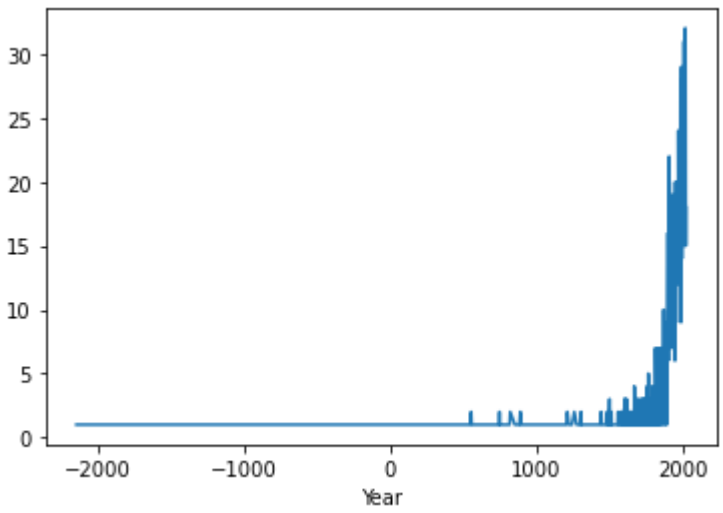
In [1]:

```
##PS1.1
import pandas as pd
#read the csv file
eqs_df = pd.read_csv("Sig_Eqs.csv")
#group by the 'Country' and sum the deaths and sort in descending order
eqs_total_death = eqs_df.groupby(['Country']).sum().sort_values('Deaths', ascending=False)
#print the top 10 term
print(eqs_total_death['Deaths'].head(10))
```

```
Country
CHINA         2074900.0
TURKEY        1074769.0
IRAN          1011437.0
SYRIA          439224.0
ITALY          434863.0
HAITI          323472.0
AZERBAIJAN     317219.0
JAPAN          278138.0
ARMENIA        191890.0
PAKISTAN       148783.0
Name: Deaths, dtype: float64
```

In [2]:

```
#####PS1.2
import matplotlib.pyplot as plt
#choose that  magnitude > 6
mag = eqs_df.loc[eqs_df['Mag']> 6.0 ]
#group by the year, and count the  magnitude>6,then plot
mag = mag.groupby(['Year']).count()['Mag'].plot()
plt.show()
```



In [3]:

```
###PS1.3
def CountEq_LargestEq(x):
# bulid a dataframe which contains input country
    country = eqs_df.loc[eqs_df['Country'] == x]
# use the new dataframe count the 'year' to calculate the total number of earthquakes
    country_sum = country.count()['Year']
# use the idxmax function to get the index of the max number of country, and get the value of year/m
    Y = int(eqs_df['Year'].iloc[country['Mag'].idxmax()])
    M = int(eqs_df['Mo'].iloc[country['Mag'].idxmax()])
    D = int(eqs_df['Dy'].iloc[country['Mag'].idxmax()])
# output the date of max magnitude
    print(x,country_sum ,str(Y) +"-"+ str(M) + "-"+ str(D))

CountEq_LargestEq('JAPAN')
```

```
JAPAN 409 2011-3-11
```

In [4]:

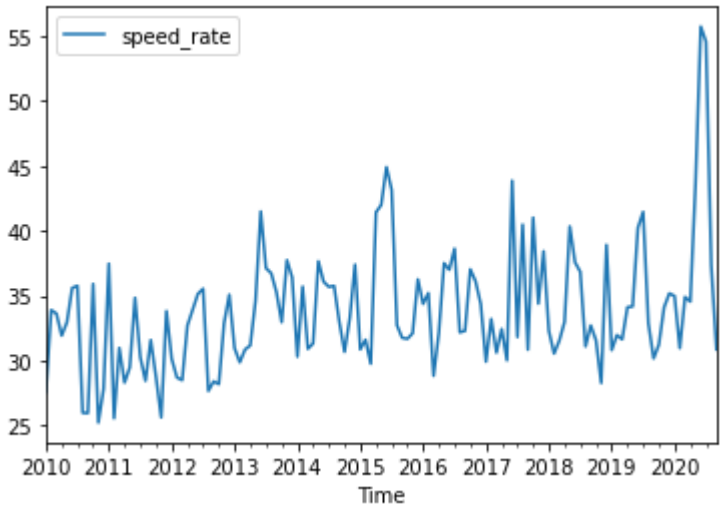
```
#WIND-OBSERVATION speed rate
#The rate of horizontal travel of air past a fixed point.
# MIN: 0000 MAX: 0900 UNITS: meters per second
# 9999 = Missing.

import pandas as pd
# read the csv file of wind
wind = pd.read_csv("2281305.csv")
# bulid a dataframe which split the 'WND' into five pieces
wind_speed = wind['WND'].str.split(',', expand = True)
# rename the head of columns
wind_speed.columns = ['dire_angle', 'dire_quality', 'type', 'speed_rate','speed_quality']
# delete the missing number
wind_speed = wind_speed[ ~ wind_speed['speed_rate'].str.contains('9999') ]
wind_speed['DATE'] = wind['DATE']
# turn the type of 'date' into datetime type
wind_speed['DATE'] = pd.to_datetime(wind_speed['DATE'])
# use the dt function sort the data
wind_speed['Time'] = wind_speed['DATE'].dt.to_period('M')
# change the type of speed_rate
wind_speed['speed_rate'] = wind_speed['speed_rate'].apply(pd.to_numeric)
# build a dataframe,index is 'Time',value is the mean of 'speed_rate'
import numpy as np
pivot = pd.pivot_table(wind_speed, index=['Time'], values=['speed_rate'], aggfunc=np.mean)
# plot
pivot.plot.line()
```

```
C:\ProgramData\Anaconda3\lib\site-packages\IPython\core\interactiveshell.py:3165: Dt
ypeWarning: Columns (4,8,9,12,15,21,22,24,26,31,33,34) have mixed types.Specify dtyp
e option on import or set low_memory=False.
  has_raised = await self.run_ast_nodes(code_ast.body, cell_name,
```

