In [ ]:

```
"""
date should be of datetime format
date column should be index
"""
```

In [4]:

```
import pandas as pd


df=pd.read_csv("/home/harshit/DataSets/YESBANK.NS.csv", parse_dates=['Date'], index_col='Date')
df.head(5)
```

Out[4]:

	Open	High	Low	Close	Adj Close	Volume
Date						
2017-12-11	313.500000	315.799988	310.600006	311.600006	300.880615	4416465.0
2017-12-12	312.000000	312.000000	305.899994	306.799988	296.245758	5457103.0
2017-12-13	306.350006	307.350006	301.049988	301.899994	291.514282	6911856.0
2017-12-14	303.899994	304.649994	301.750000	303.899994	293.445526	4904177.0
2017-12-15	307.000000	317.450012	307.000000	315.899994	305.032715	20571225.0

In [5]:

```
df.shape
```

Out[5]:

```
(738, 6)
```

In [6]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 738 entries, 2017-12-11 to 2020-12-08
Data columns (total 6 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   Open       736 non-null    float64
 1   High       736 non-null    float64
 2   Low        736 non-null    float64
 3   Close      736 non-null    float64
 4   Adj Close  736 non-null    float64
 5   Volume     736 non-null    float64
dtypes: float64(6)
memory usage: 40.4 KB
```

In [7]:

```
df.loc[   ['2017-12-13']    ]
```

Out[7]:

	Open	High	Low	Close	Adj Close	Volume
Date						
2017-12-13	306.350006	307.350006	301.049988	301.899994	291.514282	6911856.0

In [10]:

```
df.loc[ ['2017-12-15'], : ]
```

Out[10]:

	Open	High	Low	Close	Adj Close	Volume
Date						
2017-12-15	307.0	317.450012	307.0	315.899994	305.032715	20571225.0

In [11]:

```
df.loc[ ['2017-12-15', '2017-12-12'] , : ]
```

Out[11]:

	Open	High	Low	Close	Adj Close	Volume
Date						
2017-12-15	307.0	317.450012	307.000000	315.899994	305.032715	20571225.0
2017-12-12	312.0	312.000000	305.899994	306.799988	296.245758	5457103.0

In [12]:

```
pd.__version__
```

Out[12]:

```
'1.1.4'
```

Out[12]: 1.1.4

In [13]:
```
df.loc[ '2017-12-15' :  '2017-12-25' ]
```

Out[13]:

	Open	High	Low	Close	Adj Close	Volume
Date						
2017-12-15	307.000000	317.450012	307.000000	315.899994	305.032715	20571225.0
2017-12-18	315.899994	318.000000	305.299988	311.149994	300.446106	13690080.0
2017-12-19	314.000000	314.399994	309.549988	312.250000	301.508270	6290291.0
2017-12-20	312.399994	313.399994	310.000000	311.700012	300.977203	6605913.0
2017-12-21	311.200012	313.950012	309.399994	310.450012	299.770203	7130623.0
2017-12-22	310.500000	313.500000	308.700012	310.149994	299.480469	7895699.0

In [15]:
```
#3 ways to use date as index


df.loc[  '2018-09-27' ]
```

Out[15]:
```
Open         2.260000e+02
High         2.270000e+02
Low          2.020500e+02
Close        2.032500e+02
Adj Close    1.978165e+02
Volume       9.119720e+07
Name: 2018-09-27 00:00:00, dtype: float64
```

In [16]:
```
df.loc[  ['2018-09-27'] ]
```

Out[16]:

	Open	High	Low	Close	Adj Close	Volume
Date						
2018-09-27	226.0	227.0	202.050003	203.25	197.816498	91197198.0

In [21]:
```
df.loc[  ['2018-09-27','2017-12-13'] ]
```

Out[21]:

	Open	High	Low	Close	Adj Close	Volume
Date						
2018-09-27	226.000000	227.000000	202.050003	203.250000	197.816498	91197198.0
2017-12-13	306.350006	307.350006	301.049988	301.899994	291.514282	6911856.0

In [23]:
```
#records for january of 2018


df.loc[ '2018-01' ]
```

Out[23]:

	Open	High	Low	Close	Adj Close	Volume
Date						
2018-01-01	315.500000	317.750000	311.299988	312.600006	301.846252	4019878.0
2018-01-02	313.399994	314.000000	307.149994	311.649994	300.928894	5224976.0
2018-01-03	312.000000	316.500000	311.149994	315.850006	304.984436	5672263.0
2018-01-04	316.000000	318.399994	313.000000	317.100006	306.191437	5667580.0
2018-01-05	317.500000	337.899994	317.450012	332.850006	321.399628	30720675.0
2018-01-08	336.000000	341.299988	331.299988	333.600006	322.123779	12747890.0
2018-01-09	334.899994	342.799988	327.549988	341.350006	329.607208	13282560.0
2018-01-10	341.500000	342.350006	335.450012	339.799988	328.110474	10385044.0
2018-01-11	339.000000	344.250000	335.299988	343.149994	331.345276	8266126.0
2018-01-12	344.100006	344.700012	337.549988	340.899994	329.172668	5688676.0
2018-01-15	341.899994	343.700012	335.100006	336.000000	324.441223	7142164.0

2018-01-16	336.000000	338.750000	328.000000	334.850006	323.330811	7296505.0
2018-01-17	335.100006	343.500000	331.399994	342.399994	330.621063	7985222.0
2018-01-18	350.000000	356.899994	332.350006	341.200012	329.462372	35465087.0
2018-01-19	347.500000	352.250000	339.100006	349.350006	337.332001	21425789.0
2018-01-22	349.950012	358.250000	348.750000	355.350006	343.125580	13456538.0
2018-01-23	359.850006	360.399994	352.299988	359.549988	347.181091	10196645.0
2018-01-24	357.000000	366.299988	356.000000	364.799988	352.250488	11258771.0
2018-01-25	364.500000	364.500000	355.649994	361.600006	349.160583	8963188.0
2018-01-29	361.200012	363.700012	355.549988	358.000000	345.684387	7931235.0
2018-01-30	358.000000	360.799988	351.850006	353.350006	341.194397	7890491.0
2018-01-31	353.000000	356.549988	350.450012	354.399994	342.208252	8527044.0

In [24]:

```
#records between january and march 2018


df.loc[ '2018-01': '2018-03' ]
```

Out[24]:

	Open	High	Low	Close	Adj Close	Volume
Date						
2018-01-01	315.500000	317.750000	311.299988	312.600006	301.846252	4019878.0
2018-01-02	313.399994	314.000000	307.149994	311.649994	300.928894	5224976.0
2018-01-03	312.000000	316.500000	311.149994	315.850006	304.984436	5672263.0
2018-01-04	316.000000	318.399994	313.000000	317.100006	306.191437	5667580.0
2018-01-05	317.500000	337.899994	317.450012	332.850006	321.399628	30720675.0
2018-01-08	336.000000	341.299988	331.299988	333.600006	322.123779	12747890.0
2018-01-09	334.899994	342.799988	327.549988	341.350006	329.607208	13282560.0
2018-01-10	341.500000	342.350006	335.450012	339.799988	328.110474	10385044.0
2018-01-11	339.000000	344.250000	335.299988	343.149994	331.345276	8266126.0
2018-01-12	344.100006	344.700012	337.549988	340.899994	329.172668	5688676.0
2018-01-15	341.899994	343.700012	335.100006	336.000000	324.441223	7142164.0
2018-01-16	336.000000	338.750000	328.000000	334.850006	323.330811	7296505.0
2018-01-17	335.100006	343.500000	331.399994	342.399994	330.621063	7985222.0
2018-01-18	350.000000	356.899994	332.350006	341.200012	329.462372	35465087.0
2018-01-19	347.500000	352.250000	339.100006	349.350006	337.332001	21425789.0
2018-01-22	349.950012	358.250000	348.750000	355.350006	343.125580	13456538.0
2018-01-23	359.850006	360.399994	352.299988	359.549988	347.181091	10196645.0
2018-01-24	357.000000	366.299988	356.000000	364.799988	352.250488	11258771.0
2018-01-25	364.500000	364.500000	355.649994	361.600006	349.160583	8963188.0
2018-01-29	361.200012	363.700012	355.549988	358.000000	345.684387	7931235.0
2018-01-30	358.000000	360.799988	351.850006	353.350006	341.194397	7890491.0
2018-01-31	353.000000	356.549988	350.450012	354.399994	342.208252	8527044.0
2018-02-01	355.000000	367.250000	352.649994	359.899994	347.519073	15217926.0
2018-02-02	354.200012	356.000000	341.799988	349.049988	337.042297	16298953.0
2018-02-05	340.000000	349.000000	333.600006	343.600006	331.779816	13407059.0
2018-02-06	325.000000	342.899994	324.000000	338.750000	327.096649	12557261.0
2018-02-07	344.000000	344.000000	330.600006	332.899994	321.447876	11681640.0
2018-02-08	332.899994	340.350006	331.500000	335.000000	323.475647	7785799.0
2018-02-09	330.000000	331.450012	324.000000	325.549988	314.350739	9395513.0
2018-02-12	326.600006	337.200012	326.600006	335.399994	323.861877	12049356.0
2018-02-14	336.000000	337.850006	318.950012	320.350006	309.329620	13548524.0
2018-02-15	321.200012	328.799988	317.700012	319.799988	308.798523	15482667.0
2018-02-16	324.000000	325.000000	309.649994	311.799988	301.073761	18611798.0
2018-02-19	313.850006	315.000000	307.549988	312.049988	301.315125	9311433.0

2018-02-20	314.350006	316.100006	307.500000	308.700012	298.080414	11389041.0
2018-02-21	311.000000	313.899994	304.500000	312.350006	301.604858	13342678.0
2018-02-22	311.899994	316.899994	308.149994	316.100006	305.225830	14787238.0
2018-02-23	316.450012	326.000000	316.450012	323.450012	312.322998	11399732.0
2018-02-26	326.500000	328.200012	319.350006	326.149994	314.930084	9225197.0
2018-02-27	325.299988	334.250000	325.000000	327.149994	315.895691	15104405.0
2018-02-28	323.000000	325.200012	318.049988	322.299988	311.212524	10985771.0
2018-03-01	322.100006	326.000000	318.350006	321.049988	310.005524	7333939.0
2018-03-05	318.500000	319.000000	311.149994	312.950012	302.184204	7973146.0
2018-03-06	317.000000	320.500000	309.850006	312.149994	301.411713	8299126.0
2018-03-07	312.000000	314.899994	308.000000	311.950012	301.218597	8420190.0
2018-03-08	312.299988	313.250000	294.700012	308.549988	297.935547	24027679.0
2018-03-09	310.000000	310.950012	301.299988	303.250000	292.817871	11596136.0
2018-03-12	305.200012	314.500000	301.350006	311.149994	300.446106	12860205.0
2018-03-13	310.000000	315.399994	309.799988	312.799988	302.039307	11436348.0
2018-03-14	311.799988	321.899994	308.200012	318.850006	307.881226	12905495.0
2018-03-15	318.950012	321.000000	310.850006	311.850006	301.122040	9999620.0
2018-03-16	312.399994	316.850006	310.000000	312.899994	302.135895	17094363.0
2018-03-19	316.000000	316.500000	302.500000	304.799988	294.314575	11267355.0
2018-03-20	303.000000	305.799988	299.700012	302.399994	291.997131	9874137.0
2018-03-21	305.000000	309.049988	300.049988	300.750000	290.403870	13383435.0
2018-03-22	301.799988	303.700012	295.750000	298.250000	287.989868	19356591.0
2018-03-23	293.000000	293.049988	285.000000	286.649994	276.788910	21617995.0
2018-03-26	286.500000	304.899994	286.000000	303.350006	292.914459	24240033.0
2018-03-27	307.000000	309.250000	300.700012	303.500000	293.059265	15267419.0
2018-03-28	300.149994	307.500000	299.100006	304.850006	294.362823	14952643.0

In [25]:

