07-06-2023

DATA STRUCTURES IN PANDAS

In [1]:
```
#Pandas deals with following data structures(inbuild classes)
# Series : 1D labelled array,homogeneous data,size immutable
# DataFrame : 2D labelled,heterogenous data,size is totally mutable
```

In [3]:
```
import pandas as pd
```

Series

In [4]:
```
# Syntax - pandas.Series(data,index,dtype,copy)
```

Creating empty Series

In [5]:
```
s=pd.Series()
s
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_5752\521516865.py:1: FutureWarning: The d
efault dtype for empty Series will be 'object' instead of 'float64' in a future versi
on. Specify a dtype explicitly to silence this warning.
  s=pd.Series()
```

Out[5]:
```
Series([], dtype: float64)
```

Creating a Series with list

In [6]:
```
s=pd.Series(data=[1,2,3,4],index=["a","b","c","d"])
s
```

Out[6]:
```
a    1
b    2
c    3
d    4
dtype: int64
```

In [7]:
```
s=pd.Series([1,2,3,4])
s
```

Out[7]:
```
0    1
1    2
2    3
3    4
dtype: int64
```

Create a Series with dictionary

In [8]:
```
data={"a":1.0,"b":3.0,"c":4.0}
s=pd.Series(data)
```

```
s
```

Out[8]:
```
a    1.0
b    3.0
c    4.0
dtype: float64
```

Creating Series with Numpy array

In [9]:
```
import numpy as np
```

In [10]:
```
i=np.array([1,2,3,4])
d=np.array(["a","b","c","d"])

s=pd.Series(d,i)
s
```

Out[10]:
```
1    a
2    b
3    c
4    d
dtype: object
```

Creating Series with scalar values

In [11]:
```
s=pd.Series(5,index=[0,1,2,3,4])
s
```

Out[11]:
```
0    5
1    5
2    5
3    5
4    5
dtype: int64
```

DataFrame

DataFrame : 2D labelled,heterogenous data,size is totally mutable

Creating empty DataFrame

In [12]:
```
df=pd.DataFrame()
df
```

Out[12]: —

Creating DataFrame with list

In [13]:
```
data=[1,23,3,4,5]
pd.DataFrame(data)
```

Out[13]:

	0
0	1
1	23
2	3
3	4
4	5

Create DataFrame with list with multiple column

In [14]:
```
data=[["i","ii","iii"],["a","b","c"],[1,2,3]]
df=pd.DataFrame(data,columns=["A","B","C"])
df
```

Out[14]:

	A	B	C
0	i	ii	iii
1	a	b	c
2	1	2	3

Creating DataFrame with python Dictionary

In [15]:
```
data={"Name":["Ankita","Suraj","Aditya","Rutuja"],"Age":[23,26,18,23]}
df=pd.DataFrame(data)
df
```

Out[15]:

	Name	Age
0	Ankita	23
1	Suraj	26
2	Aditya	18
3	Rutuja	23

In [16]:
```
data={"Name":["Ankita","Suraj","Aditya","Rutuja"],"Age":[23,26,18,23]}
df=pd.DataFrame(data,index=["Rank 1","Rank 2","Rank 3","Rank 4"])
df
```

Out[16]:

	Name	Age
Rank 1	Ankita	23
Rank 2	Suraj	26
Rank 3	Aditya	18
Rank 4	Rutuja	23

Creating DataFrame with numpy array

In [17]:
```
df=pd.DataFrame(np.arange(1,5).reshape(2,2),columns=["A","B"])
df
```

Out[17]:

	A	B
0	1	2
1	3	4

Creating DataFrame with Series

In [18]:
```
d={"one":pd.Series([1,2,3],index=["a","b","c"]),
   "two":pd.Series([1,2,3,4],index=["a","b","c","d"])}
df=pd.DataFrame(d)
df
```

Out[18]:

	one	two
a	1.0	1
b	2.0	2
c	3.0	3
d	NaN	4

Create a DataFrame with random numpy array

In [19]:
```
df=pd.DataFrame(np.random.randn(10,4),columns=["a","b","c","d"])
df
```

Out[19]:

	a	b	c	d
0	0.842031	1.084465	-0.719838	-0.452025
1	-0.087250	-1.275749	-2.467454	-0.088052
2	-0.079520	-0.795601	0.241646	1.341177
3	-1.068067	0.134672	-0.295775	-2.074663
4	0.464736	-1.162657	-0.943563	-0.155489
5	-1.356216	0.936541	-1.641290	1.272685
6	-0.273451	0.126014	0.053592	0.704034
7	0.881173	-1.832177	-0.517593	-0.927520
8	-1.590217	0.227833	1.165662	-0.716627
9	-0.429041	-0.730115	0.273888	-1.600837

