2022	Final	Rajasthan Royals	Gujarat Titans	Narendra Modi Stadium, Ahmedabad	Rajasthan Royals	bat	N	Gujarat Titans	Wickets
4	1304116	Mumbai	2022-05-22	2022	70	Sunrisers Hyderabad	Punjab Kings	Wankhede Stadium, Mumbai	Sunrisers Hyderabad	bat	N	Punjab Kings	Wickets
5	1304115	Mumbai	2022-05-21	2022	69	Delhi Capitals	Mumbai Indians	Wankhede Stadium, Mumbai	Mumbai Indians	field	N	Mumbai Indians	Wickets

In [43]:

Loc (Location)
students

Out[43]:

	iq	marks	package
name			
peter	100	80	10
saint	90	70	7
noeum	120	100	14
parle	80	50	2
samme	13	11	15
dave	90	80	100

In [44]:

students.loc['parle']

Out[44]:

iq 80
marks 50
package 2
Name: parle, dtype: int64

```
In [45]: students.loc['saint':'samme':2]
```

```
Out[45]:
```

	iq	marks	package
name			
saint	90	70	7
parle	80	50	2

```
In [46]: # Fancy indexing
students.loc[['saint','dave']]
```

```
Out[46]:
```

	iq	marks	package
name			
saint	90	70	7
dave	90	80	100

```
In [47]: students.iloc[[0,4,3]]
```

```
Out[47]:
```

	iq	marks	package
name			
peter	100	80	10
samme	13	11	15
parle	80	50	2

Selecting both rows and cols

```
In [48]: movies.iloc[0:3,0:3]
```

```
Out[48]:
```

	title_x	imdb_id	poster_path
0	Uri: The Surgical Strike	tt8291224	https://upload.wikimedia.org/wikipedia/en/thum...
1	Battalion 609	tt9472208	NaN
2	The Accidental Prime Minister (film)	tt6986710	https://upload.wikimedia.org/wikipedia/en/thum...

```
In [49]: movies.loc[0:2,'title_x':'poster_path']
```

```
Out[49]:
```

	title_x	imdb_id	poster_path
0	Uri: The Surgical Strike	tt8291224	https://upload.wikimedia.org/wikipedia/en/thum...
1	Battalion 609	tt9472208	NaN
2	The Accidental Prime Minister (film)	tt6986710	https://upload.wikimedia.org/wikipedia/en/thum...

Filtering a DataFrame

```
In [50]: ipl.head(2)
```

```
Out[50]:
```

	ID	City	Date	Season	MatchNumber	Team1	Team2	Venue	TossWinner	TossDecision	SuperOver	WinningTeam	WonB
0	1312200	Ahmedabad	2022-05-29	2022	Final	Rajasthan Royals	Gujarat Titans	Narendra Modi Stadium, Ahmedabad	Rajasthan Royals	bat	N	Gujarat Titans	Wicke
1	1312199	Ahmedabad	2022-05-27	2022	Qualifier 2	Royal Challengers Bangalore	Rajasthan Royals	Narendra Modi Stadium, Ahmedabad	Rajasthan Royals	field	N	Rajasthan Royals	Wicke

```
In [51]: # find all the final winners

mask=ipl['MatchNumber'] == 'Final'
new_df= ipl[mask]
new_df[['Season', 'WinningTeam']]
```

Out[51]:

	Season	WinningTeam
0	2022	Gujarat Titans
74	2021	Chennai Super Kings
134	2020/21	Mumbai Indians
194	2019	Mumbai Indians
254	2018	Chennai Super Kings
314	2017	Mumbai Indians
373	2016	Sunrisers Hyderabad
433	2015	Mumbai Indians
492	2014	Kolkata Knight Riders
552	2013	Mumbai Indians
628	2012	Kolkata Knight Riders
702	2011	Chennai Super Kings
775	2009/10	Chennai Super Kings
835	2009	Deccan Chargers
892	2007/08	Rajasthan Royals

```
In [52]: # In one Line
ipl[ipl['MatchNumber']=='Final'][['Season', 'WinningTeam']]
```

Out[52]:

	Season	WinningTeam
0	2022	Gujarat Titans
74	2021	Chennai Super Kings
134	2020/21	Mumbai Indians
194	2019	Mumbai Indians
254	2018	Chennai Super Kings
314	2017	Mumbai Indians
373	2016	Sunrisers Hyderabad
433	2015	Mumbai Indians
492	2014	Kolkata Knight Riders
552	2013	Mumbai Indians
628	2012	Kolkata Knight Riders
702	2011	Chennai Super Kings
775	2009/10	Chennai Super Kings
835	2009	Deccan Chargers
892	2007/08	Rajasthan Royals

```
In [53]: # how many super over finishes have occured
ipl.head(2)
```

Out[53]:

	ID	City	Date	Season	MatchNumber	Team1	Team2	Venue	TossWinner	TossDecision	SuperOver	WinningTeam	WonB
0	1312200	Ahmedabad	2022-05-29	2022	Final	Rajasthan Royals	Gujarat Titans	Narendra Modi Stadium, Ahmedabad	Rajasthan Royals	bat	N	Gujarat Titans	Wicke
1	1312199	Ahmedabad	2022-05-27	2022	Qualifier 2	Royal Challengers Bangalore	Rajasthan Royals	Narendra Modi Stadium, Ahmedabad	Rajasthan Royals	field	N	Rajasthan Royals	Wicke

In [54]:

ipl[ipl['SuperOver']=='Y'].shape[0]

Out[54]:

14

In [55]:

how many matches has csk won in kolkata

ipl.sample(2)

Out[55]:

	ID	City	Date	Season	MatchNumber	Team1	Team2	Venue	TossWinner	TossDecision	SuperOver	WinningTeam	Wc
364	1082599	Pune	2017-04-11	2017	9	Rising Pune Supergiant	Delhi Daredevils	Maharashtra Cricket Association Stadium	Rising Pune Supergiant	field	N	Delhi Daredevils	
376	981013	Bangalore	2016-05-24	2016	Qualifier 1	Gujarat Lions	Royal Challengers Bangalore	M Chinnaswamy Stadium	Royal Challengers Bangalore	field	N	Royal Challengers Bangalore	Wi

In [56]:

ipl[(ipl['City'] == 'Kolkata') & (ipl['WinningTeam'] == 'Chennai Super Kings')].shape[0]

Out[56]:

5

In [57]:

toss winner is match winner in percentage

ipl.sample(2)

Out[57]:

	ID	City	Date	Season	MatchNumber	Team1	Team2	Venue	TossWinner	TossDecision	SuperOver	WinningTeam	WonBy	Ma
168	1216529	Abu Dhabi	2020-10-11	2020/21	27	Delhi Capitals	Mumbai Indians	Sheikh Zayed Stadium	Delhi Capitals	bat	N	Mumbai Indians	Wickets	
127	1254064	Mumbai	2021-04-15	2021	7	Delhi Capitals	Rajasthan Royals	Wankhede Stadium, Mumbai	Rajasthan Royals	field	N	Rajasthan Royals	Wickets	

In [58]:

(ipl[(ipl['TossWinner']== ipl['WinningTeam'])]).shape[0]/ipl.shape[0])*100

Out[58]:

51.473684210526315

In [59]:

movies with rating higher than 8 and votes>10000

movies.sample(2)

Out[59]:

	title_x	imdb_id	poster_path	wiki_link	title_y	original_title	is_adult	year_of
24	Junglee (2019 film)	tt7463730	https://upload.wikimedia.org/wikipedia/en/e/e2...	https://en.wikipedia.org/wiki/Junglee_(2019_film)	Junglee	Junglee	0	
390	Hey Bro	tt4512230	https://upload.wikimedia.org/wikipedia/en/thum...	https://en.wikipedia.org/wiki/Hey_Bro	Hey Bro	Hey Bro	0	

In [60]:

movies[(movies['imdb_rating'] > 8) & (movies['imdb_votes'] > 10000)].shape[0]

