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

In [2]:
```
pd.__version__
```

Out[2]:
```
'1.4.3'
```

Series

In [3]:
```
list_1 = [1,2,-3,6.5,'data values']
print(list_1)
```

```
[1, 2, -3, 6.5, 'data values']
```

In [4]:
```
series_1 = pd.Series(list_1)
print(series_1)
```

```
0              1
1              2
2             -3
3            6.5
4    data values
dtype: object
```

In [5]:
```
type(series_1)
```

Out[5]:
```
pandas.core.series.Series
```

In [6]:
```
series_2 = pd.Series([1,2,3,4])
print(series_2)
```

```
0    1
1    2
2    3
3    4
dtype: int64
```

In [7]:
```
empty_l = pd.Series([])
print(empty_l)
```

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

```
C:\Users\prasad jadhav\AppData\Local\Temp\ipykernel_12320\308653681.py:1: Futu
reWarning: The default dtype for empty Series will be 'object' instead of 'flo
at64' in a future version. Specify a dtype explicitly to silence this warning.
  empty_l = pd.Series([])
```

In [8]:
```
series_3 = pd.Series([1,2,3,4],index = ['a','b','c','d'])
print(series_3)
```

```
a    1
b    2
c    3
d    4
dtype: int64
```

In [9]:
```
series_3 = pd.Series([1,2,3,4],index = ['a','b','c','d'],dtype = float)
print(series_3)
```

```
a    1.0
b    2.0
c    3.0
d    4.0
dtype: float64
```

In [10]:
```
series_3 = pd.Series([1,2,3,4],index = ['a','b','c','d'],dtype = float,name =
print(series_3)
```

```
a    1.0
b    2.0
c    3.0
d    4.0
Name: data values, dtype: float64
```

In [11]:
```
scalar_1 = pd.Series(0.5)
print(scalar_1)
```

```
0    0.5
dtype: float64
```

In [12]:
```
scalar_1 = pd.Series(0.5,index = [1,2,3])
print(scalar_1)
```

```
1    0.5
2    0.5
3    0.5
dtype: float64
```

In [13]:
```
dict_1 = pd.Series({'a':1,'b':2})
print(dict_1)
```

```
a    1
b    2
dtype: int64
```

In [14]:
```
series_4 = pd.Series([1,2,3,4,5])
print(series_4)
```

```
0    1
1    2
2    3
3    4
4    5
dtype: int64
```

In [15]:
```
series_4[0]
```

Out[15]:
```
1
```

In [16]:
```
series_4[0:3]
```

Out[16]:
```
0    1
1    2
2    3
dtype: int64
```

In [17]:
```
max(series_4)
```

Out[17]:
```
5
```

In [18]:
```
min(series_4)
```

Out[18]:
```
1
```

In [19]:
```
series_4[series_4 > 3]
```

Out[19]:
```
3    4
4    5
dtype: int64
```

In [20]:
```
series_5 = pd.Series([1,2,3,4,5])
```

In [21]:
```
series_4 + series_5
```

Out[21]:
```
0     2
1     4
2     6
3     8
4    10
dtype: int64
```

In [22]:
```
series_6 = pd.Series([1,2,3])
series_6
```

Out[22]:
```
0    1
1    2
2    3
dtype: int64
```

In [23]:
```
# NaN Missing Values Handling
series_5 + series_6
```

Out[23]:
```
0    2.0
1    4.0
2    6.0
3    NaN
4    NaN
dtype: float64
```

DataFrame

In [24]:
```
import pandas as pd
```

In [25]:
```
emt_df = pd.DataFrame()
print(emt_df)
```

```
Empty DataFrame
Columns: []
Index: []
```

In [26]:
```
lst = ['a','b','c']
print(lst)
```

```
['a', 'b', 'c']
```

In [27]:
```
df_1 = pd.DataFrame(lst)
print(df_1)
```

```
   0
0  a
1  b
2  c
```

In [28]:
```
df_1
```

Out[28]:

	0
0	a
1	b
2	c

In [29]:
```
lst_of_lst = [[1,2,3],[2,3,4],[4,5,6]]
print(lst_of_lst)
```

```
[[1, 2, 3], [2, 3, 4], [4, 5, 6]]
```

In [30]:
```
df_2 = pd.DataFrame(lst_of_lst)
print(df_2)
```

```
   0  1  2
0  1  2  3
1  2  3  4
2  4  5  6
```

In [31]:
```
df_2
```

Out[31]:

	0	1	2
0	1	2	3
1	2	3	4
2	4	5	6

In [32]:
```
dict_1 = {'ID': [11,22,33,44]}
print(dict_1)
```

```
{'ID': [11, 22, 33, 44]}
```

In [33]:
```
df_3 = pd.DataFrame(dict_1)
df_3
```

Out[33]:

	ID
0	11
1	22
2	33
3	44

In [34]:
```
dict_2 = {'PIP_NAME': ['Python'],'VERSION': [3_10_5]}
print(dict_2)
```

```
{'PIP_NAME': ['Python'], 'VERSION': [3105]}
```

In [35]:
```
df_4 = pd.DataFrame(dict_2)
df_4
```

Out[35]:

	PIP_NAME	VERSION
0	Python	3105

List Of Dict

In [36]:
```
lst_dict = [{'a': 1,'b': 2}, {'a': 3,'b': 4}]
df_5 = pd.DataFrame(lst_dict)
df_5
```

Out[36]:

	a	b
0	1	2
1	3	4

In [37]:
```
lst_dict = [{'a': 1,'b': 2}, {'a': 3,'b': 4,'c': 5}] # Missing Data Handling
df_5 = pd.DataFrame(lst_dict)
df_5
```

Out[37]:

	a	b	c
0	1	2	NaN
1	3	4	5.0

Dict Of Series

In [38]:
```
dict_sr = {'ID': pd.Series([1,2,3]), 'SN': pd.Series([111,222,333])}
df_6 = pd.DataFrame(dict_sr)
df_6
```

Out[38]:

	ID	SN
0	1	111
1	2	222
2	3	333

CSV

In [39]:
```
pd.read_csv('student_info.csv')
```

Out[39]:

	study_hours	student_marks
0	6.83	78.50
1	6.56	76.74
2	NaN	78.68
3	5.67	71.82
4	8.67	84.19
...	...	...
195	7.53	81.67
196	8.56	84.68
197	8.94	86.75
198	6.60	78.05
199	8.35	83.50

200 rows × 2 columns

In [40]:
```
csv_1 = pd.read_csv('student_results.csv')
```

In [41]:
```
csv_1.head()
```

Out[41]:

	Student ID	Class	Study hrs	Sleeping hrs	Social Media usage hrs	Mobile Games hrs	Percantege
0	1001	10	2	9	3	5	50
1	1002	10	6	8	2	0	80
2	1003	10	3	8	2	4	60
3	1004	11	0	10	1	5	45
4	1005	11	4	7	2	0	75

In [42]:
```
# print(os.getcwd())
```

How to Write CSV File In Pandas

In [43]:
```
df = pd.read_csv('C:\\Users\prasad jadhav\Downloads\Pandas\Fortune_10.csv')
df
```

Out[43]:

	ID	Name	Industry	Inception	Revenue	Expenses	Profit	Growth
0	1	Lamtone	IT Services	2009	$11,757,018	6,482,465 Dollars	5274553	30%
1	2	Stripfind	Financial Services	2010	$12,329,371	916,455 Dollars	11412916	20%
2	3	Canecorporation	Health	2012	$10,597,009	7,591,189 Dollars	3005820	7%
3	4	Mattouch	IT Services	2013	$14,026,934	7,429,377 Dollars	6597557	not available
4	5	Techdrill	NaN	2009	$10,573,990	7,435,363 Dollars	3138627	NaN
5	6	Techline	NaN	2006	$13,898,119	5,470,303 Dollars	8427816	NaN
6	7	Cityace	NaN	2010	$9,254,614	6,249,498 Dollars	3005116	6%
7	8	Kayelectronics	NaN	2009	$9,451,943	3,878,113 Dollars	5573830	4%
8	9	Ganzlax	IT Services	2011	$14,001,180	NaN	11901180	18%
9	10	Trantraxlax	Government Services	2011	$11,088,336	5,635,276 Dollars	5453060	7%

In [44]:
```
type(df)
```

Out[44]:
```
pandas.core.frame.DataFrame
```

In [45]:
```
df.columns
```

Out[45]:
```
Index(['ID', 'Name', 'Industry', 'Inception', 'Revenue', 'Expenses', 'Profit',
       'Growth'],
      dtype='object')
```

In [46]:
```
df.head()
```

Out[46]:

	ID	Name	Industry	Inception	Revenue	Expenses	Profit	Growth
0	1	Lamtone	IT Services	2009	$11,757,018	6,482,465 Dollars	5274553	30%
1	2	Stripfind	Financial Services	2010	$12,329,371	916,455 Dollars	11412916	20%
2	3	Canecorporation	Health	2012	$10,597,009	7,591,189 Dollars	3005820	7%
3	4	Mattouch	IT Services	2013	$14,026,934	7,429,377 Dollars	6597557	not available
4	5	Techdrill	NaN	2009	$10,573,990	7,435,363 Dollars	3138627	NaN

In [47]:
```
df = pd.read_csv('C:\\Users\prasad jadhav\Downloads\Pandas\Fortune_10.csv', nr
df
```

Out[47]:

	ID	Name	Industry	Inception	Revenue	Expenses	Profit	Growth
0	1	Lamtone	IT Services	2009	$11,757,018	6,482,465 Dollars	5274553	30%

In [48]:
```
df = pd.read_csv('C:\\Users\prasad jadhav\Downloads\Pandas\Fortune_10.csv', use
df
```

Out[48]:

	ID
0	1
1	2
2	3
3	4
4	5
5	6
6	7
7	8
8	9
9	10

In [49]:
```
# df = pd.read_csv('C:\\Users\prasad jadhav\Downloads\Pandas\Fortune_10.csv', 
# df
```

In [50]:
```
# df = pd.read_csv('C:\\Users\prasad jadhav\Downloads\Pandas\Fortune_10.csv', 
# df
```

In [51]:
```
# df = pd.read_csv('C:\\Users\prasad jadhav\Downloads\Pandas\Fortune_10.csv', 
# df
```

In [52]:
```
df = pd.read_csv('C:\\Users\prasad jadhav\Downloads\Pandas\Fortune_10.csv', hea
df
```

Out[52]:

	1	Lamtone	IT Services	2009	$11,757,018	6,482,465 Dollars	5274553	30%
0	2	Stripfind	Financial Services	2010	$12,329,371	916,455 Dollars	11412916	20%
1	3	Canecorporation	Health	2012	$10,597,009	7,591,189 Dollars	3005820	7%
2	4	Mattouch	IT Services	2013	$14,026,934	7,429,377 Dollars	6597557	not available
3	5	Techdrill	NaN	2009	$10,573,990	7,435,363 Dollars	3138627	NaN
4	6	Techline	NaN	2006	$13,898,119	5,470,303 Dollars	8427816	NaN
5	7	Cityace	NaN	2010	$9,254,614	6,249,498 Dollars	3005116	6%
6	8	Kayelectronics	NaN	2009	$9,451,943	3,878,113 Dollars	5573830	4%
7	9	Ganzlax	IT Services	2011	$14,001,180	NaN	11901180	18%
8	10	Trantraxlax	Government Services	2011	$11,088,336	5,635,276 Dollars	5453060	7%

In [53]:
```
df = pd.read_csv('C:\\Users\prasad jadhav\Downloads\Pandas\Fortune_10.csv', hea
df
```

Out[53]:

	0	1	2	3	4	5	6	7
0	ID	Name	Industry	Inception	Revenue	Expenses	Profit	Growth
1	1	Lamtone	IT Services	2009	$11,757,018	6,482,465 Dollars	5274553	30%
2	2	Stripfind	Financial Services	2010	$12,329,371	916,455 Dollars	11412916	20%
3	3	Canecorporation	Health	2012	$10,597,009	7,591,189 Dollars	3005820	7%
4	4	Mattouch	IT Services	2013	$14,026,934	7,429,377 Dollars	6597557	not available
5	5	Techdrill	NaN	2009	$10,573,990	7,435,363 Dollars	3138627	NaN
6	6	Techline	NaN	2006	$13,898,119	5,470,303 Dollars	8427816	NaN
7	7	Cityace	NaN	2010	$9,254,614	6,249,498 Dollars	3005116	6%
8	8	Kayelectronics	NaN	2009	$9,451,943	3,878,113 Dollars	5573830	4%
9	9	Ganzlax	IT Services	2011	$14,001,180	NaN	11901180	18%
10	10	Trantraxlax	Government Services	2011	$11,088,336	5,635,276 Dollars	5453060	7%

In [54]:
```
# df = pd.read_csv('C:\\Users\prasad jadhav\Downloads\Pandas\Fortune_10.csv', 
# df
```

In [55]:
```
# df = pd.read_csv('C:\\Users\prasad jadhav\Downloads\Pandas\Fortune_10.csv', 
# df
```

In [56]:
```
df = pd.read_csv('C:\\Users\prasad jadhav\Downloads\Pandas\Fortune_10.csv', nar
df
```

Out[56]:

	ID	Name	Industry	Inception	Revenue	Expenses	Profit	Growth
0	ID	Name	Industry	Inception	Revenue	Expenses	Profit	Growth
1	1	Lamtone	IT Services	2009	$11,757,018	6,482,465 Dollars	5274553	30%
2	2	Stripfind	Financial Services	2010	$12,329,371	916,455 Dollars	11412916	20%
3	3	Canecorporation	Health	2012	$10,597,009	7,591,189 Dollars	3005820	7%
4	4	Mattouch	IT Services	2013	$14,026,934	7,429,377 Dollars	6597557	not available
5	5	Techdrill	NaN	2009	$10,573,990	7,435,363 Dollars	3138627	NaN
6	6	Techline	NaN	2006	$13,898,119	5,470,303 Dollars	8427816	NaN
7	7	Cityace	NaN	2010	$9,254,614	6,249,498 Dollars	3005116	6%
8	8	Kayelectronics	NaN	2009	$9,451,943	3,878,113 Dollars	5573830	4%
9	9	Ganzlax	IT Services	2011	$14,001,180	NaN	11901180	18%
10	10	Trantraxlax	Government Services	2011	$11,088,336	5,635,276 Dollars	5453060	7%

In [57]:
```
# df = pd.read_csv('C:\\Users\prasad jadhav\Downloads\Pandas\Fortune_10.csv')
# df
```

In [58]:
```
# df.head(1) #Top_Rows
```

In [59]:
```
# df.tail(1) #End_Rows
```

In [60]:
```
# df = pd.read_csv('C:\\Users\prasad jadhav\Downloads\Pandas\Fortune_10.csv', 
# df
```

In [61]:
```
# df = pd.read_csv('C:\\Users\prasad jadhav\Downloads\Pandas\Fortune_10.csv',
# df
```

Handling Missing Values

In [62]:
```
df = pd.read_csv('C:\\Users\prasad jadhav\Downloads\Pandas\Fortune_10.csv', na_
df
```

Out[62]:

	ID	Name	Industry	Inception	Revenue	Expenses	Profit	Growth
0	1	Lamtone	IT Services	2009	$11,757,018	6,482,465 Dollars	5274553	30%
1	2	Stripfind	Financial Services	2010	$12,329,371	916,455 Dollars	11412916	20%
2	3	Canecorporation	Health	2012	$10,597,009	7,591,189 Dollars	3005820	7%
3	4	Mattouch	IT Services	2013	$14,026,934	7,429,377 Dollars	6597557	NaN
4	5	Techdrill	NaN	2009	$10,573,990	7,435,363 Dollars	3138627	NaN
5	6	Techline	NaN	2006	$13,898,119	5,470,303 Dollars	8427816	NaN
6	7	Cityace	NaN	2010	$9,254,614	6,249,498 Dollars	3005116	6%
7	8	Kayelectronics	NaN	2009	$9,451,943	3,878,113 Dollars	5573830	4%
8	9	Ganzlax	IT Services	2011	$14,001,180	NaN	11901180	18%
9	10	Trantraxlax	Government Services	2011	$11,088,336	5,635,276 Dollars	5453060	7%

In [63]:
```
df = pd.read_csv('C:\\Users\prasad jadhav\Downloads\Pandas\Fortune_10.csv', na_
df
```

Out[63]:

	ID	Name	Industry	Inception	Revenue	Expenses	Profit	Growth
0	1	Lamtone	IT Services	2009	$11,757,018	6,482,465 Dollars	5274553	30%
1	2	Stripfind	Financial Services	2010	$12,329,371	916,455 Dollars	11412916	20%
2	3	Canecorporation	Health	2012	$10,597,009	7,591,189 Dollars	3005820	7%
3	4	Mattouch	IT Services	2013	$14,026,934	7,429,377 Dollars	6597557	NaN
4	5	Techdrill	NaN	2009	$10,573,990	7,435,363 Dollars	3138627	NaN
5	6	Techline	NaN	2006	$13,898,119	5,470,303 Dollars	8427816	NaN
6	7	Cityace	NaN	2010	$9,254,614	6,249,498 Dollars	3005116	6%
7	8	Kayelectronics	NaN	2009	$9,451,943	3,878,113 Dollars	5573830	4%
8	9	Ganzlax	IT Services	2011	$14,001,180	NaN	11901180	18%
9	10	Trantraxlax	Government Services	2011	$11,088,336	5,635,276 Dollars	5453060	7%

In [64]:
```
# df = pd.read_csv('C:\\Users\prasad jadhav\Downloads\Pandas\Fortune_10.csv', 
# df
```

In [65]:
```
# df = pd.read_csv('C:\\Users\prasad jadhav\Downloads\Pandas\Fortune_10.csv', 
# df
```

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

Out[66]:

	ID	Name	Industry	Inception	Revenue	Expenses	Profit	Growth
0	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	True
4	False	False	True	False	False	False	False	True
5	False	False	True	False	False	False	False	True
6	False	False	True	False	False	False	False	False
7	False	False	True	False	False	False	False	False
8	False	False	False	False	False	True	False	False
9	False	False	False	False	False	False	False	False

In [67]:
```
df.isnull().sum()
```

Out[67]:
```
ID           0
Name         0
Industry     4
Inception    0
Revenue      0
Expenses     1
Profit       0
Growth       3
dtype: int64
```

In [68]:
```
df.isnull().sum().sum()
```

Out[68]:
```
8
```

In [69]:
```
df.notnull()
```

Out[69]:

	ID	Name	Industry	Inception	Revenue	Expenses	Profit	Growth
0	True	True	True	True	True	True	True	True
1	True	True	True	True	True	True	True	True
2	True	True	True	True	True	True	True	True
3	True	True	True	True	True	True	True	False
4	True	True	False	True	True	True	True	False
5	True	True	False	True	True	True	True	False
6	True	True	False	True	True	True	True	True
7	True	True	False	True	True	True	True	True
8	True	True	True	True	True	False	True	True
9	True	True	True	True	True	True	True	True

In [70]:
```
df.notnull().sum()
```

Out[70]:
```
ID           10
Name         10
Industry      6
Inception    10
Revenue      10
Expenses      9
Profit       10
Growth        7
dtype: int64
```

In [71]:
```
df.notnull().sum().sum()
```

Out[71]:
```
72
```

Series

In [72]:
```
import numpy as np

sr = pd.Series([1,2,3,np.nan,4,np.NAN])
sr
```

Out[72]:
```
0    1.0
1    2.0
2    3.0
3    NaN
4    4.0
5    NaN
dtype: float64
```

In [73]:
```
sr.isnull()
```

Out[73]:
```
0    False
1    False
2    False
3     True
4    False
5     True
dtype: bool
```

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

Out[74]:
```
2
```

In [75]:
```
df.dropna() #NaN_Values_Removeing (Rows)
```

Out[75]:

	ID	Name	Industry	Inception	Revenue	Expenses	Profit	Growth
0	1	Lamtone	IT Services	2009	$11,757,018	6,482,465 Dollars	5274553	30%
1	2	Stripfind	Financial Services	2010	$12,329,371	916,455 Dollars	11412916	20%
2	3	Canecorporation	Health	2012	$10,597,009	7,591,189 Dollars	3005820	7%
9	10	Trantraxlax	Government Services	2011	$11,088,336	5,635,276 Dollars	5453060	7%

In [76]:
```
df.dropna(1) # 0_Rows, 1_Columns
```

```
C:\Users\prasad jadhav\AppData\Local\Temp\ipykernel_12320\2406135860.py:1: Fut
ureWarning: In a future version of pandas all arguments of DataFrame.dropna wi
ll be keyword-only.
  df.dropna(1) # 0_Rows, 1_Columns
```

Out[76]:

	ID	Name	Inception	Revenue	Profit
0	1	Lamtone	2009	$11,757,018	5274553
1	2	Stripfind	2010	$12,329,371	11412916
2	3	Canecorporation	2012	$10,597,009	3005820
3	4	Mattouch	2013	$14,026,934	6597557
4	5	Techdrill	2009	$10,573,990	3138627
5	6	Techline	2006	$13,898,119	8427816
6	7	Cityace	2010	$9,254,614	3005116
7	8	Kayelectronics	2009	$9,451,943	5573830
8	9	Ganzlax	2011	$14,001,180	11901180
9	10	Trantraxlax	2011	$11,088,336	5453060

In [77]:
```
# df.dropna(axis = 1)
```

In [78]:
```
# df.dropna(how = 'any')
```

In [79]:
```
# df.dropna(axis = 1,how = 'any')
```

In [80]:
```
# df.dropna(how = 'all') #All Rows NaN to Remove nan rows line
```

In [81]:
```
# df.dropna(thresh = 1)
```

In [ ]:
```

```

In [82]:
```
df.dropna(subset = ['Industry','Expenses'])
```

Out[82]:

	ID	Name	Industry	Inception	Revenue	Expenses	Profit	Growth
0	1	Lamtone	IT Services	2009	$11,757,018	6,482,465 Dollars	5274553	30%
1	2	Stripfind	Financial Services	2010	$12,329,371	916,455 Dollars	11412916	20%
2	3	Canecorporation	Health	2012	$10,597,009	7,591,189 Dollars	3005820	7%
3	4	Mattouch	IT Services	2013	$14,026,934	7,429,377 Dollars	6597557	NaN
9	10	Trantraxlax	Government Services	2011	$11,088,336	5,635,276 Dollars	5453060	7%

In [83]:
```
df.dropna(inplace = True)
df
```

Out[83]:

	ID	Name	Industry	Inception	Revenue	Expenses	Profit	Growth
0	1	Lamtone	IT Services	2009	$11,757,018	6,482,465 Dollars	5274553	30%
1	2	Stripfind	Financial Services	2010	$12,329,371	916,455 Dollars	11412916	20%
2	3	Canecorporation	Health	2012	$10,597,009	7,591,189 Dollars	3005820	7%
9	10	Trantraxlax	Government Services	2011	$11,088,336	5,635,276 Dollars	5453060	7%

Fillna

In [84]:
```
df.fillna({'Industry': 'Health','Expenses': '7,591,189 Dollars','Growth': '7%'}
```

Out[84]:

	ID	Name	Industry	Inception	Revenue	Expenses	Profit	Growth
0	1	Lamtone	IT Services	2009	$11,757,018	6,482,465 Dollars	5274553	30%
1	2	Stripfind	Financial Services	2010	$12,329,371	916,455 Dollars	11412916	20%
2	3	Canecorporation	Health	2012	$10,597,009	7,591,189 Dollars	3005820	7%
9	10	Trantraxlax	Government Services	2011	$11,088,336	5,635,276 Dollars	5453060	7%

In [85]:
```
df.fillna(method = 'ffill')
```

Out[85]:

	ID	Name	Industry	Inception	Revenue	Expenses	Profit	Growth
0	1	Lamtone	IT Services	2009	$11,757,018	6,482,465 Dollars	5274553	30%
1	2	Stripfind	Financial Services	2010	$12,329,371	916,455 Dollars	11412916	20%
2	3	Canecorporation	Health	2012	$10,597,009	7,591,189 Dollars	3005820	7%
9	10	Trantraxlax	Government Services	2011	$11,088,336	5,635,276 Dollars	5453060	7%

In [86]:
```
df.fillna(method = 'pad')
```

Out[86]:

	ID	Name	Industry	Inception	Revenue	Expenses	Profit	Growth
0	1	Lamtone	IT Services	2009	$11,757,018	6,482,465 Dollars	5274553	30%
1	2	Stripfind	Financial Services	2010	$12,329,371	916,455 Dollars	11412916	20%
2	3	Canecorporation	Health	2012	$10,597,009	7,591,189 Dollars	3005820	7%
9	10	Trantraxlax	Government Services	2011	$11,088,336	5,635,276 Dollars	5453060	7%

In [87]:
```
df.fillna(method = 'bfill')
```

Out[87]:

	ID	Name	Industry	Inception	Revenue	Expenses	Profit	Growth
0	1	Lamtone	IT Services	2009	$11,757,018	6,482,465 Dollars	5274553	30%
1	2	Stripfind	Financial Services	2010	$12,329,371	916,455 Dollars	11412916	20%
2	3	Canecorporation	Health	2012	$10,597,009	7,591,189 Dollars	3005820	7%
9	10	Trantraxlax	Government Services	2011	$11,088,336	5,635,276 Dollars	5453060	7%

In [88]:
```
# df.fillna(method = 'ffill', axis = 0)
```

In [89]:
```
# df.fillna(method = 'bfill', axis = 0)
```

In [90]:
```
# df.fillna(0, limit = 5)
```

In [91]:
```
# df.fillna(method = 'ffill', limit = 5)
```

In [92]:
```
# df.fillna(5,inplace = True)
# df
```

Replace

In [93]:
```
df = pd.read_csv('C:\\Users\prasad jadhav\Downloads\Pandas\Fortune_10.csv')
```

```
df
```

Out[93]:

	ID	Name	Industry	Inception	Revenue	Expenses	Profit	Growth
0	1	Lamtone	IT Services	2009	$11,757,018	6,482,465 Dollars	5274553	30%
1	2	Stripfind	Financial Services	2010	$12,329,371	916,455 Dollars	11412916	20%
2	3	Canecorporation	Health	2012	$10,597,009	7,591,189 Dollars	3005820	7%
3	4	Mattouch	IT Services	2013	$14,026,934	7,429,377 Dollars	6597557	not available
4	5	Techdrill	NaN	2009	$10,573,990	7,435,363 Dollars	3138627	NaN
5	6	Techline	NaN	2006	$13,898,119	5,470,303 Dollars	8427816	NaN
6	7	Cityace	NaN	2010	$9,254,614	6,249,498 Dollars	3005116	6%
7	8	Kayelectronics	NaN	2009	$9,451,943	3,878,113 Dollars	5573830	4%
8	9	Ganzlax	IT Services	2011	$14,001,180	NaN	11901180	18%
9	10	Trantraxlax	Government Services	2011	$11,088,336	5,635,276 Dollars	5453060	7%