In [20]:
```
# low=10,high=15,size=(10,4)
df=pd.DataFrame(np.random.randint(low=10,high=15,size=(10,4)),columns=["A","B","C","D"
```

```
df
```

Out[20]:

	A	B	C	D
0	13	14	12	11
1	13	12	14	11
2	11	10	11	14
3	14	11	13	10
4	12	13	12	14
5	10	10	14	14
6	12	11	10	12
7	14	14	13	14
8	10	10	13	10
9	14	11	13	14

In [21]:
```
df.head() #..... bydefault  head function shows first 5 rows in dataframe
```

Out[21]:

	A	B	C	D
0	13	14	12	11
1	13	12	14	11
2	11	10	11	14
3	14	11	13	10
4	12	13	12	14

In [22]:
```
df.tail()   #.........bydefault tail function shows last 5 rows in dataframe
```

Out[22]:

	A	B	C	D
5	10	10	14	14
6	12	11	10	12
7	14	14	13	14
8	10	10	13	10
9	14	11	13	14

In [27]:
```
df[5].tail()        experimental
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
File C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\indexes\base.py:3621, in
Index.get_loc(self, key, method, tolerance)
   3620 try:
-> 3621     return self._engine.get_loc(casted_key)
   3622 except KeyError as err:

File C:\ProgramData\Anaconda3\lib\site-packages\pandas\_libs\index.pyx:136, in panda
s._libs.index.IndexEngine.get_loc()

File C:\ProgramData\Anaconda3\lib\site-packages\pandas\_libs\index.pyx:163, in panda
s._libs.index.IndexEngine.get_loc()

File pandas\_libs\hashtable_class_helper.pxi:5198, in pandas._libs.hashtable.PyObject
HashTable.get_item()

File pandas\_libs\hashtable_class_helper.pxi:5206, in pandas._libs.hashtable.PyObject
HashTable.get_item()

KeyError: 5

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

KeyError                                  Traceback (most recent call last)
Input In [27], in <cell line: 1>()
----> 1 df[5].tail()

File C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\frame.py:3505, in DataFra
me.__getitem__(self, key)
   3503 if self.columns.nlevels > 1:
   3504     return self._getitem_multilevel(key)
-> 3505 indexer = self.columns.get_loc(key)
   3506 if is_integer(indexer):
   3507     indexer = [indexer]

File C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\indexes\base.py:3623, in
Index.get_loc(self, key, method, tolerance)
   3621     return self._engine.get_loc(casted_key)
   3622 except KeyError as err:
-> 3623     raise KeyError(key) from err
   3624 except TypeError:
   3625     # If we have a listlike key, _check_indexing_error will raise
   3626     #  InvalidIndexError. Otherwise we fall through and re-raise
   3627     #  the TypeError.
   3628     self._check_indexing_error(key)

KeyError: 5
```

In [23]:
```
df.sum()
```

Out[23]:
```
A    123
B    116
C    125
D    124
dtype: int64
```

In [24]:
```
df["A"].sum()
```

Out[24]:
```
123
```

In [25]:
```
df[["A","B"]].sum()
```

Out[25]:
```
A    123
B    116
dtype: int64
```

Indexing in Pandas

In [1]:
```
#Indexing functions =
#1) .loc()  .......labelled indexing
#2) .iloc()   ............integer indexing
#3) .ix()    ........both lable and integer based
```

.loc

In [2]:
```
#.loc()= it is purely labelled based indexing
```

In [3]:
```
import pandas as pd
```

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

In [8]:
```
df=pd.DataFrame(np.random.rand(8,4),index=["a","b","c","d","e","f","g","h"],columns=['
df
```

Out[8]:

	A	B	C	D
a	0.470173	0.320634	0.284495	0.254931
b	0.689851	0.216890	0.991034	0.563780
c	0.392606	0.727068	0.466700	0.143518
d	0.508231	0.110414	0.065007	0.156941
e	0.363536	0.020934	0.900171	0.333307
f	0.312195	0.780530	0.379021	0.916197
g	0.448642	0.919209	0.989431	0.364916
h	0.668993	0.130461	0.191284	0.455318

In [9]:
```
#dataframe=rows and columns
df.loc["b":]
```

Out[9]:

	A	B	C	D
b	0.689851	0.216890	0.991034	0.563780
c	0.392606	0.727068	0.466700	0.143518
d	0.508231	0.110414	0.065007	0.156941
e	0.363536	0.020934	0.900171	0.333307
f	0.312195	0.780530	0.379021	0.916197
g	0.448642	0.919209	0.989431	0.364916
h	0.668993	0.130461	0.191284	0.455318