Out[60]:

43

```
In [61]: # Action movies with rating higher than 7.5
#mask1=movies['genres'].str.split('|').apply(lambda x: 'Action' in x)
mask1=movies['genres'].str.contains('Action')
mask2=movies['imdb_rating']>7.5
movies[mask1 & mask2]
```

Out[61]:

	title_x	imdb_id	poster_path	wiki_link	title_y	original_title
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: Th Surgical Strik
41	Family of Thakurganj	tt8897986	https://upload.wikimedia.org/wikipedia/en/9/99...	https://en.wikipedia.org/wiki/Family_of_Thakur...	Family of Thakurganj	Family of Thakurga
84	Mukkabaaz	tt7180544	https://upload.wikimedia.org/wikipedia/en/thum...	https://en.wikipedia.org/wiki/Mukkabaaz	The Brawler	Mukkabaaz
106	Raazi	tt7098658	https://upload.wikimedia.org/wikipedia/en/thum...	https://en.wikipedia.org/wiki/Raazi	Raazi	Raa

Adding new cols

```
In [62]: movies['country']='India'
movies.sample(2)
```

Out[62]:

	title_x	imdb_id	poster_path	wiki_link	title_y	original_title	is_adult	year_of
915	Wanted (2009 film)	tt1084972	https://upload.wikimedia.org/wikipedia/en/thum...	https://en.wikipedia.org/wiki/Wanted_(2009_film)	Wanted	Wanted	0	
241	Shaadi Mein Zaroor Aana	tt7469726	https://upload.wikimedia.org/wikipedia/en/thum...	https://en.wikipedia.org/wiki/Shaaadi_Mein_Zaro...	Shaadi Mein Zaroor Aana	Shaadi Mein Zaroor Aana	0	

```
In [63]: movies.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1629 entries, 0 to 1628
Data columns (total 19 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
18  country                1629 non-null  object
dtypes: float64(1), int64(3), object(15)
memory usage: 241.9+ KB
```

```
In [138]: # From Existing ones
movies['actors'].str.split('|').apply(lambda x:x[0])
```

```
In [139]: # From Existing ones
movies['lead actor']= movies['actors'].str.split('|').apply(lambda x:x[0])
movies
```

Important DataFrame Functions

```
In [68]: ipl.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 950 entries, 0 to 949
Data columns (total 20 columns):
#   Column                Non-Null Count  Dtype
---  -
0   ID                    950 non-null   int64
1   City                  899 non-null   object
2   Date                  950 non-null   object
3   Season                950 non-null   object
4   MatchNumber           950 non-null   object
5   Team1                 950 non-null   object
6   Team2                 950 non-null   object
7   Venue                 950 non-null   object
8   TossWinner            950 non-null   object
9   TossDecision          950 non-null   object
10  SuperOver             946 non-null   object
11  WinningTeam           946 non-null   object
12  WonBy                 950 non-null   object
13  Margin                932 non-null   float64
14  method                19 non-null    object
15  Player_of_Match       946 non-null   object
16  Team1Players           950 non-null   object
17  Team2Players           950 non-null   object
18  Umpire1               950 non-null   object
19  Umpire2               950 non-null   object
dtypes: float64(1), int64(1), object(18)
memory usage: 148.6+ KB
```

```
In [69]: ipl['ID']=ipl['ID'].astype('int32')
```

```
In [70]: ipl.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 950 entries, 0 to 949
Data columns (total 20 columns):
#   Column                Non-Null Count  Dtype
---  -
0   ID                    950 non-null   int32
1   City                  899 non-null   object
2   Date                  950 non-null   object
3   Season                950 non-null   object
4   MatchNumber           950 non-null   object
5   Team1                 950 non-null   object
6   Team2                 950 non-null   object
7   Venue                 950 non-null   object
8   TossWinner            950 non-null   object
9   TossDecision          950 non-null   object
10  SuperOver             946 non-null   object
11  WinningTeam           946 non-null   object
12  WonBy                 950 non-null   object
13  Margin                932 non-null   float64
14  method                19 non-null    object
15  Player_of_Match       946 non-null   object
16  Team1Players           950 non-null   object
17  Team2Players           950 non-null   object
18  Umpire1               950 non-null   object
19  Umpire2               950 non-null   object
dtypes: float64(1), int32(1), object(18)
memory usage: 144.9+ KB
```

```
In [71]: ipl['Season'] = ipl['Season'].astype('category')
```


In [72]: ipl.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 950 entries, 0 to 949
Data columns (total 20 columns):
#   Column                Non-Null Count  Dtype
---  -
0   ID                    950 non-null   int32
1   City                 899 non-null   object
2   Date                 950 non-null   object
3   Season              950 non-null   category
4   MatchNumber         950 non-null   object
5   Team1               950 non-null   object
6   Team2               950 non-null   object
7   Venue               950 non-null   object
8   TossWinner          950 non-null   object
9   TossDecision        950 non-null   object
10  SuperOver           946 non-null   object
11  WinningTeam         946 non-null   object
12  WonBy               950 non-null   object
13  Margin              932 non-null   float64
14  method              19 non-null    object
15  Player_of_Match     946 non-null   object
16  Team1Players        950 non-null   object
17  Team2Players        950 non-null   object
18  Umpire1             950 non-null   object
19  Umpire2             950 non-null   object
dtypes: category(1), float64(1), int32(1), object(17)
memory usage: 139.0+ KB
```

```
In [73]: ipl['Team1'] = ipl['Team1'].astype('category')
ipl['Team2'] = ipl['Team2'].astype('category')
```

In [74]: ipl.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 950 entries, 0 to 949
Data columns (total 20 columns):
#   Column                Non-Null Count  Dtype
---  -
0   ID                    950 non-null   int32
1   City                 899 non-null   object
2   Date                 950 non-null   object
3   Season              950 non-null   category
4   MatchNumber         950 non-null   object
5   Team1               950 non-null   category
6   Team2               950 non-null   category
7   Venue               950 non-null   object
8   TossWinner          950 non-null   object
9   TossDecision        950 non-null   object
10  SuperOver           946 non-null   object
11  WinningTeam         946 non-null   object
12  WonBy               950 non-null   object
13  Margin              932 non-null   float64
14  method              19 non-null    object
15  Player_of_Match     946 non-null   object
16  Team1Players        950 non-null   object
17  Team2Players        950 non-null   object
18  Umpire1             950 non-null   object
19  Umpire2             950 non-null   object
dtypes: category(3), float64(1), int32(1), object(15)
memory usage: 127.4+ KB
```

More Important Functions

Value Counts

In [143]: `# value_counts(series and dataframe)`

```
marks = pd.DataFrame([
    [100,80,10],
    [90,70,7],
    [120,100,14],
    [80,70,14],
    [80,70,14]
],columns=['iq','marks','package'])

marks
```

Out[143]:

	iq	marks	package
0	100	80	10
1	90	70	7
2	120	100	14
3	80	70	14
4	80	70	14

In [76]: `marks.value_counts()`

```
Out[76]: iq  marks  package
80    70     14         2
90    70     7         1
100   80    10         1
120  100    14         1
dtype: int64
```

In [77]: `# find which player has won most potm -> in finals and qualifiers`
`ipl.sample(2)`

Out[77]:

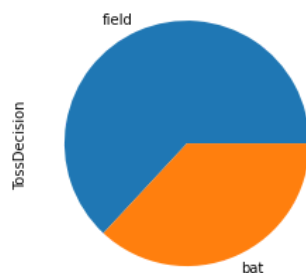
	ID	City	Date	Season	MatchNumber	Team1	Team2	Venue	TossWinner	TossDecision	SuperOver	WinningTeam	WonBy
304	1136570	Kolkata	2018-04-14	2018	10	Kolkata Knight Riders	Sunrisers Hyderabad	Eden Gardens	Sunrisers Hyderabad	field	N	Sunrisers Hyderabad	Wickets
561	598028	Dharamsala	2013-05-16	2013	67	Kings XI Punjab	Delhi Daredevils	Himachal Pradesh Cricket Association Stadium	Delhi Daredevils	field	N	Kings XI Punjab	Runs

```
In [78]: ipl[~ipl['MatchNumber'].str.isdigit()][['Player_of_Match']].value_counts() # To reverse the contet use tilt ~
```

```
Out[78]: KA Pollard          3
         F du Plessis       3
         SK Raina           3
         A Kumble           2
         MK Pandey          2
         YK Pathan          2
         M Vijay            2
         JJ Bumrah          2
         AB de Villiers     2
         SR Watson          2
         HH Pandya          1
         Harbhajan Singh    1
         A Nehra            1
         V Sehwag           1
         UT Yadav           1
         MS Bisla           1
         BJ Hodge           1
         MEK Hussey         1
         MS Dhoni           1
         CH Gayle           1
         MM Patel           1
         DE Bollinger       1
         AC Gilchrist       1
         RG Sharma          1
         DA Warner          1
         MC Henriques       1
         JC Buttler         1
         RM Patidar         1
         DA Miller          1
         VR Iyer            1
         SP Narine          1
         RD Gaikwad         1
         TA Boult           1
         MP Stoinis         1
         KS Williamson      1
         RR Pant            1
         SA Yadav           1
         Rashid Khan        1
         AD Russell         1
         KH Pandya          1
         KV Sharma          1
         NM Coulter-Nile    1
         Washington Sundar  1
         BCJ Cutting        1
         M Ntini            1
         Name: Player_of_Match, dtype: int64
```

```
In [79]: # Toss decision plot
ipl['TossDecision'].value_counts().plot(kind='pie')
```

```
Out[79]: <AxesSubplot:ylabel='TossDecision'>
```



```
In [80]: # No.of matches each team has played
(ipl['Team1'].value_counts() + ipl['Team2'].value_counts()).sort_values(ascending=False)
```

Out[80]: Mumbai Indians 231
Royal Challengers Bangalore 226
Kolkata Knight Riders 223
Chennai Super Kings 208
Rajasthan Royals 192
Kings XI Punjab 190
Delhi Daredevils 161
Sunrisers Hyderabad 152
Deccan Chargers 75
Delhi Capitals 63
Pune Warriors 46
Gujarat Lions 30
Punjab Kings 28
Gujarat Titans 16
Rising Pune Supergiant 16
Lucknow Super Giants 15
Kochi Tuskers Kerala 14
Rising Pune Supergiants 14
dtype: int64

Sort values

```
In [81]: x = pd.Series([12,14,1,56,89])
x
```

Out[81]: 0 12
1 14
2 1
3 56
4 89
dtype: int64

```
In [82]: x.sort_values(ascending=True)
```

Out[82]: 2 1
0 12
1 14
3 56
4 89
dtype: int64

```
In [83]: movies.sample(2)
```

Out[83]:

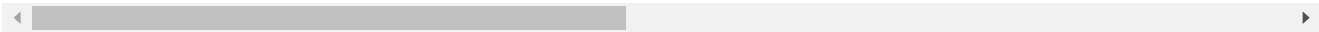
	title_x	imdb_id	poster_path	wiki_link	title_y	original_title	is_adult	year_of_r
107	Hope Aur Hum	tt8324474	https://upload.wikimedia.org/wikipedia/en/thum...	https://en.wikipedia.org/wiki/Hope_Aur_Hum	Hope Aur Hum	Hope Aur Hum	0	
666	Tere Naal Love Ho Gaya	tt2130242	https://upload.wikimedia.org/wikipedia/en/thum...	https://en.wikipedia.org/wiki/Tere_Naal_Love_H...	Tere Naal Love Ho Gaya	Tere Naal Love Ho Gaya	0	

```
In [84]: movies.sort_values('title_x', ascending=False)
```

Out[84]:

	title_x	imdb_id	poster_path	wiki_link	title_y	original_title	is_adult	y
1623	Zubeidaa	tt0255713	https://upload.wikimedia.org/wikipedia/en/thum...	https://en.wikipedia.org/wiki/Zubeidaa	Zubeidaa	Zubeidaa	0	
939	Zor Lagaa Ke...Haiya!	tt1479857	https://upload.wikimedia.org/wikipedia/en/thum...	https://en.wikipedia.org/wiki/Zor_Lagaa_Ke...H...	Zor Lagaa Ke... Haiya!	Zor Lagaa Ke... Haiya!	0	
756	Zokkomon	tt1605790	https://upload.wikimedia.org/wikipedia/en/thum...	https://en.wikipedia.org/wiki/Zokkomon	Zokkomon	Zokkomon	0	
670	Zindagi Tere Naam	tt2164702	https://upload.wikimedia.org/wikipedia/en/thum...	https://en.wikipedia.org/wiki/Zindagi_Tere_Naam	Zindagi Tere Naam	Zindagi Tere Naam	0	
778	Zindagi Na Milegi Dobara	tt1562872	https://upload.wikimedia.org/wikipedia/en/thum...	https://en.wikipedia.org/wiki/Zindagi_Na_Mileg...	Zindagi Na Milegi Dobara	Zindagi Na Milegi Dobara	0	
...
1039	1971 (2007 film)	tt0983990	https://upload.wikimedia.org/wikipedia/en/thum...	https://en.wikipedia.org/wiki/1971_(2007_film)	1971	1971	0	
723	1920: The Evil Returns	tt2222550	https://upload.wikimedia.org/wikipedia/en/e/e7...	https://en.wikipedia.org/wiki/1920:_The_Evil_R...	1920: Evil Returns	1920: Evil Returns	0	
287	1920: London	tt5638500	https://upload.wikimedia.org/wikipedia/en/thum...	https://en.wikipedia.org/wiki/1920_London	1920 London	1920 London	0	
1021	1920 (film)	tt1301698	https://upload.wikimedia.org/wikipedia/en/thum...	https://en.wikipedia.org/wiki/1920_(film)	1920	1920	0	
1498	16 December (film)	tt0313844	https://upload.wikimedia.org/wikipedia/en/thum...	https://en.wikipedia.org/wiki/16_December_(film)	16-Dec	16-Dec	0	

1629 rows × 9 columns



```
In [85]: students = pd.DataFrame(
    {
        'name': ['nitish', 'ankit', 'rupesh', np.nan, 'mrityunjay', np.nan, 'rishabh', np.nan, 'aditya', np.nan],
        'college': ['bit', 'iit', 'vit', np.nan, np.nan, 'vlsi', 'ssit', np.nan, np.nan, 'git'],
        'branch': ['eee', 'it', 'cse', np.nan, 'me', 'ce', 'civ', 'cse', 'bio', np.nan],
        'cgpa': [6.66, 8.25, 6.41, np.nan, 5.6, 9.0, 7.4, 10, 7.4, np.nan],
        'package': [4, 5, 6, np.nan, 6, 7, 8, 9, np.nan, np.nan]
    }
)

students
```

```
Out[85]:
```

	name	college	branch	cgpa	package
0	nitish	bit	eee	6.66	4.0
1	ankit	iit	it	8.25	5.0
2	rupesh	vit	cse	6.41	6.0
3	NaN	NaN	NaN	NaN	NaN
4	mrityunjay	NaN	me	5.60	6.0
5	NaN	vlsi	ce	9.00	7.0
6	rishabh	ssit	civ	7.40	8.0
7	NaN	NaN	cse	10.00	9.0
8	aditya	NaN	bio	7.40	NaN
9	NaN	git	NaN	NaN	NaN

```
In [86]: students.sort_values('name', ascending=False, na_position='first') # inplace=True ( for Permanent Changes)
```

```
Out[86]:
```