```
#records for 2019


df.loc['2019']
```

Out[25]:

	Open	High	Low	Close	Adj Close	Volume
Date						
2019-01-01	182.600006	185.899994	181.000000	184.250000	179.324417	24160878.0
2019-01-02	183.449997	187.000000	182.500000	184.649994	179.713730	32583205.0
2019-01-03	185.250000	186.000000	183.500000	184.100006	179.178436	20239949.0
2019-01-04	184.850006	190.300003	181.550003	189.649994	184.580063	45914917.0
2019-01-07	193.899994	194.399994	185.800003	187.149994	182.146896	40515242.0
...	...	...	...	...	...	...
2019-12-24	50.150002	52.000000	50.049999	51.200001	51.200001	242779981.0
2019-12-26	51.000000	51.599998	48.250000	48.650002	48.650002	216380349.0
2019-12-27	49.500000	49.750000	47.650002	48.000000	48.000000	154956583.0
2019-12-30	47.900002	48.950001	46.650002	47.349998	47.349998	152510288.0
2019-12-31	47.299999	48.049999	46.349998	46.950001	46.950001	141422188.0

243 rows × 6 columns

In [27]:

```
df.loc['2019'][['Open']]
```

Out[27]:

	Open
Date	
2019-01-01	182.600006
2019-01-02	183.449997

2019-01-03	185.250000
2019-01-04	184.850006
2019-01-07	193.899994
...	...
2019-12-24	50.150002
2019-12-26	51.000000
2019-12-27	49.500000
2019-12-30	47.900002
2019-12-31	47.299999

243 rows × 1 columns

In [28]:
```
#average opening price in 2019
df.loc['2019'][['Open']].mean()
```

Out[28]:
```
Open    130.694628
dtype: float64
```

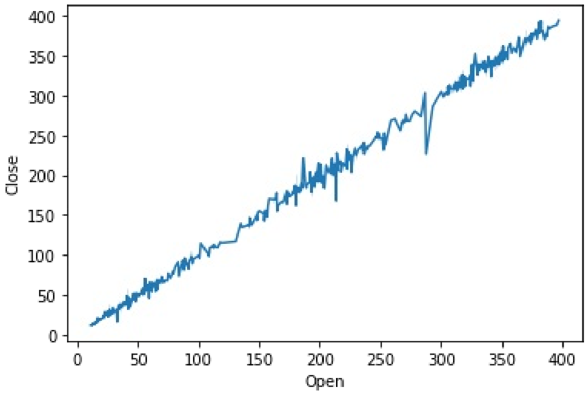
In [29]:
```
df.loc['2019'][['Open']].agg(['min','max'])
```

Out[29]:

	Open
min	35.200001
max	283.500000

In [35]:
```
import seaborn as sns

sns.lineplot(x='Open',y='Close',data=df)
```

Out[35]:
```
<AxesSubplot:xlabel='Open', ylabel='Close'>
```



In [37]:
```
df.loc['2018'][['Adj Close']].mean()
```

Out[37]:
```
Adj Close    292.377269
dtype: float64
```

In [38]:
```
df.loc['2019'][['Adj Close']].mean()
```

Out[38]:
```
Adj Close    127.361017
dtype: float64
```

In [39]:
```
df.loc['2020'][['Adj Close']].mean()
```

Out[39]:
```
Adj Close    24.277564
dtype: float64
```

In [ ]:

```
#resampling means changing the frequency of data


#my current --->daily basis data
```

In [41]:

```
df.resample('M').mean() #average opening price of yesbank share in january 2018-->340.177
```

Out[41]:

Date	Open	High	Low	Close	Adj Close	Volume
2017-12-31	311.528571	314.225000	308.385712	311.078570	300.377130	8.193004e+06
2018-01-31	340.177274	344.797725	334.713634	340.895455	329.168286	1.132792e+07
2018-02-28	327.960528	333.439476	321.978948	327.386839	316.124390	1.271484e+07
2018-03-31	308.563157	312.842104	302.755265	307.473683	296.896260	1.378452e+07
2018-04-30	318.330954	324.219051	313.407144	319.123807	308.145602	1.783510e+07
2018-05-31	345.490910	349.952273	339.484092	343.768182	331.942187	9.989732e+06
2018-06-30	336.297620	339.721431	332.349997	335.669050	326.570445	8.075843e+06
2018-07-31	368.772727	374.197728	363.429545	369.111360	359.243873	1.369120e+07
2018-08-31	377.964286	382.923809	371.240474	376.383336	366.321445	1.702048e+07
2018-09-30	298.030555	301.908333	282.663890	288.986109	281.260617	6.394889e+07
2018-10-31	214.142857	223.661905	206.469048	214.519047	208.784291	5.582258e+07
2018-11-30	201.437500	206.057500	194.965001	200.140000	194.789640	6.738196e+07
2018-12-31	178.225000	181.992500	173.760001	178.002500	173.243938	5.471256e+07
2019-01-31	197.308696	202.391303	191.093478	195.954348	190.715881	5.996651e+07
2019-02-28	201.615790	207.344737	196.734213	202.921053	197.496346	6.611809e+07
2019-03-31	244.713888	249.150001	241.675000	246.150002	239.569654	3.511142e+07
2019-04-30	260.205264	263.447368	251.610526	254.781578	247.970474	4.279916e+07
2019-05-31	154.370454	158.093180	149.013635	152.640909	148.560345	7.116448e+07
2019-06-30	124.628947	127.639474	119.736842	122.994737	122.889473	9.031750e+07
2019-07-31	94.150000	97.519566	89.997826	93.452174	93.452174	1.442202e+08
2019-08-31	75.467500	77.417500	70.572500	73.200000	73.200000	1.874254e+08
2019-09-30	60.134210	62.355263	57.205263	59.157895	59.157895	2.251040e+08
2019-10-31	46.347368	50.078947	43.531579	47.128948	47.128948	3.876632e+08
2019-11-30	67.615000	70.020001	65.150000	67.110000	67.110000	2.631308e+08
2019-12-31	52.200000	54.083334	49.792857	51.452381	51.452381	2.720934e+08
2020-01-31	42.582609	43.634783	41.323913	42.250000	42.250000	1.577796e+08
2020-02-29	36.794737	37.936842	35.507895	36.447368	36.447368	1.456856e+08
2020-03-31	35.142857	40.392858	28.661905	33.628571	33.628571	2.325265e+08
2020-04-30	26.183333	27.216667	25.191666	26.355555	26.355555	4.250341e+07
2020-05-31	27.478948	28.273684	26.736842	27.410526	27.410526	2.381605e+07
2020-06-30	28.270454	28.895455	27.661363	28.093182	28.093182	1.697237e+07
2020-07-31	19.982609	20.610870	19.286956	19.869565	19.869565	1.651249e+08
2020-08-31	14.730952	14.945238	14.350000	14.614286	14.614286	2.715367e+08
2020-09-30	13.990909	14.172727	13.695455	13.906818	13.906818	1.079332e+08
2020-10-31	13.000000	13.130952	12.788095	12.930952	12.930952	6.548526e+07
2020-11-30	13.744737	13.921053	13.486842	13.744737	13.744737	1.475623e+08
2020-12-31	15.666667	15.975000	15.283333	15.783333	15.783333	2.922877e+08

In [43]:

```
"""
their performance over the 4 quarters in 2019
"""


df.loc['2019'].resample('Q').mean()
```

Out[43]:

Date	Open	High	Low	Close	Adj Close	Volume
2019-03-31	212.894167	217.987500	208.054167	213.219168	207.519160	5.445798e+07

2019-06-30	178.466667	181.811666	172.231666	175.597500	171.911110	6.824725e+07
2019-09-30	77.699194	80.258871	73.682258	76.409678	76.409678	1.829444e+08
2019-12-31	55.485000	58.127500	52.929167	55.302500	55.302500	3.057030e+08

In [44]:

```
"""
their performance over the every 15 days in 2019
"""
df.loc['2019'].resample('15D').mean()
```

Out[44]:

	Open	High	Low	Close	Adj Close	Volume
Date						
2019-01-01	188.209091	192.049998	184.200001	188.918181	183.867812	4.708511e+07
2019-01-16	205.981818	212.727273	198.072726	203.159091	197.728019	7.213623e+07
2019-01-31	182.940001	188.334999	177.205002	183.125001	178.229503	6.786232e+07
2019-02-15	221.654545	227.013637	217.286366	223.268183	217.299534	6.249290e+07
2019-03-02	238.666667	242.655555	234.949999	238.677780	232.297189	3.406046e+07
2019-03-17	252.743748	257.787502	250.187500	255.625002	248.791357	3.543188e+07
2019-04-01	271.781821	274.968181	266.281816	269.649997	262.441416	2.797472e+07
2019-04-16	244.287497	247.606251	231.437502	234.337502	228.072929	6.318277e+07
2019-05-01	164.600000	168.844998	158.534999	162.905000	158.550044	6.908044e+07
2019-05-16	144.868181	148.418180	140.540907	143.750000	139.907118	7.376774e+07
2019-05-31	140.745000	143.550000	134.725001	137.569999	136.974883	8.564852e+07
2019-06-15	111.710001	114.665000	107.475000	110.900001	110.900001	9.229162e+07
2019-06-30	97.570001	99.680000	93.870001	95.825000	95.825000	1.178986e+08
2019-07-15	91.704545	96.136365	87.759091	92.172728	92.172728	1.712769e+08
2019-07-30	87.820000	90.405000	82.255000	85.220000	85.220000	1.430944e+08
2019-08-14	69.465000	71.269999	64.430000	67.170000	67.170000	2.156449e+08
2019-08-29	61.555555	64.344443	59.538889	62.283333	62.283333	1.966145e+08
2019-09-13	59.800000	61.527273	56.418182	58.127273	58.127273	2.332662e+08
2019-09-28	43.068750	44.962500	38.387500	40.937501	40.937501	4.385426e+08
2019-10-13	46.083333	49.205555	44.105556	47.111111	47.111111	3.057833e+08
2019-10-28	64.325000	69.950001	61.985001	66.350000	66.350000	3.702342e+08
2019-11-12	66.949999	68.530000	64.560000	65.745000	65.745000	2.116185e+08
2019-11-27	61.990908	63.727273	58.118182	60.077273	60.077273	3.395012e+08
2019-12-12	47.870001	50.165000	46.725000	48.395001	48.395001	2.436526e+08
2019-12-27	48.233334	48.916667	46.883334	47.433333	47.433333	1.496297e+08

In [45]:

```
"""
their performance every 6 months in 2019
"""
df.loc['2019'].resample('6M').mean()
```

Out[45]:

	Open	High	Low	Close	Adj Close	Volume
Date						
2019-01-31	197.308696	202.391303	191.093478	195.954348	190.715881	5.996651e+07
2019-07-31	175.908333	179.799167	170.766250	174.762084	171.072925	7.750125e+07
2020-01-31	60.414141	62.835859	57.314142	59.658081	59.658081	2.663400e+08

In [48]:

```
#minimum & maximum closing price every quarter of 2018 and 2019


"""
1) Select time frame first
2) Select the frequency
3) Select aggregation operation
"""
```

```
df.loc[  '2018' : '2019' ].resample('Q')[['Close']].agg(['min','max'])
```

Out[48]:

	Close	
	min	max
Date		
2018-03-31	286.649994	364.799988
2018-06-30	305.450012	362.000000
2018-09-30	183.649994	394.000000
2018-12-31	160.449997	248.899994
2019-03-31	172.649994	276.100006
2019-06-30	103.199997	280.299988
2019-09-30	41.400002	109.150002
2019-12-31	32.000000	73.000000

In [51]:

```
df.loc[  '2018' : '2019' ].resample('Q').agg(['min','max']).loc[ : , 'Close']
```

Out[51]:

	min	max
Date		
2018-03-31	286.649994	364.799988
2018-06-30	305.450012	362.000000
2018-09-30	183.649994	394.000000
2018-12-31	160.449997	248.899994
2019-03-31	172.649994	276.100006
2019-06-30	103.199997	280.299988
2019-09-30	41.400002	109.150002
2019-12-31	32.000000	73.000000

In [52]:

```
#find average closing price and minimum and maximum volume value for every month in 2019


df.loc['2019'].resample('M')[['Close','Volume']].agg({'Close':'mean','Volume':['max','min']})
```

Out[52]:

	Close	Volume	
	mean	max	min
Date			
2019-01-31	195.954348	183425643.0	20239949.0
2019-02-28	202.921053	264725005.0	22298518.0
2019-03-31	246.150002	84155309.0	20571726.0
2019-04-30	254.781578	217219923.0	13509220.0
2019-05-31	152.640909	124177861.0	40446604.0
2019-06-30	122.994737	197154833.0	40272408.0
2019-07-31	93.452174	238375009.0	42545442.0
2019-08-31	73.200000	314692568.0	106397716.0
2019-09-30	59.157895	398347296.0	128189225.0
2019-10-31	47.128948	836967454.0	188799876.0
2019-11-30	67.110000	431574597.0	121791625.0
2019-12-31	51.452381	661012673.0	130282558.0

In [ ]:

```
#find the total shares sold in month of january in 2018, 2019 and 2020 on weekly basis(can't reample by week if g
```

In [ ]:

In [ ]:

In [ ]:

In [94]:

```
temp=df.loc['2018':'2020']

temp.groupby(temp.index.month_name())[['Volume']].sum().loc['January']
```

Out[94]:

```
Volume    5.257375e+09
Name: January, dtype: float64
```

In [58]:

```
df.loc[ ['2018-01','2019-01'] ]
```

Out[58]:

	Open	High	Low	Close	Adj Close	Volume
Date						
2018-01-01	315.500000	317.750000	311.299988	312.600006	301.846252	4019878.0
2019-01-01	182.600006	185.899994	181.000000	184.250000	179.324417	24160878.0

In [ ]:

```
#find the total share traded  in the year 2018 for second quarter
```

In [96]:

```
df.loc['2018-04':'2018-06'].resample('Q')[['Volume']].sum()
```

Out[96]:

	Volume
Date	
2018-06-30	763903916.0

In [98]:

```
#find the minimum opening in first quarter of 2019 and 2020

df.loc['2019-01':'2019-03'].resample('Q')[['Open']].agg(['min'])
```

Out[98]:

	Open
	min
Date	
2019-03-31	173.0

In [99]:

```
df.loc['2020-01':'2020-03'].resample('Q')[['Open']].agg(['min'])
```

Out[99]:

	Open
	min
Date	
2020-03-31	17.0

In [ ]:

```
#performing the same operation by combining the two frames
```

In [116…

```
df1=df.loc['2019-01':'2019-03']
df2=df.loc['2020-01':'2020-03']

pd.concat([df1,df2],axis=0).resample('Q')[['Open']].min().dropna()
```

Out[116…

	Open
Date	
2019-03-31	173.0
2020-03-31	17.0

In [103…