In [94]:
```
# df.replace(to_replace = 'IT Services', value = 'Health')
```

In [95]:
```
df.replace({'Growth': 'not available'}, '30%')
```

Out[95]:

	ID	Name	Industry	Inception	Revenue	Expenses	Profit	Growth
0	1	Lamtone	IT Services	2009	$11,757,018	6,482,465 Dollars	5274553	30%
1	2	Stripfind	Financial Services	2010	$12,329,371	916,455 Dollars	11412916	20%
2	3	Canecorporation	Health	2012	$10,597,009	7,591,189 Dollars	3005820	7%
3	4	Mattouch	IT Services	2013	$14,026,934	7,429,377 Dollars	6597557	30%
4	5	Techdrill	NaN	2009	$10,573,990	7,435,363 Dollars	3138627	NaN
5	6	Techline	NaN	2006	$13,898,119	5,470,303 Dollars	8427816	NaN
6	7	Cityace	NaN	2010	$9,254,614	6,249,498 Dollars	3005116	6%
7	8	Kayelectronics	NaN	2009	$9,451,943	3,878,113 Dollars	5573830	4%
8	9	Ganzlax	IT Services	2011	$14,001,180	NaN	11901180	18%
9	10	Trantraxlax	Government Services	2011	$11,088,336	5,635,276 Dollars	5453060	7%

In [96]:
```
# df.replace({'Industry': 'Health'}, 'Financial Services')
```

In [97]:
```
df.replace('[A-Za-z]',0,regex = True)
```

Out[97]:

	ID	Name	Industry	Inception	Revenue	Expenses	Profit	Growth
0	1	0	0.0	2009	$11,757,018	0.0	5274553	30%
1	2	0	0.0	2010	$12,329,371	0.0	11412916	20%
2	3	0	0.0	2012	$10,597,009	0.0	3005820	7%
3	4	0	0.0	2013	$14,026,934	0.0	6597557	0
4	5	0	NaN	2009	$10,573,990	0.0	3138627	NaN
5	6	0	NaN	2006	$13,898,119	0.0	8427816	NaN
6	7	0	NaN	2010	$9,254,614	0.0	3005116	6%
7	8	0	NaN	2009	$9,451,943	0.0	5573830	4%
8	9	0	0.0	2011	$14,001,180	NaN	11901180	18%
9	10	0	0.0	2011	$11,088,336	0.0	5453060	7%

In [98]:
```
df.replace({'Industry': '[A-Za-z]'},0,regex = True)
```

Out[98]:

	ID	Name	Industry	Inception	Revenue	Expenses	Profit	Growth
0	1	Lamtone	0.0	2009	$11,757,018	6,482,465 Dollars	5274553	30%
1	2	Stripfind	0.0	2010	$12,329,371	916,455 Dollars	11412916	20%
2	3	Canecorporation	0.0	2012	$10,597,009	7,591,189 Dollars	3005820	7%
3	4	Mattouch	0.0	2013	$14,026,934	7,429,377 Dollars	6597557	not available
4	5	Techdrill	NaN	2009	$10,573,990	7,435,363 Dollars	3138627	NaN
5	6	Techline	NaN	2006	$13,898,119	5,470,303 Dollars	8427816	NaN
6	7	Cityace	NaN	2010	$9,254,614	6,249,498 Dollars	3005116	6%
7	8	Kayelectronics	NaN	2009	$9,451,943	3,878,113 Dollars	5573830	4%
8	9	Ganzlax	0.0	2011	$14,001,180	NaN	11901180	18%
9	10	Trantraxlax	0.0	2011	$11,088,336	5,635,276 Dollars	5453060	7%

In [99]:
```
df.replace('Health',method = 'ffill')
```

Out[99]:

	ID	Name	Industry	Inception	Revenue	Expenses	Profit	Growth
0	1	Lamtone	IT Services	2009	$11,757,018	6,482,465 Dollars	5274553	30%
1	2	Stripfind	Financial Services	2010	$12,329,371	916,455 Dollars	11412916	20%
2	3	Canecorporation	Financial Services	2012	$10,597,009	7,591,189 Dollars	3005820	7%
3	4	Mattouch	IT Services	2013	$14,026,934	7,429,377 Dollars	6597557	not available
4	5	Techdrill	NaN	2009	$10,573,990	7,435,363 Dollars	3138627	NaN
5	6	Techline	NaN	2006	$13,898,119	5,470,303 Dollars	8427816	NaN
6	7	Cityace	NaN	2010	$9,254,614	6,249,498 Dollars	3005116	6%
7	8	Kayelectronics	NaN	2009	$9,451,943	3,878,113 Dollars	5573830	4%
8	9	Ganzlax	IT Services	2011	$14,001,180	NaN	11901180	18%
9	10	Trantraxlax	Government Services	2011	$11,088,336	5,635,276 Dollars	5453060	7%

In [100…

```
df.replace('Health',method = 'bfill')
```

Out[100]:

	ID	Name	Industry	Inception	Revenue	Expenses	Profit	Growth
0	1	Lamtone	IT Services	2009	$11,757,018	6,482,465 Dollars	5274553	30%
1	2	Stripfind	Financial Services	2010	$12,329,371	916,455 Dollars	11412916	20%
2	3	Canecorporation	IT Services	2012	$10,597,009	7,591,189 Dollars	3005820	7%
3	4	Mattouch	IT Services	2013	$14,026,934	7,429,377 Dollars	6597557	not available
4	5	Techdrill	NaN	2009	$10,573,990	7,435,363 Dollars	3138627	NaN
5	6	Techline	NaN	2006	$13,898,119	5,470,303 Dollars	8427816	NaN
6	7	Cityace	NaN	2010	$9,254,614	6,249,498 Dollars	3005116	6%
7	8	Kayelectronics	NaN	2009	$9,451,943	3,878,113 Dollars	5573830	4%
8	9	Ganzlax	IT Services	2011	$14,001,180	NaN	11901180	18%
9	10	Trantraxlax	Government Services	2011	$11,088,336	5,635,276 Dollars	5453060	7%

In [101…

```
# df.replace(0,method = 'bfill',limit = 1)
```

In [102…

```
df.replace(0,100,inplace = True)
```

```
df
```

Out[102]:

	ID	Name	Industry	Inception	Revenue	Expenses	Profit	Growth
0	1	Lamtone	IT Services	2009	$11,757,018	6,482,465 Dollars	5274553	30%
1	2	Stripfind	Financial Services	2010	$12,329,371	916,455 Dollars	11412916	20%
2	3	Canecorporation	Health	2012	$10,597,009	7,591,189 Dollars	3005820	7%
3	4	Mattouch	IT Services	2013	$14,026,934	7,429,377 Dollars	6597557	not available
4	5	Techdrill	NaN	2009	$10,573,990	7,435,363 Dollars	3138627	NaN
5	6	Techline	NaN	2006	$13,898,119	5,470,303 Dollars	8427816	NaN
6	7	Cityace	NaN	2010	$9,254,614	6,249,498 Dollars	3005116	6%
7	8	Kayelectronics	NaN	2009	$9,451,943	3,878,113 Dollars	5573830	4%
8	9	Ganzlax	IT Services	2011	$14,001,180	NaN	11901180	18%
9	10	Trantraxlax	Government Services	2011	$11,088,336	5,635,276 Dollars	5453060	7%

Interpolate

In [103…

```
df = pd.read_csv('C:\\Users\prasad jadhav\Downloads\Pandas\student_10.csv',pars
df
```

Out[103]:

Date	Student ID	Class	Section	Study Hrs	Percantege
2019-01-07	1001.0	10	A	2.0	50.0
2019-02-07	1002.0	10	NaN	NaN	80.0
2019-03-07	NaN	10	NaN	3.0	60.0
2019-06-07	NaN	11	NaN	0.0	NaN
2019-07-07	1005.0	11	A	NaN	NaN

In [104…

```
df.interpolate(method = 'time')
```

Out[104]:

Date	Student ID	Class	Section	Study Hrs	Percantege
2019-01-07	1001.00	10	A	2.000000	50.0
2019-02-07	1002.00	10	NaN	2.525424	80.0
2019-03-07	1002.56	10	NaN	3.000000	60.0
2019-06-07	1004.40	11	NaN	0.000000	60.0
2019-07-07	1005.00	11	A	0.000000	60.0

In [105…
```
df.interpolate(method = 'linear')
```

Out[105]:

Date	Student ID	Class	Section	Study Hrs	Percantege
2019-01-07	1001.0	10	A	2.0	50.0
2019-02-07	1002.0	10	NaN	2.5	80.0
2019-03-07	1003.0	10	NaN	3.0	60.0
2019-06-07	1004.0	11	NaN	0.0	60.0
2019-07-07	1005.0	11	A	0.0	60.0

In [106…
```
df.interpolate(method = 'index')
```

Out[106]:

Date	Student ID	Class	Section	Study Hrs	Percantege
2019-01-07	1001.00	10	A	2.000000	50.0
2019-02-07	1002.00	10	NaN	2.525424	80.0
2019-03-07	1002.56	10	NaN	3.000000	60.0
2019-06-07	1004.40	11	NaN	0.000000	60.0
2019-07-07	1005.00	11	A	0.000000	60.0

In [107…
```
# df.interpolate(method = 'polynomial',order = 1)
```

In [108…
```
df.interpolate(method = 'spline',order = 2)
```

Out[108]:

Date	Student ID	Class	Section	Study Hrs	Percantege
2019-01-07	1001.000000	10	A	2.000000	50.000000
2019-02-07	1002.000000	10	NaN	2.810299	80.000000
2019-03-07	1002.791346	10	NaN	3.000000	60.000000
2019-06-07	1004.643807	11	NaN	0.000000	-320.453019
2019-07-07	1005.000000	11	A	-2.179464	-548.856518

```

```

In [109…
```
df.interpolate()
```

Out[109]:

Date	Student ID	Class	Section	Study Hrs	Percantege
2019-01-07	1001.0	10	A	2.0	50.0
2019-02-07	1002.0	10	NaN	2.5	80.0
2019-03-07	1003.0	10	NaN	3.0	60.0
2019-06-07	1004.0	11	NaN	0.0	60.0
2019-07-07	1005.0	11	A	0.0	60.0

In [110…
```
df = pd.read_csv('C:\\Users\prasad jadhav\Downloads\Pandas\student_10.csv')
df
```

Out[110]:

	Date	Student ID	Class	Section	Study Hrs	Percantege
0	01-07-2019	1001.0	10	A	2.0	50.0
1	02-07-2019	1002.0	10	NaN	NaN	80.0
2	03-07-2019	NaN	10	NaN	3.0	60.0
3	06-07-2019	NaN	11	NaN	0.0	NaN
4	07-07-2019	1005.0	11	A	NaN	NaN

In [111…
```
df.dtypes
```

Out[111]:
```
Date           object
Student ID    float64
Class           int64
Section        object
Study Hrs     float64
Percantege    float64
dtype: object
```

In [112…
```
df.interpolate(limit = 5)
```

Out[112]:

	Date	Student ID	Class	Section	Study Hrs	Percantege
0	01-07-2019	1001.0	10	A	2.0	50.0
1	02-07-2019	1002.0	10	NaN	2.5	80.0
2	03-07-2019	1003.0	10	NaN	3.0	60.0
3	06-07-2019	1004.0	11	NaN	0.0	60.0
4	07-07-2019	1005.0	11	A	0.0	60.0

In [113…
```
# df.interpolate(limit = 10,limit_direction = 'backward')
```

In [114…
```
df.interpolate(limit = 10,limit_direction = 'both')
```

Out[114]:

	Date	Student ID	Class	Section	Study Hrs	Percantege
0	01-07-2019	1001.0	10	A	2.0	50.0
1	02-07-2019	1002.0	10	NaN	2.5	80.0
2	03-07-2019	1003.0	10	NaN	3.0	60.0
3	06-07-2019	1004.0	11	NaN	0.0	60.0
4	07-07-2019	1005.0	11	A	0.0	60.0

In [115…
```
# df.interpolate(limit_area = 'inside')
```

In [116…
```
# df.interpolate(limit_area = 'outside')
```

In [117…
```
df.interpolate(inplace = True)
df
```

Out[117]:

	Date	Student ID	Class	Section	Study Hrs	Percantege
0	01-07-2019	1001.0	10	A	2.0	50.0
1	02-07-2019	1002.0	10	NaN	2.5	80.0
2	03-07-2019	1003.0	10	NaN	3.0	60.0
3	06-07-2019	1004.0	11	NaN	0.0	60.0
4	07-07-2019	1005.0	11	A	0.0	60.0

In [118…
```
# df.drop('Section',axis = 1)
```

In [119…
```
# df.interpolate(method = 'polynomial',order = 1)
```

In [120…
```
# df.interpolate(
#     method:
#     axis:
#     limit:
#     inplace:
#     limit_direction:
#     limit_area:
#     downcast: )
```

Loc & Iloc

In [121…
```
df = pd.read_csv('C:\\Users\prasad jadhav\Downloads\Pandas\student_results.csv
df
```

Out[121]:

	Student ID	Class	Study hrs	Sleeping hrs	Social Media usage hrs	Mobile Games hrs	Percantege
0	1001	10	2	9	3	5	50
1	1002	10	6	8	2	0	80
2	1003	10	3	8	2	4	60
3	1004	11	0	10	1	5	45
4	1005	11	4	7	2	0	75
5	1006	11	10	7	0	0	96
6	1007	12	4	6	0	0	80
7	1008	12	10	6	2	0	90
8	1009	12	2	8	2	4	60
9	1010	12	6	9	1	0	85

In [122…
```
df.loc[0]
```

Out[122]:
```
Student ID                1001
Class                       10
Study hrs                    2
Sleeping hrs                 9
Social Media usage hrs       3
Mobile Games hrs             5
Percantege                  50
Name: 0, dtype: int64
```

In [123…
```
df.loc[4]
```

Out[123]:
```
Student ID                1005
Class                       11
Study hrs                    4
Sleeping hrs                 7
Social Media usage hrs       2
Mobile Games hrs             0
Percantege                  75
Name: 4, dtype: int64
```

In [124…
```
df.loc[[0,1]]
```

Out[124]:

	Student ID	Class	Study hrs	Sleeping hrs	Social Media usage hrs	Mobile Games hrs	Percantege
0	1001	10	2	9	3	5	50
1	1002	10	6	8	2	0	80

In [125…
```
df.loc[[2,4]]
```

Out[125]:

	Student ID	Class	Study hrs	Sleeping hrs	Social Media usage hrs	Mobile Games hrs	Percantege
2	1003	10	3	8	2	4	60
4	1005	11	4	7	2	0	75

In [126…
```
df.loc[4,'Class']
```

Out[126]:
```
11
```

In [127…
```
df.loc[0:3,'Class']
```

Out[127]:
```
0    10
1    10
2    10
3    11
Name: Class, dtype: int64
```

In [128…
```
df.loc[0:2,'Percantege']
```

Out[128]:
```
0    50
1    80
2    60
Name: Percantege, dtype: int64
```

In [129…
```
df.loc[[0,False,False,True]]
```

Out[129]:

	Student ID	Class	Study hrs	Sleeping hrs	Social Media usage hrs	Mobile Games hrs	Percantege
0	1001	10	2	9	3	5	50
0	1001	10	2	9	3	5	50
0	1001	10	2	9	3	5	50
1	1002	10	6	8	2	0	80

In [130…
```
df.loc[df['Class'] < 11,['Percantege']]
```

Out[130]:

	Percantege
0	50
1	80
2	60

Iloc

In [131…
```
df.iloc[0]
```

Out[131]:
```
Student ID                1001
Class                       10
Study hrs                    2
Sleeping hrs                 9
Social Media usage hrs       3
Mobile Games hrs             5
Percantege                  50
Name: 0, dtype: int64
```

In [132…
```
df.iloc[[0]]
```

Out[132]:

	Student ID	Class	Study hrs	Sleeping hrs	Social Media usage hrs	Mobile Games hrs	Percantege
0	1001	10	2	9	3	5	50

In [133…
```
df.iloc[:,0]
```

Out[133]:
```
0    1001
1    1002
2    1003
3    1004
4    1005
5    1006
6    1007
7    1008
8    1009
9    1010
Name: Student ID, dtype: int64
```

In [134…
```
df.iloc[[0,1]]
```

Out[134]:

	Student ID	Class	Study hrs	Sleeping hrs	Social Media usage hrs	Mobile Games hrs	Percantege
0	1001	10	2	9	3	5	50
1	1002	10	6	8	2	0	80

In [135…
```
df.iloc[[0,True,False,True]]
```

Out[135]:

	Student ID	Class	Study hrs	Sleeping hrs	Social Media usage hrs	Mobile Games hrs	Percantege
0	1001	10	2	9	3	5	50
1	1002	10	6	8	2	0	80
0	1001	10	2	9	3	5	50
1	1002	10	6	8	2	0	80

GroupBy

In [136…
```
# -Splitting the object
# -Applying a function
# -Combining the result
```

In [137…
```
df = pd.read_csv('C:\\Users\prasad jadhav\Downloads\Pandas\student_result_5.csv
df
```

Out[137]:

	Student ID	Section	Class	Study hrs	Social Media usage hrs	Percentage
0	1001	A	10	2	3	50
1	1002	B	10	6	2	80
2	1003	A	10	3	2	60
3	1004	C	11	0	1	45
4	1005	C	12	5	2	75

In [138…
```
gr_1 = df.groupby(by = 'Section')
gr_1
```

Out[138]:
```
<pandas.core.groupby.generic.DataFrameGroupBy object at 0x000002EA73AB3AC0>
```

In [139…
```
gr_1.groups
```

Out[139]:
```
{'A': [0, 2], 'B': [1], 'C': [3, 4]}
```

In [140…
```
df.groupby(['Section','Class']).groups
```

Out[140]:
```
{('A', 10): [0, 2], ('B', 10): [1], ('C', 11): [3], ('C', 12): [4]}
```

In [141…
```
for Section, df_1 in gr_1:
    print(Section)
    print(df_1)
```

```
A
   Student ID Section  Class  Study hrs  Social Media usage hrs  Percentage
0        1001       A     10          2                       3          50
2        1003       A     10          3                       2          60
B
   Student ID Section  Class  Study hrs  Social Media usage hrs  Percentage
1        1002       B     10          6                       2          80
C
   Student ID Section  Class  Study hrs  Social Media usage hrs  Percentage
3        1004       C     11          0                       1          45
4        1005       C     12          5                       2          75
```

In [142…
```
dict(list(gr_1))
```

Out[142]:
```
{'A':    Student ID Section  Class  Study hrs  Social Media usage hrs  Percen
tage
 0        1001       A     10          2                       3          50
 2        1003       A     10          3                       2          60,
 'B':    Student ID Section  Class  Study hrs  Social Media usage hrs  Percen
tage
 1        1002       B     10          6                       2          80,
 'C':    Student ID Section  Class  Study hrs  Social Media usage hrs  Percen
tage
 3        1004       C     11          0                       1          45
 4        1005       C     12          5                       2          75}
```

In [143…
```
gr_2 = df.groupby('Class').get_group(10)
gr_2
```

Out[143]:

	Student ID	Section	Class	Study hrs	Social Media usage hrs	Percentage
0	1001	A	10	2	3	50
1	1002	B	10	6	2	80
2	1003	A	10	3	2	60

In [144…
```
gr_3 = df.groupby('Section').get_group('A')
gr_3
```

Out[144]:

	Student ID	Section	Class	Study hrs	Social Media usage hrs	Percentage
0	1001	A	10	2	3	50
2	1003	A	10	3	2	60

In [145…
```
gr_1.sum()
```

Out[145]:

Section	Student ID	Class	Study hrs	Social Media usage hrs	Percentage
A	2004	20	5	5	110
B	1002	10	6	2	80
C	2009	23	5	3	120

In [146…
```
gr_1.mean()
```

Out[146]:

Section	Student ID	Class	Study hrs	Social Media usage hrs	Percentage
A	1002.0	10.0	2.5	2.5	55.0
B	1002.0	10.0	6.0	2.0	80.0
C	1004.5	11.5	2.5	1.5	60.0

In [147…
```
gr_1.describe()
```

Out[147]:

	Student ID								Class		...	S M usag
	count	mean	std	min	25%	50%	75%	max	count	mean	...	75%
Section												
A	2.0	1002.0	1.414214	1001.0	1001.50	1002.0	1002.50	1003.0	2.0	10.0	...	2.75
B	1.0	1002.0	NaN	1002.0	1002.00	1002.0	1002.00	1002.0	1.0	10.0	...	2.00
C	2.0	1004.5	0.707107	1004.0	1004.25	1004.5	1004.75	1005.0	2.0	11.5	...	1.75

3 rows × 40 columns

In [148…
```
gr_1.agg(['sum','max','min'])
```

Out[148]:

	Student ID			Class			Study hrs			Social Media usage hrs			Percentage		
	sum	max	min	sum	max	min	sum	max	min	sum	max	min	sum	max	min
Section															
A	2004	1003	1001	20	10	10	5	3	2	5	3	2	110	60	50
B	1002	1002	1002	10	10	10	6	6	6	2	2	2	80	80	80
C	2009	1005	1004	23	12	11	5	5	0	3	2	1	120	75	45

Merge

In [149…
```
df_1 = pd.DataFrame({'ID': [1,2,3,4],'Class': [9,10,11,12]})
df_1
```

Out[149]:

	ID	Class
0	1	9
1	2	10
2	3	11
3	4	12

In [150…
```
df_2 = pd.DataFrame({'ID': [1,2,3,4],'Name': ['A','B','C','D']})
df_2
```

Out[150]:

	ID	Name
0	1	A
1	2	B
2	3	C
3	4	D

In [151…
```
pd.merge(df_1,df_2,on = 'ID')
```

Out[151]:

	ID	Class	Name
0	1	9	A
1	2	10	B
2	3	11	C
3	4	12	D

In [152…
```
pd.merge(df_1,df_2,on = 'ID')
```

Out[152]:

	ID	Class	Name
0	1	9	A
1	2	10	B
2	3	11	C
3	4	12	D

In [153…
```
# pd.merge(df_2,df_1,on = 'ID')
```

In [154…
```
# pd.merge(df_1,df_2,on = 'ID',how = 'left')
```

In [155…
```
# pd.merge(df_1,df_2,on = 'ID',how = 'right')
```

In [156…
```
# pd.merge(df_1,df_2,on = 'ID',how = 'outer')
```

In [157…
```
# pd.merge(df_1,df_2,on = 'ID',how = 'left', indicator = True)
```

In [158…
```
df_3 = pd.DataFrame({'ID': [5,6,7,8],'Name': ['A','B','C','D']})
df_3
```

Out[158]:

	ID	Name
0	5	A
1	6	B
2	7	C
3	8	D

In [159…
```
pd.merge(df_1,df_3,left_index = True,right_index = True)
```

Out[159]:

	ID_x	Class	ID_y	Name
0	1	9	5	A
1	2	10	6	B
2	3	11	7	C
3	4	12	8	D

In [160…
```
df_2 = pd.DataFrame({'ID': [1,2,3,4],'Class': [9,10,11,12]})
df_2
```

Out[160]:

	ID	Class
0	1	9
1	2	10
2	3	11
3	4	12

In [161…
```
pd.merge(df_1,df_2,on = 'ID',suffixes = ('_Higher','_Middle'))
```

Out[161]:

	ID	Class_Higher	Class_Middle
0	1	9	9
1	2	10	10
2	3	11	11
3	4	12	12

Concat

In [162…
```
sr_1 = pd.Series([0,1,2])
sr_1
```

Out[162]:
```
0    0
1    1
2    2
dtype: int64
```

In [163…
```
sr_2 = pd.Series([3,4,5])
sr_2
```

Out[163]:
```
0    3
1    4
2    5
dtype: int64
```

In [164…
```
pd.concat([sr_1,sr_2])
```

Out[164]:
```
0    0
1    1
2    2
0    3
1    4
2    5
dtype: int64
```

In [165…
```
_df1 = pd.DataFrame({'ID': [1,2,3,4],'Name': ['A','B','C','D'],'Class': [5,6,7
_df1
```

Out[165]:

	ID	Name	Class
0	1	A	5
1	2	B	6
2	3	C	7
3	4	D	8

In [166…
```
_df2 = pd.DataFrame({'ID': [5,6,7,8],'Name': ['E','F','G','H'],'Class': [9,10,
_df2
```

Out[166]:

	ID	Name	Class
0	5	E	9
1	6	F	10
2	7	G	11
3	8	H	12

In [167…
```
pd.concat([_df1,_df2])
```

Out[167]:

	ID	Name	Class
0	1	A	5
1	2	B	6
2	3	C	7
3	4	D	8
0	5	E	9
1	6	F	10
2	7	G	11
3	8	H	12

In [168…
```
pd.concat([_df1,_df2],axis = 1)
```

Out[168]:

	ID	Name	Class	ID	Name	Class
0	1	A	5	5	E	9
1	2	B	6	6	F	10
2	3	C	7	7	G	11
3	4	D	8	8	H	12

In [169…
```
pd.concat([_df1,_df2],axis = 0,ignore_index = True)
```

Out[169]:

	ID	Name	Class
0	1	A	5
1	2	B	6
2	3	C	7
3	4	D	8
4	5	E	9
5	6	F	10
6	7	G	11
7	8	H	12

In [170…
```
_df3 = pd.DataFrame({'ID': [1,2,3,4],'Name': ['A','B','C','D'],'Class': [5,6,7
_df3
```

Out[170]:

	ID	Name	Class
0	1	A	5
1	2	B	6
2	3	C	7
3	4	D	8

In [171…
```
_df4 = pd.DataFrame({'ID': [3,4],'Name': ['C','D'],'Class': [7,8]})
_df4
```

Out[171]:

	ID	Name	Class
0	3	C	7
1	4	D	8

In [172…
```
pd.concat([_df3,_df4])
```

Out[172]:

	ID	Name	Class
0	1	A	5
1	2	B	6
2	3	C	7
3	4	D	8
0	3	C	7
1	4	D	8

In [173…
```
pd.concat([_df3,_df4],axis = 1)
```

Out[173]:

	ID	Name	Class	ID	Name	Class
0	1	A	5	3.0	C	7.0
1	2	B	6	4.0	D	8.0
2	3	C	7	NaN	NaN	NaN
3	4	D	8	NaN	NaN	NaN

In [174…
```
pd.concat([_df3,_df4],axis = 1,join = 'inner')
```

Out[174]:

	ID	Name	Class	ID	Name	Class
0	1	A	5	3	C	7
1	2	B	6	4	D	8

In [175…
```
# pd.concat([_df3,_df4],axis = 1,join = [_df3.index])
```

In [176…
```
pd.concat([_df1,_df2],keys = ['df_1','df_2'])
```

Out[176]:

		ID	Name	Class
df_1	**0**	1	A	5
	1	2	B	6
	2	3	C	7
	3	4	D	8
df_2	**0**	5	E	9
	1	6	F	10
	2	7	G	11
	3	8	H	12

In [177…
```
pd.concat([_df1,_df2],keys = ['First_df','Second_df'])
```

Out[177]:

		ID	Name	Class
First_df	**0**	1	A	5
	1	2	B	6
	2	3	C	7
	3	4	D	8
Second_df	**0**	5	E	9
	1	6	F	10
	2	7	G	11
	3	8	H	12

In [178…
```
pd.concat([_df1,_df2],axis =1,keys = ['First_df','Second_df'])
```

Out[178]:

	First_df			Second_df		
	ID	**Name**	**Class**	**ID**	**Name**	**Class**
0	1	A	5	5	E	9
1	2	B	6	6	F	10
2	3	C	7	7	G	11
3	4	D	8	8	H	12

In [179…
```
_df5 = pd.DataFrame({'ID': [1,2,3,4],'Name': ['A','B','C','D'],'Class': [5,6,7
_df5
```

Out[179]:

	ID	Name	Class
0	1	A	5
1	2	B	6
2	3	C	7
3	4	D	8

In [180…
```
_df6 = pd.DataFrame({'Marks': [40,63,91,34]})
_df6
```

Out[180]:

	Marks
0	40
1	63
2	91
3	34

In [181…
```
pd.concat([_df5,_df6],sort = False)
```

Out[181]:

	ID	Name	Class	Marks
0	1.0	A	5.0	NaN
1	2.0	B	6.0	NaN
2	3.0	C	7.0	NaN
3	4.0	D	8.0	NaN
0	NaN	NaN	NaN	40.0
1	NaN	NaN	NaN	63.0
2	NaN	NaN	NaN	91.0
3	NaN	NaN	NaN	34.0

Join

In [182…
```
df__1 = pd.DataFrame({'A': [1,2,3],'B': [10,20,30]})
df__2 = pd.DataFrame({'C': [4,5,6],'D': [40,50,60]})
```

In [183…
```
display(df__1,df__2)
```

	A	B
0	1	10
1	2	20
2	3	30

	C	D
0	4	40
1	5	50
2	6	60

In [184…
```
df__1.join(df__2)
```

Out[184]:

	A	B	C	D
0	1	10	4	40
1	2	20	5	50
2	3	30	6	60

Append

In [185…
```
df_1 = pd.DataFrame({'A': [1,2,3],'B': [10,20,30]})
df_2 = pd.DataFrame({'A': [4,5,6],'B': [40,50,60]})

display(df_1,df_2)
```

	A	B
0	1	10
1	2	20
2	3	30

	A	B
0	4	40
1	5	50
2	6	60

In [186…
```
df_1.append(df_2,ignore_index = True,sort = False)
```

```
C:\Users\prasad jadhav\AppData\Local\Temp\ipykernel_12320\4145945755.py:1: Fut
ureWarning: The frame.append method is deprecated and will be removed from pan
das in a future version. Use pandas.concat instead.
  df_1.append(df_2,ignore_index = True,sort = False)
```

Out[186]:

	A	B
0	1	10
1	2	20
2	3	30
3	4	40
4	5	50
5	6	60

Pivot Table

In [187…
```
df = pd.read_csv('C:\\Users\prasad jadhav\Downloads\Pandas\Movie_ratings.csv')
df
```

Out[187]:

	Genre	Year	Budget(MS)	Ratings %
0	Action	2008	28	48
1	Action	2009	200	63
2	Action	2008	32	57
3	Action	2010	20	52
4	Adventure	2009	18	84
5	Adventure	2010	105	44
6	Comedy	2010	20	70
7	Comedy	2008	8	81
8	Comedy	2009	30	71

In [188…
```
df.pivot_table(index = 'Genre') #Total_Mean/Avg
```

Out[188]:

Genre	Budget(MS)	Ratings %	Year
Action	70.000000	55	2008.75
Adventure	61.500000	64	2009.50
Comedy	19.333333	74	2009.00

In [189…
```
df.pivot_table(index = 'Genre',columns = 'Year') #Mean_Avg
```

Out[189]:

	Budget(MS)			Ratings %		
Year	**2008**	**2009**	**2010**	**2008**	**2009**	**2010**
Genre						
Action	30.0	200.0	20.0	52.5	63.0	52.0
Adventure	NaN	18.0	105.0	NaN	84.0	44.0
Comedy	8.0	30.0	20.0	81.0	71.0	70.0

In [190…
```
df.pivot_table(index = 'Genre',columns = 'Year',aggfunc = 'count')
```

Out[190]:

	Budget(MS)			Ratings %		
Year	**2008**	**2009**	**2010**	**2008**	**2009**	**2010**
Genre						
Action	2.0	1.0	1.0	2.0	1.0	1.0
Adventure	NaN	1.0	1.0	NaN	1.0	1.0
Comedy	1.0	1.0	1.0	1.0	1.0	1.0

In [191…
```
df.pivot_table(index = 'Genre',columns = 'Year',aggfunc = 'max')
```

Out[191]:

	Budget(MS)			Ratings %		
Year	**2008**	**2009**	**2010**	**2008**	**2009**	**2010**
Genre						
Action	32.0	200.0	20.0	57.0	63.0	52.0
Adventure	NaN	18.0	105.0	NaN	84.0	44.0
Comedy	8.0	30.0	20.0	81.0	71.0	70.0

In [192…
```
df.pivot_table(index = 'Genre',columns = 'Year',aggfunc = 'sum')
```

Out[192]:

	Budget(MS)			Ratings %		
Year	**2008**	**2009**	**2010**	**2008**	**2009**	**2010**
Genre						
Action	60.0	200.0	20.0	105.0	63.0	52.0
Adventure	NaN	18.0	105.0	NaN	84.0	44.0
Comedy	8.0	30.0	20.0	81.0	71.0	70.0

In [193…
```
df.pivot_table(index = 'Genre',columns = 'Year',fill_value = 'none')
```

Out[193]:

	Budget(MS)			Ratings %		
Year	**2008**	**2009**	**2010**	**2008**	**2009**	**2010**
Genre						
Action	30.0	200.0	20.0	52.5	63.0	52.0
Adventure	none	18.0	105.0	none	84.0	44.0
Comedy	8.0	30.0	20.0	81.0	71.0	70.0

In [194…
```
df.pivot_table(index = 'Genre',columns = 'Year',fill_value = 0,margins = True)
```

Out[194]:

	Budget(MS)				Ratings %			
Year	**2008**	**2009**	**2010**	**All**	**2008**	**2009**	**2010**	**All**
Genre								
Action	30.000000	200.000000	20.000000	70.000000	52.5	63.000000	52.000000	55.000000
Adventure	0.000000	18.000000	105.000000	61.500000	0.0	84.000000	44.000000	64.000000
Comedy	8.000000	30.000000	20.000000	19.333333	81.0	71.000000	70.000000	74.000000
All	22.666667	82.666667	48.333333	51.222222	62.0	72.666667	55.333333	63.333333

In [195…
```
df.pivot_table(index = 'Genre',columns = 'Year',aggfunc = 'sum',fill_value = 0
```

Out[195]:

	Budget(MS)				Ratings %			
Year	**2008**	**2009**	**2010**	**All**	**2008**	**2009**	**2010**	**All**
Genre								
Action	60	200	20	280	105	63	52	220
Adventure	0	18	105	123	0	84	44	128
Comedy	8	30	20	58	81	71	70	222
All	68	248	145	461	186	218	166	570

Melt

In [196…
```
df = pd.read_csv('C:\\Users\prasad jadhav\Downloads\Pandas\Movie_ratings.csv')
df
```

Out[196]:

	Genre	Year	Budget(MS)	Ratings %
0	Action	2008	28	48
1	Action	2009	200	63
2	Action	2008	32	57
3	Action	2010	20	52
4	Adventure	2009	18	84
5	Adventure	2010	105	44
6	Comedy	2010	20	70
7	Comedy	2008	8	81
8	Comedy	2009	30	71

In [197…
```
pd.melt(df)
```

Out[197]:

	variable	value
0	Genre	Action
1	Genre	Action
2	Genre	Action
3	Genre	Action
4	Genre	Adventure
5	Genre	Adventure
6	Genre	Comedy
7	Genre	Comedy
8	Genre	Comedy
9	Year	2008
10	Year	2009
11	Year	2008
12	Year	2010
13	Year	2009
14	Year	2010
15	Year	2010
16	Year	2008
17	Year	2009
18	Budget(MS)	28
19	Budget(MS)	200
20	Budget(MS)	32
21	Budget(MS)	20
22	Budget(MS)	18
23	Budget(MS)	105
24	Budget(MS)	20
25	Budget(MS)	8
26	Budget(MS)	30
27	Ratings %	48
28	Ratings %	63
29	Ratings %	57
30	Ratings %	52
31	Ratings %	84
32	Ratings %	44
33	Ratings %	70
34	Ratings %	81
35	Ratings %	71

In [198…
```
pd.melt(df,id_vars = ['Genre'])
```

Out[198]:

	Genre	variable	value
0	Action	Year	2008
1	Action	Year	2009
2	Action	Year	2008
3	Action	Year	2010
4	Adventure	Year	2009
5	Adventure	Year	2010
6	Comedy	Year	2010
7	Comedy	Year	2008
8	Comedy	Year	2009
9	Action	Budget(MS)	28
10	Action	Budget(MS)	200
11	Action	Budget(MS)	32
12	Action	Budget(MS)	20
13	Adventure	Budget(MS)	18
14	Adventure	Budget(MS)	105
15	Comedy	Budget(MS)	20
16	Comedy	Budget(MS)	8
17	Comedy	Budget(MS)	30
18	Action	Ratings %	48
19	Action	Ratings %	63
20	Action	Ratings %	57
21	Action	Ratings %	52
22	Adventure	Ratings %	84
23	Adventure	Ratings %	44
24	Comedy	Ratings %	70
25	Comedy	Ratings %	81
26	Comedy	Ratings %	71

In [199…
```
pd.melt(df,id_vars = ['Year'],value_vars = ['Genre'])
```

Out[199]:

	Year	variable	value
0	2008	Genre	Action
1	2009	Genre	Action
2	2008	Genre	Action
3	2010	Genre	Action
4	2009	Genre	Adventure
5	2010	Genre	Adventure
6	2010	Genre	Comedy
7	2008	Genre	Comedy
8	2009	Genre	Comedy

In [200…
```
pd.melt(df,id_vars = ['Year'],value_vars = ['Ratings %'])
```

Out[200]:

	Year	variable	value
0	2008	Ratings %	48
1	2009	Ratings %	63
2	2008	Ratings %	57
3	2010	Ratings %	52
4	2009	Ratings %	84
5	2010	Ratings %	44
6	2010	Ratings %	70
7	2008	Ratings %	81
8	2009	Ratings %	71

In [201…
```
pd.melt(df,id_vars = ['Year'],value_vars = ['Ratings %'],var_name = 'Category'
```

Out[201]:

	Year	Category	Data
0	2008	Ratings %	48
1	2009	Ratings %	63
2	2008	Ratings %	57
3	2010	Ratings %	52
4	2009	Ratings %	84
5	2010	Ratings %	44
6	2010	Ratings %	70
7	2008	Ratings %	81
8	2009	Ratings %	71

DatetimeIndex

In [202…
```
df = pd.read_csv('C:\\Users\prasad jadhav\Downloads\Pandas\VC_Starups.csv', pa
```

In [203…
```
df.dtypes
```

Out[203]:
```
Date              datetime64[ns]
R&D Spend                  int64
Marketing Spend            int64
Profit                     int64
dtype: object
```

In [204…
```
type(df.Date[0])
```

Out[204]:
```
pandas._libs.tslibs.timestamps.Timestamp
```

CampusX

In [394…
```
import numpy as np
import pandas as pd
```

In [206…
```
data = pd.read_csv('C:\\Users\prasad jadhav\Downloads\Pandas\matches.csv')
data.head()
```

Out[206]:

	id	season	city	date	team1	team2	toss_winner	toss_decision	result	dl_app
0	1	2017	Hyderabad	2017-04-05	Sunrisers Hyderabad	Royal Challengers Bangalore	Royal Challengers Bangalore	field	normal	
1	2	2017	Pune	2017-04-06	Mumbai Indians	Rising Pune Supergiant	Rising Pune Supergiant	field	normal	
2	3	2017	Rajkot	2017-04-07	Gujarat Lions	Kolkata Knight Riders	Kolkata Knight Riders	field	normal	
3	4	2017	Indore	2017-04-08	Rising Pune Supergiant	Kings XI Punjab	Kings XI Punjab	field	normal	
4	5	2017	Bangalore	2017-04-08	Royal Challengers Bangalore	Delhi Daredevils	Royal Challengers Bangalore	bat	normal	

In [207…
```
type(data)
```

Out[207]:
```
pandas.core.frame.DataFrame
```

In [208…
```
data.tail()
```

Out[208]:

	id	season	city	date	team1	team2	toss_winner	toss_decision	result	dl_a
631	632	2016	Raipur	2016-05-22	Delhi Daredevils	Royal Challengers Bangalore	Royal Challengers Bangalore	field	normal	
632	633	2016	Bangalore	2016-05-24	Gujarat Lions	Royal Challengers Bangalore	Royal Challengers Bangalore	field	normal	
633	634	2016	Delhi	2016-05-25	Sunrisers Hyderabad	Kolkata Knight Riders	Kolkata Knight Riders	field	normal	
634	635	2016	Delhi	2016-05-27	Gujarat Lions	Sunrisers Hyderabad	Sunrisers Hyderabad	field	normal	
635	636	2016	Bangalore	2016-05-29	Sunrisers Hyderabad	Royal Challengers Bangalore	Sunrisers Hyderabad	bat	normal	

In [209…
```
data.shape #Rows_Columns
```

Out[209]:
```
(636, 18)
```

In [210…
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 636 entries, 0 to 635
Data columns (total 18 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   id               636 non-null    int64  
 1   season           636 non-null    int64  
 2   city             629 non-null    object 
 3   date             636 non-null    object 
 4   team1            636 non-null    object 
 5   team2            636 non-null    object 
 6   toss_winner      636 non-null    object 
 7   toss_decision    636 non-null    object 
 8   result           636 non-null    object 
 9   dl_applied       636 non-null    int64  
 10  winner           633 non-null    object 
 11  win_by_runs      636 non-null    int64  
 12  win_by_wickets   636 non-null    int64  
 13  player_of_match  633 non-null    object 
 14  venue            636 non-null    object 
 15  umpire1          635 non-null    object 
 16  umpire2          635 non-null    object 
 17  umpire3          0 non-null      float64
dtypes: float64(1), int64(5), object(12)
memory usage: 89.6+ KB
```

In [211…
```
data.describe() #Numerical
```

Out[211]:

	id	season	dl_applied	win_by_runs	win_by_wickets	umpire3
count	636.000000	636.000000	636.000000	636.000000	636.000000	0.0
mean	318.500000	2012.490566	0.025157	13.682390	3.372642	NaN
std	183.741666	2.773026	0.156726	23.908877	3.420338	NaN
min	1.000000	2008.000000	0.000000	0.000000	0.000000	NaN
25%	159.750000	2010.000000	0.000000	0.000000	0.000000	NaN
50%	318.500000	2012.000000	0.000000	0.000000	4.000000	NaN
75%	477.250000	2015.000000	0.000000	20.000000	7.000000	NaN
max	636.000000	2017.000000	1.000000	146.000000	10.000000	NaN

In [212…
```
type(data['winner'])
```

Out[212]:
```
pandas.core.series.Series
```

In [213…
```
data[{'team1','team2','winner'}] #Dtype_DF
```

```
C:\Users\prasad jadhav\AppData\Local\Temp\ipykernel_12320\2511332080.py:1: Fut
ureWarning: Passing a set as an indexer is deprecated and will raise in a futu
re version. Use a list instead.
  data[{'team1','team2','winner'}] #Dtype_DF
```

Out[213]:

	team1	team2	winner
0	Sunrisers Hyderabad	Royal Challengers Bangalore	Sunrisers Hyderabad
1	Mumbai Indians	Rising Pune Supergiant	Rising Pune Supergiant
2	Gujarat Lions	Kolkata Knight Riders	Kolkata Knight Riders
3	Rising Pune Supergiant	Kings XI Punjab	Kings XI Punjab
4	Royal Challengers Bangalore	Delhi Daredevils	Royal Challengers Bangalore
...	...	...	...
631	Delhi Daredevils	Royal Challengers Bangalore	Royal Challengers Bangalore
632	Gujarat Lions	Royal Challengers Bangalore	Royal Challengers Bangalore
633	Sunrisers Hyderabad	Kolkata Knight Riders	Sunrisers Hyderabad
634	Gujarat Lions	Sunrisers Hyderabad	Sunrisers Hyderabad
635	Sunrisers Hyderabad	Royal Challengers Bangalore	Sunrisers Hyderabad

636 rows × 3 columns

In [214…
```
data.iloc[0] #Rows
```

Out[214]:
```
id                                                1
season                                         2017
city                                        Hyderabad
date                                       2017-04-05
team1                                Sunrisers Hyderabad
team2                     Royal Challengers Bangalore
toss_winner               Royal Challengers Bangalore
toss_decision                                  field
result                                        normal
dl_applied                                         0
winner                               Sunrisers Hyderabad
win_by_runs                                       35
win_by_wickets                                     0
player_of_match                           Yuvraj Singh
venue             Rajiv Gandhi International Stadium, Uppal
umpire1                                     AY Dandekar
umpire2                                        NJ Llong
umpire3                                             NaN
Name: 0, dtype: object
```

In [215…
```
data.iloc[1:11:2]
```

Out[215]:

	id	season	city	date	team1	team2	toss_winner	toss_decision	result	dl_appl
1	2	2017	Pune	2017-04-06	Mumbai Indians	Rising Pune Supergiant	Rising Pune Supergiant	field	normal	
3	4	2017	Indore	2017-04-08	Rising Pune Supergiant	Kings XI Punjab	Kings XI Punjab	field	normal	
5	6	2017	Hyderabad	2017-04-09	Gujarat Lions	Sunrisers Hyderabad	Sunrisers Hyderabad	field	normal	
7	8	2017	Indore	2017-04-10	Royal Challengers Bangalore	Kings XI Punjab	Royal Challengers Bangalore	bat	normal	
9	10	2017	Mumbai	2017-04-12	Sunrisers Hyderabad	Mumbai Indians	Mumbai Indians	field	normal	

In [216…
```
data.iloc[[1,5,6]]
```

Out[216]:

	id	season	city	date	team1	team2	toss_winner	toss_decision	result	dl_applied
1	2	2017	Pune	2017-04-06	Mumbai Indians	Rising Pune Supergiant	Rising Pune Supergiant	field	normal	0
5	6	2017	Hyderabad	2017-04-09	Gujarat Lions	Sunrisers Hyderabad	Sunrisers Hyderabad	field	normal	0
6	7	2017	Mumbai	2017-04-09	Kolkata Knight Riders	Mumbai Indians	Mumbai Indians	field	normal	0

In [217…
```
data.iloc[:,[4,5,10]]
```

Out[217]:

	team1	team2	winner
0	Sunrisers Hyderabad	Royal Challengers Bangalore	Sunrisers Hyderabad
1	Mumbai Indians	Rising Pune Supergiant	Rising Pune Supergiant
2	Gujarat Lions	Kolkata Knight Riders	Kolkata Knight Riders
3	Rising Pune Supergiant	Kings XI Punjab	Kings XI Punjab
4	Royal Challengers Bangalore	Delhi Daredevils	Royal Challengers Bangalore
...	...	...	...
631	Delhi Daredevils	Royal Challengers Bangalore	Royal Challengers Bangalore
632	Gujarat Lions	Royal Challengers Bangalore	Royal Challengers Bangalore
633	Sunrisers Hyderabad	Kolkata Knight Riders	Sunrisers Hyderabad
634	Gujarat Lions	Sunrisers Hyderabad	Sunrisers Hyderabad
635	Sunrisers Hyderabad	Royal Challengers Bangalore	Sunrisers Hyderabad

636 rows × 3 columns

In [218…
```
mask = data['city'] == 'Pune'
data[mask].shape[0]
```

Out[218]: 32

In [219…
```
def get_city_match_count(city):
    mask = data['city'] == city
    return data[mask].shape[0]
```

In [220…
```
get_city_match_count('Hyderabad')
```

Out[220]: 49

In [221…
```
get_city_match_count('Bangalore')
```

Out[221]: 66

In [222…
```
mask_1 = data['city'] == 'Pune'
mask_2 = data['date'] > '2010-01-01'

data[mask_1 & mask_2].shape[0]
```

Out[222]: 32

In [223…
```
mask_1 = data['city'] == 'Hyderabad'
mask_2 = data['date'] > '2010-01-01'

data[mask_1 & mask_2].shape[0]
```

Out[223]: 42

In [224…
```
mask_1 = data['city'] == 'Hyderabad'
```

```
mask_2 = data['date'] < '2010-01-01'

data[mask_1 & mask_2].shape[0]
```

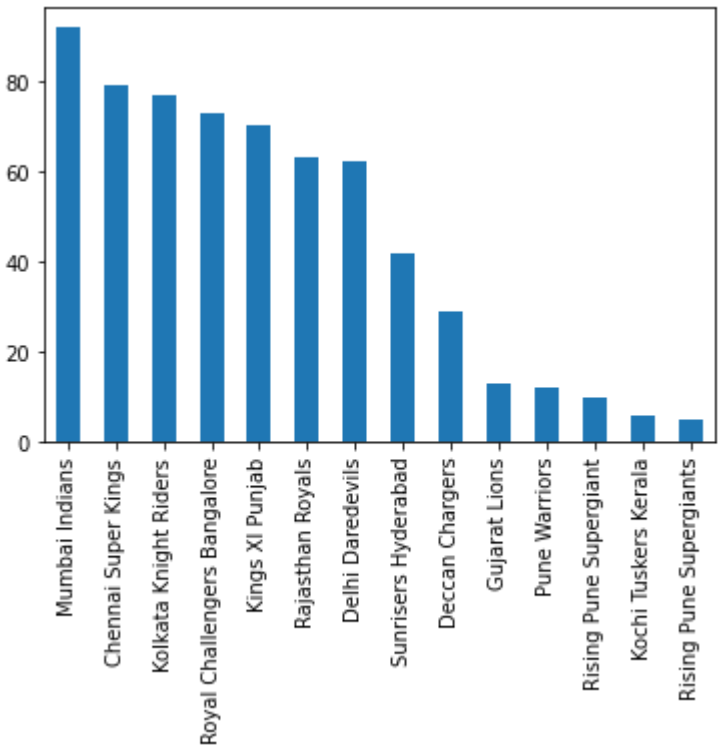
Out[224]:
```
7
```

In [225…
```
data['winner'].value_counts()
```

Out[225]:
```
Mumbai Indians                 92
Chennai Super Kings            79
Kolkata Knight Riders          77
Royal Challengers Bangalore    73
Kings XI Punjab                70
Rajasthan Royals               63
Delhi Daredevils               62
Sunrisers Hyderabad            42
Deccan Chargers                29
Gujarat Lions                  13
Pune Warriors                  12
Rising Pune Supergiant         10
Kochi Tuskers Kerala            6
Rising Pune Supergiants         5
Name: winner, dtype: int64
```

In [226…
```
data['winner'].value_counts().plot(kind = 'bar')
```

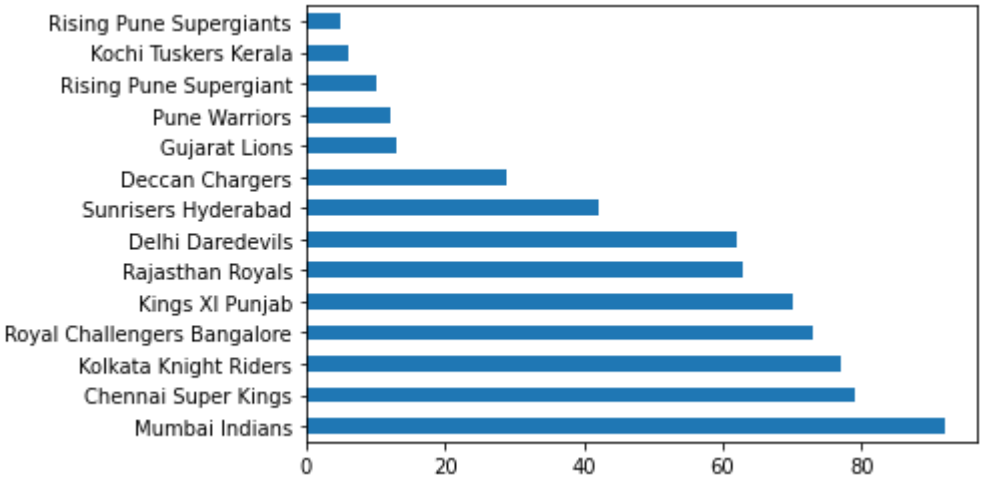
Out[226]:
```
<AxesSubplot:>
```



In [227…
```
# data['winner'].value_counts().head(5).plot(kind = 'bar')
```

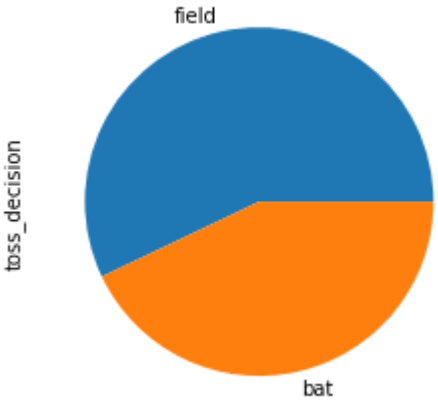
In [228…
```
data['winner'].value_counts().plot(kind = 'barh')
```

Out[228]:
```
<AxesSubplot:>
```



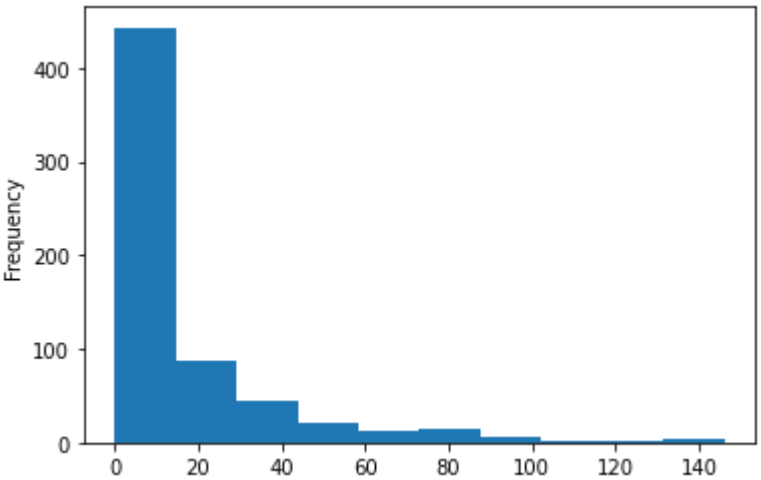
In [229…
```
data['toss_decision'].value_counts().plot(kind = 'pie')
```

Out[229]:
```
<AxesSubplot:ylabel='toss_decision'>
```



In [230…
```
data['win_by_runs'].plot(kind = 'hist')
```

Out[230]:
```
<AxesSubplot:ylabel='Frequency'>
```



Series

In [231…
```
my_sr = data['winner'].value_counts()
```

In [232…]
```
my_sr.index
```

Out[232]:
```
Index(['Mumbai Indians', 'Chennai Super Kings', 'Kolkata Knight Riders',
       'Royal Challengers Bangalore', 'Kings XI Punjab', 'Rajasthan Royals',
       'Delhi Daredevils', 'Sunrisers Hyderabad', 'Deccan Chargers',
       'Gujarat Lions', 'Pune Warriors', 'Rising Pune Supergiant',
       'Kochi Tuskers Kerala', 'Rising Pune Supergiants'],
      dtype='object')
```

In [233…]
```
my_sr.values
```

Out[233]:
```
array([92, 79, 77, 73, 70, 63, 62, 42, 29, 13, 12, 10,  6,  5],
      dtype=int64)
```

In [234…]
```
# my_sr['Pune']
```

In [235…]
```
my_sr['Mumbai Indians']
```

Out[235]:
```
92
```

In [236…]
```
data['team1'].value_counts() + data['team2'].value_counts()
```

Out[236]:
```
Chennai Super Kings            131
Deccan Chargers                 75
Delhi Daredevils               147
Gujarat Lions                   30
Kings XI Punjab                148
Kochi Tuskers Kerala            14
Kolkata Knight Riders          148
Mumbai Indians                 157
Pune Warriors                   46
Rajasthan Royals               118
Rising Pune Supergiant          16
Rising Pune Supergiants         14
Royal Challengers Bangalore    152
Sunrisers Hyderabad             76
dtype: int64
```

In [237…]
```
# (data['team1'].value_counts() + data['team2'].value_counts()).head()
```

Sort Values

In [238…]
```
(data['team1'].value_counts() + data['team2'].value_counts()).sort_values(ascer
```

Out[238]:
```
Mumbai Indians                 157
Royal Challengers Bangalore    152
Kings XI Punjab                148
Kolkata Knight Riders          148
Delhi Daredevils               147
Chennai Super Kings            131
Rajasthan Royals               118
Sunrisers Hyderabad             76
Deccan Chargers                 75
Pune Warriors                   46
Gujarat Lions                   30
Rising Pune Supergiant          16
Kochi Tuskers Kerala            14
Rising Pune Supergiants         14
dtype: int64
```

In [239…
```
# data.sort_values('city',ascending = False)
```

In [240…
```
# data.sort_values(['city','date'], ascending = [True,False])
```

Drop Duplicates

In [241…
```
data.drop_duplicates('city').shape
```

Out[241]:
```
(31, 18)
```

In [242…
```
data.drop_duplicates(subset = ['city','season']).shape
```

Out[242]:
```
(107, 18)
```

In [243…
```
data.drop_duplicates('season',keep = 'last')[['season','winner']].sort_values(
```

Out[243]:

	season	winner
116	2008	Rajasthan Royals
173	2009	Deccan Chargers
233	2010	Chennai Super Kings
306	2011	Chennai Super Kings
380	2012	Kolkata Knight Riders
456	2013	Mumbai Indians
516	2014	Kolkata Knight Riders
575	2015	Mumbai Indians
635	2016	Sunrisers Hyderabad
58	2017	Mumbai Indians

GroupBy

In [244…
```
company = pd.read_csv('C:\\Users\prasad jadhav\Downloads\Pandas\Fortune500.csv
company.head()
```

Out[244]:

	Rank	Title	Website	Employees	Sector	Industry	Hqloc
0	1	Walmart	http://www.walmart.com	2300000	Retailing	General Merchandisers	Bentor
1	2	Berkshire Hathaway	http://www.berkshirehathaway.com	367700	Financials	Insurance: Property and Casualty (Stock)	Omah
2	3	Apple	http://www.apple.com	116000	Technology	Computers, Office Equipment	Cupe
3	4	Exxon Mobil	http://www.exxonmobil.com	72700	Energy	Petroleum Refining	Irvin
4	5	McKesson	http://www.mckesson.com	68000	Wholesalers	Wholesalers: Health Care	Franc

5 rows × 23 columns

In [245…
```
sectors = company.groupby('Sector')
sectors
```

Out[245]:
```
<pandas.core.groupby.generic.DataFrameGroupBy object at 0x000002EA73BA1720>
```

In [246…
```
len(company)
```

Out[246]:
```
500
```

In [247…
```
len(sectors)
```

Out[247]:
```
21
```

In [248…
```
sectors.size().sort_values(ascending = False)
```

Out[248]:
```
Sector
Financials                          84
Energy                              57
Retailing                           47
Technology                          43
Health Care                         38
Wholesalers                         29
Food, Beverages & Tobacco           24
Business Services                   20
Materials                           19
Industrials                         19
Transportation                      17
Chemicals                           14
Engineering & Construction          13
Aerospace & Defense                 12
Household Products                  12
Media                               11
Telecommunications                  10
Hotels, Restaurants & Leisure       10
Motor Vehicles & Parts               9
Food & Drug Stores                   7
Apparel                              5
dtype: int64
```

In [249…
```
# sectors.first()
```

In [250…
```
# sectors.last()
```

In [251…
```
sectors.groups

#company.iloc[23,:]
```

Out[251]:
{'Aerospace & Defense': [23, 49, 55, 89, 113, 115, 199, 227, 272, 379, 392, 491], 'Apparel': [87, 229, 334, 370, 431], 'Business Services': [145, 178, 186, 191, 200, 239, 241, 263, 266, 298, 300, 344, 377, 451, 458, 470, 477, 489, 492, 499], 'Chemicals': [61, 112, 182, 203, 210, 235, 274, 288, 293, 308, 376, 466, 481, 483], 'Energy': [3, 18, 33, 36, 50, 78, 88, 102, 114, 116, 120, 121, 140, 144, 156, 166, 169, 171, 172, 174, 193, 195, 214, 228, 230, 233, 234, 236, 237, 255, 262, 271, 273, 277, 279, 284, 307, 311, 327, 342, 343, 348, 355, 357, 361, 364, 366, 367, 374, 399, 401, 418, 430, 456, 479, 487, 498], 'Engineering & Construction': [148, 160, 231, 258, 259, 323, 352, 354, 359, 414, 445, 493, 496], 'Financials': [1, 19, 20, 24, 25, 29, 32, 38, 41, 47, 54, 64, 67, 74, 75, 76, 77, 79, 83, 85, 96, 98, 99, 101, 119, 124, 125, 152, 165, 167, 176, 184, 188, 206, 209, 212, 213, 217, 221, 226, 238, 244, 245, 254, 257, 261, 265, 270, 276, 285, 292, 301, 302, 314, 315, 328, 341, 353, 356, 360, 381, 388, 390, 397, 402, 404, 410, 427, 435, 436, 438, 447, 450, 454, 459, 461, 463, 468, 471, 474, 475, 476, 478, 482], 'Food & Drug Stores': [16, 17, 48, 84, 90, 157, 175], 'Food, Beverages & Tobacco': [43, 44, 63, 81, 92, 103, 105, 108, 147, 164, 196, 208, 215, 222, 294, 338, 345, 350, 368, 407, 415, 426, 455, 485], 'Health Care': [5, 6, 21, 28, 34, 42, 52, 53, 62, 65, 68, 69, 91, 110, 122, 129, 131, 133, 134, 146, 155, 180, 194, 224, 247, 251, 253, 275, 280, 289, 326, 351, 365, 373, 375, 389, 433, 453], 'Hotels, Restaurants & Leisure': [111, 130, 162, 240, 248, 296, 384, 398, 421, 460], 'Household Products': [35, 154, 181, 207, 249, 252, 310, 371, 417, 443, 452, 473], 'Industrials': [12, 72, 73, 93, 104, 136, 138, 158, 163, 201, 250, 297, 336, 369, 385, 391, 424, 441, 444], 'Materials': [132, 168, 189, 278, 299, 305, 319, 332, 340, 346, 394, 396, 400, 412, 420, 429, 440, 449, 457], 'Media': [51, 94, 100, 192, 223, 329, 331, 387, 411, 425, 490], 'Motor Vehicles & Parts': [7, 9, 150, 183, 282, 304, 321, 382, 446], 'Retailing': [0, 15, 22, 37, 39, 71, 86, 109, 126, 127, 128, 135, 139, 141, 149, 173, 177, 187, 202, 218, 219, 220, 232, 243, 260, 268, 269, 286, 290, 291, 295, 306, 317, 318, 320, 322, 339, 347, 383, 395, 409, 416, 422, 462, 469, 494, 495], 'Technology': [2, 11, 26, 27, 31, 40, 46, 58, 59, 60, 80, 97, 118, 143, 151, 153, 161, 198, 204, 205, 216, 225, 264, 267, 309, 313, 316, 325, 362, 378, 380, 386, 405, 408, 413, 423, 432, 439, 442, 464, 467, 480, 497], 'Telecommunications': [8, 13, 30, 95, 159, 185, 312, 335, 448, 484], 'Transportation': [45, 57, 66, 70, 82, 137, 142, 190, 211, 256, 283, 393, 403, 406, 428, 434, 437], 'Wholesalers': [4, 10, 14, 56, 106, 107, 117, 123, 170, 179, 197, 242, 246, 281, 287, 303, 324, 330, 333, 337, 349, 358, 363, 372, 419, 465, 472, 486, 488]}

In [252…
```
sectors.get_group('Energy').shape
```

Out[252]:
```
(57, 23)
```

In [253…
```
sectors.mean()
```

Out[253]:

Sector	Rank	Employees	Hqzip	Revenues	Revchange	Pro
Aerospace & Defense	201.333333	71986.666667	25020.500000	28191.833333	-3.208333	2093.3083
Apparel	291.200000	50930.000000	34311.200000	13250.800000	1.940000	1263.7000
Business Services	329.200000	42252.500000	45069.200000	9734.250000	5.590000	1155.3550
Chemicals	284.714286	26064.714286	42833.142857	13476.214286	2.228571	1137.0214
Energy	249.421053	15254.578947	59845.789474	20638.894737	-7.792982	6.5017
Engineering & Construction	331.153846	25314.615385	51807.692308	9590.923077	13.369231	390.1692
Financials	247.797619	37642.833333	34289.547619	24614.369048	2.573810	2719.7761
Food & Drug Stores	84.857143	198585.000000	53398.857143	55669.428571	22.271429	1217.4285
Food, Beverages & Tobacco	237.416667	41679.541667	44271.833333	19700.250000	5.404167	2346.1833
Health Care	182.947368	61616.394737	37741.552632	34694.447368	10.223684	2773.2605
Hotels, Restaurants & Leisure	286.000000	183874.200000	53492.200000	12118.900000	-1.950000	1451.0600
Household Products	296.333333	40278.666667	36486.500000	15070.500000	8.775000	1650.3083
Industrials	237.157895	64436.631579	44082.684211	22018.894737	-5.036842	1727.6894
Materials	343.368421	20280.000000	51266.789474	8934.473684	8.042105	272.4684
Media	276.727273	34431.454545	42882.636364	16433.363636	-1.145455	1821.3363
Motor Vehicles & Parts	232.555556	91025.000000	53697.555556	43607.333333	11.300000	1919.5333
Retailing	242.851064	123827.382979	46602.000000	29431.510638	2.485106	991.7851
Technology	244.744186	70915.069767	68534.069767	29246.255814	7.162791	4137.2418
Telecommunications	207.900000	79324.900000	42375.700000	45997.800000	28.380000	4127.2800
Transportation	238.588235	78305.588235	50787.705882	20866.647059	5.476471	1670.2941
Wholesalers	260.931034	21753.172414	49092.620690	27566.068966	3.834483	391.2793

In [254…

```
sectors['Revenues'].mean().sort_values(ascending = False)
```

Out[254]:
```
Sector
Food & Drug Stores                55669.428571
Telecommunications                45997.800000
Motor Vehicles & Parts            43607.333333
Health Care                       34694.447368
Retailing                         29431.510638
Technology                        29246.255814
Aerospace & Defense               28191.833333
Wholesalers                       27566.068966
Financials                        24614.369048
Industrials                       22018.894737
Transportation                    20866.647059
Energy                            20638.894737
Food, Beverages & Tobacco         19700.250000
Media                             16433.363636
Household Products                15070.500000
Chemicals                         13476.214286
Apparel                           13250.800000
Hotels, Restaurants & Leisure     12118.900000
Business Services                  9734.250000
Engineering & Construction         9590.923077
Materials                          8934.473684
Name: Revenues, dtype: float64
```

In [255…
```
delivery = pd.read_csv('C:\\Users\prasad jadhav\Downloads\Pandas\deliveries.csv
delivery.head()
```

Out[255]:

	match_id	inning	batting_team	bowling_team	over	ball	batsman	non_striker	bowler	is_super_o
0	1	1	Sunrisers Hyderabad	Royal Challengers Bangalore	1	1	DA Warner	S Dhawan	TS Mills	
1	1	1	Sunrisers Hyderabad	Royal Challengers Bangalore	1	2	DA Warner	S Dhawan	TS Mills	
2	1	1	Sunrisers Hyderabad	Royal Challengers Bangalore	1	3	DA Warner	S Dhawan	TS Mills	
3	1	1	Sunrisers Hyderabad	Royal Challengers Bangalore	1	4	DA Warner	S Dhawan	TS Mills	
4	1	1	Sunrisers Hyderabad	Royal Challengers Bangalore	1	5	DA Warner	S Dhawan	TS Mills	

5 rows × 21 columns

In [256…
```
runs = delivery.groupby('batsman')
```

In [257…
```
# runs.get_group('V Kohli').shape
```

In [258…
```
runs['batsman_runs'].sum().sort_values(ascending = False).head(10)
```

Out[258]:
```
batsman
SK Raina          4548
V Kohli           4423
RG Sharma         4207
G Gambhir         4132
DA Warner         4014
RV Uthappa        3778
CH Gayle          3651
S Dhawan          3561
MS Dhoni          3560
AB de Villiers    3486
Name: batsman_runs, dtype: int64
```

In [259…
```
mask = delivery['batsman_runs'] == 4 #6
new_delivery = delivery[mask]
```

In [260…
```
new_delivery.shape[0]
```

Out[260]:
```
17033
```

In [261…
```
mask = delivery['batsman_runs'] == 3
new_delivery = delivery[mask]
```

In [262…
```
new_delivery.shape[0]
```

Out[262]:
```
509
```

In [263…
```
new_delivery.groupby('batsman')['batsman_runs'].count().sort_values(ascending =
```

Out[263]:
```
batsman
M Vijay      17
S Dhawan     16
G Gambhir    15
AM Rahane    14
DA Warner    13
Name: batsman_runs, dtype: int64
```

In [264…
```
# new_delivery.groupby('batsman')['batsman_runs'].size().sort_values(ascending
```

In [271…
```
vk_mask = delivery[delivery['batsman'] == 'V Kohli']
```

In [272…
```
vk_mask.groupby('bowling_team')['batsman_runs'].sum().sort_values(ascending = F
```

Out[272]:
```
bowling_team
Chennai Super Kings    706
Delhi Daredevils       661
Kings XI Punjab        483
Name: batsman_runs, dtype: int64
```

In [273…
```
def run_scored(batsman_name):
    vk_mask = delivery[delivery['batsman'] == batsman_name]
    return vk_mask.groupby('bowling_team')['batsman_runs'].sum().sort_values(as
```

In [274…
```
run_scored('V Kohli')
```

Out[274]:
```
'Chennai Super Kings'
```

In [275…
```
run_scored('MS Dhoni')
```

Out[275]:
```
'Royal Challengers Bangalore'
```

IsIn

In [282…
```
death_over = delivery[delivery['over'] > 15]
```

In [296…
```
all_batsman = death_over.groupby('batsman')['batsman_runs'].count()
bt = all_batsman > 200
batsman_list = all_batsman[bt].index.tolist()
```

In [301…
```
final = delivery[delivery['batsman'].isin(batsman_list)]
```

In [302…
```
runs = final.groupby('batsman')['batsman_runs'].sum()
```

In [303…
```
balls = final.groupby('batsman')['batsman_runs'].count()
```

In [304…
```
# sr = (runs/balls)*100
```

Merge

In [307…
```
match = pd.read_csv('C:\\Users\prasad jadhav\Downloads\Pandas\matches.csv')
match.head()
```

Out[307]:

	id	season	city	date	team1	team2	toss_winner	toss_decision	result	dl_app
0	1	2017	Hyderabad	2017-04-05	Sunrisers Hyderabad	Royal Challengers Bangalore	Royal Challengers Bangalore	field	normal	
1	2	2017	Pune	2017-04-06	Mumbai Indians	Rising Pune Supergiant	Rising Pune Supergiant	field	normal	
2	3	2017	Rajkot	2017-04-07	Gujarat Lions	Kolkata Knight Riders	Kolkata Knight Riders	field	normal	
3	4	2017	Indore	2017-04-08	Rising Pune Supergiant	Kings XI Punjab	Kings XI Punjab	field	normal	
4	5	2017	Bangalore	2017-04-08	Royal Challengers Bangalore	Delhi Daredevils	Royal Challengers Bangalore	bat	normal	

In [316…
```
new_1 = delivery.merge(match,left_on = 'match_id',right_on = 'id')
```

In [314…
```
print(delivery.shape)
print(match.shape)
```

```
(150460, 21)
(636, 18)
```

In [326…
```
new_1.groupby(['season','batsman'])['batsman_runs'].sum().sort_values(ascendin
```

Out[326]:

	season	batsman
10	2008	SE Marsh
14	2009	ML Hayden
9	2010	SR Tendulkar
11	2011	CH Gayle
3	2012	CH Gayle
2	2013	MEK Hussey
6	2014	RV Uthappa
17	2015	DA Warner
0	2016	V Kohli
7	2017	DA Warner

Pivot Table

In [343…
```
food = pd.read_csv('C:\\Users\prasad jadhav\Downloads\Pandas\food.csv')
food
```

Out[343]:

	Name	Gender	City	Frequency	Item	Spends
0	Nitish	M	Kolkata	Weekly	Burger	11
1	Anu	F	Gurgaon	Daily	Sandwich	14
2	Mukku	M	Kolkata	Once	Vada	25
3	Suri	M	Kolkata	Monthly	Pizza	56
4	Rajiv	M	Patna	Never	Paneer	34

In [349…
```
food.pivot_table(index = ['City','Gender'],columns = ['Item','Frequency'],value
```

Out[349]:

	Item	Burger	Paneer	Pizza	Sandwich	Vada
	Frequency	Weekly	Never	Monthly	Daily	Once
City	Gender					
Gurgaon	**F**	NaN	NaN	NaN	14.0	NaN
Kolkata	**M**	11.0	NaN	56.0	NaN	25.0
Patna	**M**	NaN	34.0	NaN	NaN	NaN

In [351…
```
mask = delivery['batsman_runs'] == 6
six = delivery[mask]
six.shape
```

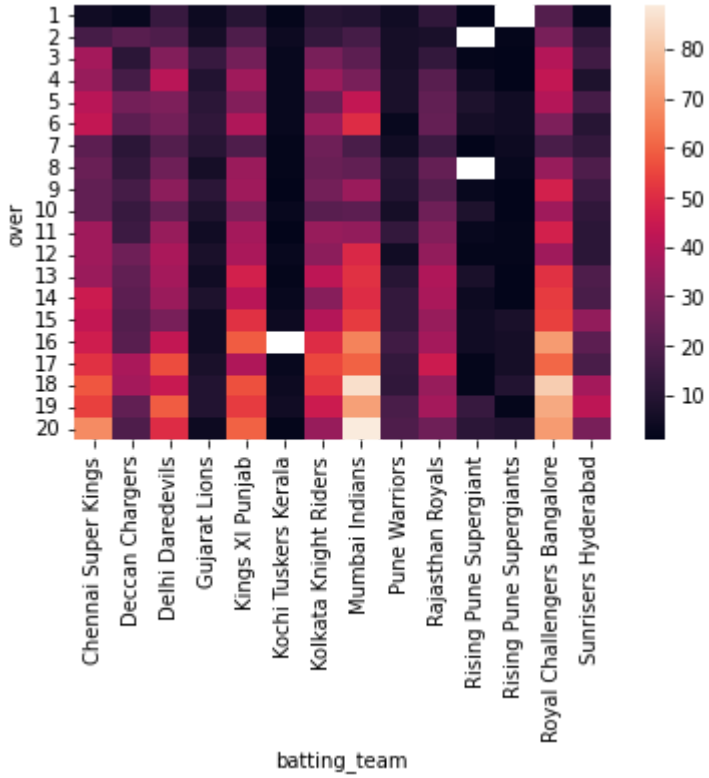
Out[351]:
```
(6523, 21)
```

In [354…
```
pvt = six.pivot_table(index = 'over',columns = 'batting_team',values = 'batsma
```

In [353…
```
import seaborn as sns
```

In [356…]
```
sns.heatmap(pvt)
```

Out[356]:
```
<AxesSubplot:xlabel='batting_team', ylabel='over'>
```



Corr

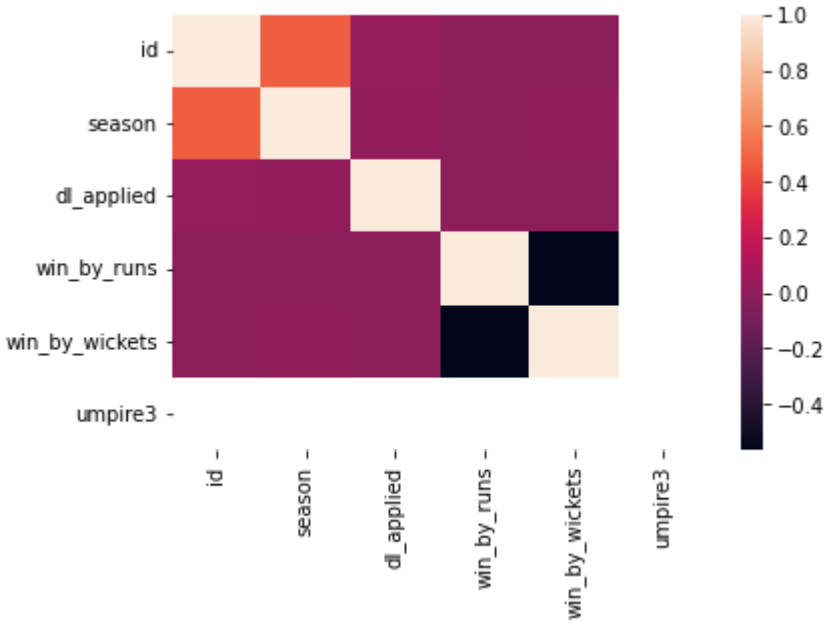
In [357…]
```
match.corr()
```

Out[357]:

	id	season	dl_applied	win_by_runs	win_by_wickets	umpire3
id	1.000000	0.471087	0.024281	-0.010263	-0.015510	NaN
season	0.471087	1.000000	0.004170	-0.016815	-0.000708	NaN
dl_applied	0.024281	0.004170	1.000000	-0.010893	-0.011640	NaN
win_by_runs	-0.010263	-0.016815	-0.010893	1.000000	-0.565181	NaN
win_by_wickets	-0.015510	-0.000708	-0.011640	-0.565181	1.000000	NaN
umpire3	NaN	NaN	NaN	NaN	NaN	NaN

In [358…]
```
sns.heatmap(match.corr())
```

Out[358]:
```
<AxesSubplot:>
```



In [364…
```
#Rename_Cols

match.rename(columns = {'city': 'place','date': 'dom'}).head(3)
```

Out[364]:

	id	season	place	dom	team1	team2	toss_winner	toss_decision	result	dl_appli
0	1	2017	Hyderabad	2017-04-05	Sunrisers Hyderabad	Royal Challengers Bangalore	Royal Challengers Bangalore	field	normal	
1	2	2017	Pune	2017-04-06	Mumbai Indians	Rising Pune Supergiant	Rising Pune Supergiant	field	normal	
2	3	2017	Rajkot	2017-04-07	Gujarat Lions	Kolkata Knight Riders	Kolkata Knight Riders	field	normal	

Set_Index and Reset_Index

In [379…
```
data.head(1)
```

Out[379]:

	season	city	date	team1	team2	toss_winner	toss_decision	result	dl_applied
id									
1	2017	Hyderabad	2017-04-05	Sunrisers Hyderabad	Royal Challengers Bangalore	Royal Challengers Bangalore	field	normal	0

In [388…
```
# data.set_index()
```

In [387…
```
# data.reset_index()
```

In [393…
```
data['winner'].value_counts().reset_index()
```

Out[393]:

	index	winner
0	Mumbai Indians	92
1	Chennai Super Kings	79
2	Kolkata Knight Riders	77
3	Royal Challengers Bangalore	73
4	Kings XI Punjab	70
5	Rajasthan Royals	63
6	Delhi Daredevils	62
7	Sunrisers Hyderabad	42
8	Deccan Chargers	29
9	Gujarat Lions	13
10	Pune Warriors	12
11	Rising Pune Supergiant	10
12	Kochi Tuskers Kerala	6
13	Rising Pune Supergiants	5

Handling Missing Values

In [395…

```
data_1 = pd.read_csv('train.csv')
data_1.head()
```

Out[395]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN

In [403…

```
data_1.dropna(subset = ['Cabin','Embarked']).shape
```

Out[403]:

```
(202, 12)
```

In [405…]
```
data_1.fillna(0).head()
```

Out[405]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	0
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	0
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	0

In [416…]
```
data_1['Cabin'].fillna('Not Specified').to_frame()
```

Out[416]:

	Cabin
0	Not Specified
1	C85
2	Not Specified
3	C123
4	Not Specified
...	...
886	Not Specified
887	B42
888	Not Specified
889	C148
890	Not Specified

891 rows × 1 columns

In [415…]
```
data_1['Age'].fillna(30).to_frame()
```

Out[415]:

	Age
0	22.0
1	38.0
2	26.0
3	35.0
4	35.0
...	...
886	27.0
887	19.0
888	30.0
889	26.0
890	32.0

891 rows × 1 columns

In [419…

```
# data_1['Age'].fillna(method = 'bfill')
```

Thank You

CONNECT WITH ME :

LinkedIn GitHub kaggle Medium

PRASADMJADHAV2