	name	college	branch	cgpa	package
3	NaN	NaN	NaN	NaN	NaN
5	NaN	vlsi	ce	9.00	7.0
7	NaN	NaN	cse	10.00	9.0
9	NaN	git	NaN	NaN	NaN
2	rupesh	vit	cse	6.41	6.0
6	rishabh	ssit	civ	7.40	8.0
0	nitish	bit	eee	6.66	4.0
4	mrityunjay	NaN	me	5.60	6.0
1	ankit	iit	it	8.25	5.0
8	aditya	NaN	bio	7.40	NaN

```
In [87]: students
```

```
Out[87]:
```

	name	college	branch	cgpa	package
0	nitish	bit	eee	6.66	4.0
1	ankit	iit	it	8.25	5.0
2	rupesh	vit	cse	6.41	6.0
3	NaN	NaN	NaN	NaN	NaN
4	mrityunjay	NaN	me	5.60	6.0
5	NaN	vlsi	ce	9.00	7.0
6	rishabh	ssit	civ	7.40	8.0
7	NaN	NaN	cse	10.00	9.0
8	aditya	NaN	bio	7.40	NaN
9	NaN	git	NaN	NaN	NaN

```
In [88]: movies.sort_values(['year_of_release', 'title_x'], ascending=[True, False]).head(2)
```

```
Out[88]:
```

	title_x	imdb_id	poster_path	wiki_link	title_y	original_title	is_adult	ye
1623	Zubeidaa	tt0255713	https://upload.wikimedia.org/wikipedia/en/thum...	https://en.wikipedia.org/wiki/Zubeidaa	Zubeidaa	Zubeidaa	0	
1625	Yeh Zindagi Ka Safar	tt0298607	https://upload.wikimedia.org/wikipedia/en/thum...	https://en.wikipedia.org/wiki/Yeh_Zindagi_Ka_S...	Yeh Zindagi Ka Safar	Yeh Zindagi Ka Safar	0	

rank(Series)

```
In [89]: batsman=pd.read_csv("batsman_runs_ipl.csv")
```

```
In [90]: batsman.head(2)
```

```
Out[90]:
```

	batter	batsman_run
0	A Ashish Reddy	280
1	A Badoni	161

```
In [91]: batsman['batsman_run'].rank(ascending=False)
```

```
Out[91]: 0    166.5
1    226.0
2    535.0
3    329.0
4    402.5
...
600  594.0
601  343.0
602  547.5
603   27.0
604  256.0
Name: batsman_run, Length: 605, dtype: float64
```

```
In [92]: batsman['batsman_rank'] = batsman['batsman_run'].rank(ascending=False)
batsman.sort_values('batsman_rank')
```

```
Out[92]:
```

	batter	batsman_run	batsman_rank
569	V Kohli	6634	1.0
462	S Dhawan	6244	2.0
130	DA Warner	5883	3.0
430	RG Sharma	5881	4.0
493	SK Raina	5536	5.0
...
512	SS Cottrell	0	594.0
466	S Kaushik	0	594.0
203	IC Pandey	0	594.0
467	S Ladda	0	594.0
468	S Lamichhane	0	594.0

605 rows × 3 columns

sort_index (Series and dataframe)

```
In [93]: marks = {  
    'maths':67,  
    'english':57,  
    'science':89,  
    'hindi':100  
}  
  
marks_series = pd.Series(marks)  
marks_series
```

```
Out[93]: maths      67  
english    57  
science    89  
hindi     100  
dtype: int64
```

```
In [94]: marks_series.sort_index()
```

```
Out[94]: english    57  
hindi     100  
maths      67  
science    89  
dtype: int64
```

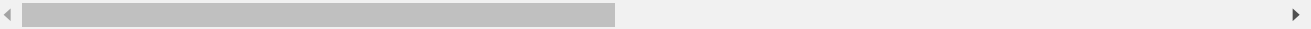

In [95]:

movies.sort_index(ascending=False)

Out[95]:

	title_x	imdb_id	poster_path	wiki_link	title_y	original_title	is_adult
1628	Humsafar	tt2403201	https://upload.wikimedia.org/wikipedia/en/thum...	https://en.wikipedia.org/wiki/Humsafar	Humsafar	Humsafar	0
1627	Daaka	tt10833860	https://upload.wikimedia.org/wikipedia/en/thum...	https://en.wikipedia.org/wiki/Daaka	Daaka	Daaka	0
1626	Sabse Bada Sukh	tt0069204	NaN	https://en.wikipedia.org/wiki/Sabse_Bada_Sukh	Sabse Bada Sukh	Sabse Bada Sukh	0
1625	Yeh Zindagi Ka Safar	tt0298607	https://upload.wikimedia.org/wikipedia/en/thum...	https://en.wikipedia.org/wiki/Yeh_Zindagi_Ka_S...	Yeh Zindagi Ka Safar	Yeh Zindagi Ka Safar	0
1624	Tera Mera Saath Rahen	tt0301250	https://upload.wikimedia.org/wikipedia/en/2/2b...	https://en.wikipedia.org/wiki/Tera_Mera_Saath_...	Tera Mera Saath Rahen	Tera Mera Saath Rahen	0
...
4	Evening Shadows	tt6028796	NaN	https://en.wikipedia.org/wiki/Evening_Shadows	Evening Shadows	Evening Shadows	0
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	0
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	0
1	Battalion 609	tt9472208	NaN	https://en.wikipedia.org/wiki/Battalion_609	Battalion 609	Battalion 609	0
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	0

1629 rows × 19 columns



In [96]:

set_index(dataframe) -> inplace
batsman.set_index('batter',inplace=True)

In [99]:

how to delete columns

File "C:\Users\user\AppData\Local\Temp\ipykernel_17496\2400687195.py", line 1
how to delete columns
^
SyntaxError: invalid syntax

In [140]:

This code drops two columns from the 'batsman' dataframe: 'Level_0' and 'index'.
The 'inplace=True' parameter ensures that the original dataframe is modified instead of creating a new one.

batsman.drop(['index'], axis=1, inplace=True)

```
In [101]: # reset_index(series + dataframe) -> drop parameter
batsman.reset_index(inplace=True)
```

```
In [102]: batsman
```

```
Out[102]:
```

	batter	batsman_run	batsman_rank
0	A Ashish Reddy	280	166.5
1	A Badoni	161	226.0
2	A Chandila	4	535.0
3	A Chopra	53	329.0
4	A Choudhary	25	402.5
...
600	Yash Dayal	0	594.0
601	Yashpal Singh	47	343.0
602	Younis Khan	3	547.5
603	Yuvraj Singh	2754	27.0
604	Z Khan	117	256.0

605 rows × 3 columns

```
In [103]: # how to replace existing index without loosing
batsman.reset_index().set_index('batsman_rank')
```

```
Out[103]:
```

	index	batter	batsman_run
batsman_rank			
166.5	0	A Ashish Reddy	280
226.0	1	A Badoni	161
535.0	2	A Chandila	4
329.0	3	A Chopra	53
402.5	4	A Choudhary	25
...
594.0	600	Yash Dayal	0
343.0	601	Yashpal Singh	47
547.5	602	Younis Khan	3
27.0	603	Yuvraj Singh	2754
256.0	604	Z Khan	117

605 rows × 3 columns

```
In [104]: batsman
```

```
Out[104]:
```

	batter	batsman_run	batsman_rank
0	A Ashish Reddy	280	166.5
1	A Badoni	161	226.0
2	A Chandila	4	535.0
3	A Chopra	53	329.0
4	A Choudhary	25	402.5
...
600	Yash Dayal	0	594.0
601	Yashpal Singh	47	343.0
602	Younis Khan	3	547.5
603	Yuvraj Singh	2754	27.0
604	Z Khan	117	256.0

605 rows × 3 columns

```
In [105]: # series to dataframe using reset_index  
marks_series.reset_index()
```

```
Out[105]:
```

	index	0
0	maths	67
1	english	57
2	science	89
3	hindi	100

```
In [106]: type(marks_series.reset_index())
```

```
Out[106]: pandas.core.frame.DataFrame
```

rename(dataframe) -> index

```
In [107]: movies.set_index('title_x',inplace=True)
```

```
In [108]: movies
```

