```
import pandas as pd
import numpy as np
```

```
# till now
# Timestamp
pd.Timestamp('6th jan 2023 8:10')
# DatetimeIndex -> df and series index
pd.DatetimeIndex([pd.Timestamp('6th jan 2023 8:10'),pd.Timestamp('7th jan 2023 8:10'),pd.Timestamp('8th jan 2023 8:10')])[0]
# date_range()
pd.date_range(start='2023-1-6',end='2023-1-31',freq='D')
# to_datetime()
s = pd.Series(['2023/1/6','2023/1/7','2023/1/7'])
pd.to_datetime(s).dt.day_name()
```

```
0      Friday
1    Saturday
2    Saturday
dtype: object
```

Timedelta Object

Represents a duration, the difference between two dates or times.

```
# create using Timestamp objects
t1 = pd.Timestamp('6th Jan 2023 8:20:14')
t2 = pd.Timestamp('26th Jan 2023 10:00:00')

t2 - t1
```

```
Timedelta('20 days 01:39:46')
```

```
# standalone creation
pd.Timedelta(days=2,hours=10,minutes=35)
```

```
Timedelta('2 days 10:35:00')
```

```
# Arithmetic
pd.Timestamp('6th jan 2023') + pd.Timedelta(days=2,hours=10,minutes=35)
```

```
Timestamp('2023-01-08 10:35:00')
```

```
pd.date_range(start='2023-1-6',end='2023-1-31',freq='D') - pd.Timedelta(days=2,hours=10,minutes=35)
```

```
DatetimeIndex(['2023-01-03 13:25:00', '2023-01-04 13:25:00',
               '2023-01-05 13:25:00', '2023-01-06 13:25:00',
               '2023-01-07 13:25:00', '2023-01-08 13:25:00',
               '2023-01-09 13:25:00', '2023-01-10 13:25:00',
               '2023-01-11 13:25:00', '2023-01-12 13:25:00',
               '2023-01-13 13:25:00', '2023-01-14 13:25:00',
               '2023-01-15 13:25:00', '2023-01-16 13:25:00',
               '2023-01-17 13:25:00', '2023-01-18 13:25:00',
               '2023-01-19 13:25:00', '2023-01-20 13:25:00',
               '2023-01-21 13:25:00', '2023-01-22 13:25:00',
               '2023-01-23 13:25:00', '2023-01-24 13:25:00',
               '2023-01-25 13:25:00', '2023-01-26 13:25:00',
               '2023-01-27 13:25:00', '2023-01-28 13:25:00'],
              dtype='datetime64[ns]', freq='D')
```

```
# real life example
df = pd.read_csv('deliveries.csv')
df.head()
```

	order_date	delivery_date
0	5/24/98	2/5/99
1	4/22/92	3/6/98
2	2/10/91	8/26/92
3	7/21/92	11/20/97
4	9/2/93	6/10/98

```
df['order_date'] = pd.to_datetime(df['order_date'])
df['delivery_date'] = pd.to_datetime(df['delivery_date'])
```

```
df['delivery_time_period'] = df['delivery_date'] - df['order_date']

df['delivery_time_period'].mean()
```

```
    Timedelta('1217 days 22:53:53.532934128')
```

Time series

A time series is a data set that tracks a sample over time. In particular, a time series allows one to see what factors influence certain variables from period to period. Time series analysis can be useful to see how a given asset, security, or economic variable changes over time.