In [10]:
```
df.loc["c":"f"]
```

Out[10]:

	A	B	C	D
c	0.392606	0.727068	0.466700	0.143518
d	0.508231	0.110414	0.065007	0.156941
e	0.363536	0.020934	0.900171	0.333307
f	0.312195	0.780530	0.379021	0.916197

In [11]:
```
df.loc["a":"h":2]
```

Out[11]:

	A	B	C	D
a	0.470173	0.320634	0.284495	0.254931
c	0.392606	0.727068	0.466700	0.143518
e	0.363536	0.020934	0.900171	0.333307
g	0.448642	0.919209	0.989431	0.364916

In [12]:
```
df.loc[::2]       #...........same like upper example
```

Out[12]:

	A	B	C	D
a	0.470173	0.320634	0.284495	0.254931
c	0.392606	0.727068	0.466700	0.143518
e	0.363536	0.020934	0.900171	0.333307
g	0.448642	0.919209	0.989431	0.364916

In [14]:
```
df.loc[:,"A"]
```

Out[14]:
```
a    0.470173
b    0.689851
c    0.392606
d    0.508231
e    0.363536
f    0.312195
g    0.448642
h    0.668993
Name: A, dtype: float64
```

In [15]:
```
df.loc[:,"B"]
```

Out[15]:
```
a    0.320634
b    0.216890
c    0.727068
d    0.110414
e    0.020934
f    0.780530
g    0.919209
h    0.130461
Name: B, dtype: float64
```

In [16]:
```
df.loc[:,"A"::2]
```

Out[16]:

	A	C
a	0.470173	0.284495
b	0.689851	0.991034
c	0.392606	0.466700
d	0.508231	0.065007
e	0.363536	0.900171
f	0.312195	0.379021
g	0.448642	0.989431
h	0.668993	0.191284

In [17]:
```
df.loc[::2,::2]
```

Out[17]:

	A	C
a	0.470173	0.284495
c	0.392606	0.466700
e	0.363536	0.900171
g	0.448642	0.989431

.iloc

In [18]:
```
#iloc.....Purely integer based
```

In [20]:
```
df.iloc[1:4,1:]
```

Out[20]:

	B	C	D
b	0.216890	0.991034	0.563780
c	0.727068	0.466700	0.143518
d	0.110414	0.065007	0.156941

In [22]:
```
df.iloc[:,:2]
```

Out[22]:

	A	B
a	0.470173	0.320634
b	0.689851	0.216890
c	0.392606	0.727068
d	0.508231	0.110414
e	0.363536	0.020934
f	0.312195	0.780530
g	0.448642	0.919209
h	0.668993	0.130461

In [23]:
```
df.iloc[:,:-2]
```

Out[23]:

	A	B
a	0.470173	0.320634
b	0.689851	0.216890
c	0.392606	0.727068
d	0.508231	0.110414
e	0.363536	0.020934
f	0.312195	0.780530
g	0.448642	0.919209
h	0.668993	0.130461

In [25]:
```
df.iloc[:-2,:]
```

Out[25]:

	A	B	C	D
a	0.470173	0.320634	0.284495	0.254931
b	0.689851	0.216890	0.991034	0.563780
c	0.392606	0.727068	0.466700	0.143518
d	0.508231	0.110414	0.065007	0.156941
e	0.363536	0.020934	0.900171	0.333307
f	0.312195	0.780530	0.379021	0.916197

In [26]:
```
# Display first 2 rows and columns
```

In [29]:
```
df.iloc[:2,:2]
```

Out[29]:

	A	B
a	0.470173	0.320634
b	0.689851	0.216890

In [30]:
```
# Display columns and rows with 2 steps
```

In [31]:
```
df.iloc[::2,::2]
```

Out[31]:

	A	C
a	0.470173	0.284495
c	0.392606	0.466700
e	0.363536	0.900171
g	0.448642	0.989431

In [32]:
```
#first row and first column
```

In [33]:
```
df.iloc[:1,:1]
```

Out[33]:

	A
a	0.470173

In [36]:
```
#last one values
```

In [35]:
```
df.iloc[-1,-1]
```

Out[35]:
```
0.45531784722621993
```

In [37]:
```
df.iloc[4:6,:]
```

Out[37]:

	A	B	C	D
e	0.363536	0.020934	0.900171	0.333307
f	0.312195	0.780530	0.379021	0.916197

