1. Significant earthquakes since 2150 B.C.

1.1 [5 points] Compute the total number of deaths caused by earthquakes since 2150 B.C. in each country, and then print the top 20 countries along with the total number of deaths.

1.2 [10 points] Compute the total number of earthquakes with magnitude larger than 3.0 (use column Ms as the magnitude) worldwide each year, and then plot the time series. Do you observe any trend? Explain why or why not?

1.3 [10 points] Write a function CountEq_LargestEq that returns (1) the total number of earthquakes since 2150 B.C. in a given country AND (2) date and location of the largest earthquake ever happened in this country. Apply CountEq_LargestEq to every country in the file, report your results in a descending order.

In [255…

```
# 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

#1.1 top 20 countries along with the total number of deaths
earthquake = pd.read_csv("Sig_Eqs.tsv",delimiter='\t')
earthquake.groupby(['Country']).sum()['Deaths'].sort_values(ascending=False
```

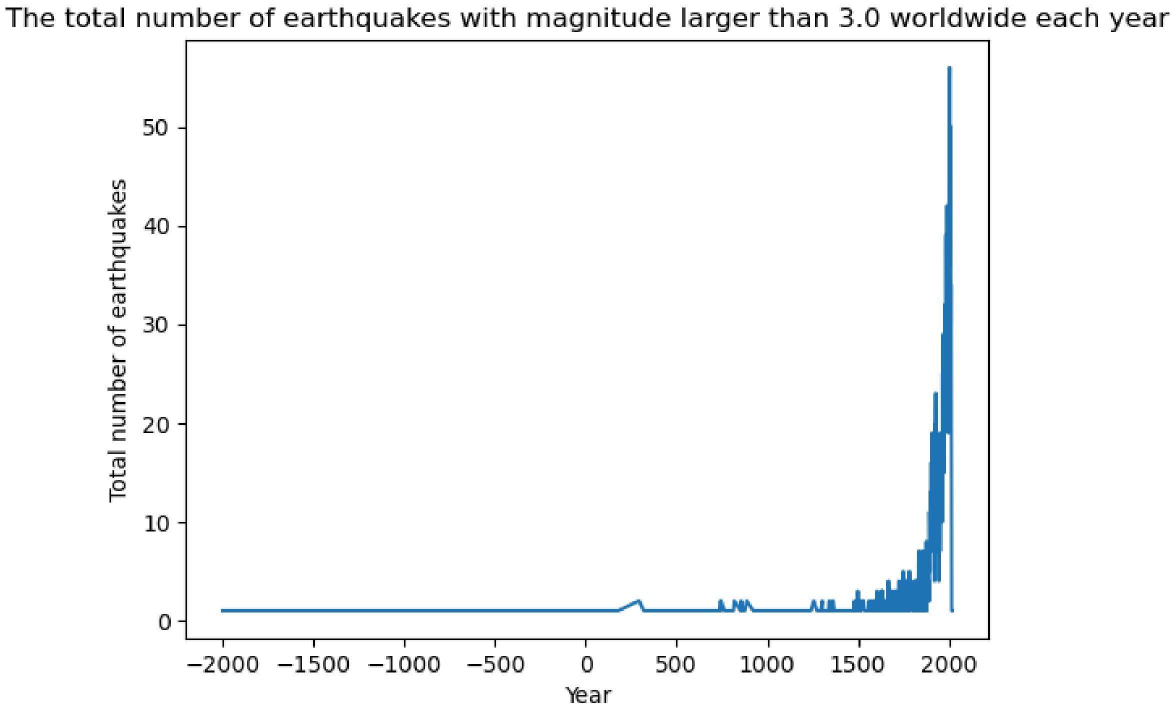
Out[255]:

```
Country
CHINA           2075947.0
TURKEY          1188881.0
IRAN            1011453.0
ITALY            498418.0
SYRIA            439224.0
HAITI            323478.0
AZERBAIJAN       317219.0
JAPAN            279607.0
ARMENIA          191890.0
PAKISTAN         145083.0
IRAQ             136200.0
ECUADOR          135496.0
TURKMENISTAN     117412.0
PERU             102169.0
ISRAEL            90388.0
PORTUGAL          83572.0
GREECE            80378.0
CHILE             64277.0
INDIA             63507.0
TAIWAN            57153.0
Name: Deaths, dtype: float64
```

In [42]:

```
#1.2
count=earthquake[(earthquake['Ms']>3.0)].groupby(['Year']).count()['Id']
count.plot()
plt.xlabel("Year")
plt.ylabel("Total number of earthquakes")
plt.title("The total number of earthquakes with magnitude larger than 3.0 wo
```

Out[42]:
```
Text(0.5, 1.0, 'The total number of earthquakes with magnitude larger than 3.0 worldwide each year')
```



trend:total number of earthquakes is increasing especially in recent 500 years, may be due to more sensitive and comprehensive seismic detection techniques to record the earthquakes happended

In [ ]:
```
#1.3
country=earthquake['Country']
#earthquake[earthquake['Country']==country]
#def CountEq_LargestEq():
    #country=str(input("Country:"))
    #total number
    #total=earthquake.groupby(['Country']).sum()['Deaths']
    country_total=total[country]
    #largest earthquake
    Ms_max=earthquake[earthquake['Country']==(country)].max()
    max_info=Ms_max['Year'],Ms_max['Mo'],Ms_max['Dy'],Ms_max['Location Name
    #Ms_max_info=str(Ms_max['Location Name'],str(Ms_max['Year'],Ms_max['Mo'
    #print(Ms_max['Location Name'],Ms_max['Year'],Ms_max['Mo'],Ms_max['Dy'],
    return country_total, max_info
#CountEq_LargestEq()
#total_number_of_earthquakes=[country_total]
#earthquakes_info=[max_info]
```

In [396…
```
def CountEq_LargestEq(country):
#total number
    total=earthquake.groupby(['Country']).sum()['Deaths'].get(country, 0)
#largest earthquake
    Ms_max=earthquake.groupby(['Country'])['Ms'].max().get(country, None)
#select the information
    if Ms_max is not None:
        max_info = earthquake[(earthquake['Country'] == country) & (earthqua
#deal with the missing information
        if not max_info.empty:
            max_information = (max_info['Year'].iloc[0], max_info['Mo'].iloc
                               max_info['Dy'].iloc[0], max_info['Location Na
        else:
            max_information = (None, None, None, None)
```

```
    else:
        max_information = (None, None, None, None)
    return total,max_information
#apply to every country
all_country = pd.DataFrame(columns=['Country', 'TotalDeaths','Year','Month',
all_info = []
for country in earthquake['Country'].unique():
    total, max_info = CountEq_LargestEq(country)
    info = {
        'Country': country,
        'TotalDeaths': total,
        'Year': max_info[0],
        'Month': max_info[1],
        'Date': max_info[2],
        'Location': max_info[3]
    }
    all_info.append(info)
all_country = pd.DataFrame(all_info)
all_country
all_country.sort_values("TotalDeaths", ascending=False)
```

Out[396]:

	Country	TotalDeaths	Year	Month	Date	Location
15	CHINA	2075947.0	1920.0	12.0	16.0	CHINA: GANSU PROVINCE, SHANXI PROVINCE
10	TURKEY	1188881.0	1939.0	12.0	26.0	TURKEY: ERZINCAN
8	IRAN	1011453.0	856.0	12.0	22.0	IRAN: DAMGHAN, QUMIS
6	ITALY	498418.0	1915.0	1.0	13.0	ITALY: MARSICA, AVEZZANO, ABRUZZI
2	SYRIA	439224.0	1202.0	5.0	20.0	SYRIA: SOUTHWESTERN
...	...	...	...	...	...	...
120	KIRIBATI	0.0	NaN	NaN	NaN	None
122	CAMEROON	0.0	NaN	NaN	NaN	None
124	MICRONESIA, FED. STATES OF	0.0	1911.0	8.0	16.0	MICRONESIA, FED. STATES OF: CAROLINE ISLANDS
126	PALAU	0.0	1914.0	10.0	23.0	MICRONESIA, FED. STATES OF: CAROLINE ISLANDS
157	COMOROS	0.0	NaN	NaN	NaN	None

158 rows × 6 columns

2. Air temperature in Shenzhen during the past 25 years

Explain how you filter the data in your report. [10 points] Plot monthly averaged air temperature against the observation time. Is there a trend in monthly averaged air temperature in the past 25 years?

fliter the data: delete the data 1）which TMP out the range of -0932 to +0618 2）quality code with suspect and erroneous and divided by scaling factor(10) to get the actual temperature

In [397…

```
temperature_change = pd.read_csv("Baoan_Weather_1998_2022.csv",low_memory=Fa
#split the TMP data to temperature and quality code
temperature_change[['tmp', 'code']] = temperature_change['TMP'].str.split(',
temperature_change['code'] = pd.to_numeric(temperature_change['code'])
temperature_change['tmp'] = pd.to_numeric(temperature_change['tmp'].str.rep
#filter the bad value
temperature_change=temperature_change[(temperature_change['tmp']>-9.32)&(ter
temperature_change[(temperature_change['code']==1)]
```

Out[397]:

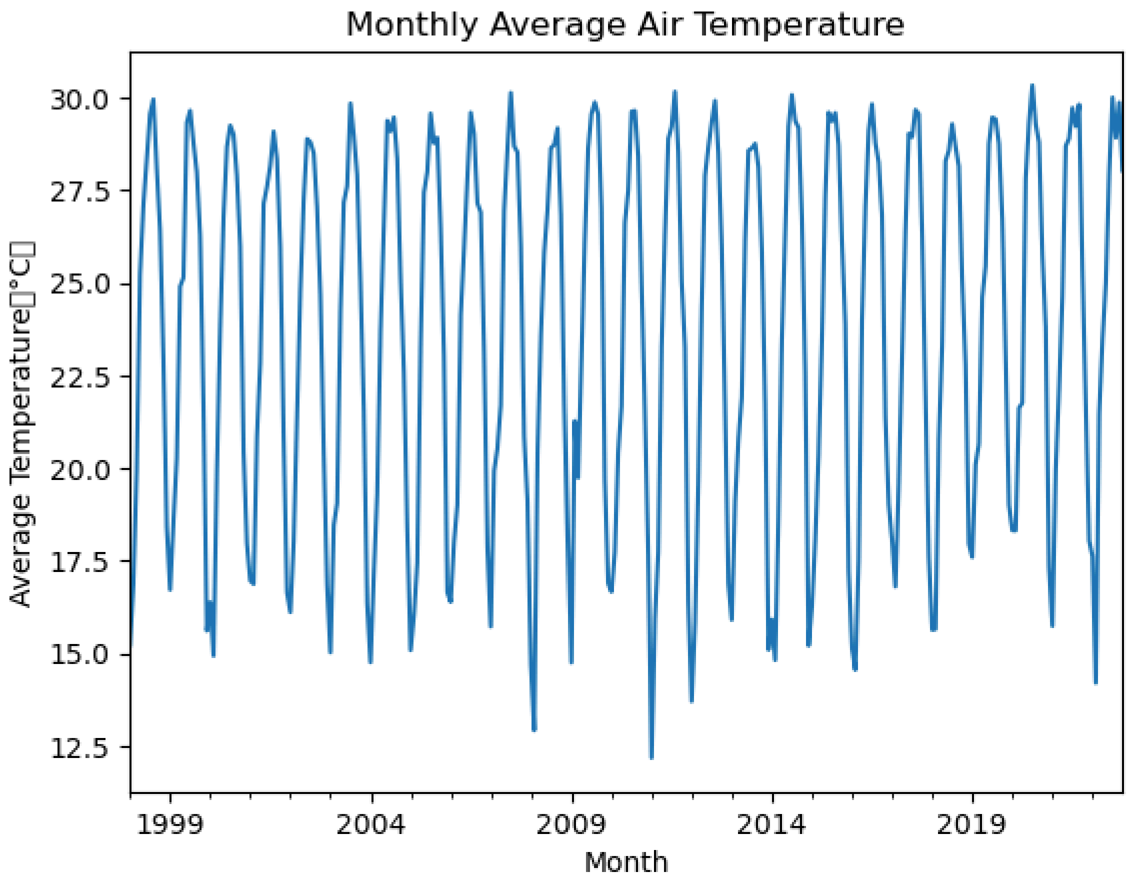
	STATION	DATE	SOURCE	REPORT_TYPE	CALL_SIGN	QUALITY_CONTI
0	59493099999	1998-01-01T00:00:00	4	SY-MT	ZGSZ	V
1	59493099999	1998-01-01T01:00:00	4	FM-15	ZGSZ	V
2	59493099999	1998-01-01T02:00:00	4	FM-15	ZGSZ	V
3	59493099999	1998-01-01T03:00:00	4	SY-MT	ZGSZ	V
4	59493099999	1998-01-01T04:00:00	4	FM-15	ZGSZ	V
...	...	...	...	...	...	
235669	59493099999	2022-10-10T20:00:00	4	FM-15	99999	V
235670	59493099999	2022-10-10T21:00:00	4	FM-12	99999	V
235671	59493099999	2022-10-10T21:00:00	4	FM-15	99999	V
235672	59493099999	2022-10-10T22:00:00	4	FM-15	99999	V
235673	59493099999	2022-10-10T23:00:00	4	FM-15	99999	V

234863 rows × 56 columns

In [8]:

```
#converts time string to datetime with month periodic
temperature_change['date'] = pd.to_datetime(temperature_change['DATE'])
temperature_change['month'] = temperature_change['date'].dt.to_period('M')
temperature_change.head()
#group by and plot monthly averaged air temperature
monthly_avg_temp = temperature_change.groupby('month')['tmp'].mean()
monthly_avg_temp.plot()
plt.title('Monthly Average Air Temperature')
plt.xlabel('Month')
plt.ylabel('Average Temperature (°C) ')
plt.show()
```

```
/Users/apple/anaconda3/lib/python3.11/site-packages/IPython/core/pylabtool
s.py:152: UserWarning: Glyph 65288 (\N{FULLWIDTH LEFT PARENTHESIS}) missing
from current font.
  fig.canvas.print_figure(bytes_io, **kw)
/Users/apple/anaconda3/lib/python3.11/site-packages/IPython/core/pylabtool
s.py:152: UserWarning: Glyph 65289 (\N{FULLWIDTH RIGHT PARENTHESIS}) missin
g from current font.
  fig.canvas.print_figure(bytes_io, **kw)
```



There is no obvious trend of monthly averaged air temperature change, but it changes regularly on a yearly basis.

3. Global collection of hurricanes

3.1 [5 points] Group the data on Storm Identifie (SID), report names (NAME) of the 10 largest hurricanes according to wind speed (WMO_WIND).

3.2 [5 points] Make a bar chart of the wind speed (WMO_WIND) of the 20 strongest-wind hurricanes.

3.3 [5 points] Plot the count of all datapoints by Basin as a bar chart.

3.4 [5 points] Make a hexbin plot of the location of datapoints in Latitude and Longitude.

3.5 [5 points] Find Typhoon Mangkhut (from 2018) and plot its track as a scatter plot.

3.6 [5 points] Create a filtered dataframe that contains only data since 1970 from the Western North Pacific ("WP") and Eastern North Pacific ("EP") Basin. Use this for the rest of the problem set.

3.7 [5 points] Plot the number of datapoints per day.

3.8 [5 points] Calculate the climatology of datapoint counts as a function of day of year. The day of year is the sequential day number starting with day 1 on January 1st.

3.9 [5 points] Calculate the anomaly of daily counts from the climatology.

3.10 [5 points] Resample the anomaly timeseries at annual resolution and plot. So which years stand out as having anomalous hurricane activity?

In [25]:
```
#example
df = pd.read_csv('ibtracs.ALL.list.v04r00.csv',
                 usecols=range(17),
                 skiprows=[1, 2],
                 parse_dates=['ISO_TIME'],
                 na_values=['NOT_NAMED', 'NAME'])
df.head()
```

```
/var/folders/yx/js3jvr652bx7g_xn_lgjpcy40000gn/T/ipykernel_48599/126128260
1.py:2: DtypeWarning: Columns (5) have mixed types. Specify dtype option on
import or set low_memory=False.
  df = pd.read_csv('ibtracs.ALL.list.v04r00.csv',
```

Out[25]:

	SID	SEASON	NUMBER	BASIN	SUBBASIN	NAME	ISO_TIME	NATURE	L
0	1842298N11080	1842	1	NI	BB	NaN	1842-10-25 06:00:00	NR	10.87
1	1842298N11080	1842	1	NI	BB	NaN	1842-10-25 09:00:00	NR	10.84
2	1842298N11080	1842	1	NI	BB	NaN	1842-10-25 12:00:00	NR	10.81
3	1842298N11080	1842	1	NI	BB	NaN	1842-10-25 15:00:00	NR	10.80
4	1842298N11080	1842	1	NI	AS	NaN	1842-10-25 18:00:00	NR	10.78

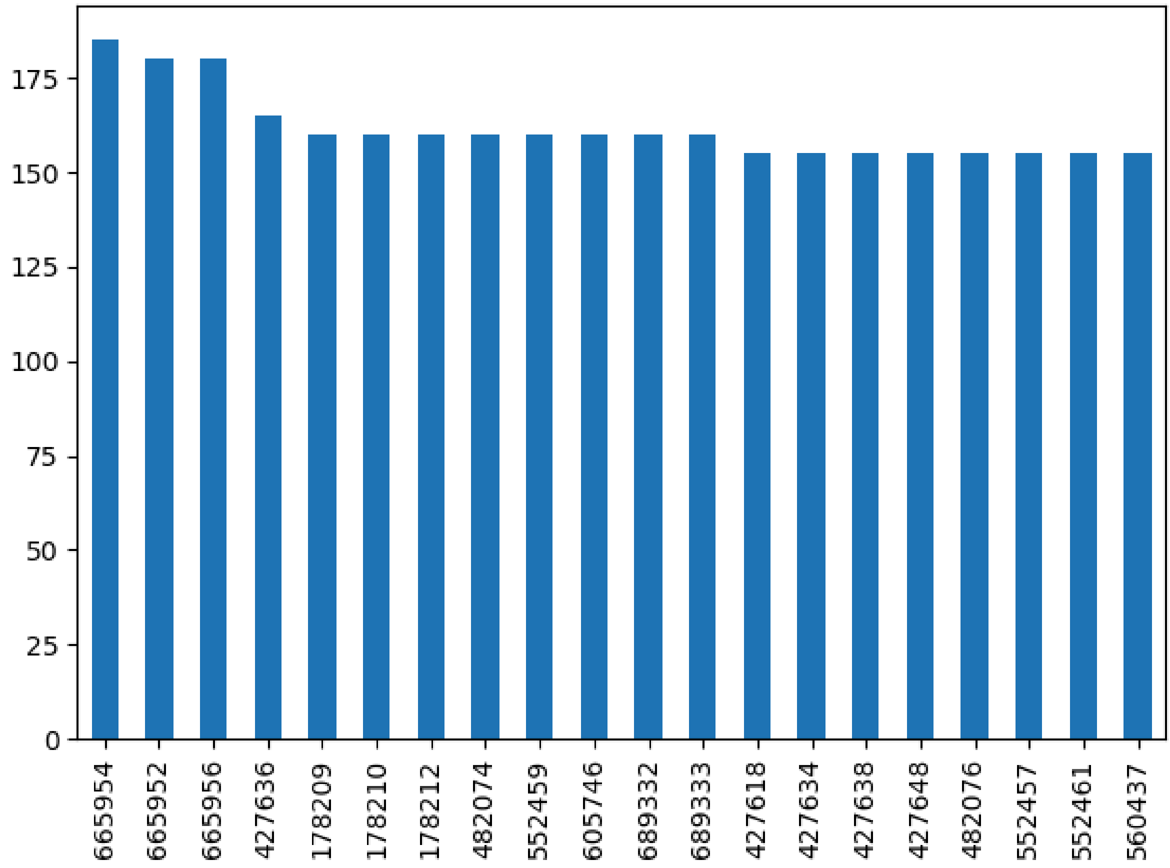
In [424…
```
#3.1
grouped = df.groupby('SID')
df.sort_values('WMO_WIND',ascending=False).head(10)['NAME']
```

Out[424]:
```
665954    PATRICIA
665952    PATRICIA
665956    PATRICIA
427636       ALLEN
178212         NaN
178210         NaN
178209         NaN
552459       LINDA
605746       WILMA
482074     GILBERT
Name: NAME, dtype: object
```

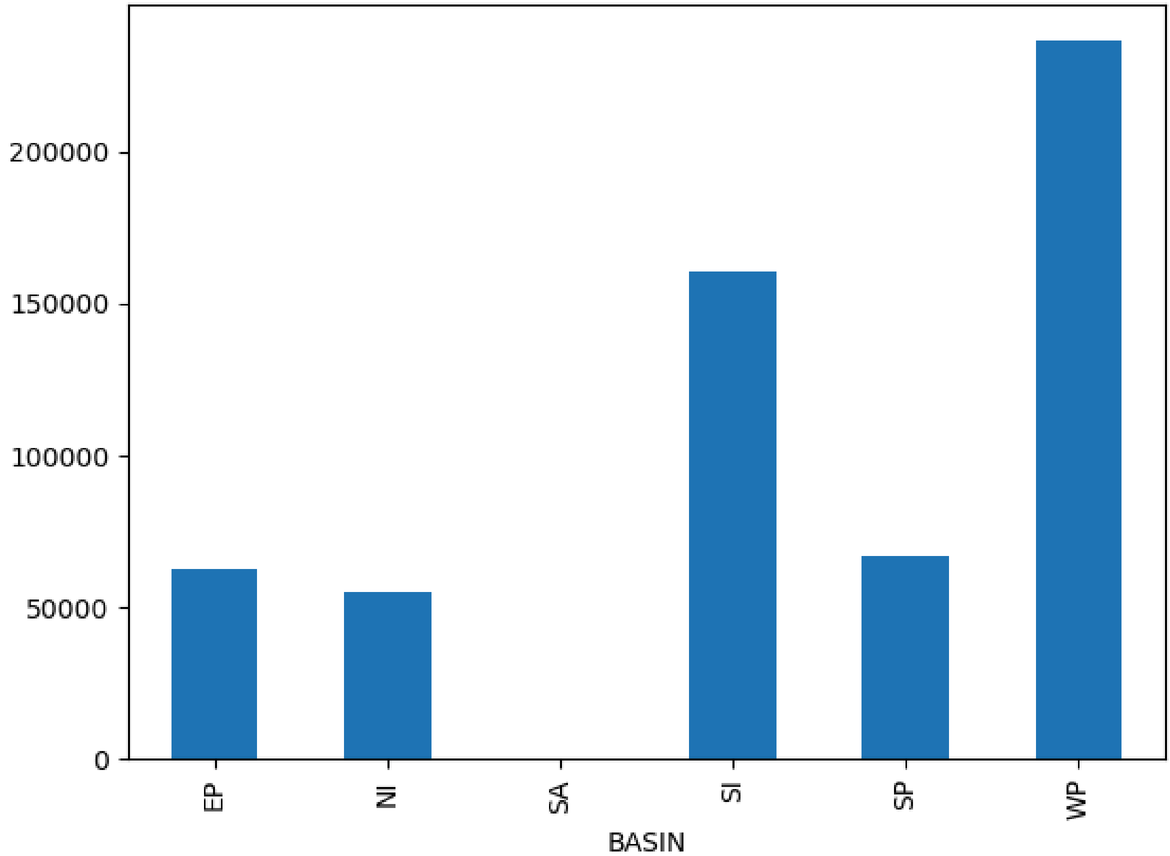
In [408…
```
print(df['WMO_WIND'].dtype)
```

```
object
```

In [426…
```
#3.2
#strongest_wind=df.sort_values('WMO_WIND',ascending=False).head(20)
df['WMO_WIND']=pd.to_numeric(df['WMO_WIND'],errors='coerce')
strongest_wind=df['WMO_WIND'].nlargest(20)
```

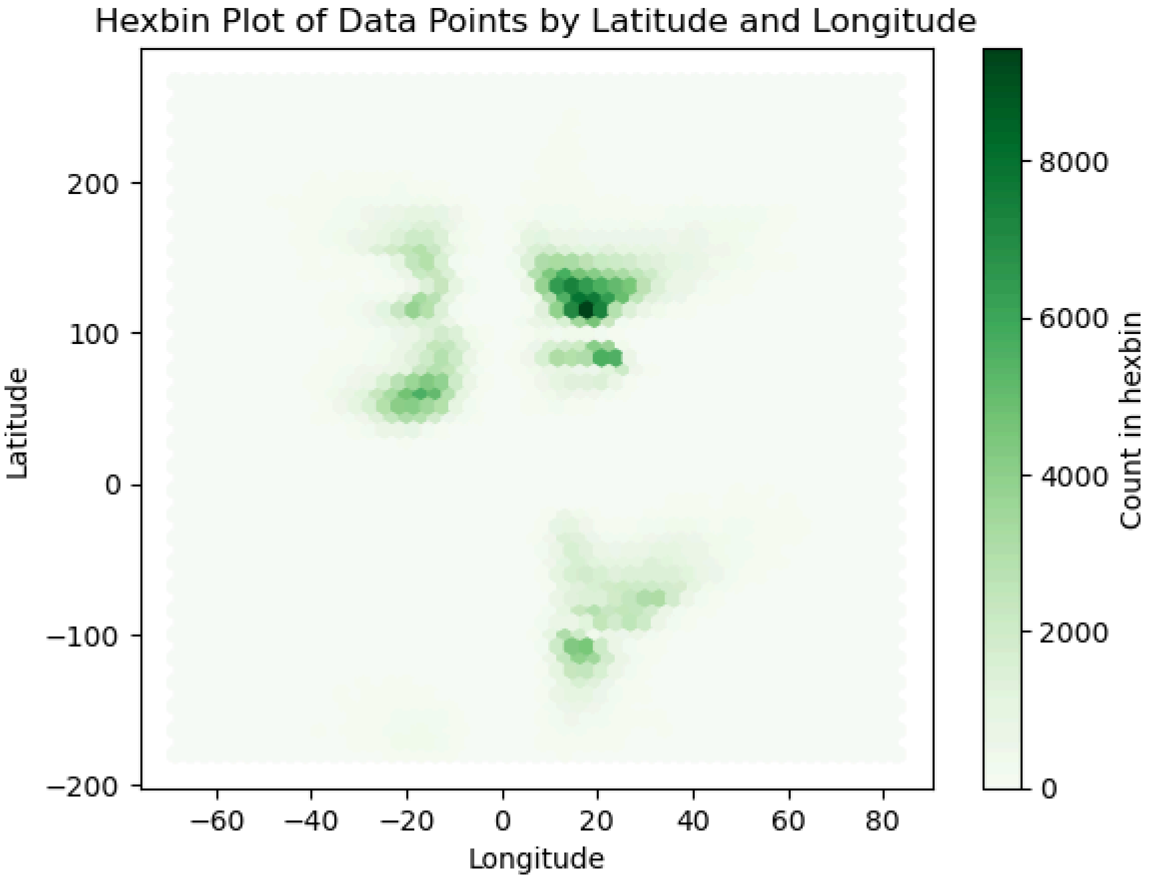


In [28]:
```
#3.3
(df.groupby(['BASIN']).count())['SID'].plot(kind='bar')
plt.tight_layout()
```

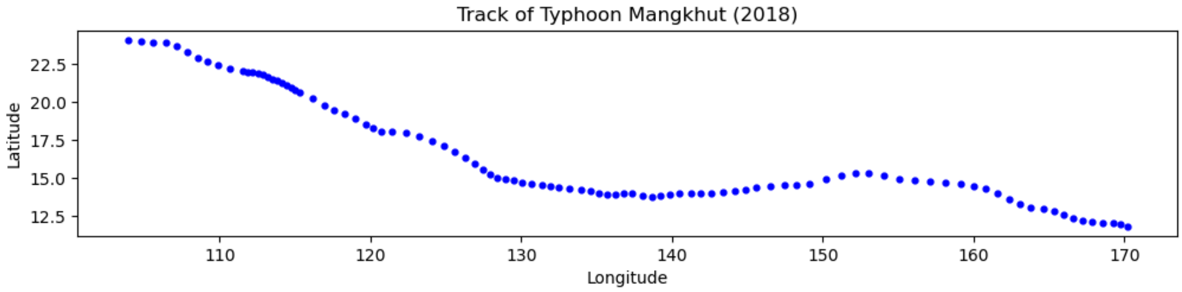


BASIN EP 62412 NI 55401 SA 119 (Too little quantity is not shown in the figure！) SI 160668 SP 67119 WP 236576

In [29]:
```
#3.4
hb = plt.hexbin(df['LAT'], df['LON'], gridsize=50, cmap='Greens')
cb = plt.colorbar(hb)
cb.set_label('Count in hexbin')
plt.xlabel('Longitude')
plt.ylabel('Latitude')
plt.title('Hexbin Plot of Data Points by Latitude and Longitude')
plt.show()
```



In [30]:
```
#3.5
MANGKHUT_2018=df.loc[(df['NAME'].str.contains('MANGKHUT',na=False))&(df['IS(
MANGKHUT_2018
fig, ax = plt.subplots(figsize=(12, 6))
ax.scatter(MANGKHUT_2018['LON'], MANGKHUT_2018['LAT'], c='blue', s=12, labe
ax.set_aspect('equal')
ax.set_xlabel('Longitude')
ax.set_ylabel('Latitude')
ax.set_title('Track of Typhoon Mangkhut (2018)')
plt.show()
```



In [89]:
```
#3.6
filtered_df=df.loc[(df['SEASON']>=1970)&(df['BASIN'].str.contains('EP|WP'))]
filtered_df
```

Out[89]:

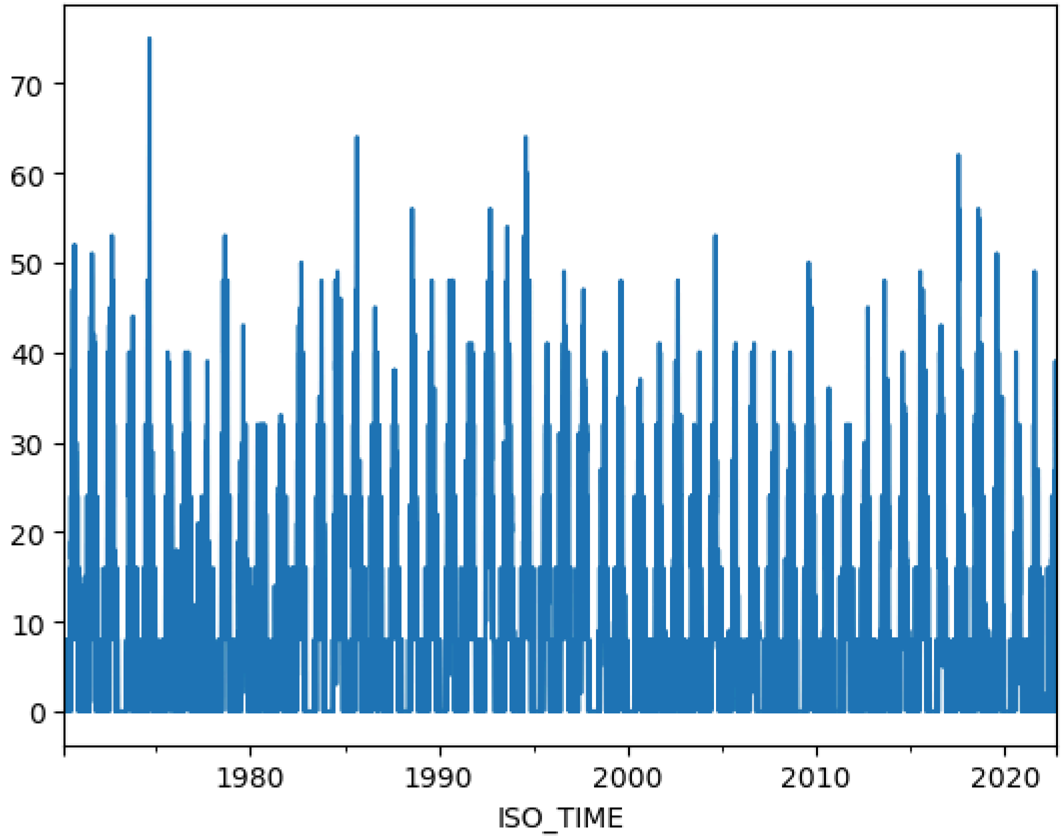
	SID	SEASON	NUMBER	BASIN	SUBBASIN	NAME	ISO_TIME	NATURE
350393	1970050N07151	1970	22	WP	MM	NANCY	1970-02-19 00:00:00	TS
350394	1970050N07151	1970	22	WP	MM	NANCY	1970-02-19 03:00:00	TS
350395	1970050N07151	1970	22	WP	MM	NANCY	1970-02-19 06:00:00	TS
350396	1970050N07151	1970	22	WP	MM	NANCY	1970-02-19 09:00:00	TS
350397	1970050N07151	1970	22	WP	MM	NANCY	1970-02-19 12:00:00	TS
...	...	...	...	...	...	...	...	..
707084	2022275N10316	2022	76	EP	MM	JULIA	2022-10-10 15:00:00	TS
707085	2022275N10316	2022	76	EP	MM	JULIA	2022-10-10 18:00:00	NR
707173	2022286N15151	2022	80	WP	MM	NaN	2022-10-12 12:00:00	NR
707174	2022286N15151	2022	80	WP	MM	NaN	2022-10-12 15:00:00	NR
707175	2022286N15151	2022	80	WP	MM	NaN	2022-10-12 18:00:00	NR

176352 rows × 17 columns

In [122…

```
#3.7 per day
datapoint_day= filtered_df.groupby(pd.Grouper(key='ISO_TIME', freq='D')).si:
datapoint_day.plot()
```

Out[122]: <Axes: xlabel='ISO_TIME'>



In [123…
```
datapoint_day = datapoint_day.reset_index()
datapoint_day.columns=['Date', 'Datapoint_Counts']
datapoint_day
```

Out[123]:

	Date	Datapoint_Counts
0	1970-02-19	8
1	1970-02-20	8
2	1970-02-21	8
3	1970-02-22	8
4	1970-02-23	8
...	...	...
19224	2022-10-08	0
19225	2022-10-09	1
19226	2022-10-10	7
19227	2022-10-11	0
19228	2022-10-12	3

19229 rows × 2 columns

In [220…
```
datapoint_day['Day_of_Year']=datapoint_day['Date'].dt.dayofyear
datapoint_day
```

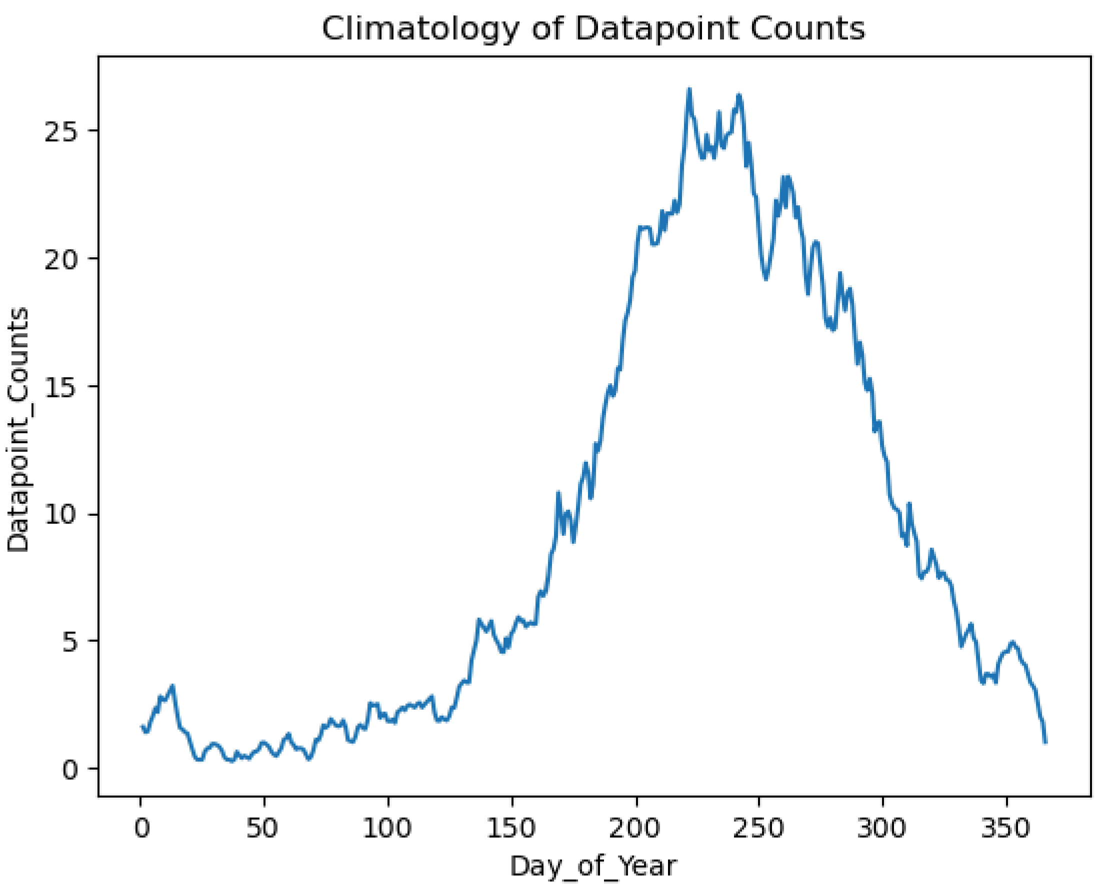
Out[220]:

	Date	Datapoint_Counts	Day_of_Year	anomaly_daily
0	1970-02-19	8	50	7.018868
1	1970-02-20	8	51	7.094340
2	1970-02-21	8	52	7.188679
3	1970-02-22	8	53	7.377358
4	1970-02-23	8	54	7.490566
...	...	...	...	...
19224	2022-10-08	0	281	-17.245283
19225	2022-10-09	1	282	-17.301887
19226	2022-10-10	7	283	-12.396226
19227	2022-10-11	0	284	-18.528302
19228	2022-10-12	3	285	-14.924528

19229 rows × 4 columns

In [143…

```
#3.8
climatology=datapoint_day.groupby(['Day_of_Year']).mean()['Datapoint_Counts
climatology.plot()
plt.xlabel('Day_of_Year')
plt.ylabel('Datapoint_Counts')
plt.title('Climatology of Datapoint Counts')
plt.show()
climatology
```



Out[143]:
```
Day_of_Year
1      1.596154
2      1.384615
3      1.423077
4      1.788462
5      2.019231
         ...
362    3.038462
363    2.538462
364    2.000000
365    1.788462
366    1.000000
Name: Datapoint_Counts, Length: 366, dtype: float64
```

In [194…
```
day_count=datapoint_day['Datapoint_Counts'].astype(float)
day_count
```

Out[194]:
```
0        8.0
1        8.0
2        8.0
3        8.0
4        8.0
        ...
19224    0.0
19225    1.0
19226    7.0
19227    0.0
19228    3.0
Name: Datapoint_Counts, Length: 19229, dtype: float64
```

In [199…
```
daily=climatology[datapoint_day['Day_of_Year']]
daily = daily.reset_index()
daily.colomns=['Day_of_Year','Datapoint_Counts']
daily
```

```
/var/folders/yx/js3jvr652bx7g_xn_lgjpcy40000gn/T/ipykernel_48599/120541413
3.py:3: UserWarning: Pandas doesn't allow columns to be created via a new a
ttribute name - see https://pandas.pydata.org/pandas-docs/stable/indexing.h
tml#attribute-access
  daily.colomns=['Day_of_Year','Datapoint_Counts']
```

Out[199]:

	Day_of_Year	Datapoint_Counts
0	50	0.981132
1	51	0.905660
2	52	0.811321
3	53	0.622642
4	54	0.509434
...	...	...
19224	281	17.245283
19225	282	18.301887
19226	283	19.396226
19227	284	18.528302
19228	285	17.924528

19229 rows × 2 columns

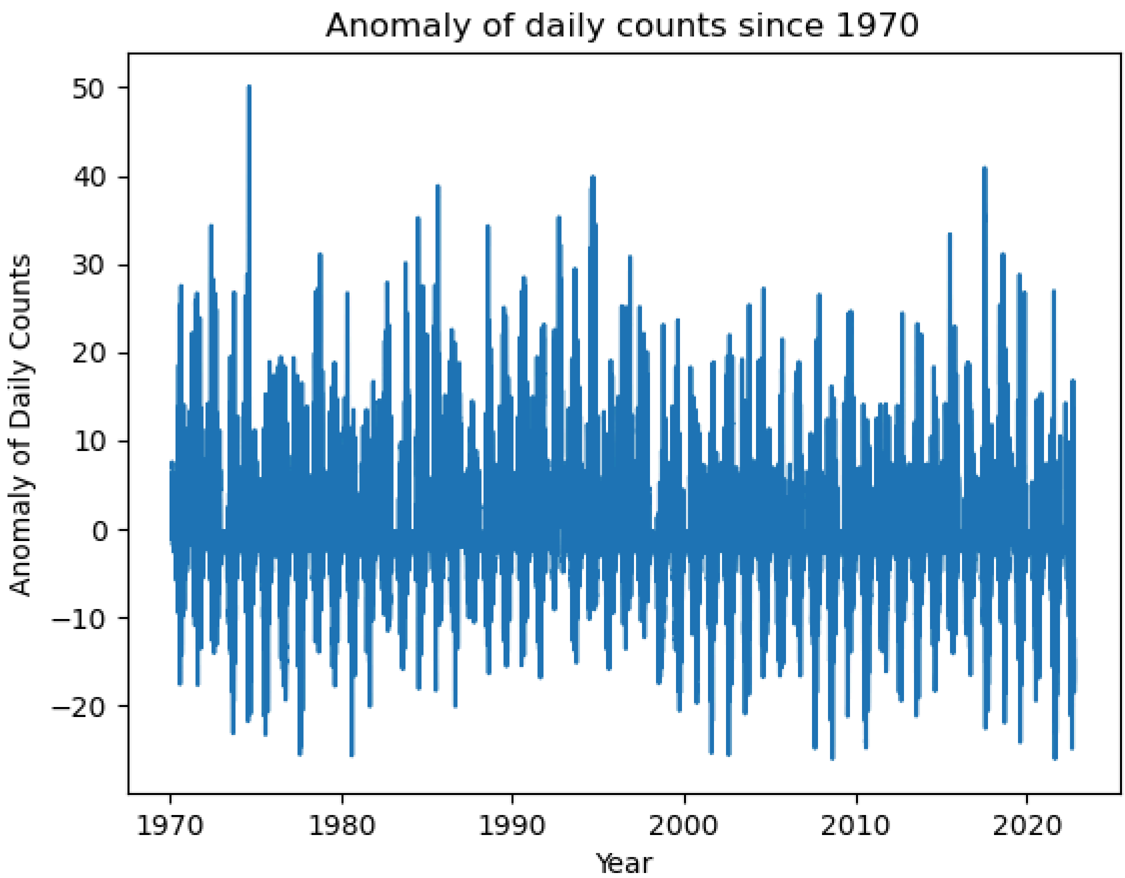
In [240…

```
#3.9
datapoint_day['anomaly_daily']=day_count-daily['Datapoint_Counts']
plt.plot(datapoint_day['Date'],datapoint_day['anomaly_daily'])
plt.title('Anomaly of daily counts since 1970')
plt.xlabel('Year')
plt.ylabel('Anomaly of Daily Counts')
datapoint_day
```

Out[240]:

	Date	Datapoint_Counts	Day_of_Year	anomaly_daily
0	1970-02-19	8	50	7.018868
1	1970-02-20	8	51	7.094340
2	1970-02-21	8	52	7.188679
3	1970-02-22	8	53	7.377358
4	1970-02-23	8	54	7.490566
...	...	...	...	...
19224	2022-10-08	0	281	-17.245283
19225	2022-10-09	1	282	-17.301887
19226	2022-10-10	7	283	-12.396226
19227	2022-10-11	0	284	-18.528302
19228	2022-10-12	3	285	-14.924528

19229 rows × 4 columns



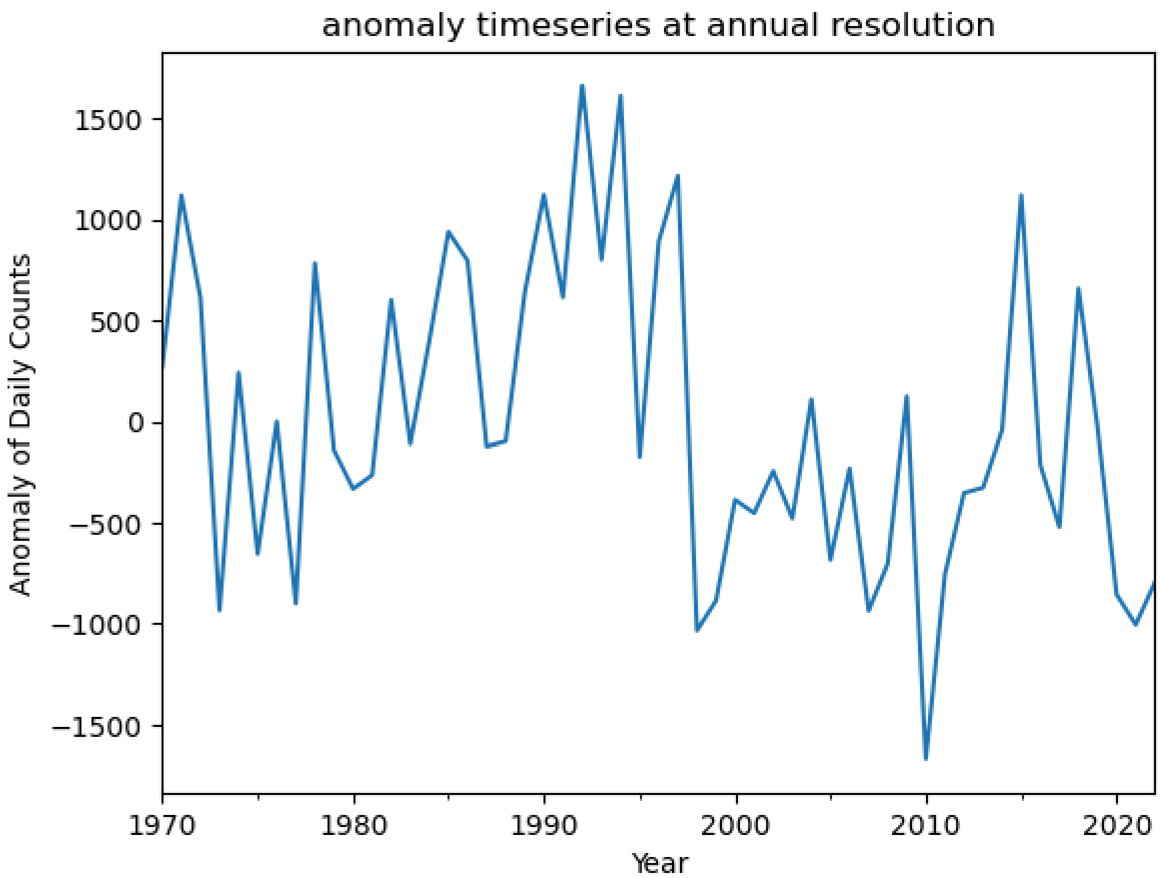
In [245…

```
#3.10
datapoint_day_new=datapoint_day[['Date','anomaly_daily']]
datapoint_day_new['Date']
datapoint_day_new.set_index('Date', inplace=True)
```

```
print(type(datapoint_day_new.index))
anomaly_annual=datapoint_day_new['anomaly_daily'].resample('A').sum()
anomaly_annual.plot()
plt.xlabel('Year')
plt.ylabel('Anomaly of Daily Counts')
plt.title('anomaly timeseries at annual resolution')
```

```
<class 'pandas.core.indexes.datetimes.DatetimeIndex'>
```

Out[245]:
```
Text(0.5, 1.0, 'anomaly timeseries at annual resolution')
```



1917,1990,1992,1994,1997,2010,2015 stand out as having anomalous hurricane activity

4. Explore a data set

4.1 [5 points] Load the csv, XLS, or XLSX file, and clean possible data points with missing values or bad quality.

4.2 [5 points] Plot the time series of a certain variable.

4.3 [5 points] Conduct at least 5 simple statistical checks with the variable, and report your findings.

In [6]:
```
#4.1 load and clean data
air= pd.read_csv("Chengdu People's Park.csv")
print(air.isnull().sum())
filtered_air=air.dropna()
print(filtered_air.isnull().sum())
```

```
Time       0
PM2_5     29
CO        31
PM10     110
SO2       31
O3        35
NO2       33
dtype: int64
Time     0
PM2_5    0
CO       0
PM10     0
SO2      0
O3       0
NO2      0
dtype: int64
```

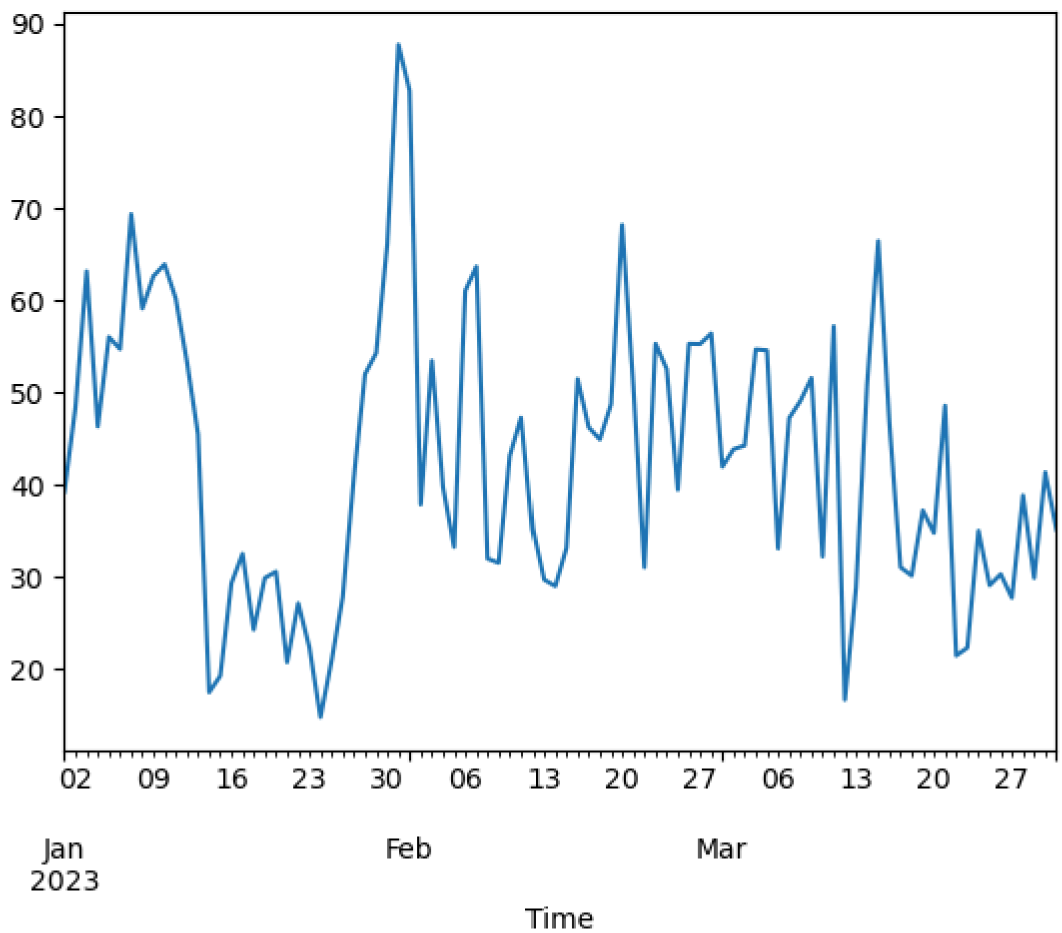
Data resources from： China Meteorological Data website

In [7]:
```
#4.2 time series of NO2
filtered_air['Time'] = pd.to_datetime(filtered_air['Time'])
NO2=filtered_air.groupby(pd.Grouper(key='Time', freq='D')).mean()['NO2']
NO2.plot()
```

```
/var/folders/yx/js3jvr652bx7g_xn_lgjpcy40000gn/T/ipykernel_36146/220244945
2.py:2: 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-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  filtered_air['Time'] = pd.to_datetime(filtered_air['Time'])
```

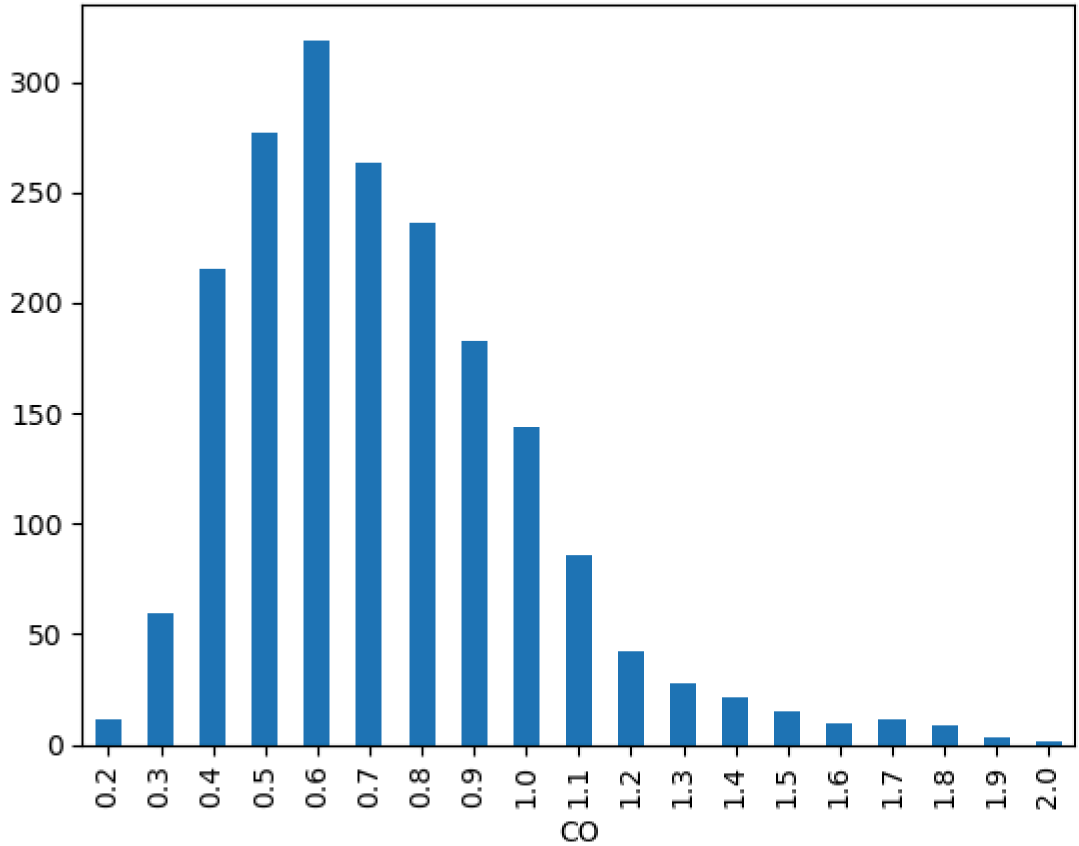
Out[7]: <Axes: xlabel='Time'>



In [9]:
```
#4.3 statistical checks
from scipy import stats
```

In [21]:
```
#check the CO value
CO=(filtered_air.groupby(['CO'])).count()['Time']
CO.plot(kind='bar')
```

Out[21]: <Axes: xlabel='CO'>



In [27]:
```
#Normality Test for CO value
stat, p_value = stats.shapiro(filtered_air['CO'])
print("p-value:", p_value)
```

```
p-value: 1.8113189066396898e-29
```

p<0.05 fit the normal distribution

In [28]:
```
#Independent samples t-test for PM2.5 and PM10
t_stat, p_value = stats.ttest_ind(filtered_air['PM2_5'],filtered_air['PM10']
print("p-value:", p_value)
```

```
p-value: 4.652913931253025e-68
```

In [31]:
```
#Paired samples t-test for PM2.5 and PM10
t_stat, p_value = stats.ttest_rel(filtered_air['PM2_5'],filtered_air['PM10']
print("p-value:", p_value)
```

```
p-value: 0.0
```

In [33]:
```
#Pearson's correlation for PM2.5 and PM10
correlation, p_value = stats.pearsonr(filtered_air['PM2_5'],filtered_air['PM
print("Correlation coefficient:", correlation)
print("p-value:", p_value)
```

```
Correlation coefficient: 0.8953445747838471
p-value: 0.0
```

PM2_5 and PM10 has linear correlation