1.
1.1
首先引入pandas，numpy，xarray和pyplot。
使用read_csv读取下载的文件（后面要加分隔符，虽然不知道为什么要加，但是我之前没加就报错），命名为Sig_Eqs。
用groupby('Country')对总死亡人数按照国家分组，再用['Deaths'].sum()对Deaths求和。
用sort_values(ascending=False).head(10)把总死亡人数按照降序排序，输出前10行。

1.2
large_quakes是Sig_Eqs中Mag那一列中数值大于6.0的项，然后按照年份分组，再用count()求数量。
使用plot()直接画图。从图中我们可以看到，大约从公元1500年开始地震频率不断增高，并且有越来越高的趋势。

1.3
在CountEq_LargestEq函数中，Sig_Eqs中特定国家的地震命名为country_eqs，用len(country_eqs)计算地震的数量。如果地震数大于0，最大地震就等于按照降序排序的Mag列里的最大值，然后再找到其对应时间，转化为字符串，如果地震数不大于0就说明没有地震。然后返回出地震数和最大地震日期两个值。
新建一个列表，用for循环把国家名、地震数和最大地震日期添加进列表。
最后用f-string格式化字符串，打印出国家名称、地震数和最大地震时间。

2.
2.1
我向我的师兄赵望超寻求了帮助。
首先读取2281305.csv，命名为data。
用q、w、e、r、t五个字母把WND中的五个字符串拆开，然后把q（风速）中等于9999的值过滤掉，将 DATE 列中的字符串转换为 datetime 对象，以特定格式%Y-%m-%dT%H:%M:%S表示。最后创建图形，大小为15×8，m代表1-12月，plt.plot (m, months, marker='o')表示在x轴上标注1-12月，以months的值绘图，用o点作为记号，y轴为平均风速。这样计算的是所有年份的各个月份的平均风速，可以看到10-12月的平均风速明显小于其他各个月份，8月份的平均风速最大。
另一种计算方法是按照各个年份的各个月份的平均风速来绘图，这样看波动比较大，也可以看出来10-12月的风速大致处于低谷。

3.
3.1
我从地球大数据科学工程数据共享服务系统 (casearth.cn)下载了青藏高原纳木错高山草原下垫面上高分辨率（逐小时）CO2 和 H2O数据集（2005-2019）：NAMORS_EC_2005-2019.csv。（原文件太大了上传不了）
首先读取NAMORS_EC_2005-2019.csv，命名为namors，由于原文件中数据太多，故只取其前38列进行后续计算。然后把表格中co2_mixing_ratio列的-9999的值去掉。
3.2
我向我的师兄赵望超寻求了帮助。
文件中的D/T是在excel中把date和time组合在一起的新列，便于python画图。过滤掉co2_mixing_ratio中的异常值，画出散点图。
3.3
计算出co2_mixing_ratio的平均值、中位数、标准差、最大值和最小值，在图上用不同颜色的线段标注。

In [1]:
```python
# 1. Significant earthquakes since 2150 B.C.
#1.1
# import pandas
import pandas as pd
# import numpy
import numpy as np
# import matplotlib
from matplotlib import pyplot as plt
# make plots appear and be stored within the notebook
%matplotlib inline
# filter warnings
import warnings
warnings.filterwarnings("ignore")
import datetime as dt
from datetime import datetime
```

In [2]:
```python
Sig_Eqs = pd.read_csv('earthquakes-2023-11-01_21-35-45_+0800.tsv',sep='\t')
```

In [3]:
```python
# Compute the total number of deaths caused by earthquakes since 2150 B.C. in each
total_deaths = Sig_Eqs.groupby('Country')['Deaths'].sum()
total_deaths
```

Out[3]:
```
Country
AFGHANISTAN                              15644.0
ALBANIA                                   3132.0
ALGERIA                                  39339.0
ANTARCTICA                                   0.0
ANTIGUA AND BARBUDA                          0.0
                                          ...
VENEZUELA                                44480.0
VIETNAM                                      0.0
WALLIS AND FUTUNA (FRENCH TERRITORY)         5.0
YEMEN                                     4192.0
ZAMBIA                                       0.0
Name: Deaths, Length: 156, dtype: float64
```