Group by clause

In [38]:
```
ipl_data={"Team":["Riders","Riders","Devils","Devils","Kings","Kings","Kings","Royals"
         "Rank":[1,2,2,3,3,4,1,3,4,2],
          "Year":[2012,2014,2017,2019,2020,2018,2016,2019,2014,2017],
         "Points":[870,789,863,741,676,777,767,456,765,887]}
ipl_data
```

Out[38]:
```
{'Team': ['Riders',
  'Riders',
  'Devils',
  'Devils',
  'Kings',
  'Kings',
  'Kings',
  'Royals',
  'Riders',
  'Riders'],
 'Rank': [1, 2, 2, 3, 3, 4, 1, 3, 4, 2],
 'Year': [2012, 2014, 2017, 2019, 2020, 2018, 2016, 2019, 2014, 2017],
 'Points': [870, 789, 863, 741, 676, 777, 767, 456, 765, 887]}
```

In [40]:
```
df=pd.DataFrame(ipl_data)
df
```

Out[40]:

	Team	Rank	Year	Points
0	Riders	1	2012	870
1	Riders	2	2014	789
2	Devils	2	2017	863
3	Devils	3	2019	741
4	Kings	3	2020	676
5	Kings	4	2018	777
6	Kings	1	2016	767
7	Royals	3	2019	456
8	Riders	4	2014	765
9	Riders	2	2017	887

In [41]:
```
tg=df.groupby(["Team"])
```

In [43]:
```
for i in tg:
    print(i[0])  #.........i of zero... its indicates individual teams
    print(i[1])  #...........i of one... its indicates groupby with team
```

```
Devils
     Team  Rank  Year  Points
2  Devils     2  2017     863
3  Devils     3  2019     741
Kings
    Team  Rank  Year  Points
4  Kings     3  2020     676
5  Kings     4  2018     777
6  Kings     1  2016     767
Riders
     Team  Rank  Year  Points
0  Riders     1  2012     870
1  Riders     2  2014     789
8  Riders     4  2014     765
9  Riders     2  2017     887
Royals
     Team  Rank  Year  Points
7  Royals     3  2019     456
```

In [44]:
```
df.groupby(["Team"]).count()
```

Out[44]:

Team	Rank	Year	Points
Devils	2	2	2
Kings	3	3	3
Riders	4	4	4
Royals	1	1	1

In [46]:
```
df.groupby(["Team"])["Points"].sum()
```

Out[46]:
```
Team
Devils    1604
Kings     2220
Riders    3311
Royals     456
Name: Points, dtype: int64
```

In [47]:
```
df.groupby(["Team"])["Points"].max()
```

Out[47]:
```
Team
Devils    863
Kings     777
Riders    887
Royals    456
Name: Points, dtype: int64
```

In [48]:
```
df.groupby(["Team"])["Points"].min()
```

Out[48]:
```
Team
Devils    741
Kings     676
Riders    765
Royals    456
Name: Points, dtype: int64
```

In [49]:
```
df.groupby(["Team"])["Points"].mean()
```

Out[49]:
```
Team
Devils    802.00
Kings     740.00
Riders    827.75
Royals    456.00
Name: Points, dtype: float64
```

In [50]:
```
gb=df.groupby(["Team","Year"])
gb
```

Out[50]:
```
<pandas.core.groupby.generic.DataFrameGroupBy object at 0x00000119247606A0>
```

In [51]:
```
for i in gb:    #.................not imp
    print(i)
```

```
(('Devils', 2017),      Team  Rank  Year  Points
2  Devils     2  2017     863)
(('Devils', 2019),      Team  Rank  Year  Points
3  Devils     3  2019     741)
(('Kings', 2016),     Team  Rank  Year  Points
6  Kings     1  2016     767)
(('Kings', 2018),     Team  Rank  Year  Points
5  Kings     4  2018     777)
(('Kings', 2020),     Team  Rank  Year  Points
4  Kings     3  2020     676)
(('Riders', 2012),      Team  Rank  Year  Points
0  Riders     1  2012     870)
(('Riders', 2014),      Team  Rank  Year  Points
1  Riders     2  2014     789
8  Riders     4  2014     765)
(('Riders', 2017),      Team  Rank  Year  Points
9  Riders     2  2017     887)
(('Royals', 2019),      Team  Rank  Year  Points
7  Royals     3  2019     456)
```

In [53]:
```
xy=df.groupby(["Year"])
```

In [54]:
```
for i in xy:
    print(i[0])
    print(i[1])
```

```
2012
     Team  Rank  Year  Points
0  Riders     1  2012     870
2014
     Team  Rank  Year  Points
1  Riders     2  2014     789
8  Riders     4  2014     765
2016
    Team  Rank  Year  Points
6  Kings     1  2016     767
2017
     Team  Rank  Year  Points
2  Devils     2  2017     863
9  Riders     2  2017     887
2018
    Team  Rank  Year  Points
5  Kings     4  2018     777
2019
     Team  Rank  Year  Points
3  Devils     3  2019     741
7  Royals     3  2019     456
2020
    Team  Rank  Year  Points
4  Kings     3  2020     676
```

