Assignment 01

LinQiangMa

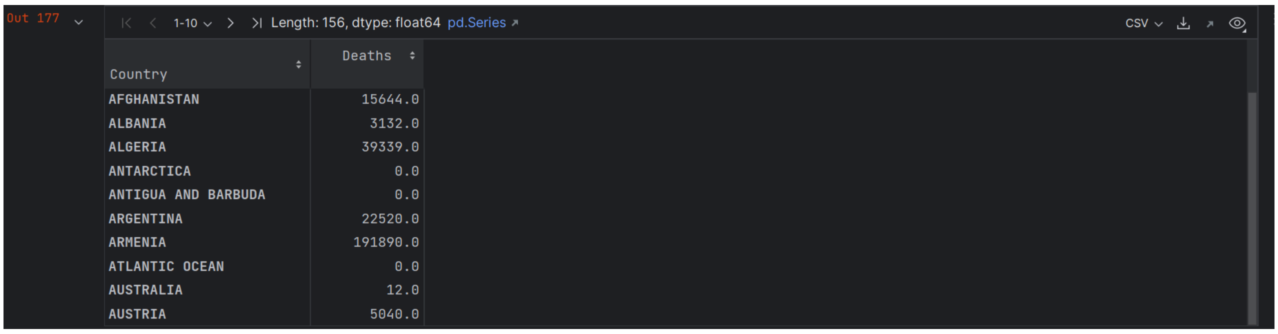
October 2023

Problem1

1.1

```
# ZhunleiZhou taught me many handy functions, and some materials form csdn inspired me.
# import library
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
#%%
# Promblem 1
# Read the file  as an object and name it Sig_Eqs.
Sig_Eqs = pd.read_csv('data.tsv', sep='\t')

# Check information
Sig_Eqs.info()

#%%
# Show header.head()
Sig_Eqs.head()
#%%
# 1.1
# calculate the sum of death for every country
deaths_for_country = Sig_Eqs.groupby('Country')['Deaths'].sum()
deaths_for_country
# sort and print the top ten
top_ten = deaths_for_country.sort_values(ascending=False).head(10)
print(top_ten)
```

output:



Length: 156, dtype: float64 pd.Series

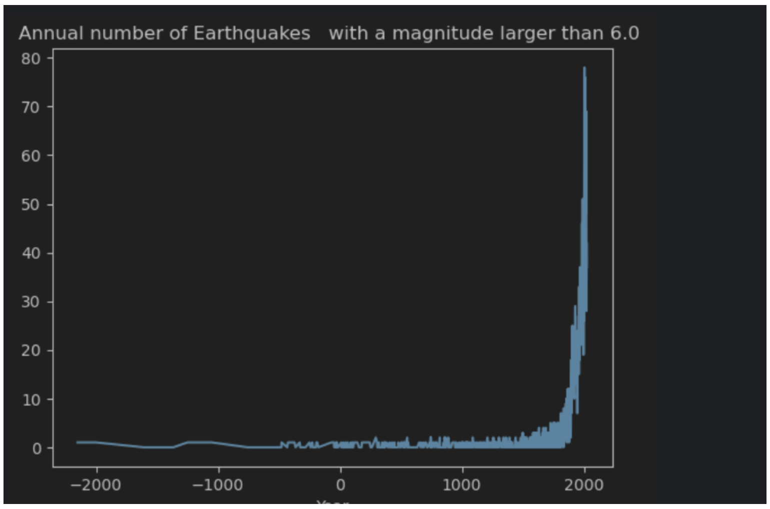
Country	Deaths
AFGHANISTAN	15644.0
ALBANIA	3132.0
ALGERIA	39339.0
ANTARCTICA	0.0
ANTIGUA AND BARBUDA	0.0
ARGENTINA	22520.0
ARMENIA	191890.0
ATLANTIC OCEAN	0.0
AUSTRALIA	12.0
AUSTRIA	5040.0

1.2

```
#1.2
# find out earthquakes with a magnitude larger than 6.0
strong_earthquake = Sig_Eqs[Sig_Eqs['Mag'] > 6.0]
#%%
# count
annual_earthquake = Sig_Eqs.groupby('Year')['Mag'].count()
annual_earthquake.plot(title='Annual number of Earthquakes   with a magnitude larger than 6.0 ')
```

The graph indicates that over the past century, the number of annual earthquakes has significantly increased. One possible reason for this is that we have entered a period of increased seismic activity, and some of the past data may have been missing.

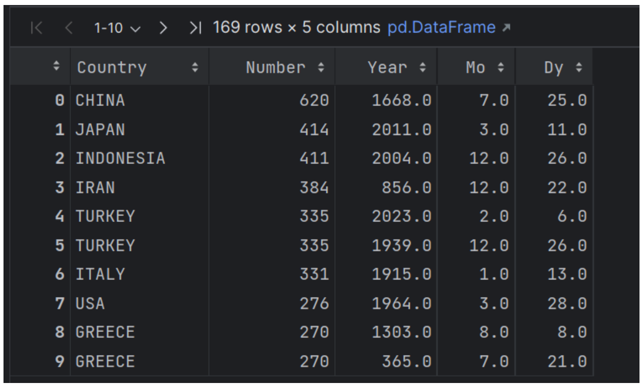
output:



1.3

```
#1.3
def CountEq_LargestEq(country):
    country_data = Sig_Eqs[Sig_Eqs['Country'] == country]

    #return the number of lines
    number = Sig_Eqs[Sig_Eqs['Country'] == country].shape[0]

    # find the maximum
    rank = country_data['Mag'].rank(ascending = False,method="min")

    # find the data corresponding to the maximum
    data = country_data.loc[rank == 1, ['Year', 'Mo', 'Dy']].reset_index(drop=True)
    data.index = data.index + 1
    data.insert(0, 'Number', number)
    data.insert(0, 'Country', country)

    return  data
#%%
#clear the nan
Sig_Eqs_nonan= Sig_Eqs.dropna(subset=['Country'])

# store the all result
Data = pd.DataFrame()

# apply the CountEq_LargestEq(country) to all countries
for Country in Sig_Eqs_nonan['Country'].unique():
    country_result = CountEq_LargestEq(Country)
    Data = pd.concat([Data, country_result])

Data_sorted=Data.sort_values('Number',ascending=False)
Data_sorted.reset_index(drop=True, inplace=True)
Data_sorted
```

output:



```
1-10   169 rows × 5 columns pd.DataFrame

   Country     Number    Year    Mo    Dy
0  CHINA          620  1668.0   7.0  25.0
1  JAPAN          414  2011.0   3.0  11.0
2  INDONESIA      411  2004.0  12.0  26.0
3  IRAN           384   856.0  12.0  22.0
4  TURKEY         335  2023.0   2.0   6.0
5  TURKEY         335  1939.0  12.0  26.0
6  ITALY          331  1915.0   1.0  13.0
7  USA            276  1964.0   3.0  28.0
8  GREECE         270  1303.0   8.0   8.0
9  GREECE         270   365.0   7.0  21.0
```

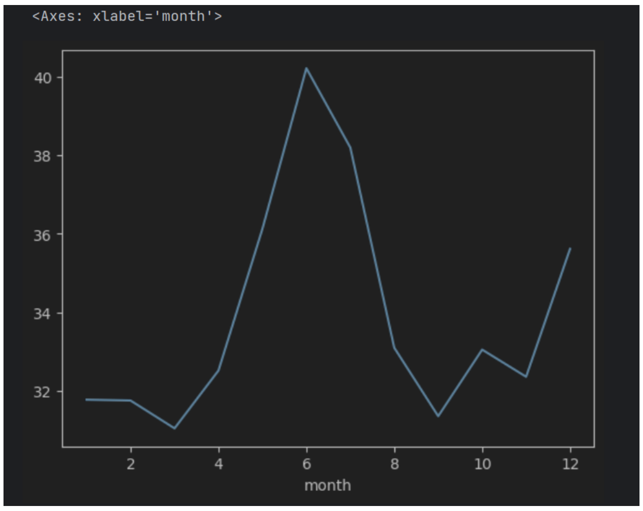
Problem2

```
# problem2
wind_data = pd.read_csv("2281305.csv")
#%%
# check and extract the needed data
wind_data.info()
wind_date_need=wind_data[["DATE", "WND"]]
#%%
wind_date_need
#%%
# split wnd into five columns
new_columns = wind_date_need['WND'].str.split(',', expand=True)
new_columns.columns = ['direction', 'type', 'dir', 'speed','quality']
wind_date_need = pd.concat([wind_date_need, new_columns], axis=1)
#%%
wind_date_need
#%%
# data filter
drop_data = wind_date_need[(wind_date_need['dir'] == '9') & (wind_date_need['speed'] != '0000')]
wind_date_need = wind_date_need.drop(drop_data.index)
drop_data = wind_date_need[(wind_date_need['speed'] == '9999')]
wind_date_need = wind_date_need.drop(drop_data.index)
drop_data = wind_date_need[(wind_date_need['quality'] != '1') & (wind_date_need['quality'] != '5')]
wind_date_need = wind_date_need.drop(drop_data.index)
wind_date_need
#%%
# transform speed from str to number and add a column month
wind_date_need['SPEED'] = pd.to_numeric(wind_date_need['speed'], errors='coerce')
wind_date_need['month'] = pd.to_datetime(wind_date_need['DATE']).dt.month
wind_date_need['SPEED'] = wind_date_need['SPEED'].astype(float)
wind_date_need
#%%
mean_speed_monthly = wind_date_need.groupby('month')['SPEED'].mean()
mean_speed_monthly
mean_speed_monthly.plot()
```

filter the data process:
1. extract the needed data column "DATA" and "WND"
2. according to the user guide, drop the missing data and others may be precise, those speed = 9999, speed=0 but WIND-OBSERVATION type!=9, and those WIND-OBSERVATION speed quality !=1 and 5

From the graph, we can observe that the monthly average wind speed shows a trend of increasing from January, reaching its peak in June, and then gradually decreasing until September. Afterward, it starts to rise again, reaching another lower peak in December.

output:



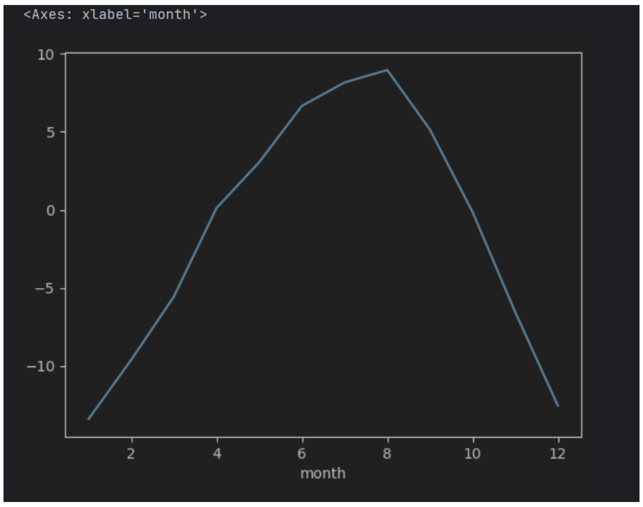
```
<Axes: xlabel='month'>
```

Problem3

```python
#problem3
Climate_data = pd.read_csv('2003-climate.csv')
Climate_data
#%%
# check if nan exist
nan_count = Climate_data['Temperature1'].count()
nan_count
#%%

#check and find the max record
max_indices = Climate_data[Climate_data['Temperature1'] == Climate_data['Temperature1'].max()].index
max_rows =Climate_data.loc[max_indices]
max_rows

#%%
#check and find the min record
min_indices = Climate_data[Climate_data['Temperature1'] == Climate_data['Temperature1'].min()].index
min_rows =Climate_data.loc[min_indices]
min_rows
#%%
#find the median
median = Climate_data['Temperature1'].median()
median


#%%
#find the mean
median = Climate_data['Temperature1'].mean()
median
#%%
std = Climate_data['Temperature1'].std()
std

#%%
mean_temperature1_monthly = Climate_data.groupby('month')['Temperature1'].mean()
mean_temperature1_monthly
mean_temperature1_monthly.plot()
```

Finding:
The mean temperature1 in 2003 = -1.27 °C
The median temperature1 in 2003 = -0.5345 °C
The max temperature1 in 2003 = 21.04 °C
The min temperature1 in 2003 = -28.66 °C
The std of temperature1 in 2003 = 9.69 °C

output:



```
<Axes: xlabel='month'>
```

Figure 1: Enter Caption