In [4]:
```python
# print the top ten countries along with the total number of deaths
total_deaths.sort_values(ascending=False).head(10)
```

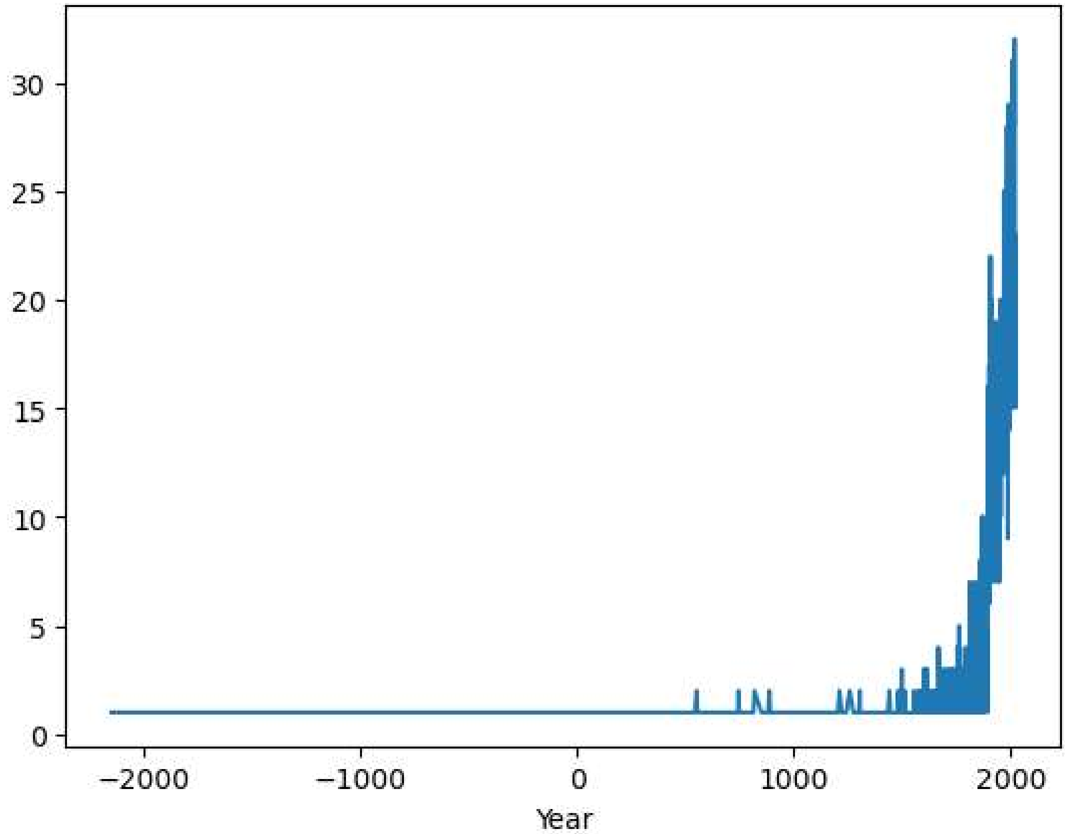
Out[4]:
```
Country
CHINA         2075045.0
TURKEY        1188881.0
IRAN          1011449.0
ITALY          498478.0
SYRIA          439224.0
HAITI          323478.0
AZERBAIJAN     317219.0
JAPAN          279085.0
ARMENIA        191890.0
PAKISTAN       145083.0
Name: Deaths, dtype: float64
```

In [5]:
```
# 1.2
# Compute the total number of earthquakes with magnitude larger than 6.0 worldwide
large_quakes = Sig_Eqs[Sig_Eqs['Mag'] > 6.0]
earthquakes_per_year = large_quakes.groupby(Sig_Eqs['Year'])['Mag'].count()
earthquakes_per_year
```

Out[5]:
```
Year
-2150.0     1
-2000.0     1
-1250.0     1
-1050.0     1
-479.0      1
           ..
 2019.0    27
 2020.0    15
 2021.0    23
 2022.0    20
 2023.0    22
Name: Mag, Length: 536, dtype: int64
```