```
df1=df.loc['2018-01']
```

In [104…

```
df2=df.loc['2020-08']
```

In [105…

```
pd.concat(  [ df1,df2 ] ,axis=0 ) #row-like
```

Out[105…

	Open	High	Low	Close	Adj Close	Volume
Date						

2018-01-01	315.500000	317.750000	311.299988	312.600006	301.846252	4019878.0
2018-01-02	313.399994	314.000000	307.149994	311.649994	300.928894	5224976.0
2018-01-03	312.000000	316.500000	311.149994	315.850006	304.984436	5672263.0
2018-01-04	316.000000	318.399994	313.000000	317.100006	306.191437	5667580.0
2018-01-05	317.500000	337.899994	317.450012	332.850006	321.399628	30720675.0
2018-01-08	336.000000	341.299988	331.299988	333.600006	322.123779	12747890.0
2018-01-09	334.899994	342.799988	327.549988	341.350006	329.607208	13282560.0
2018-01-10	341.500000	342.350006	335.450012	339.799988	328.110474	10385044.0
2018-01-11	339.000000	344.250000	335.299988	343.149994	331.345276	8266126.0
2018-01-12	344.100006	344.700012	337.549988	340.899994	329.172668	5688676.0
2018-01-15	341.899994	343.700012	335.100006	336.000000	324.441223	7142164.0
2018-01-16	336.000000	338.750000	328.000000	334.850006	323.330811	7296505.0
2018-01-17	335.100006	343.500000	331.399994	342.399994	330.621063	7985222.0
2018-01-18	350.000000	356.899994	332.350006	341.200012	329.462372	35465087.0
2018-01-19	347.500000	352.250000	339.100006	349.350006	337.332001	21425789.0
2018-01-22	349.950012	358.250000	348.750000	355.350006	343.125580	13456538.0
2018-01-23	359.850006	360.399994	352.299988	359.549988	347.181091	10196645.0
2018-01-24	357.000000	366.299988	356.000000	364.799988	352.250488	11258771.0
2018-01-25	364.500000	364.500000	355.649994	361.600006	349.160583	8963188.0
2018-01-29	361.200012	363.700012	355.549988	358.000000	345.684387	7931235.0
2018-01-30	358.000000	360.799988	351.850006	353.350006	341.194397	7890491.0
2018-01-31	353.000000	356.549988	350.450012	354.399994	342.208252	8527044.0
2020-08-03	12.000000	12.050000	11.900000	12.000000	12.000000	90175030.0
2020-08-04	12.300000	12.400000	12.150000	12.250000	12.250000	156911412.0
2020-08-05	12.250000	12.850000	12.200000	12.850000	12.850000	372809488.0
2020-08-06	13.450000	13.450000	13.150000	13.450000	13.450000	426044666.0
2020-08-07	14.000000	14.100000	13.350000	14.100000	14.100000	706275586.0
2020-08-10	14.500000	14.800000	14.300000	14.800000	14.800000	221529507.0
2020-08-11	15.500000	15.500000	15.500000	15.500000	15.500000	34750719.0
2020-08-12	16.250000	16.250000	16.250000	16.250000	16.250000	31868213.0
2020-08-13	17.049999	17.049999	15.650000	15.850000	15.850000	722489696.0
2020-08-14	15.900000	15.900000	15.100000	15.100000	15.100000	190647399.0
2020-08-17	14.700000	14.900000	14.350000	14.350000	14.350000	276737240.0
2020-08-18	14.100000	15.050000	13.900000	15.050000	15.050000	404132200.0
2020-08-19	15.500000	15.800000	15.350000	15.800000	15.800000	257838316.0
2020-08-20	15.800000	16.400000	15.350000	15.750000	15.750000	401468763.0
2020-08-21	16.000000	16.100000	15.300000	15.550000	15.550000	146216818.0
2020-08-24	15.250000	15.450000	14.800000	14.800000	14.800000	331316063.0
2020-08-25	14.750000	15.000000	14.350000	14.750000	14.750000	206492592.0
2020-08-26	14.850000	14.900000	14.600000	14.650000	14.650000	107557951.0
2020-08-27	14.750000	14.900000	14.600000	14.700000	14.700000	99011520.0
2020-08-28	15.100000	15.400000	14.900000	15.000000	15.000000	273107176.0
2020-08-31	15.350000	15.600000	14.300000	14.350000	14.350000	244889307.0

In [121…

```
df1=df[ ['Open','Close'] ].head(5)
```

In [120…

```
df2=df[ ['Volume'] ].head(5)
```

In [122…

```
pd.concat(  [  df1,df2 ]  , axis= 1)
```

Out[122…

	Open	**Close**	**Volume**
Date			
2017-12-11	313.500000	311.600006	4416465.0

2017-12-12	312.000000	306.799988	5457103.0
2017-12-13	306.350006	301.899994	6911856.0
2017-12-14	303.899994	303.899994	4904177.0
2017-12-15	307.000000	315.899994	20571225.0

In [131…

```
d1={
    "Id":[1,2,3,4, 15],
    "Name":['John',"Marie","Anne","Joseph",'Aston'],
    "Age":[19,25,17,20,35],
}

df1=pd.DataFrame(d1)
df1
```

Out[131…

	Id	Name	Age
0	1	John	19
1	2	Marie	25
2	3	Anne	17
3	4	Joseph	20
4	15	Aston	35

In [132…

```
d2={
    "Id":[2,3,1,4,100],
    "Salary":[20000,30000,10000,40000,80000]

}

df2=pd.DataFrame(d2)
df2
```

Out[132…

	Id	Salary
0	2	20000
1	3	30000
2	1	10000
3	4	40000
4	100	80000

In [133…

```
pd.concat([df1,df2],axis=1)
```

Out[133…

	Id	Name	Age	Id	Salary
0	1	John	19	2	20000
1	2	Marie	25	3	30000
2	3	Anne	17	1	10000
3	4	Joseph	20	4	40000
4	15	Aston	35	100	80000

In [135…

```
df1
```

Out[135…

	Id	Name	Age
0	1	John	19
1	2	Marie	25
2	3	Anne	17
3	4	Joseph	20
4	15	Aston	35

In [136…

```
df2
```

Out[136…

	Id	Salary

0	2	20000
1	3	30000
2	1	10000
3	4	40000
4	100	80000

In [134…

```
#identify a common column between 2 frames?

#only consider records with id values common in both
df1.merge(df2, on='Id',how='inner' ) #inner, outer, left and right
```

Out[134…

	Id	Name	Age	Salary
0	1	John	19	10000
1	2	Marie	25	20000
2	3	Anne	17	30000
3	4	Joseph	20	40000

In [137…

```
#consider all records
df1.merge(df2, on='Id',how='outer' ) #inner, outer, left and right
```

Out[137…

	Id	Name	Age	Salary
0	1	John	19.0	10000.0
1	2	Marie	25.0	20000.0
2	3	Anne	17.0	30000.0
3	4	Joseph	20.0	40000.0
4	15	Aston	35.0	NaN
5	100	NaN	NaN	80000.0

In [138…

```
#all the common records and all records from the first data frame even if its unique

df1.merge(df2, on='Id',how='left' ) #inner, outer, left and right
```

Out[138…

	Id	Name	Age	Salary
0	1	John	19	10000.0
1	2	Marie	25	20000.0
2	3	Anne	17	30000.0
3	4	Joseph	20	40000.0
4	15	Aston	35	NaN

In [140…

```
#all the common records and all records from the second data frame even if its unique
df1.merge(df2, on='Id',how='right' ) #inner, outer, left and right
```

Out[140…

	Id	Name	Age	Salary
0	2	Marie	25.0	20000
1	3	Anne	17.0	30000
2	1	John	19.0	10000
3	4	Joseph	20.0	40000
4	100	NaN	NaN	80000

In [142…

```
d1={
    "Name":['John',"Marie","Anne","Joseph",'Aston'],
    "Age":[19,25,17,20,35],
}

df1=pd.DataFrame(d1,index=[9,10,11,12,13])

d2={
    "Salary":[20000,30000,10000,40000,80000]

}

df2=pd.DataFrame(d2, index=[10,11,12,9,13] )
```

In [143…

```
df1
```

Out[143…

	Name	Age
9	John	19
10	Marie	25
11	Anne	17
12	Joseph	20
13	Aston	35

In [144…

```
df2
```

Out[144…

	Salary
10	20000
11	30000
12	10000
9	40000
13	80000

In [146…

```
d3={
    "Gender":["Male","Female","Female","Male","Male"]

}

df3=pd.DataFrame(d3, index=[9,10,11,12,13] )
df3
```

Out[146…

	Gender
9	Male
10	Female
11	Female
12	Male
13	Male

In [149…

```
#inner join on the basis of index of a record
df1.merge(df2,  left_index=True, right_index=True,how='inner'  )\
    .merge( df3,left_index=True,right_index=True,how='inner' )
```

Out[149…

	Name	Age	Salary	Gender
9	John	19	40000	Male
10	Marie	25	20000	Female
11	Anne	17	30000	Female
12	Joseph	20	10000	Male
13	Aston	35	80000	Male

In [ ]:

```
#what if common column does not have the same name?
```

In [152…

```
d1={
    "Id":[1,2,3,4, 15],
    "Name":['John',"Marie","Anne","Joseph",'Aston'],
    "Age":[19,25,17,20,35],
}

df1=pd.DataFrame(d1)


d2={
    "EId":[2,3,1,4,100],
    "Salary":[20000,30000,10000,40000,80000]

}

df2=pd.DataFrame(d2)
```

In [155…

```
df1.merge(df2,   left_on= 'Id', right_on='EId', how='inner').drop( columns=['EId'], axis=1 )
```

Out[155…

	Id	Name	Age	Salary
0	1	John	19	10000
1	2	Marie	25	20000
2	3	Anne	17	30000
3	4	Joseph	20	40000

In [157…

```
df2.T
```

Out[157…

	0	1	2	3	4
EId	2	3	1	4	100
Salary	20000	30000	10000	40000	80000

In [158…

```
import seaborn as sns
```

In [160…

```
df1=pd.read_csv("https://raw.githubusercontent.com/datasciencedojo/datasets/master/titanic.csv")
df2=pd.read_csv("/home/harshit/DataSets/YESBANK.NS.csv")
```

In [162…