Out[4]:

```
<AxesSubplot:xlabel='Time'>
```



In [6]:

```
#ps3.1
import pandas as pd
flow_df = pd.read_csv("load.csv")
#delete the NAN
flow_df = flow_df.dropna()
flow_df
```

Out[6]:

	CDATE	Time	Flow	AMLE	MLE	LAD
0	2004-01-05	0	48380.0	3.14E+05	3.14E+05	325000.0
1	2004-01-12	0	51910.0	3.37E+05	3.37E+05	349000.0
2	2004-01-26	0	54390.0	3.57E+05	3.57E+05	367000.0
3	2004-02-02	0	68510.0	4.49E+05	4.49E+05	466000.0
4	2004-02-09	0	58620.0	3.90E+05	3.90E+05	400000.0
...	...	...	...	...	...	...
503	2013-11-26	0	104500.0	4.16E+05	4.16E+05	439000.0
504	2013-12-03	0	69190.0	2.79E+05	2.79E+05	289000.0
505	2013-12-10	0	435000.0	1.70E+06	1.70E+06	1840000.0
506	2013-12-17	0	175800.0	6.96E+05	6.96E+05	743000.0
507	2013-12-24	0	114000.0	4.59E+05	4.59E+05	482000.0

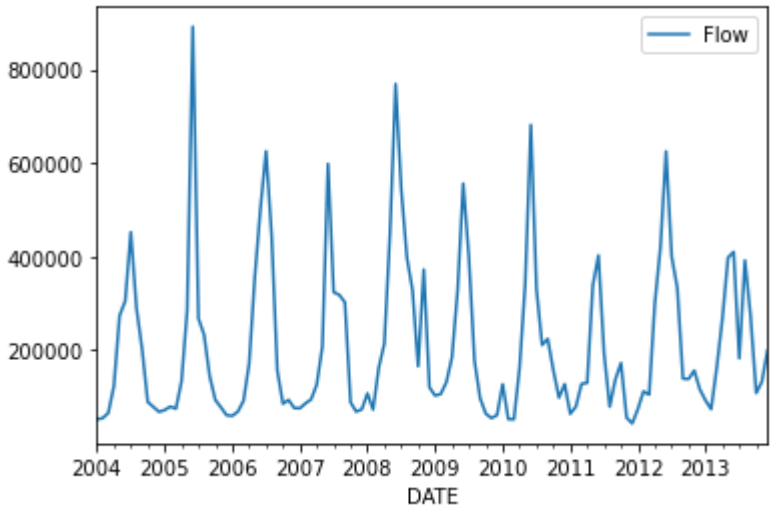
498 rows × 6 columns

In [7]:

```
#3.2
import numpy as np
flow_df['CDATE'] = pd.to_datetime(flow_df['CDATE'])
flow_df['DATE'] = flow_df['CDATE'].dt.to_period('M')
pivot = pd.pivot_table(flow_df, index=['DATE'], values=['Flow'], aggfunc=np.mean)
# plot
pivot.plot.line()
```

Out[7]:

```
<AxesSubplot:xlabel='DATE'>
```



In [8]:

```
#3.3
# bulid a function which calculte these stastic number
def stats(x):
    return pd.Series([x.count(),x.min(),x.median(),x.mean(),x.max(),x.var()],
                     index = ['Count','Min','Median','Mean','Max','Var'])
stats(flow_df['Flow'])
```

Out[8]:

```
Count     4.980000e+02
Min       3.019000e+04
Median    1.298000e+05
Mean      2.121341e+05
Max       1.610000e+06
Var       4.038502e+10
dtype: float64
```

In [ ]:

```

```