In [6]:
```
# plot the time series
earthquakes_per_year.plot()
# From the graph, we can see that the frequency of earthquakes has been increasing
```

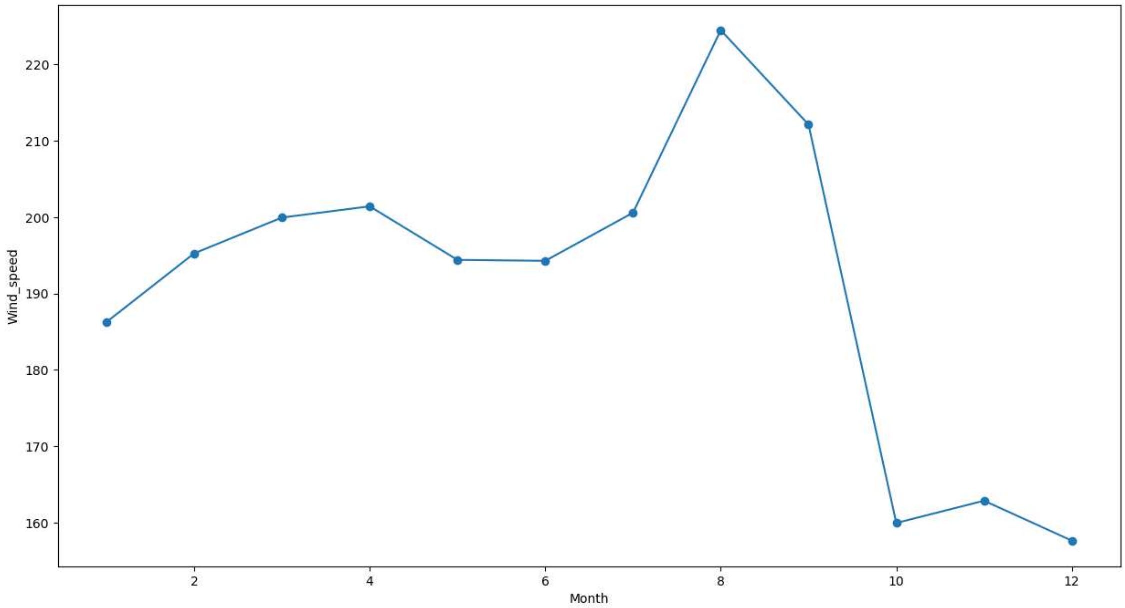
Out[6]: <Axes: xlabel='Year'>



In [7]:
```
# 1.3
# function CountEq_LargestEq

def CountEq_LargestEq(country):
    country_eqs = Sig_Eqs[Sig_Eqs['Country'] == country]
    eq_count = len(country_eqs)

    if eq_count > 0:
        largest_eq = country_eqs.sort_values('Mag', ascending=False).iloc[0]
        largest_eq_date = f"{largest_eq['Year']}-{largest_eq['Mo']}-{largest_eq['Dy
    else:
        largest_eq_date = "No earthquakes"

    return eq_count, largest_eq_date

results = []
for country in Sig_Eqs['Country'].unique():
    eq_count, largest_eq_date = CountEq_LargestEq(country)
    results.append((country, eq_count, largest_eq_date))

results.sort(key=lambda x: x[1], reverse=True)

for country, eq_count, largest_eq_date in results:
    print(f"Country: {country}\nTotal Earthquakes: {eq_count}\nLargest Earthquake D
```

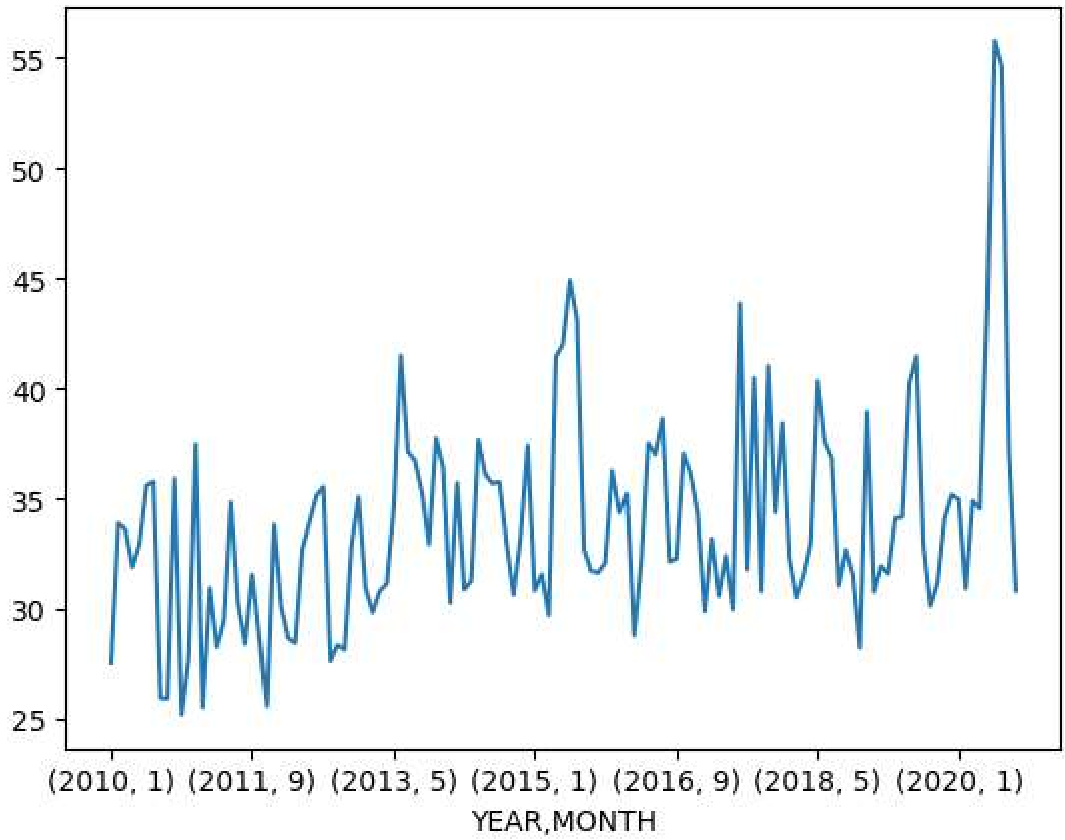
```
Country: CHINA
Total Earthquakes: 620
Largest Earthquake Date: 1668.0-7.0-25.0

Country: JAPAN
Total Earthquakes: 414
Largest Earthquake Date: 2011.0-3.0-11.0

Country: INDONESIA
Total Earthquakes: 411
Largest Earthquake Date: 2004.0-12.0-26.0

Country: IRAN
Total Earthquakes: 384
Largest Earthquake Date: 856.0-12.0-22.0

Country: TURKEY
Total Earthquakes: 335
Largest Earthquake Date: 2023.0-2.0-6.0
```

In [9]:
```
# 2. Wind speed in Shenzhen during the past 10 years
# I asked my Senior fellow apprentice Wangchao Zhao for help.
data = pd.read_csv("2281305.csv")
data['q'], data['w'], data['e'], data['r'], data['t'] = data['WND'].str.split(',').
data = data[data['q'] != 9999]
data['DATE'] = pd.to_datetime(data['DATE'], format='%Y-%m-%dT%H:%M:%S')  # Convert
data['q'] = data['q'].astype(float)
months = data.groupby(data['DATE'].dt.month)['q'].mean()
plt.figure(figsize=(15,8))
m = range(1,13)
plt.plot(m, months, marker='o')
plt.xlabel('Month')
plt.ylabel('Wind_speed')
plt.show()
```



In [10]:
```
# This is another method to show the mean wind speed.
data['WS'] = data['WND'].str.split(',', expand=True)[3]
data['YEAR'] = pd.to_datetime(data['DATE']).dt.year
data['MONTH'] = pd.to_datetime(data['DATE']).dt.month
data1 = data[data['WS'] != '9999'].copy()

data1['WS'] = data1['WS'].astype(float)
data1.groupby(["YEAR", "MONTH"])['WS'].mean().plot()
```

Out[10]: <Axes: xlabel='YEAR,MONTH'>



In [11]:
```
# 3 Explore a data set
# 3.1
namors = pd.read_csv("NAMORS_EC_2005-2019.csv",
                usecols=range(38))
namors.head()
```

Out[11]:

	filename	date	time	D/T	DOY	daytime	file_records
0	2005_12_04_2230_ECNM.DAT	2005/12/4	23:00	2005/12/4 23:00	338.958	0	18000
1	2005_12_04_2300_ECNM.DAT	2005/12/4	23:30	2005/12/4 23:30	338.979	0	18000
2	2005_12_04_2330_ECNM.DAT	2005/12/5	0:00	2005/12/5 0:00	339.000	0	18000
3	2005_12_05_0000_ECNM.DAT	2005/12/5	0:30	2005/12/5 0:30	339.021	0	18000
4	2005_12_05_0030_ECNM.DAT	2005/12/5	1:00	2005/12/5 1:00	339.042	0	18000

5 rows × 38 columns

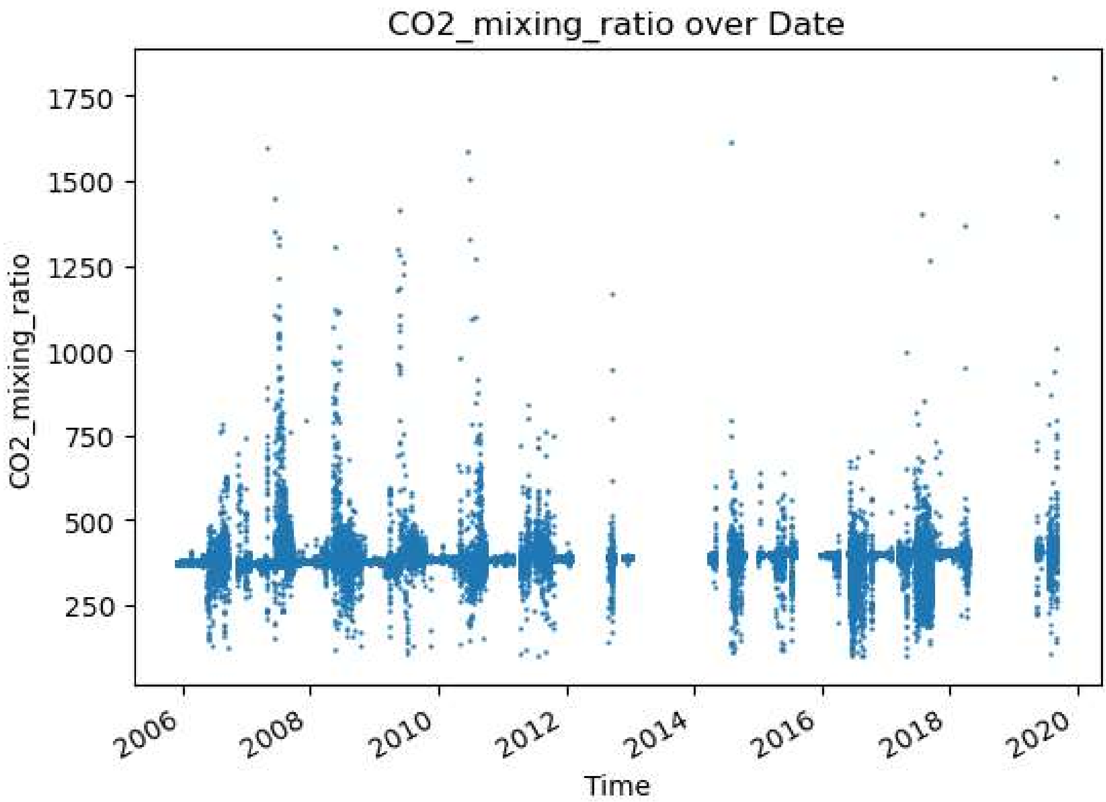
In [12]:
```
# clean possible data points with missing values or bad quality
avail_namors = namors[namors['co2_mixing_ratio'] != -9999]
avail_namors
```