Out[108]:

	imdb_id	poster_path	wiki_link	title_y	original_title	is_adult	year_
title_x							
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	0	
Battalion 609	tt9472208	NaN	https://en.wikipedia.org/wiki/Battalion_609	Battalion 609	Battalion 609	0	
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	0	
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	0	
Evening Shadows	tt6028796	NaN	https://en.wikipedia.org/wiki/Evening_Shadows	Evening Shadows	Evening Shadows	0	
...
Tera Mera Saath Rahen	tt0301250	https://upload.wikimedia.org/wikipedia/en/2/2b...	https://en.wikipedia.org/wiki/Tera_Mera_Saath_...	Tera Mera Saath Rahen	Tera Mera Saath Rahen	0	
Yeh Zindagi Ka Safar	tt0298607	https://upload.wikimedia.org/wikipedia/en/thum...	https://en.wikipedia.org/wiki/Yeh_Zindagi_Ka_S...	Yeh Zindagi Ka Safar	Yeh Zindagi Ka Safar	0	
Sabse Bada Sukh	tt0069204	NaN	https://en.wikipedia.org/wiki/Sabse_Bada_Sukh	Sabse Bada Sukh	Sabse Bada Sukh	0	
Daaka	tt10833860	https://upload.wikimedia.org/wikipedia/en/thum...	https://en.wikipedia.org/wiki/Daaka	Daaka	Daaka	0	
Humsafar	tt2403201	https://upload.wikimedia.org/wikipedia/en/thum...	https://en.wikipedia.org/wiki/Humsafar	Humsafar	Humsafar	0	

1629 rows × 8 columns

```
In [109]: #Rename the columns
movies.rename(columns={'imdb_id' : 'imdb', 'poster_path':'link'},inplace=True)
```

```
In [110]: movies
```

Out[110]:

	imdb	link	wiki_link	title_y	original_title	is_adult	year_
title_x							
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	0	
Battalion 609	tt9472208	NaN	https://en.wikipedia.org/wiki/Battalion_609	Battalion 609	Battalion 609	0	
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	0	
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	0	
Evening Shadows	tt6028796	NaN	https://en.wikipedia.org/wiki/Evening_Shadows	Evening Shadows	Evening Shadows	0	
...
Tera Mera Saath Rahen	tt0301250	https://upload.wikimedia.org/wikipedia/en/2/2b...	https://en.wikipedia.org/wiki/Tera_Mera_Saath_...	Tera Mera Saath Rahen	Tera Mera Saath Rahen	0	
Yeh Zindagi Ka Safar	tt0298607	https://upload.wikimedia.org/wikipedia/en/thum...	https://en.wikipedia.org/wiki/Yeh_Zindagi_Ka_S...	Yeh Zindagi Ka Safar	Yeh Zindagi Ka Safar	0	
Sabse Bada Sukh	tt0069204	NaN	https://en.wikipedia.org/wiki/Sabse_Bada_Sukh	Sabse Bada Sukh	Sabse Bada Sukh	0	
Daaka	tt10833860	https://upload.wikimedia.org/wikipedia/en/thum...	https://en.wikipedia.org/wiki/Daaka	Daaka	Daaka	0	
Humsafar	tt2403201	https://upload.wikimedia.org/wikipedia/en/thum...	https://en.wikipedia.org/wiki/Humsafar	Humsafar	Humsafar	0	

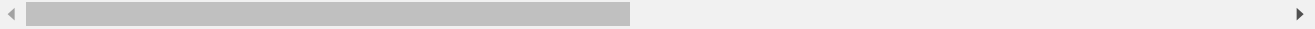
1629 rows × 8 columns

```
In [111]: # Rename the index
movies.rename(index={'Uri: The Surgical Strike':'uri', 'Humsafar':'Hum'})
```

Out[111]:

	imdb		link	wiki_link	title_y	original_title	is_adult	year_
title_x								
uri	tt8291224	https://upload.wikimedia.org/wikipedia/en/thum...	https://en.wikipedia.org/wiki/Uri:_The_Surgica...		Uri: The Surgical Strike	Uri: The Surgical Strike	0	
Battalion 609	tt9472208		NaN	https://en.wikipedia.org/wiki/Battalion_609	Battalion 609	Battalion 609	0	
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	0	
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	0	
Evening Shadows	tt6028796		NaN	https://en.wikipedia.org/wiki/Evening_Shadows	Evening Shadows	Evening Shadows	0	
...
Tera Mera Saath Rahen	tt0301250	https://upload.wikimedia.org/wikipedia/en/2/2b...	https://en.wikipedia.org/wiki/Tera_Mera_Saath_...		Tera Mera Saath Rahen	Tera Mera Saath Rahen	0	
Yeh Zindagi Ka Safar	tt0298607	https://upload.wikimedia.org/wikipedia/en/thum...	https://en.wikipedia.org/wiki/Yeh_Zindagi_Ka_S...		Yeh Zindagi Ka Safar	Yeh Zindagi Ka Safar	0	
Sabse Bada Sukh	tt0069204		NaN	https://en.wikipedia.org/wiki/Sabse_Bada_Sukh	Sabse Bada Sukh	Sabse Bada Sukh	0	
Daaka	tt10833860	https://upload.wikimedia.org/wikipedia/en/thum...	https://en.wikipedia.org/wiki/Daaka		Daaka	Daaka	0	
Hum	tt2403201	https://upload.wikimedia.org/wikipedia/en/thum...	https://en.wikipedia.org/wiki/Humsafar		Humsafar	Humsafar	0	

1629 rows × 18 columns



unique

```
In [112]: # unique(series)
temp = pd.Series([1,1,2,2,3,3,4,4,5,5,np.nan,np.nan])
print(temp)
temp.unique()

0    1.0
1    1.0
2    2.0
3    2.0
4    3.0
5    3.0
6    4.0
7    4.0
8    5.0
9    5.0
10   NaN
11   NaN
dtype: float64
```

```
Out[112]: array([ 1.,  2.,  3.,  4.,  5., nan])
```

```
In [113]: ipl['Season'].unique()
```

```
Out[113]: ['2022', '2021', '2020/21', '2019', '2018', ..., '2012', '2011', '2009/10', '2009', '2007/08']
Length: 15
Categories (15, object): ['2007/08', '2009', '2009/10', '2011', ..., '2019', '2020/21', '2021', '2022']
```

- `nunique` : returns the number of unique elements in a pandas Series or DataFrame. It doesn't count the missing values (NaNs) by default. If you want to count the missing values, you can set the argument "dropna" to False.
- `unique`: returns the unique elements in a pandas Series or DataFrame. It counts the missing values (NaNs) by default. If you don't want to count the missing values, you can use the "dropna" argument and set it to True.

```
In [114]: len(ipl['Season'].unique())
```

```
Out[114]: 15
```

```
In [115]: # nunique(series + dataframe) -> does not count nan -> dropna parameter
ipl['Season'].nunique()
```

```
Out[115]: 15
```

isnull(series + dataframe)

```
In [116]: students
```

```
Out[116]:
```

	name	college	branch	cgpa	package
0	nitish	bit	eee	6.66	4.0
1	ankit	iit	it	8.25	5.0
2	rupesh	vit	cse	6.41	6.0
3	NaN	NaN	NaN	NaN	NaN
4	mrityunjay	NaN	me	5.60	6.0
5	NaN	vlsi	ce	9.00	7.0
6	rishabh	ssit	civ	7.40	8.0
7	NaN	NaN	cse	10.00	9.0
8	aditya	NaN	bio	7.40	NaN
9	NaN	git	NaN	NaN	NaN

```
In [117]: students['name'].isnull()
```

```
Out[117]: 0    False
          1    False
          2    False
          3     True
          4    False
          5     True
          6    False
          7     True
          8    False
          9     True
          Name: name, dtype: bool
```

```
In [118]: # notnull(series + dataframe)
          students['name'].notnull()
```

```
Out[118]: 0     True
          1     True
          2     True
          3    False
          4     True
          5    False
          6     True
          7    False
          8     True
          9    False
          Name: name, dtype: bool
```

```
In [119]: students['name'][students['name'].notnull()]
```

```
Out[119]: 0      nitish
          1      ankit
          2      rupesh
          4  mrityunjay
          6      rishabh
          8      aditya
          Name: name, dtype: object
```

```
In [120]: # hasnans(series)
          students['college'].hasnans
```

```
Out[120]: True
```

```
In [121]: students.isnull()
```

```
Out[121]:
```