```
sns.set_style('darkgrid')
```

In [161…

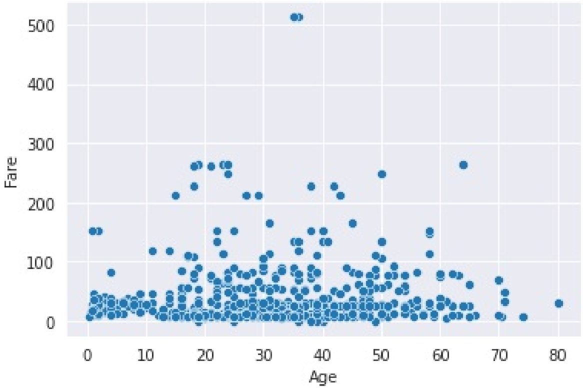
```
#scatterplot as well as a line plot
```

In [164…

```
sns.scatterplot(x='Age',y='Fare',data=df1)
```

Out[164…

```
<AxesSubplot:xlabel='Age', ylabel='Fare'>
```

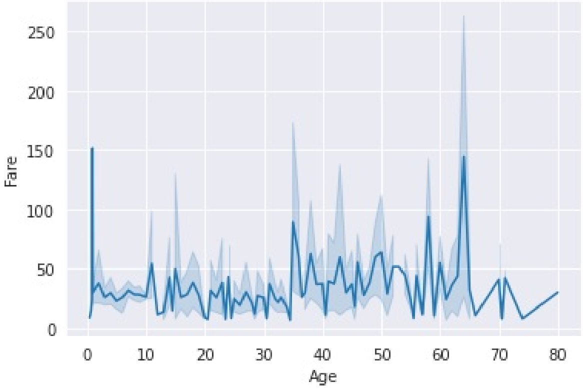


In [165…

```
sns.lineplot(x='Age',y='Fare',data=df1)
```

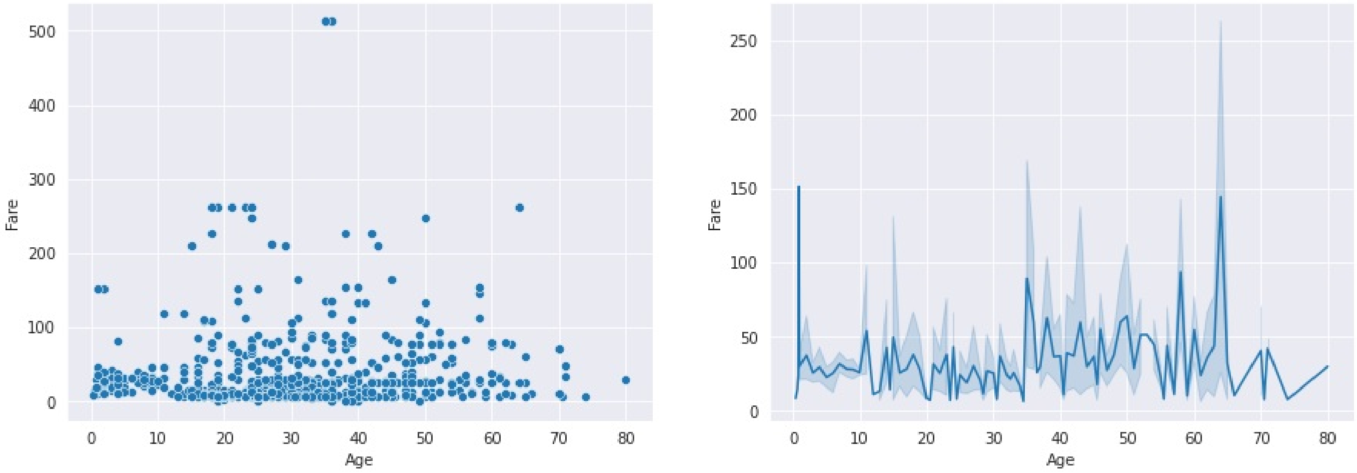
Out[165…

```
<AxesSubplot:xlabel='Age', ylabel='Fare'>
```



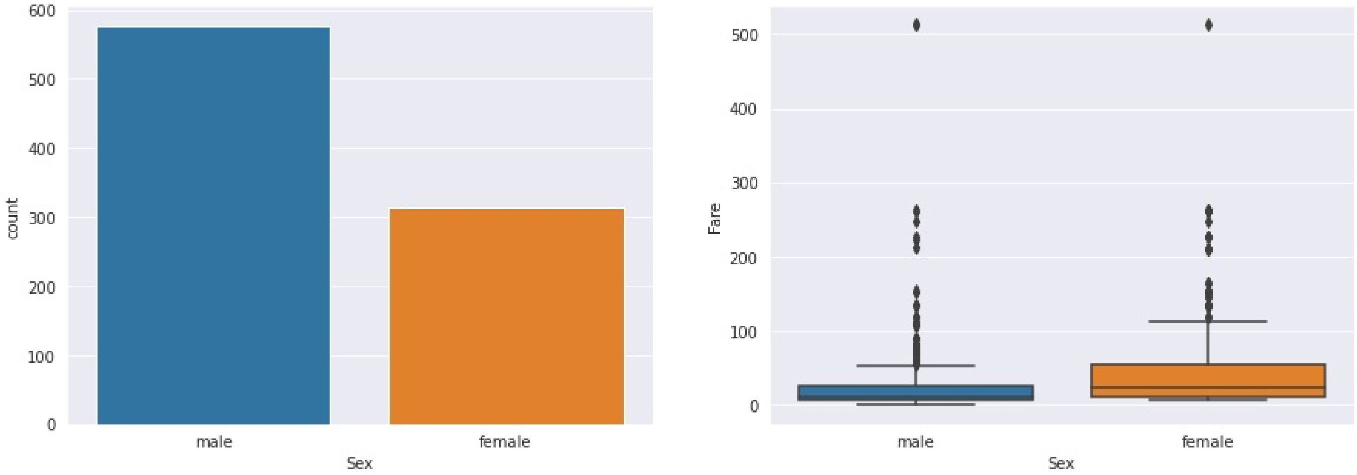
In [178…

```
#multiplot--->Generating multiple plots together in the same canvas area!


import matplotlib.pyplot as plt


fig , ax =  plt.subplots( 1,2,figsize=(15,5) )

sns.scatterplot(x='Age',y='Fare',data=df1, ax=ax[0] )
sns.lineplot(x='Age',y='Fare',data=df1,  ax=ax[1] )


plt.show()
```



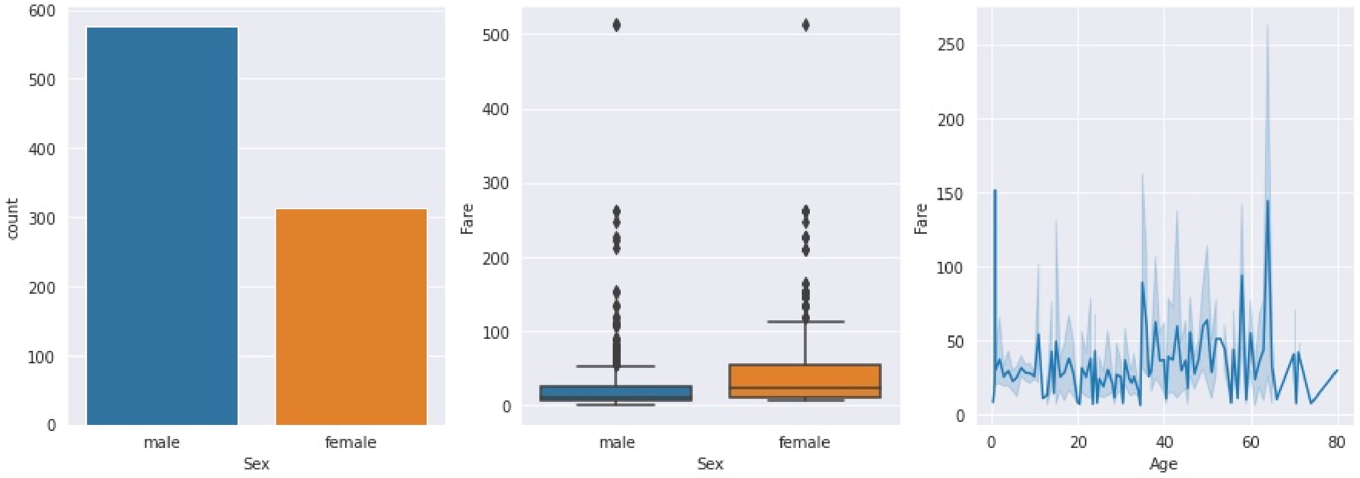
In [179…

```
#multiplot--->Generating multiple plots together in the same canvas area!