Out[12]:

	filename	date	time	D/T	DOY	daytime	file_recc
0	2005_12_04_2230_ECNM.DAT	2005/12/4	23:00	2005/12/4 23:00	338.958	0	18
1	2005_12_04_2300_ECNM.DAT	2005/12/4	23:30	2005/12/4 23:30	338.979	0	18
2	2005_12_04_2330_ECNM.DAT	2005/12/5	0:00	2005/12/5 0:00	339.000	0	18
3	2005_12_05_0000_ECNM.DAT	2005/12/5	0:30	2005/12/5 0:30	339.021	0	18
4	2005_12_05_0030_ECNM.DAT	2005/12/5	1:00	2005/12/5 1:00	339.042	0	18
...	...	...	...	...	...	...	
241173	2019_09_07_0900_ECNM.DAT	2019/9/7	9:30	2019/9/7 9:30	250.396	1	18
241174	2019_09_07_0930_ECNM.DAT	2019/9/7	10:00	2019/9/7 10:00	250.417	1	18
241175	2019_09_07_1000_ECNM.DAT	2019/9/7	10:30	2019/9/7 10:30	250.437	1	18
241176	2019_09_07_1030_ECNM.DAT	2019/9/7	11:00	2019/9/7 11:00	250.458	1	18
241177	2019_09_07_1100_ECNM.DAT	2019/9/7	11:30	2019/9/7 11:30	250.479	1	18

127121 rows × 38 columns

In [17]:
```
# 3.2 Plot the time series of a certain variable.
# I asked my Senior fellow apprentice Wangchao Zhao for help.
avail_namors['time'] = pd.to_datetime(avail_namors['D/T'], format='%Y/%m/%d %H:%M:%
avail_namors = avail_namors[avail_namors['co2_mixing_ratio'] > 100]
avail_namors = avail_namors[avail_namors['co2_mixing_ratio']<2000]

plt.figure()
plt.scatter(avail_namors['time'], avail_namors['co2_mixing_ratio'], s=0.5)
plt.title('CO2_mixing_ratio over Date')
plt.xlabel('Time')
plt.ylabel('CO2_mixing_ratio')
plt.gcf().autofmt_xdate()
plt.show()
```



In [18]:
```
# 3.3 Conduct at least 5 simple statistical checks with the variable, and report yo

mean_avail_namors = avail_namors['co2_mixing_ratio'].mean()
median_avail_namors = avail_namors['co2_mixing_ratio'].median()
std_avail_namors = avail_namors['co2_mixing_ratio'].std()
min_avail_namors = avail_namors['co2_mixing_ratio'].min()
max_avail_namors = avail_namors['co2_mixing_ratio'].max()

print(f"Mean co2_mixing_ratio: {mean_avail_namors}")
print(f"Median co2_mixing_ratio: {median_avail_namors}")
print(f"Standard Deviation of co2_mixing_ratio: {std_avail_namors}")
print(f"Minimum co2_mixing_ratio: {min_avail_namors}")
print(f"Maximum co2_mixing_ratio: {max_avail_namors}")
```

```
Mean co2_mixing_ratio: 389.5477290182768
Median co2_mixing_ratio: 386.6
Standard Deviation of co2_mixing_ratio: 32.41726133226181
Minimum co2_mixing_ratio: 101.18
Maximum co2_mixing_ratio: 1801.71
```

In [19]:
```
plt.axhline(mean_avail_namors, color='red', linestyle='--', label=f'Mean: {mean_ava
plt.axhline(median_avail_namors, color='green', linestyle='-.', label=f'Mean: {medi
plt.axhline(min_avail_namors, color='blue', linestyle=':', label=f'Mean: {min_avail
plt.axhline(max_avail_namors, color='yellow', linestyle='-', label=f'Mean: {max_ava

plt.scatter(avail_namors['time'], avail_namors['co2_mixing_ratio'], s=0.5)
plt.xlabel('Time')
plt.ylabel('CO2_mixing_ratio')
plt.gcf().autofmt_xdate()
plt.show()
```