Examples

- Financial Data (Company stocks)
- Natural Data (Rainfall measurement)
- Event Data (Covid)
- Medical Data (Heart rate monitoring)

Types of Operations done on Time Series

- Time Series Analysis
- Time Series Forecasting

```
google = pd.read_csv('google.csv')
google.head()
```

	Date	Open	High	Low	Close	Adj Close	Volume
0	2004-08-19	49.813290	51.835709	47.800831	49.982655	49.982655	44871361
1	2004-08-20	50.316402	54.336334	50.062355	53.952770	53.952770	22942874
2	2004-08-23	55.168217	56.528118	54.321388	54.495735	54.495735	18342897
3	2004-08-24	55.412300	55.591629	51.591621	52.239197	52.239197	15319808
4	2004-08-25	52.284027	53.798351	51.746044	52.802086	52.802086	9232276

```
google.tail()
```

	Date	Open	High	Low	Close	Adj Close	Volume
4466	2022-05-16	2307.679932	2332.149902	2286.699951	2295.850098	2295.850098	1164100
4467	2022-05-17	2344.550049	2344.550049	2306.750000	2334.030029	2334.030029	1078800
4468	2022-05-18	2304.750000	2313.913086	2242.840088	2248.020020	2248.020020	1399100
4469	2022-05-19	2236.820068	2271.750000	2209.360107	2214.909912	2214.909912	1459600
4470	2022-05-20	2241.709961	2251.000000	2127.459961	2186.260010	2186.260010	1878100

```
subs = pd.read_csv('subscribers.csv')
views = pd.read_csv('views.csv')
wt = pd.read_csv('watch-time.csv')
comments = pd.read_csv('comments.csv')
```

```
comments.tail()
```

	Date	Comments added
1354	2022-12-31	20
1355	2023-01-01	13
1356	2023-01-02	66
1357	2023-01-03	22
1358	2023-01-04	21

```
yt = subs.merge(views,on='Date').merge(wt,on='Date').merge(comments,on='Date')
```

```
yt.head()
```

	Date	Subscribers	Views	Watch time (hours)	Comments added
0	2019-04-17	0	0	0.0	0
1	2019-04-18	0	0	0.0	0
2	2019-04-19	0	0	0.0	0
3	2019-04-20	0	0	0.0	0
4	2019-04-21	0	0	0.0	0

```
google['Date'] = pd.to_datetime(google['Date'])
yt['Date'] = pd.to_datetime(yt['Date'])
```

```
yt.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1359 entries, 0 to 1358
Data columns (total 5 columns):
 #   Column              Non-Null Count  Dtype         
---  ------              --------------  -----         
 0   Date                1359 non-null   datetime64[ns]
 1   Subscribers         1359 non-null   int64         
 2   Views               1359 non-null   int64         
 3   Watch time (hours)  1359 non-null   float64       
 4   Comments added      1359 non-null   int64         
dtypes: datetime64[ns](1), float64(1), int64(3)
memory usage: 63.7 KB
```

```
google.set_index('Date',inplace=True)
yt.set_index('Date',inplace=True)
```

```
yt.head()
```

	Subscribers	Views	Watch time (hours)	Comments added
Date				
2019-04-17	0	0	0.0	0
2019-04-18	0	0	0.0	0
2019-04-19	0	0	0.0	0
2019-04-20	0	0	0.0	0
2019-04-21	0	0	0.0	0

```
# fetch a specific date
google.loc['2021-12-30']
```

```
Open           2929.000000
High           2941.250000
Low            2915.169922
Close          2920.050049
Adj Close      2920.050049
Volume       648900.000000
Name: 2021-12-30 00:00:00, dtype: float64
```

```
# partial indexing -> select a particular year/month
yt.loc['2022-12']
yt.loc['2022']
```

	Subscribers	Views	Watch time (hours)	Comments added
Date				
2022-01-01	30	3111	327.3357	11
2022-01-02	42	4109	402.5877	5
2022-01-03	45	4264	431.8496	13
2022-01-04	53	4160	427.6701	7
2022-01-05	32	3971	433.6201	8
...	...	...	...	...
2022-12-27	201	18245	2228.4891	18
2022-12-31	116	12468	1446.5901	20

```
# create year month-name and day-name
yt['month_name'] = yt.index.month_name()
yt['weekday_name'] = yt.index.day_name()
yt['quarter'] = yt.index.quarter
```

```
yt.head()
```