import matplotlib.pyplot as plt


fig , ax =  plt.subplots( 1,2,figsize=(15,5) )

sns.countplot(x='Sex',data=df1, ax=ax[0] )
sns.boxplot(x='Sex',y='Fare',data=df1,  ax=ax[1] )


plt.show()
```



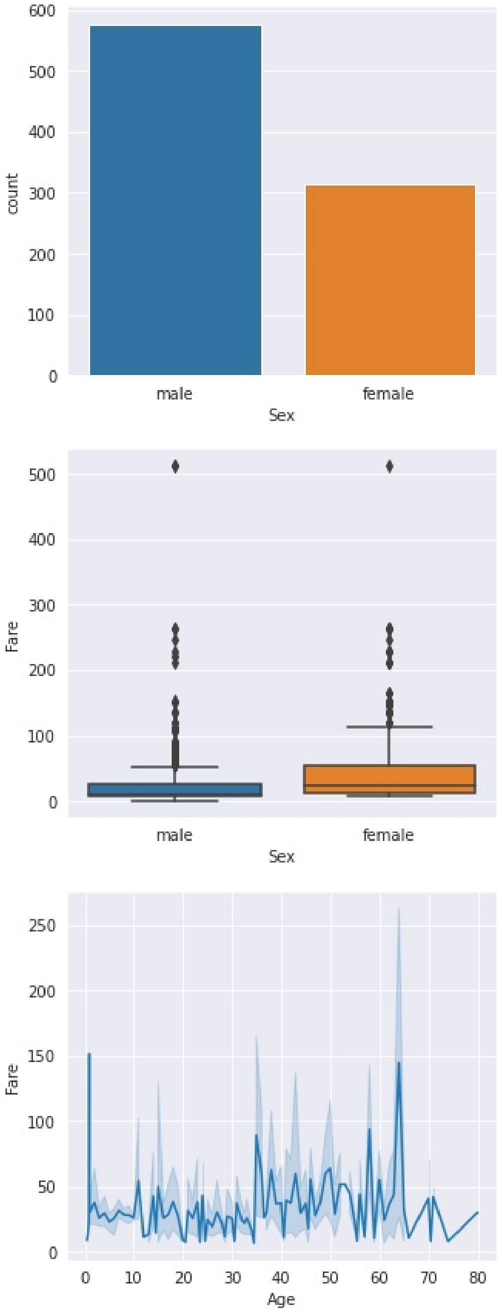
In [180…

```
#multiplot--->Generating multiple plots together in the same canvas area!


import matplotlib.pyplot as plt


fig , ax =  plt.subplots( 1,3,figsize=(15,5) )

sns.countplot(x='Sex',data=df1, ax=ax[0] )
sns.boxplot(x='Sex',y='Fare',data=df1,  ax=ax[1] )
sns.lineplot(x='Age',y='Fare',data=df1,  ax=ax[2] )

plt.show()
```



In [185…

```
#multiplot--->Generating multiple plots together in the same canvas area!


import matplotlib.pyplot as plt

fig,ax =  plt.subplots( 3,1,figsize=(5,15) )
sns.countplot(x='Sex',data=df1, ax=ax[0] )
sns.boxplot(x='Sex',y='Fare',data=df1,  ax=ax[1] )
sns.lineplot(x='Age',y='Fare',data=df1,  ax=ax[2] )

plt.show()
```

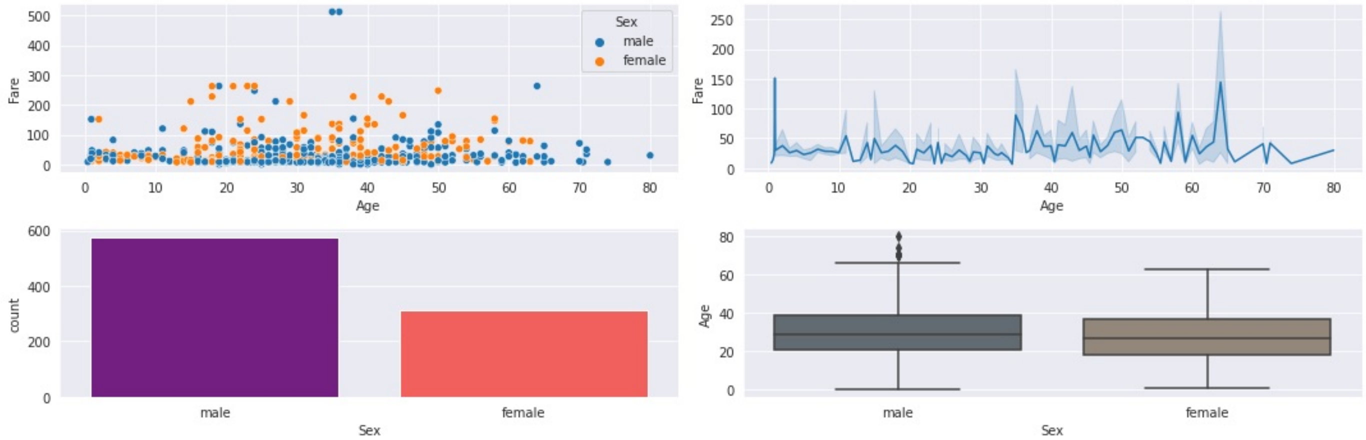


In [ ]:

```
#generate 4 multiplots to represent count of titanic passengers categorized by Pclass, Sex,
#Embarked and Survived value
```

In [202…

```
import matplotlib.pyplot as plt

fig,ax =  plt.subplots( 2,2,figsize=(15,5) )


sns.scatterplot(x='Age',
                y='Fare',
                data=df1,
                hue='Sex',
                ax=ax[0][0] )

sns.lineplot(x='Age',
             y='Fare',
             data=df1,
             ax=ax[0][1] )

sns.countplot(x='Sex',
              data=df1,
              palette='magma',
              saturation=1,
              ax=ax[1][0] )

sns.boxplot(x='Sex',
            y='Age',
            data=df1,
            saturation=0.1,
            ax=ax[1][1] )


plt.tight_layout()
plt.show()
```



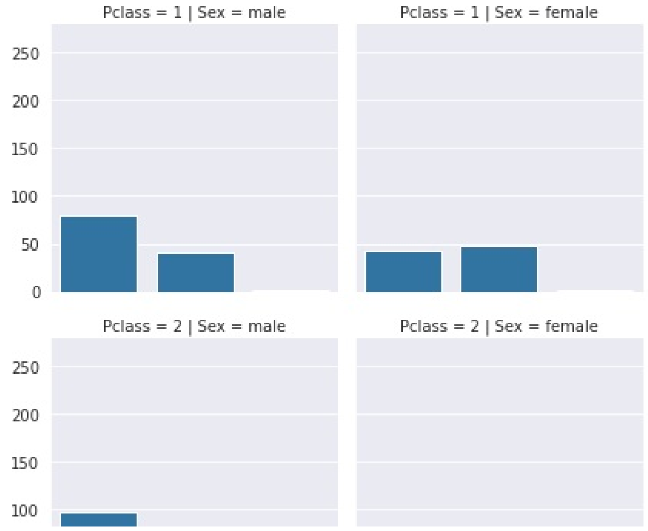
In [209…

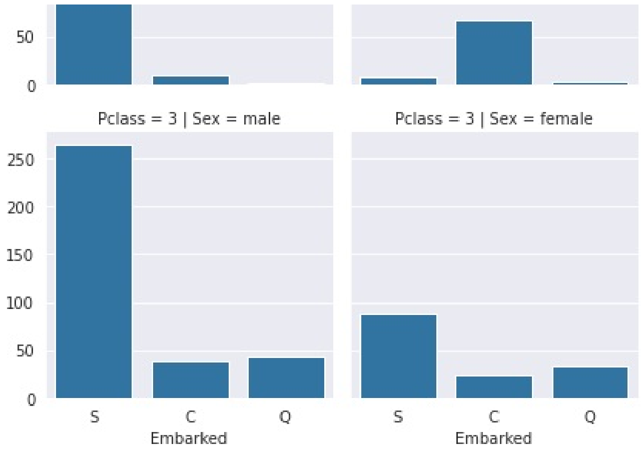
```
g=sns.FacetGrid(data=df1, row='Pclass',col='Sex')

g.map(sns.countplot,'Embarked')
```

```
/home/harshit/.local/lib/python3.8/site-packages/seaborn/axisgrid.py:645: UserWarning: Using the countplot functi
on without specifying `order` is likely to produce an incorrect plot.
  warnings.warn(warning)
```

Out[209…

```
<seaborn.axisgrid.FacetGrid at 0x7f5d6ef3d0a0>
```



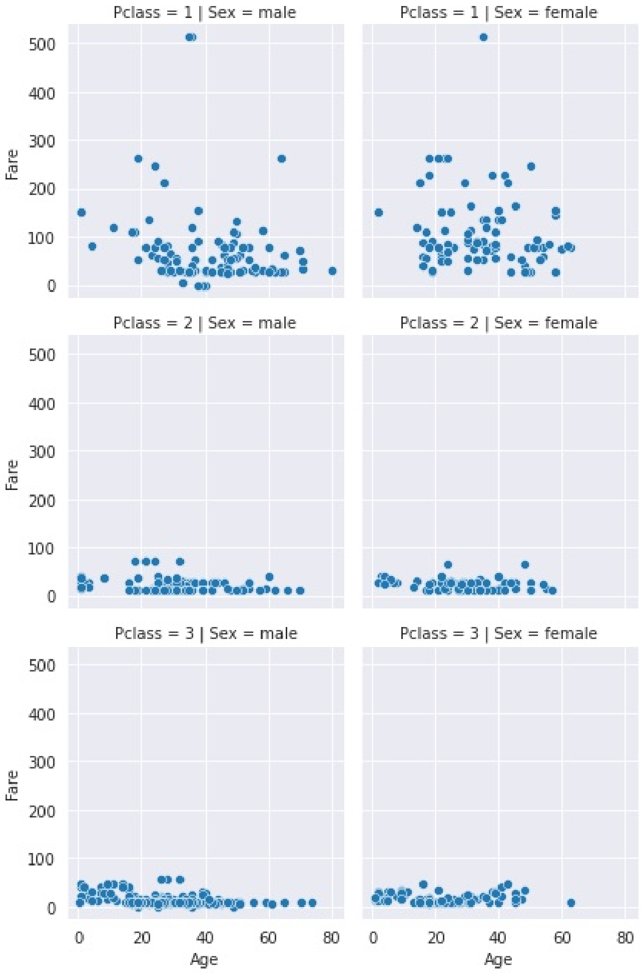


In [210…