	name	college	branch	cgpa	package
0	False	False	False	False	False
1	False	False	False	False	False
2	False	False	False	False	False
3	True	True	True	True	True
4	False	True	False	False	False
5	True	False	False	False	False
6	False	False	False	False	False
7	True	True	False	False	False
8	False	True	False	False	True
9	True	False	True	True	True


```
In [122]: students.notnull()
```

```
Out[122]:
```

	name	college	branch	cgpa	package
0	True	True	True	True	True
1	True	True	True	True	True
2	True	True	True	True	True
3	False	False	False	False	False
4	True	False	True	True	True
5	False	True	True	True	True
6	True	True	True	True	True
7	False	False	True	True	True
8	True	False	True	True	False
9	False	True	False	False	False

dropna -> for Missing Values

```
In [123]: # dropna(series + dataframe) -> how parameter -> works like or
students['name'].dropna()
```

```
Out[123]:
```

```
0      nitish
1      ankit
2      rupesh
4  mrityunjay
6      rishabh
8      aditya
Name: name, dtype: object
```

how : {'any', 'all'}, default 'any'

Determine if row or column is removed from DataFrame, when we have at least one NA or all NA.

- * 'any' : If any NA values are present, drop that row or column.
- * 'all' : If all values are NA, drop that row or column.

```
In [124]: students.dropna(how='any')
```

```
Out[124]:
```

	name	college	branch	cgpa	package
0	nitish	bit	eee	6.66	4.0
1	ankit	iit	it	8.25	5.0
2	rupesh	vit	cse	6.41	6.0
6	rishabh	ssit	civ	7.40	8.0

```
In [125]: students.dropna(how='all')
```

```
Out[125]:
```

	name	college	branch	cgpa	package
0	nitish	bit	eee	6.66	4.0
1	ankit	iit	it	8.25	5.0
2	rupesh	vit	cse	6.41	6.0
4	mrityunjay	NaN	me	5.60	6.0
5	NaN	vlsi	ce	9.00	7.0
6	rishabh	ssit	civ	7.40	8.0
7	NaN	NaN	cse	10.00	9.0
8	aditya	NaN	bio	7.40	NaN
9	NaN	git	NaN	NaN	NaN

subset : array-like, optional

Labels along other axis to consider, e.g. if you are dropping rows
these would be a list of columns to include.

```
In [126]: students.dropna(subset=['name'])
```

Out[126]:

	name	college	branch	cgpa	package
0	nitish	bit	eee	6.66	4.0
1	ankit	iit	it	8.25	5.0
2	rupesh	vit	cse	6.41	6.0
4	mrityunjay	NaN	me	5.60	6.0
6	rishabh	ssit	civ	7.40	8.0
8	aditya	NaN	bio	7.40	NaN

```
In [127]: students.dropna(subset=['name', 'college'])
```

Out[127]:

	name	college	branch	cgpa	package
0	nitish	bit	eee	6.66	4.0
1	ankit	iit	it	8.25	5.0
2	rupesh	vit	cse	6.41	6.0
6	rishabh	ssit	civ	7.40	8.0

fill na (series + dataframe)

```
In [128]: students
```

Out[128]:

	name	college	branch	cgpa	package
0	nitish	bit	eee	6.66	4.0
1	ankit	iit	it	8.25	5.0
2	rupesh	vit	cse	6.41	6.0
3	NaN	NaN	NaN	NaN	NaN
4	mrityunjay	NaN	me	5.60	6.0
5	NaN	vlsi	ce	9.00	7.0
6	rishabh	ssit	civ	7.40	8.0
7	NaN	NaN	cse	10.00	9.0
8	aditya	NaN	bio	7.40	NaN
9	NaN	git	NaN	NaN	NaN

```
In [129]: students['name'].fillna('unknown')
```

Out[129]:

```
0    nitish
1    ankit
2    rupesh
3    unknown
4    mrityunjay
5    unknown
6    rishabh
7    unknown
8    aditya
9    unknown
Name: name, dtype: object
```

```
In [130]: students.fillna('0')
```

```
Out[130]:
```

	name	college	branch	cgpa	package
0	nitish	bit	eee	6.66	4.0
1	ankit	iit	it	8.25	5.0
2	rupesh	vit	cse	6.41	6.0
3	0	0	0	0	0
4	mrityunjay	0	me	5.6	6.0
5	0	vlsi	ce	9.0	7.0
6	rishabh	ssit	civ	7.4	8.0
7	0	0	cse	10.0	9.0
8	aditya	0	bio	7.4	0
9	0	git	0	0	0

```
In [131]: students['package'].fillna(students['package'].mean())
```

```
Out[131]: 0    4.000000
1    5.000000
2    6.000000
3    6.428571
4    6.000000
5    7.000000
6    8.000000
7    9.000000
8    6.428571
9    6.428571
Name: package, dtype: float64
```

```
In [132]: students['name'].fillna(method='ffill') #forward fill method
```

```
Out[132]: 0    nitish
1    ankit
2    rupesh
3    rupesh
4    mrityunjay
5    mrityunjay
6    rishabh
7    rishabh
8    aditya
9    aditya
Name: name, dtype: object
```

```
In [133]: students['name'].fillna(method='bfill') # backward fill method
```

```
Out[133]: 0    nitish
1    ankit
2    rupesh
3    mrityunjay
4    mrityunjay
5    rishabh
6    rishabh
7    aditya
8    aditya
9    NaN
Name: name, dtype: object
```

drop_duplicates

```
In [134]: # drop_duplicates(series + dataframe) -> works like and -> duplicated()
```

```
In [135]: marks
```

```
Out[135]: {'maths': 67, 'english': 57, 'science': 89, 'hindi': 100}
```

```
In [144]: marks
```

Out[144]:

	iq	marks	package
0	100	80	10
1	90	70	7
2	120	100	14
3	80	70	14
4	80	70	14

```
In [145]: marks.drop_duplicates()
```

Out[145]:

	iq	marks	package
0	100	80	10
1	90	70	7
2	120	100	14
3	80	70	14

```
In [146]: marks.drop_duplicates(keep='last')
```

Out[146]:

	iq	marks	package
0	100	80	10
1	90	70	7
2	120	100	14
4	80	70	14

```
In [148]: # find the last match played by virat kohli in Delhi
ipl.sample(2)
```

Out[148]:

	ID	City	Date	Season	MatchNumber	Team1	Team2	Venue	TossWinner	TossDecision	SuperOver	WinningTeam	WonBy	M
6	1304114	Mumbai	2022-05-20	2022	68	Chennai Super Kings	Rajasthan Royals	Brabourne Stadium, Mumbai	Chennai Super Kings	bat	N	Rajasthan Royals	Wickets	
276	1136598	Indore	2018-05-06	2018	38	Rajasthan Royals	Kings XI Punjab	Holkar Cricket Stadium	Kings XI Punjab	field	N	Kings XI Punjab	Wickets	

```
In [149]: ipl['all Players'] = ipl['Team1Players'] + ipl['Team2Players']
ipl.sample(2)
```

