In [1]:

```
#1.1
import pandas as pd
Sig_Eqs = pd.read_csv("earthquakes-2025-11-06_20-59-25_+0800.tsv",sep="\t")
#计算各个国家死亡人数总和
death_country = Sig_Eqs.groupby('Country')['Deaths'].sum()
#排序
death_country_top10 = death_country.sort_values(ascending = False).head(10)
death_country_top10
```

Out[1]:

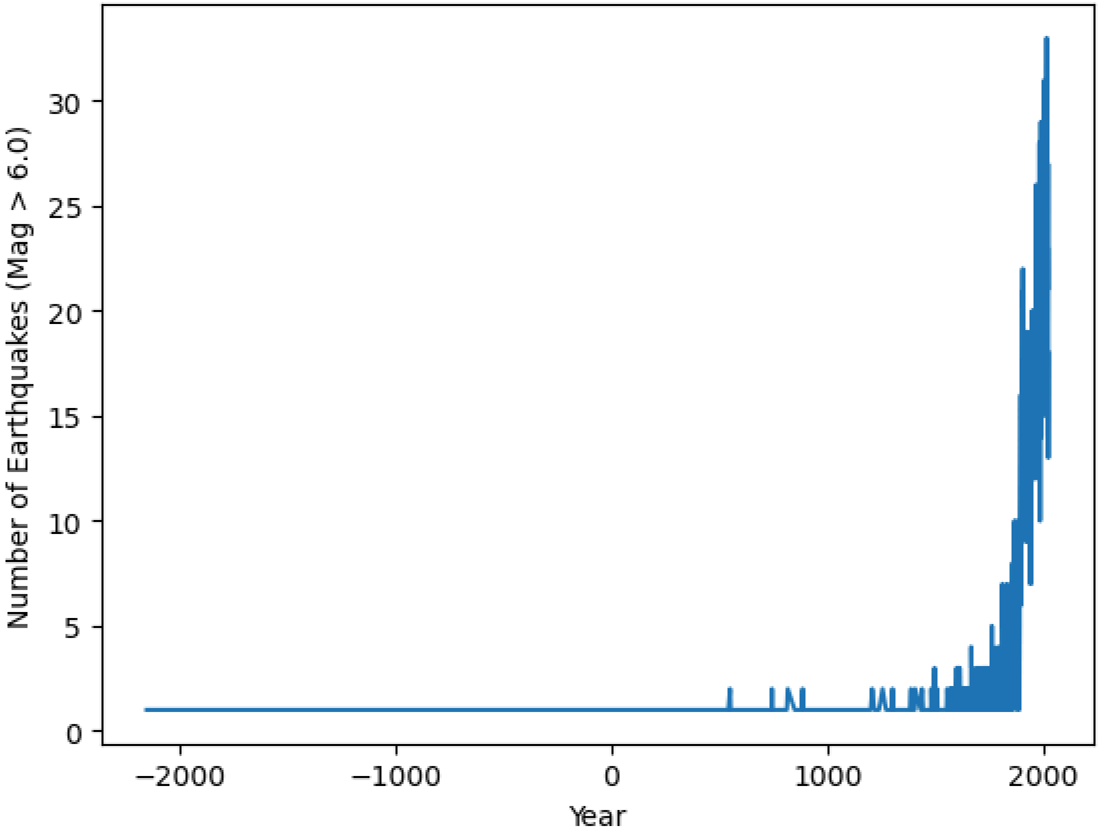
```
Country
CHINA         2139210.0
TURKEY        1199742.0
IRAN          1014453.0
ITALY          498219.0
SYRIA          419226.0
HAITI          323484.0
AZERBAIJAN     319251.0
JAPAN          242445.0
ARMENIA        191890.0
PAKISTAN       145083.0
Name: Deaths, dtype: float64
```