In [55]:
```
xy.get_group(2017)
```

Out[55]:

	Team	Rank	Year	Points
2	Devils	2	2017	863
9	Riders	2	2017	887

In [56]:
```
xy.get_group(2017)["Points"].min()
```

Out[56]:
```
863
```

In [57]:
```
xy.get_group(2019)["Points"].min()
```

Out[57]:
```
456
```

In [58]:
```
xy.get_group(2019)["Points"].max()
```

Out[58]:
```
741
```

Example

In [59]:
```
import random
```

In [64]:
```
def play_game():
    print("Welcome to Rock,paper,scissor....!")
    print("Choose one: 1:Rock,2:Paper,3:Scissor")
    choice=["rock","paper","scossor"]
    player_choice=int(input("Enter your choice(1-3)"))
```

```
    if player_choice<0 or player_choice>3:
        print("Invalid choice. Please try again")
        return
    computer_choice=random.randint(0,2)


    print("You Chose:",choice[player_choice])
    print("Computer Chose:",choice[computer_choice])

    if player_choice==computer_choice:
        print("Its a tia!")
    elif(player_choice==0 and computer_choice==2)or\
        (player_choice==1 and computer_choice==0)or\
        (player_choice==2 and computer_choice==1):
        print("Congratulations!!! You Win")
    else:
        print("Sorry!!! You Loss")

    play_again=input("Do you want to play again{y/n}?")
    if play_again.lower()=="y":
        play_game()
    else:
        print("Thank you for playing")
play_game()
```

```
Welcome to Rock,paper,scissor....!
Choose one: 1:Rock,2:Paper,3:Scissor
Enter your choice(1-3)1
You Chose: paper
Computer Chose: paper
Its a tia!
Do you want to play again{y/n}?y
Welcome to Rock,paper,scissor....!
Choose one: 1:Rock,2:Paper,3:Scissor
Enter your choice(1-3)2
You Chose: scossor
Computer Chose: scossor
Its a tia!
Do you want to play again{y/n}?y
Welcome to Rock,paper,scissor....!
Choose one: 1:Rock,2:Paper,3:Scissor
Enter your choice(1-3)1
You Chose: paper
Computer Chose: scossor
Sorry!!! You Loss
Do you want to play again{y/n}?n
Thank you for playing
```

How to deal with missing data

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

In [3]:
```
dt={'Name':['Tom','James','Ricky','Messi','Steve',np.nan,'Jack'],
    'Age':pd.Series([25,26,np.nan,23,np.nan,29,23]),
    'Rateing':pd.Series([4.23,3.24,3.98,2.56,np.nan,4.6,np.nan])}
dt
```

Out[3]:
```
{'Name': ['Tom', 'James', 'Ricky', 'Messi', 'Steve', nan, 'Jack'],
 'Age': 0    25.0
 1    26.0
 2     NaN
 3    23.0
 4     NaN
 5    29.0
 6    23.0
 dtype: float64,
 'Rateing': 0    4.23
 1    3.24
 2    3.98
 3    2.56
 4     NaN
 5    4.60
 6     NaN
 dtype: float64}
```

In [8]:
```
df=pd.DataFrame(dt)
df
```

Out[8]:

	Name	Age	Rateing
0	Tom	25.0	4.23
1	James	26.0	3.24
2	Ricky	NaN	3.98
3	Messi	23.0	2.56
4	Steve	NaN	NaN
5	NaN	29.0	4.60
6	Jack	23.0	NaN

In [9]:
```
df.isnull() #..........to find null values     null=true
```

Out[9]:

	Name	Age	Rateing
0	False	False	False
1	False	False	False
2	False	True	False
3	False	False	False
4	False	True	True
5	True	False	False
6	False	False	True

In [10]:
```
df.notnull()  #.....to find not null vlues    not null=true
```

Out[10]:

	Name	Age	Rateing
0	True	True	True
1	True	True	True
2	True	False	True
3	True	True	True
4	True	False	False
5	False	True	True
6	True	True	False

In [12]:
```
df.isnull().sum()   #......how many null values is shows
```

Out[12]:
```
Name       1
Age        2
Rateing    2
dtype: int64
```

In [13]:
```
df.notnull().sum()      #......how many notnull values is show
```

Out[13]:
```
Name       6
Age        5
Rateing    5
dtype: int64
```

In [14]:
```
df.info()     #..for whole data null and not nulll
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7 entries, 0 to 6
Data columns (total 3 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   Name     6 non-null      object 
 1   Age      5 non-null      float64
 2   Rateing  5 non-null      float64
dtypes: float64(2), object(1)
memory usage: 296.0+ bytes
```