Out[149]:

nue	TossWinner	TossDecision	...	WinningTeam	WonBy	Margin	method	Player_of_Match	Team1Players	Team2Players	Umpire1	Umpire2
MA ram ium, auk	Rajasthan Royals	bat	...	Chennai Super Kings	Wickets	8.0	NaN	MEK Hussey	['MEK Hussey', 'M Vijay', 'SK Raina', 'JA Mork...	['SR Watson', 'R Dravid', 'AL Menaria', 'J Bot...	SS Hazare	RB Tiffin
iede ium, nbai	Delhi Capitals	field	...	Royal Challengers Bangalore	Runs	16.0	NaN	KD Karthik	['F du Plessis', 'Anuj Rawat', 'V Kohli', 'GJ ...	['PP Shaw', 'DA Warner', 'MR Marsh', 'RR Pant'...	Chirra Ravikanthreddy	J Madanagopal

```
In [154]: def did_kohli_play(players_list):
          return 'V Kohli' in players_list
```

```
In [160]: ipl['did_kohli_play'] = ipl['all Players'].apply(did_kohli_play)
```

```
In [165]: ipl.sample(2)
```

Out[165]:

	ID	City	Date	Season	MatchNumber	Team1	Team2	Venue	TossWinner	TossDecision	...	Margin	method	Player_of_Match
940	335991	Chandigarh	2008-04-25	2007/08	10	Kings XI Punjab	Mumbai Indians	Punjab Cricket Association Stadium, Mohali	Mumbai Indians	field	...	66.0	NaN	KC Sangakka
826	419114	Delhi	2010-03-17	2009/10	9	Delhi Daredevils	Mumbai Indians	Feroz Shah Kotla	Delhi Daredevils	field	...	98.0	NaN	SR Tendulkar

2 rows × 23 columns

```
In [166]: ipl[(ipl['City'] == 'Delhi') & (ipl['did_kohli_play'] == True)]
```

Out[166]:

	ID	City	Date	Season	MatchNumber	Team1	Team2	Venue	TossWinner	TossDecision	...	Margin	method	Player_of_Match
208	1178421	Delhi	2019-04-28	2019	46	Delhi Capitals	Royal Challengers Bangalore	Arun Jaitley Stadium	Delhi Capitals	bat	...	16.0	NaN	S Dhawan
269	1136605	Delhi	2018-05-12	2018	45	Delhi Daredevils	Royal Challengers Bangalore	Arun Jaitley Stadium	Royal Challengers Bangalore	field	...	5.0	NaN	AB de Villiers
318	1082646	Delhi	2017-05-14	2017	56	Delhi Daredevils	Royal Challengers Bangalore	Feroz Shah Kotla	Royal Challengers Bangalore	bat	...	10.0	NaN	HV Patel
467	829757	Delhi	2015-04-26	2015	26	Delhi Daredevils	Royal Challengers Bangalore	Feroz Shah Kotla	Royal Challengers Bangalore	field	...	10.0	NaN	VR Aaron
571	598054	Delhi	2013-05-10	2013	57	Delhi Daredevils	Royal Challengers Bangalore	Feroz Shah Kotla	Delhi Daredevils	field	...	4.0	NaN	JD Unadkat
638	548372	Delhi	2012-05-17	2012	67	Delhi Daredevils	Royal Challengers Bangalore	Feroz Shah Kotla	Delhi Daredevils	field	...	21.0	NaN	CH Gayle
746	501227	Delhi	2011-04-26	2011	30	Delhi Daredevils	Royal Challengers Bangalore	Feroz Shah Kotla	Royal Challengers Bangalore	field	...	3.0	NaN	V Kohli
801	419140	Delhi	2010-04-04	2009/10	35	Delhi Daredevils	Royal Challengers Bangalore	Feroz Shah Kotla	Delhi Daredevils	bat	...	37.0	NaN	PD Collingwood
933	335998	Delhi	2008-04-30	2007/08	17	Delhi Daredevils	Royal Challengers Bangalore	Feroz Shah Kotla	Royal Challengers Bangalore	field	...	10.0	NaN	GD McGrath

9 rows × 23 columns

```
In [168]: ipl[( ipl['City'] == 'Delhi') & (ipl['did_kohli_play'] == True)].drop_duplicates(subset=['City','did_kohli_play'],keep='first')
```

Out[168]:

	ID	City	Date	Season	MatchNumber	Team1	Team2	Venue	TossWinner	TossDecision	...	Margin	method	Player_of_Match	Test
208	1178421	Delhi	2019-04-28	2019	46	Delhi Capitals	Royal Challengers Bangalore	Arun Jaitley Stadium	Delhi Capitals	bat	...	16.0	NaN	S Dhawan	[F D]

1 rows × 23 columns

drop(series + dataframe)

```
In [169]: # Series
temp = pd.Series([10,2,3,16,45,78,10])
temp
```

Out[169]:

0	10
1	2
2	3
3	16
4	45
5	78
6	10

dtype: int64

```
In [170]: temp.drop(index=[0,6])
```

Out[170]:

1	2
2	3
3	16
4	45
5	78

dtype: int64

```
In [171]: students
```

Out[171]:

	name	college	branch	cgpa	package
0	nitish	bit	eee	6.66	4.0
1	ankit	iit	it	8.25	5.0
2	rupesh	vit	cse	6.41	6.0
3	NaN	NaN	NaN	NaN	NaN
4	mrityunjay	NaN	me	5.60	6.0
5	NaN	vlsi	ce	9.00	7.0
6	rishabh	ssit	civ	7.40	8.0
7	NaN	NaN	cse	10.00	9.0
8	aditya	NaN	bio	7.40	NaN
9	NaN	git	NaN	NaN	NaN

```
In [172]: students.drop(columns=['branch', 'cgpa']) # To delete Columns
```

```
Out[172]:
```

	name	college	package
0	nitish	bit	4.0
1	ankit	iit	5.0
2	rupesh	vit	6.0
3	NaN	NaN	NaN
4	mrityunjay	NaN	6.0
5	NaN	vlsi	7.0
6	rishabh	ssit	8.0
7	NaN	NaN	9.0
8	aditya	NaN	NaN
9	NaN	git	NaN

```
In [173]: students.drop(index=[0,8]) # to delete rows
```

```
Out[173]:
```

	name	college	branch	cgpa	package
1	ankit	iit	it	8.25	5.0
2	rupesh	vit	cse	6.41	6.0
3	NaN	NaN	NaN	NaN	NaN
4	mrityunjay	NaN	me	5.60	6.0
5	NaN	vlsi	ce	9.00	7.0
6	rishabh	ssit	civ	7.40	8.0
7	NaN	NaN	cse	10.00	9.0
9	NaN	git	NaN	NaN	NaN

```
In [174]: students.set_index('name')
```

```
Out[174]:
```

	college	branch	cgpa	package
name				
nitish	bit	eee	6.66	4.0
ankit	iit	it	8.25	5.0
rupesh	vit	cse	6.41	6.0
NaN	NaN	NaN	NaN	NaN
mrityunjay	NaN	me	5.60	6.0
NaN	vlsi	ce	9.00	7.0
rishabh	ssit	civ	7.40	8.0
NaN	NaN	cse	10.00	9.0
aditya	NaN	bio	7.40	NaN
NaN	git	NaN	NaN	NaN

```
In [175]: students.set_index('name').drop(index=['nitish', 'aditya']) # delete by names
```

```
Out[175]:
```

	college	branch	cgpa	package
name				
ankit	iit	it	8.25	5.0
rupesh	vit	cse	6.41	6.0
NaN	NaN	NaN	NaN	NaN
mrityunjay	NaN	me	5.60	6.0
NaN	vlsi	ce	9.00	7.0
rishabh	ssit	civ	7.40	8.0
NaN	NaN	cse	10.00	9.0
NaN	git	NaN	NaN	NaN

apply(series + dataframe)

```
In [176]: # series
temp = pd.Series([10,20,30,40,50])

temp
```

```
Out[176]: 0    10
          1    20
          2    30
          3    40
          4    50
          dtype: int64
```

```
In [177]: def sigmoid(value):
          return 1/(1+np.exp(-value))
```

```
In [178]: temp.apply(sigmoid)
```

```
Out[178]: 0    1.000045
          1    1.000000
          2    1.000000
          3    1.000000
          4    1.000000
          dtype: float64
```

```
In [179]: # On data Frame
points = pd.DataFrame(
    {
        '1st point':[(3,4),(-6,5),(0,0),(-10,1),(4,5)],
        '2nd point':[(-3,4),(0,0),(2,2),(10,10),(1,1)]
    }
)

points
```

```
Out[179]:
```