	Subscribers	Views	Watch time (hours)	Comments added	month_name	weekdday_name	quarter	weekday_name
Date								
2019-04-17	0	0	0.0	0	April	April	2	Wednesday
2019-04-18	0	0	0.0	0	April	April	2	Thursday
2019-04-19	0	0	0.0	0	April	April	2	Friday

```
google['month_name'] = google.index.month_name()
google['weekday_name'] = google.index.day_name()
google['quarter'] = google.index.quarter

google.head()
```

	Open	High	Low	Close	Adj Close	Volume	month_name	weekday_name	quarter
Date									
2004-08-19	49.813290	51.835709	47.800831	49.982655	49.982655	44871361	August	Thursday	3
2004-08-20	50.316402	54.336334	50.062355	53.952770	53.952770	22942874	August	Friday	3
2004-08-23	55.168217	56.528118	54.321388	54.495735	54.495735	18342897	August	Monday	3

```
# slicing
yt.loc['2022-12-15':'2023-1-1':2]
```

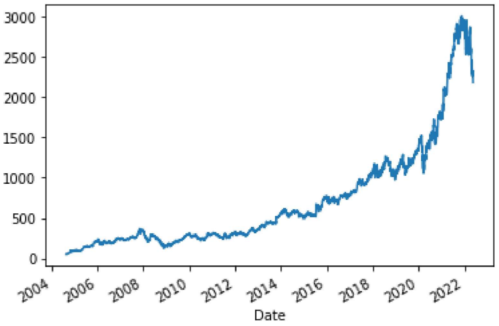
```
# challenge -> fetch info for a particular date every year -> limitation of timedelta
google.head()


google[google.index.isin(pd.date_range(start='2005-1-6',end='2022-1-6',freq=pd.DateOffset(years=1)))]
```

	Open	High	Low	Close	Adj Close	Volume	month_name	weekday_nam
Date								
2005-01-06	97.175758	97.584229	93.509506	93.922951	93.922951	20852067	January	Thursda
2006-01-06	227.581970	234.371521	225.773743	231.960556	231.960556	35646914	January	Frida
2009-01-06	165.868286	169.763687	162.585587	166.406265	166.406265	12898566	January	Tuesda
2010-01-06	311.761444	311.761444	302.047852	302.994293	302.994293	7987226	January	Wednesda
2011-01-06	304.199799	308.060303	303.885956	305.604523	305.604523	4131026	January	Thursda
2012-01-06	328.344299	328.767700	323.681763	323.796326	323.796326	5405987	January	Frida
2014-01-06	554.426880	557.340942	551.154114	556.573853	556.573853	3551864	January	Monda
2015-01-06	513.589966	514.761719	499.678131	500.585632	500.585632	2899940	January	Tuesda
2016-	730.000000	747.179993	728.919983	743.619995	743.619995	1947000	January	Wednesda

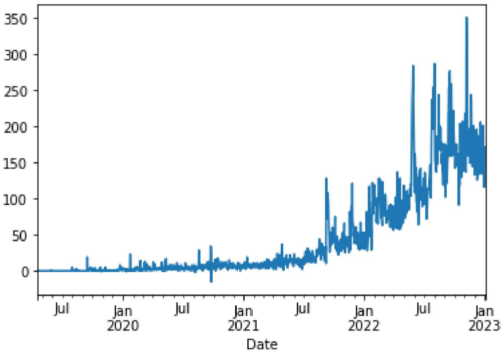
```
# viz a single col
google['Close'].plot()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f2a760e42e0>
```