```
g=sns.FacetGrid(data=df1, row='Pclass',col='Sex')

g.map(sns.scatterplot,'Age','Fare')
```

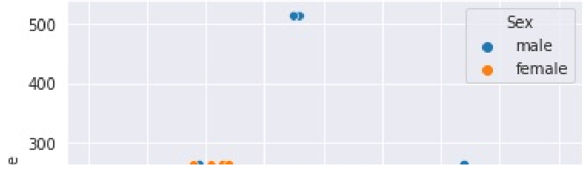
Out[210…

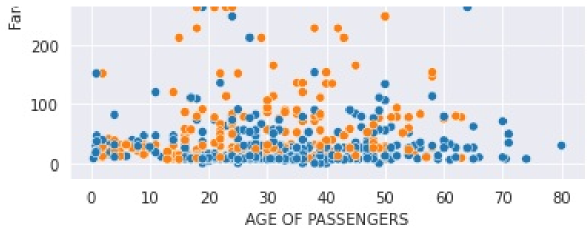
```
<seaborn.axisgrid.FacetGrid at 0x7f5d6f32a670>
```



In [212…

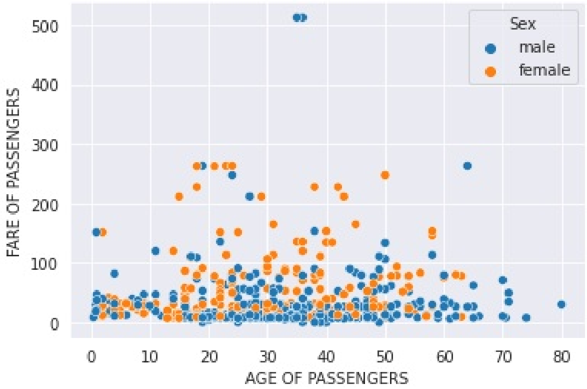
```
sns.scatterplot(x='Age',
                y='Fare',
                data=df1,
                hue='Sex',)

plt.xlabel('AGE OF PASSENGERS')
plt.show()
```





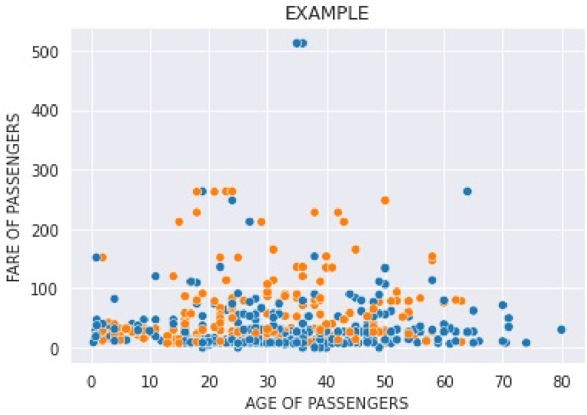
In [213…

```
sns.scatterplot(x='Age',
                y='Fare',
                data=df1,
                hue='Sex',)

plt.xlabel('AGE OF PASSENGERS')
plt.ylabel('FARE OF PASSENGERS')
plt.show()
```



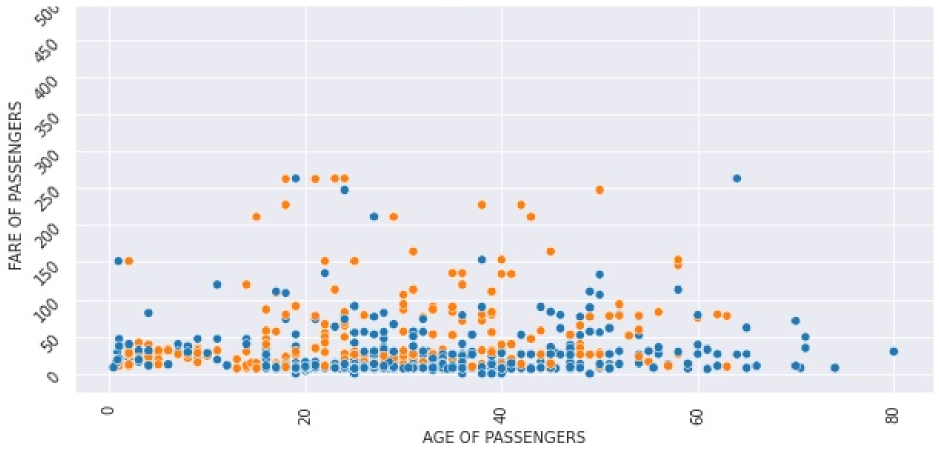
In [218…

```
sns.scatterplot(x='Age',
                y='Fare',
                data=df1,
                hue='Sex',legend=False)

plt.xlabel('AGE OF PASSENGERS')
plt.ylabel('FARE OF PASSENGERS')
plt.title('EXAMPLE')
plt.show()
```



In [226…

```
plt.figure(figsize=(10,5))

sns.scatterplot(x='Age',
                y='Fare',
                data=df1,
                hue='Sex',legend=False)

plt.xlabel('AGE OF PASSENGERS')
plt.ylabel('FARE OF PASSENGERS')
plt.title('EXAMPLE')

plt.yticks( [  num for num in range(0,550,50) ],rotation=45   )

plt.xticks( [  num for num in range(0,100,20) ],rotation=90   )
plt.show()
```

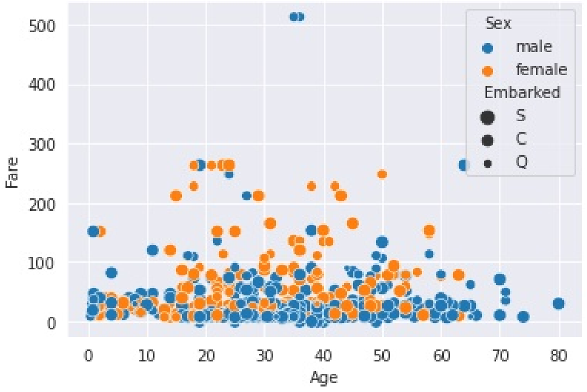


In [229…

```python
#change size of the markers according to a continuous/categorical property
sns.scatterplot(x='Age',
            y='Fare',
            data=df1,
            hue='Sex', size='Embarked')
```

Out[229…

```
<AxesSubplot:xlabel='Age', ylabel='Fare'>
```



In [ ]:

```python
sns.scatterplot(x='Age',
            y='Fare',
            data=df1,
            hue='Sex', size='Embarked')
```

In [231…

```python
df2
```

Out[231…

	Date	Open	High	Low	Close	Adj Close	Volume
0	2017-12-11	313.500000	315.799988	310.600006	311.600006	300.880615	4416465.0
1	2017-12-12	312.000000	312.000000	305.899994	306.799988	296.245758	5457103.0
2	2017-12-13	306.350006	307.350006	301.049988	301.899994	291.514282	6911856.0
3	2017-12-14	303.899994	304.649994	301.750000	303.899994	293.445526	4904177.0
4	2017-12-15	307.000000	317.450012	307.000000	315.899994	305.032715	20571225.0
...	...	...	...	...	...	...	...
733	2020-12-02	15.700000	15.900000	14.850000	15.450000	15.450000	311349886.0
734	2020-12-03	15.650000	15.800000	15.250000	15.450000	15.450000	152445535.0
735	2020-12-04	15.600000	15.600000	15.050000	15.350000	15.350000	149691622.0
736	2020-12-07	15.650000	15.850000	15.500000	15.750000	15.750000	193242183.0
737	2020-12-08	16.000000	17.299999	16.000000	17.299999	17.299999	562741066.0

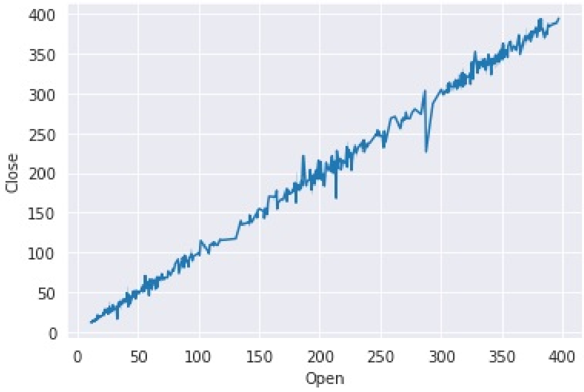
738 rows × 7 columns

In [232…

```python
#lineplot


sns.lineplot(x='Open',y='Close',data=df2,palette='magma')
```

Out[232…

```
<AxesSubplot:xlabel='Open', ylabel='Close'>
```

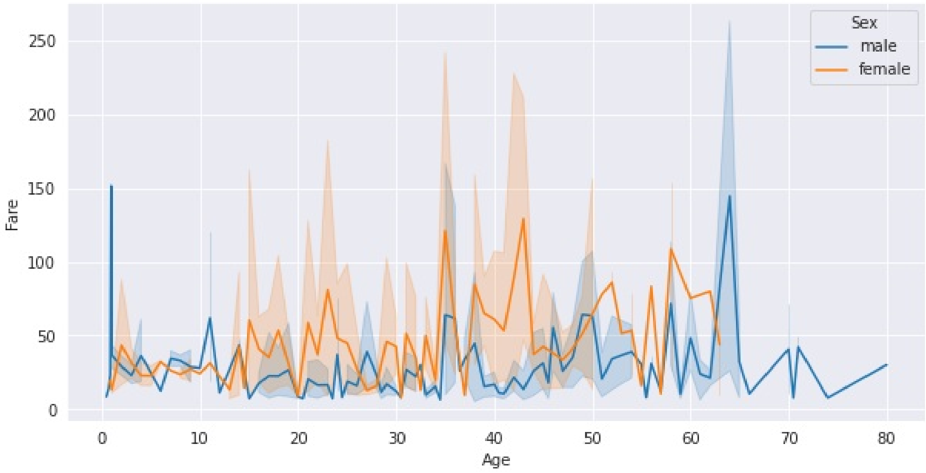


In [235…

```
plt.figure(figsize=(10,5))

sns.lineplot(x='Age',
             y='Fare',
             data=df1,
             hue='Sex'
            )
```

Out[235…

```
<AxesSubplot:xlabel='Age', ylabel='Fare'>
```

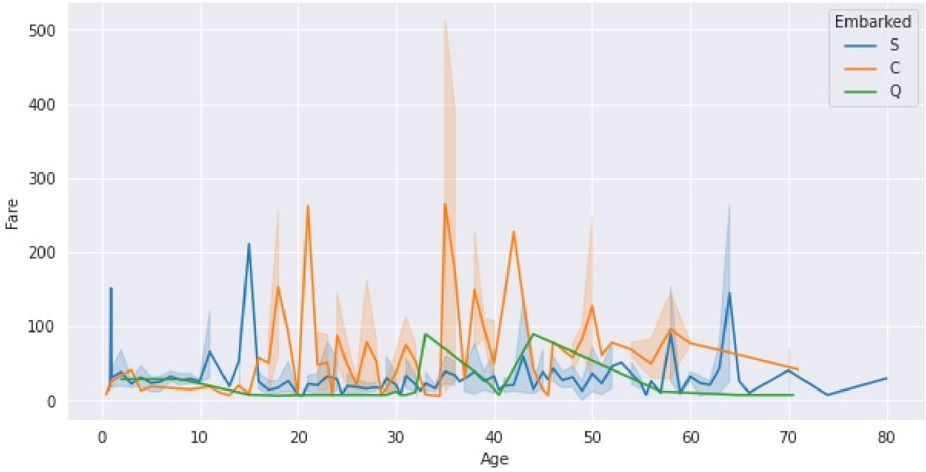


In [236…

```
plt.figure(figsize=(10,5))

sns.lineplot(x='Age',
             y='Fare',
             data=df1,
             hue='Embarked'
            )
```

Out[236…

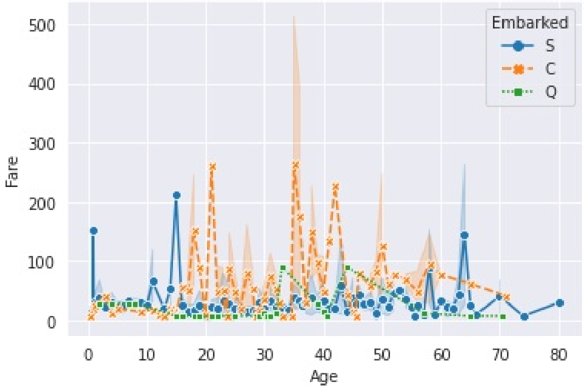
```
<AxesSubplot:xlabel='Age', ylabel='Fare'>
```



In [248…

```
sns.lineplot(x='Age',
             y='Fare',
             data=df1,
             hue='Embarked',
             style='Embarked',
             markers=True
            )
```

Out[248… <AxesSubplot:xlabel='Age', ylabel='Fare'>



In [243…

```
#find the fare for passengers embarking from 'Q' across various age brackets
import numpy as np
cuts=pd.cut(df1['Age'],np.arange(0,90,10))

df1.groupby([cuts,'Embarked'])[['Fare']].mean()
```

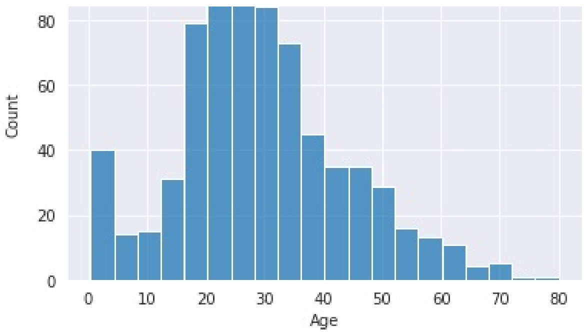
Out[243…

		Fare
Age	Embarked	
(0, 10]	C	21.031022
	Q	29.125000
	S	32.196567
(10, 20]	C	50.639143
	Q	7.481950
	S	25.422141
(20, 30]	C	55.557443
	Q	8.512500
	S	22.670647
(30, 40]	C	111.727996
	Q	24.767857
	S	29.019021
(40, 50]	C	86.327350
	Q	48.875000
	S	30.309500
(50, 60]	C	72.551042
	Q	12.350000
	S	34.399283
(60, 70]	C	61.979200
	Q	7.750000
	S	45.053864
(70, 80]	C	42.079200
	Q	7.750000
	S	18.887500

In [254…

```
sns.histplot(x='Age',data=df1)
```

Out[254… <AxesSubplot:xlabel='Age', ylabel='Count'>

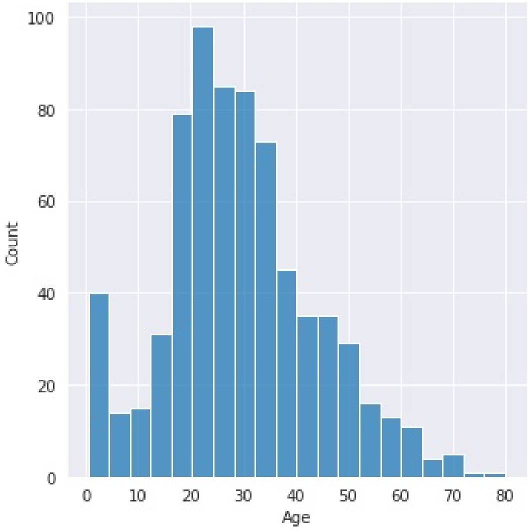


In [253…

```
sns.displot(x='Age',data=df1,kind='hist')
```

Out[253…

```
<seaborn.axisgrid.FacetGrid at 0x7f5d740607f0>
```



In [ ]:

In [251…

```
sns.__version__
```

Out[251…

```
'0.11.0'
```

In [257…

```
df2['50% Close'] = df2['Close']*0.5
df2
```

Out[257…

	Date	Open	High	Low	Close	Adj Close	Volume	50% Close
0	2017-12-11	313.500000	315.799988	310.600006	311.600006	300.880615	4416465.0	155.800003
1	2017-12-12	312.000000	312.000000	305.899994	306.799988	296.245758	5457103.0	153.399994
2	2017-12-13	306.350006	307.350006	301.049988	301.899994	291.514282	6911856.0	150.949997
3	2017-12-14	303.899994	304.649994	301.750000	303.899994	293.445526	4904177.0	151.949997
4	2017-12-15	307.000000	317.450012	307.000000	315.899994	305.032715	20571225.0	157.949997
...	...	...	...	...	...	...	...	...
733	2020-12-02	15.700000	15.900000	14.850000	15.450000	15.450000	311349886.0	7.725000
734	2020-12-03	15.650000	15.800000	15.250000	15.450000	15.450000	152445535.0	7.725000
735	2020-12-04	15.600000	15.600000	15.050000	15.350000	15.350000	149691622.0	7.675000
736	2020-12-07	15.650000	15.850000	15.500000	15.750000	15.750000	193242183.0	7.875000
737	2020-12-08	16.000000	17.299999	16.000000	17.299999	17.299999	562741066.0	8.649999

738 rows × 8 columns

In [258…

```
#create a column called difference. It should show diff in open and close price

df['Diff'] = df['Close'] -df['Open']
```

```
df
```

Out[258…

Date	Open	High	Low	Close	Adj Close	Volume	Diff
2017-12-11	313.500000	315.799988	310.600006	311.600006	300.880615	4416465.0	-1.899994
2017-12-12	312.000000	312.000000	305.899994	306.799988	296.245758	5457103.0	-5.200012
2017-12-13	306.350006	307.350006	301.049988	301.899994	291.514282	6911856.0	-4.450012
2017-12-14	303.899994	304.649994	301.750000	303.899994	293.445526	4904177.0	0.000000
2017-12-15	307.000000	317.450012	307.000000	315.899994	305.032715	20571225.0	8.899994
...	...	...	...	...	...	...	...
2020-12-02	15.700000	15.900000	14.850000	15.450000	15.450000	311349886.0	-0.250000
2020-12-03	15.650000	15.800000	15.250000	15.450000	15.450000	152445535.0	-0.200000
2020-12-04	15.600000	15.600000	15.050000	15.350000	15.350000	149691622.0	-0.250000
2020-12-07	15.650000	15.850000	15.500000	15.750000	15.750000	193242183.0	0.100000
2020-12-08	16.000000	17.299999	16.000000	17.299999	17.299999	562741066.0	1.299999

738 rows × 7 columns

In [ ]:

```
8779092028

harshitshukla36@gmail.com
```