	1st point	2nd point
0	(3, 4)	(-3, 4)
1	(-6, 5)	(0, 0)
2	(0, 0)	(2, 2)
3	(-10, 1)	(10, 10)
4	(4, 5)	(1, 1)

```
In [180]: def euclidean(row):
          point_A = row['1st point']
          point_B = row['2nd point']
          return ((point_A[0] - point_B[0])**2 + (point_A[1] - point_B[1])**2)**0.5
```

```
In [182]: points.apply(euclidean,axis=1) # mention axis=1 because its row wise
```

```
Out[182]: 0    6.000000
          1    7.810250
          2    2.828427
          3    21.931712
          4    5.000000
          dtype: float64
```

```
In [184]: points['distance'] = points.apply(euclidean,axis=1)
points
```

```
Out[184]:
```

	1st point	2nd point	distance
0	(3, 4)	(-3, 4)	6.000000
1	(-6, 5)	(0, 0)	7.810250
2	(0, 0)	(2, 2)	2.828427
3	(-10, 1)	(10, 10)	21.931712
4	(4, 5)	(1, 1)	5.000000

Groupby (Applies on Categorical data)


```
In [187]: movies = pd.read_csv("imdb-top-1000.csv")
```

```
In [189]: movies.head(1)
```

Out[189]:

	Series_Title	Released_Year	Runtime	Genre	IMDB_Rating	Director	Star1	No_of_Votes	Gross	Metascore
0	The Shawshank Redemption	1994	142	Drama	9.3	Frank Darabont	Tim Robbins	2343110	28341469.0	80.0

```
In [190]: movies.groupby('Genre')
```

Out[190]: <pandas.core.groupby.generic.DataFrameGroupBy object at 0x000001EFE1E1F670>

```
In [195]: genres = movies.groupby('Genre')
```

```
In [196]: # Applying builtin aggregation fuctions on groupby objects
genres.sum()
```

Out[196]:

	Runtime	IMDB_Rating	No_of_Votes	Gross	Metascore
Genre					
Action	22196	1367.3	72282412	3.263226e+10	10499.0
Adventure	9656	571.5	22576163	9.496922e+09	5020.0
Animation	8166	650.3	21978630	1.463147e+10	6082.0
Biography	11970	698.6	24006844	8.276358e+09	6023.0
Comedy	17380	1224.7	27620327	1.566387e+10	9840.0
Crime	13524	857.8	33533615	8.452632e+09	6706.0
Drama	36049	2299.7	61367304	3.540997e+10	19208.0
Family	215	15.6	551221	4.391106e+08	158.0
Fantasy	170	16.0	146222	7.827267e+08	0.0
Film-Noir	312	23.9	367215	1.259105e+08	287.0
Horror	1123	87.0	3742556	1.034649e+09	880.0
Mystery	1429	95.7	4203004	1.256417e+09	633.0
Thriller	108	7.8	27733	1.755074e+07	81.0
Western	593	33.4	1289665	5.822151e+07	313.0

```
In [197]: genres.mean()
```

Out[197]:

	Runtime	IMDB_Rating	No_of_Votes	Gross	Metascore
Genre					
Action	129.046512	7.949419	420246.581395	1.897224e+08	73.419580
Adventure	134.111111	7.937500	313557.819444	1.319017e+08	78.437500
Animation	99.585366	7.930488	268032.073171	1.784326e+08	81.093333
Biography	136.022727	7.938636	272805.045455	9.404952e+07	76.240506
Comedy	112.129032	7.901290	178195.658065	1.010572e+08	78.720000
Crime	126.392523	8.016822	313398.271028	7.899656e+07	77.080460
Drama	124.737024	7.957439	212343.612457	1.225259e+08	79.701245
Family	107.500000	7.800000	275610.500000	2.195553e+08	79.000000
Fantasy	85.000000	8.000000	73111.000000	3.913633e+08	NaN
Film-Noir	104.000000	7.966667	122405.000000	4.197018e+07	95.666667
Horror	102.090909	7.909091	340232.363636	9.405902e+07	80.000000
Mystery	119.083333	7.975000	350250.333333	1.047014e+08	79.125000
Thriller	108.000000	7.800000	27733.000000	1.755074e+07	81.000000
Western	148.250000	8.350000	322416.250000	1.455538e+07	78.250000

In [198]: `generes.min()`

Out[198]:

	Series_Title	Released_Year	Runtime	IMDB_Rating	Director	Star1	No_of_Votes	Gross	Metascore
Genre									
Action	300	1924	45	7.6	Abhishek Chaubey	Aamir Khan	25312	3296.0	33.0
Adventure	2001: A Space Odyssey	1925	88	7.6	Akira Kurosawa	Aamir Khan	29999	61001.0	41.0
Animation	Akira	1940	71	7.6	Adam Elliot	Adrian Molina	25229	128985.0	61.0
Biography	12 Years a Slave	1928	93	7.6	Adam McKay	Adrien Brody	27254	21877.0	48.0
Comedy	(500) Days of Summer	1921	68	7.6	Alejandro G. Iñárritu	Aamir Khan	26337	1305.0	45.0
Crime	12 Angry Men	1931	80	7.6	Akira Kurosawa	Ajay Devgn	27712	6013.0	47.0
Drama	1917	1925	64	7.6	Aamir Khan	Abhay Deol	25088	3600.0	28.0
Family	E.T. the Extra-Terrestrial	1971	100	7.8	Mel Stuart	Gene Wilder	178731	4000000.0	67.0
Fantasy	Das Cabinet des Dr. Caligari	1920	76	7.9	F.W. Murnau	Max Schreck	57428	337574718.0	NaN
Film-Noir	Shadow of a Doubt	1941	100	7.8	Alfred Hitchcock	Humphrey Bogart	59556	449191.0	94.0
Horror	Alien	1933	71	7.6	Alejandro Amenábar	Anthony Perkins	27007	89029.0	46.0
Mystery	Dark City	1938	96	7.6	Alex Proyas	Bernard-Pierre Donnadiou	33982	1035953.0	52.0
Thriller	Wait Until Dark	1967	108	7.8	Terence Young	Audrey Hepburn	27733	17550741.0	81.0
Western	Il buono, il brutto, il cattivo	1965	132	7.8	Clint Eastwood	Clint Eastwood	65659	5321508.0	69.0

In [206]: `# find the top 3 genres by total earning`
`movies.groupby('Genre').sum()['Gross'].sort_values(ascending=False).head(3)`

Out[206]: Genre
Drama 3.540997e+10
Action 3.263226e+10
Comedy 1.566387e+10
Name: Gross, dtype: float64

In [212]: `#Second Approach`
`movies.groupby('Genre')['Gross'].sum().sort_values()`

Out[212]: Genre
Thriller 1.755074e+07
Western 5.822151e+07
Film-Noir 1.259105e+08
Family 4.391106e+08
Fantasy 7.827267e+08
Horror 1.034649e+09
Mystery 1.256417e+09
Biography 8.276358e+09
Crime 8.452632e+09
Adventure 9.496922e+09
Animation 1.463147e+10
Comedy 1.566387e+10
Action 3.263226e+10
Drama 3.540997e+10
Name: Gross, dtype: float64

In [214]: `movies.groupby('Genre')['Gross'].sum().sort_values(ascending=False).head(3)`

Out[214]: Genre
Drama 3.540997e+10
Action 3.263226e+10
Comedy 1.566387e+10
Name: Gross, dtype: float64

In [217]: `# find the genre with highest avg IMDB rating`
`movies.groupby('Genre')['IMDB_Rating'].mean().sort_values(ascending=False).head(1)`

Out[217]: Genre
Western 8.35
Name: IMDB_Rating, dtype: float64