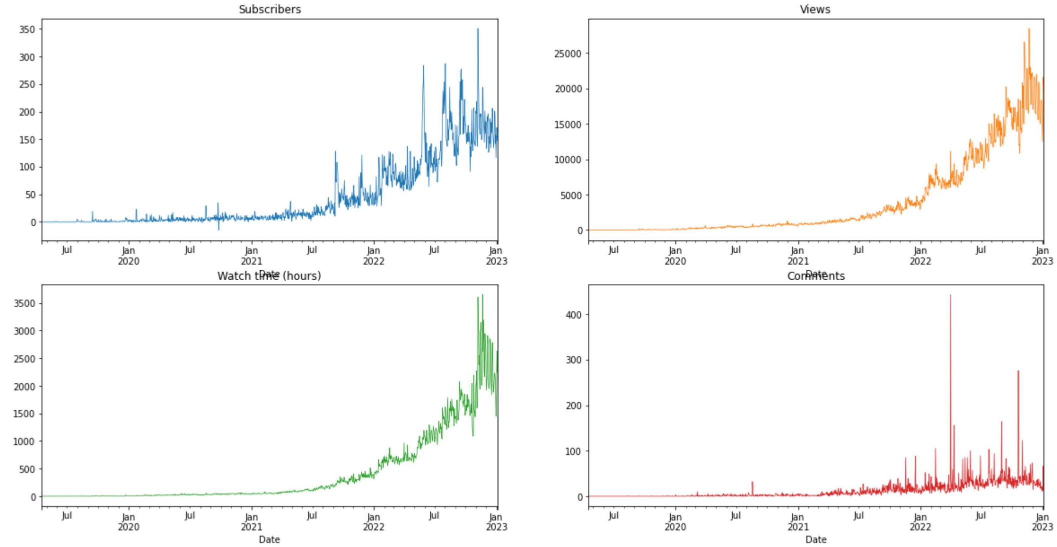


```
yt['Subscribers'].plot()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f2a77a8c100>
```

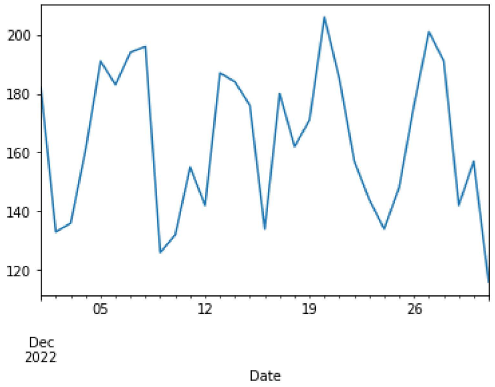


```
# viz all cols together
subset_yt = yt[['Subscribers', 'Views', 'Watch time (hours)', 'Comments added']]
ax = subset_yt.plot(subplots=True,
                    layout=(2, 2),
                     sharex=False,
                     sharey=False,
                     linewidth=0.7,
                     fontsize=10,
                     legend=False,
                      figsize=(20,10),
                      title=['Subscribers', 'Views', 'Watch time (hours)', 'Comments'])
```



```
# plot for a particular year/month/week
yt.loc['2022-12']['Subscribers'].plot()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f2a77d25eb0>
```

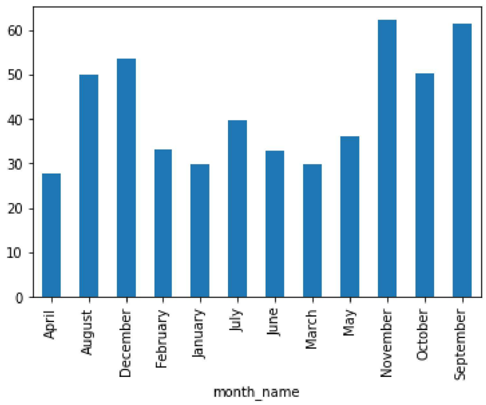


```
google.loc['2021-12']['Close'].plot()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f2a77d25700>
```

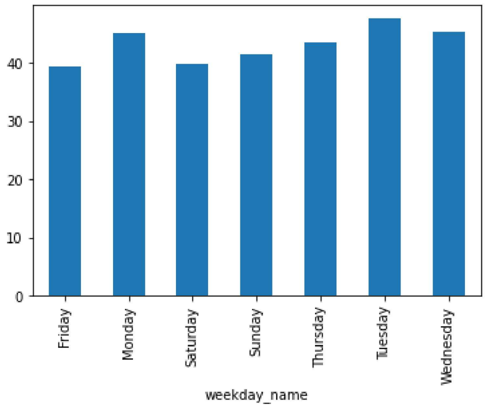
```
# monthly trend
yt.groupby('month_name')['Subscribers'].mean().plot(kind='bar')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f2a75581730>
```



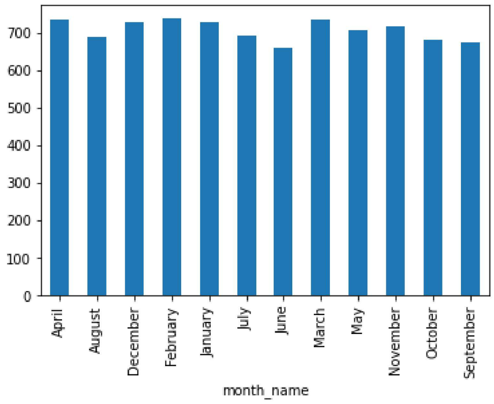
```
# yearly trend
yt.groupby('weekday_name')['Subscribers'].mean().plot(kind='bar')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f2a75504fa0>
```



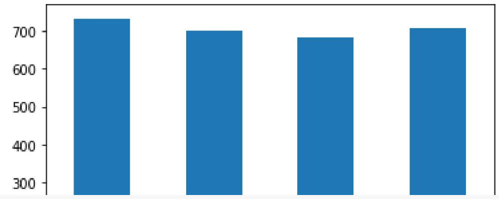
```
google.groupby('month_name')['Close'].mean().plot(kind='bar')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f2a75441a90>
```



```
# quaterly trend
google.groupby('quarter')['Close'].mean().plot(kind='bar')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f2a7552d6a0>
```



```
# frequency
google.index
```

```
DatetimeIndex(['2004-08-19', '2004-08-20', '2004-08-23', '2004-08-24',
               '2004-08-25', '2004-08-26', '2004-08-27', '2004-08-30',
               '2004-08-31', '2004-09-01',
               ...
               '2022-05-09', '2022-05-10', '2022-05-11', '2022-05-12',
               '2022-05-13', '2022-05-16', '2022-05-17', '2022-05-18',
               '2022-05-19', '2022-05-20'],
              dtype='datetime64[ns]', name='Date', length=4632, freq='B')
```

```
# asfreq
google.asfreq('6H',method='bfill')
```

	Open	High	Low	Close	Adj Close	Volume	month_name	weekday
Date								
2004-08-19 00:00:00	49.813290	51.835709	47.800831	49.982655	49.982655	44871361.0	August	Thu
2004-08-19 06:00:00	50.316402	54.336334	50.062355	53.952770	53.952770	22942874.0	August	
2004-08-19 12:00:00	50.316402	54.336334	50.062355	53.952770	53.952770	22942874.0	August	
2004-08-19 18:00:00	50.316402	54.336334	50.062355	53.952770	53.952770	22942874.0	August	
2004-08-20 00:00:00	50.316402	54.336334	50.062355	53.952770	53.952770	22942874.0	August	
...	...	...	...	...	...	...	...	
2022-05-								

Resampling

Resampling involves changing the frequency of your time series observations.

Two types of resampling are:

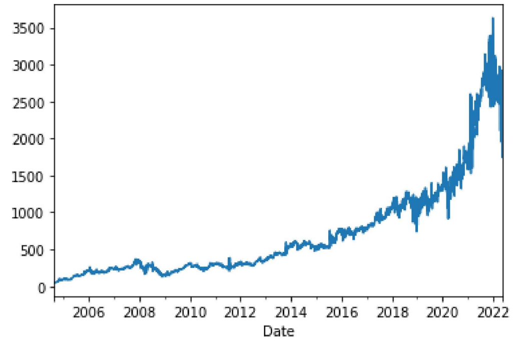
Upsampling: Where you increase the frequency of the samples, such as from minutes to seconds.
Downsampling: Where you decrease the frequency of the samples, such as from days to months.

```
# Downsampling
yt['Subscribers'].plot()
yt['Subscribers'].resample('Y').mean().plot()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f2a74e2f280>
```

```
# Upsampling
google['Close'].resample('12H').interpolate(method='spline',order=2).plot()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f2a751e70d0>
```

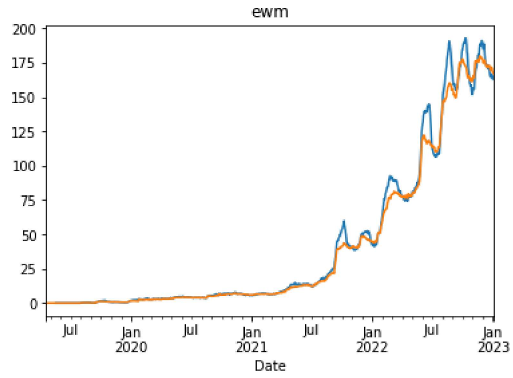


Rolling Window(Smoothing)

Time series data in original format can be quite volatile, especially on smaller aggregation levels. The concept of rolling, or moving averages is a useful technique for smoothing time series data.

```
# Rolling window
yt['Subscribers'].rolling(30).mean().plot(title='rolling')
yt['Subscribers'].ewm(30).mean().plot(title='ewm')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f2a6d4ae9d0>
```



Shifting

The shift() function is Pandas is used to, well, shift the entire series up or down by the desired number of periods.

```
# shift
yt['Subscribers']
```

```
Date
2019-04-17      0
2019-04-18      0
2019-04-19      0
2019-04-20      0
2019-04-21      0
             ... 
2022-12-31    116
2023-01-01    142
2023-01-02    171
2023-01-03    162
2023-01-04    147
Name: Subscribers, Length: 1359, dtype: int64
```

```
yt['Subscribers'].shift(-1)
```

```
Date
2019-04-17      0.0
```

```
2019-04-18      0.0
2019-04-19      0.0
2019-04-20      0.0
2019-04-21      0.0
              ...
2022-12-31    142.0
2023-01-01    171.0
2023-01-02    162.0
2023-01-03    147.0
2023-01-04      NaN
Name: Subscribers, Length: 1359, dtype: float64
```

```
# shift example
df = pd.read_csv('/content/login.csv',header=None)
df = df[[1,2]]
df.head()
df.rename(columns={1:'user_id',2:'login_time'},inplace=True)
df.head()
```

	user_id	login_time
0	466	2017-01-07 18:24:07
1	466	2017-01-07 18:24:55
2	458	2017-01-07 18:25:18
3	458	2017-01-07 18:26:21
4	592	2017-01-07 19:09:59

```
user_df = df[df['user_id'] == 458]
user_df.head()
```

	user_id	login_time
2	458	2017-01-07 18:25:18
3	458	2017-01-07 18:26:21
9	458	2017-01-09 11:13:12
10	458	2017-01-09 11:34:02
25	458	2017-01-10 12:14:11

```
user_df['login_time'] = pd.to_datetime(user_df['login_time'])
user_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 208 entries, 2 to 1018
Data columns (total 2 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   user_id     208 non-null    int64
 1   login_time  208 non-null    datetime64[ns]
dtypes: datetime64[ns](1), int64(1)
memory usage: 4.9 KB
<ipython-input-269-fd41e73e6ce6>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-co
  user_df['login_time'] = pd.to_datetime(user_df['login_time'])
```

```
user_df['shifted'] = user_df['login_time'].shift(1)
(user_df['login_time'] - user_df['shifted']).mean()
```

```
<ipython-input-270-091b95be4a6b>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-co
  user_df['shifted'] = user_df['login_time'].shift(1)
Timedelta('0 days 17:29:22.053140096')
```

```
ax = df.plot(subplots=True,
                    layout=(3, 2),
                     sharex=False,
                     sharey=False,
                     linewidth=0.7,
                     fontsize=10,
                     legend=False,
                      figsize=(20,15),
                      title=['Open', 'High', 'Low', 'Close','Adjusted Close', 'Volume'])
```