**《Python程序设计》**

**小作业2：链家、PM2.5**



**学院：计算机学院（国家示范性软件学院）**

**班级：2021211313**

**姓名：吕子健**

**学号：2023523012**

1. **实验目的**

**作业1：爬取并存储链家的新房数据，并进行预处理 （本题7分）**

（1）爬取起始网页：<https://bj.fang.lianjia.com/loupan/>

（2）爬取信息的提取及存储要求（单条数据示例在第4页） • 信息以csv文件存储，应包括以下字段：名称，类别，地理位置（3个字段分别存储），房 型（只保留最小房型），面积（区间取中值并取整），均价（元，整数），总价（万元， 区间取中值并取整） • 对于所有字符串字段，要求去掉所有的前后空格 • 删除面积缺失的房屋数据

（3）数据统计 • 找出总价最贵和最便宜的房子，以及总价的中位数 • 找出均价最贵和最便宜的房子，以及均价的中位数

（4）异常值处理 • 列出总价在均值三倍标准差以外的房屋，展示其基本信息（如果太多可以只展示 一部分），并分析其原因（找4条数据即可） • 通过箱型图原则判断并列出均价为异常值的房屋，展示其基本信息（如果太多可 以只展示一部分），并分析其原因（找4条数据即可）

（5）离散化处理 • 对房屋的均价进行离散化处理，自行设定每个区间的长度并给出设置的理由，给 出每个区间的房屋数量和所占比例

**作业2：分析处理2015年北京市PM2.5指数数据集空值（本题3分）**

（1）原始数据集：BeijingPM20100101\_20151231.csv（列信息见第6页说明）

（2）数据抽取及存储：从原始数据集中抽取2015年度数据，存储为新的csv文件 （3）找出空值：对新的csv文件，找出存在的空值列及相应的空值数量

（4）空值处理方法：对所有存在空值的列，给出空值的处理方法及理由，要求处 理方法必须可在本数据集范围内执行

（5）空值处理并存储：按照自己的处理方法，通过pandas、numpy或python方法 对空值进行处理，完成后给出新的空值列信息，并将处理后的数据（不涉及空值的 列应原样保留）存储为新的csv文件

1. **实验内容（代码）**

**作业1**

# 导入所需的库

import requests

from bs4 import BeautifulSoup

import pandas as pd

import numpy as np

import time

import matplotlib.pyplot as plt

import os

os.chdir(os.path.dirname(\_\_file\_\_))

plt.rcParams['font.sans-serif']=['SimHei']

plt.rcParams['axes.unicode\_minus'] = False

# 定义一个函数，用于从字符串中提取数字

def extract\_number(s):

    # 如果字符串为空，返回None

    if not s:

        return None

    # 否则，去掉所有的非数字字符，包括小数点和逗号

    s = s.replace('.', '').replace(',', '')

    # 尝试将字符串转换为整数，如果失败，返回None

    try:

        return int(s)

    except ValueError:

        return None

# 定义一个函数，用于从面积区间中取中值并取整

def get\_area\_midpoint(s):

    # 如果字符串为空，返回None

    if not s:

        return None

    # 否则，去掉所有的非数字字符，包括平方米和逗号

    s = s.replace('平方米', '').replace(',', '').replace('㎡', '').replace('建面', '')

    # 尝试将字符串分割为两个数字，如果失败，返回None

    try:

        a, b = map(int, s.split('-'))

    except ValueError:

        return None

    # 返回两个数字的平均值并取整

    return round((a + b) / 2)

# 定义一个函数，用于从总价区间中取中值并取整

def get\_price\_midpoint(s):

    # 如果字符串为空，返回None

    if not s:

        return None

    # 否则，去掉所有的非数字字符，包括万元和逗号

    s = s.replace('(万/套)', '').replace('总价', '')

    # 尝试将字符串分割为两个数字，如果失败，返回None

    if s.count('-') != 0:

        try:

            a, b = map(int, s.split('-'))

        except ValueError:

            return None

        # 返回两个数字的平均值并取整

        return round((a + b) / 2)

    else:

        return int(s)

# 定义一个空列表，用于存储房屋数据

data = []

# 定义起始网页的url

url = 'https://bj.fang.lianjia.com/loupan/'

pg=1

# 定义一个循环，用于遍历所有的网页

while url:

    pg += 1

    # 发送请求，获取网页内容

    response = requests.get(url)

    # 解析网页内容，使用beautiful soup

    soup = BeautifulSoup(response.text, 'html.parser')

    # 找到所有的房屋信息的div标签

    divs = soup.find\_all('div', class\_='resblock-desc-wrapper')

    # 找到当前页面有几个房屋信息

    house\_value = soup.find('span', class\_='value')

    house\_value\_num = int(house\_value.contents[0])

    if house\_value\_num == 0:

        break

    # 遍历每个div标签，提取房屋信息

    for div in divs:

        # 提取名称，去掉前后空格

        name = div.find('a', class\_='name').text.strip()

        # 提取类别，去掉前后空格

        category = div.find('span', class\_='resblock-type').text.strip()

        # 提取地理位置，分别存储区域，板块和地址，去掉前后空格

        location = div.find('div', class\_='resblock-location').text.strip()

        location = location.replace('\n/\n', ' ').replace(', ', ',')

        region, block, address = location.split(" ", 2)

        # 提取房型，只保留最小房型，去掉前后空格

        room\_type = div.find('a', class\_='resblock-room').text.strip()

        room\_type = room\_type.split('/')[0]

        room\_type = room\_type.replace('\n', '')

        # 提取面积，区间取中值并取整

        area = div.find('div', class\_='resblock-area').text.strip()

        area = get\_area\_midpoint(area)

        # 提取均价，去掉前后空格

        unit\_price = div.find('span', class\_='number').text.strip()

        # 提取总价，区间取中值并取整

        total\_price = div.find('div', class\_='second')

        if total\_price:

            total\_price = total\_price.text.strip()

        else:

            break

        total\_price = get\_price\_midpoint(total\_price)

        # 将房屋信息以字典的形式添加到列表中

        data.append({

            '名称': name,

            '类别': category,

            '区域': region,

            '板块': block,

            '地址': address,

            '房型': room\_type,

            '面积': area,

            '均价': unit\_price,

            '总价': total\_price

        })

    time.sleep(2)

    url = 'https://bj.fang.lianjia.com/loupan/pg' + str(pg) + '/'

    #url = None

# 将列表转换为数据框

df = pd.DataFrame(data)

# 删除面积缺失的房屋数据

df = df.dropna(subset=['面积'])

# 将均价和总价的字符串转换为数字

df['均价'] = df['均价'].apply(extract\_number)

# 将数据框保存为csv文件

df.to\_csv('lianjia.csv', index=False)

# 数据统计

# 找出总价最贵和最便宜的房子，以及总价的中位数

max\_total\_price = df['总价'].max()

min\_total\_price = df['总价'].min()

median\_total\_price = df['总价'].median()

print(f'总价最贵的房子是：{df[df["总价"] == max\_total\_price]["名称"].values[0]}，总价为：{max\_total\_price}万元')

print(f'总价最便宜的房子是：{df[df["总价"] == min\_total\_price]["名称"].values[0]}，总价为：{min\_total\_price}万元')

print(f'总价的中位数是：{median\_total\_price}万元')

# 找出均价最贵和最便宜的房子，以及均价的中位数

max\_unit\_price = df['均价'].max()

min\_unit\_price = df['均价'].min()

median\_unit\_price = df['均价'].median()

print(f'均价最贵的房子是：{df[df["均价"] == max\_unit\_price]["名称"].values[0]}，均价为：{max\_unit\_price}元')

print(f'均价最便宜的房子是：{df[df["均价"] == min\_unit\_price]["名称"].values[0]}，均价为：{min\_unit\_price}元')

print(f'均价的中位数是：{median\_unit\_price}元')

# 异常值处理

# 列出总价在均值三倍标准差以外的房屋，展示其基本信息

mean\_total\_price = df['总价'].mean()

std\_total\_price = df['总价'].std()

outlier\_total\_price = df[(df['总价'] > mean\_total\_price + 3 \* std\_total\_price) | (df['总价'] < mean\_total\_price - 3 \* std\_total\_price)]

print(f'总价在均值三倍标准差以外的房屋有{len(outlier\_total\_price)}个，它们的基本信息如下：')

print(outlier\_total\_price)

# 通过箱型图原则判断并列出均价为异常值的房屋

q1\_unit\_price = df['均价'].quantile(0.25)

q3\_unit\_price = df['均价'].quantile(0.75)

iqr\_unit\_price = q3\_unit\_price - q1\_unit\_price

outlier\_unit\_price = df[(df['均价'] > q3\_unit\_price + 1.5 \* iqr\_unit\_price) | (df['均价'] < q1\_unit\_price - 1.5 \* iqr\_unit\_price)]

print(f'均价为异常值的房屋有{len(outlier\_unit\_price)}个，它们的基本信息如下：')

print(outlier\_unit\_price)

# 离散化处理

bins = np.arange(int(df['均价'].min()/5000)\*5000, df['均价'].max() + 10000, 5000)

labels = [f'{a}-{b}' for a, b in zip(bins[:-1], bins[1:])]

print(f'区间为：{labels}')

df['均价区间'] = pd.cut(df['均价'], bins=bins, labels=labels, right=False)

counts = df['均价区间'].value\_counts(sort=False)

percentages = df['均价区间'].value\_counts(normalize=True) \* 100

# 将结果保存为csv文件

result = pd.DataFrame({'均价区间': labels, '房屋数量': counts, '所占比例': percentages})

result.to\_csv('lianjia\_result.csv', index=False)  # 不保存索引列

# 绘制直方图

plt.bar(result['均价区间'], result['房屋数量'])

plt.xticks(rotation=90)  # 旋转x轴刻度标签，以避免重叠

plt.xlabel('均价区间')

plt.ylabel('房屋数量')

plt.title('房屋均价离散图')

plt.show()

# 打印结果

print(result.to\_string(index=False))

**作业2**

import pandas as pd

import os

os.chdir(os.path.dirname(\_\_file\_\_))

df = pd.read\_csv("BeijingPM20100101\_20151231.csv")

# 筛选出2015年的数据

df\_2015 = df[df["year"] == 2015]

# 保存为新的csv文件

df\_2015.to\_csv("BeijingPM2015.csv", index=False)

print(df\_2015.isnull().sum())

# 读取新的csv文件

df\_2015 = pd.read\_csv("BeijingPM2015.csv")

# 对PM\_xx列使用线性插值法填充缺失值

df\_2015["PM\_Dongsi"] = df\_2015["PM\_Dongsi"].interpolate(method="linear")

df\_2015["PM\_Dongsihuan"] = df\_2015["PM\_Dongsihuan"].interpolate(method="linear")

df\_2015["PM\_Nongzhanguan"] = df\_2015["PM\_Nongzhanguan"].interpolate(method="linear")

df\_2015["PM\_US Post"] = df\_2015["PM\_US Post"].interpolate(method="linear")

# 对DEWP和TEMP列使用均值填充缺失值

df\_2015["DEWP"] = df\_2015["DEWP"].fillna(df\_2015["DEWP"].mean())

df\_2015["TEMP"] = df\_2015["TEMP"].fillna(df\_2015["TEMP"].mean())

# 对HUMI、PRES、cbwd和Iws列使用相邻位置的数据填充缺失值

df\_2015["HUMI"] = df\_2015["HUMI"].fillna(df\_2015["PM\_Nongzhanguan"])

df\_2015["PRES"] = df\_2015["PRES"].fillna(df\_2015["PM\_Nongzhanguan"])

df\_2015["cbwd"] = df\_2015["cbwd"].fillna(df\_2015["PM\_Nongzhanguan"])

df\_2015["Iws"] = df\_2015["Iws"].fillna(df\_2015["PM\_Nongzhanguan"])

# 对precipitation和Iprec列使用0填充缺失值

df\_2015["precipitation"] = df\_2015["precipitation"].fillna(0)

df\_2015["Iprec"] = df\_2015["Iprec"].fillna(0)

# 保存为新的csv文件

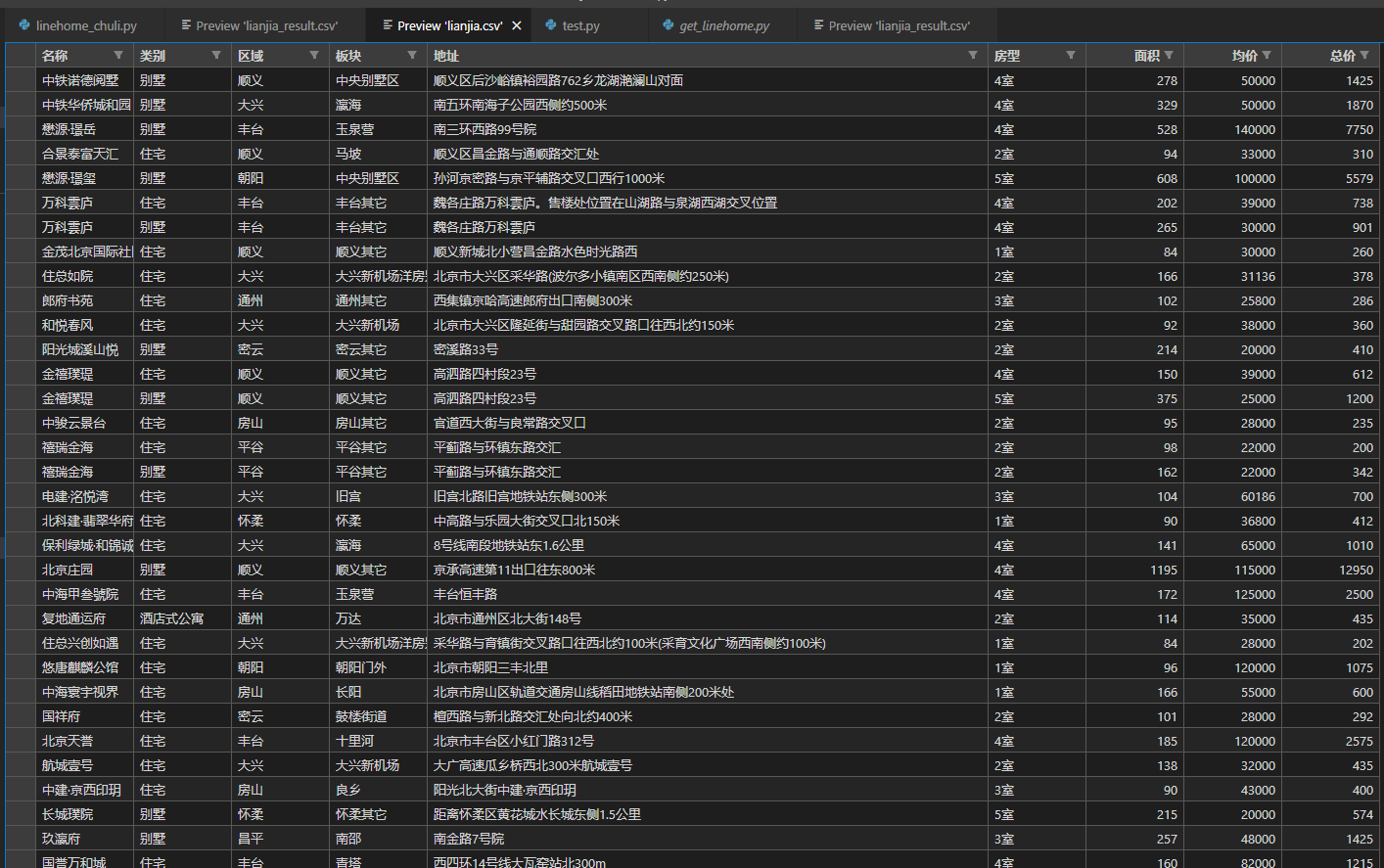
df\_2015.to\_csv("BeijingPM2015\_clean.csv", index=False)

print(df\_2015.isnull().sum())

1. **实验结果及其分析**

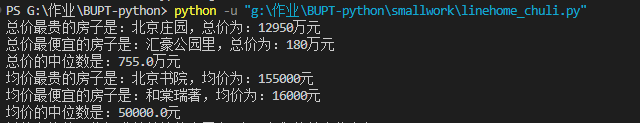
**作业1**

爬出数据的csv文件：

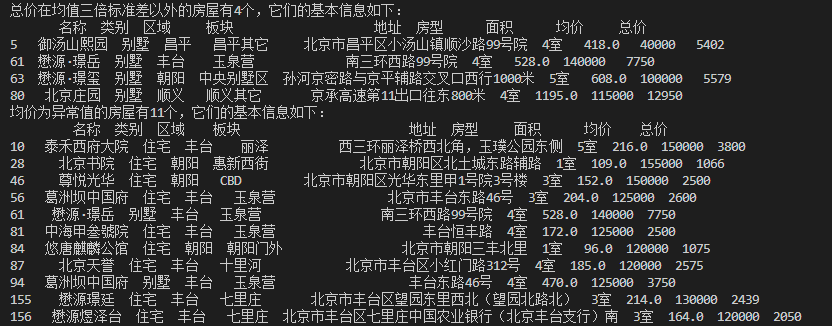
 

（部分）

数据统计：



异常值处理：



总价异常共性分析：

1. 总面积大于平均，属于大面积住宅。
2. 都是别墅，房型比较豪华，整体价值高。
3. 异常值只有高于均值没有低于均值，分析为新房中小面积房子较少。

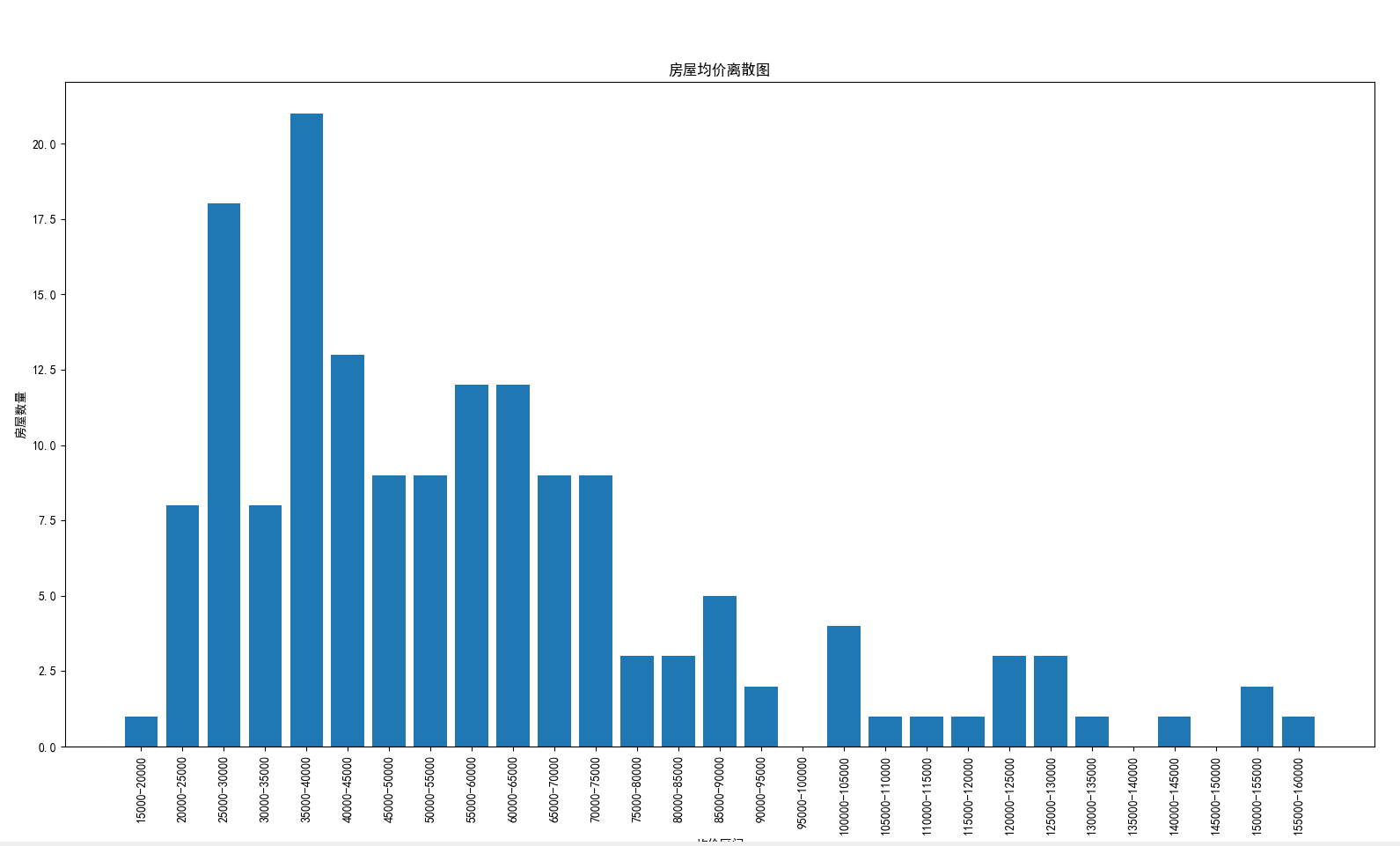
均价异常共性分析：

1. 异常值只有高于均值没有低于均值。
2. 仅位于丰台和朝阳两地，地理位置优越，配套资源齐全，且作为后发展辖区相较于类似老辖区有更多空地开发新房。

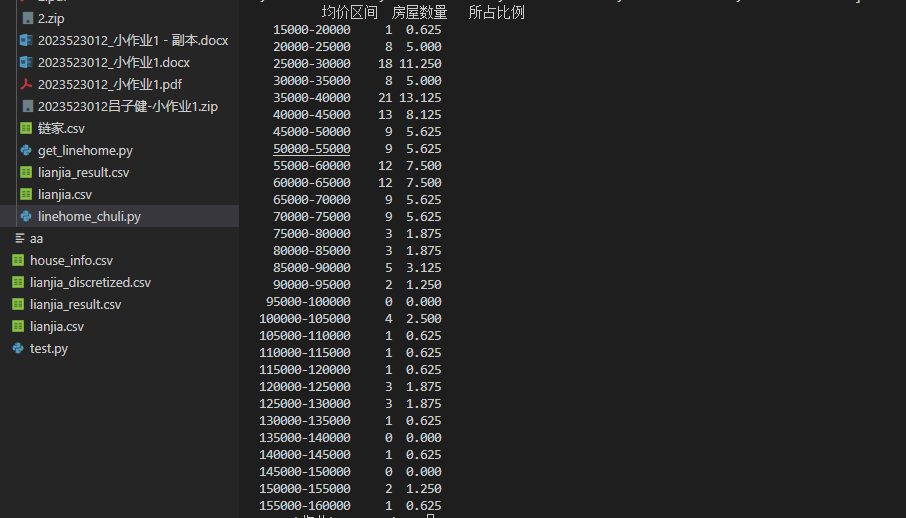
离散化处理：

我选择每个区间的长度为5000元，因为这样可以保证区间的数量不会太多或太少，也可以反映均价的分布情况。

matplotlib.pyplot输出柱状图：

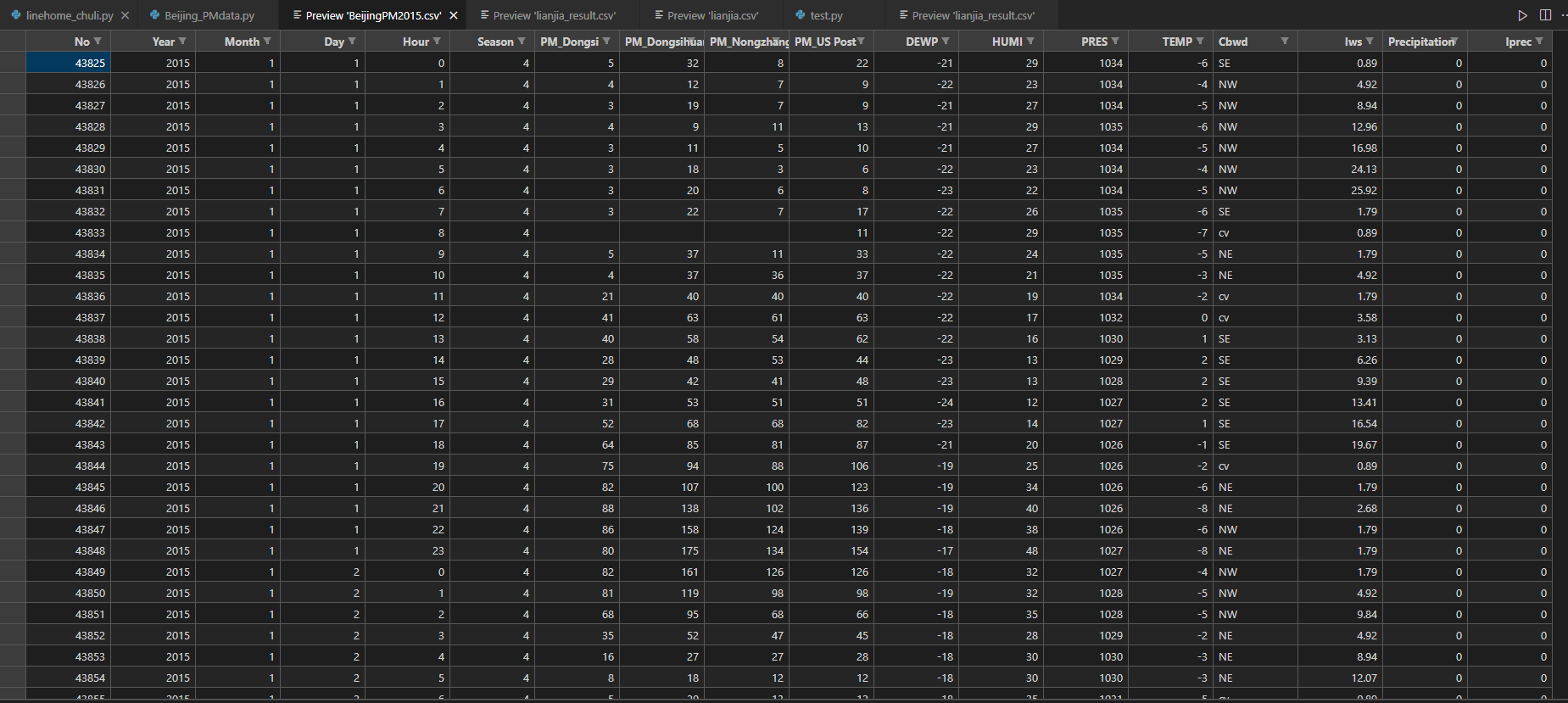


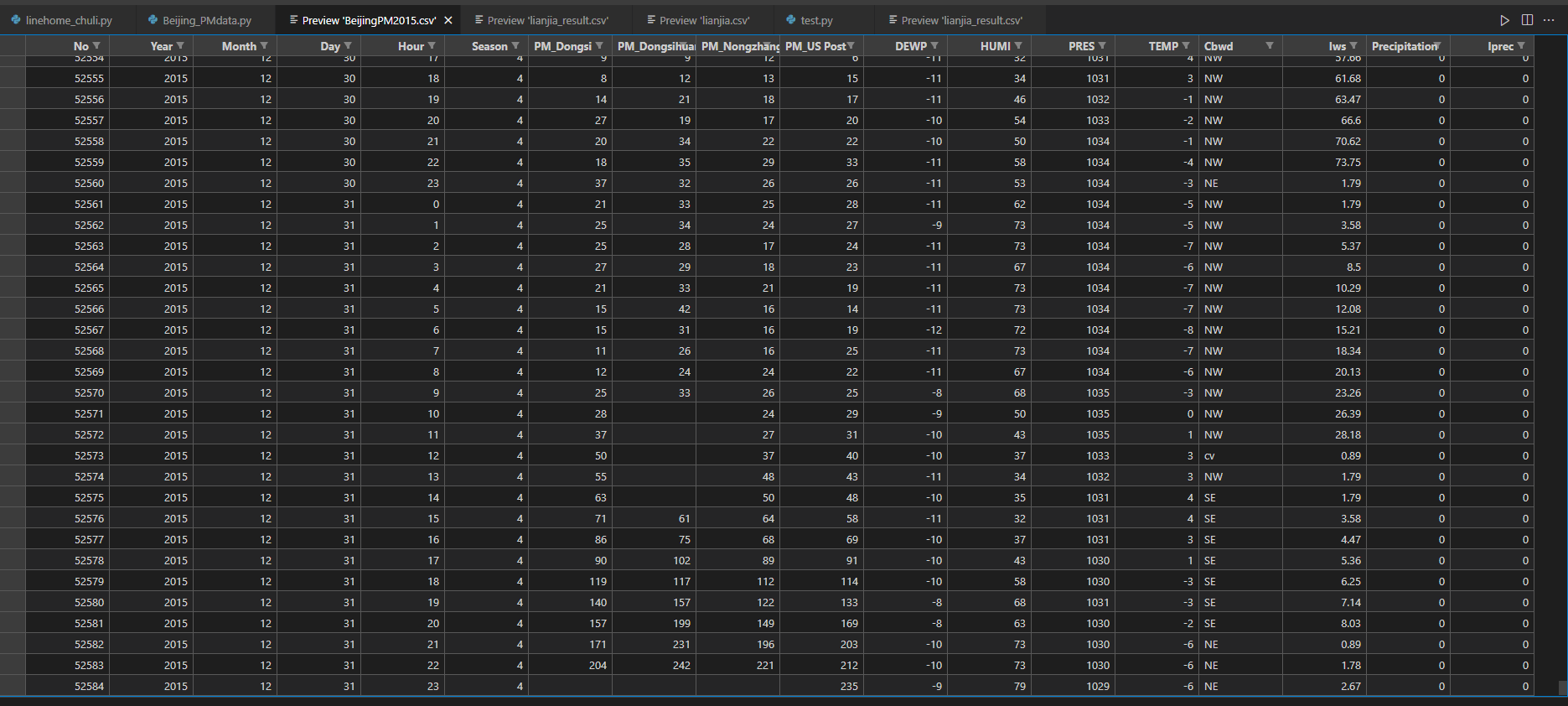
终端中显示比例：



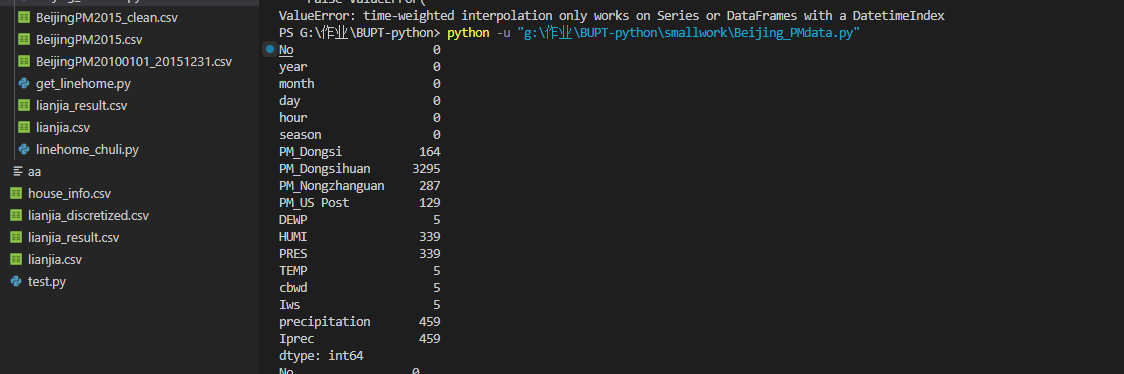
**作业2**

1. 从原始数据集中抽取2015年度数据，存储为新的csv文件：





1. 找出空值：对新的csv文件，找出存在的空值列及相应的空值数量：



1. 空值处理方法：对所有存在空值的列，给出空值的处理方法及理由：

对于PM2.5浓度的四个列（PM\_Dongsi, PM\_Dongsihuan, PM\_Nongzhanguan, PM\_US Post）它们是目标变量，不能随意填充或删除，所以我使用插值法来估计缺失值，例如使用线性插值或三次样条插值。这样可以保持数据的连续性和平滑性，避免引入额外的误差。

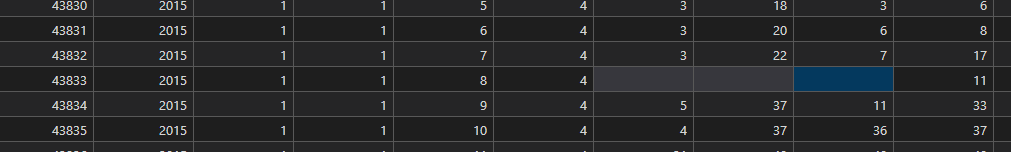
对于DEWP和TEMP列，由于它们只有很少的空值，使用均值或中位数来填充缺失值，这样可以保持数据的一致性和稳定性，避免影响数据的分布和统计特征。

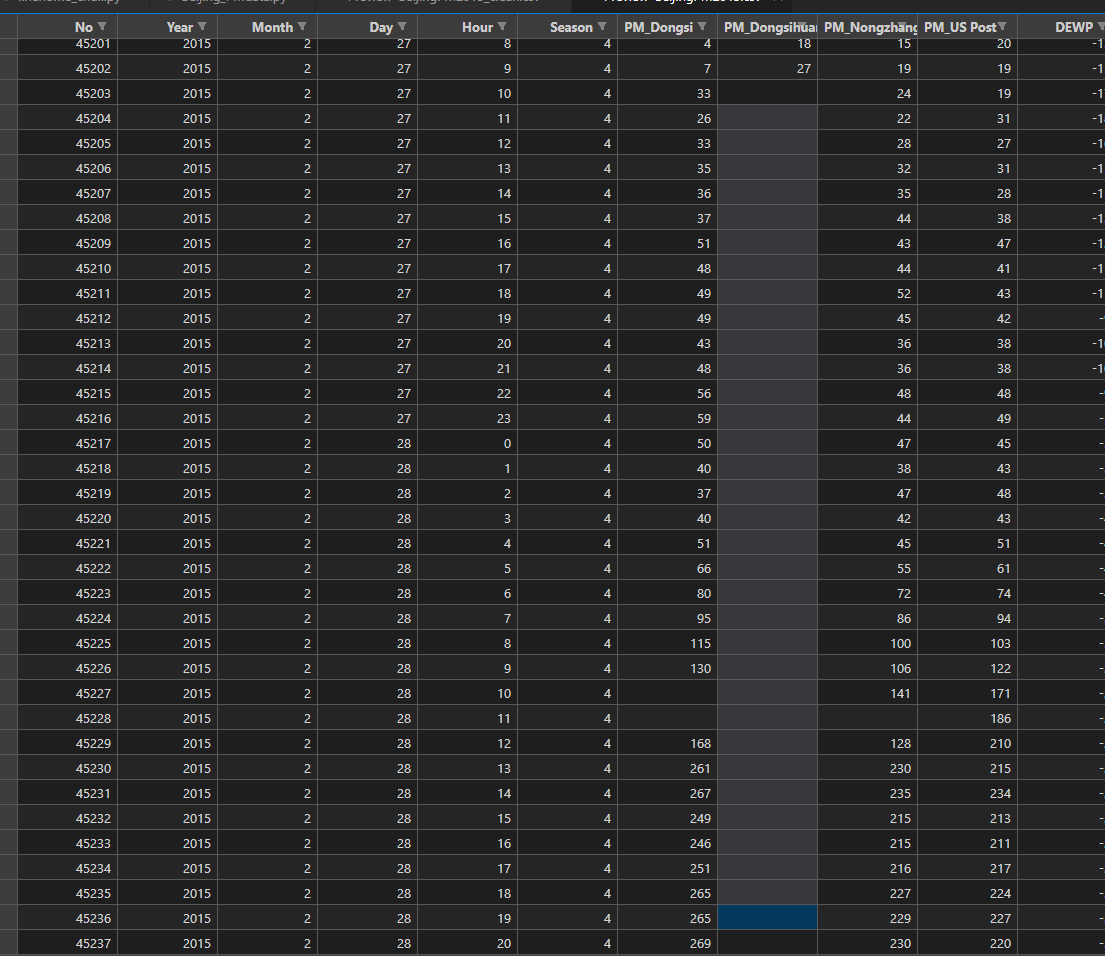
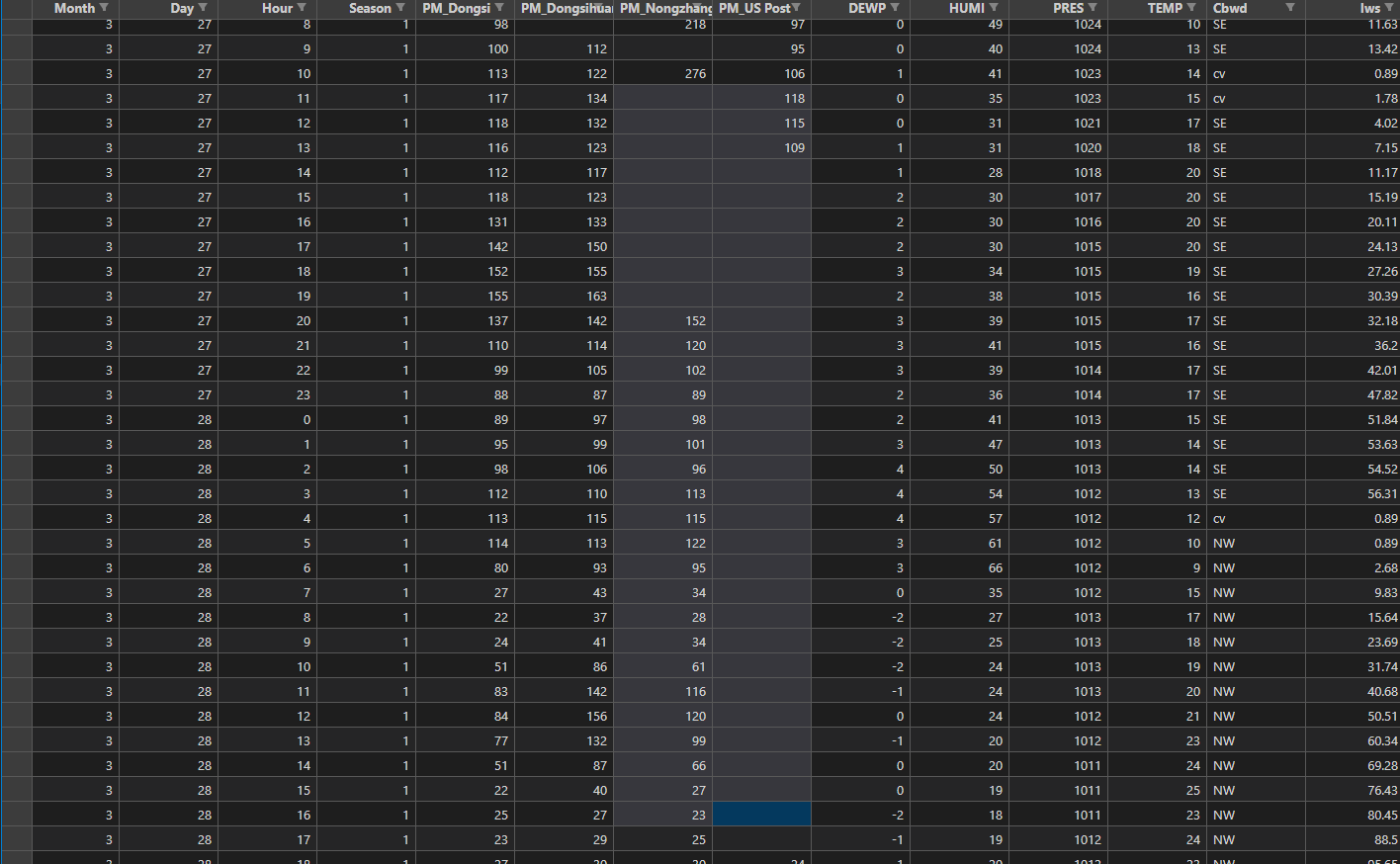
对于HUMI、PRES、cbwd和Iws列，由于它们的空值数量相同，且与气象站的位置有关，使用相邻位置的数据来填充缺失值，例如使用PM\_Nongzhanguan的数据来填充PM\_Dongsihuan的数据。这样可以利用数据的空间相关性，避免引入不合理的数据。

对于precipitation和Iprec列，由于它们的空值数量相同，且与降水量有关，我建议使用0来填充缺失值，这样可以假设缺失值代表没有降水，避免影响数据的总和和比例。

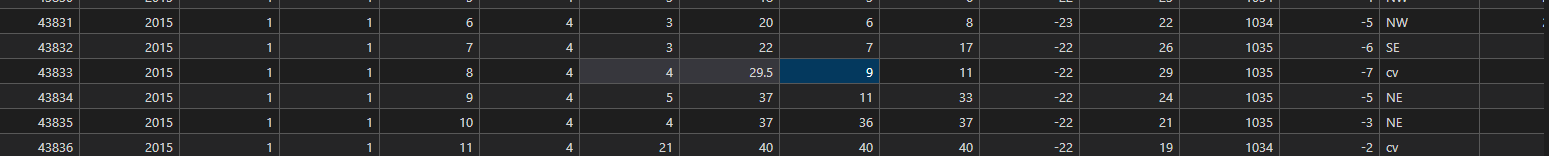
1. 处理结果：

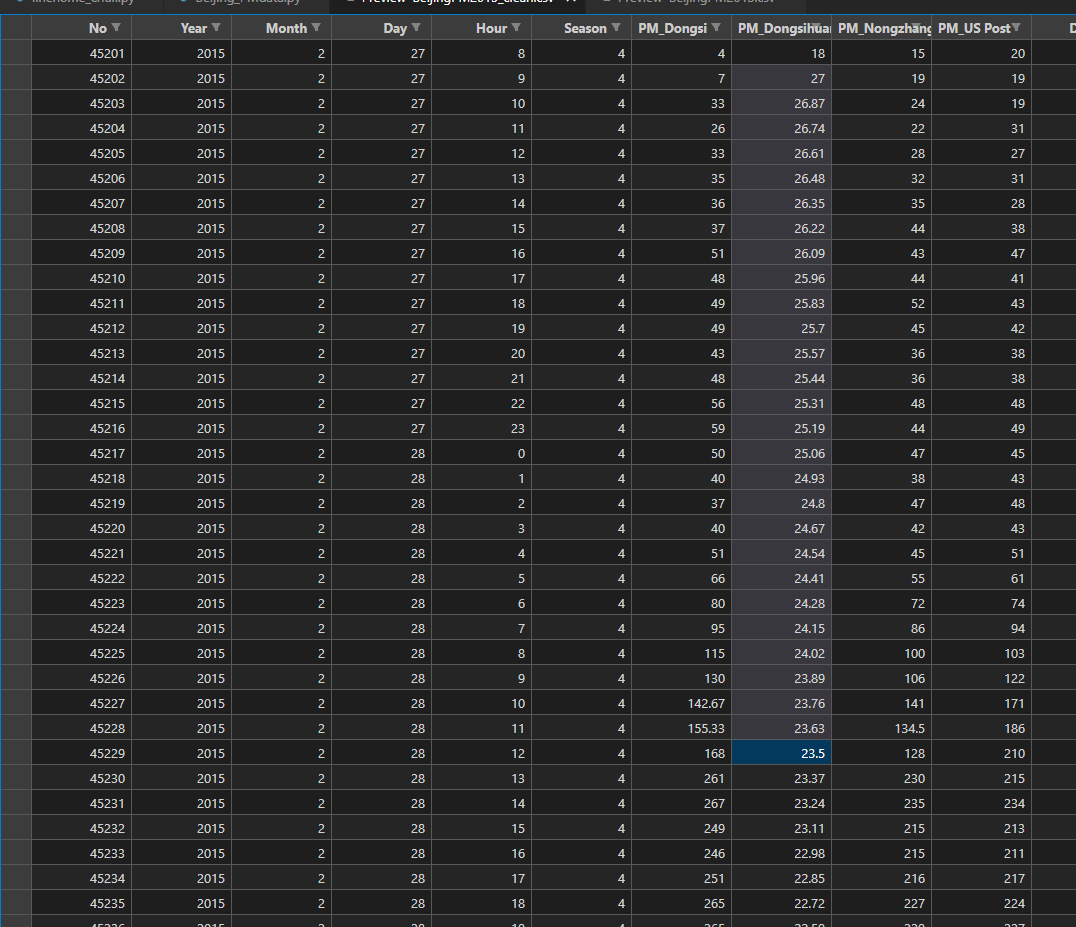
原：

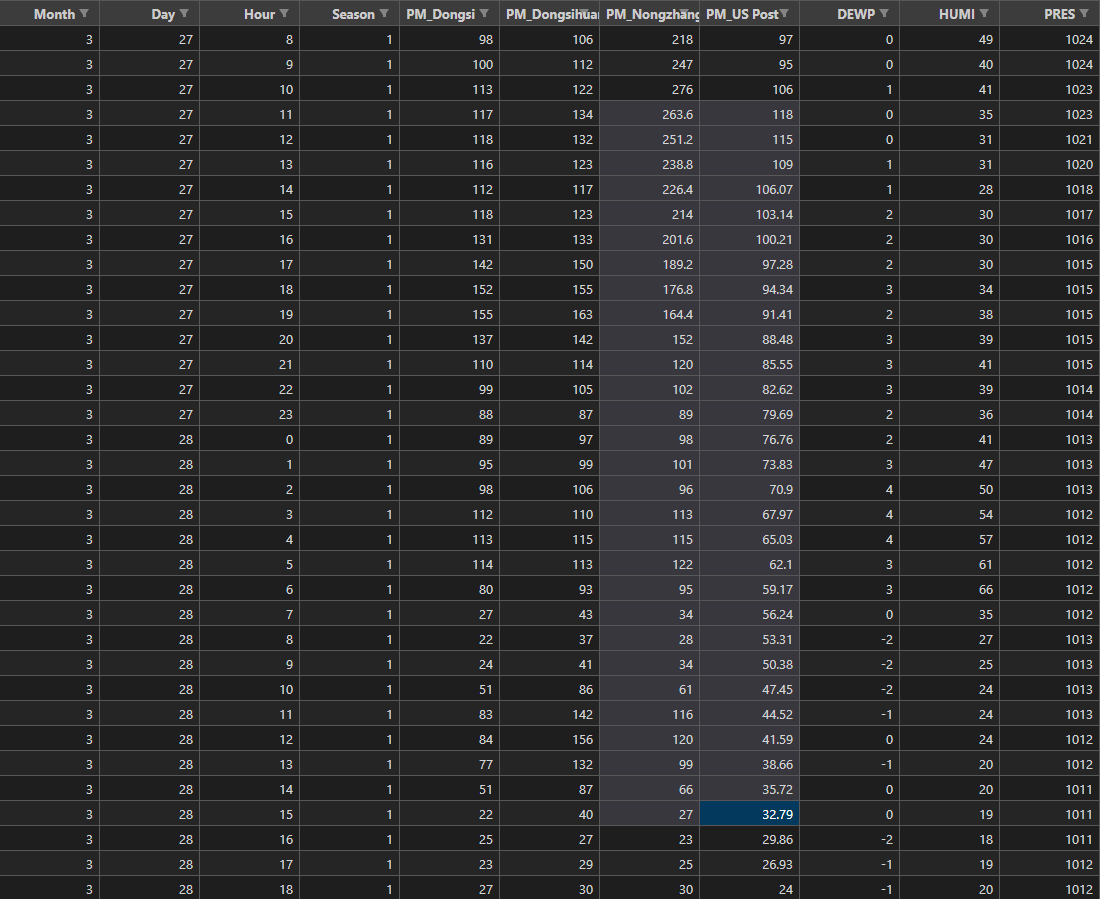


后：







查询结果：