Cleaning data and filling missing data

In [15]:
```
# Fill Hi at all null values
```

In [16]:
```
df.fillna("Hi")       #......fillna=fill null values
```

Out[16]:

	Name	Age	Rateing
0	Tom	25.0	4.23
1	James	26.0	3.24
2	Ricky	Hi	3.98
3	Messi	23.0	2.56
4	Steve	Hi	Hi
5	Hi	29.0	4.6
6	Jack	23.0	Hi

In [17]:
```
df["Name"].fillna("Admin")    #...for fill null data in name column
```

Out[17]:
```
0      Tom
1    James
2    Ricky
3    Messi
4    Steve
5    Admin
6     Jack
Name: Name, dtype: object
```

In [19]:
```
df["Name"].fillna("Admin",inplace=True)  #.....for prmenantatly placed value admin
df["Name"]
```

Out[19]:
```
0      Tom
1    James
2    Ricky
3    Messi
4    Steve
5    Admin
6     Jack
Name: Name, dtype: object
```

In [20]:
```
df
```

Out[20]:

	Name	Age	Rateing
0	Tom	25.0	4.23
1	James	26.0	3.24
2	Ricky	NaN	3.98
3	Messi	23.0	2.56
4	Steve	NaN	NaN
5	Admin	29.0	4.60
6	Jack	23.0	NaN

Handling the missing values with drop

In [21]:
```
#   drop =  missing values columns are droped
```

In [22]:
```
df.dropna()
```

Out[22]:

	Name	Age	Rateing
0	Tom	25.0	4.23
1	James	26.0	3.24
3	Messi	23.0	2.56
5	Admin	29.0	4.60

In [24]:
```
df.dropna(axis=1)
```

Out[24]:

	Name
0	Tom
1	James
2	Ricky
3	Messi
4	Steve
5	Admin
6	Jack

In [25]:
```
df
```

Out[25]:

	Name	Age	Rateing
0	Tom	25.0	4.23
1	James	26.0	3.24
2	Ricky	NaN	3.98
3	Messi	23.0	2.56
4	Steve	NaN	NaN
5	Admin	29.0	4.60
6	Jack	23.0	NaN

Replace missing data or generic values

In [27]:
```
df["Age"].mean()
```

Out[27]:
```
25.2
```

In [55]:
```
df.replace(np.nan,df["Age"].mean())
print(df["Name"].replace("Admin","Ankita"))
```

```
0       Tom
1     James
2     Ricky
3     Messi
4     Steve
5    Ankita
6      Jack
Name: Name, dtype: object
```

In [30]:
```
df["Rateing"].mean()
```

Out[30]:
```
3.722
```

In [31]:
```
df.replace(np.nan,df["Rateing"].mean())
```

Out[31]:

	Name	Age	Rateing
0	Tom	25.000	4.230
1	James	26.000	3.240
2	Ricky	3.722	3.980
3	Messi	23.000	2.560
4	Steve	3.722	3.722
5	Admin	29.000	4.600
6	Jack	23.000	3.722

In [34]:
```
#dif betn fillna and replace is fillna use to fill only null values and replace used f
```

Drop function in pandas

In [35]:
```
df.drop(["Age"],axis=1)   # ....to write axis is compulsory
```

Out[35]:

	Name	Rateing
0	Tom	4.23
1	James	3.24
2	Ricky	3.98
3	Messi	2.56
4	Steve	NaN
5	Admin	4.60
6	Jack	NaN

In [39]:
```
df.drop(["Name","Age"],axis=1)
```

Out[39]:

	Rateing
0	4.23
1	3.24
2	3.98
3	2.56
4	NaN
5	4.60
6	NaN

In [41]:
```
df.drop([0,2])      # .....indexing wise drop
```

Out[41]:

	Name	Age	Rateing
1	James	26.0	3.24
3	Messi	23.0	2.56
4	Steve	NaN	NaN
5	Admin	29.0	4.60
6	Jack	23.0	NaN

In [43]:
```
# drop columns and rows

df.drop(index=1,columns="Age")
```

Out[43]:

	Name	Rateing
0	Tom	4.23
2	Ricky	3.98
3	Messi	2.56
4	Steve	NaN
5	Admin	4.60
6	Jack	NaN

Sorting in pandas

In [45]:
```
# Sort by index

df.sort_index()
```

Out[45]:

	Name	Age	Rateing
0	Tom	25.0	4.23
1	James	26.0	3.24
2	Ricky	NaN	3.98
3	Messi	23.0	2.56
4	Steve	NaN	NaN
5	Admin	29.0	4.60
6	Jack	23.0	NaN

In [46]:
```
df.sort_index(ascending=False)
```

Out[46]:

	Name	Age	Rateing
6	Jack	23.0	NaN
5	Admin	29.0	4.60
4	Steve	NaN	NaN
3	Messi	23.0	2.56
2	Ricky	NaN	3.98
1	James	26.0	3.24
0	Tom	25.0	4.23

In [47]:
```
df
```

Out[47]:

	Name	Age	Rateing
0	Tom	25.0	4.23
1	James	26.0	3.24
2	Ricky	NaN	3.98
3	Messi	23.0	2.56
4	Steve	NaN	NaN
5	Admin	29.0	4.60
6	Jack	23.0	NaN

In [48]:
```
df.sort_index(axis=1)
```

Out[48]:

	Age	Name	Rateing
0	25.0	Tom	4.23
1	26.0	James	3.24
2	NaN	Ricky	3.98
3	23.0	Messi	2.56
4	NaN	Steve	NaN
5	29.0	Admin	4.60
6	23.0	Jack	NaN

In [50]:
```
#Sort by values

df.sort_values(by="Age")
```

Out[50]:

	Name	Age	Rateing
3	Messi	23.0	2.56
6	Jack	23.0	NaN
0	Tom	25.0	4.23
1	James	26.0	3.24
5	Admin	29.0	4.60
2	Ricky	NaN	3.98
4	Steve	NaN	NaN

In [53]:
```
df.sort_values(["Age","Rateing"],ascending=False)
```

Out[53]:

	Name	Age	Rateing
5	Admin	29.0	4.60
1	James	26.0	3.24
0	Tom	25.0	4.23
3	Messi	23.0	2.56
6	Jack	23.0	NaN
2	Ricky	NaN	3.98
4	Steve	NaN	NaN

CSV

In [56]:
```
import pandas as pd
```

In [57]:
```
import pandas as np
```

In [59]:
```
# How to read csv file

df=pd.read_csv(r"C:\Users\admin\gplay.csv")
```

In [60]:
```
df
```

Out[60]:

	Unnamed: 0	App	Category	Rating	Reviews	Size	Installs	Type	P
0	0	Photo Editor & Candy Camera & Grid & ScrapBook	ART_AND_DESIGN	4.1	159	19M	10,000+	Free	
1	1	Coloring book moana	ART_AND_DESIGN	3.9	967	14M	500,000+	Free	
2	2	U Launcher Lite – FREE Live Cool Themes, Hide ...	ART_AND_DESIGN	4.7	87510	8.7M	5,000,000+	Free	
3	3	Sketch - Draw & Paint	ART_AND_DESIGN	4.5	215644	25M	50,000,000+	Free	
4	4	Pixel Draw - Number Art Coloring Book	ART_AND_DESIGN	4.3	967	2.8M	100,000+	Free	
...	...	...	...	...	...	...	...	...	
10836	10836	Sya9a Maroc - FR	FAMILY	4.5	38	53M	5,000+	Free	
10837	10837	Fr. Mike Schmitz Audio Teachings	FAMILY	5.0	4	3.6M	100+	Free	
10838	10838	Parkinson Exercices FR	MEDICAL	NaN	3	9.5M	1,000+	Free	
10839	10839	The SCP Foundation DB fr nn5n	BOOKS_AND_REFERENCE	4.5	114	Varies with device	1,000+	Free	
10840	10840	iHoroscope - 2018 Daily Horoscope & Astrology	LIFESTYLE	4.5	398307	19M	10,000,000+	Free	

10841 rows × 10 columns

In [64]:
```
# find stratinf 10 values of rows
df.head(10)
```

Out[64]:

	Unnamed: 0	App	Category	Rating	Reviews	Size	Installs	Type	Price	Conter Ratin
0	0	Photo Editor & Candy Camera & Grid & ScrapBook	ART_AND_DESIGN	4.1	159	19M	10,000+	Free	0	Everyor
1	1	Coloring book moana	ART_AND_DESIGN	3.9	967	14M	500,000+	Free	0	Everyor
2	2	U Launcher Lite – FREE Live Cool Themes, Hide ...	ART_AND_DESIGN	4.7	87510	8.7M	5,000,000+	Free	0	Everyor
3	3	Sketch - Draw & Paint	ART_AND_DESIGN	4.5	215644	25M	50,000,000+	Free	0	Tee
4	4	Pixel Draw - Number Art Coloring Book	ART_AND_DESIGN	4.3	967	2.8M	100,000+	Free	0	Everyor
5	5	Paper flowers instructions	ART_AND_DESIGN	4.4	167	5.6M	50,000+	Free	0	Everyor
6	6	Smoke Effect Photo Maker - Smoke Editor	ART_AND_DESIGN	3.8	178	19M	50,000+	Free	0	Everyor
7	7	Infinite Painter	ART_AND_DESIGN	4.1	36815	29M	1,000,000+	Free	0	Everyor
8	8	Garden Coloring Book	ART_AND_DESIGN	4.4	13791	33M	1,000,000+	Free	0	Everyor
9	9	Kids Paint Free - Drawing Fun	ART_AND_DESIGN	4.7	121	3.1M	10,000+	Free	0	Everyor

In [65]:
```
df.isnull().sum()
```

Out[65]:
```
Unnamed: 0          0
App                 0
Category            0
Rating           1474
Reviews             0
Size                0
Installs            0
Type                1
Price               0
Content Rating      1
dtype: int64
```

In [66]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10841 entries, 0 to 10840
Data columns (total 10 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   Unnamed: 0      10841 non-null  int64  
 1   App             10841 non-null  object 
 2   Category        10841 non-null  object 
 3   Rating          9367 non-null   float64
 4   Reviews         10841 non-null  object 
 5   Size            10841 non-null  object 
 6   Installs        10841 non-null  object 
 7   Type            10840 non-null  object 
 8   Price           10841 non-null  object 
 9   Content Rating  10840 non-null  object 
dtypes: float64(1), int64(1), object(8)
memory usage: 847.1+ KB
```

In [71]:
```
df["Rating"].fillna(df["Rating"].mean(),inplace=True)
```

In [72]:
```
df.isnull().sum()
```

Out[72]:
```
Unnamed: 0        0
App               0
Category          0
Rating            0
Reviews           0
Size              0
Installs          0
Type              1
Price             0
Content Rating    1
dtype: int64
```

In [73]:
```
df.dropna(inplace=True)
```

In [74]:
```
df.isnull().sum()
```

Out[74]:
```
Unnamed: 0        0
App               0
Category          0
Rating            0
Reviews           0
Size              0
Installs          0
Type              0
Price             0
Content Rating    0
dtype: int64
```

In [75]:
```
df["Reviews"]=df["Reviews"].astype(float)
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 10839 entries, 0 to 10840
Data columns (total 10 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   Unnamed: 0      10839 non-null  int64  
 1   App             10839 non-null  object 
 2   Category        10839 non-null  object 
 3   Rating          10839 non-null  float64
 4   Reviews         10839 non-null  float64
 5   Size            10839 non-null  object 
 6   Installs        10839 non-null  object 
 7   Type            10839 non-null  object 
 8   Price           10839 non-null  object 
 9   Content Rating  10839 non-null  object 
dtypes: float64(2), int64(1), object(7)
memory usage: 931.5+ KB
```

In [76]:
```
df
```

Out[76]:

	Unnamed: 0	App	Category	Rating	Reviews	Size	Installs	Type
0	0	Photo Editor & Candy Camera & Grid & ScrapBook	ART_AND_DESIGN	4.100000	159.0	19M	10,000+	Free
1	1	Coloring book moana	ART_AND_DESIGN	3.900000	967.0	14M	500,000+	Free
2	2	U Launcher Lite – FREE Live Cool Themes, Hide ...	ART_AND_DESIGN	4.700000	87510.0	8.7M	5,000,000+	Free
3	3	Sketch - Draw & Paint	ART_AND_DESIGN	4.500000	215644.0	25M	50,000,000+	Free
4	4	Pixel Draw - Number Art Coloring Book	ART_AND_DESIGN	4.300000	967.0	2.8M	100,000+	Free
...	...	...	...	...	...	...	...	...
10836	10836	Sya9a Maroc - FR	FAMILY	4.500000	38.0	53M	5,000+	Free
10837	10837	Fr. Mike Schmitz Audio Teachings	FAMILY	5.000000	4.0	3.6M	100+	Free
10838	10838	Parkinson Exercices FR	MEDICAL	4.193338	3.0	9.5M	1,000+	Free
10839	10839	The SCP Foundation DB fr nn5n	BOOKS_AND_REFERENCE	4.500000	114.0	Varies with device	1,000+	Free
10840	10840	iHoroscope - 2018 Daily Horoscope & Astrology	LIFESTYLE	4.500000	398307.0	19M	10,000,000+	Free

10839 rows × 10 columns

In [ ]: