Mini project 1: air quality in U.S. cities

In a way, this project is simple: you are given some data on air quality in U.S. metropolitan areas over time together with several questions of interest, and your objective is to answer the questions.

However, unlike the homeworks and labs, there is no explicit instruction provided about *how* to answer the questions or where exactly to begin. Thus, you will need to discern for yourself how to manipulate and summarize the data in order to answer the questions of interest, and you will need to write your own codes from scratch to obtain results. It is recommended that you examine the data, consider the questions, and plan a rough approach before you begin doing any computations.

You have some latitude for creativity: **although there are accurate answers to each question** -- namely, those that are consistent with the data -- **there is no singularly correct answer**. Most students will perform similar operations and obtain similar answers, but there's no specific result that must be considered to answer the questions accurately. As a result, your approaches and answers may differ from those of your classmates. If you choose to discuss your work with others, you may even find that disagreements prove to be fertile learning opportunities.

The questions can be answered using computing skills taught in class so far and basic internet searches for domain background; for this project, you may wish to refer to HW1 and Lab1 for code examples and the EPA website on PM pollution for background. However, you are also encouraged to refer to external resources (package documentation, vignettes, stackexchange, internet searches, etc.) as needed -- this may be an especially good idea if you find yourself thinking, 'it would be really handy to do X, but I haven't seen that in class anywhere'.

The broader goal of these mini projects is to cultivate your problem-solving ability in an unstructured setting. Your work will be evaluated based on the following:

- choice of method(s) used to answer questions;
- clarity of presentation;
- code style and documentation.

Please write up your results separately from your codes; codes should be included at the end of the notebook.

Part I: Dataset

Merge the city information with the air quality data and tidy the dataset (see notes below). Write a brief description of the data.

In your description, answer the following questions:

- What is a CBSA (the geographic unit of measurement)?
- How many CBSA's are included in the data?
- In how many states and territories do the CBSA's reside? (*Hint:* `str.split()`)
- In which years were data values recorded?
- How many observations are recorded?
- How many variables are measured?
- Which variables are non-missing most of the time (*i.e.*, in at least 50% of instances)?
- What is PM 2.5 and why is it important?

Please write your description in narrative fashion; ***please do not list answers to the questions above one by one***. A few brief paragraphs should suffice; please limit your data description to three paragraphs or less.

Air quality data

The CBSA is A U.S. geographic area defined by the Office of Management and Budget (OMB) based on census data and referred collectively to both metropolitan statistical areas and micropolitan areas. To calculate how many CBSA's are included in the data, i calculated the length of each unique CBSA's, and the answer is 351. To calculate how many different states are included, i similalr used strsplit function to get every names of states after the commas and then strssplit them accordingly to slashes between states. I then used the set function to count how many unique states there are, and it returned as 52. To calculate how many territories are included i used strsplit function to get every names states and territories. I used an if statement to find append all the territories in my territory list. I then measure the length of the list(unique) and it returned as 36.The data was recored from 2000 to 2019, and it could be seen by just looking at how many columns of years there were.

In order to get how many variables are measured, I firstly melted the data according to years, which means letting different years become rows. Then, I pivoted the dataframe by different variables in pollutant and trend statsitic. There are in total 1134 observations measured across 20 years(because the dataset has 1134 rows). It turns out there are 9 vairbales(some pollutants have multiple trend statistics, and there are 7 pollutants in total, but 9 variables) measured accrossing 20 years. Moreover, I then used isna.sum() to calculate the number of missing values for different pollutants. Note that there are in total 7020 datasets in this pivoted table, thus any sums below the half of 7020 are non-missing most of the time. It turns out that PM2.5 and O3 are the variables of nonmissing most of the time. PM2.5 is an air pollutant that is a concern for people's health when levels in air are high. PM2.5 are tiny particles in the air that reduce visibility and cause the air to appear hazy when levels are elevated. It is important because it could harm people's health

Part II: Descriptive analysis

Focus on the PM2.5 measurements that are non-missing most of the time. Answer each of the following questions in a brief paragraph or two. Your paragraph(s) should indicate both your answer and a description of how you obtained it; ***please do not include codes with your answers***.

Has PM 2.5 air pollution improved in the U.S. on the whole since 2000?

In order to calculate if pm2.5 air pollution has improved, I melted the dataset according to different years and then groupby years. The index that i looked at was weighted annual mean which was only included in pm2.5. I took the average after the groupby function in different years, and we can see a decline across 20 years. So yes, the pollution has improved from city to city in US.

Over time, has PM 2.5 pollution become more variable, less variable, or about equally variable from city to city in the U.S.?

In order to see if the pollution has become more or less variable, I used the same melted dataset from previous question. Similarly to the last question, except that I did not take the average after grouping by the melted dataframe with different years, i took the standard deviation function and we could see a decline across 20 years. So, it has become less variable

Which state has seen the greatest improvement in PM 2.5 pollution over time? Which city has seen the greatest improvement?

The way that i intepreted the word "improvement" was by looking at the difference at PM2.5 pollution between year 2019 and 2000. To look at the difference of cities between 2019 and 2000, I selected the dataframes in melted dataset(where years were the index) with years equal to 2019 and 2000. I then took out the numeric values of different years as two long vectors(or dataframe, to be more precise). I renamed these vectors with corresponding years and merged theses two vectors to a dataframe together. I then added an empty column to the new dataframe to calculate the difference between two years. Lastly, I used idxmax to get the index of the greatest difference between two years. It turns out to be Portsmouth, OH.

To answer the questions that which state has seen the greatest improvement, I used similar methodology and the only difference is that I have to strsplit the column Core Based statistical area after the comma to get different state names. I did it so and grouped by the dataset with different states and looked at their mean(since there are multiple territories within a state). Similarly I inserted a column with empty entries to the dataframe and calculated the difference betweeen two years. I then used idxmax() to get the index of greatest diffrence between two years. The answer turns out to be WV. I used a for loop to go through all the entries without '-' because it means territory in the dataset.

Choose a location with some meaning to you (e.g. hometown, family lives there, took a vacation there, etc.). Was that location in compliance with EPA primary standards as of the most recent measurement?

I took a look at my cleaned dataset where it only included the weighted annual mean for pm2.5. The city that I chose was Cincinnati because i had high school over there. At year 2019, it had 9.2 unit of PM2.5. I looked up online and the epa standards for anual mean was 12. So yes, it has reached the epa primary standard for pm2.5 in the most recent measurement

Extra credit: Imputation

One strategy for filling in missing values ('imputation') is to use non-missing values to predict the missing ones; the success of this strategy depends in part on the strength of relationship between the variable(s) used as predictors of missing values.

Identify one other pollutant that might be a good candidate for imputation based on the PM 2.5 measurements and explain why you selected the variable you did. Can you envision any potential pitfalls to this technique?

The way that i looked this question is by looking at the correlation matrix between two variables PM2.5 and other pollutants. It turns out that O3 has the strongest correlation with PM2.5 weighted annual mean. So O3 is the best variable to be predicted. Some pitfalls about this technique is that the prediction might not be good because there are biases exist in different cities due to missing values. So the best solution is to find a way to fill up the missing values.

Codes

In [1]:
```python
# packages
import numpy as np
import pandas as pd
import sklearn
# raw data
air_raw = pd.read_csv('air-quality.csv')
cbsa_info = pd.read_csv('cbsa-info.csv')
data=pd.merge(air_raw, cbsa_info, how = 'left', on='CBSA')
## PART I
data
```

Out[1]:

	CBSA	Pollutant	Trend Statistic	Number of Trends Sites	2000	2001	2002	2003	2004	2005	...	2011	2012	2013	2014	2015	2016	2017	2018	2019	Core Based Statistical Area
0	10100	PM10	2nd Max	1	50.000	58.000	59.000	66.000	39.000	48.000	...	29.000	62.000	66.000	36.000	43.000	65.000	40.000	49.000	35.000	Aberdeen, SD
1	10100	PM2.5	Weighted Annual Mean	1	8.600	8.600	7.900	8.400	8.100	9.000	...	7.100	7.500	7.300	6.200	6.200	5.400	5.800	6.600	5.900	Aberdeen, SD
2	10100	PM2.5	98th Percentile	1	23.000	23.000	20.000	21.000	23.000	23.000	...	18.000	23.000	22.000	17.000	14.000	14.000	13.000	22.000	18.000	Aberdeen, SD
3	10300	O3	4th Max	1	0.082	0.086	0.089	0.088	0.074	0.082	...	0.076	0.087	0.064	0.068	0.065	0.069	0.066	0.071	0.059	Adrian, MI
4	10420	CO	2nd Max	1	2.400	2.700	1.800	1.900	2.100	1.600	...	1.000	1.100	0.800	0.800	1.000	1.100	0.900	1.800	1.800	Akron, OH
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1129	49700	NO2	Annual Mean	1	13.000	14.000	15.000	14.000	12.000	12.000	...	8.000	10.000	10.000	8.000	7.000	7.000	7.000	7.000	6.000	Yuba City, CA
1130	49700	NO2	98th Percentile	1	62.000	62.000	62.000	62.000	52.000	51.000	...	44.000	46.000	52.000	44.000	39.000	40.000	42.000	41.000	40.000	Yuba City, CA
1131	49700	O3	4th Max	2	0.081	0.077	0.090	0.085	0.076	0.075	...	0.070	0.073	0.066	0.072	0.068	0.072	0.074	0.073	0.063	Yuba City, CA
1132	49700	PM2.5	Weighted Annual Mean	1	10.600	11.900	13.100	9.500	10.000	9.500	...	8.000	6.900	8.200	9.400	9.600	8.100	9.300	10.300	8.400	Yuba City, CA
1133	49700	PM2.5	98th Percentile	1	38.000	54.000	34.000	29.000	38.000	42.000	...	37.000	24.000	25.000	25.000	31.000	22.000	32.000	37.000	27.000	Yuba City, CA

1134 rows × 25 columns

In [2]:
```python
air_raw
```

Out[2]:

	CBSA	Pollutant	Trend Statistic	Number of Trends Sites	2000	2001	2002	2003	2004	2005	...	2010	2011	2012	2013	2014	2015	2016	2017	2018	2019
0	10100	PM10	2nd Max	1	50.000	58.000	59.000	66.000	39.000	48.000	...	46.000	29.000	62.000	66.000	36.000	43.000	65.000	40.000	49.000	35.000
1	10100	PM2.5	Weighted Annual Mean	1	8.600	8.600	7.900	8.400	8.100	9.000	...	8.700	7.100	7.500	7.300	6.200	6.200	5.400	5.800	6.600	5.900
2	10100	PM2.5	98th Percentile	1	23.000	23.000	20.000	21.000	23.000	23.000	...	27.000	18.000	23.000	22.000	17.000	14.000	14.000	13.000	22.000	18.000
3	10300	O3	4th Max	1	0.082	0.086	0.089	0.088	0.074	0.082	...	0.066	0.076	0.087	0.064	0.068	0.065	0.069	0.066	0.071	0.059
4	10420	CO	2nd Max	1	2.400	2.700	1.800	1.900	2.100	1.600	...	1.400	1.000	1.100	0.800	0.800	1.000	1.100	0.900	1.800	1.800
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1129	49700	NO2	Annual Mean	1	13.000	14.000	15.000	14.000	12.000	12.000	...	8.000	8.000	10.000	10.000	8.000	7.000	7.000	7.000	7.000	6.000
1130	49700	NO2	98th Percentile	1	62.000	62.000	62.000	62.000	52.000	51.000	...	45.000	44.000	46.000	52.000	44.000	39.000	40.000	42.000	41.000	40.000
1131	49700	O3	4th Max	2	0.081	0.077	0.090	0.085	0.076	0.075	...	0.067	0.070	0.073	0.066	0.072	0.068	0.072	0.074	0.073	0.063
1132	49700	PM2.5	Weighted Annual Mean	1	10.600	11.900	13.100	9.500	10.000	9.500	...	5.900	8.000	6.900	8.200	9.400	9.600	8.100	9.300	10.300	8.400
1133	49700	PM2.5	98th Percentile	1	38.000	54.000	34.000	29.000	38.000	42.000	...	17.000	37.000	24.000	25.000	25.000	31.000	22.000	32.000	37.000	27.000

1134 rows × 24 columns

In [3]:
```python
cbsa_number=len(pd.unique(data['CBSA']))
cbsa_number #this counts how many different cbsa's are included
```

Out[3]:
```
351
```

In [4]:
```python
territory_list=data.loc[:,'Core Based Statistical Area']
territory_name=[]
for i in range(0,len(territory_list)):
    territory_name.append(territory_list[i].split(',')[0])
len(pd.unique(territory_name))
#this counts how many different territories(cities) the dataset includes
```

Out[4]:
```
341
```

In [5]:
```python
state_name=[]
for i in range(0,len(territory_list)):
    state_name.append(territory_list[i].split(', ')[1])
state_name_unique=[]
for i in range(0,len(state_name)):
    state_name_unique.append(state_name[i].split('-'))
state_name_unique_flat=[]
for slist in state_name_unique:
    for item in slist:
        state_name_unique_flat.append(item)
print(set(state_name_unique_flat))
len(set(state_name_unique_flat))
#this counts how many different states are included in the dataset
```

```
{'KS', 'CA', 'RI', 'ID', 'AZ', 'MN', 'NM', 'MD', 'NJ', 'OR', 'NC', 'PA', 'WV', 'LA', 'HI', 'PR', 'MT', 'SC', 'MI', 'AL', 'VA', 'VT', 'NY', 'UT', 'ME', 'NV', 'OH', 'ND', 'I
A', 'TN', 'MA', 'FL', 'NE', 'CT', 'MS', 'MO', 'OK', 'CO', 'AR', 'DE', 'GA', 'DC', 'IL', 'TX', 'NH', 'IN', 'KY', 'WA', 'WY', 'AK', 'SD', 'WI'}
```

Out[5]:
```
52
```

In [6]:
```python
state_name
territory_real=[]
for i in range(0,len(state_name)):
    if(state_name[i].find('-')!=-1):
        territory_real.append(state_name[i])
len(pd.unique(territory_real))
```

Out[6]:
```
36
```

In [7]:
```python
years=['2000','2001','2002','2003','2004','2005','2006','2007','2008','2009','2010','2011','2012','2013','2014','2015','2016','2017'
,'2018','2019']
melteddata_pollutant=pd.melt(data, id_vars=['Pollutant','Trend Statistic','CBSA','Core Based Statistical Area'],
value_vars=years,
var_name='years'
)
melteddata_pollutant_pivot=melteddata_pollutant.pivot_table(melteddata_pollutant, index=['years', 'CBSA','Core Based Statistical Area'],columns=['Pollutant','Trend Statisti
melteddata_pollutant_pivot
```

Out[7]:

			value								
		Pollutant	CO		NO2	O3	PM10	PM2.5		Pb	SO2
		Trend Statistic	2nd Max	98th Percentile	Annual Mean	4th Max	2nd Max	98th Percentile	Weighted Annual Mean	Max 3-Month Average	99th Percentile
years	CBSA	Core Based Statistical Area									
2000	10100	Aberdeen, SD	NaN	NaN	NaN	NaN	50.0	23.0	8.6	NaN	NaN
	10300	Adrian, MI	NaN	NaN	NaN	0.082	NaN	NaN	NaN	NaN	NaN
	10420	Akron, OH	2.4	NaN	NaN	0.085	NaN	37.0	16.2	NaN	163.0
	10500	Albany, GA	NaN	NaN	NaN	NaN	NaN	38.0	16.6	NaN	NaN
	10580	Albany-Schenectady-Troy, NY	1.1	NaN	NaN	0.070	NaN	30.0	12.4	NaN	56.0
...	...	...	...	...	...	...	...	...	...	...	...
2019	49340	Worcester, MA-CT	NaN	NaN	NaN	0.060	NaN	NaN	NaN	NaN	NaN
	49420	Yakima, WA	NaN	NaN	NaN	NaN	NaN	32.0	9.2	NaN	NaN
	49620	York-Hanover, PA	0.7	42.0	7.0	0.062	NaN	20.0	8.8	NaN	8.0
	49660	Youngstown-Warren-Boardman, OH-PA	NaN	NaN	NaN	0.065	31.3	19.0	7.7	NaN	5.0
	49700	Yuba City, CA	NaN	40.0	6.0	0.063	NaN	27.0	8.4	NaN	NaN

7020 rows × 9 columns

In [8]:
```python
melteddata_pollutant_pivot.isna().sum()
#note that there are in total 7020 datasets in this pivoted table, thus we divide the nasum by 7020 to get the percentage of nonmissing,
# thus PM2.5 and O3 are the variables of nonmissing most of the time
```

Out[8]:
```
       Pollutant  Trend Statistic
value  CO         2nd Max                   5840
       NO2        98th Percentile           5680
                  Annual Mean               5240
       O3         4th Max                   1340
       PM10       2nd Max                   4960
       PM2.5      98th Percentile           2740
                  Weighted Annual Mean      2740
       Pb         Max 3-Month Average       6720
       SO2        99th Percentile           5240
dtype: int64
```

In [9]:
```python
years=['2000','2001','2002','2003','2004','2005','2006','2007','2008','2009','2010','2011','2012','2013','2014','2015','2016','2017'
,'2018','2019']
melteddata=pd.melt(data, id_vars=['Trend Statistic','Core Based Statistical Area','CBSA','Pollutant'],
value_vars=years,
var_name='years'
)
melteddata[melteddata['Trend Statistic']=='Weighted Annual Mean'].groupby('years').mean().value
```

Out[9]:
```
years
2000    13.057944
2001    12.688318
2002    12.352336
2003    11.853271
2004    11.642056
2005    12.479439
2006    11.360748
2007    11.573364
2008    10.625234
2009     9.671028
2010     9.830374
2011     9.638318
2012     8.973364
2013     8.798598
2014     8.660748
2015     8.342523
2016     7.585047
2017     7.942991
2018     8.115421
2019     7.559813
Name: value, dtype: float64
```

In [10]:
```python
melteddata[melteddata['Trend Statistic']=='Weighted Annual Mean'].groupby('years').std()
# we can see it has become less variable
```

Out[10]:

	CBSA	value
years		
2000	11339.276927	3.484359
2001	11339.276927	3.243586
2002	11339.276927	3.227867
2003	11339.276927	2.947421
2004	11339.276927	3.931336
2005	11339.276927	3.548080
2006	11339.276927	2.745368
2007	11339.276927	3.039411
2008	11339.276927	2.484377
2009	11339.276927	2.224681
2010	11339.276927	2.400482
2011	11339.276927	2.188624
2012	11339.276927	1.792889
2013	11339.276927	2.153248
2014	11339.276927	1.998415
2015	11339.276927	1.856069
2016	11339.276927	1.687641
2017	11339.276927	1.853767
2018	11339.276927	2.296680
2019	11339.276927	1.610754

In [11]:
```python
melteddata[melteddata['Trend Statistic']=='Weighted Annual Mean'].set_index('years')
```

Out[11]:

	Trend Statistic	Core Based Statistical Area	CBSA	Pollutant	value
years					
2000	Weighted Annual Mean	Aberdeen, SD	10100	PM2.5	8.6
2000	Weighted Annual Mean	Akron, OH	10420	PM2.5	16.2
2000	Weighted Annual Mean	Albany, GA	10500	PM2.5	16.6
2000	Weighted Annual Mean	Albany-Schenectady-Troy, NY	10580	PM2.5	12.4
2000	Weighted Annual Mean	Albuquerque, NM	10740	PM2.5	6.6
...	...	...	...	...	...
2019	Weighted Annual Mean	Winston-Salem, NC	49180	PM2.5	8.5
2019	Weighted Annual Mean	Yakima, WA	49420	PM2.5	9.2
2019	Weighted Annual Mean	York-Hanover, PA	49620	PM2.5	8.8
2019	Weighted Annual Mean	Youngstown-Warren-Boardman, OH-PA	49660	PM2.5	7.7
2019	Weighted Annual Mean	Yuba City, CA	49700	PM2.5	8.4

4280 rows × 5 columns

In [12]:
```python
data_clean=melteddata[melteddata['Trend Statistic']=='Weighted Annual Mean']
data_2000=data_clean[data_clean['years']=='2000']
data_2019=data_clean[data_clean['years']=='2019']
data_2019_vector=data_2019.loc[:,['value','Core Based Statistical Area']]
data_2000_vector=data_2000.loc[:,['value','Core Based Statistical Area']]
data_2000_vector.rename(columns={'value':'2000 value'},inplace=True)
data_2019_vector.rename(columns={'value':'2019 value'},inplace=True)
data_begin_end_merge=pd.merge(data_2019_vector,data_2000_vector,how='left',on='Core Based Statistical Area').set_index('Core Based Statistical Area')
data_begin_end_merge.insert(2,"improvement over 20 years",[0 for element in range(len(data_begin_end_merge))])
# create a new column that is used to measure the difference between 20 years
for i in range(0,len(data_begin_end_merge)):
    data_begin_end_merge.iloc[i,2]=data_begin_end_merge.iloc[i,1]-data_begin_end_merge.iloc[i,0]
# used a for loop to calculate the difference between year 2019 and 2000 to measure the improvments
data_begin_end_merge.idxmax()

# we can see that Portsmouth,OH has the greatest difference between year 2019 and 2000, which means that it has the greatest improvement
```

Out[12]:
```
2019 value                             Medford, OR
2000 value                   Visalia-Porterville, CA
improvement over 20 years            Portsmouth, OH
dtype: object
```

In [13]:
```python
#now we are trying to find the greatest improvement among the states
data_for_state=pd.merge(data_2019_vector,data_2000_vector,how='left',on='Core Based Statistical Area')
state_different=data_for_state.loc[:,'Core Based Statistical Area']
for i in range(0,len(state_different)):
    state_different.iloc[i]=state_different.iloc[i].split(',')[1]
for i in range(0,len(data_for_state)):
    data_for_state.iloc[i,1]=state_different[i]
data_for_state_meanreal=[]
for i in range(0,len(data_for_state)):
    if data_for_state.iloc[i,1].find('-')==-1:
        data_for_state_meanreal.append(data_for_state.iloc[i,:])
data_for_state_meanreal=pd.DataFrame(data_for_state_meanreal)
data_for_state_mean=data_for_state_meanreal.groupby('Core Based Statistical Area').mean()
data_for_state_mean.insert(2,"improvement over 20 years",[0 for element in range(len(data_for_state_mean))])

for i in range(0,len(data_for_state_mean)):
   data_for_state_mean.iloc[i,2]=data_for_state_mean.iloc[i,1]-data_for_state_mean.iloc[i,0]
data_for_state_mean.idxmax()
# now we can see the state TN-VA has the greatest improvement among 20 years
```

```
/var/folders/xv/l8hd10lx5gb4q4s4dpg_6xj40000gn/T/ipykernel_41151/1306497273.py:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  state_different.iloc[i]=state_different.iloc[i].split(',')[1]
```

Out[13]:
```
2019 value                   OR
2000 value                   AL
improvement over 20 years    WV
dtype: object
```

In [14]:
```python
data_clean[data_clean['Core Based Statistical Area']=='Cincinnati, OH-KY-IN']
```

Out[14]:

	Trend Statistic	Core Based Statistical Area	CBSA	Pollutant	years	value
192	Weighted Annual Mean	Cincinnati, OH-KY-IN	17140	PM2.5	2000	17.4
1326	Weighted Annual Mean	Cincinnati, OH-KY-IN	17140	PM2.5	2001	17.0
2460	Weighted Annual Mean	Cincinnati, OH-KY-IN	17140	PM2.5	2002	16.3
3594	Weighted Annual Mean	Cincinnati, OH-KY-IN	17140	PM2.5	2003	16.2
4728	Weighted Annual Mean	Cincinnati, OH-KY-IN	17140	PM2.5	2004	15.3
5862	Weighted Annual Mean	Cincinnati, OH-KY-IN	17140	PM2.5	2005	18.6
6996	Weighted Annual Mean	Cincinnati, OH-KY-IN	17140	PM2.5	2006	14.5
8130	Weighted Annual Mean	Cincinnati, OH-KY-IN	17140	PM2.5	2007	15.6
9264	Weighted Annual Mean	Cincinnati, OH-KY-IN	17140	PM2.5	2008	14.0
10398	Weighted Annual Mean	Cincinnati, OH-KY-IN	17140	PM2.5	2009	13.2
11532	Weighted Annual Mean	Cincinnati, OH-KY-IN	17140	PM2.5	2010	14.0
12666	Weighted Annual Mean	Cincinnati, OH-KY-IN	17140	PM2.5	2011	12.8
13800	Weighted Annual Mean	Cincinnati, OH-KY-IN	17140	PM2.5	2012	11.3
14934	Weighted Annual Mean	Cincinnati, OH-KY-IN	17140	PM2.5	2013	11.1
16068	Weighted Annual Mean	Cincinnati, OH-KY-IN	17140	PM2.5	2014	10.9
17202	Weighted Annual Mean	Cincinnati, OH-KY-IN	17140	PM2.5	2015	9.9
18336	Weighted Annual Mean	Cincinnati, OH-KY-IN	17140	PM2.5	2016	9.4
19470	Weighted Annual Mean	Cincinnati, OH-KY-IN	17140	PM2.5	2017	8.9
20604	Weighted Annual Mean	Cincinnati, OH-KY-IN	17140	PM2.5	2018	9.3
21738	Weighted Annual Mean	Cincinnati, OH-KY-IN	17140	PM2.5	2019	9.2

In [15]:
```python
# Extra credit
melteddata_pollutant_pivot.corr()
```

Out[15]:

			value								
		Pollutant	CO		NO2	O3	PM10	PM2.5		Pb	SO2
		Trend Statistic	2nd Max	98th Percentile	Annual Mean	4th Max	2nd Max	98th Percentile	Weighted Annual Mean	Max 3-Month Average	99th Percentile
	Pollutant	Trend Statistic									
value	CO	2nd Max	1.000000	0.566425	0.609102	0.329461	0.344990	0.282994	0.217951	0.029316	0.314266
	NO2	98th Percentile	0.566425	1.000000	0.871111	0.572854	0.391540	0.461943	0.562625	-0.143588	0.274508
		Annual Mean	0.609102	0.871111	1.000000	0.520512	0.415780	0.423815	0.431196	-0.172479	0.278752
	O3	4th Max	0.329461	0.572854	0.520512	1.000000	0.099132	0.558023	0.680921	0.040126	0.348519
	PM10	2nd Max	0.344990	0.391540	0.415780	0.099132	1.000000	0.079598	-0.045930	-0.251197	0.020013
	PM2.5	98th Percentile	0.282994	0.461943	0.423815	0.558023	0.079598	1.000000	0.738820	-0.066222	0.473462
		Weighted Annual Mean	0.217951	0.562625	0.431196	0.680921	-0.045930	0.738820	1.000000	0.027223	0.637939
	Pb	Max 3-Month Average	0.029316	-0.143588	-0.172479	0.040126	-0.251197	-0.066222	0.027223	1.000000	0.209959
	SO2	99th Percentile	0.314266	0.274508	0.278752	0.348519	0.020013	0.473462	0.637939	0.209959	1.000000

Notes on merging (keep at bottom of notebook)

To combine datasets based on shared information, you can use the `pd.merge(A, B, how = ..., on = SHARED_COLS)` function, which will match the rows of `A` and `B` based on the shared columns `SHARED_COLS`. If `how = 'left'`, then only rows in `A` will be retained in the output (so `B` will be merged *to* `A`); conversely, if `how = 'right'`, then only rows in `B` will be retained in the output (so `A` will be merged *to* `B`).

A simple example of the use of `pd.merge` is illustrated below:

In [16]:
```python
# toy data frames
A = pd.DataFrame(
    {'shared_col': ['a', 'b', 'c'],
    'x1': [1, 2, 3],
    'x2': [4, 5, 6]}
)

B = pd.DataFrame(
    {'shared_col': ['a', 'b'],
    'y1': [7, 8]}
)
```

In [17]:
```python
A
```

Out[17]:

	shared_col	x1	x2
0	a	1	4
1	b	2	5
2	c	3	6

In [18]:
```python
B
```

Out[18]:

	shared_col	y1
0	a	7
1	b	8

Below, if `A` and `B` are merged retaining the rows in `A`, notice that a missing value is input because `B` has no row where the shared column (on which the merging is done) has value `c`. In other words, the third row of `A` has no match in `B`.

In [19]:
```python
# left join
pd.merge(A, B, how = 'left', on = 'shared_col')
```

Out[19]:

	shared_col	x1	x2	y1
0	a	1	4	7.0
1	b	2	5	8.0
2	c	3	6	NaN

If the direction of merging is reversed, and the row structure of `B` is dominant, then the third row of `A` is dropped altogether because it has no match in `B`.

In [20]:
```python
# right join
pd.merge(A, B, how = 'right', on = 'shared_col')
```

Out[20]:

	shared_col	x1	x2	y1
0	a	1	4	7
1	b	2	5	8

Submission Checklist

1. Save file to confirm all changes are on disk
2. Run *Kernel > Restart & Run All* to execute all code from top to bottom
3. Save file again to write any new output to disk
4. Select *File > Download as > HTML*.
5. Open in Google Chrome and print to PDF.
6. Submit to Gradescope