In [3]:

```
#1.2
import matplotlib.pyplot as plt
#筛选大于6的震级
L6_Eqs = Sig_Eqs[Sig_Eqs['Mag'] >6.0]
#按照年份分组
L6_Eqs_years = L6_Eqs.groupby('Year').size()
#时间序列图

L6_Eqs_years.plot(kind='line')
plt.ylabel('Number of Earthquakes (Mag > 6.0)')
```

Out[3]:

```
Text(0, 0.5, 'Number of Earthquakes (Mag > 6.0)')
```

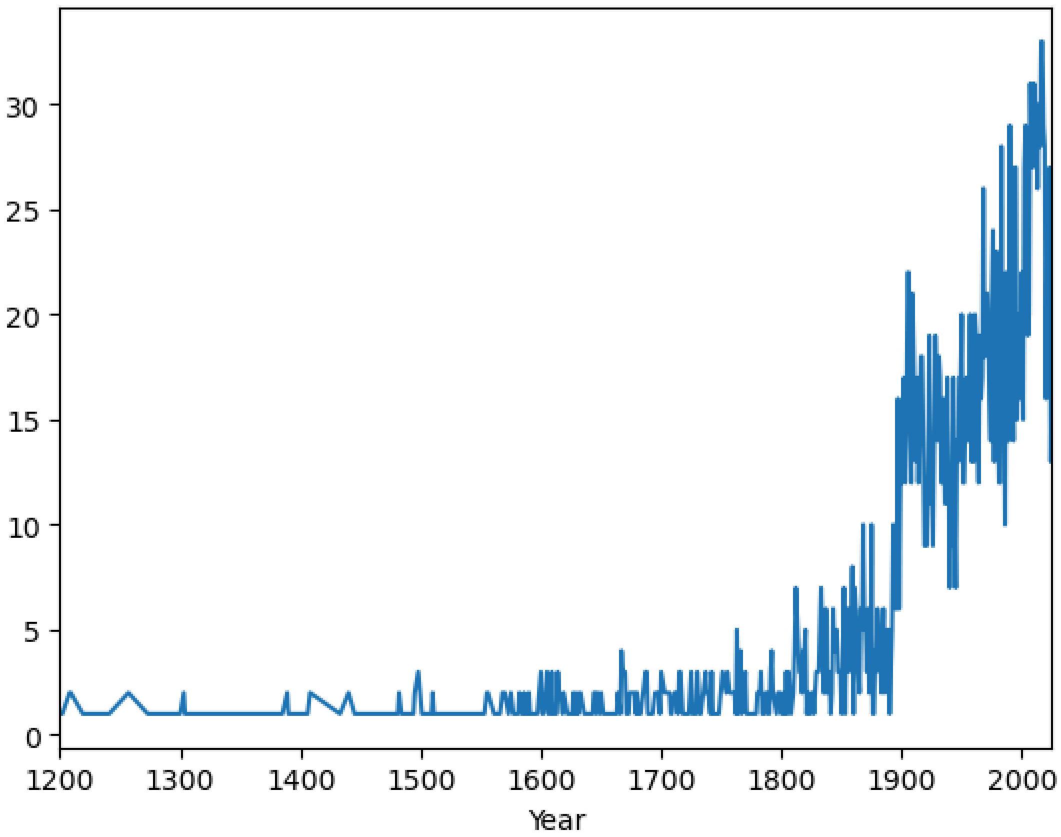


In [5]:

```
#放大观察趋势
L6_Eqs_years.plot().set_xlim(1200, 2025)
```

Out[5]:

```
(1200.0, 2025.0)
```



In [7]:

```python
#1.3
#给定国家，返回（总地震次数、该国最大地震的日期）
def CountEq_LargestEq(country):
    # 该国家的全部记录
    sub = Sig_Eqs[Sig_Eqs['Country'] == country]
    total = len(sub)

    # 没有任何记录
    if total == 0:
        return pd.Series({'Total_Eq': 0, 'Largest_Eq_Date': pd.NA})

    # 去掉震级缺失的行；若全缺失，则无法给出“最大地震日期”
    sub_nonan = sub[sub['Mag'].notna()]
    if len(sub_nonan) == 0:
        return pd.Series({'Total_Eq': total, 'Largest_Eq_Date': pd.NA})

    # 用 idxmax 找到最大震级所在行（避免与 NaN 比较）
    idx = sub_nonan['Mag'].idxmax()
    row = sub.loc[idx]

    # 组装日期（若月/日缺失，用 1 占位；若年缺失，则返回 NA）
    if pd.notna(row['Year']):
        year  = int(row['Year'])
        month = int(row['Mo']) if pd.notna(row['Mo']) else 1
        day   = int(row['Dy']) if pd.notna(row['Dy']) else 1
        date_str = f"{year:04d}-{month:02d}-{day:02d}"
    else:
        date_str = pd.NA
    return pd.Series({'Total_Eq': total, 'Largest_Eq_Date': date_str})


# 对文件中的每个国家应用该函数，并按总数降序输出
countries = Sig_Eqs['Country'].unique()
results = []

for i in countries:
    stats = CountEq_LargestEq(i)
    results.append([i, stats['Total_Eq'], stats['Largest_Eq_Date']])

result_df = pd.DataFrame(results, columns=['Country', 'Total_Earthquakes', '
result_df = result_df.sort_values('Total_Earthquakes', ascending=False)


result_df.head(20)
```

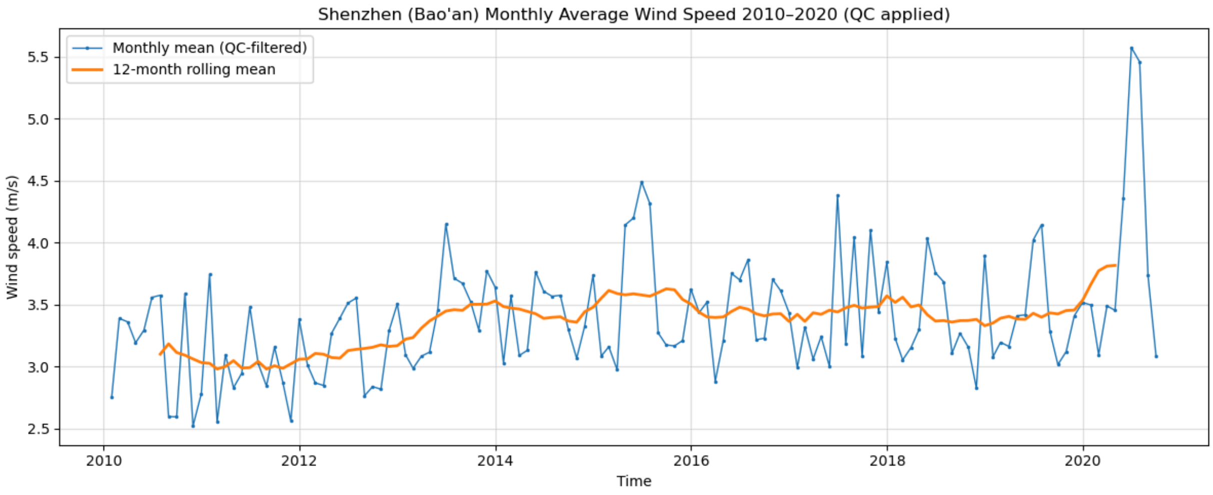
Out[7]:

	Country	Total_Earthquakes	Largest_Eq_Date
15	CHINA	623	1668-07-25
34	JAPAN	424	2011-03-11
73	INDONESIA	421	2004-12-26
8	IRAN	388	0856-12-22
10	TURKEY	358	1939-12-26
6	ITALY	333	1915-01-13
4	GREECE	289	0365-07-21
56	USA	280	1964-03-28
71	PHILIPPINES	230	1897-09-21
52	MEXICO	214	1787-03-28
60	CHILE	200	1960-05-22
51	PERU	194	1716-02-06
16	RUSSIA	158	1952-11-04
91	PAPUA NEW GUINEA	107	1919-05-06
9	INDIA	102	1950-08-15
77	TAIWAN	101	1920-06-05
67	COLOMBIA	82	1826-06-18
104	NEW ZEALAND	72	1826-01-01
64	ECUADOR	69	1906-01-31
23	AFGHANISTAN	68	1909-07-07

In [9]:

```
#2
import pandas as pd
```

```python
import numpy as np
import matplotlib.pyplot as plt

df = pd.read_csv('2281305.csv', dtype=str, low_memory=False)
start_year = 2010
end_year = 2020

# 解析时间
df['datetime'] = pd.to_datetime(df['DATE'], errors='coerce')
df = df.dropna(subset=['datetime'])

# 解析风速与质量码
def parse_wind_and_qc(wnd_field):
    if pd.isna(wnd_field):
        return pd.Series({'wind_m_s': np.nan, 'wind_qc': None})
    parts = str(wnd_field).split(',')
    if len(parts) < 5:
        return pd.Series({'wind_m_s': np.nan, 'wind_qc': None})
    speed_str = parts[3].strip()
    qc_str = parts[4].strip()

    if (speed_str == '') or (not speed_str.isdigit()):
        speed = np.nan
    else:
        val = int(speed_str)
        speed = np.nan if val == 9999 else val / 10.0  # 0.1 m/s → m/s

    return pd.Series({'wind_m_s': speed, 'wind_qc': qc_str})

df[['wind_m_s', 'wind_qc']] = df['WND'].apply(parse_wind_and_qc)

# 时间范围与质量控制过滤
df = df[(df['datetime'].dt.year >= start_year) & (df['datetime'].dt.year <=

# 仅保留通过质量检查的数据
allowed_qc = {'1', '5', '9'}
df = df.dropna(subset=['wind_m_s'])
df = df[df['wind_qc'].isin(allowed_qc)]

# 按月平均
df = df.set_index('datetime')
monthly_wind = df['wind_m_s'].resample('M').mean()

# 绘图
plt.figure(figsize=(12, 5))
plt.plot(monthly_wind.index, monthly_wind.values, marker='.', markersize=3,
         label='Monthly mean (QC-filtered)')

# 12个月滑动的平均
rolling12 = monthly_wind.rolling(window=12, center=True).mean()
plt.plot(monthly_wind.index, rolling12.values, linewidth=2, label='12-month

plt.title(f"Shenzhen (Bao'an) Monthly Average Wind Speed {start_year}–{end_y
plt.xlabel("Time")
plt.ylabel("Wind speed (m/s)")
plt.grid(alpha=0.4)
plt.legend()
plt.tight_layout()
plt.show()
```


Shenzhen (Bao'an) Monthly Average Wind Speed 2010–2020 (QC applied)

In [11]:

```
# 3.1

import pandas as pd
import numpy as np

# 读取
AQC = pd.read_csv('全国主要城市空气质量.csv', encoding='gbk')

city = AQC.columns[0]
date = AQC.columns[1]
pm   = 'PM2.5'

# 日期排序
AQC[date] = pd.to_datetime(AQC[date], errors='coerce')
AQC = AQC[AQC[date].notna()]
AQC = AQC.sort_values([city, date])

# 数据清洗
num_cols = []
for col in AQC.columns:
    if col != city and col != date:
        AQC[col] = pd.to_numeric(AQC[col], errors='coerce')
        num_cols.append(col)
AQC[num_cols] = AQC[num_cols].fillna(0)

# 保存清洗结果
AQC_clean = "air_quality_citys_cleaned.csv"
AQC.to_csv(AQC_clean, index=False)

# 我们仅分析四个一线城市
target_cn = ["北京", "上海", "广州", "深圳"]
city_en   = {"北京": "Beijing", "上海": "Shanghai", "广州": "Guangzhou", "深圳'
AQC_four = AQC[AQC[city].isin(target_cn)].copy()

# 根据城市和日聚合生成日均序列
df_daily = AQC_four[[city, date, pm]].copy()
df_daily = df_daily[df_daily[date].notna()]
df_daily["date_only"] = df_daily[date].dt.date
daily_mean = df_daily.groupby([city, "date_only"], as_index=False)[pm].mean(

# 计算城市每月的日均再平均
daily_mean["year"]  = pd.to_datetime(daily_mean["date_only"]).dt.year
daily_mean["month"] = pd.to_datetime(daily_mean["date_only"]).dt.month
monthly_mean = daily_mean.groupby([city, "year", "month"], as_index=False)[p

# 城市月均定义
monthly_mean = monthly_mean[monthly_mean[city].isin(target_cn)].copy()
monthly_mean["YYYY_MM"] = monthly_mean["year"].astype(str) + "-" + monthly_m


monthly_out = "pm25_first_tier_monthly_means.csv"
monthly_mean[[city, "year", "month", "YYYY_MM", pm]].to_csv(monthly_out, inc
monthly_mean
```

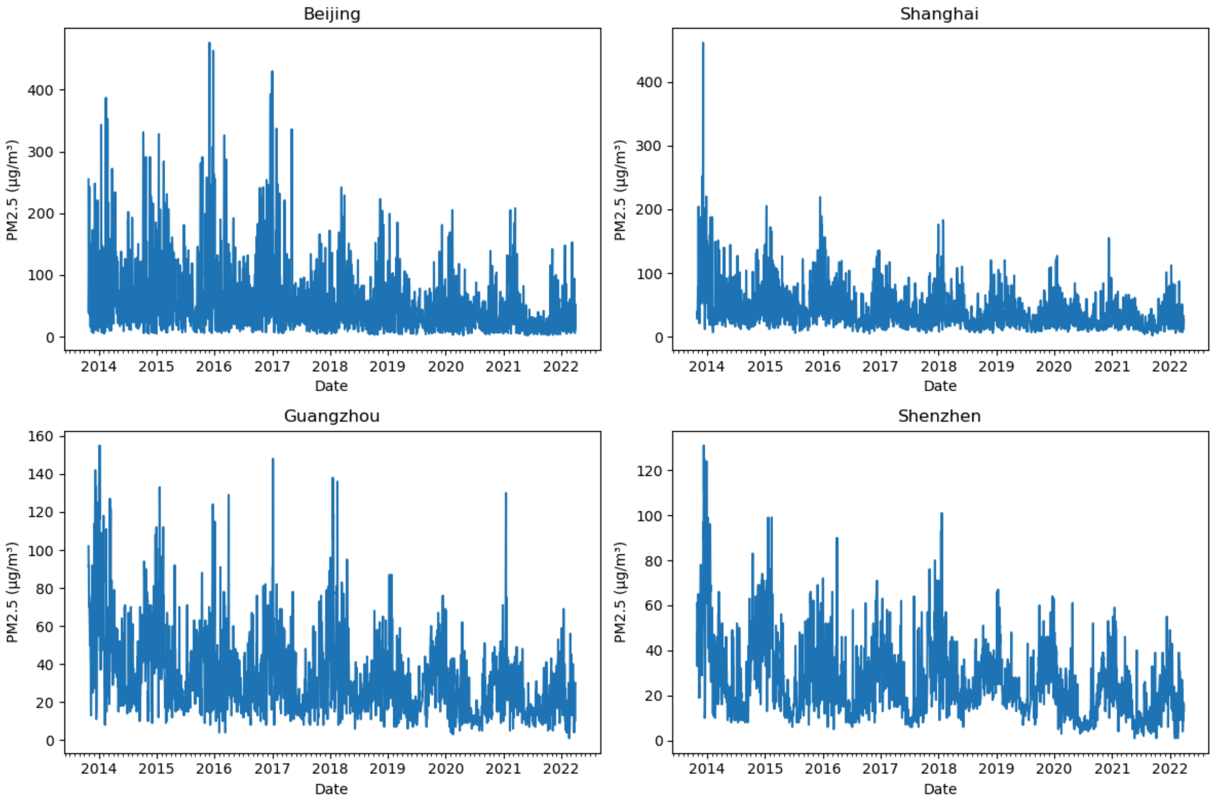
Out[11]:

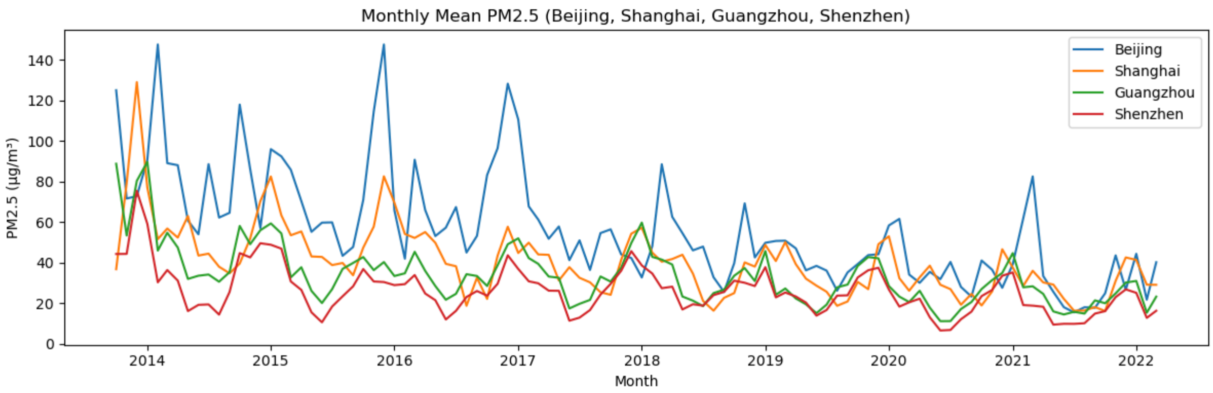
	城市	year	month	PM2.5	YYYY_MM
0	上海	2013	10	36.750000	2013-10
1	上海	2013	11	79.200000	2013-11
2	上海	2013	12	129.000000	2013-12
3	上海	2014	1	77.066667	2014-01
4	上海	2014	2	51.571429	2014-02
...	...	...	...	...	...
403	深圳	2021	11	22.966667	2021-11
404	深圳	2021	12	26.806452	2021-12
405	深圳	2022	1	25.096774	2022-01
406	深圳	2022	2	12.750000	2022-02
407	深圳	2022	3	16.258065	2022-03

408 rows × 5 columns

In [15]:

```
# 3.2
import matplotlib.pyplot as plt
import matplotlib.dates as mdates

# 绘制四个城市的pm2.5时序图（pm2.5/日）
fig, axes = plt.subplots(2, 2, figsize=(12, 8))
axes = axes.ravel()

for i in range(4):
    if i < len(target_cn):
        cn = target_cn[i]
        sub = AQC_four[AQC_four[city] == cn].copy().sort_values(date)
        if len(sub) > 0 and pm in sub.columns:
            axes[i].plot(sub[date], sub[pm])
        axes[i].set_title(city_en.get(cn, cn))
        axes[i].set_xlabel("Date")
        axes[i].set_ylabel("PM2.5 (µg/m³)")
         # 刻度设置
        axes[i].xaxis.set_major_locator(mdates.AutoDateLocator())
        axes[i].xaxis.set_minor_locator(mdates.MonthLocator())
        axes[i].tick_params(axis='x', which='minor', length=2)
    else:
        fig.delaxes(axes[i])

plt.tight_layout()
plt.savefig("pm25_first_tier_cities.png", dpi=150, bbox_inches="tight")
plt.show()


# 图2：四个城市的月均时序对比
monthly_mean["month_start"] = pd.to_datetime(monthly_mean["YYYY_MM"] + "-01"

plt.figure(figsize=(12, 4))
for cn in target_cn:
    subm = monthly_mean[monthly_mean[city] == cn].copy().sort_values("month_
    if len(subm) > 0:
        plt.plot(subm["month_start"], subm[pm], label=city_en.get(cn, cn))

plt.title("Monthly Mean PM2.5 (Beijing, Shanghai, Guangzhou, Shenzhen)")
plt.xlabel("Month")
plt.ylabel("PM2.5 (µg/m³)")
plt.legend()
plt.tight_layout()
plt.savefig("pm25_first_tier_monthly.png", dpi=150, bbox_inches="tight")
plt.show()
```





In [17]:

```
# 3.3

# 国标pm2.5的限值
# 年平均浓度限值（GB 3095-2012 二类区）
gb_35 = 35.0
# 24小时平均浓度限值（GB 3095-2012 二类区）
gb_75 = 75.0

# 月份字符和超75定义
daily_mean["YYYY_MM"] = pd.to_datetime(daily_mean["date_only"]).dt.to_perioc
daily_mean["exceed75"] = (daily_mean[pm] > gb_75).astype(int)

# 每城每月的pm2.5大雨75的天数
exceed_month = daily_mean.groupby([city, "YYYY_MM"], as_index=False)["exceec
exceed_month = exceed_month[exceed_month[city].isin(target_cn)].copy()

# 年平均表（基于日均）
annual_mean = daily_mean[daily_mean[city].isin(target_cn)].copy()
annual_mean["year"] = pd.to_datetime(annual_mean["date_only"]).dt.year
annual_mean = annual_mean.groupby([city, "year"], as_index=False)[pm].mean()
annual_out = "pm25_first_tier_annual_means.csv"
annual_mean.to_csv(annual_out, index=False)

# 综合汇总统计
# 空表，存放每个城市的统计结果
rows = []
# 四个城市循环
for cn in target_cn:
    dsub = daily_mean[daily_mean[city] == cn].copy().sort_values("date_only"
    # 统计总共的有效天数
    total_days = int(len(dsub))
    # 统计日均值 > 35 的天数
    exceed_35  = int((dsub[pm] > gb_35).sum())
    # 统计日均值 > 75 的天数
    exceed_75  = int((dsub[pm] > gb_75).sum())

    if total_days > 0:
        # 计算该城市所有日期的总均值
        pm_mean   = float(dsub[pm].mean())
        # 计算中位数
        pm_median = float(dsub[pm].median())
        # 计算标准差
        pm_std    = float(dsub[pm].std())
        # 计算最小值
        pm_min    = float(dsub[pm].min())
        # 计算最大值
        pm_max    = float(dsub[pm].max())
    else:
        # 如果没有数据，所有统计值记为 0
        pm_mean = pm_median = pm_std = pm_min = pm_max = 0.0

    # 找出城市超75天数最多的月份
    em_sub = exceed_month[exceed_month[city] == cn].copy()
    if len(em_sub) > 0:
        idx = em_sub["exceed75"].idxmax()
        max_month = str(em_sub.loc[idx, "YYYY_MM"])
        max_days  = int(em_sub.loc[idx, "exceed75"])
    else:
        max_month = ""
        max_days  = 0
    # 计算结果保存
    rows.append({
        "City": city_en.get(cn, cn),
        "Total_Days": total_days,
        "Daily_Mean": round(pm_mean, 3),
        "Daily_Median": round(pm_median, 3),
        "Daily_Std": round(pm_std, 3),
        "Daily_Min": round(pm_min, 3),
        "Daily_Max": round(pm_max, 3),
        "Exceed_Days_35": exceed_35,
        "Exceed_Days_75": exceed_75,
        "Max_Exceed75_Month": max_month,
        "Max_Exceed75_Days": max_days
    })

stats = pd.DataFrame(rows)
stats.to_csv('pm25_first_tier_stats.csv', index=False)
stats
```

Out[17]:

	City	Total_Days	Daily_Mean	Daily_Median	Daily_Std	Daily_Min	Daily_Max
0	Beijing	3028	56.442	40.0	55.192	2.0	476.0
1	Shanghai	3026	40.783	33.0	30.534	2.0	461.0
2	Guangzhou	3028	33.306	28.0	20.765	1.0	155.0
3	Shenzhen	3028	26.042	23.0	15.870	1.0	131.0

In [ ]: