

# AQI Data ( Year- 2019, Year- 2020, Year-2021)

```
In [1]: #importing libraries
import pandas as pd
import numpy as np
```

```
In [2]: #Reading the AQI datasets of Year- 2019, Year- 2020, Year-2021
AQI_df1 = pd.read_csv('/Users/vishesh/Downloads/DA/daily_aqi_by_county_2019.csv'
AQI_df2 = pd.read_csv('/Users/vishesh/Downloads/DA/daily_aqi_by_county_2020.csv'
AQI_df3 = pd.read_csv('/Users/vishesh/Downloads/DA/daily_aqi_by_county_2021.csv')
```

## Checking the Dataframe

```
In [3]: # Top 5 Rows of AQI_Year-2019
AQI_df1.head()
```

Out[3]:

|   | State Name | county Name | State Code | County Code | Date       | AQI | Category | Defining Parameter | Defining Site | Number of Sites Reporting |
|---|------------|-------------|------------|-------------|------------|-----|----------|--------------------|---------------|---------------------------|
| 0 | Alabama    | Baldwin     | 1          | 3           | 2019-01-03 | 18  | Good     | PM2.5              | 01-003-0010   | 1                         |
| 1 | Alabama    | Baldwin     | 1          | 3           | 2019-01-06 | 35  | Good     | PM2.5              | 01-003-0010   | 1                         |
| 2 | Alabama    | Baldwin     | 1          | 3           | 2019-01-09 | 14  | Good     | PM2.5              | 01-003-0010   | 1                         |
| 3 | Alabama    | Baldwin     | 1          | 3           | 2019-01-12 | 36  | Good     | PM2.5              | 01-003-0010   | 1                         |
| 4 | Alabama    | Baldwin     | 1          | 3           | 2019-01-15 | 38  | Good     | PM2.5              | 01-003-0010   | 1                         |

```
In [4]: # Top 5 Rows of AQI_Year-2020
AQI_df2.head()
```

Out[4]:

|   | State Name | county Name | State Code | County Code | Date       | AQI | Category | Defining Parameter | Defining Site | Number of Sites Reporting |
|---|------------|-------------|------------|-------------|------------|-----|----------|--------------------|---------------|---------------------------|
| 0 | Alabama    | Baldwin     | 1          | 3           | 2020-01-01 | 48  | Good     | PM2.5              | 01-003-0010   | 1                         |
| 1 | Alabama    | Baldwin     | 1          | 3           | 2020-01-04 | 13  | Good     | PM2.5              | 01-003-0010   | 1                         |
| 2 | Alabama    | Baldwin     | 1          | 3           | 2020-01-07 | 14  | Good     | PM2.5              | 01-003-0010   | 1                         |
| 3 | Alabama    | Baldwin     | 1          | 3           | 2020-01-10 | 39  | Good     | PM2.5              | 01-003-0010   | 1                         |
| 4 | Alabama    | Baldwin     | 1          | 3           | 2020-01-13 | 29  | Good     | PM2.5              | 01-003-0010   | 1                         |

In [5]: # Top 5 Rows of AQI\_Year-2021  
AQI\_df3.head()

Out[5]:

|   | State Name | county Name | State Code | County Code | Date       | AQI | Category | Defining Parameter | Defining Site | Number of Sites Reporting |
|---|------------|-------------|------------|-------------|------------|-----|----------|--------------------|---------------|---------------------------|
| 0 | Alabama    | Baldwin     | 1          | 3           | 2021-01-01 | 27  | Good     | PM2.5              | 01-003-0010   | 1                         |
| 1 | Alabama    | Baldwin     | 1          | 3           | 2021-01-04 | 47  | Good     | PM2.5              | 01-003-0010   | 1                         |
| 2 | Alabama    | Baldwin     | 1          | 3           | 2021-01-07 | 24  | Good     | PM2.5              | 01-003-0010   | 1                         |
| 3 | Alabama    | Baldwin     | 1          | 3           | 2021-01-10 | 39  | Good     | PM2.5              | 01-003-0010   | 1                         |
| 4 | Alabama    | Baldwin     | 1          | 3           | 2021-01-13 | 46  | Good     | PM2.5              | 01-003-0010   | 1                         |

## Data Wrangling and Cleaning

In [6]: # Summary of the AQI\_df1 DataFrame, including the number of non-null values in AQI\_df1.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 325331 entries, 0 to 325330
Data columns (total 10 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   State Name       325331 non-null   object 
 1   county Name      325331 non-null   object 
 2   State Code        325331 non-null   int64  
 3   County Code       325331 non-null   int64  
 4   Date              325331 non-null   object 
 5   AQI               325331 non-null   int64  
 6   Category          325331 non-null   object 
 7   Defining Parameter 325331 non-null   object 
 8   Defining Site      325331 non-null   object 
 9   Number of Sites Reporting 325331 non-null   int64  
dtypes: int64(4), object(6)
memory usage: 24.8+ MB
```

In [7]: # Summary of the AQI\_df2 DataFrame, including the number of non-null values in AQI\_df2.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 324338 entries, 0 to 324337
Data columns (total 10 columns):
 #   Column           Non-Null Count   Dtype  
--- 
 0   State Name       324338 non-null    object  
 1   county Name      324338 non-null    object  
 2   State Code        324338 non-null    int64   
 3   County Code       324338 non-null    int64   
 4   Date              324338 non-null    object  
 5   AQI               324338 non-null    int64   
 6   Category          324338 non-null    object  
 7   Defining Parameter 324338 non-null    object  
 8   Defining Site      324338 non-null    object  
 9   Number of Sites Reporting 324338 non-null    int64  
dtypes: int64(4), object(6)
memory usage: 24.7+ MB
```

In [8]: `# Summary of the AQI_df3 DataFrame, including the number of non-null values in AQI_df3.info()`

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 218196 entries, 0 to 218195
Data columns (total 10 columns):
 #   Column           Non-Null Count   Dtype  
--- 
 0   State Name       218196 non-null    object  
 1   county Name      218196 non-null    object  
 2   State Code        218196 non-null    int64   
 3   County Code       218196 non-null    int64   
 4   Date              218196 non-null    object  
 5   AQI               218196 non-null    int64   
 6   Category          218196 non-null    object  
 7   Defining Parameter 218196 non-null    object  
 8   Defining Site      218196 non-null    object  
 9   Number of Sites Reporting 218196 non-null    int64  
dtypes: int64(4), object(6)
memory usage: 16.6+ MB
```

In [9]: `# checking null values in AQI_df1 DataFrame`  
`AQI_df1.isna().sum()`

Out[9]:

|                           |   |
|---------------------------|---|
| State Name                | 0 |
| county Name               | 0 |
| State Code                | 0 |
| County Code               | 0 |
| Date                      | 0 |
| AQI                       | 0 |
| Category                  | 0 |
| Defining Parameter        | 0 |
| Defining Site             | 0 |
| Number of Sites Reporting | 0 |

dtype: int64

In [10]: `# checking null values in AQI_df2 DataFrame`  
`AQI_df2.isna().sum()`

```
Out[10]: State Name          0  
county Name        0  
State Code          0  
County Code         0  
Date                0  
AQI                 0  
Category             0  
Defining Parameter   0  
Defining Site        0  
Number of Sites Reporting 0  
dtype: int64
```

```
In [11]: # checking null values in AQI_df3 DataFrame  
AQI_df3.isna().sum()
```

```
Out[11]: State Name          0  
county Name        0  
State Code          0  
County Code         0  
Date                0  
AQI                 0  
Category             0  
Defining Parameter   0  
Defining Site        0  
Number of Sites Reporting 0  
dtype: int64
```

According to the 3 Dataframes it means that all the three datasets have the same number of columns and the column names are also the same in all the three datasets. So, we can concatenate the three datasets together into one and the resulting DataFrame will have the same structure as the individual DataFrames. It will be easier to analyze and work with one large dataset rather than working with three different datasets.

```
In [12]: # Concatenating the three AQI Dataframes  
AQI_df = pd.concat([AQI_df1, AQI_df2, AQI_df3], ignore_index=True)
```

```
In [13]: # Checking the Concatenated dataframe first few rows  
AQI_df.head()
```

Out[13]:

|   | State Name | county Name | State Code | County Code | Date       | AQI | Category | Defining Parameter | Defining Site | Number of Sites Reporting |
|---|------------|-------------|------------|-------------|------------|-----|----------|--------------------|---------------|---------------------------|
| 0 | Alabama    | Baldwin     | 1          | 3           | 2019-01-03 | 18  | Good     | PM2.5              | 01-003-0010   | 1                         |
| 1 | Alabama    | Baldwin     | 1          | 3           | 2019-01-06 | 35  | Good     | PM2.5              | 01-003-0010   | 1                         |
| 2 | Alabama    | Baldwin     | 1          | 3           | 2019-01-09 | 14  | Good     | PM2.5              | 01-003-0010   | 1                         |
| 3 | Alabama    | Baldwin     | 1          | 3           | 2019-01-12 | 36  | Good     | PM2.5              | 01-003-0010   | 1                         |
| 4 | Alabama    | Baldwin     | 1          | 3           | 2019-01-15 | 38  | Good     | PM2.5              | 01-003-0010   | 1                         |

In [14]:

```
# Checking the Concatenated dataframe last few rows
AQI_df.tail()
```

Out[14]:

|        | State Name | county Name | State Code | County Code | Date       | AQI | Category | Defining Parameter | Defining Site | Number of Site Reportin |
|--------|------------|-------------|------------|-------------|------------|-----|----------|--------------------|---------------|-------------------------|
| 867860 | Wyoming    | Weston      | 56         | 45          | 2021-06-26 | 41  | Good     | Ozone              | 56-045-0003   |                         |
| 867861 | Wyoming    | Weston      | 56         | 45          | 2021-06-27 | 40  | Good     | Ozone              | 56-045-0003   |                         |
| 867862 | Wyoming    | Weston      | 56         | 45          | 2021-06-28 | 41  | Good     | Ozone              | 56-045-0003   |                         |
| 867863 | Wyoming    | Weston      | 56         | 45          | 2021-06-29 | 48  | Good     | Ozone              | 56-045-0003   |                         |
| 867864 | Wyoming    | Weston      | 56         | 45          | 2021-06-30 | 54  | Moderate | Ozone              | 56-045-0003   |                         |

In [15]:

```
# Checking the number of columns and rows
AQI_df.shape
```

Out[15]:

(867865, 10)

In [16]:

```
# Dropping unnecessary columns from the dataframe
AQI_df = AQI_df.drop(columns = ['Defining Parameter', 'Defining Site', 'Number
```

In [17]:

```
# Checking the random rows
AQI_df.sample(5)
```

Out[17]:

|        | State Name     | State Code | Date       | AQI | Category |
|--------|----------------|------------|------------|-----|----------|
| 808471 | Oregon         | 41         | 2021-04-14 | 16  | Good     |
| 577331 | South Carolina | 45         | 2020-04-30 | 11  | Good     |
| 265903 | Tennessee      | 47         | 2019-09-14 | 49  | Good     |
| 313491 | Wisconsin      | 55         | 2019-10-12 | 32  | Good     |
| 230385 | Oregon         | 41         | 2019-05-13 | 19  | Good     |

In [18]:

```
# Checking the final information again in order to process further
AQI_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 867865 entries, 0 to 867864
Data columns (total 5 columns):
 #   Column      Non-Null Count  Dtype  
---  --          -----          ----  
 0   State Name  867865 non-null  object  
 1   State Code  867865 non-null  int64  
 2   Date        867865 non-null  object  
 3   AQI         867865 non-null  int64  
 4   Category    867865 non-null  object  
dtypes: int64(2), object(3)
memory usage: 33.1+ MB
```

**It's important to make sure that the data in the column is in the format that can be converted to the desired datatype, otherwise it will throw an error.**

In [19]:

```
# converting the 'Date' column of AQI_df DataFrame into datetime datatype.
AQI_df['Date'] = pd.to_datetime(AQI_df['Date'])

# converting the 'State Name' column of AQI_df DataFrame into a string datatype
AQI_df['State_Name'] = AQI_df['State Name'].astype(str)
```

In [20]:

```
# Setting the index
AQI_df.set_index("Date", inplace=True)
```

In [21]:

```
# Changing the format of date to Y-M
AQI_df['year_month'] = AQI_df.index.strftime('%Y-%m')

# Grouping the AQI_df DataFrame by 'State_Name' and 'year_month' columns and then
df_monthly = AQI_df.groupby(['State_Name', 'year_month'], as_index=False).mean()
```

**It is a useful technique when you want to analyze or visualize data based on a specific time period (in this case, month) or when you want to aggregate the data based on a specific column.**

In [22]: `df_monthly.head()`

Out[22]:

|   | State_Name | year_month | State Code | AQI       |
|---|------------|------------|------------|-----------|
| 0 | Alabama    | 2019-01    | 1.0        | 32.812925 |
| 1 | Alabama    | 2019-02    | 1.0        | 31.391608 |
| 2 | Alabama    | 2019-03    | 1.0        | 43.827214 |
| 3 | Alabama    | 2019-04    | 1.0        | 44.484979 |
| 4 | Alabama    | 2019-05    | 1.0        | 42.683992 |

In [23]: `df_monthly.shape`

Out[23]: (1796, 4)

In [24]: `df_monthly.dtypes`

Out[24]:

|            |         |
|------------|---------|
| State_Name | object  |
| year_month | object  |
| State Code | float64 |
| AQI        | float64 |
| dtype:     | object  |

In [25]: `# changing the format of statecode type from float to INT  
df_monthly['State Code'] = df_monthly['State Code'].astype(int)`

In [26]:

```
import warnings
warnings.filterwarnings('ignore')
df_monthly['year_month'] = pd.to_datetime(df_monthly['year_month'], format='%Y-%m')
df_monthly['Month'] = df_monthly['year_month'].dt.strftime('%m')
df_monthly['Year'] = df_monthly['year_month'].dt.strftime('%y')
df_monthly['Month'] = df_monthly['Month'].astype(int)
df_monthly['Year'] = df_monthly['Year'].astype(str)
```

In [27]: `df_monthly.head()`

Out[27]:

|   | State_Name | year_month | State Code | AQI       | Month | Year |
|---|------------|------------|------------|-----------|-------|------|
| 0 | Alabama    | 2019-01-01 | 1          | 32.812925 | 1     | 19   |
| 1 | Alabama    | 2019-02-01 | 1          | 31.391608 | 2     | 19   |
| 2 | Alabama    | 2019-03-01 | 1          | 43.827214 | 3     | 19   |
| 3 | Alabama    | 2019-04-01 | 1          | 44.484979 | 4     | 19   |
| 4 | Alabama    | 2019-05-01 | 1          | 42.683992 | 5     | 19   |

In [28]: `df_monthly['State_Name'].unique()`

```
Out[28]: array(['Alabama', 'Alaska', 'Arizona', 'Arkansas', 'California',
   'Colorado', 'Connecticut', 'Country Of Mexico', 'Delaware',
   'District Of Columbia', 'Florida', 'Georgia', 'Hawaii', 'Idaho',
   'Illinois', 'Indiana', 'Iowa', 'Kansas', 'Kentucky', 'Louisiana',
   'Maine', 'Maryland', 'Massachusetts', 'Michigan', 'Minnesota',
   'Mississippi', 'Missouri', 'Montana', 'Nebraska', 'Nevada',
   'New Hampshire', 'New Jersey', 'New Mexico', 'New York',
   'North Carolina', 'North Dakota', 'Ohio', 'Oklahoma', 'Oregon',
   'Pennsylvania', 'Puerto Rico', 'Rhode Island', 'South Carolina',
   'South Dakota', 'Tennessee', 'Texas', 'Utah', 'Vermont',
   'Virgin Islands', 'Virginia', 'Washington', 'West Virginia',
   'Wisconsin', 'Wyoming'], dtype=object)
```

```
In [29]: country_list=['Alabama', 'Alaska', 'Arizona', 'Arkansas', 'California',
   'Colorado', 'Connecticut', 'Country Of Mexico', 'Delaware',
   'District Of Columbia', 'Florida', 'Georgia', 'Hawaii', 'Idaho',
   'Illinois', 'Indiana', 'Iowa', 'Kansas', 'Kentucky', 'Louisiana',
   'Maine', 'Maryland', 'Massachusetts', 'Michigan', 'Minnesota',
   'Mississippi', 'Missouri', 'Montana', 'Nebraska', 'Nevada',
   'New Hampshire', 'New Jersey', 'New Mexico', 'New York',
   'North Carolina', 'North Dakota', 'Ohio', 'Oklahoma', 'Oregon',
   'Pennsylvania', 'Puerto Rico', 'Rhode Island', 'South Carolina',
   'South Dakota', 'Tennessee', 'Texas', 'Utah', 'Vermont',
   'Virgin Islands', 'Virginia', 'Washington', 'West Virginia',
   'Wisconsin', 'Wyoming']
```

## { Data Visualization }

```
In [30]: # importing these libraries for data visualization and creating different types

# Seaborn library
import seaborn as sns

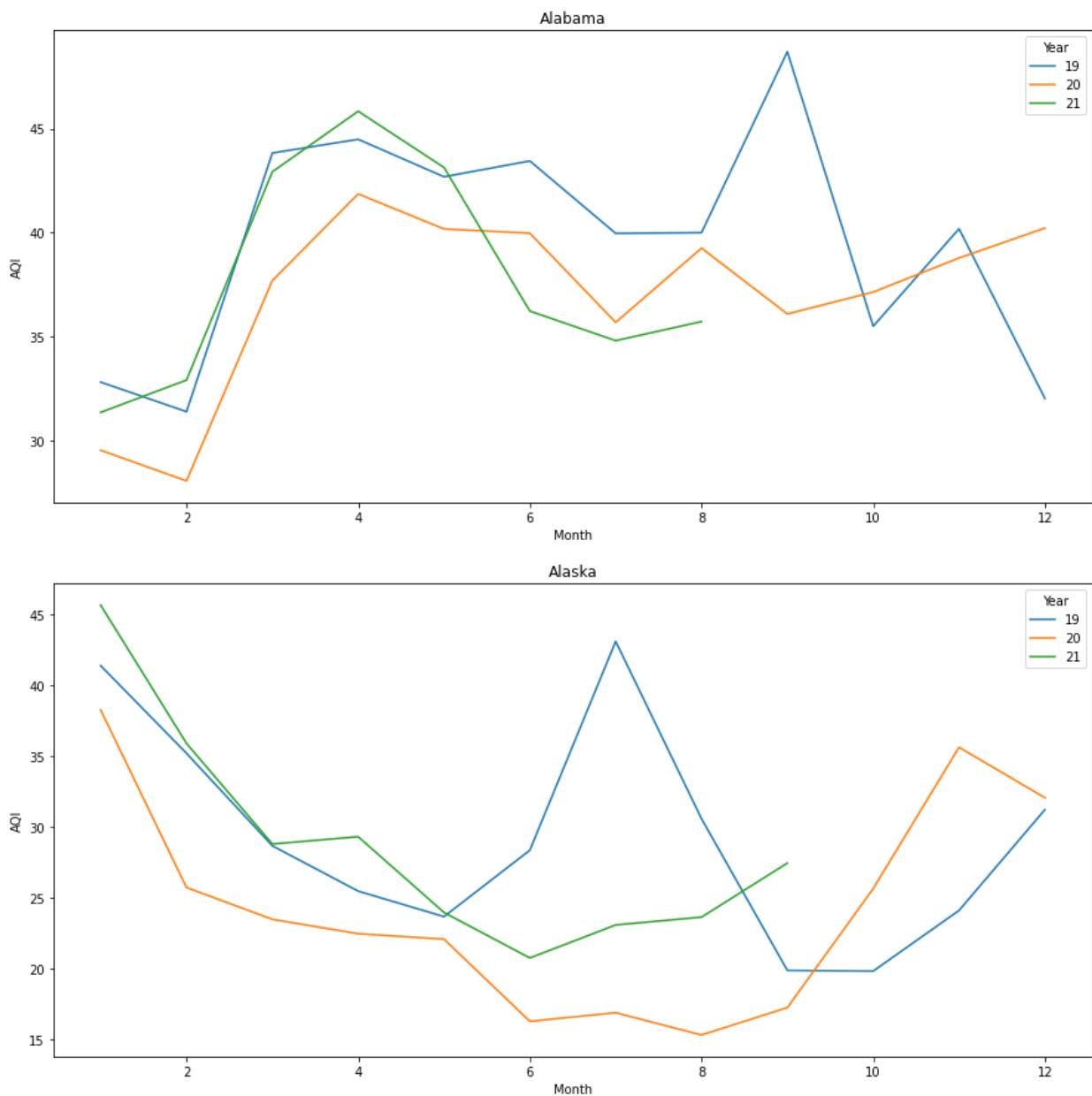
#matplotlib library
import matplotlib
import matplotlib.pyplot as plt
```

The below cell creates line plots of AQI data for each country over a 3-year period (2019-2020-2021), with the blue line representing the data for the year 2019, the orange line representing the data for the year 2020, and the green line representing the data for the year 2021.

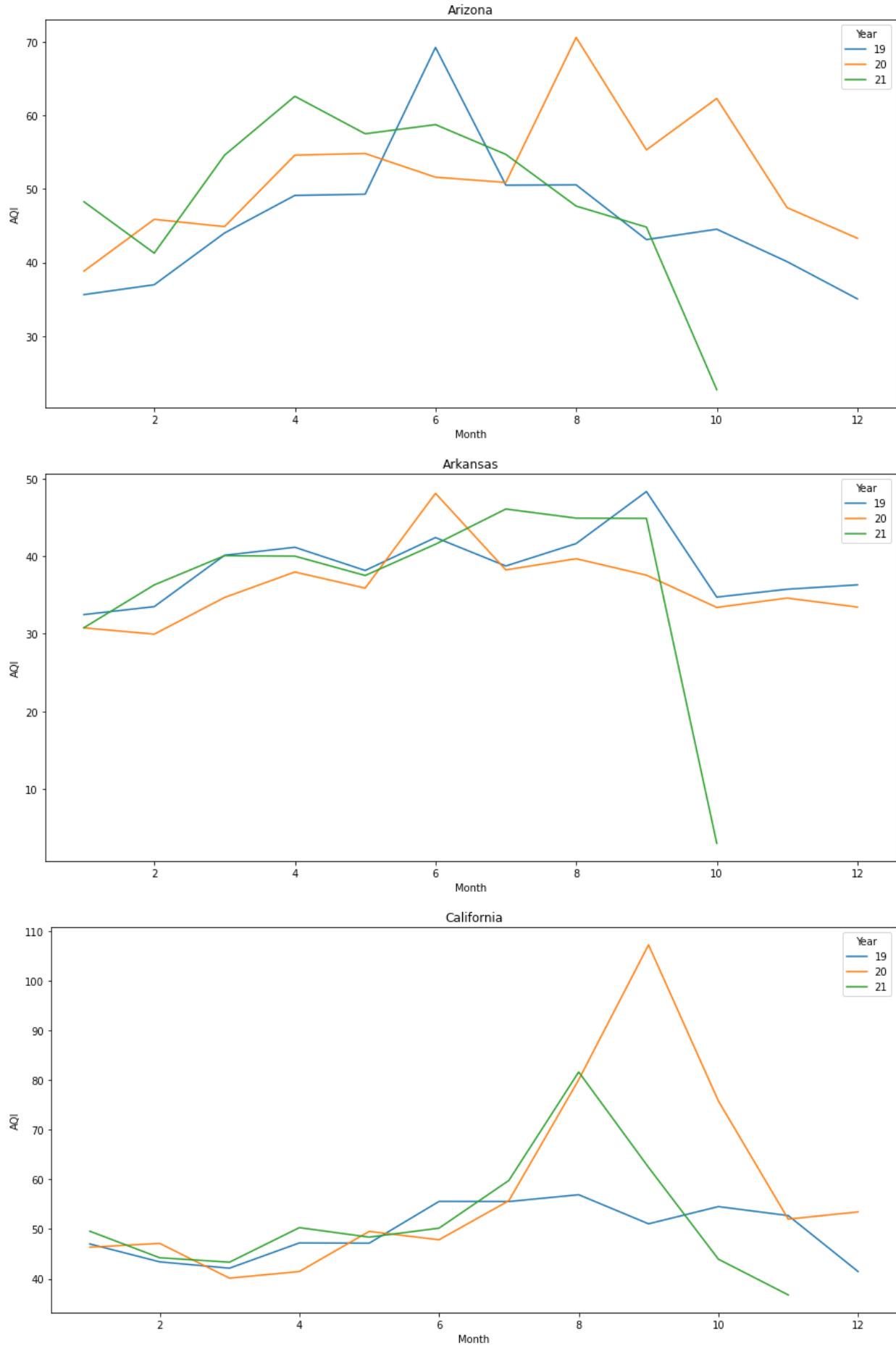
This allows for easy visualization of AQI trends for each country over the 3-year period and makes it easy to see how AQI levels changed during the lockdown period from month February to April .

It is easy to understand if the AQI was high or low during the lockdown period for each country, and it also allows to compare AQI levels between different years. Overall, the visualizations of all the countries provides a useful charts for analyzing AQI data and understanding how it has changed over time.

```
In [31]: for country in country_list:
    cn=[ ]
    cn.append(country)
    df_plotter = pd.DataFrame()
    df_plotter=df_monthly[df_monthly['State_Name'].isin(cn)]  
  
    plt.figure(figsize=(15,7))
    country= sns.lineplot(x='Month',y='AQI',data=df_plotter, hue='Year').set(tit
```

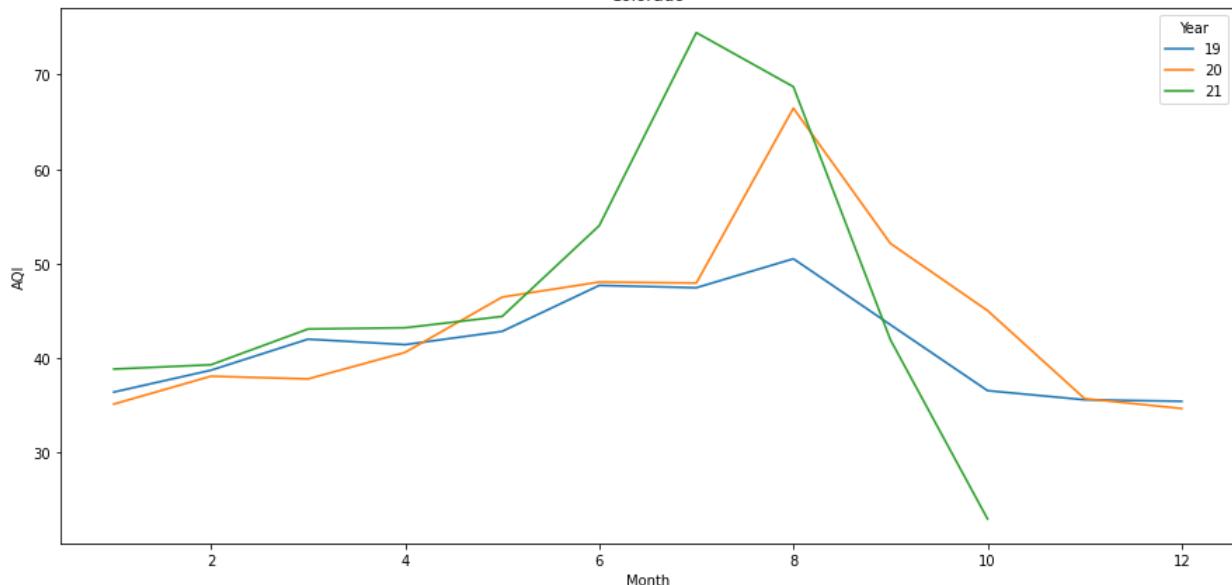


## TASK1(AQI)

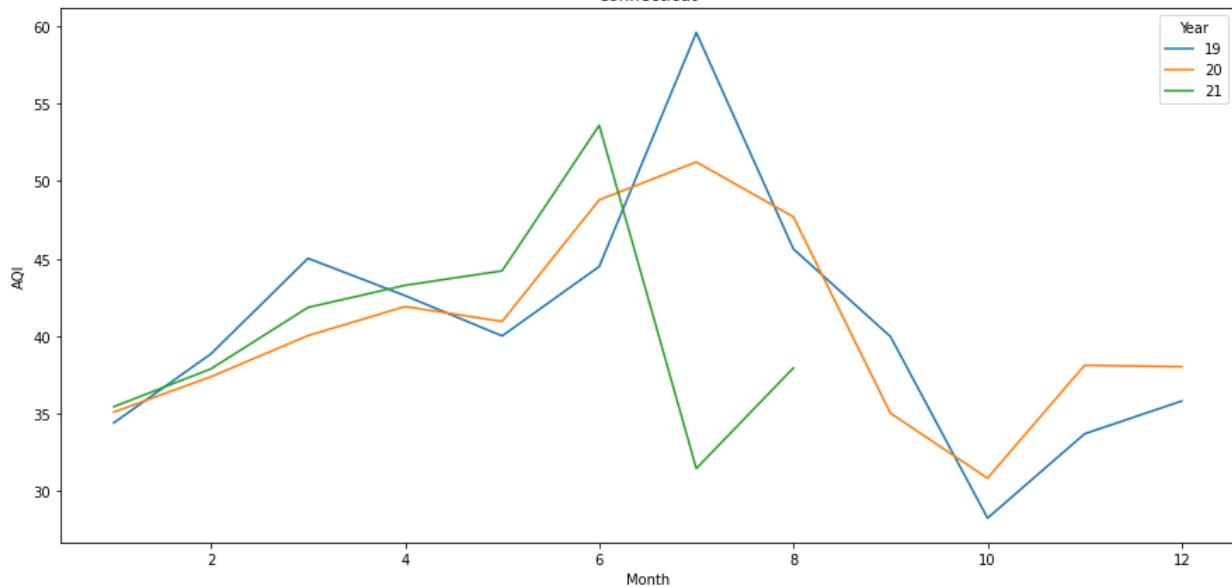


## TASK1(AQI)

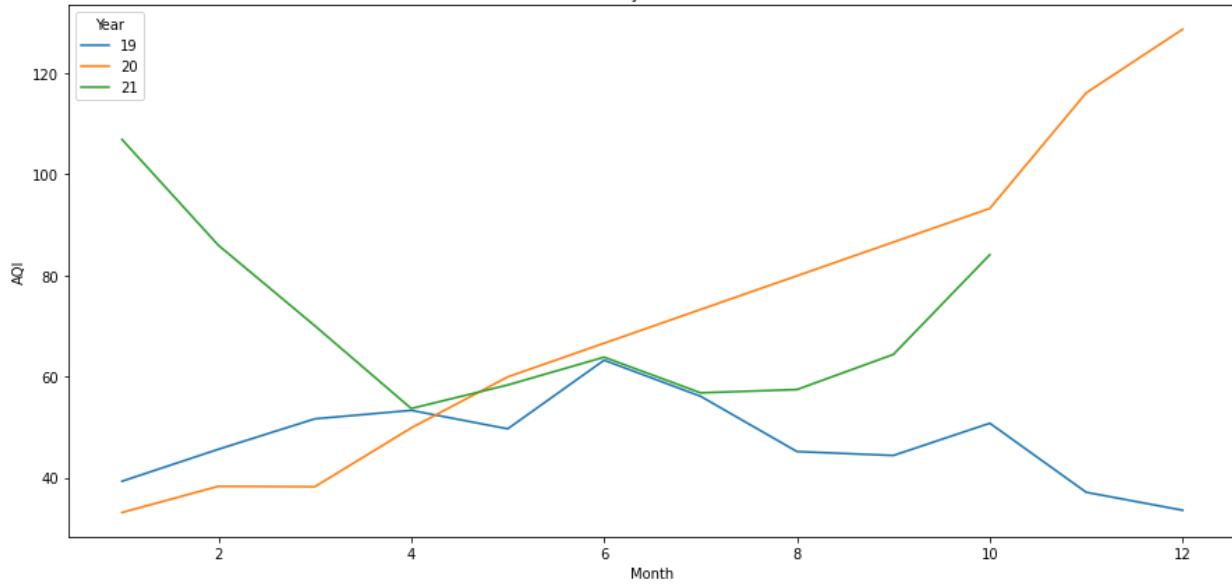
Colorado



Connecticut

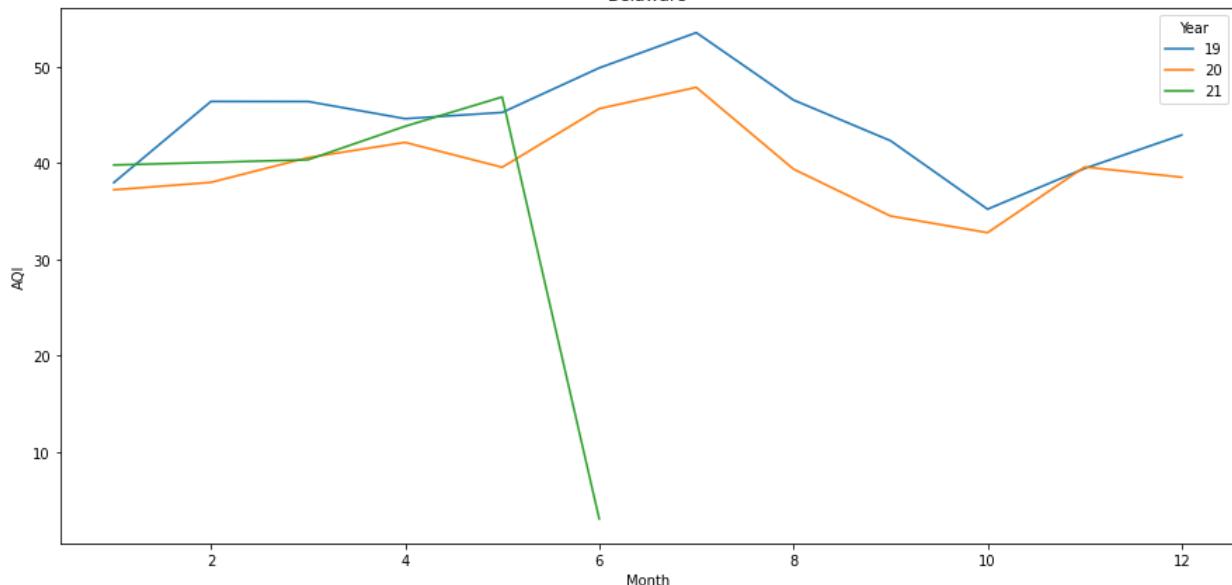


Country Of Mexico

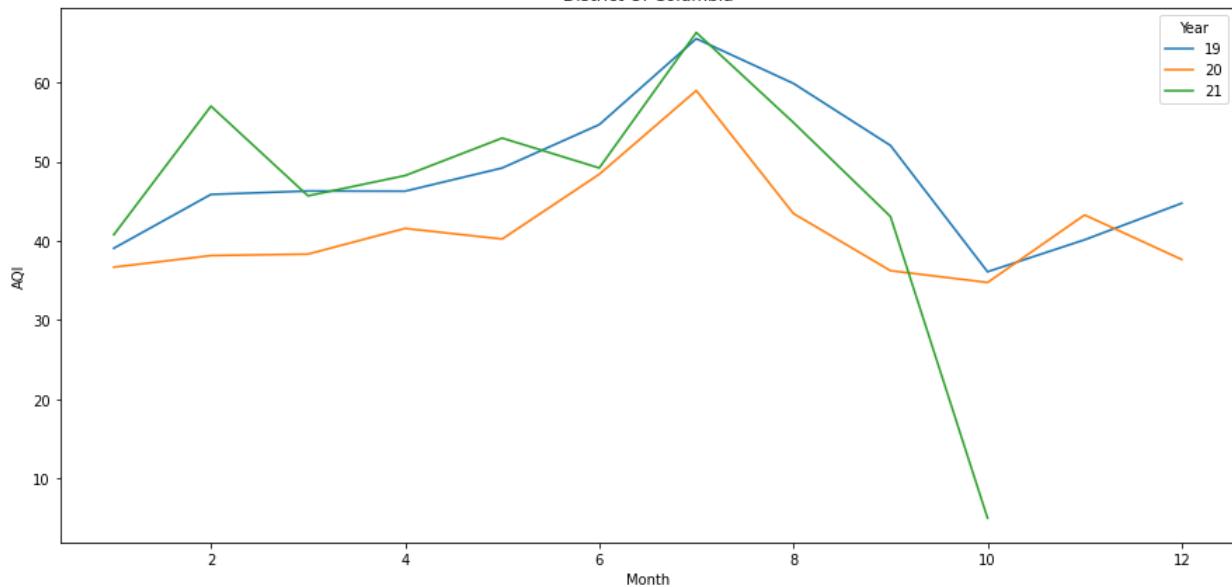


## TASK1(AQI)

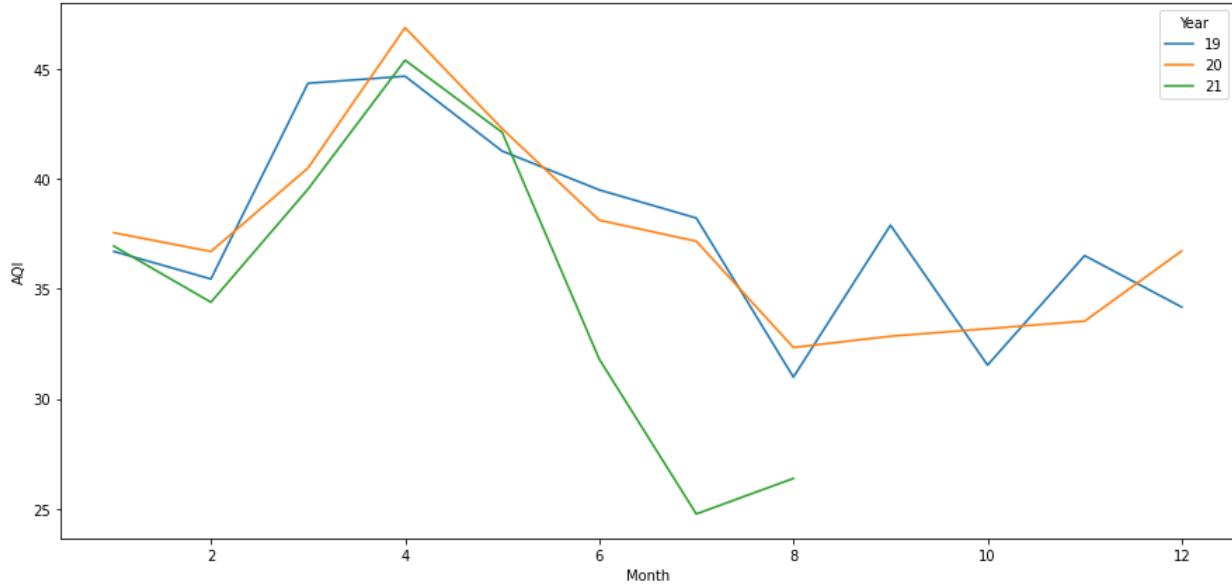
Delaware



District Of Columbia

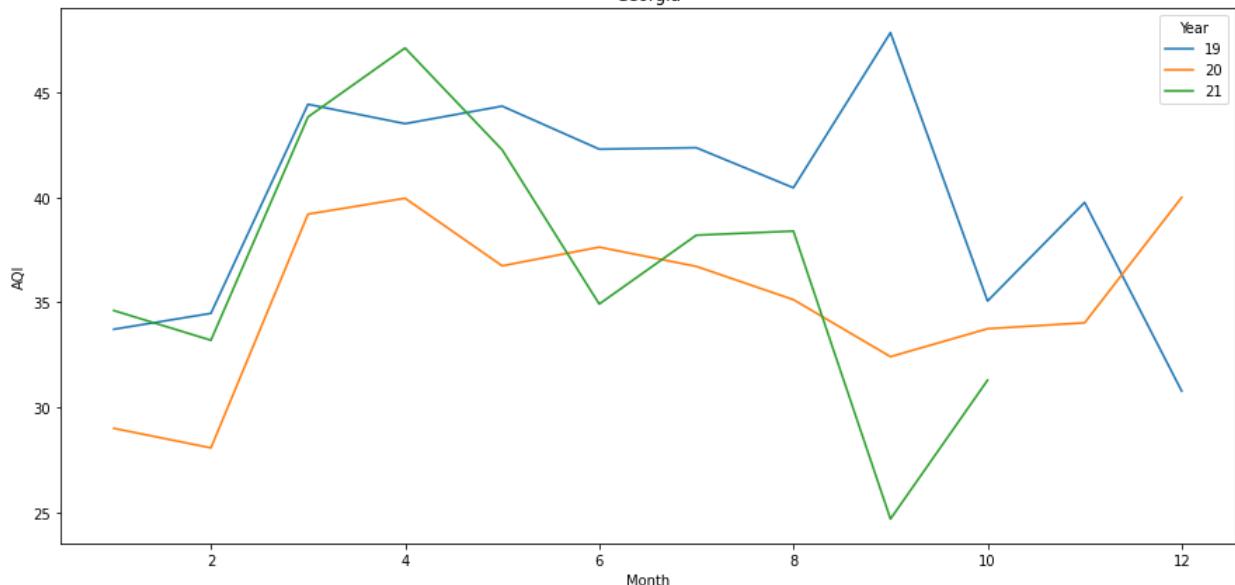


Florida

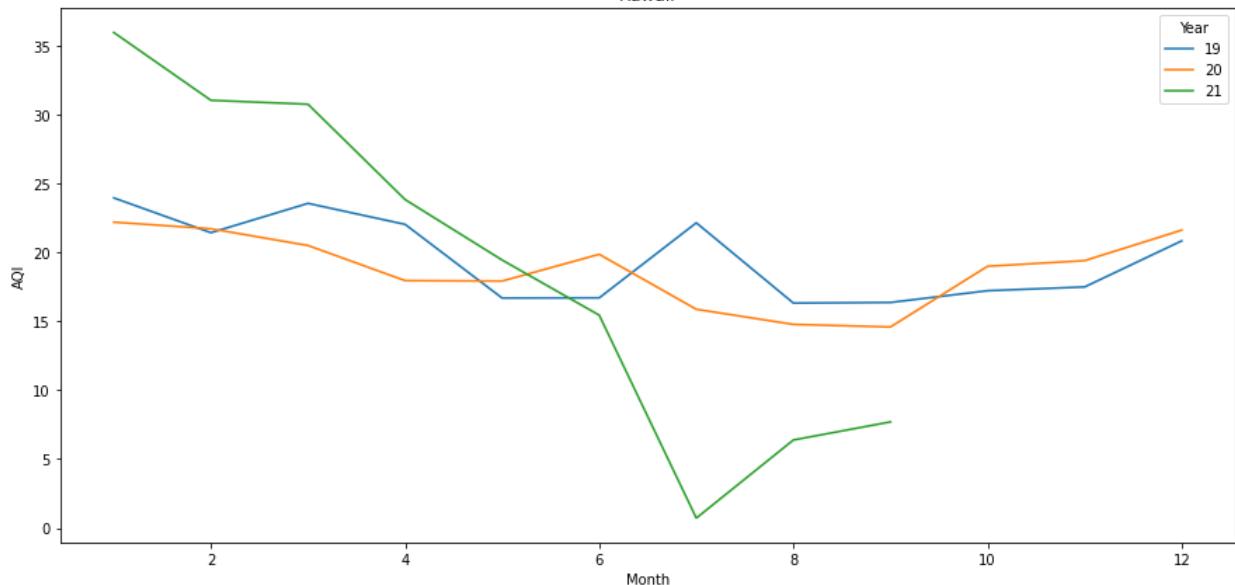


## TASK1(AQI)

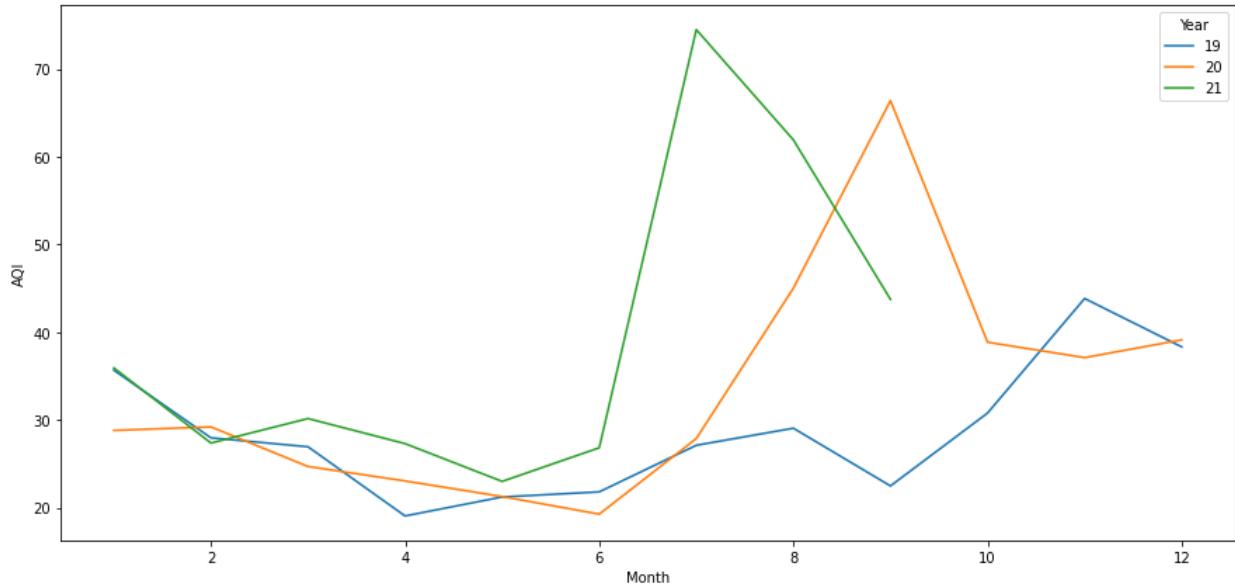
Georgia



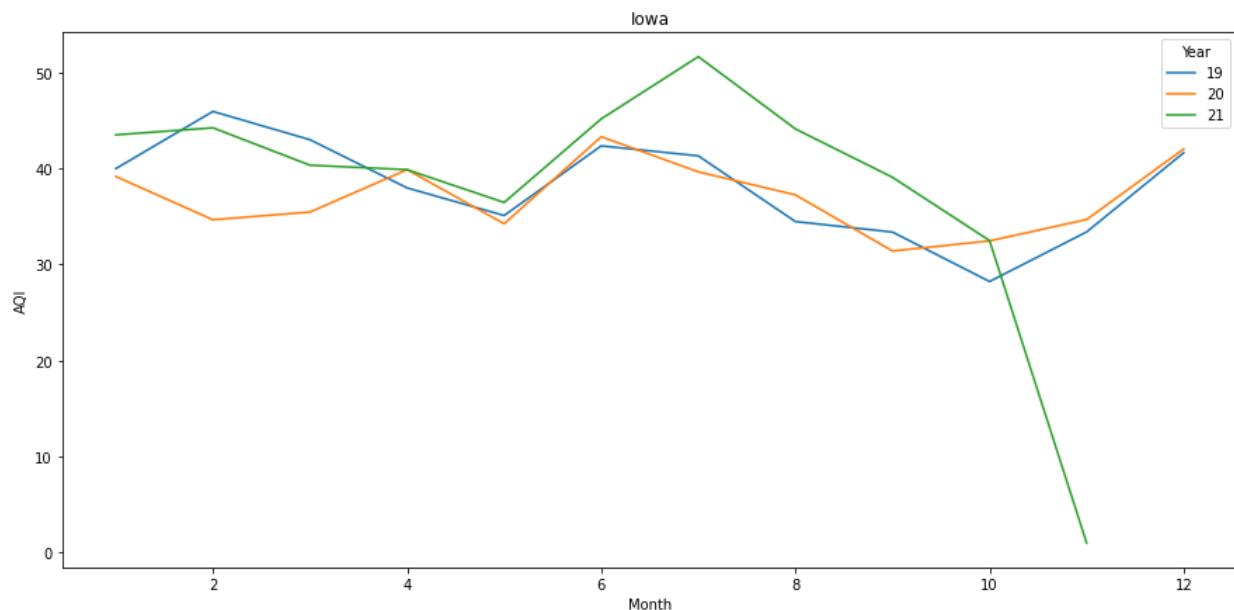
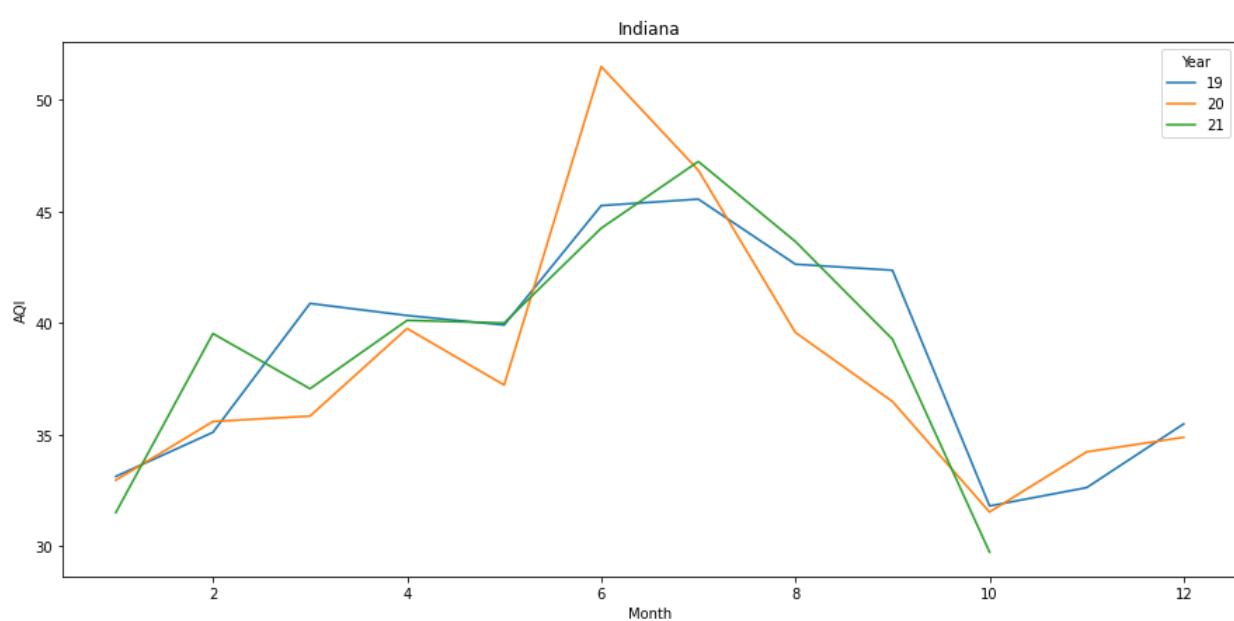
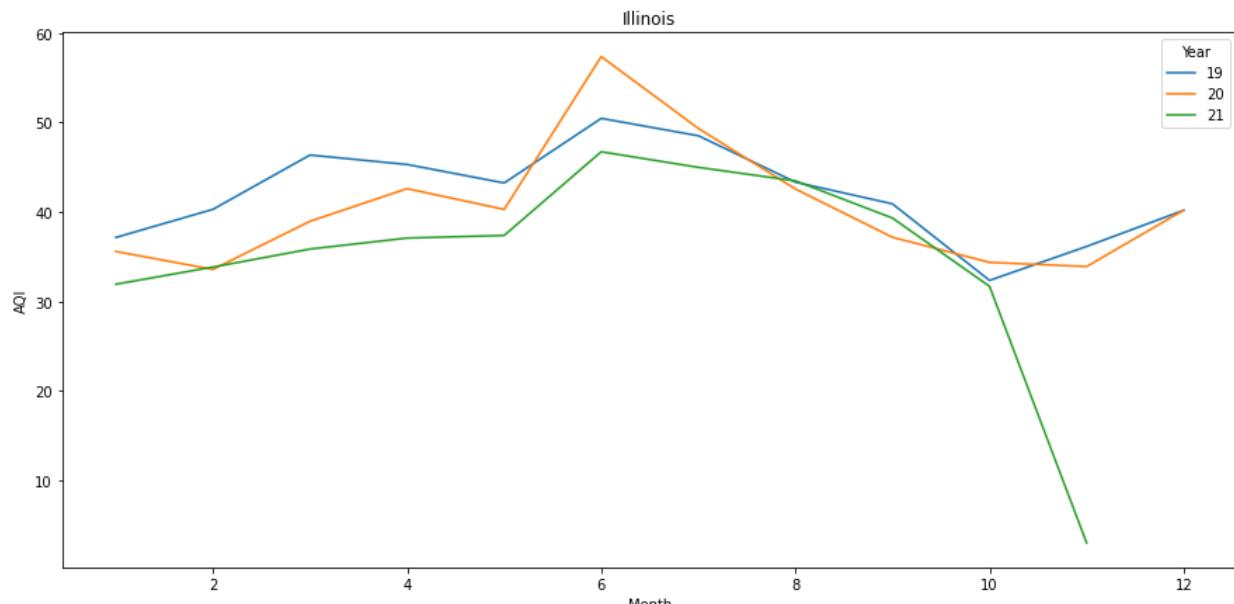
Hawaii



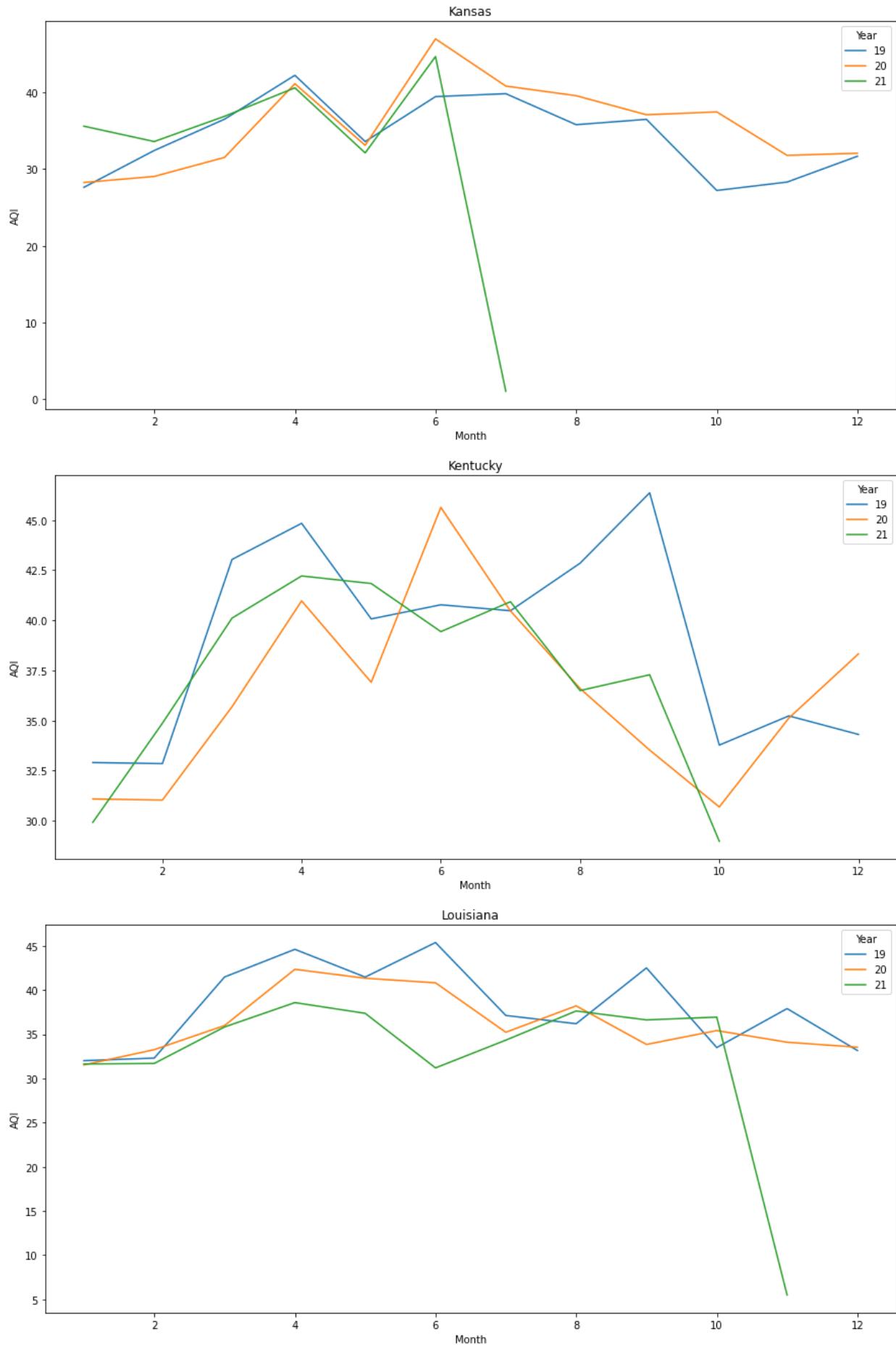
Idaho



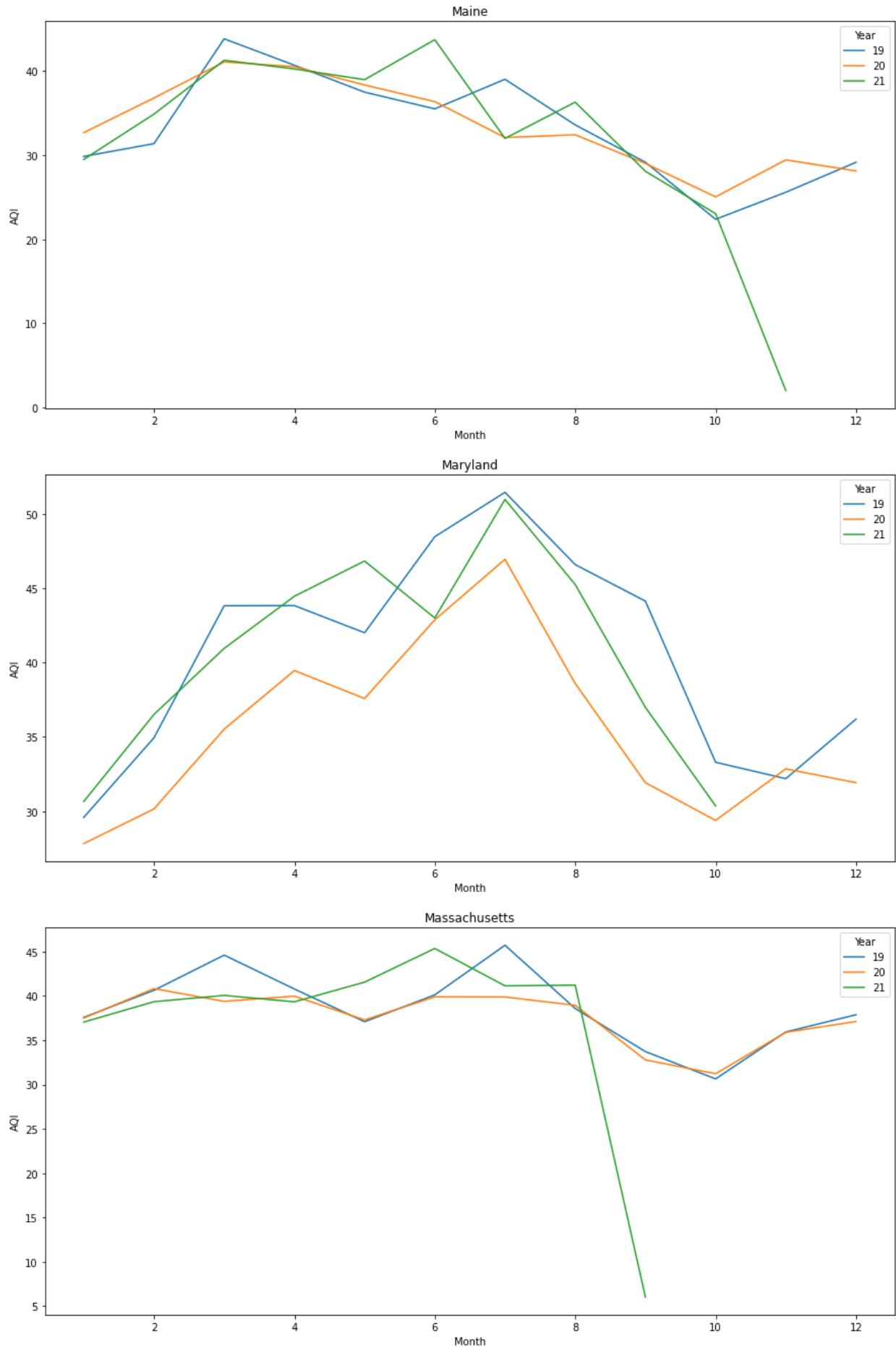
## TASK1(AQI)



## TASK1(AQI)

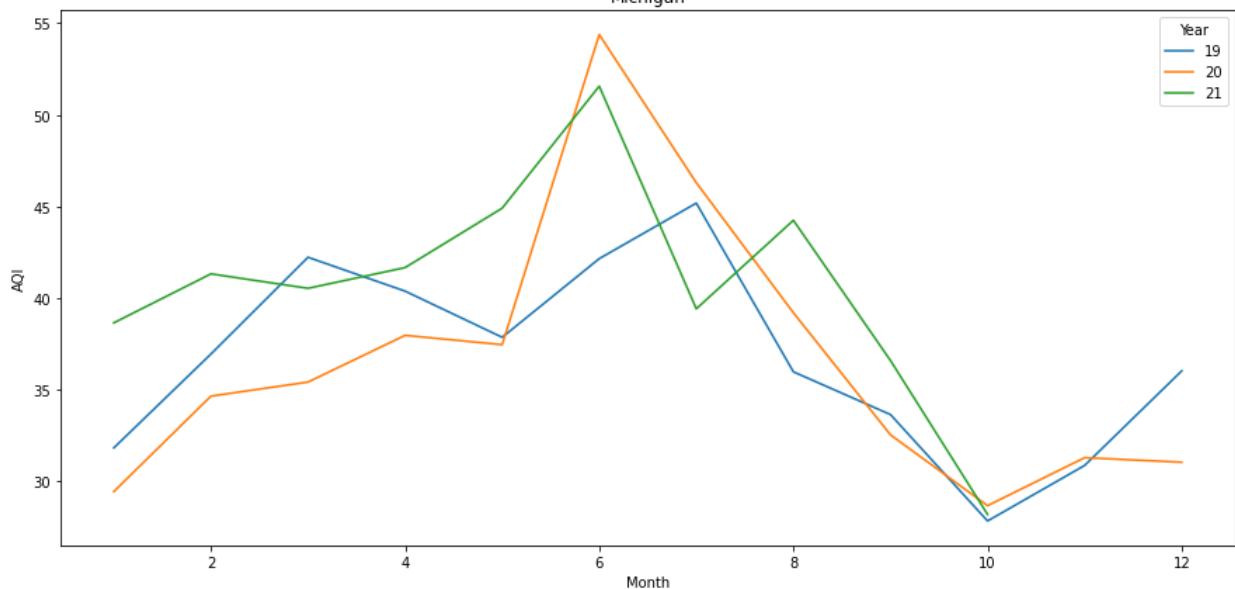


## TASK1(AQI)

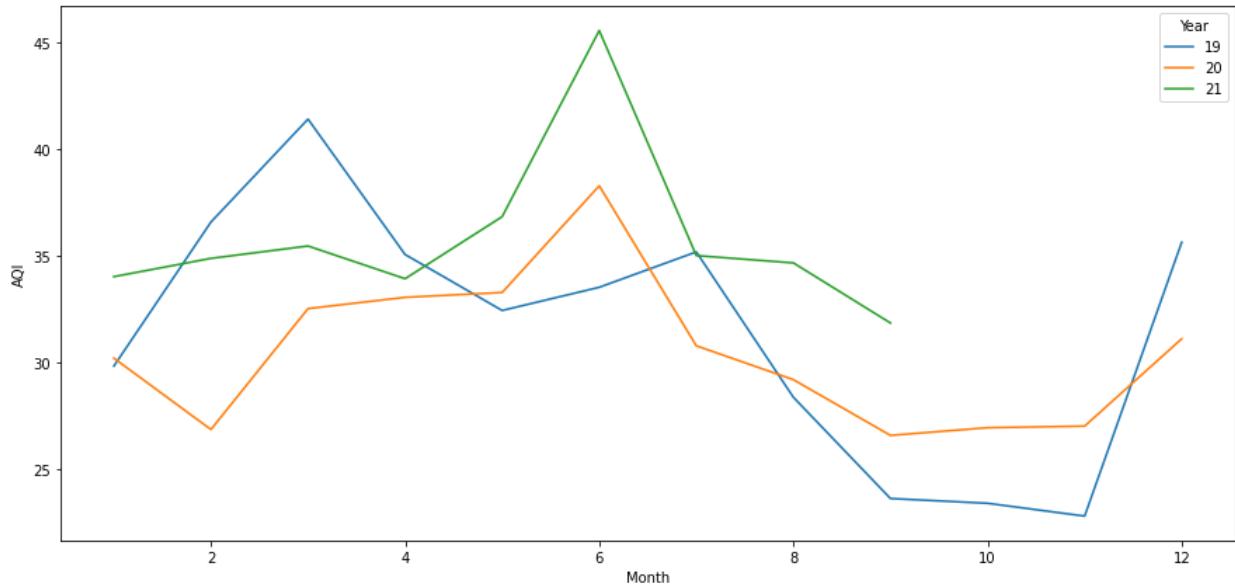


## TASK1(AQI)

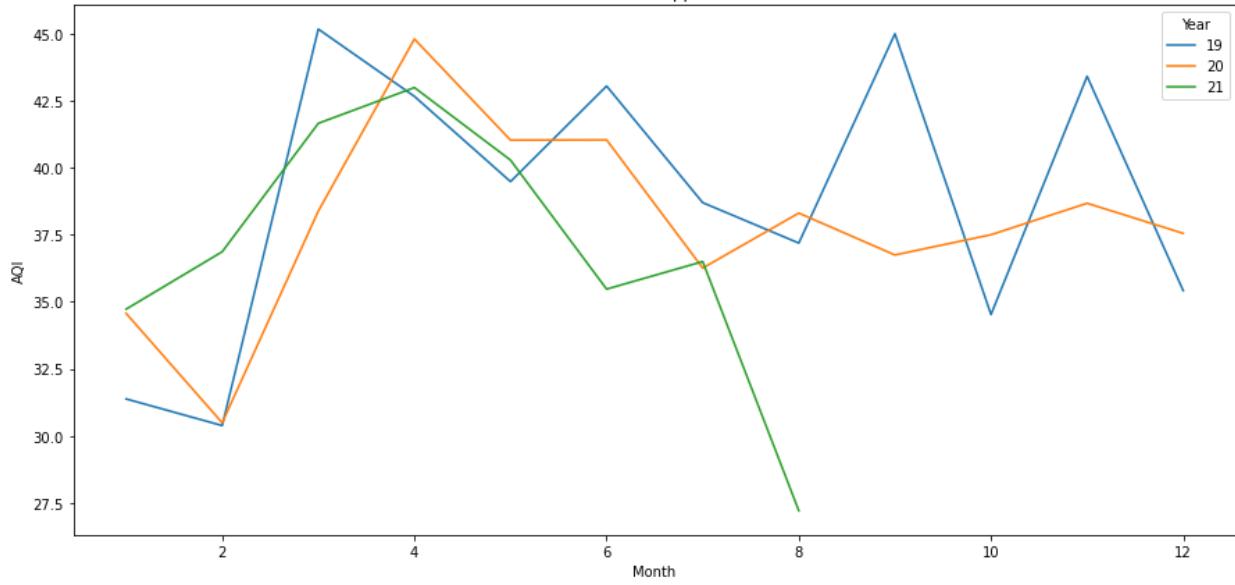
Michigan



Minnesota

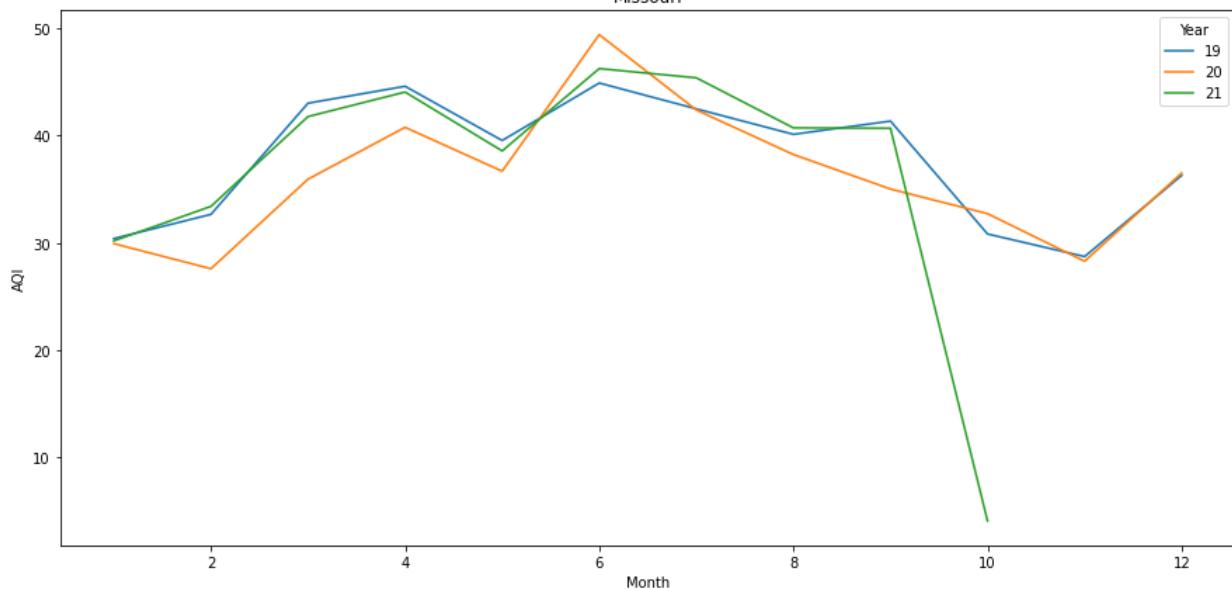


Mississippi

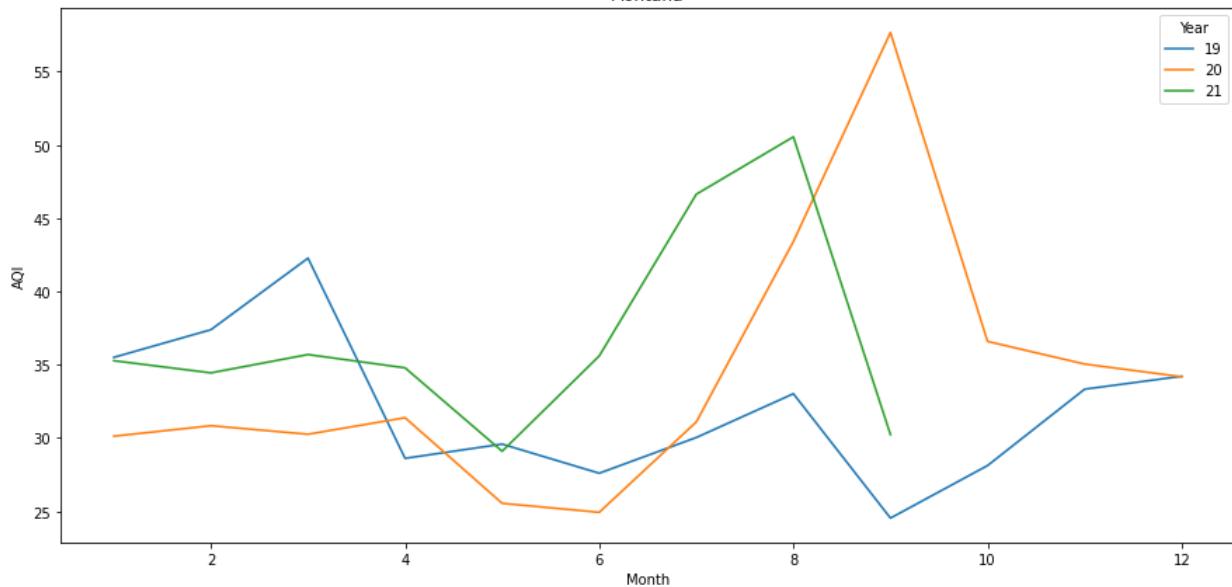


## TASK1(AQI)

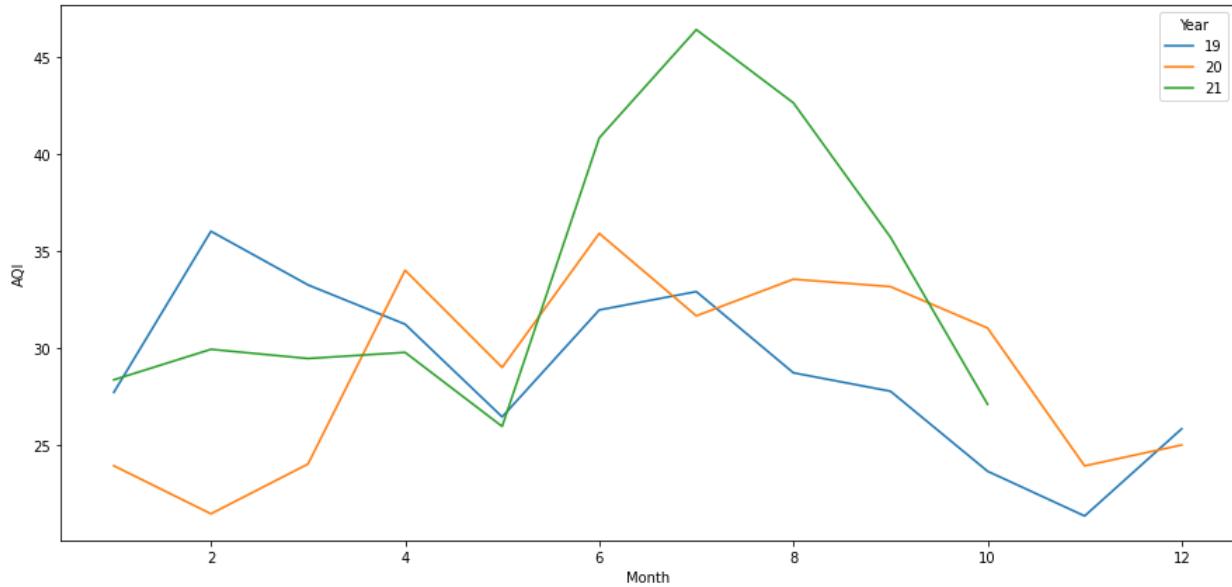
Missouri



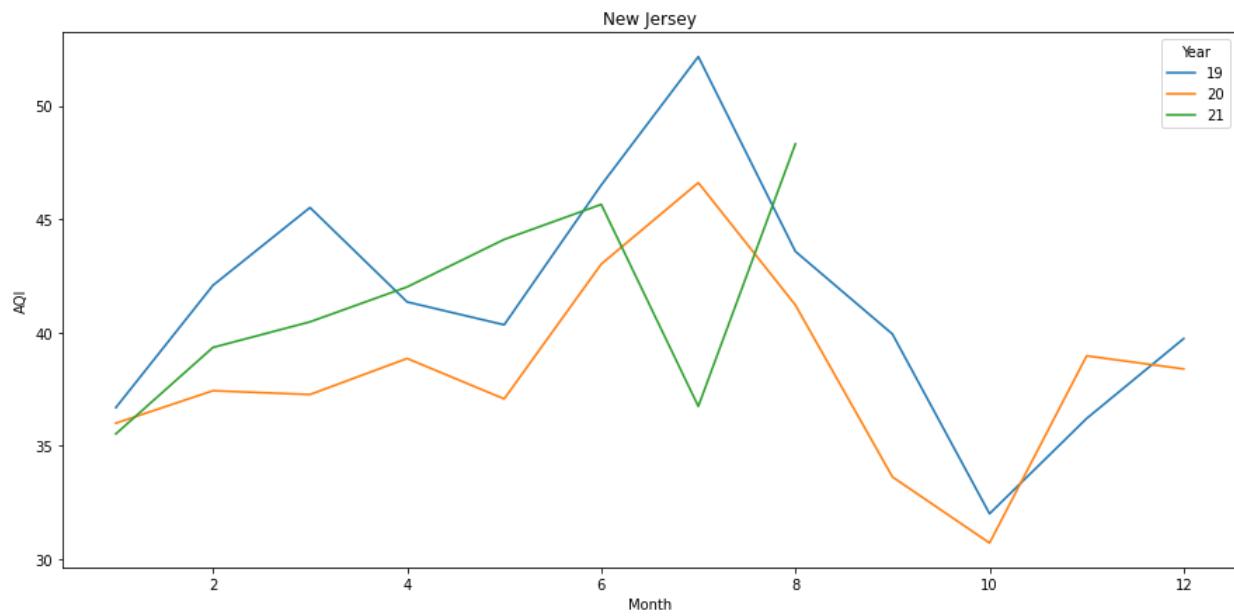
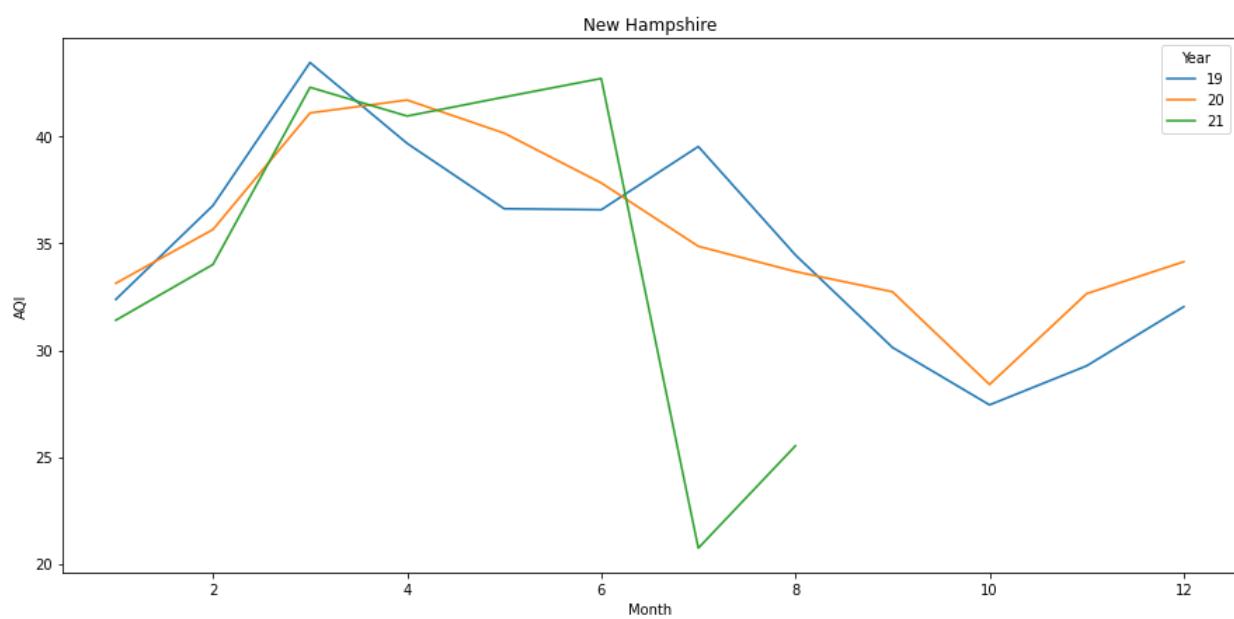
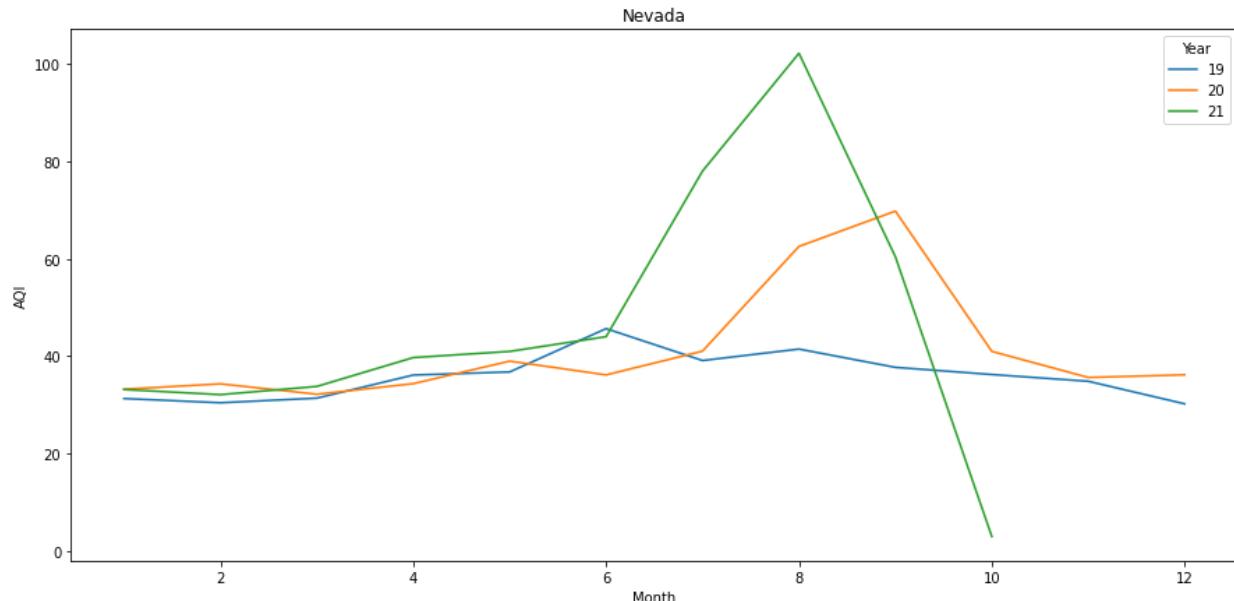
Montana



Nebraska

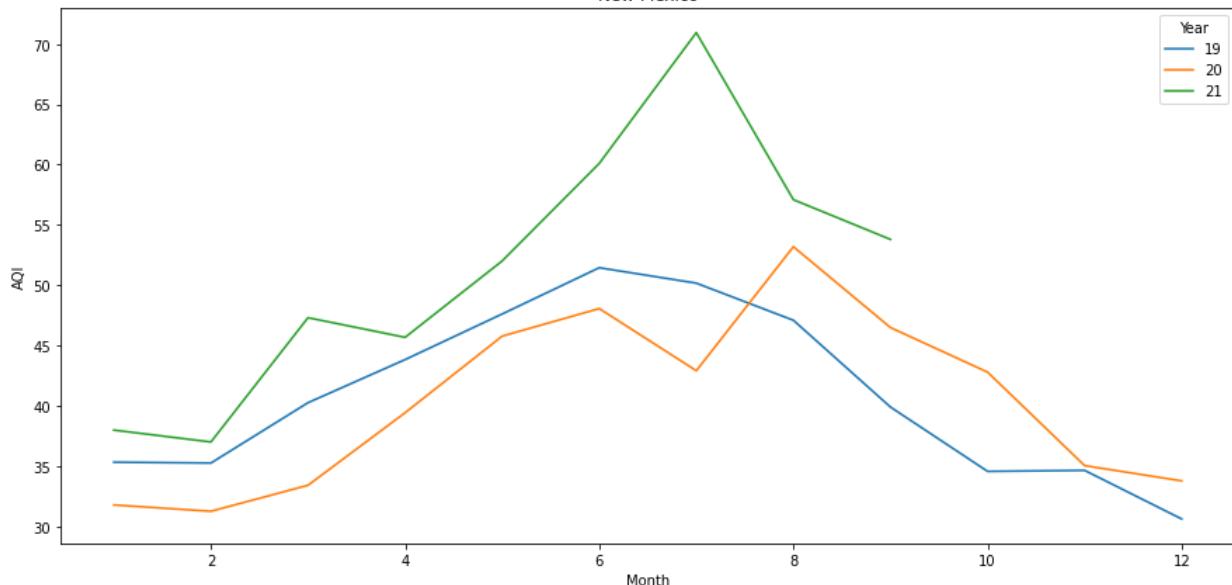


## TASK1(AQI)

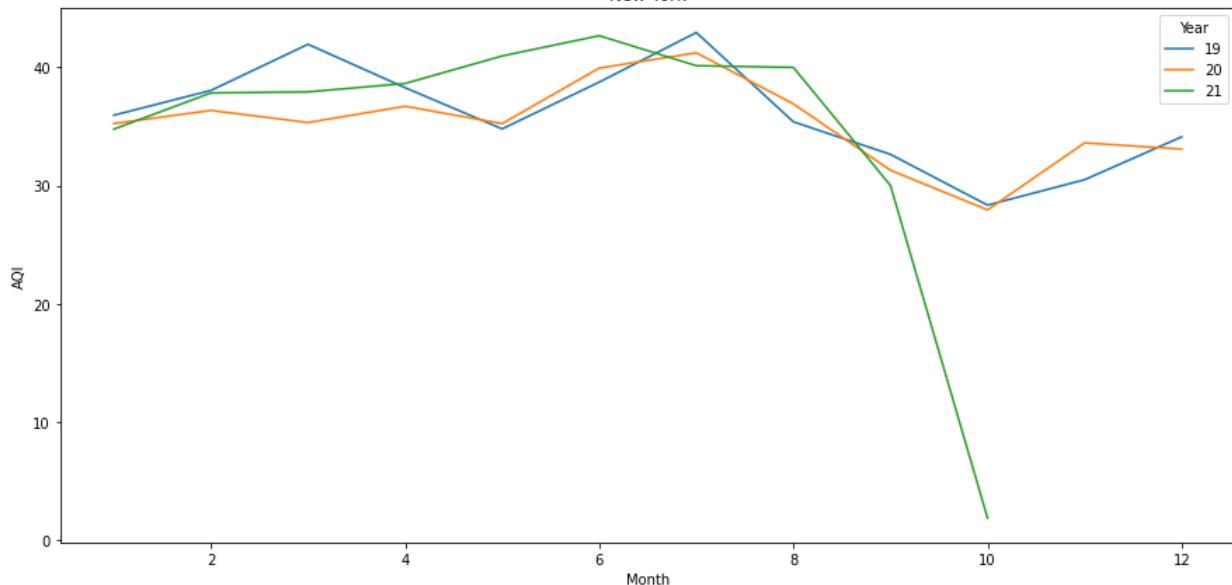


## TASK1(AQI)

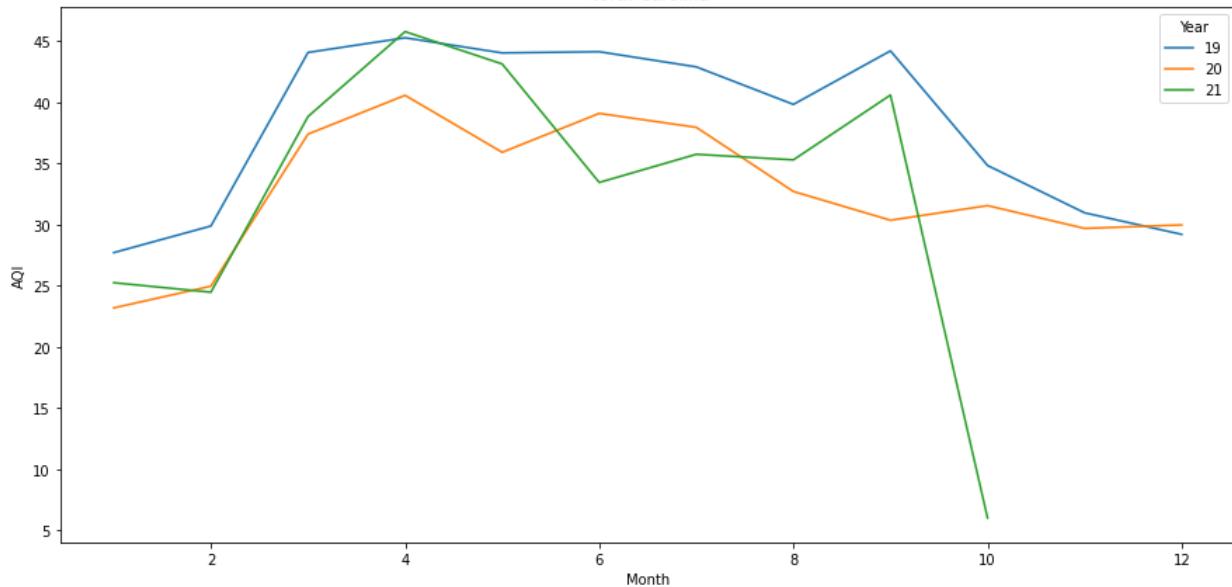
New Mexico



New York

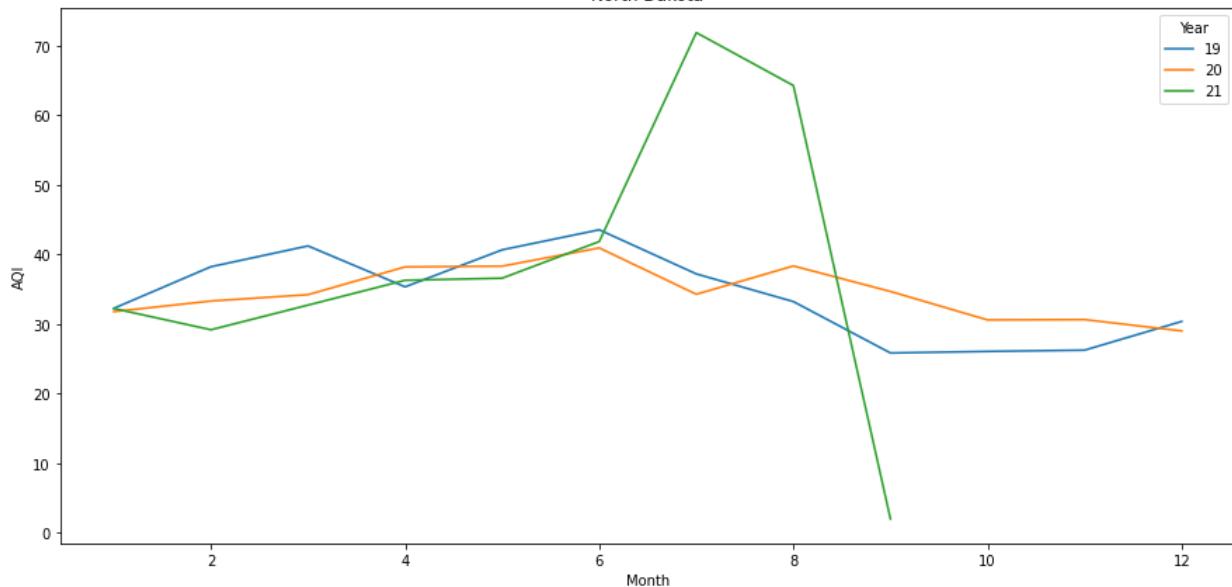


North Carolina

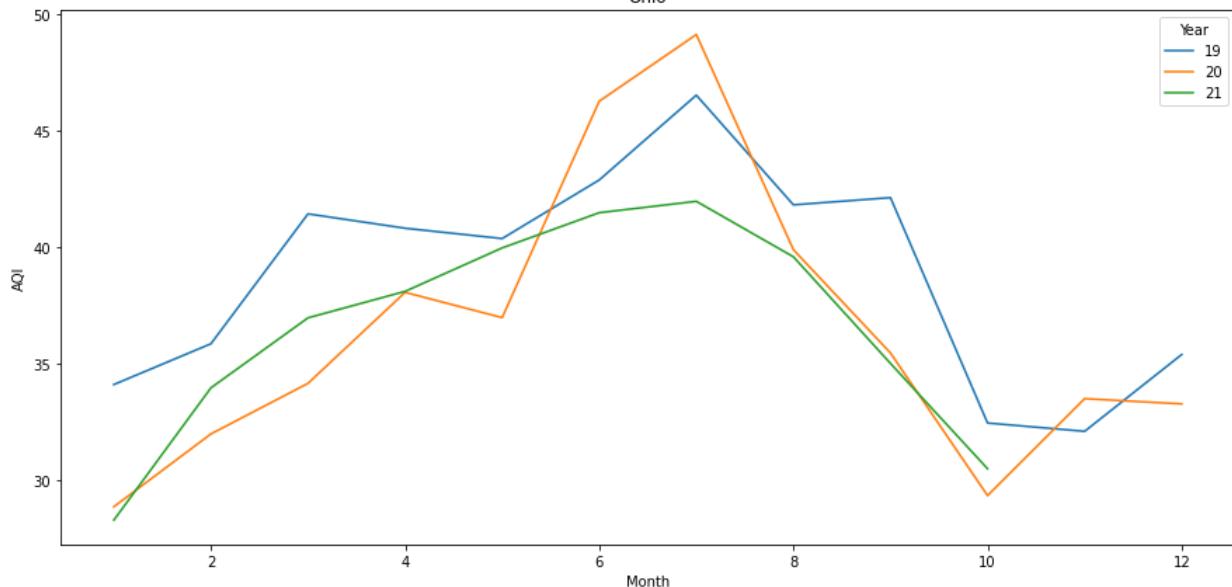


## TASK1(AQI)

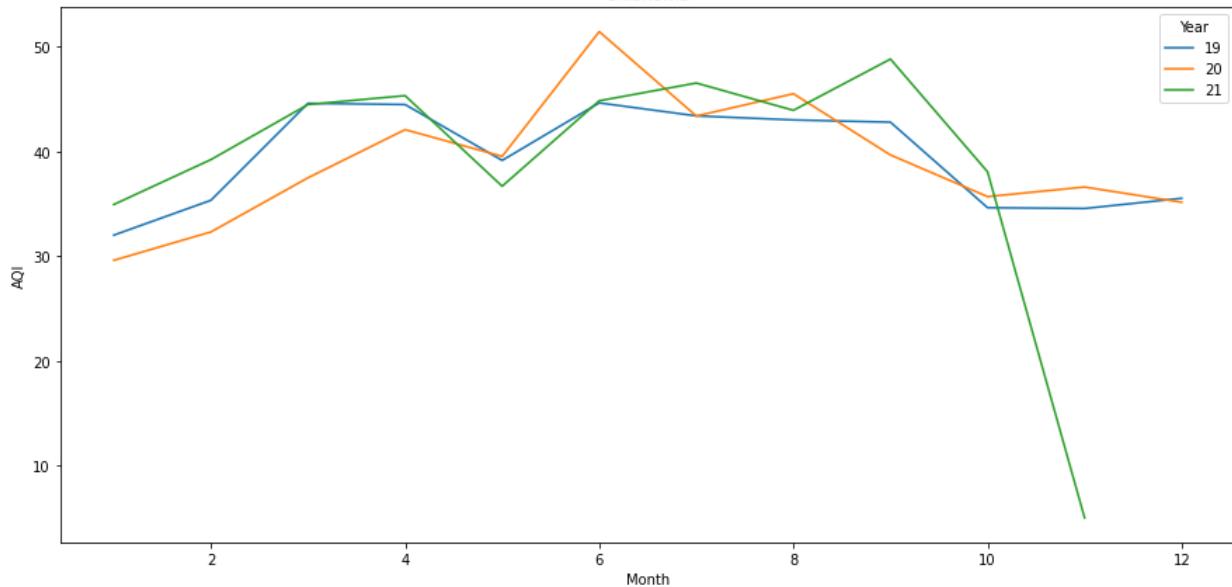
North Dakota



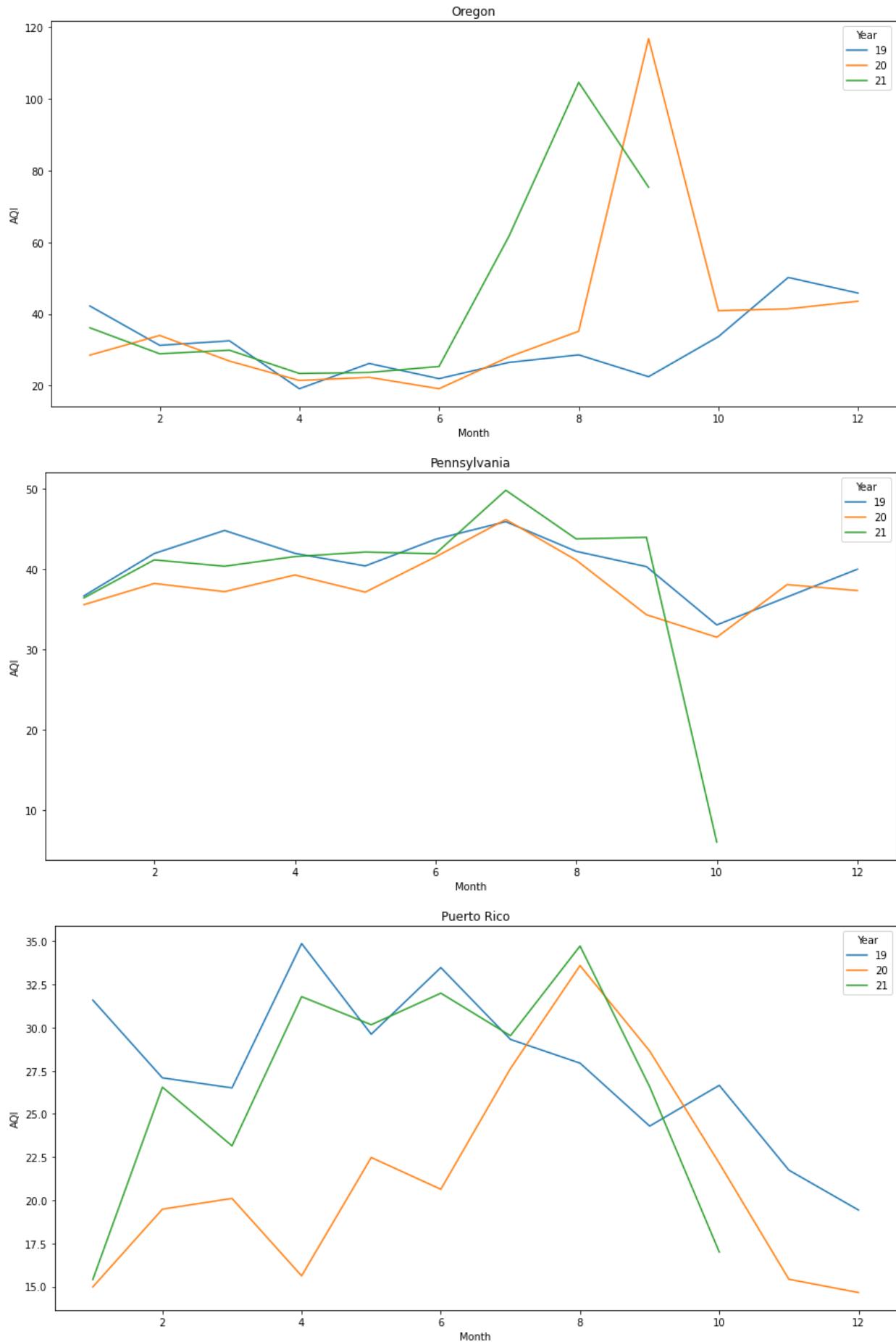
Ohio



Oklahoma

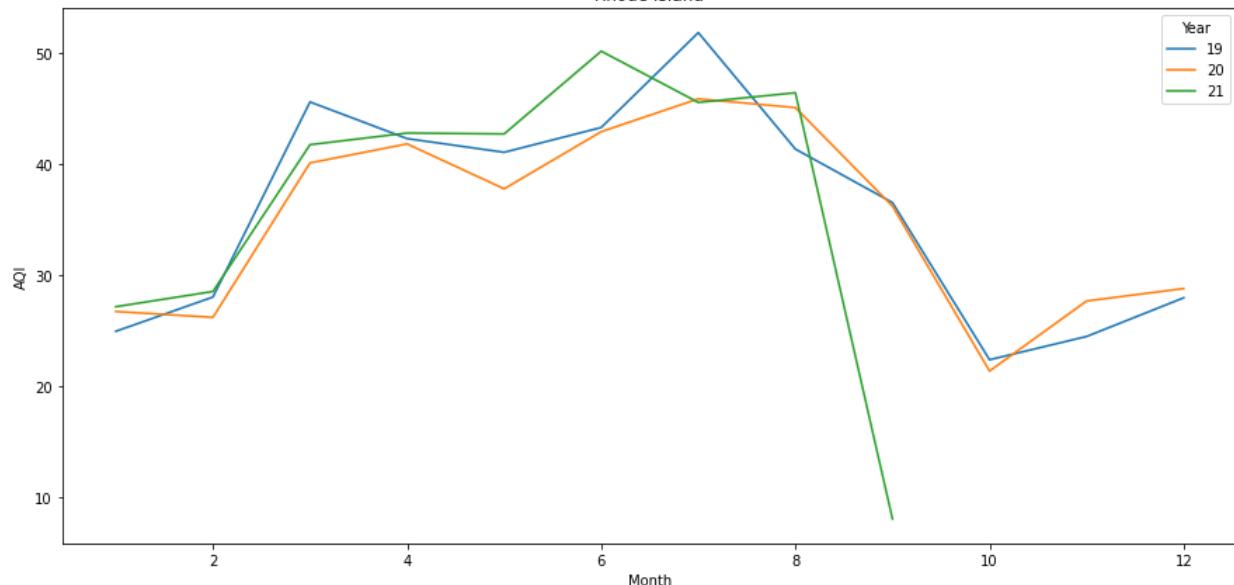


## TASK1(AQI)

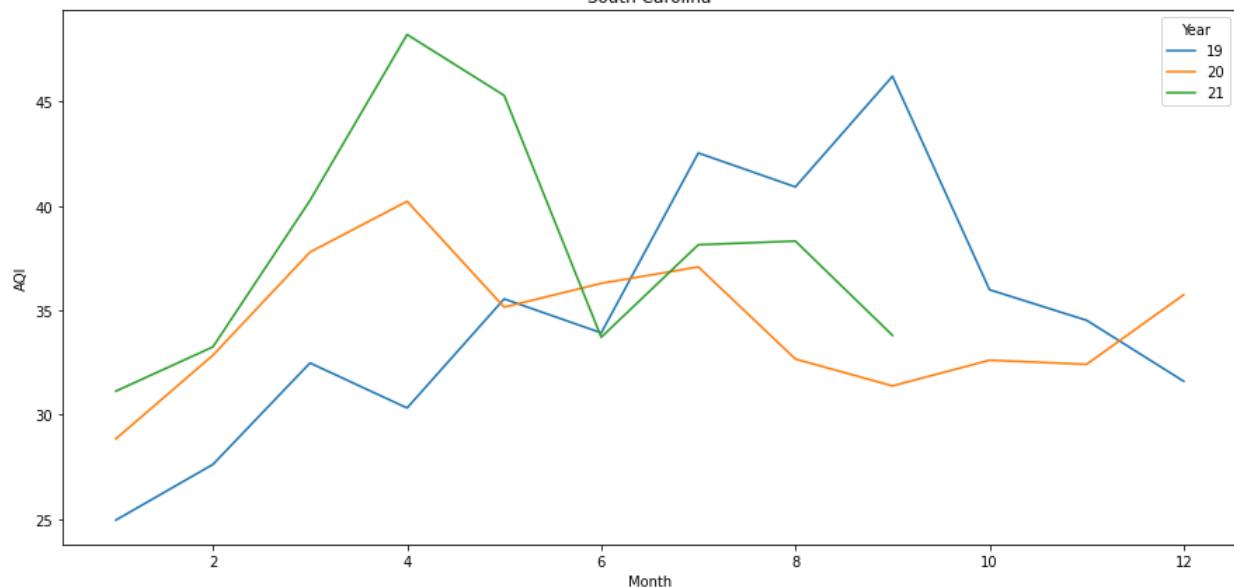


## TASK1(AQI)

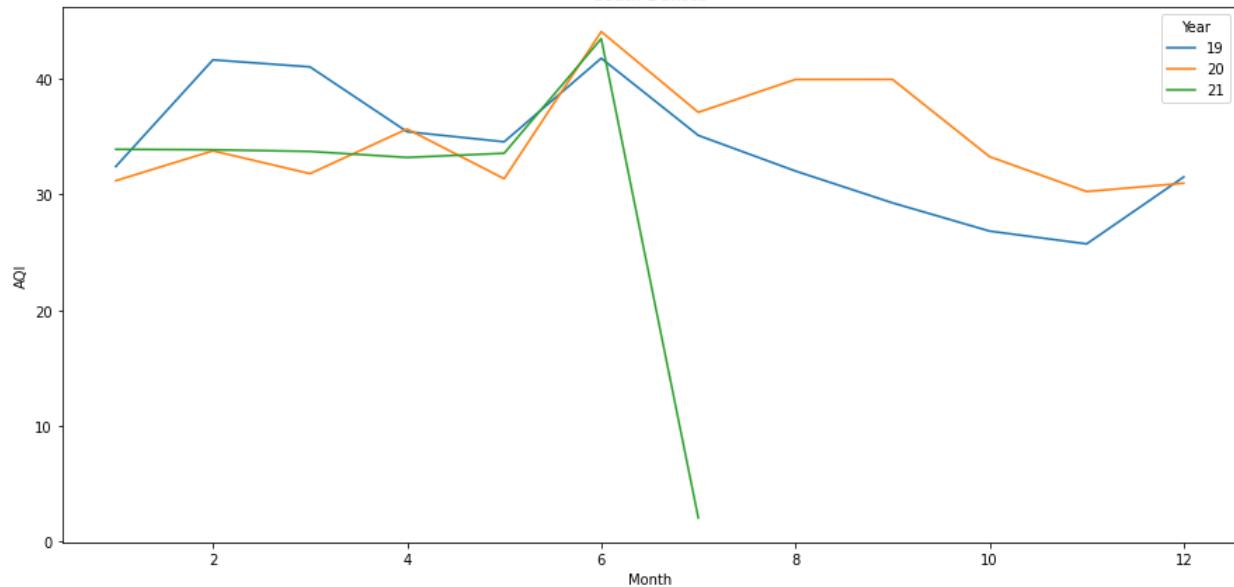
Rhode Island



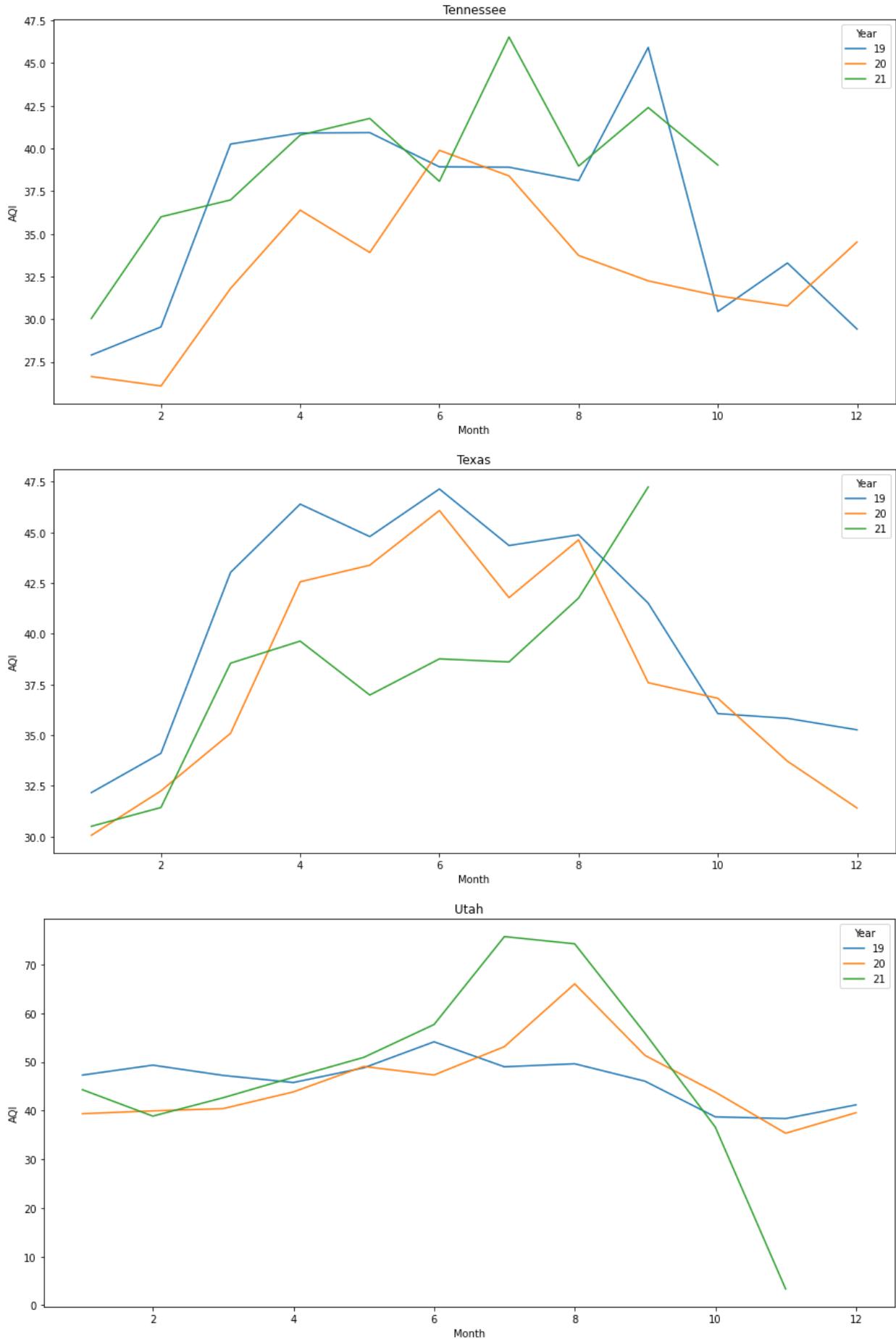
South Carolina



South Dakota

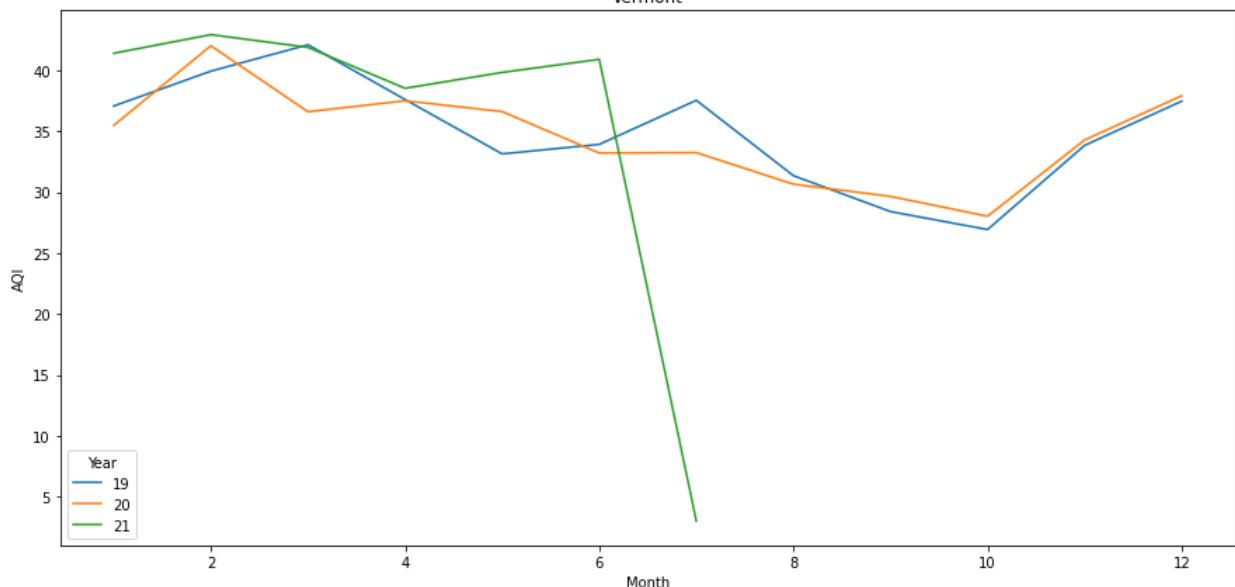


## TASK1(AQI)

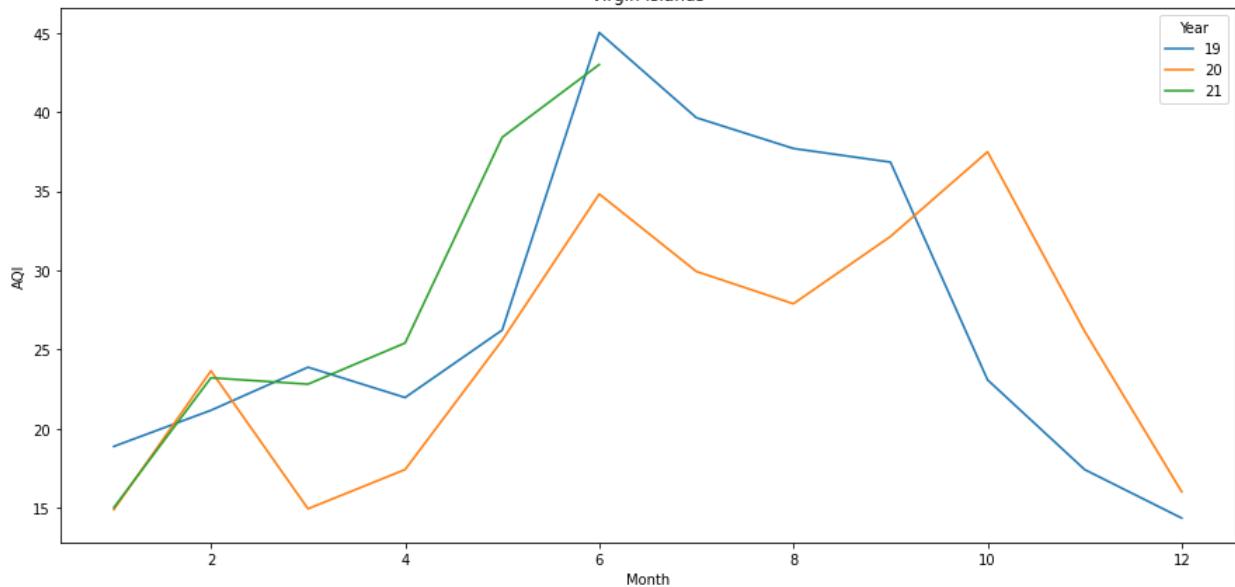


## TASK1(AQI)

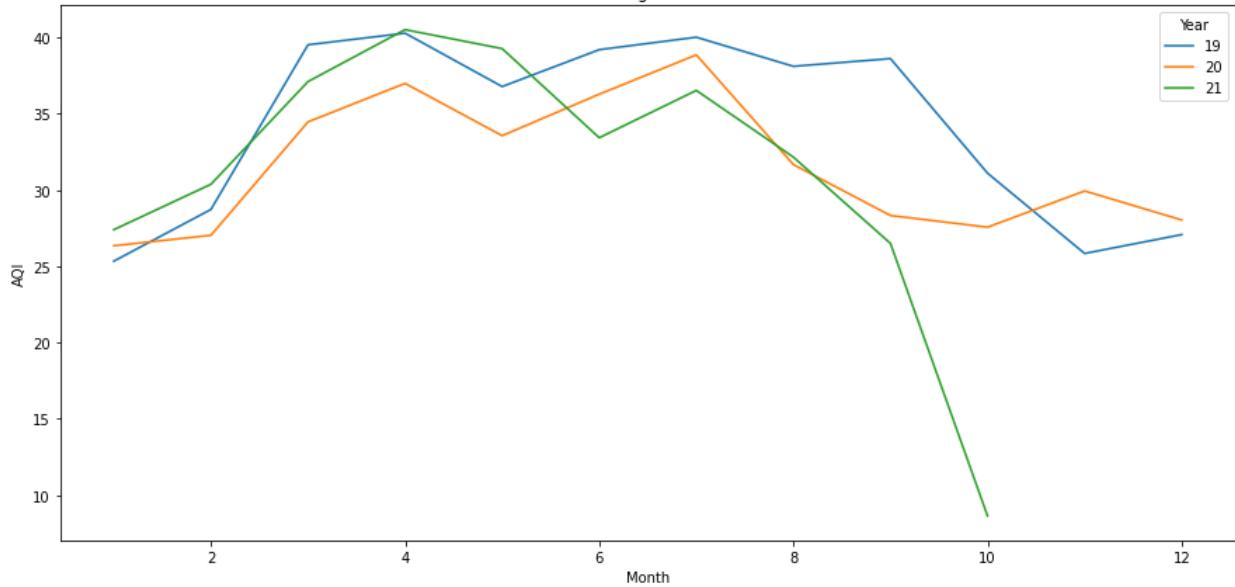
Vermont



Virgin Islands

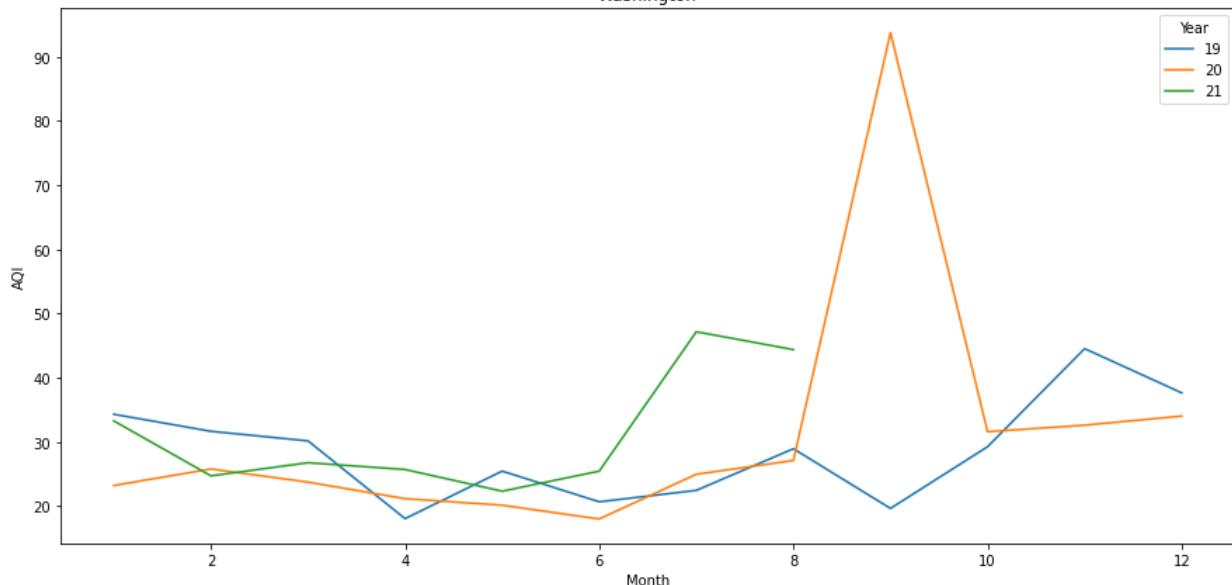


Virginia

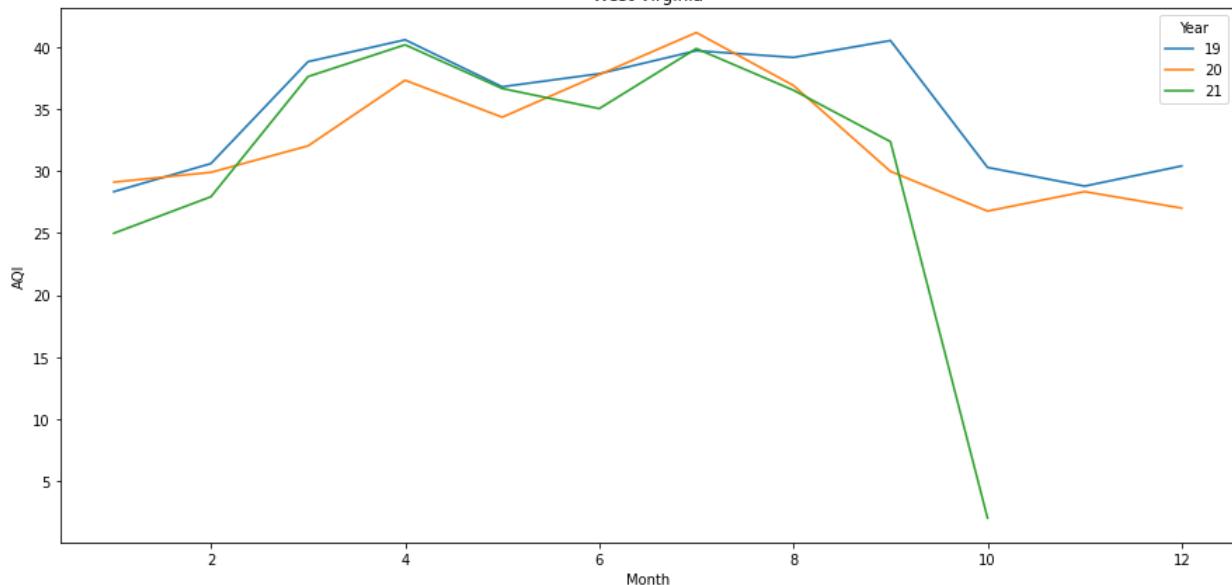


## TASK1(AQI)

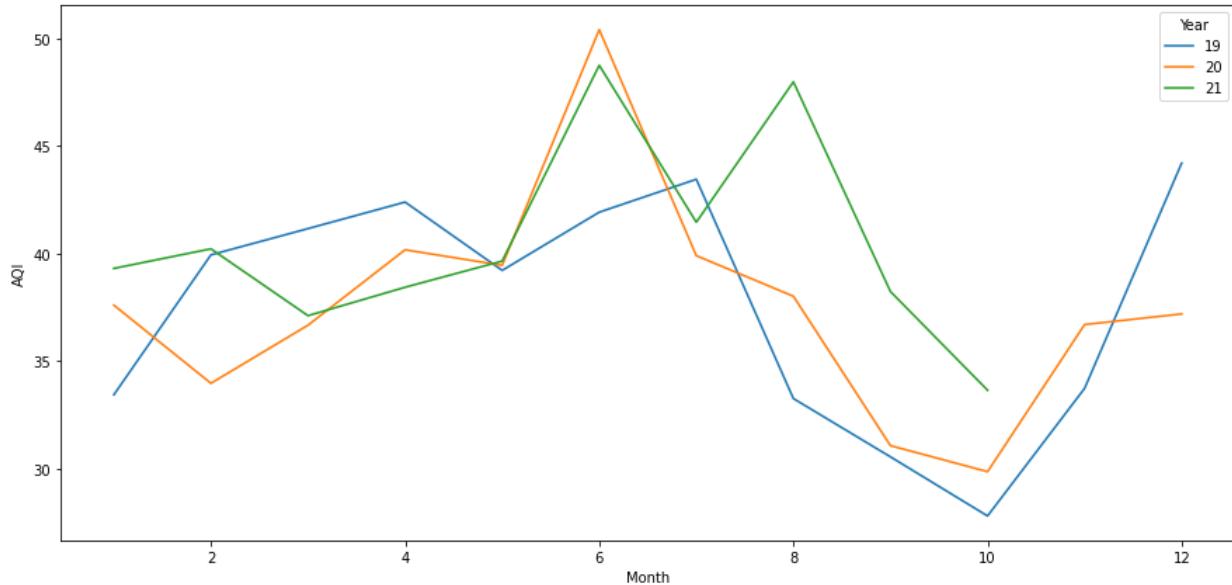
Washington

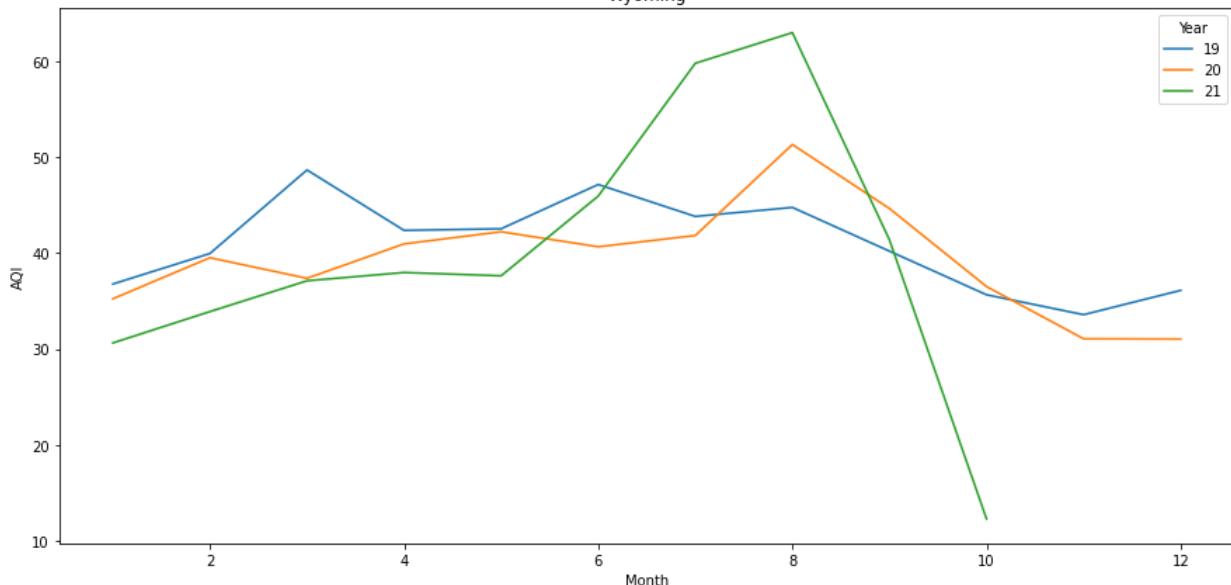


West Virginia



Wisconsin





## Did the lockdown result in a noticeable improvement in air quality? Most notably in what regions?

Ans:

After analyzing the AQI data for all the states of the United States, it was found that the states: {Alaska}, {Mexico}, {District of Columbia}, {Hawaii}, {Idaho}, {Illinois}, {Indiana}, {Iowa}, {Michigan}, {Minnesota}, {Montana}, {Nebraska}, {Puerto Rico}, {Tennessee} had low AQI levels during the lockdown period of February-2020 - April-2020. However, after the lockdown period, there was a sudden increase in AQI levels in all the listed states. This suggests that the lockdown measures implemented in the state had a positive impact on reducing air pollution during the lockdown period, but that the levels returned to normal or increased after the lockdown period.

## {Finall Summary}

Metro States: {Alabama, Arizona, Arkansas, California, Colorado, Connecticut, Delaware, Florida, Georgia, Kansas, Kentucky, Louisiana, Maine, Maryland, Massachusetts, Mississippi, Missouri, Nevada, New Hampshire, New Jersey, New Mexico, New York, North Carolina, North Dakota, Ohio, Oklahoma, Oregon, Pennsylvania, Rhode Island, South Carolina, Texas, Utah, Vermont, Virginia, Washington, Wyoming, West Virginia}

The AQI levels in metropolitan states and big county were found to be relatively same as well as minute low compared to previous years, but not as low as in other states during the lockdown period. This is likely due to the high population density and number of industries present in these metropolitan areas, which contribute to ongoing air pollution even during a lockdown period. Despite this, it is still notable that the AQI levels were lower than in previous years, indicating that some measures to reduce air pollution may have been

successful in these cities. Overall, while more work needs to be done to further reduce air pollution in metropolitan areas, the data suggests that progress is being made.

# {COVID-19 Case} Surveillance Public Use Data with Geography and {Air Quality Index (AQI) }

```
In [1]: #importing libraries
import pandas as pd
import numpy as np
```

## Checking the Dataframe

```
In [2]: #Reading the COVID dataset, Also after reading the file and seeing its properties
covid_df = pd.read_csv('/Users/vishesh/Downloads/DA/cases.csv')

/var/folders/nr/x5htffh97tvfw1rt32pzkmco0000gn/T/ipykernel_51259/4244949551.py:3: DtypeWarning: Columns (3) have mixed types. Specify dtype option on import or set low_memory=False.
    covid_df = pd.read_csv('/Users/vishesh/Downloads/DA/cases.csv')
```

```
In [3]: # checking all the columns name and understading the column data
covid_df.columns
```

```
Out[3]: Index(['case_month', 'res_state', 'state_fips_code', 'res_county',
       'county_fips_code', 'age_group', 'sex', 'race', 'ethnicity',
       'case_positive_specimen_interval', 'case_onset_interval', 'process',
       'exposure_yn', 'current_status', 'symptom_status', 'hosp_yn', 'icu_yn',
       'death_yn', 'underlying_conditions_yn'],
      dtype='object')
```

```
In [4]: # Top 5 Rows of Covid_Dataframe
covid_df.head()
```

|   | case_month | res_state | state_fips_code | res_county | county_fips_code | age_group    | sex    |
|---|------------|-----------|-----------------|------------|------------------|--------------|--------|
| 0 | 2020-12    | AR        | 5.0             | BOONE      | 5009.0           | 0 - 17 years | NaN    |
| 1 | 2021-10    | MO        | 29.0            | PHELPS     | 29161.0          | 0 - 17 years | NaN    |
| 2 | 2020-10    | MO        | 29.0            | CASS       | 29037.0          | 0 - 17 years | Female |
| 3 | 2021-08    | TN        | 47.0            | CLAIBORNE  | 47025.0          | 0 - 17 years | Female |
| 4 | 2020-11    | GA        | 13.0            | EFFINGHAM  | 13103.0          | 0 - 17 years | Female |

## { Data Cleaning and Wrangling }

```
In [5]: # Summary of the Covid DataFrame, including the number of non-null values in each column
```

```
covid_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 37532072 entries, 0 to 37532071
Data columns (total 19 columns):
 #   Column           Dtype  
 --- 
 0   case_month       object  
 1   res_state        object  
 2   state_fips_code float64 
 3   res_county       object  
 4   county_fips_code float64 
 5   age_group        object  
 6   sex              object  
 7   race             object  
 8   ethnicity        object  
 9   case_positive_specimen_interval float64 
 10  case_onset_interval  float64 
 11  process          object  
 12  exposure_yn     object  
 13  current_status   object  
 14  symptom_status   object  
 15  hosp_yn          object  
 16  icu_yn           object  
 17  death_yn         object  
 18  underlying_conditions_yn object  
dtypes: float64(4), object(15)
memory usage: 5.3+ GB
```

In [6]: *## Check for missing values in each column of the covid\_df DataFrame and return the count of missing values for each column.*

```
## Check for missing values in each column of the covid_df DataFrame and return the count of missing values for each column.
covid_df.isna().sum()
```

Out[6]:

|                                 |          |
|---------------------------------|----------|
| case_month                      | 12       |
| res_state                       | 920      |
| state_fips_code                 | 920      |
| res_county                      | 2561284  |
| county_fips_code                | 2561284  |
| age_group                       | 406634   |
| sex                             | 1244276  |
| race                            | 6597416  |
| ethnicity                       | 7837672  |
| case_positive_specimen_interval | 25113195 |
| case_onset_interval             | 20221411 |
| process                         | 0        |
| exposure_yn                     | 0        |
| current_status                  | 0        |
| symptom_status                  | 0        |
| hosp_yn                         | 0        |
| icu_yn                          | 0        |
| death_yn                        | 1752914  |
| underlying_conditions_yn        | 35206303 |
| dtype: int64                    |          |

## Dealing with All the Missing Values now

In [7]: `covid_df.iloc[1300:1320]`

| Out[7]: | case_month | res_state | state_fips_code | res_county     | county_fips_code | age_grou     |
|---------|------------|-----------|-----------------|----------------|------------------|--------------|
| 1300    | 2020-05    | MD        | 24.0            | CAROLINE       | 24011.0          | 18 to 4 year |
| 1301    | 2021-08    | AL        | 1.0             | HOUSTON        | 13153.0          | Missin       |
| 1302    | 2021-03    | NE        | 31.0            | DOUGLAS        | 20045.0          | 18 to 4 year |
| 1303    | 2020-06    | VA        | 51.0            | BUCHANAN       | 51027.0          | Nal          |
| 1304    | 2020-10    | KY        | 21.0            | LINCOLN        | 21137.0          | 0 - 1 year   |
| 1305    | 2020-12    | MI        | 26.0            | HURON          | 26063.0          | 18 to 4 year |
| 1306    | 2020-06    | UT        | 49.0            | SANPETE        | 49039.0          | 18 to 4 year |
| 1307    | 2021-10    | OK        | 40.0            | BECKHAM        | 40009.0          | 0 - 1 year   |
| 1308    | 2021-03    | CO        | 8.0             | BROOMFIELD     | 8014.0           | 0 - 1 year   |
| 1309    | 2021-08    | AR        | 5.0             | CRITTENDEN     | 5035.0           | 0 - 1 year   |
| 1310    | 2021-03    | WI        | 55.0            | DOUGLAS        | 55031.0          | 0 - 1 year   |
| 1311    | 2021-09    | MO        | 29.0            | DUNKLIN        | 29069.0          | 0 - 1 year   |
| 1312    | 2021-07    | ID        | 16.0            | ELMORE         | 16039.0          | 0 - 1 year   |
| 1313    | 2020-12    | OH        | 39.0            | FAYETTE        | 39047.0          | 0 - 1 year   |
| 1314    | 2021-10    | KS        | 20.0            | MONTGOMERY     | 20125.0          | 0 - 1 year   |
| 1315    | 2021-02    | VA        | 51.0            | NEW KENT       | 51127.0          | 0 - 1 year   |
| 1316    | 2020-12    | PA        | 42.0            | NORTHUMBERLAND | 42097.0          | 0 - 1 year   |
| 1317    | 2021-09    | MT        | 30.0            | SILVER BOW     | 30093.0          | 0 - 1 year   |
| 1318    | 2021-08    | NY        | 36.0            | ST. LAWRENCE   | 36089.0          | 0 - 1 year   |
| 1319    | 2020-09    | OH        | 39.0            | AUGLAIZE       | 39011.0          | 18 to 4 year |

After Analyzing the above cell: I observed that there are a large number of missing values in columns {Sex}, {Age\_Group}. I also recognized that simply dropping these

rows with missing values would result in a biased and inaccurate dataset. Instead, I will be using a different approach, which is utilizing the backward fill method to fill in the missing values in the 'SEX' column. This approach is chosen because it uses the previous value to fill the missing values because there is a pattern of missing values in the Sex and Age\_group columns, thus it will be more accurate and will not make the dataset biased. By doing this I am trying to improve the results of their analysis by ensuring that the 'SEX' column has a complete and accurate set of data.

```
In [8]: # filling the missing values with Backword fill aproach  
covid_df['sex'].fillna(method='bfill', inplace=True)
```

```
In [9]: # checking weather the records are filled or not  
covid_df.iloc[1300:1320]
```

| Out[9]: | case_month | res_state | state_fips_code | res_county     | county_fips_code | age_grou     |
|---------|------------|-----------|-----------------|----------------|------------------|--------------|
| 1300    | 2020-05    | MD        | 24.0            | CAROLINE       | 24011.0          | 18 to 4 year |
| 1301    | 2021-08    | AL        | 1.0             | HOUSTON        | 13153.0          | Missin       |
| 1302    | 2021-03    | NE        | 31.0            | DOUGLAS        | 20045.0          | 18 to 4 year |
| 1303    | 2020-06    | VA        | 51.0            | BUCHANAN       | 51027.0          | Nal          |
| 1304    | 2020-10    | KY        | 21.0            | LINCOLN        | 21137.0          | 0 - 1 year   |
| 1305    | 2020-12    | MI        | 26.0            | HURON          | 26063.0          | 18 to 4 year |
| 1306    | 2020-06    | UT        | 49.0            | SANPETE        | 49039.0          | 18 to 4 year |
| 1307    | 2021-10    | OK        | 40.0            | BECKHAM        | 40009.0          | 0 - 1 year   |
| 1308    | 2021-03    | CO        | 8.0             | BROOMFIELD     | 8014.0           | 0 - 1 year   |
| 1309    | 2021-08    | AR        | 5.0             | CRITTENDEN     | 5035.0           | 0 - 1 year   |
| 1310    | 2021-03    | WI        | 55.0            | DOUGLAS        | 55031.0          | 0 - 1 year   |
| 1311    | 2021-09    | MO        | 29.0            | DUNKLIN        | 29069.0          | 0 - 1 year   |
| 1312    | 2021-07    | ID        | 16.0            | ELMORE         | 16039.0          | 0 - 1 year   |
| 1313    | 2020-12    | OH        | 39.0            | FAYETTE        | 39047.0          | 0 - 1 year   |
| 1314    | 2021-10    | KS        | 20.0            | MONTGOMERY     | 20125.0          | 0 - 1 year   |
| 1315    | 2021-02    | VA        | 51.0            | NEW KENT       | 51127.0          | 0 - 1 year   |
| 1316    | 2020-12    | PA        | 42.0            | NORTHUMBERLAND | 42097.0          | 0 - 1 year   |
| 1317    | 2021-09    | MT        | 30.0            | SILVER BOW     | 30093.0          | 0 - 1 year   |
| 1318    | 2021-08    | NY        | 36.0            | ST. LAWRENCE   | 36089.0          | 0 - 1 year   |
| 1319    | 2020-09    | OH        | 39.0            | AUGLAIZE       | 39011.0          | 18 to 4 year |

In [10]: `# filling the missing values with Backword fill aproach  
covid_df['age_group'].fillna(method='bfill', inplace=True)`

In [11]: covid\_df.iloc[1300:1320]

|      | case_month | res_state | state_fips_code | res_county     | county_fips_code | age_grou     |
|------|------------|-----------|-----------------|----------------|------------------|--------------|
| 1300 | 2020-05    | MD        | 24.0            | CAROLINE       | 24011.0          | 18 to 4 year |
| 1301 | 2021-08    | AL        | 1.0             | HOUSTON        | 13153.0          | Missin       |
| 1302 | 2021-03    | NE        | 31.0            | DOUGLAS        | 20045.0          | 18 to 4 year |
| 1303 | 2020-06    | VA        | 51.0            | BUCHANAN       | 51027.0          | 0 - 1 year   |
| 1304 | 2020-10    | KY        | 21.0            | LINCOLN        | 21137.0          | 0 - 1 year   |
| 1305 | 2020-12    | MI        | 26.0            | HURON          | 26063.0          | 18 to 4 year |
| 1306 | 2020-06    | UT        | 49.0            | SANPETE        | 49039.0          | 18 to 4 year |
| 1307 | 2021-10    | OK        | 40.0            | BECKHAM        | 40009.0          | 0 - 1 year   |
| 1308 | 2021-03    | CO        | 8.0             | BROOMFIELD     | 8014.0           | 0 - 1 year   |
| 1309 | 2021-08    | AR        | 5.0             | CRITTENDEN     | 5035.0           | 0 - 1 year   |
| 1310 | 2021-03    | WI        | 55.0            | DOUGLAS        | 55031.0          | 0 - 1 year   |
| 1311 | 2021-09    | MO        | 29.0            | DUNKLIN        | 29069.0          | 0 - 1 year   |
| 1312 | 2021-07    | ID        | 16.0            | ELMORE         | 16039.0          | 0 - 1 year   |
| 1313 | 2020-12    | OH        | 39.0            | FAYETTE        | 39047.0          | 0 - 1 year   |
| 1314 | 2021-10    | KS        | 20.0            | MONTGOMERY     | 20125.0          | 0 - 1 year   |
| 1315 | 2021-02    | VA        | 51.0            | NEW KENT       | 51127.0          | 0 - 1 year   |
| 1316 | 2020-12    | PA        | 42.0            | NORTHUMBERLAND | 42097.0          | 0 - 1 year   |
| 1317 | 2021-09    | MT        | 30.0            | SILVER BOW     | 30093.0          | 0 - 1 year   |
| 1318 | 2021-08    | NY        | 36.0            | ST. LAWRENCE   | 36089.0          | 0 - 1 year   |
| 1319 | 2020-09    | OH        | 39.0            | AUGLAIZE       | 39011.0          | 18 to 4 year |

After analyzing a random sample of rows and checking the 'Race', 'Ethnicity', 'Underling\_Conditions\_yn', and 'death\_yn' columns of the DataFrame, it was determined that there were no patterns of missing values in those columns. Dropping rows with missing values in these columns would result in a biased and inaccurate analysis. As the dataframe does not have information about the race and ethnicity of the missing values, I choose an alternative approach to fill the missing values with the word 'Unknown'. This approach will provide better results as it will not compromise the integrity of the data and will not result in a biased analysis. This way I can have a more accurate understanding of the data and the missing values will not affect the analysis results.

```
In [12]: # filling the missing values with "Unknown" word
covid_df['race'].fillna(value = 'Unknown', inplace = True)
```

```
In [13]: # checking weather the records are filled or not
covid_df['race']
```

```
Out[13]: 0           Unknown
1           Unknown
2           Unknown
3           Missing
4           Unknown
...
37532067    White
37532068    White
37532069    White
37532070    White
37532071    White
Name: race, Length: 37532072, dtype: object
```

```
In [14]: # filling the missing values with "Unknown" word
covid_df['ethnicity'].fillna(value = 'Unknown', inplace = True)
```

```
In [15]: # checking weather the records are filled or not
covid_df['ethnicity'].head(50)
```

```
Out[15]: 0      Unknown
          1      Unknown
          2      Unknown
          3      Missing
          4      Missing
          5      Unknown
          6      Unknown
          7      Missing
          8      Missing
          9      Missing
         10     Missing
         11     Missing
         12     Missing
         13     Unknown
         14     Unknown
         15     Unknown
         16     Unknown
         17     Unknown
         18     Missing
         19     Missing
         20     Missing
         21     Unknown
         22     Unknown
         23     Unknown
         24     Unknown
         25     Missing
         26     Unknown
         27     Missing
         28     Unknown
         29     Unknown
         30    Hispanic/Latino
         31    Hispanic/Latino
         32    Hispanic/Latino
         33    Hispanic/Latino
         34    Hispanic/Latino
         35    Hispanic/Latino
         36    Hispanic/Latino
         37    Hispanic/Latino
         38    Hispanic/Latino
         39    Hispanic/Latino
         40    Hispanic/Latino
         41    Hispanic/Latino
         42    Hispanic/Latino
         43    Hispanic/Latino
         44    Hispanic/Latino
         45    Hispanic/Latino
         46    Hispanic/Latino
         47    Hispanic/Latino
         48    Hispanic/Latino
         49    Non-Hispanic/Latino
Name: ethnicity, dtype: object
```

```
In [16]: covid_df['underlying_conditions_yn'].fillna(value = 'Unknown', inplace = True)
```

```
In [17]: covid_df['underlying_conditions_yn'].head(50)
```

```
Out[17]: 0    Unknown  
1    Unknown  
2    Unknown  
3    Unknown  
4    Unknown  
5    Unknown  
6    Unknown  
7    Unknown  
8    Unknown  
9    Unknown  
10   Unknown  
11   Unknown  
12   Unknown  
13   Unknown  
14   Unknown  
15   Unknown  
16   Unknown  
17   Unknown  
18   Unknown  
19   Unknown  
20   Unknown  
21   Unknown  
22   Unknown  
23   Unknown  
24   Unknown  
25   Unknown  
26   Unknown  
27   Unknown  
28   Unknown  
29   Unknown  
30   Unknown  
31   Unknown  
32   Unknown  
33   Unknown  
34   Unknown  
35   Unknown  
36   Unknown  
37   Unknown  
38   Unknown  
39   Unknown  
40   Unknown  
41   Unknown  
42   Unknown  
43   Unknown  
44   Unknown  
45   Unknown  
46   Unknown  
47   Unknown  
48      Yes  
49   Unknown  
Name: underlying_conditions_yn, dtype: object
```

```
In [18]: covid_df['death_yn'].fillna(value = 'Unknown', inplace = True)
```

```
In [19]: covid_df['death_yn'].head(50)
```

```
Out[19]: 0    Unknown
          1    Unknown
          2    Unknown
          3    Missing
          4    Missing
          5        No
          6        No
          7        No
          8    Missing
          9    Missing
         10   Missing
         11   Missing
         12   Missing
         13   Unknown
         14   Unknown
         15   Unknown
         16   Missing
         17   Unknown
         18   Missing
         19   Missing
         20   Missing
         21   Unknown
         22   Unknown
         23   Unknown
         24        No
         25   Missing
         26        No
         27   Missing
         28   Unknown
         29   Unknown
         30   Missing
         31   Missing
         32   Missing
         33        No
         34   Missing
         35   Unknown
         36   Missing
         37   Missing
         38   Unknown
         39        No
         40        No
         41   Unknown
         42        No
         43        No
         44        No
         45        No
         46        No
         47        No
         48   Unknown
         49   Missing
```

Name: death\_yn, dtype: object

After analyzing the 'case\_positive\_specimen\_interval','case\_onset\_interval' columns of the DataFrame and observing that it has a relatively numerical values. I recognized that dropping the rows with missing values in this column will significantly impact the overall dataset. Therefore, I decided to take the mean of the whole column. This approach is chosen because the number of missing values in this column is large and randomly generated, so mean will surely affect the overall analysis.

```
In [20]: # Using imputation techniques to fill the missing values with mean of the column

from sklearn.impute import SimpleImputer
imputer = SimpleImputer(missing_values=np.nan, strategy='mean')

# Fitting the imputer on the data
imputer.fit(covid_df[['case_positive_specimen_interval','case_onset_interval']])

# replacing missing values with the mean value
covid_df[['case_positive_specimen_interval','case_onset_interval']] = imputer.t
```

```
In [21]: # checking both case_positive_specimen_interval and case_onset_interval records
covid_df[['case_positive_specimen_interval','case_onset_interval']].head(50)
```

Out[21]:

|    | case_positive_specimen_interval | case_onset_interval |
|----|---------------------------------|---------------------|
| 0  | 0.000000                        | 0.099948            |
| 1  | 0.000000                        | 0.099948            |
| 2  | 0.000000                        | 0.000000            |
| 3  | 0.000000                        | 0.099948            |
| 4  | 0.200811                        | 0.000000            |
| 5  | 0.200811                        | 0.000000            |
| 6  | 0.200811                        | 0.099948            |
| 7  | 0.000000                        | 0.099948            |
| 8  | 0.200811                        | 0.000000            |
| 9  | 0.200811                        | 0.099948            |
| 10 | 0.200811                        | 0.099948            |
| 11 | 0.200811                        | 0.099948            |
| 12 | 0.200811                        | 0.000000            |
| 13 | 0.000000                        | 0.000000            |
| 14 | 0.200811                        | 0.000000            |
| 15 | 0.200811                        | 0.000000            |
| 16 | 0.200811                        | 0.000000            |
| 17 | 0.200811                        | 0.099948            |
| 18 | 0.000000                        | 0.099948            |
| 19 | 0.200811                        | 0.099948            |
| 20 | 0.200811                        | 0.000000            |
| 21 | 0.200811                        | 0.099948            |
| 22 | 0.200811                        | 0.099948            |
| 23 | 0.000000                        | 0.000000            |
| 24 | 0.200811                        | 0.000000            |
| 25 | 0.200811                        | 0.000000            |
| 26 | 0.200811                        | 0.099948            |
| 27 | 0.200811                        | 0.099948            |
| 28 | 0.000000                        | 0.099948            |
| 29 | 0.200811                        | 0.000000            |
| 30 | 0.200811                        | 0.099948            |
| 31 | 0.200811                        | 0.099948            |
| 32 | 0.200811                        | 0.099948            |
| 33 | 0.200811                        | 0.099948            |
| 34 | 0.200811                        | 0.099948            |

|    | case_positive_specimen_interval | case_onset_interval |
|----|---------------------------------|---------------------|
| 35 | 0.200811                        | 0.099948            |
| 36 | 0.200811                        | 0.099948            |
| 37 | 0.200811                        | 0.000000            |
| 38 | 0.200811                        | 0.099948            |
| 39 | 0.200811                        | 0.000000            |
| 40 | 0.000000                        | 0.099948            |
| 41 | 0.200811                        | 0.099948            |
| 42 | 0.200811                        | 0.099948            |
| 43 | 0.000000                        | 0.099948            |
| 44 | 0.200811                        | 0.099948            |
| 45 | 0.200811                        | 0.000000            |
| 46 | 0.200811                        | 0.000000            |
| 47 | 0.000000                        | 0.000000            |
| 48 | 0.000000                        | 0.000000            |
| 49 | 0.200811                        | 0.000000            |

After analyzing the 'res\_state' column of the DataFrame and observing that it has a relatively low number of missing values compared to other columns. I recognized that dropping the rows with missing values in this column will not significantly impact the overall dataset. Therefore, I decided to drop the rows with missing values in the 'res\_state' column in order to maintain the integrity and accuracy of the overall dataset. This approach is chosen because the number of missing values in this column is not large enough to affect the overall analysis and by dropping the missing values the data will be more accurate.

In [22]: `# Dropping the Column 'res_county' and 'County_fips_code' based on the missing covid_df.dropna(subset = ['res_state', 'county_fips_code'], inplace = True)`

In [23]: `covid_df.isna().sum()`

```
Out[23]: case_month          0
res_state           0
state_fips_code    0
res_county          0
county_fips_code   0
age_group           0
sex                 0
race                0
ethnicity           0
case_positive_specimen_interval 0
case_onset_interval 0
process              0
exposure_yn         0
current_status      0
symptom_status      0
hosp_yn              0
icu_yn               0
death_yn             0
underlying_conditions_yn 0
dtype: int64
```

All the missing values are done with processing now !!

---

## Dealing with Relevant Columns now

```
In [24]: covid_df.head()
```

|   | case_month | res_state | state_fips_code | res_county | county_fips_code | age_group    | sex    |
|---|------------|-----------|-----------------|------------|------------------|--------------|--------|
| 0 | 2020-12    | AR        | 5.0             | BOONE      | 5009.0           | 0 - 17 years | Female |
| 1 | 2021-10    | MO        | 29.0            | PHELPS     | 29161.0          | 0 - 17 years | Female |
| 2 | 2020-10    | MO        | 29.0            | CASS       | 29037.0          | 0 - 17 years | Female |
| 3 | 2021-08    | TN        | 47.0            | CLAIBORNE  | 47025.0          | 0 - 17 years | Female |
| 4 | 2020-11    | GA        | 13.0            | EFFINGHAM  | 13103.0          | 0 - 17 years | Female |

```
In [25]: # Checking the datatype of columns
covid_df.dtypes
```

```
Out[25]: case_month          object
res_state            object
state_fips_code      float64
res_county           object
county_fips_code     float64
age_group            object
sex                  object
race                 object
ethnicity            object
case_positive_specimen_interval float64
case_onset_interval   float64
process              object
exposure_yn          object
current_status       object
symptom_status       object
hosp_yn              object
icu_yn               object
death_yn              object
underlying_conditions_yn object
dtype: object
```

```
In [26]: ## Changing the datatype of column state_fips_code from float to 'Int'.
covid_df['state_fips_code'] = covid_df['state_fips_code'].astype(int)
```

I am renaming the 'state\_fips\_code' column of the current dataset to 'State Code'. The reason for this is that I noticed that there is a similar column with the same name in the dataset they were working on in task 1, which is the AQI dataset. By renaming the column, I am trying to make it easy to merge both the datasets on this column. This will enable them to join the data from both datasets and analyze it together.

```
In [27]: # Renaming the column from state_fips_code' to 'State Code'
covid_df = covid_df.rename(columns = {'state_fips_code':'State Code'})
```

## Taking relevant columns for further Analysis and dropping the rest of the unnecessary columns for this Task-2

```
In [28]: # Dropping Columns from dataframe

covid_df = covid_df.drop(columns = ['res_state', 'res_county', 'county_fips_code',
                                     'case_positive_specimen_interval', 'case_onset_interval', 'process',
                                     'exposure_yn', 'current_status', 'symptom_status'])
```

```
In [29]: #Checking the dataframe
covid_df.head()
```

```
Out[29]: case_month  State Code  hosp_yn  icu_yn  death_yn  underlying_conditions_yn
0    2020-12        5  Missing  Missing  Unknown      Unknown
1    2021-10       29  Unknown  Missing  Unknown      Unknown
2    2020-10       29  Unknown  Missing  Unknown      Unknown
3    2021-08       47  Missing  Missing  Missing      Unknown
4    2020-11       13  Missing  Missing  Missing      Unknown
```

```
In [30]: # Checking the dtypes
covid_df.dtypes
```

```
Out[30]: case_month          object
State_Code           int64
hosp_yn              object
icu_yn               object
death_yn              object
underlying_conditions_yn   object
dtype: object
```

```
In [31]: # Checking the size of dataframe
covid_df.shape
```

```
Out[31]: (34970788, 6)
```

```
In [32]: # converting the 'case_month' column of covid DataFrame into datetime datatype.
covid_df['case_month'] = pd.to_datetime(covid_df['case_month'])
covid_df['case_month'].head()
```

```
Out[32]: 0    2020-12-01
1    2021-10-01
2    2020-10-01
3    2021-08-01
4    2020-11-01
Name: case_month, dtype: datetime64[ns]
```

```
In [33]: ## Changing the datatype of column state_fips_code from float to 'Int'.
covid_df['State Code'] = covid_df['State Code'].astype(int)
```

```
In [34]: covid_df.dtypes
```

```
Out[34]: case_month          datetime64[ns]
State Code           int64
hosp_yn              object
icu_yn               object
death_yn              object
underlying_conditions_yn   object
dtype: object
```

```
In [35]: # filtering data for further analysis based on AQI_Datasets of year 2020 and 2021
filtered_df = covid_df[covid_df['case_month'].dt.year.isin([2020, 2021])]
```

```
In [36]: filtered_df.shape
```

```
Out[36]: (34970788, 6)
```

I identified that in order to calculate correlation and other factors related to the data I am working on. However, I realized that using the categorical values of the data will not be possible. So, I have decided to use an approach of converting the categorical values to numerical values. This will enable me to calculate factors such as hospital rate, death\_yn, and underlying conditions based on the numerical values. This approach was chosen because numerical values can be used for mathematical calculations, making it possible to calculate correlation rates and other factors.

**Additionally, having numerical values instead of categorical values will make it easier to perform good analysis.**

```
In [37]: # Converting the hosp_yn column categorical values to numerical  
filtered_df['hosp_yn'].replace({'Yes':1, 'No':0, 'Unknown':0, 'Missing':0}, inplace=True)  
  
In [38]: # Converting the death_yn column categorical values to numerical  
filtered_df['death_yn'].replace({'Yes':1, 'No':0, 'Unknown':0, 'Missing':0}, inplace=True)  
  
In [39]: # Converting the icu_yn column categorical values to numerical  
filtered_df['icu_yn'].replace({'Yes':1, 'No':0, 'Unknown':0, 'Missing':0}, inplace=True)  
  
In [40]: # Converting the underlying_conditions_yn column categorical values to numerical  
filtered_df['underlying_conditions_yn'].replace({'Yes':1, 'No':0, 'Unknown':0}, inplace=True)  
  
In [41]: # checking the size of covid dataframe  
filtered_df.shape  
  
Out[41]: (34970788, 6)  
  
In [42]: # Grouping the covid DataFrame by 'State_code' and 'case_month' columns and calculating the sum of cases  
df_covid = filtered_df.groupby(['State Code', 'case_month'], as_index=False).sum()
```

**The Dataframe looks good with relevant columns now**

```
In [43]: df_covid.head(25)
```

Out[43]:

|    | State Code | case_month | hosp_yn | death_yn | underlying_conditions_yn |
|----|------------|------------|---------|----------|--------------------------|
| 0  | 1          | 2020-03-01 | 405     | 11       | 22                       |
| 1  | 1          | 2020-04-01 | 673     | 164      | 29                       |
| 2  | 1          | 2020-05-01 | 671     | 74       | 70                       |
| 3  | 1          | 2020-06-01 | 943     | 82       | 163                      |
| 4  | 1          | 2020-07-01 | 1339    | 309      | 126                      |
| 5  | 1          | 2020-08-01 | 1079    | 179      | 116                      |
| 6  | 1          | 2020-09-01 | 700     | 0        | 84                       |
| 7  | 1          | 2020-10-01 | 744     | 24       | 104                      |
| 8  | 1          | 2020-11-01 | 673     | 234      | 113                      |
| 9  | 1          | 2020-12-01 | 985     | 855      | 148                      |
| 10 | 1          | 2021-01-01 | 1001    | 675      | 147                      |
| 11 | 1          | 2021-02-01 | 664     | 23       | 100                      |
| 12 | 1          | 2021-03-01 | 445     | 0        | 76                       |
| 13 | 1          | 2021-04-01 | 418     | 0        | 83                       |
| 14 | 1          | 2021-05-01 | 347     | 0        | 58                       |
| 15 | 1          | 2021-06-01 | 240     | 0        | 45                       |
| 16 | 1          | 2021-07-01 | 904     | 129      | 180                      |
| 17 | 1          | 2021-08-01 | 1996    | 555      | 480                      |
| 18 | 1          | 2021-09-01 | 1392    | 134      | 337                      |
| 19 | 1          | 2021-10-01 | 384     | 0        | 98                       |
| 20 | 2          | 2020-03-01 | 26      | 0        | 114                      |
| 21 | 2          | 2020-04-01 | 11      | 0        | 65                       |
| 22 | 2          | 2020-05-01 | 11      | 0        | 77                       |
| 23 | 2          | 2020-06-01 | 28      | 0        | 246                      |
| 24 | 2          | 2020-07-01 | 70      | 0        | 402                      |

## Highly Important Note below !!:

The AQI Concatenated dataset was imported and read separately to avoid kernel crashes due to the large size of the COVID dataset. This allows for more efficient processing and analysis of the data without running out of

**memory. Now, the AQI dataset is being called directly in this notebook to merge with the COVID dataset for further analysis.**

```
In [44]: ## Reading the Concatenated AQI Dataset of the relevant years
AQI_df = pd.read_csv('~/Users/vishesh/Downloads/DA/f_AQI.csv')
```

```
In [45]: AQI_df.head()
```

|   | Unnamed: 0 | State_Name | year_month | State Code | AQI       |
|---|------------|------------|------------|------------|-----------|
| 0 | 0          | Alabama    | 2020-01    | 1.0        | 29.538732 |
| 1 | 1          | Alabama    | 2020-02    | 1.0        | 28.063433 |
| 2 | 2          | Alabama    | 2020-03    | 1.0        | 37.696078 |
| 3 | 3          | Alabama    | 2020-04    | 1.0        | 41.858209 |
| 4 | 4          | Alabama    | 2020-05    | 1.0        | 40.176329 |

```
In [46]: # Dropping the irrelevant columns
AQI_df = AQI_df.drop(columns=['Unnamed: 0'])
```

```
In [47]: df_covid
```

|      | State Code | case_month | hosp_yn | death_yn | underlying_conditions_yn |
|------|------------|------------|---------|----------|--------------------------|
| 0    | 1          | 2020-03-01 | 405     | 11       | 22                       |
| 1    | 1          | 2020-04-01 | 673     | 164      | 29                       |
| 2    | 1          | 2020-05-01 | 671     | 74       | 70                       |
| 3    | 1          | 2020-06-01 | 943     | 82       | 163                      |
| 4    | 1          | 2020-07-01 | 1339    | 309      | 126                      |
| ...  | ...        | ...        | ...     | ...      | ...                      |
| 998  | 56         | 2021-06-01 | 66      | 0        | 645                      |
| 999  | 56         | 2021-07-01 | 104     | 0        | 846                      |
| 1000 | 56         | 2021-08-01 | 165     | 0        | 2141                     |
| 1001 | 56         | 2021-09-01 | 188     | 0        | 2348                     |
| 1002 | 56         | 2021-10-01 | 162     | 0        | 2030                     |

1003 rows × 5 columns

```
In [48]: df_covid.dtypes
```

```
Out[48]: State Code           int64
case_month        datetime64[ns]
hosp_yn          int64
death_yn          int64
underlying_conditions_yn  int64
dtype: object
```

By generating unique columns in both the covid and AQI dataframes, it allows for easy merging of the two datasets by creating a common identifier. This will prevent duplicate records and ensure accurate data analysis. The approach used here is to create a unique ID for each row in both datasets, making it possible to merge them with ease.

```
In [49]: # Converting case_month to string format of '%Y-%m' which represents the year-month
df_covid['case_month'] = df_covid['case_month'].dt.strftime('%Y-%m')
```

```
In [50]: # Generating Unique 'ID' Columns in covid dataframe
df_covid['ID'] = df_covid['State Code'].astype(str)+df_covid['case_month'].astype(str)
```

```
In [51]: # Generating Unique 'ID' Columns in AQI dataframe
AQI_df['ID'] = (AQI_df['State Code'].astype(int)).astype(str)+AQI_df['year_month'].astype(str)
```

```
In [52]: # Checking the ID Column in AQI DataFrame
AQI_df
```

```
Out[52]:
```

|      | State_Name | year_month | State Code | AQI       | ID        |
|------|------------|------------|------------|-----------|-----------|
| 0    | Alabama    | 2020-01    | 1.0        | 29.538732 | 12020-01  |
| 1    | Alabama    | 2020-02    | 1.0        | 28.063433 | 12020-02  |
| 2    | Alabama    | 2020-03    | 1.0        | 37.696078 | 12020-03  |
| 3    | Alabama    | 2020-04    | 1.0        | 41.858209 | 12020-04  |
| 4    | Alabama    | 2020-05    | 1.0        | 40.176329 | 12020-05  |
| ...  | ...        | ...        | ...        | ...       | ...       |
| 1143 | Wyoming    | 2021-06    | 56.0       | 45.946939 | 562021-06 |
| 1144 | Wyoming    | 2021-07    | 56.0       | 59.803191 | 562021-07 |
| 1145 | Wyoming    | 2021-08    | 56.0       | 63.000000 | 562021-08 |
| 1146 | Wyoming    | 2021-09    | 56.0       | 41.371622 | 562021-09 |
| 1147 | Wyoming    | 2021-10    | 56.0       | 12.250000 | 562021-10 |

1148 rows × 5 columns

```
In [53]: # Checking the ID Column in Covid DataFrame
df_covid
```

| Out [53]: | State Code | case_month | hosp_yn | death_yn | underlying_conditions_yn | ID             |
|-----------|------------|------------|---------|----------|--------------------------|----------------|
| 0         | 1          | 2020-03    | 405     | 11       |                          | 22 12020-03    |
| 1         | 1          | 2020-04    | 673     | 164      |                          | 29 12020-04    |
| 2         | 1          | 2020-05    | 671     | 74       |                          | 70 12020-05    |
| 3         | 1          | 2020-06    | 943     | 82       |                          | 163 12020-06   |
| 4         | 1          | 2020-07    | 1339    | 309      |                          | 126 12020-07   |
| ...       | ...        | ...        | ...     | ...      |                          | ...            |
| 998       | 56         | 2021-06    | 66      | 0        |                          | 645 562021-06  |
| 999       | 56         | 2021-07    | 104     | 0        |                          | 846 562021-07  |
| 1000      | 56         | 2021-08    | 165     | 0        |                          | 2141 562021-08 |
| 1001      | 56         | 2021-09    | 188     | 0        |                          | 2348 562021-09 |
| 1002      | 56         | 2021-10    | 162     | 0        |                          | 2030 562021-10 |

1003 rows × 6 columns

## Performing Some Visualizations

### Visualizations over State vs Hospitalization

```
In [54]: fg = AQI_df.groupby(['State_Name'],as_index=False)[['AQI']].mean()

In [55]: # Generating State_Name vs AQI charts

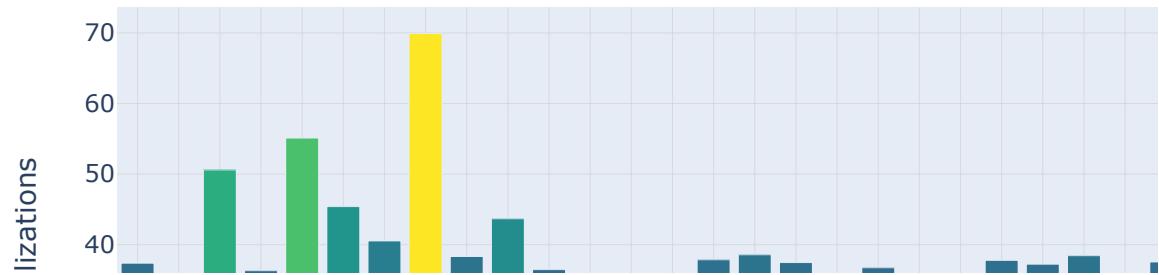
import plotly.express as px

fig = px.bar(fg, x='State_Name', y='AQI', title='Hospitalizations over time',
              color='AQI', color_continuous_scale='Viridis',
              labels={'State_Name':'State', 'AQI':'Hospitalizations'})
fig.update_layout(title={'x':0.5,'text':'Hospitalizations over State'},
                  xaxis_title='State', yaxis_title='Hospitalizations',
                  xaxis=dict(showgrid=True, gridcolor='lightgray', gridwidth=0),
                  yaxis=dict(showgrid=True, gridcolor='lightgray', gridwidth=0),
                  title_font=dict(size=24, family='Arial, sans-serif'),
                  paper_bgcolor='white')

fig.update_layout(
    yaxis=dict(
        tickformat=',d',
        showgrid=True,
        gridcolor='lightgray',
        gridwidth=0.5
    )
)

fig.show()
```

Hospitalizations



In [56]:

```
rg = df_covid.groupby(['State Code'], as_index=False)[['hosp_yn', 'death_yn']].mean()
import plotly.express as px

fig = px.line(rg, x='State Code', y='death_yn', title='State Code over Death')
fig.update_layout(title={'x':0.5, 'text':'State Code over Death'}, template='plotly')
fig.update_layout(
    yaxis=dict(
        tickformat='d'
    )
)

fig.update_layout(
    xaxis=dict(
        tickmode='linear'
    )
)

fig.show()
```

```
/var/folders/nr/x5htffh97tvfw1rt32pkmc0000gn/T/ipykernel_51259/3166128952.py:1: FutureWarning:
```

```
Indexing with multiple keys (implicitly converted to a tuple of keys) will be deprecated, use a list instead.
```



```
In [57]: AQI_df.dtypes
```

```
Out[57]: State_Name      object
year_month       object
State_Code      float64
AQI            float64
ID              object
dtype: object
```

```
In [58]: df_covid.dtypes
```

```
Out[58]: State_Code          int64
case_month        object
hosp_yn           int64
death_yn          int64
underlying_conditions_yn  int64
ID              object
dtype: object
```

## { Merging AQI Dataframe and Covid Dataframe }

```
In [59]: # Merging AQI and Covid dataframe  
merged_df2 = pd.merge(AQI_df, df_covid, on='ID')
```

```
In [60]: merged_df2.iloc[20:40]
```

Out[60]:

|    | State_Name | year_month | State Code_x | AQI       | ID       | State Code_y | case_month | hosp_yn | dea |
|----|------------|------------|--------------|-----------|----------|--------------|------------|---------|-----|
| 20 | Alaska     | 2020-05    | 2.0          | 22.057416 | 22020-05 | 2            | 2020-05    |         | 11  |
| 21 | Alaska     | 2020-06    | 2.0          | 16.250000 | 22020-06 | 2            | 2020-06    |         | 28  |
| 22 | Alaska     | 2020-07    | 2.0          | 16.859459 | 22020-07 | 2            | 2020-07    |         | 70  |
| 23 | Alaska     | 2020-08    | 2.0          | 15.285714 | 22020-08 | 2            | 2020-08    |         | 80  |
| 24 | Alaska     | 2020-09    | 2.0          | 17.220339 | 22020-09 | 2            | 2020-09    |         | 62  |
| 25 | Alaska     | 2020-10    | 2.0          | 25.626374 | 22020-10 | 2            | 2020-10    |         | 169 |
| 26 | Alaska     | 2020-11    | 2.0          | 35.616667 | 22020-11 | 2            | 2020-11    |         | 283 |
| 27 | Alaska     | 2020-12    | 2.0          | 32.054945 | 22020-12 | 2            | 2020-12    |         | 196 |
| 28 | Alaska     | 2021-01    | 2.0          | 45.664516 | 22021-01 | 2            | 2021-01    |         | 74  |
| 29 | Alaska     | 2021-02    | 2.0          | 35.892857 | 22021-02 | 2            | 2021-02    |         | 55  |
| 30 | Alaska     | 2021-03    | 2.0          | 28.780645 | 22021-03 | 2            | 2021-03    |         | 88  |
| 31 | Alaska     | 2021-04    | 2.0          | 29.298611 | 22021-04 | 2            | 2021-04    |         | 115 |
| 32 | Alaska     | 2021-05    | 2.0          | 23.941935 | 22021-05 | 2            | 2021-05    |         | 44  |
| 33 | Alaska     | 2021-06    | 2.0          | 20.729167 | 22021-06 | 2            | 2021-06    |         | 35  |
| 34 | Alaska     | 2021-07    | 2.0          | 23.062500 | 22021-07 | 2            | 2021-07    |         | 142 |
| 35 | Alaska     | 2021-08    | 2.0          | 23.612903 | 22021-08 | 2            | 2021-08    |         | 286 |
| 36 | Alaska     | 2021-09    | 2.0          | 27.433333 | 22021-09 | 2            | 2021-09    |         | 364 |
| 37 | Arizona    | 2020-01    | 4.0          | 38.815920 | 42020-01 | 4            | 2020-01    |         | 0   |
| 38 | Arizona    | 2020-02    | 4.0          | 45.862069 | 42020-02 | 4            | 2020-02    |         | 10  |
| 39 | Arizona    | 2020-03    | 4.0          | 44.865672 | 42020-03 | 4            | 2020-03    |         | 769 |

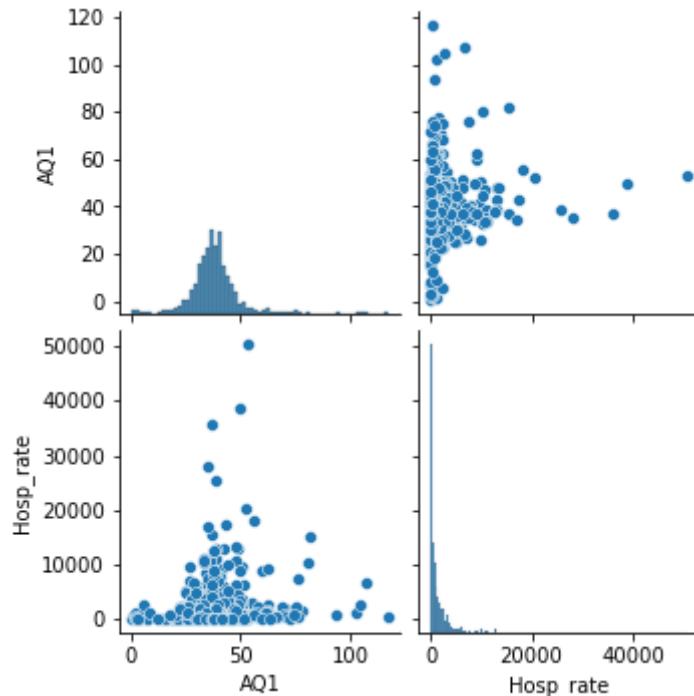
In [61]: 

```
import matplotlib.pyplot as plt
import seaborn as sns
```

```
t1 = merged_df2[['AQI', 'hosp_yn']]
t1.columns = ['AQ1', 'Hosp_rate']
plt.figure(figsize=(12,7))
sns.pairplot(t1)
```

Out[61]: <seaborn.axisgrid.PairGrid at 0x7f86c0c36040>

<Figure size 864x504 with 0 Axes>



In [62]: # Checking the null values  
merged\_df2.isnull().sum()

Out[62]:

|                          |   |
|--------------------------|---|
| State_Name               | 0 |
| year_month               | 0 |
| State_Code_x             | 0 |
| AQI                      | 0 |
| ID                       | 0 |
| State_Code_y             | 0 |
| case_month               | 0 |
| hosp_yn                  | 0 |
| death_yn                 | 0 |
| underlying_conditions_yn | 0 |
| dtype: int64             |   |

In [63]: # Taking the unique State codes  
merged\_df2['State\_Code\_y'].unique()

Out[63]: array([ 1, 2, 4, 5, 6, 8, 9, 10, 12, 13, 15, 16, 17, 18, 19, 20, 21,
 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38,
 39, 40, 41, 42, 44, 45, 46, 47, 48, 49, 50, 51, 53, 54, 55, 56])

In [64]: # Generating the list of Unique State Codes  
sc = list([ 1, 2, 4, 5, 6, 8, 9, 10, 12, 13, 15, 16, 17, 18, 19, 20, 21,
 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38,
 39, 40, 41, 42, 44, 45, 46, 47, 48, 49, 50, 51, 53, 54, 55, 56])

In [65]: ## prevent warnings from cluttering the output  
import warnings

```
warnings.filterwarnings('ignore')

# Importing plotting libraries
import seaborn as sns
import matplotlib
import matplotlib.pyplot as plt
```

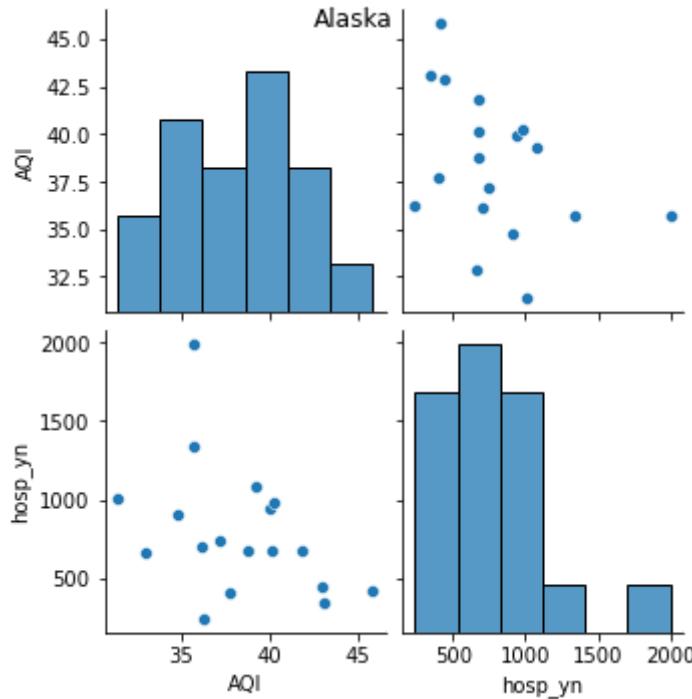
In [66]:

```
country_list=['Alabama', 'Alaska', 'Arizona', 'Arkansas', 'California',
'Colorado', 'Connecticut', 'Country Of Mexico', 'Delaware',
'District Of Columbia', 'Florida', 'Georgia', 'Hawaii', 'Idaho',
'Illinois', 'Indiana', 'Iowa', 'Kansas', 'Kentucky', 'Louisiana',
'Maine', 'Maryland', 'Massachusetts', 'Michigan', 'Minnesota',
'Mississippi', 'Missouri', 'Montana', 'Nebraska', 'Nevada',
'New Hampshire', 'New Jersey', 'New Mexico', 'New York',
'North Carolina', 'North Dakota', 'Ohio', 'Oklahoma', 'Oregon',
'Pennsylvania', 'Puerto Rico', 'Rhode Island', 'South Carolina',
'South Dakota', 'Tennessee', 'Texas', 'Utah', 'Vermont',
'Virgin Islands', 'Virginia', 'Washington', 'West Virginia',
'Wisconsin', 'Wyoming']
```

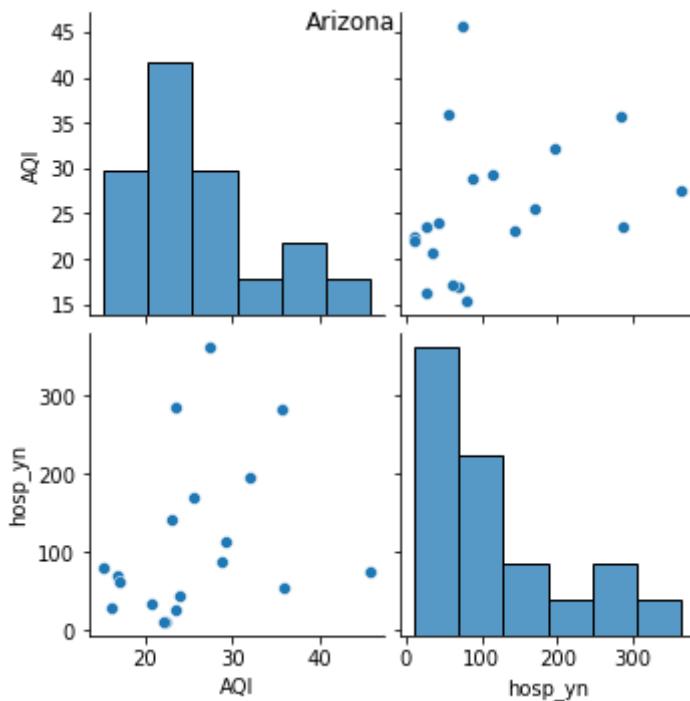
In [67]:

```
for country in country_list:
    cn=[ ]
    cn.append(country)
    df_plotter = pd.DataFrame()
    df_plotter=merged_df2[merged_df2['State_Name'].isin(cn)]
    t1 = df_plotter[['AQI', 'hosp_yn']]
    plt.suptitle(country)
    plt.figure(figsize=(15,7))
    sns.pairplot(t1,diag_kind='hist')
```

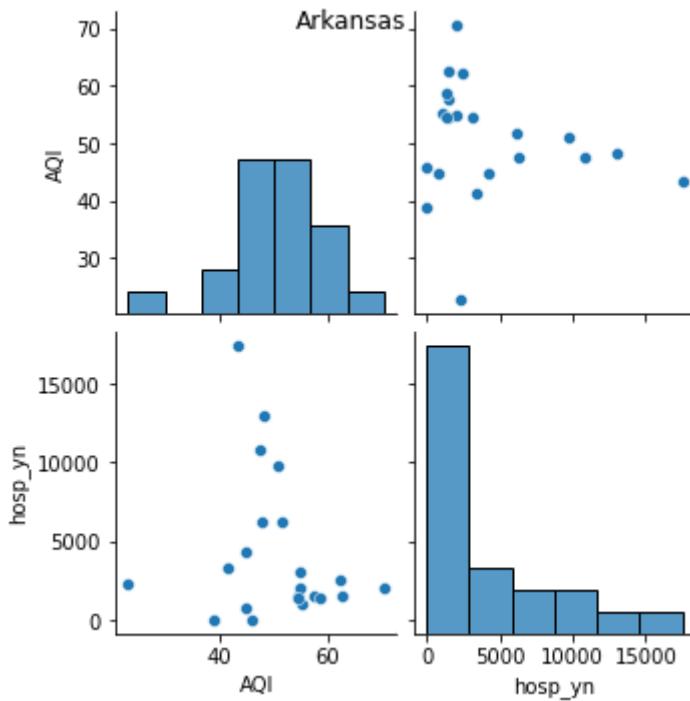
<Figure size 432x288 with 0 Axes>  
<Figure size 1080x504 with 0 Axes>



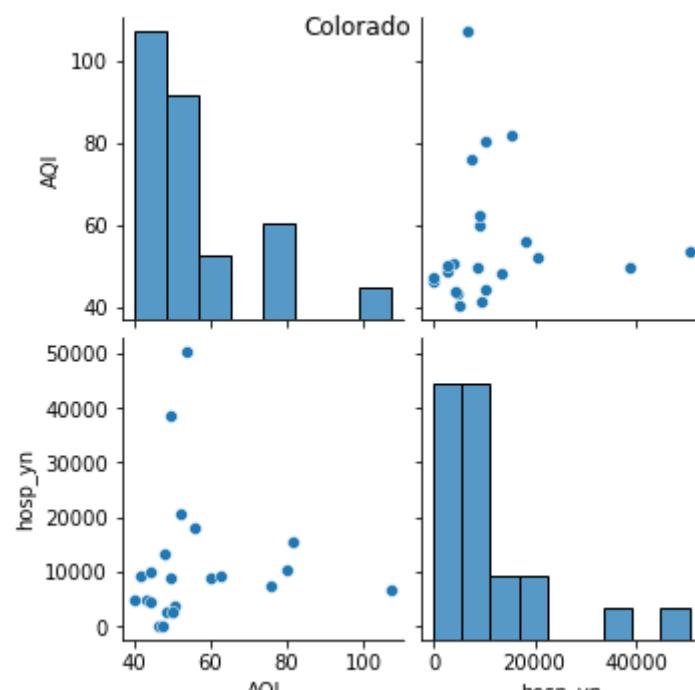
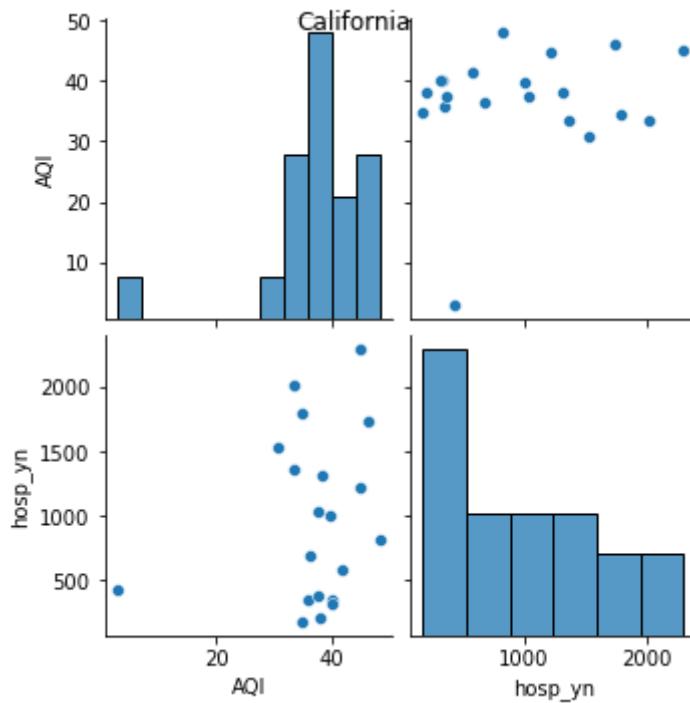
<Figure size 1080x504 with 0 Axes>



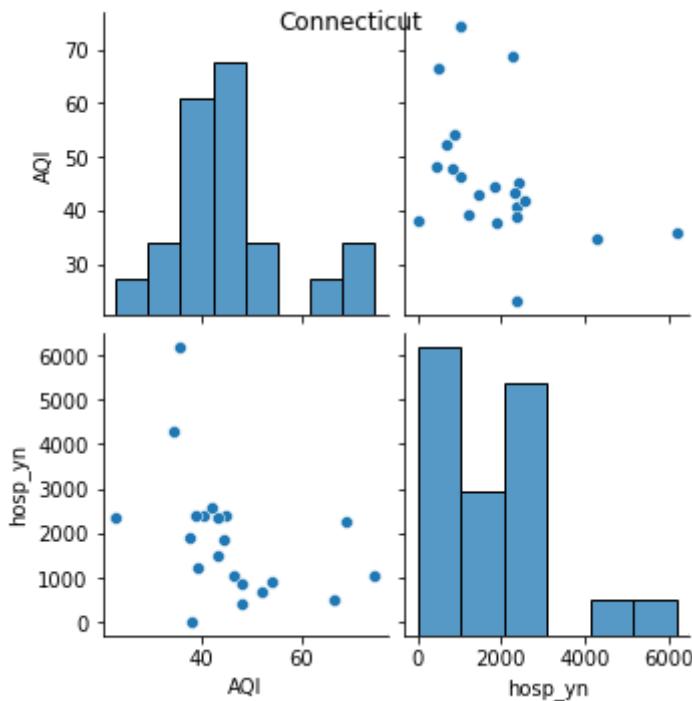
&lt;Figure size 1080x504 with 0 Axes&gt;



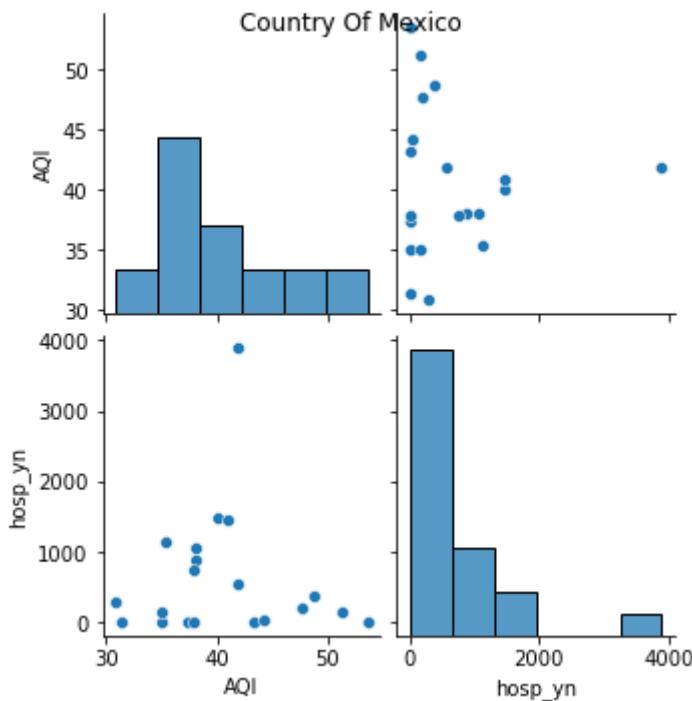
&lt;Figure size 1080x504 with 0 Axes&gt;



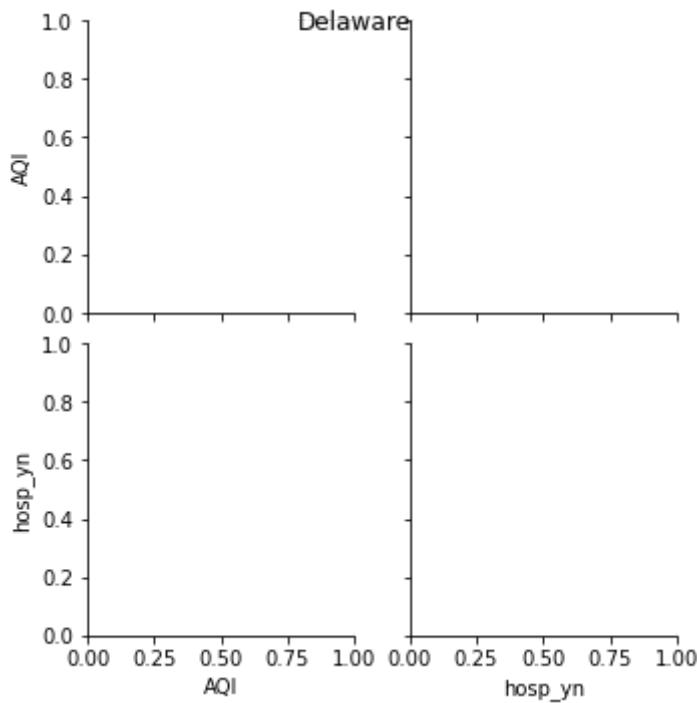
file:///Users/vishesh/Downloads/Final\_COVID-19\_Case



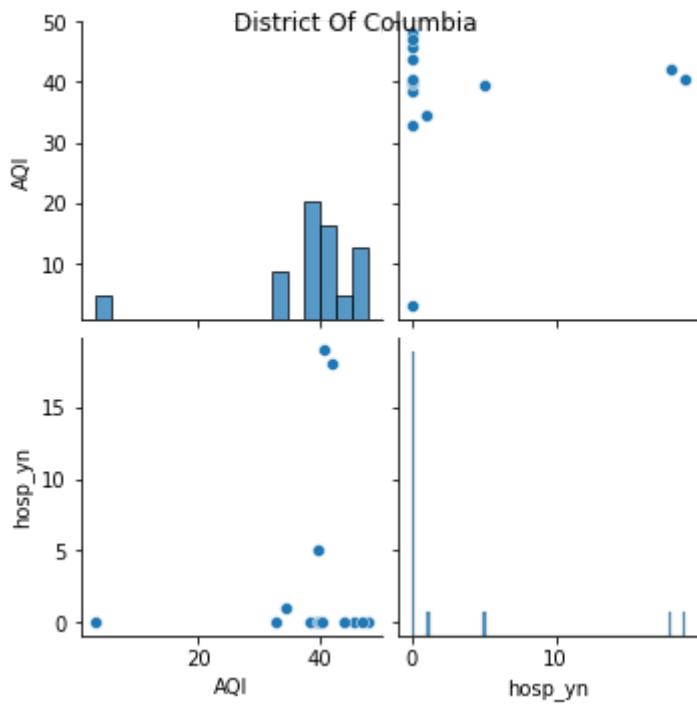
&lt;Figure size 1080x504 with 0 Axes&gt;



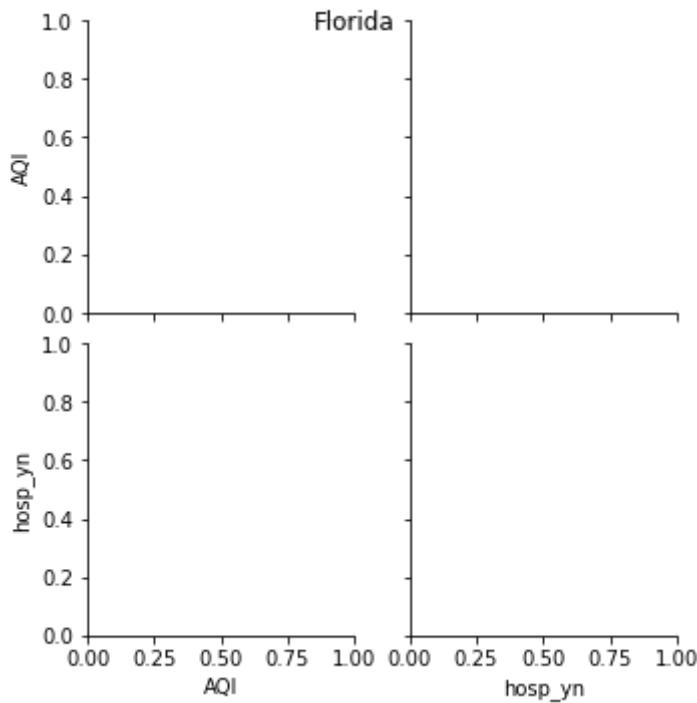
&lt;Figure size 1080x504 with 0 Axes&gt;



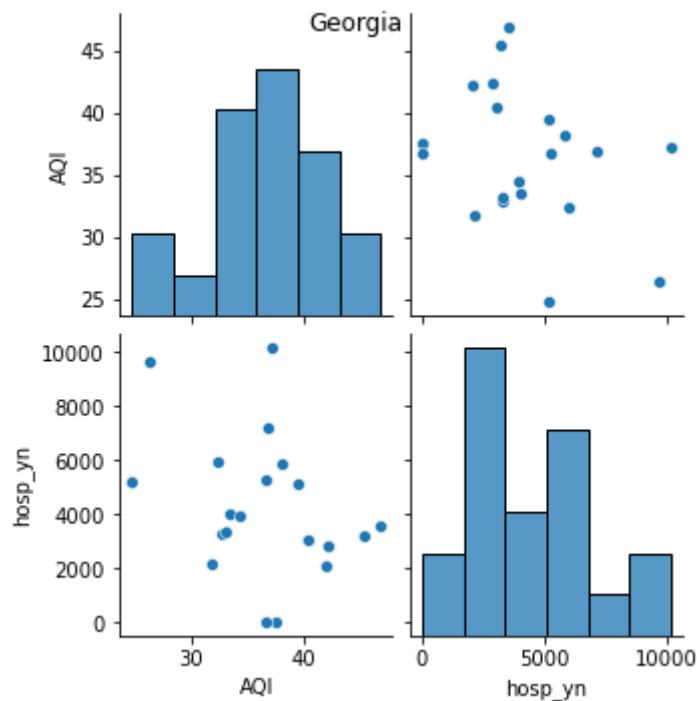
<Figure size 1080x504 with 0 Axes>



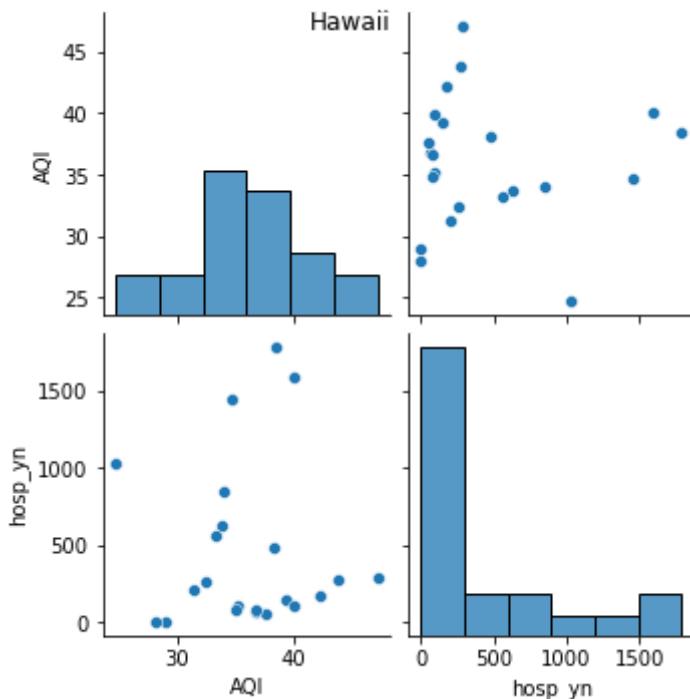
<Figure size 1080x504 with 0 Axes>



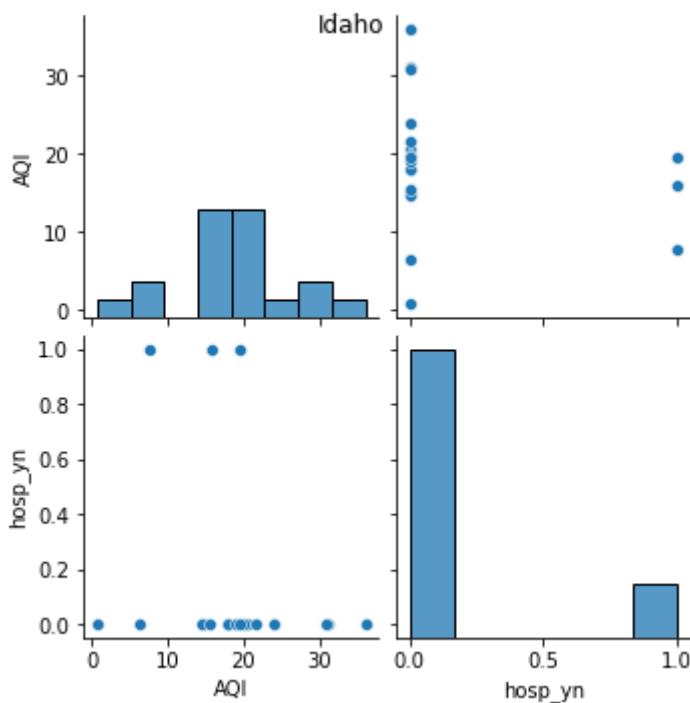
<Figure size 1080x504 with 0 Axes>



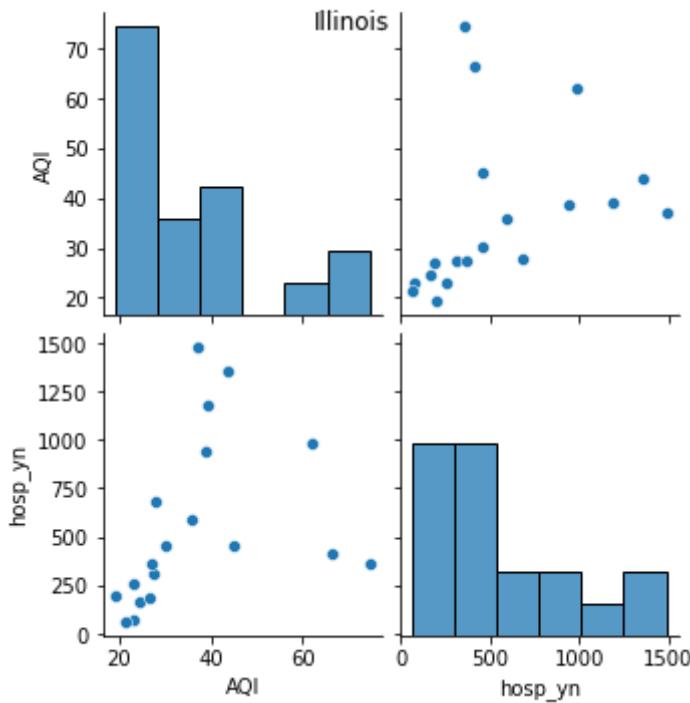
<Figure size 1080x504 with 0 Axes>



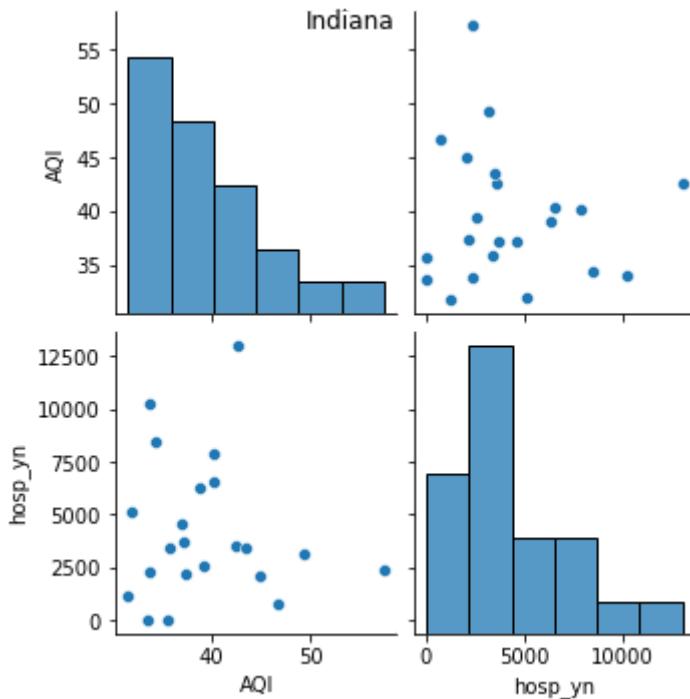
&lt;Figure size 1080x504 with 0 Axes&gt;



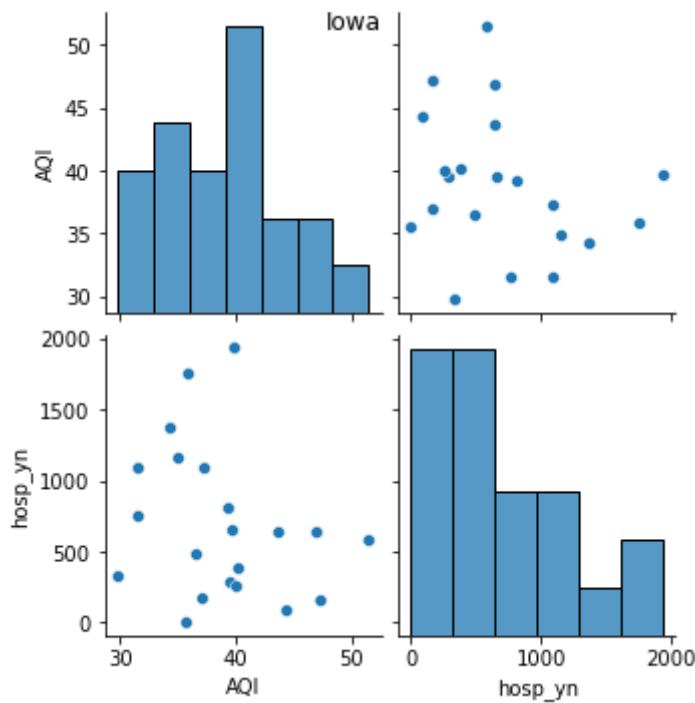
&lt;Figure size 1080x504 with 0 Axes&gt;



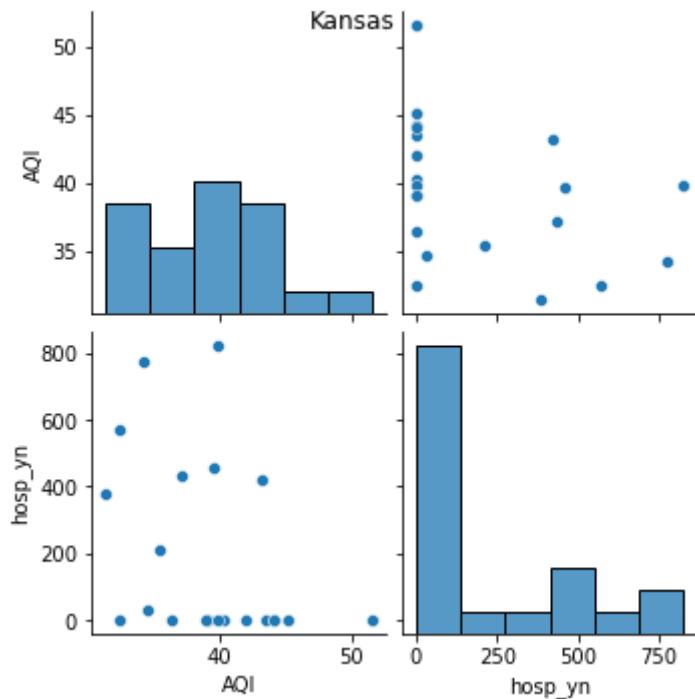
&lt;Figure size 1080x504 with 0 Axes&gt;



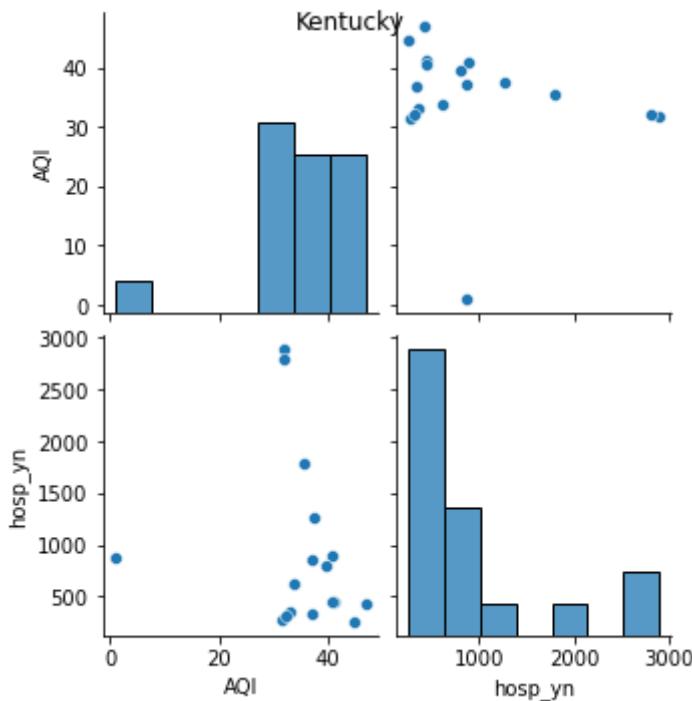
&lt;Figure size 1080x504 with 0 Axes&gt;



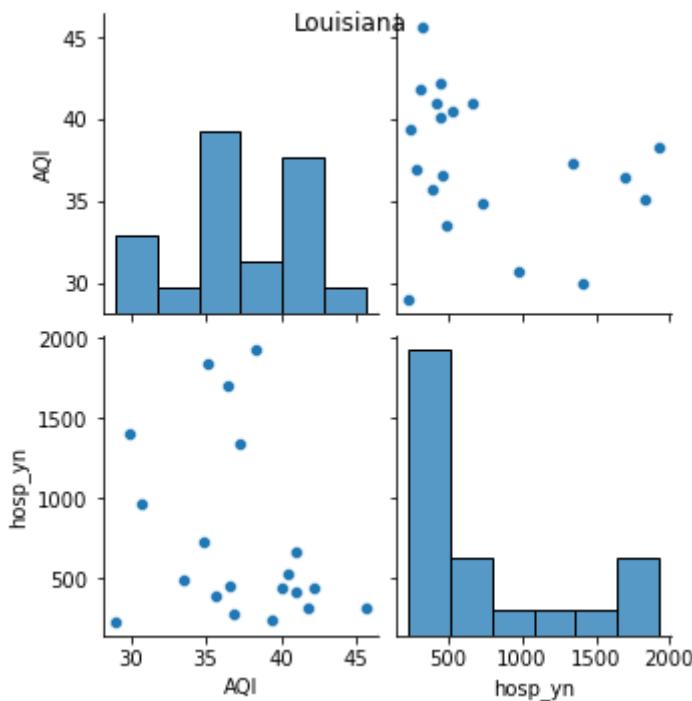
&lt;Figure size 1080x504 with 0 Axes&gt;



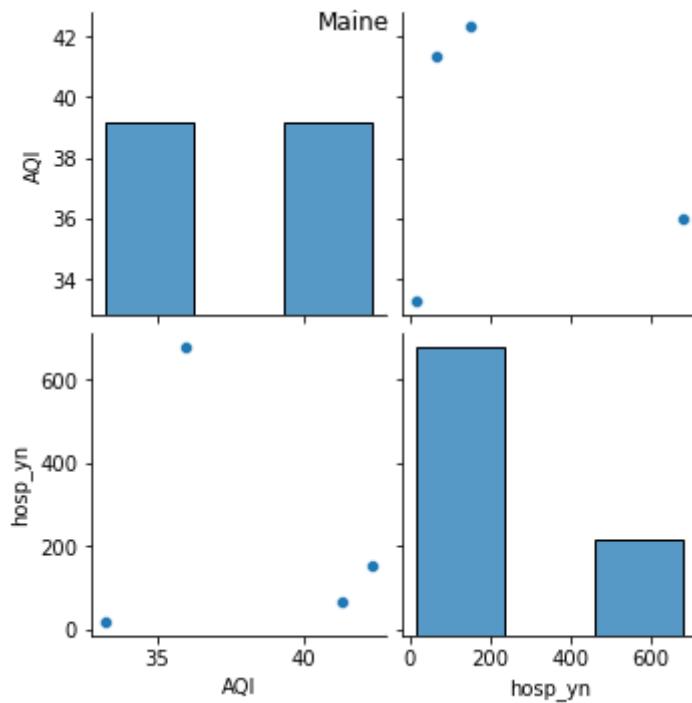
&lt;Figure size 1080x504 with 0 Axes&gt;



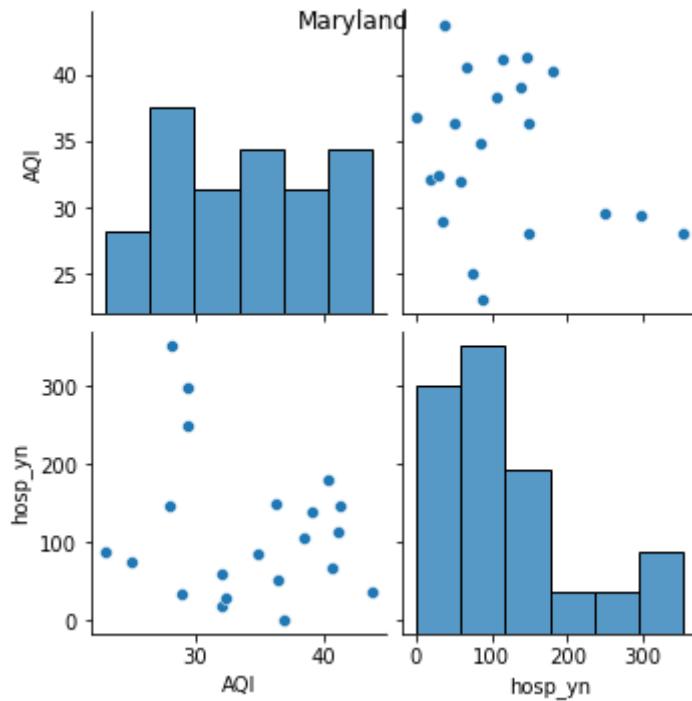
&lt;Figure size 1080x504 with 0 Axes&gt;



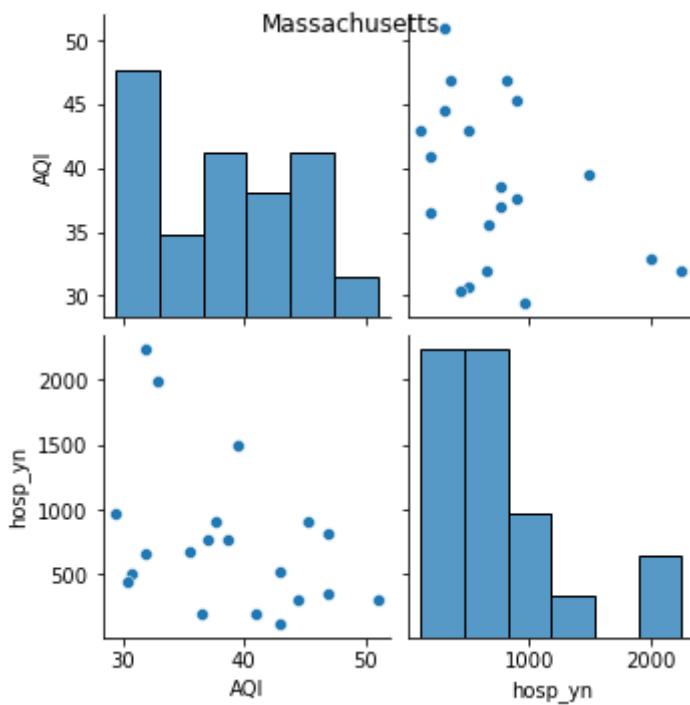
&lt;Figure size 1080x504 with 0 Axes&gt;



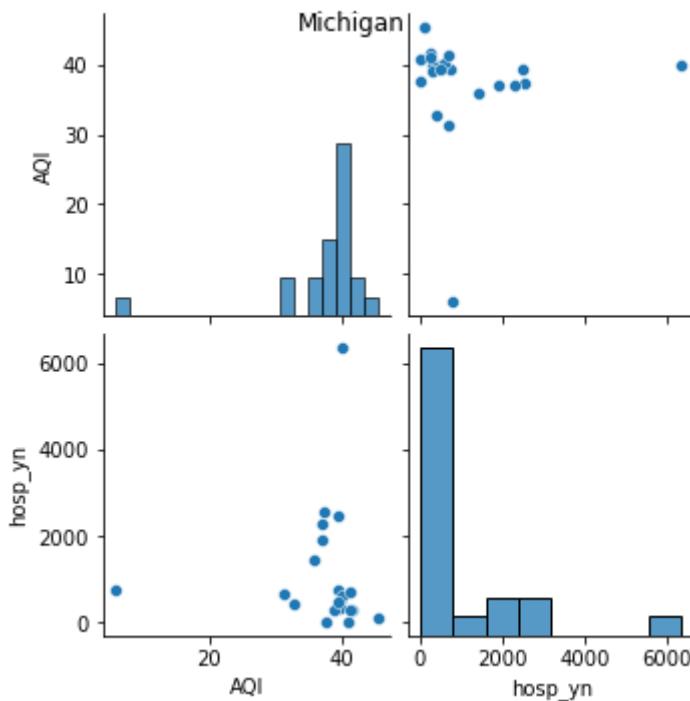
&lt;Figure size 1080x504 with 0 Axes&gt;



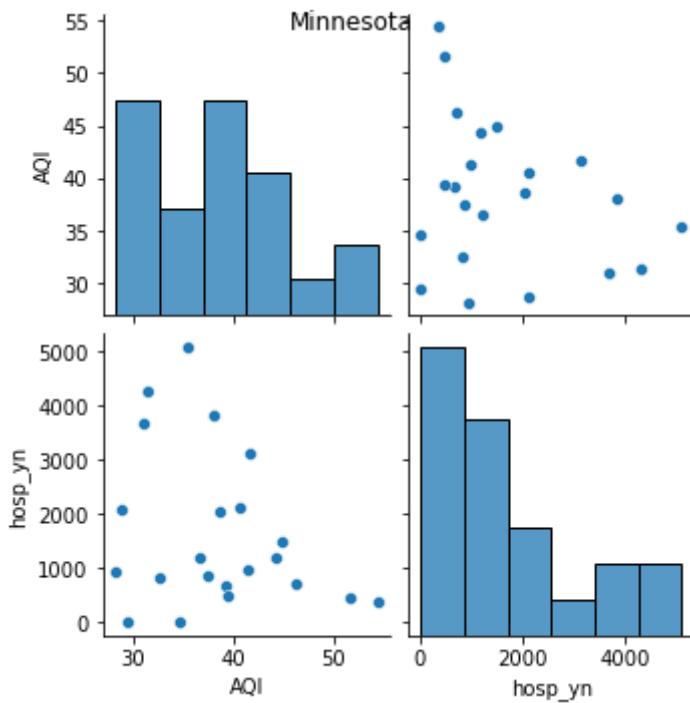
&lt;Figure size 1080x504 with 0 Axes&gt;



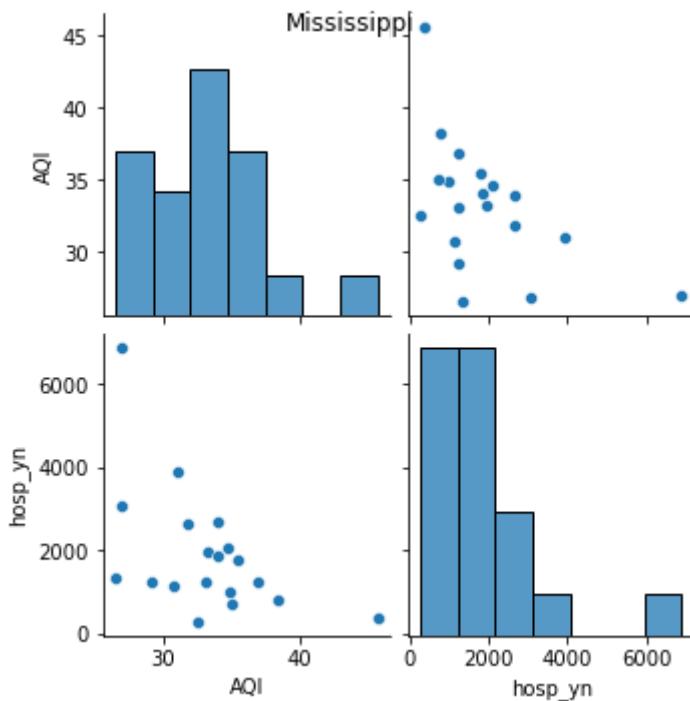
&lt;Figure size 1080x504 with 0 Axes&gt;



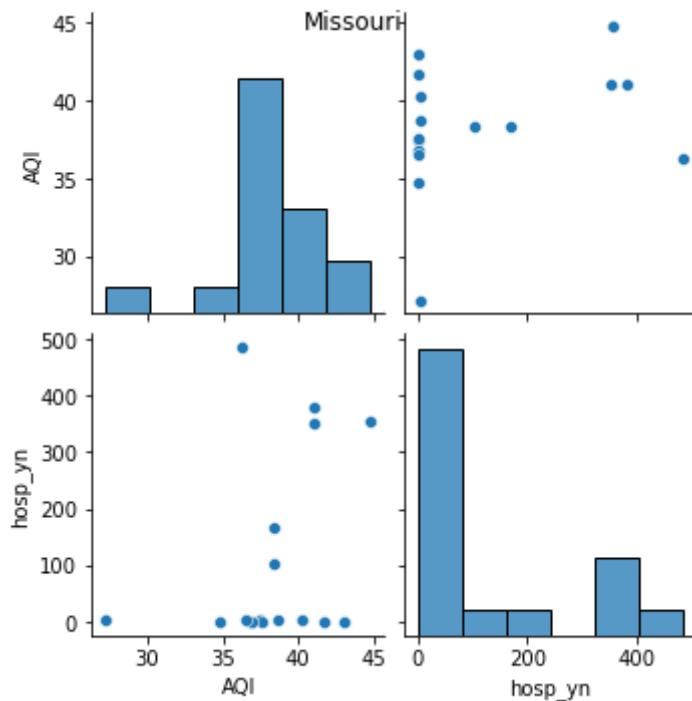
&lt;Figure size 1080x504 with 0 Axes&gt;



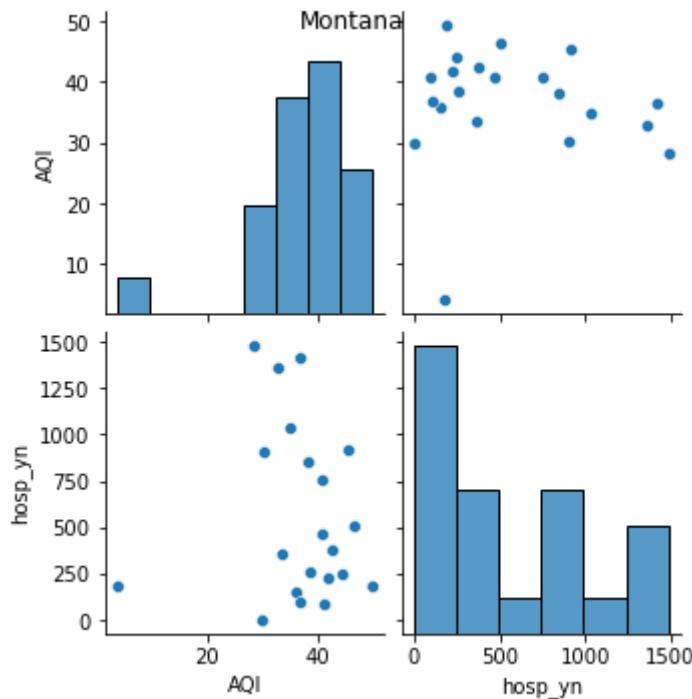
&lt;Figure size 1080x504 with 0 Axes&gt;



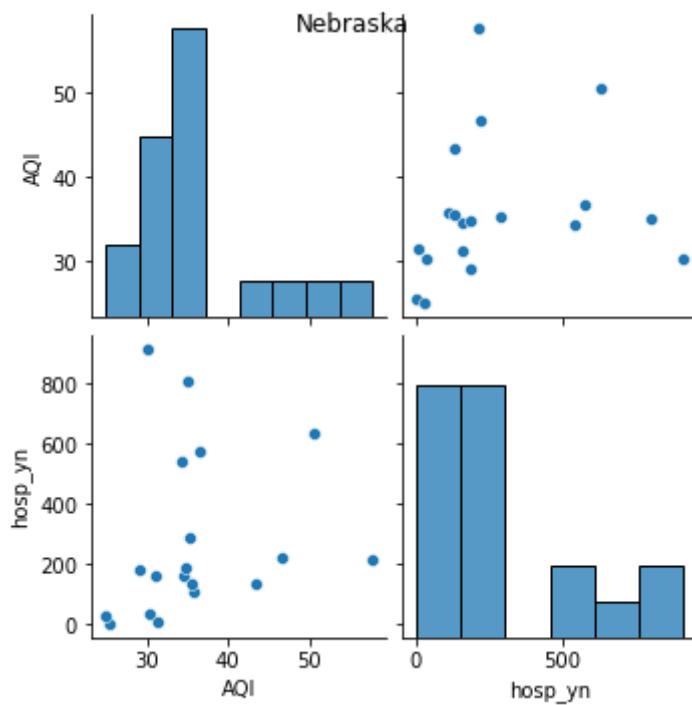
&lt;Figure size 1080x504 with 0 Axes&gt;



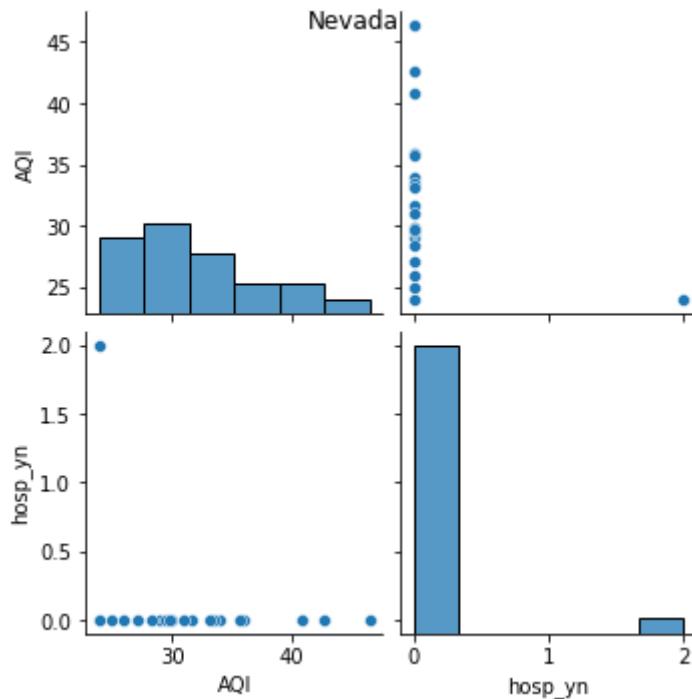
&lt;Figure size 1080x504 with 0 Axes&gt;



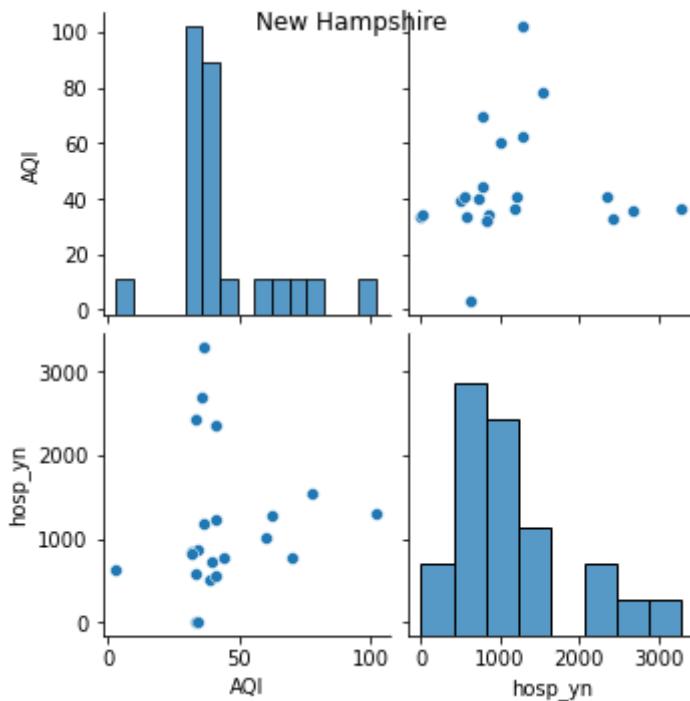
&lt;Figure size 1080x504 with 0 Axes&gt;



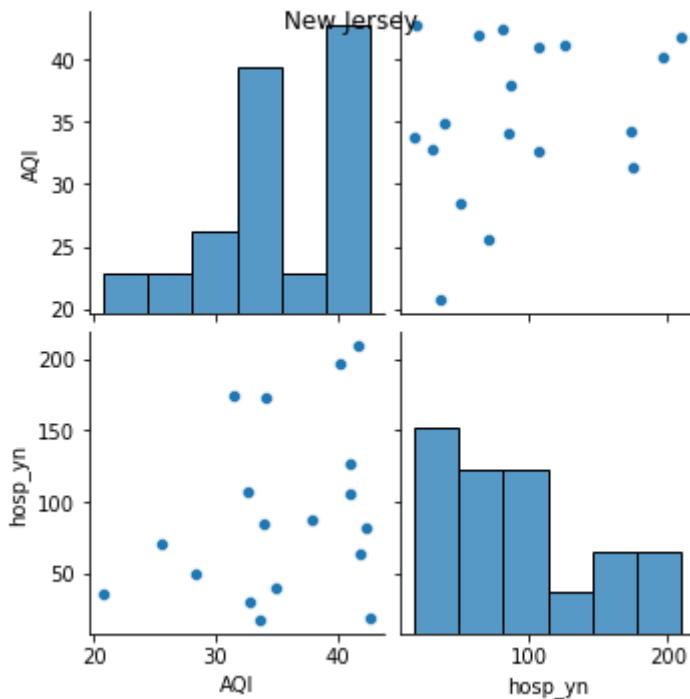
&lt;Figure size 1080x504 with 0 Axes&gt;



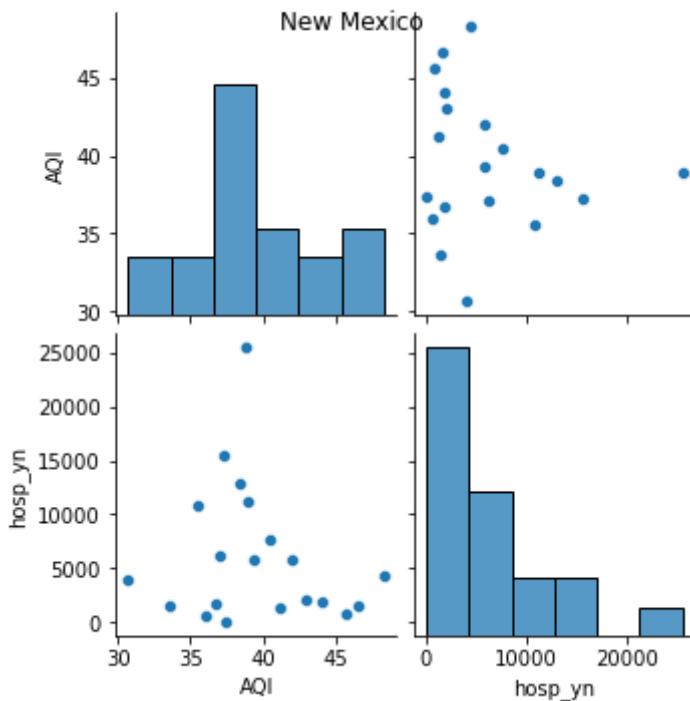
&lt;Figure size 1080x504 with 0 Axes&gt;



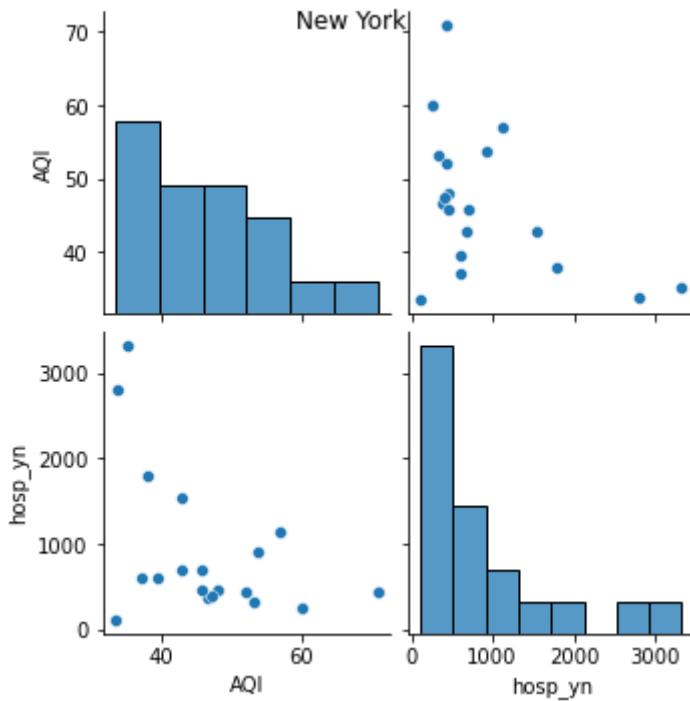
&lt;Figure size 1080x504 with 0 Axes&gt;



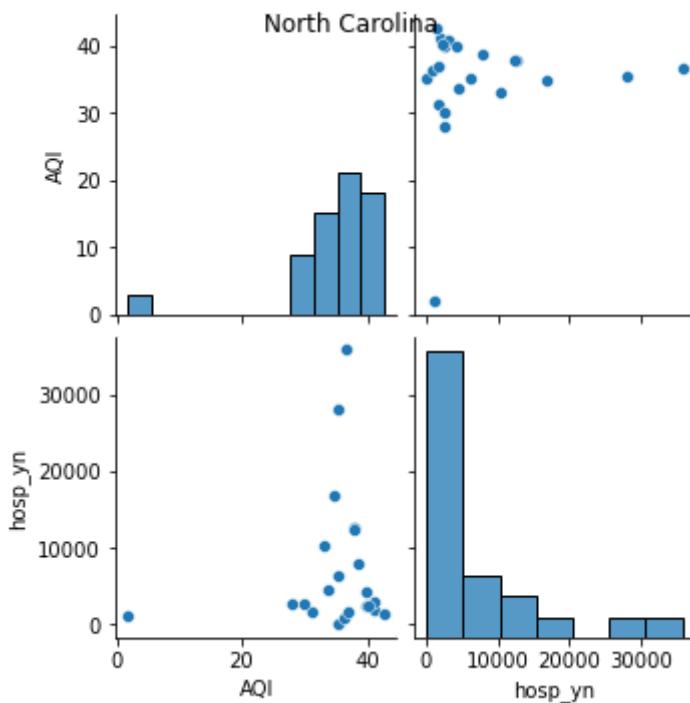
&lt;Figure size 1080x504 with 0 Axes&gt;



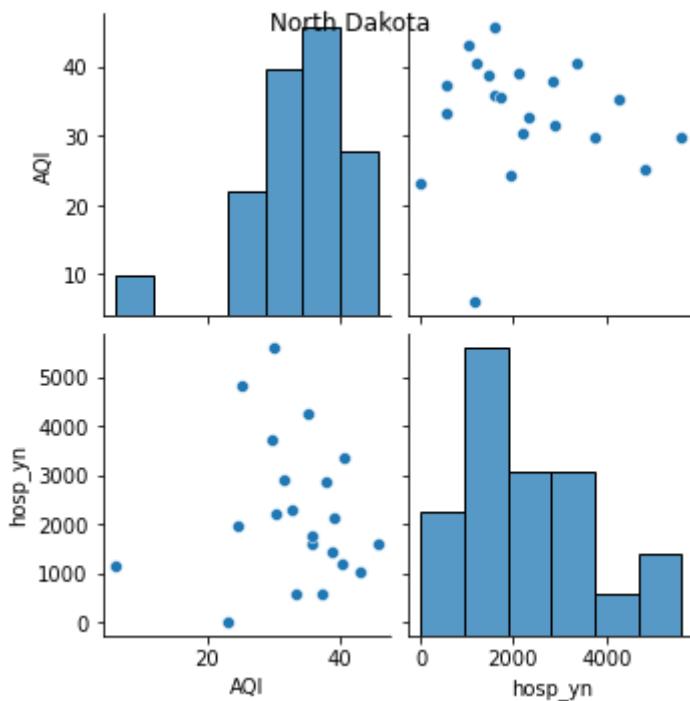
&lt;Figure size 1080x504 with 0 Axes&gt;



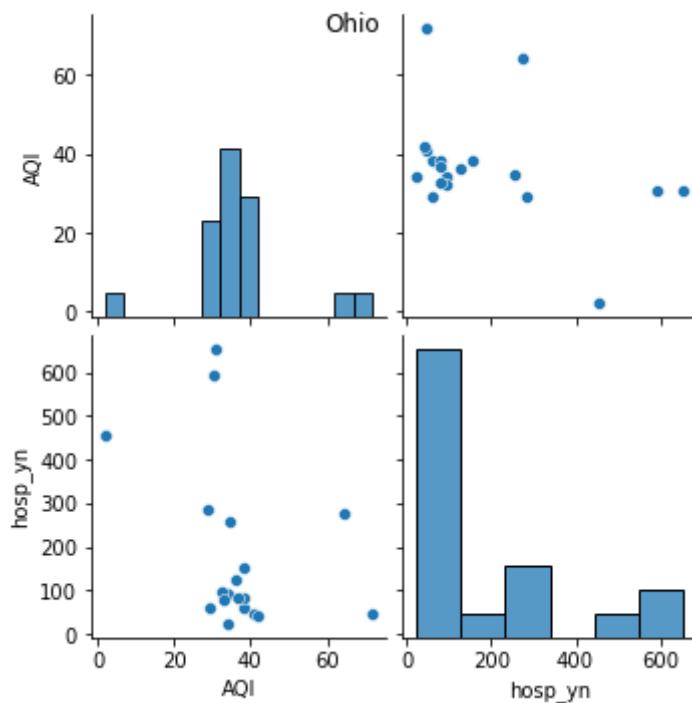
&lt;Figure size 1080x504 with 0 Axes&gt;



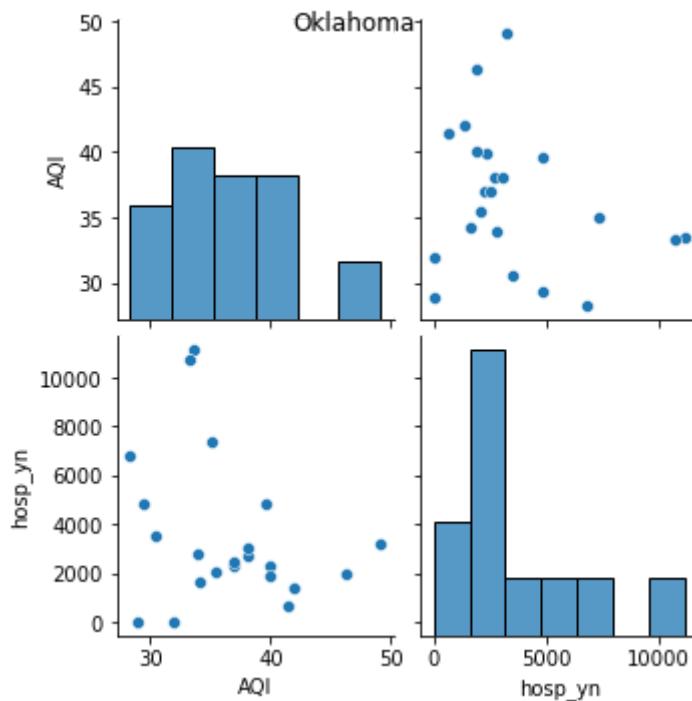
&lt;Figure size 1080x504 with 0 Axes&gt;



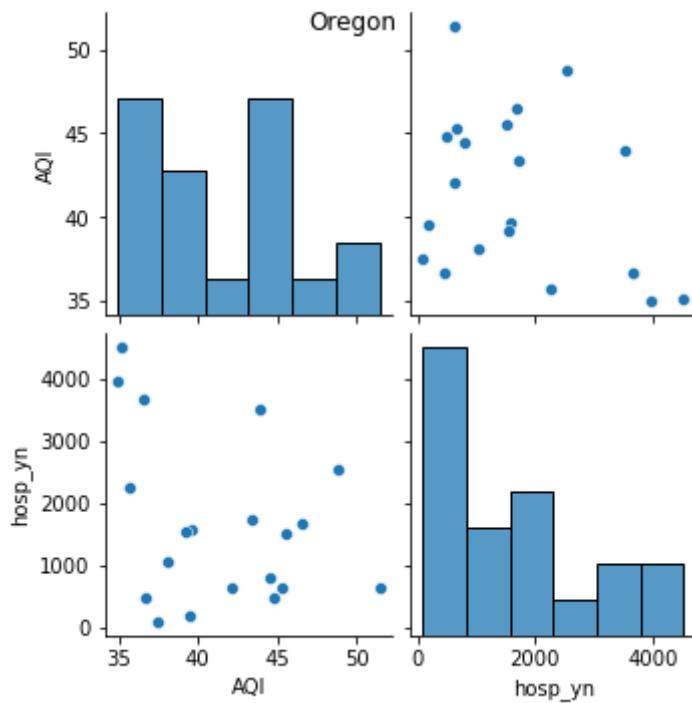
&lt;Figure size 1080x504 with 0 Axes&gt;



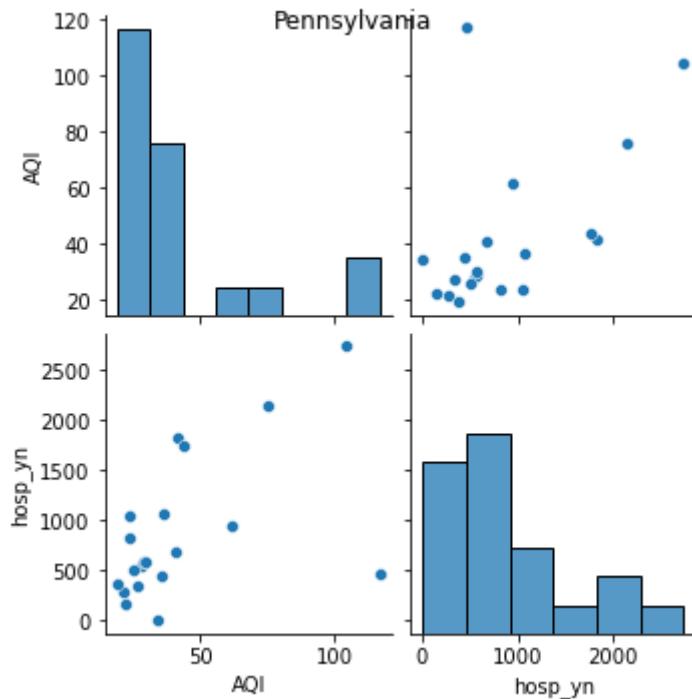
&lt;Figure size 1080x504 with 0 Axes&gt;



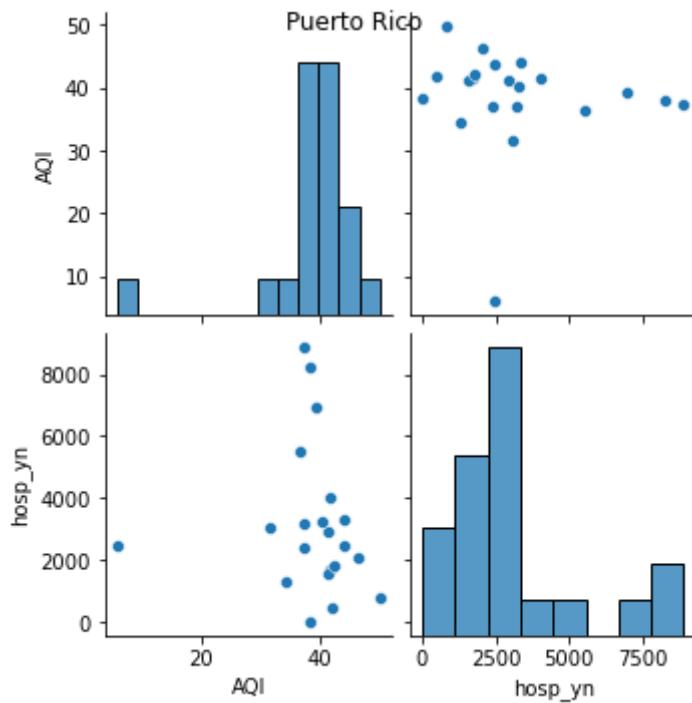
&lt;Figure size 1080x504 with 0 Axes&gt;



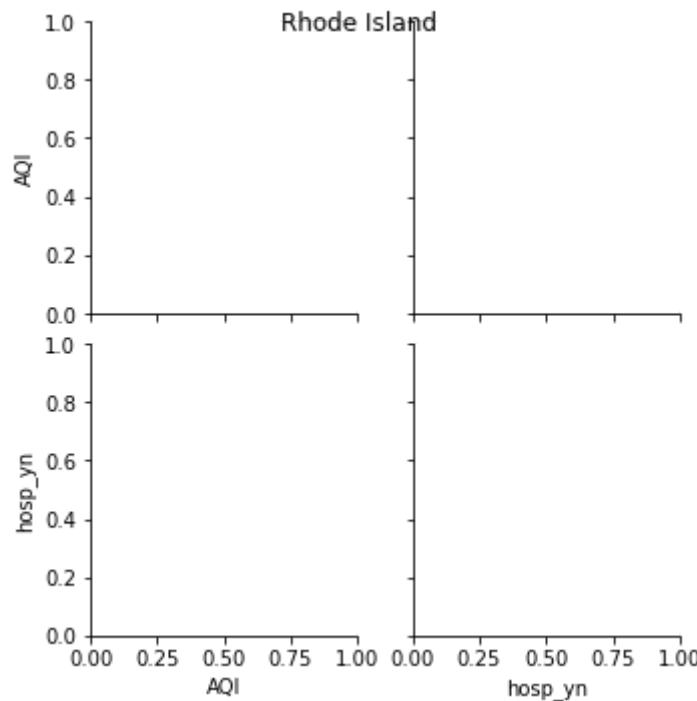
&lt;Figure size 1080x504 with 0 Axes&gt;



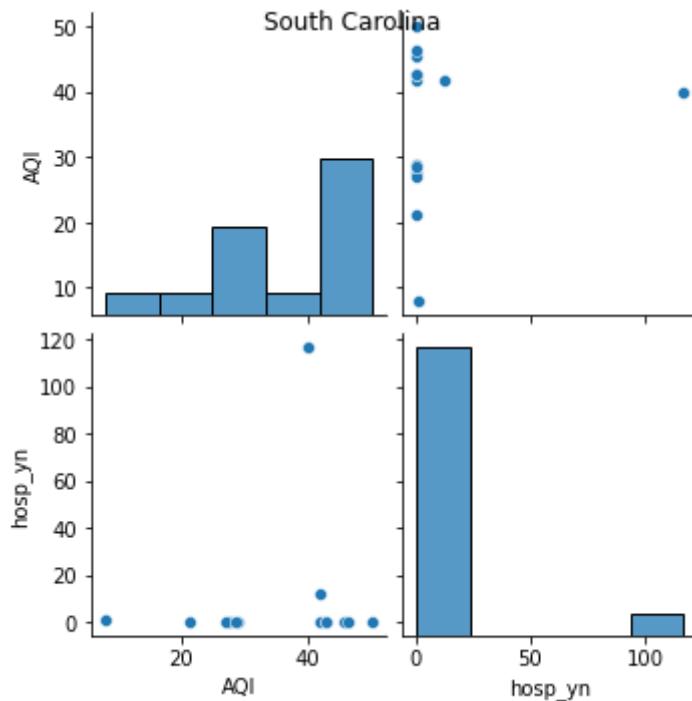
&lt;Figure size 1080x504 with 0 Axes&gt;



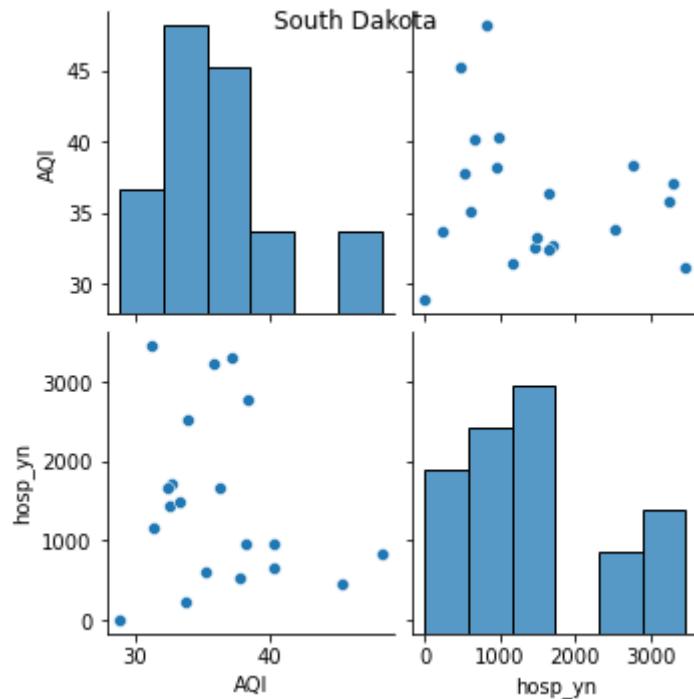
&lt;Figure size 1080x504 with 0 Axes&gt;



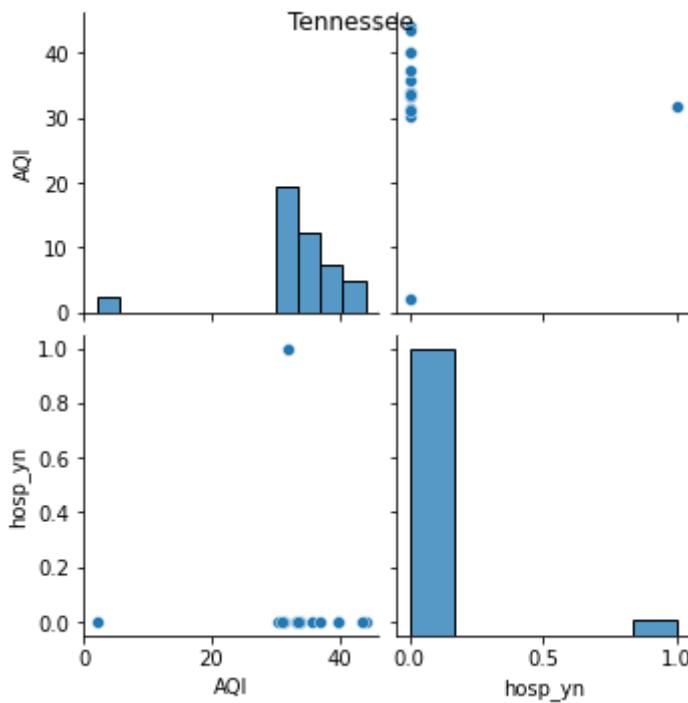
&lt;Figure size 1080x504 with 0 Axes&gt;



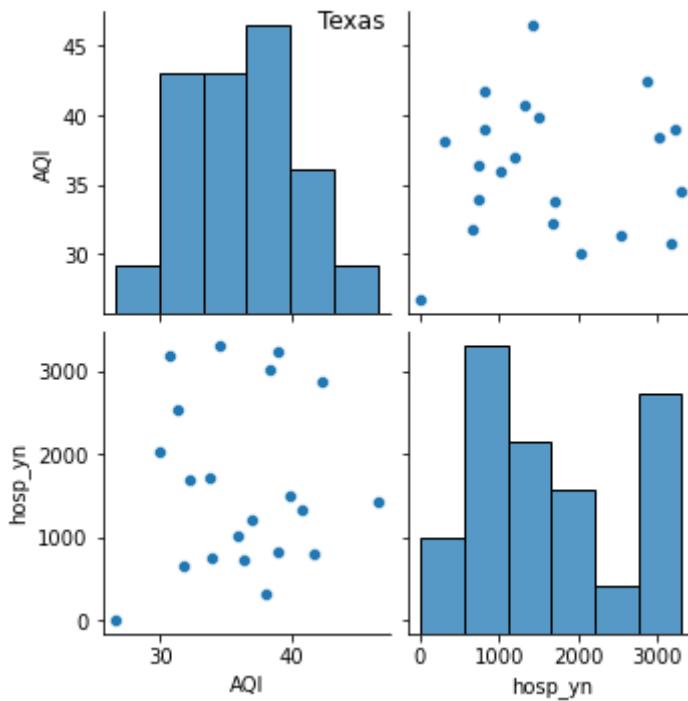
&lt;Figure size 1080x504 with 0 Axes&gt;



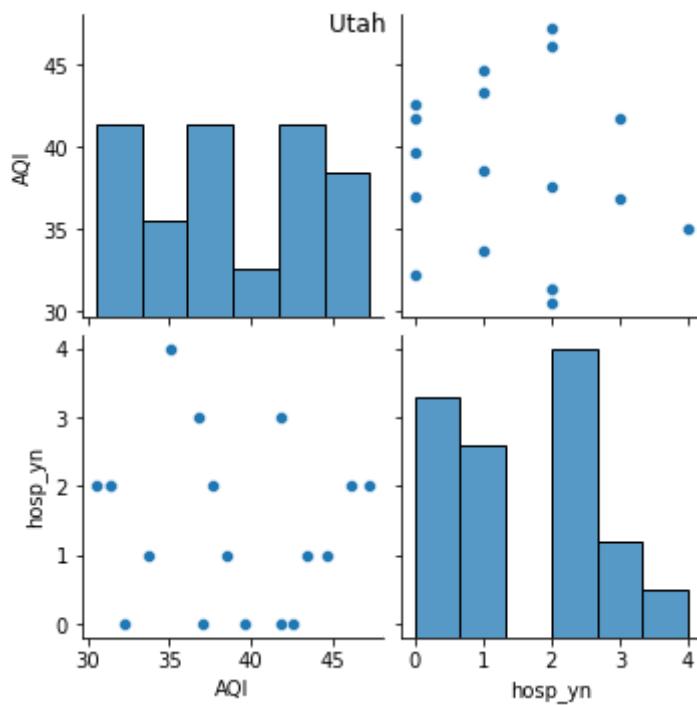
&lt;Figure size 1080x504 with 0 Axes&gt;



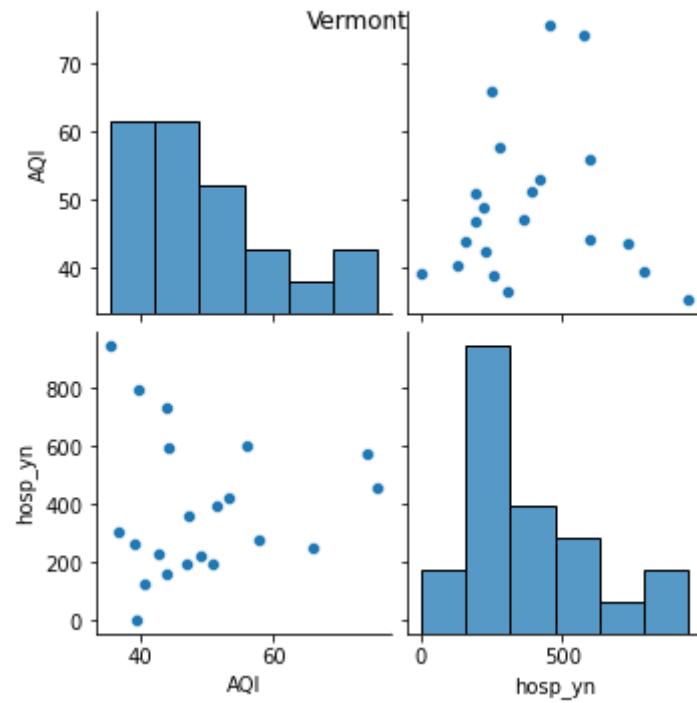
&lt;Figure size 1080x504 with 0 Axes&gt;



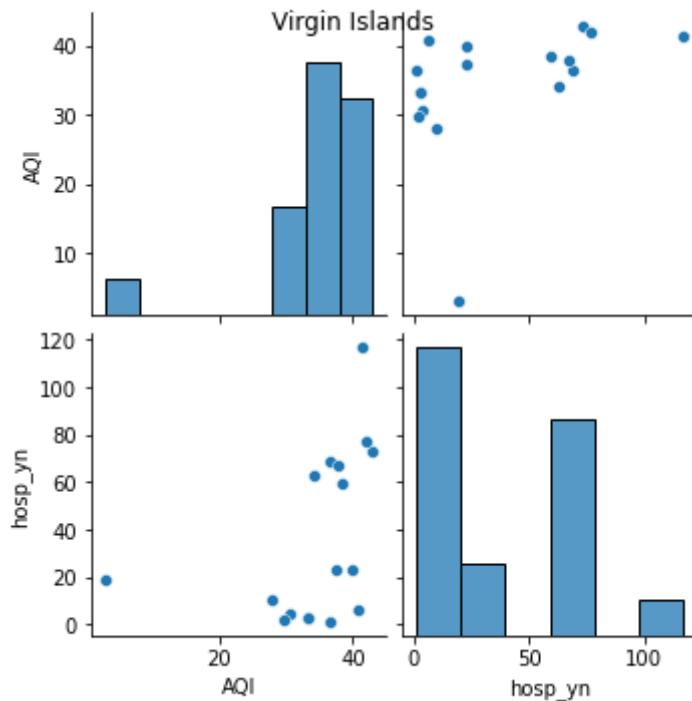
&lt;Figure size 1080x504 with 0 Axes&gt;



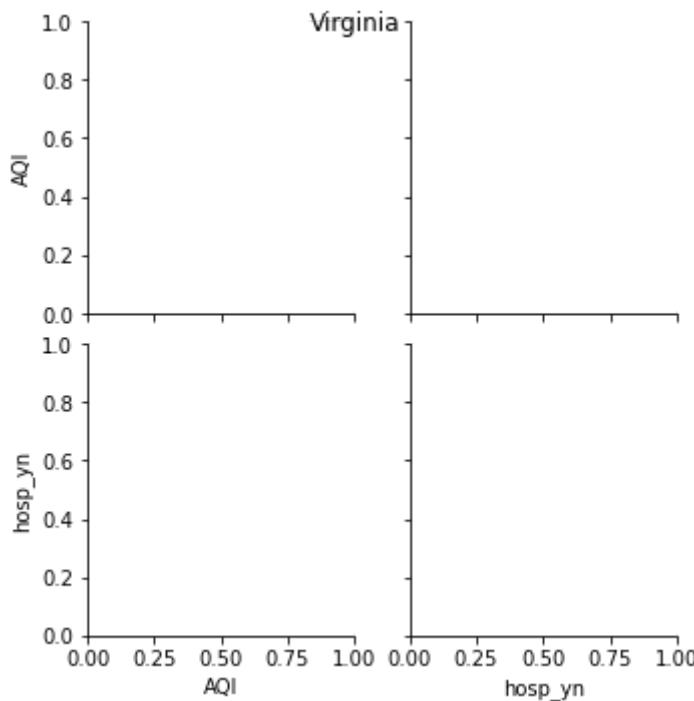
&lt;Figure size 1080x504 with 0 Axes&gt;



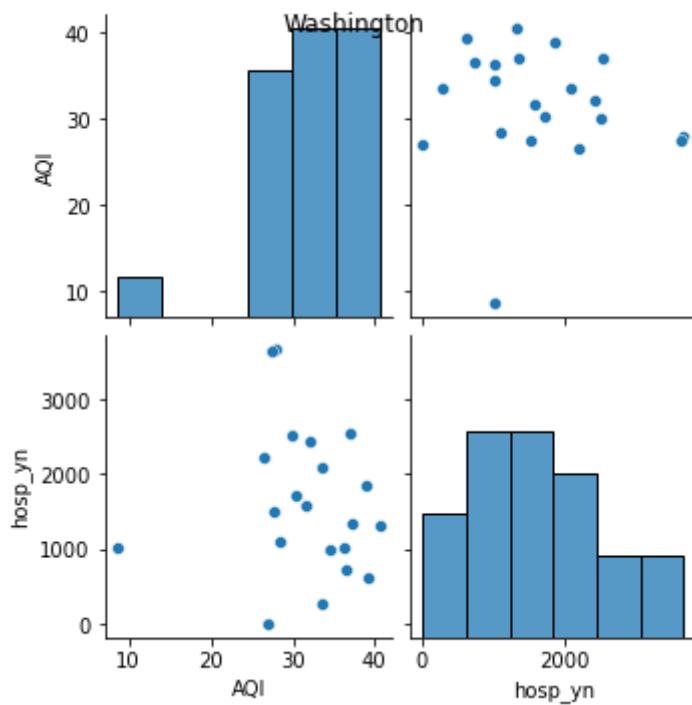
&lt;Figure size 1080x504 with 0 Axes&gt;



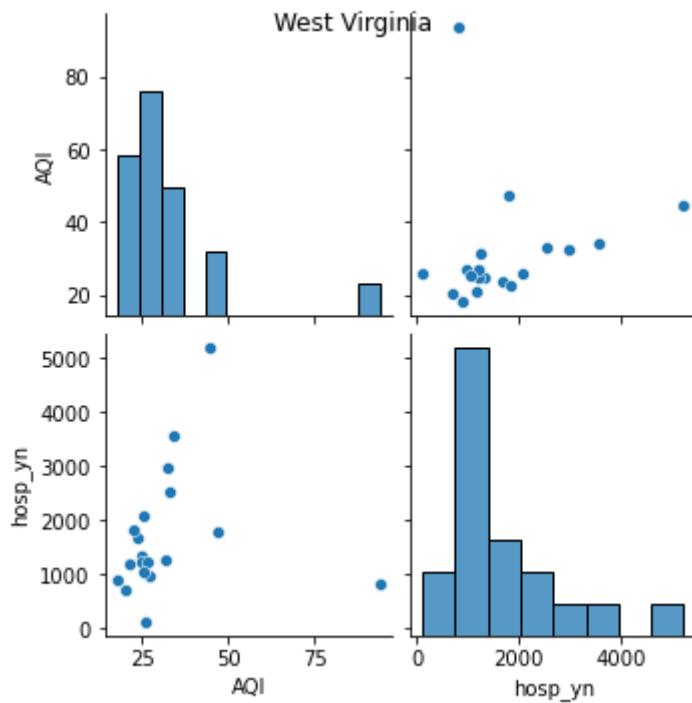
&lt;Figure size 1080x504 with 0 Axes&gt;



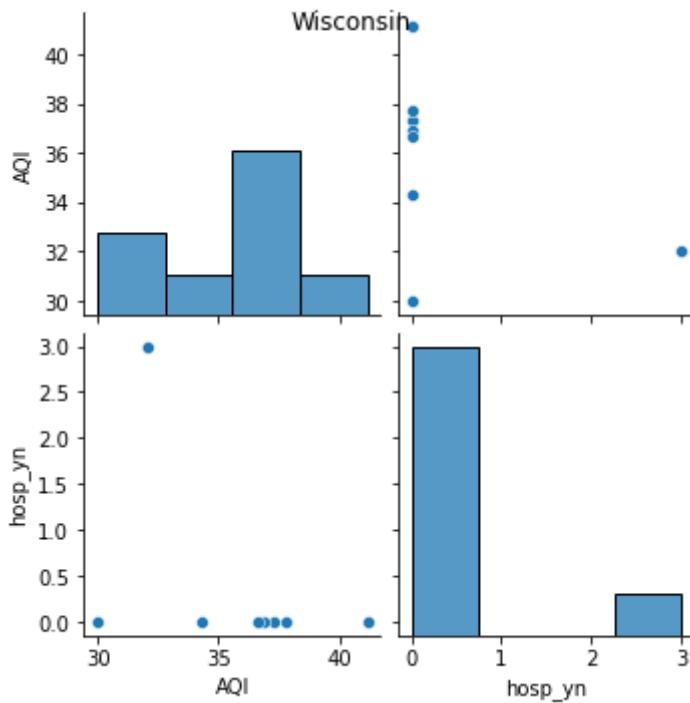
&lt;Figure size 1080x504 with 0 Axes&gt;



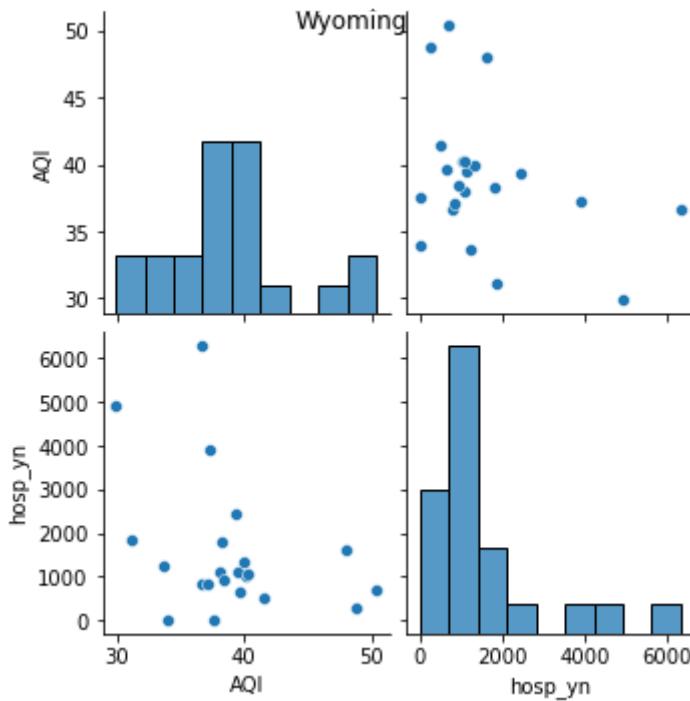
&lt;Figure size 1080x504 with 0 Axes&gt;



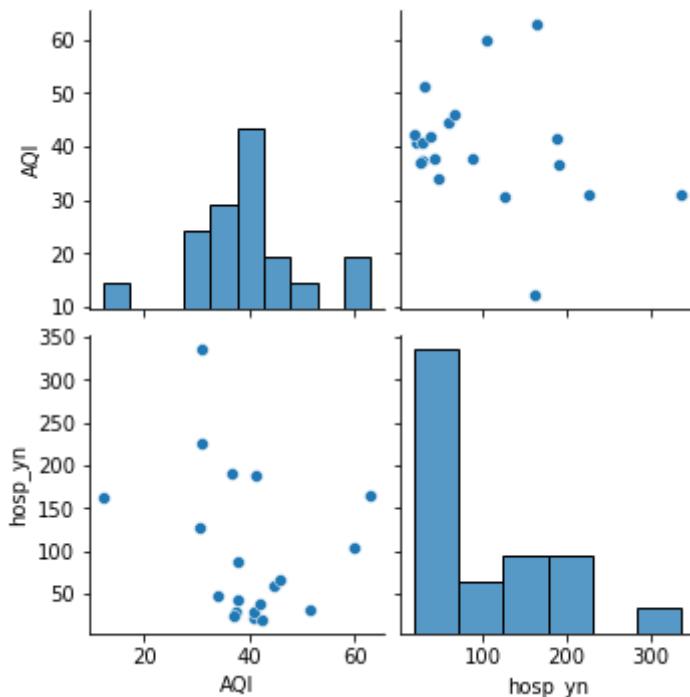
&lt;Figure size 1080x504 with 0 Axes&gt;



&lt;Figure size 1080x504 with 0 Axes&gt;



&lt;Figure size 1080x504 with 0 Axes&gt;



```
In [68]: ## For the above graphs Each graph has its own ""Title"" with State Name
```

---



---

## Is there a correlation between Air Quality Index (AQI) and COVID-19 hospitalization rate?

Ans:

Yes there is a correlation between AQI and hosp\_yn,

For State Code: {44, 41, 36, 25, 20, 21, 18, 8, 6, 5, 2} which represents State Name: {Rhode Island, Oregon, New York, Massachusetts, Kansas, Kentucky, Indiana, Colorado, California, } is directly correlated with AQI and Hosp\_yn

For few States Code: {1, 5, 26, 27, 42, 39, 45, 48, 17, 19 } which represents State Name: {Alabama, Arkansas, Minnesota, Michigan, Missouri, Montana, Nebraska, Ohio, Pennsylvania, South Carolina, Texas, Iowa, Illinois } is not directly correlated with AQI. There could be several reasons why the correlation between AQI and hospitalization rate is not direct for certain states. Some possible explanations include:

- 1) Other factors, such as symptom\_status, age distribution{Age\_group}, and access to healthcare, may play a larger role in determining hospitalization rates in those states.
- 2) The data used in the analysis may not be comprehensive enough to accurately capture the relationship between AQI and hospitalization rate in those states.

3) There may be differences in how AQI is measured or reported in those states, leading to discrepancies in the data.

For Reference: The Above Answer is Based on the heat plots of AQI vs Hosp\_yn for each state code|

## {Final Summary on Correlation}

The analysis of the heatmap graph suggests that there is a correlation between AQI and COVID-19 hospitalization rate. The data shows that for certain state codes, the AQI is directly correlated with the hospitalization rate. However, for other states, the correlation is not as clear. The Heat graph displays a total of 12 stacks, indicating that the data covers a 12 month period, or 8 months in the case of 8 stacks. Overall, it can be inferred that AQI may play a role in the hospitalization rate of COVID-19 patients in some states.

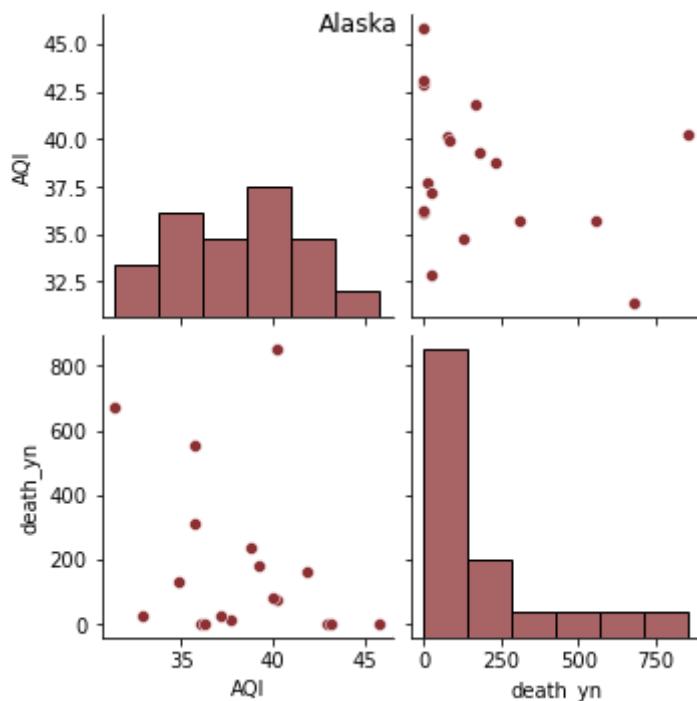
---

---

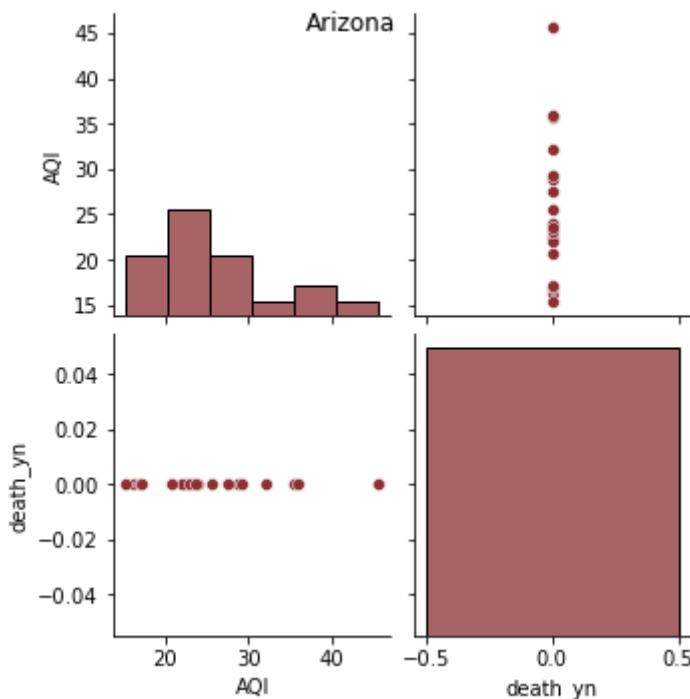
## Analyzing other Co-founding Factors

```
In [69]: for country in country_list:  
    cn=[ ]  
    cn.append(country)  
    df_plotter = pd.DataFrame()  
    df_plotter=merged_df2[merged_df2['State_Name'].isin(cn)]  
    t1 = df_plotter[['AQI', 'death_yn']]  
    plt.suptitle(country)  
    sns.set_palette("Reds_r", desat=0.6)  
    plt.figure(figsize=(15,7))  
    sns.pairplot(t1,diag_kind='hist')
```

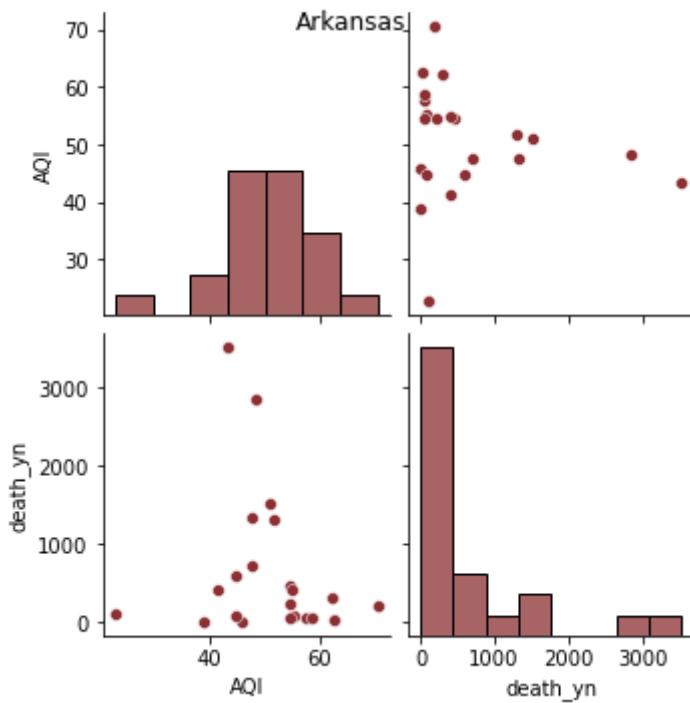
<Figure size 432x288 with 0 Axes>  
<Figure size 1080x504 with 0 Axes>



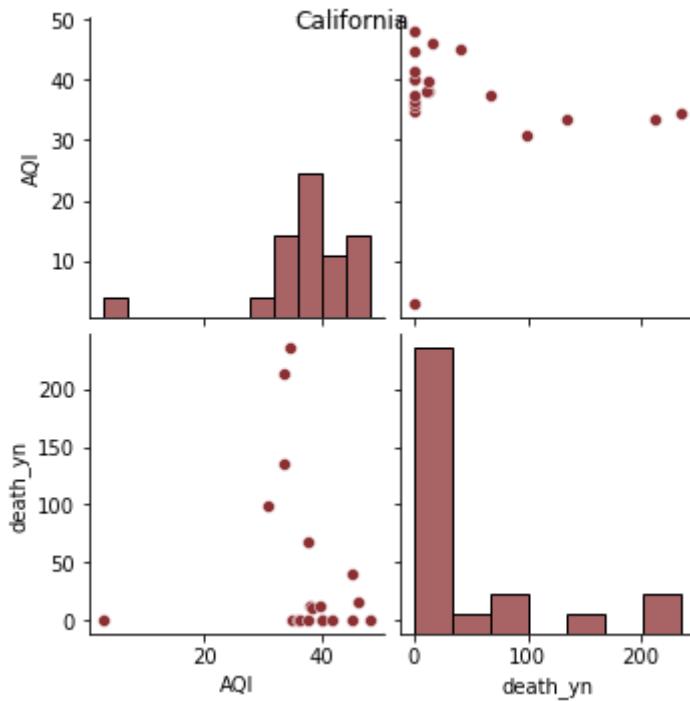
&lt;Figure size 1080x504 with 0 Axes&gt;



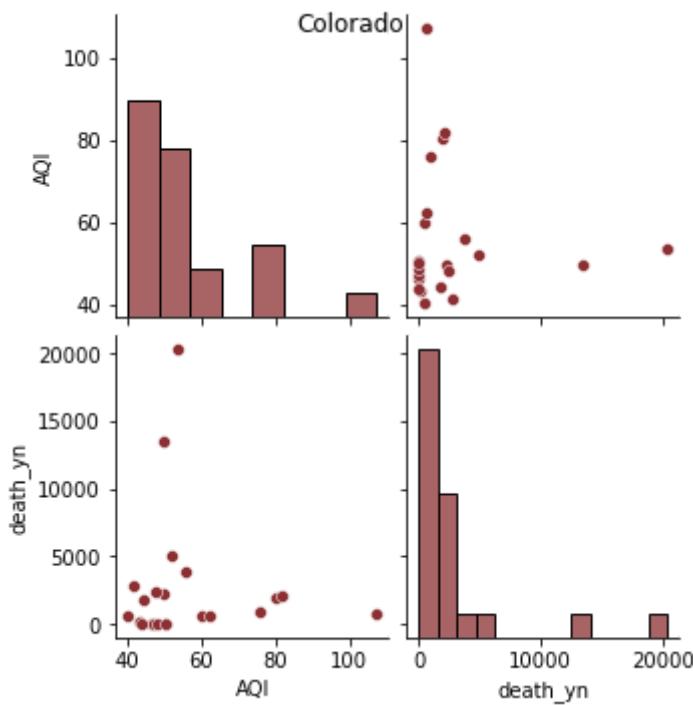
&lt;Figure size 1080x504 with 0 Axes&gt;



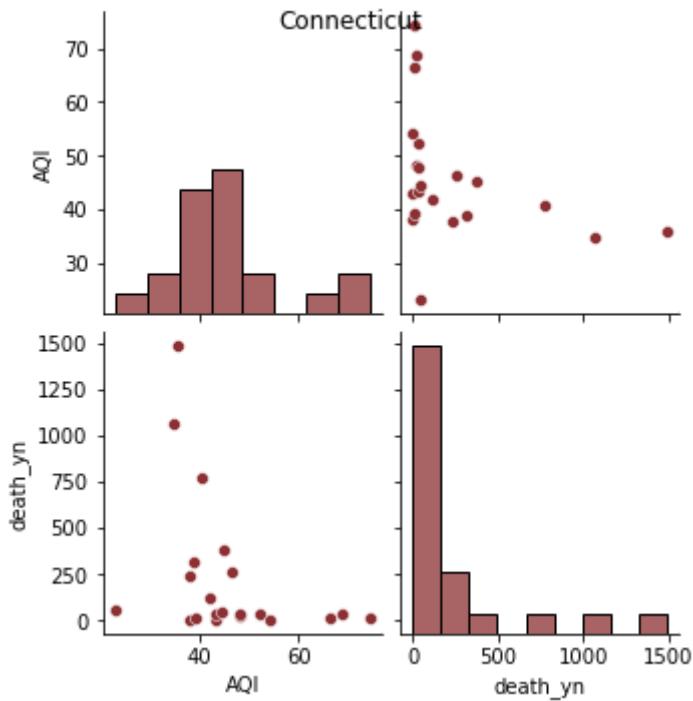
&lt;Figure size 1080x504 with 0 Axes&gt;



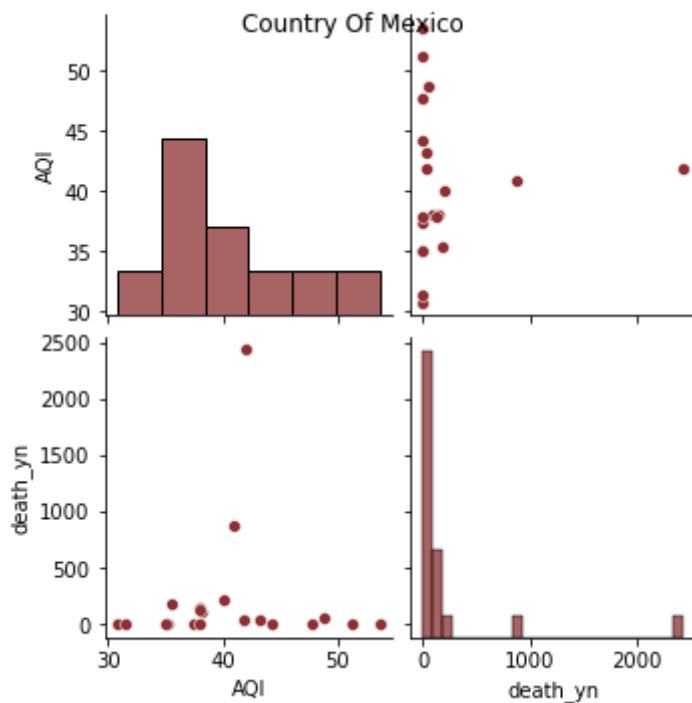
&lt;Figure size 1080x504 with 0 Axes&gt;



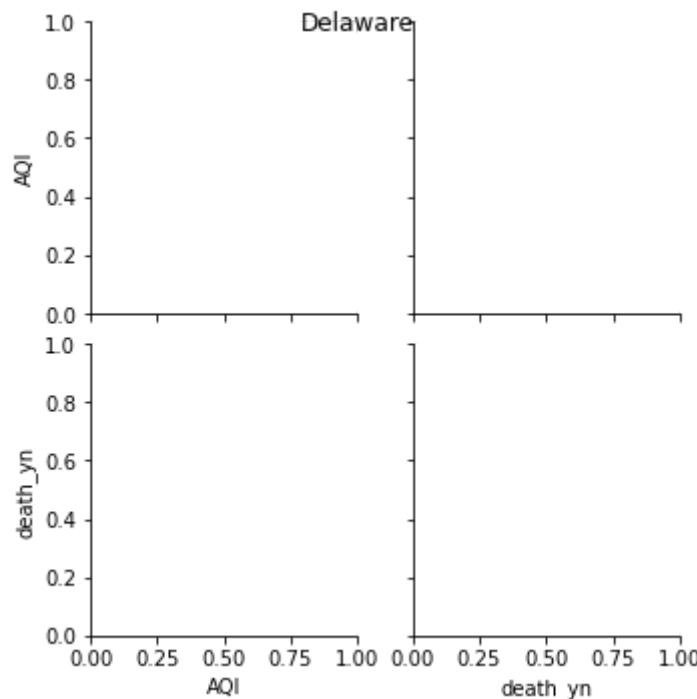
&lt;Figure size 1080x504 with 0 Axes&gt;



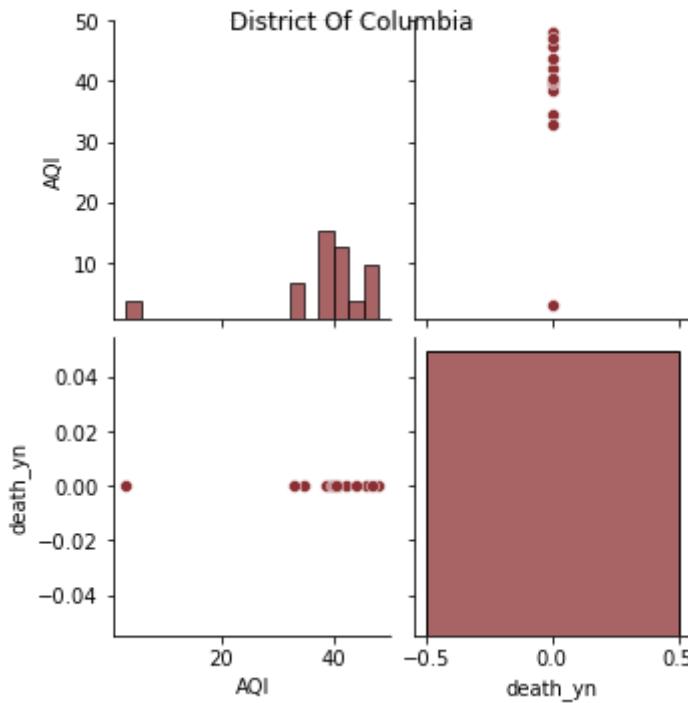
&lt;Figure size 1080x504 with 0 Axes&gt;



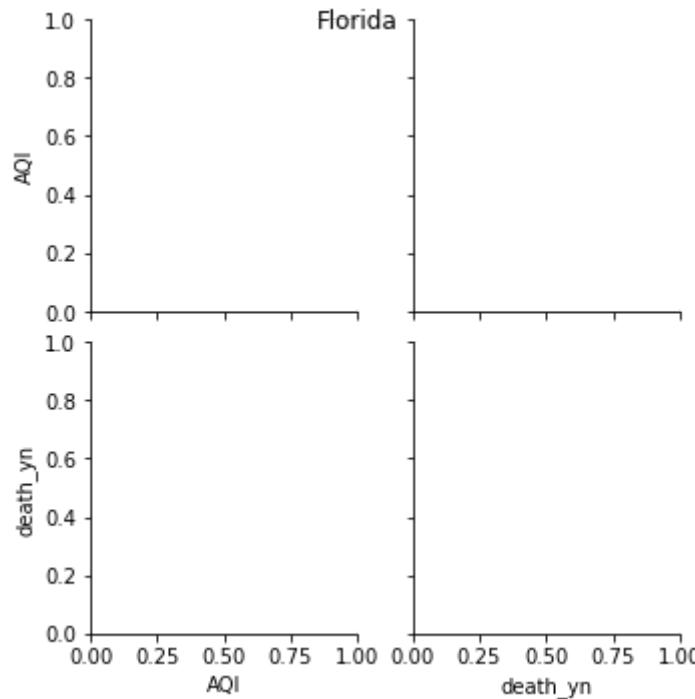
&lt;Figure size 1080x504 with 0 Axes&gt;



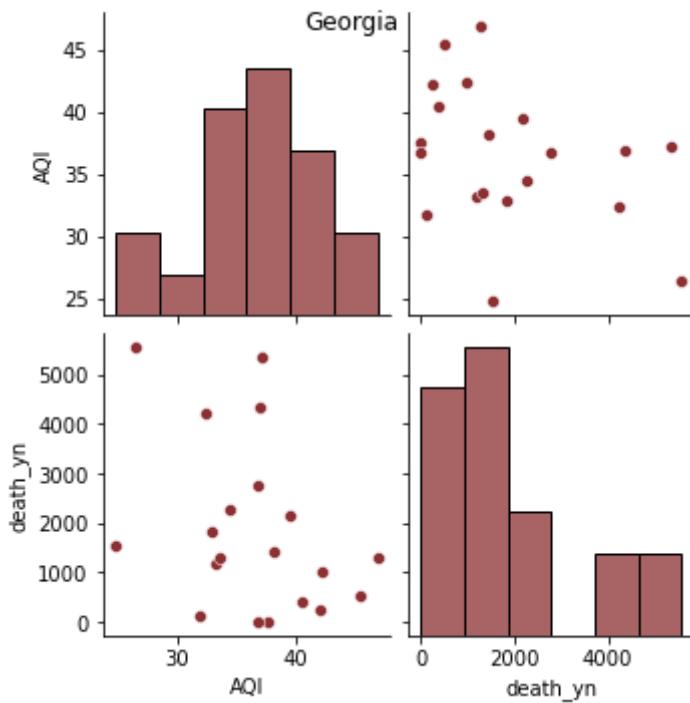
&lt;Figure size 1080x504 with 0 Axes&gt;



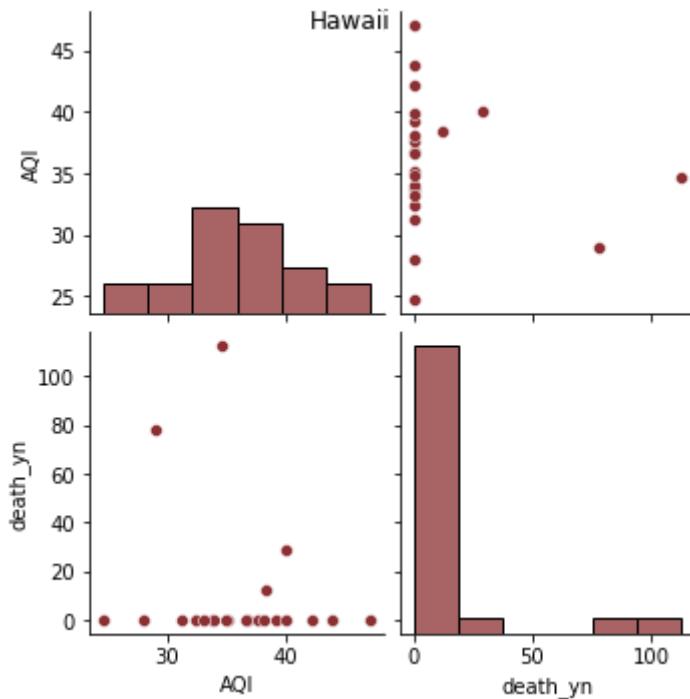
&lt;Figure size 1080x504 with 0 Axes&gt;



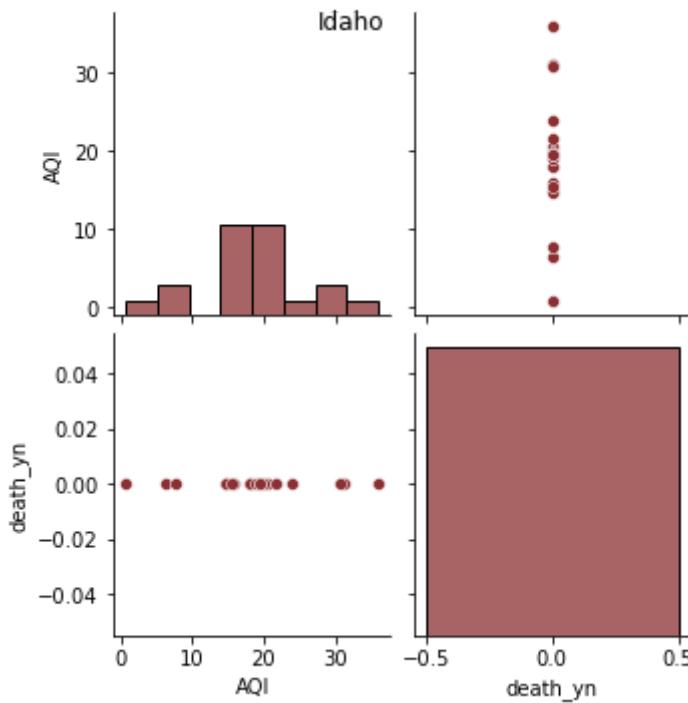
&lt;Figure size 1080x504 with 0 Axes&gt;



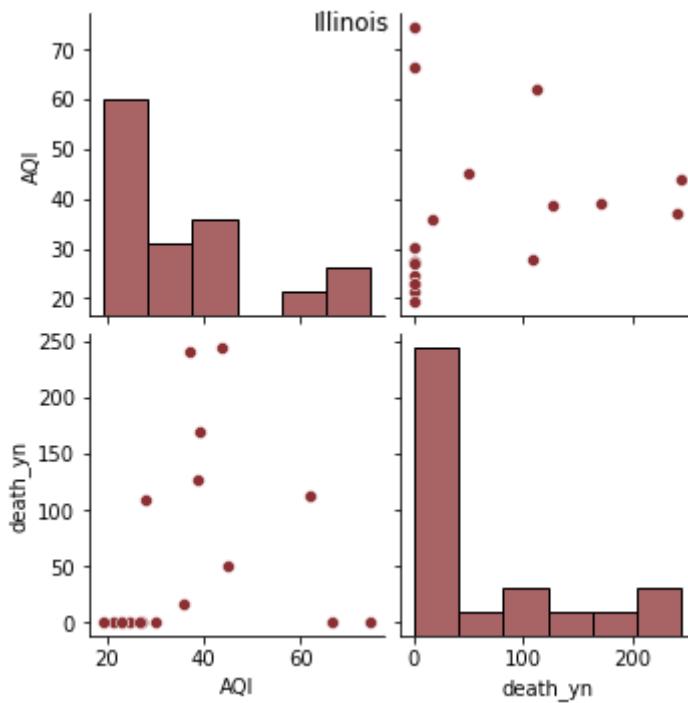
&lt;Figure size 1080x504 with 0 Axes&gt;



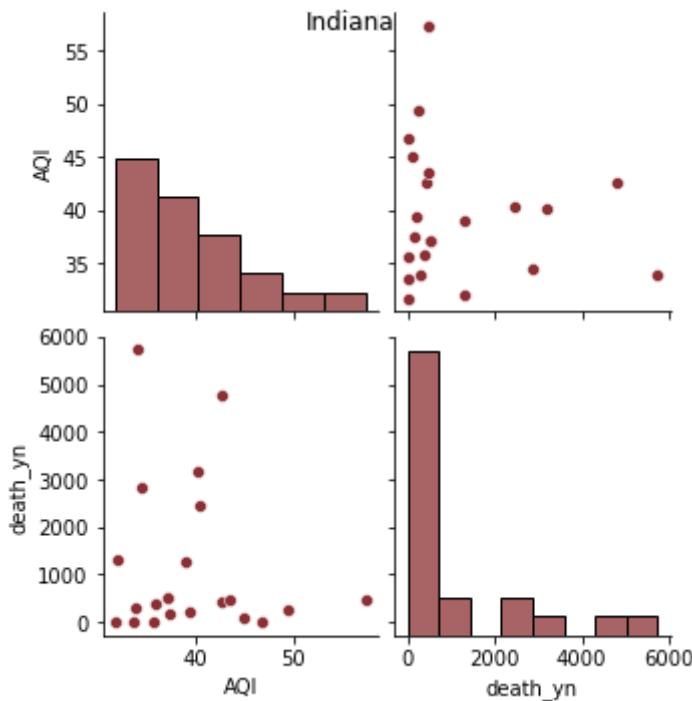
&lt;Figure size 1080x504 with 0 Axes&gt;



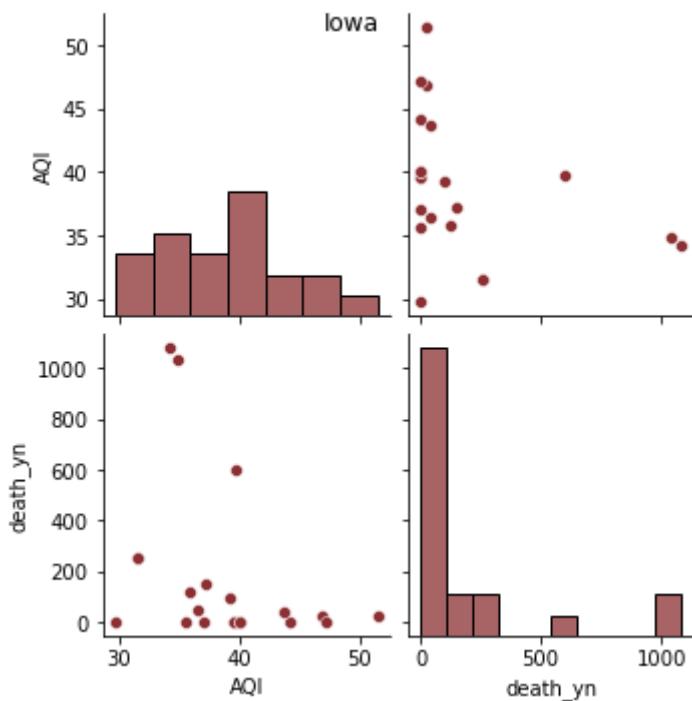
&lt;Figure size 1080x504 with 0 Axes&gt;



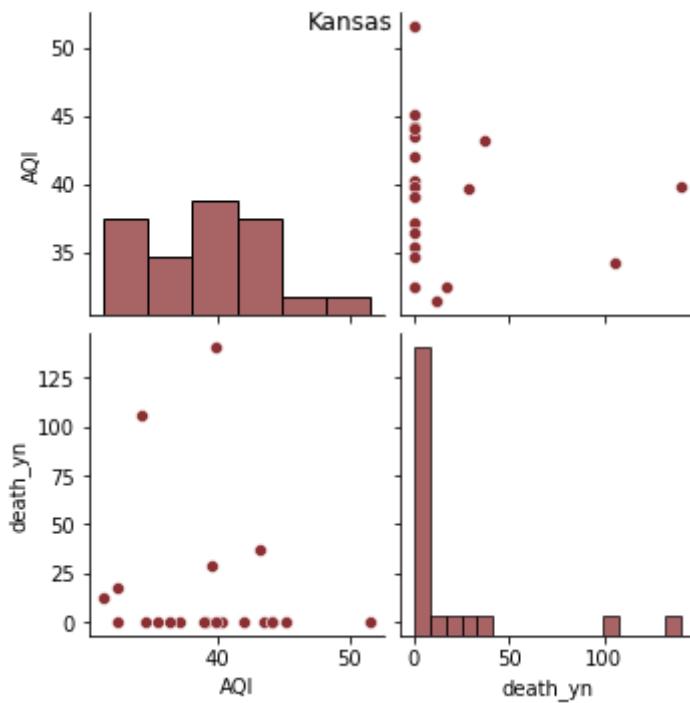
&lt;Figure size 1080x504 with 0 Axes&gt;



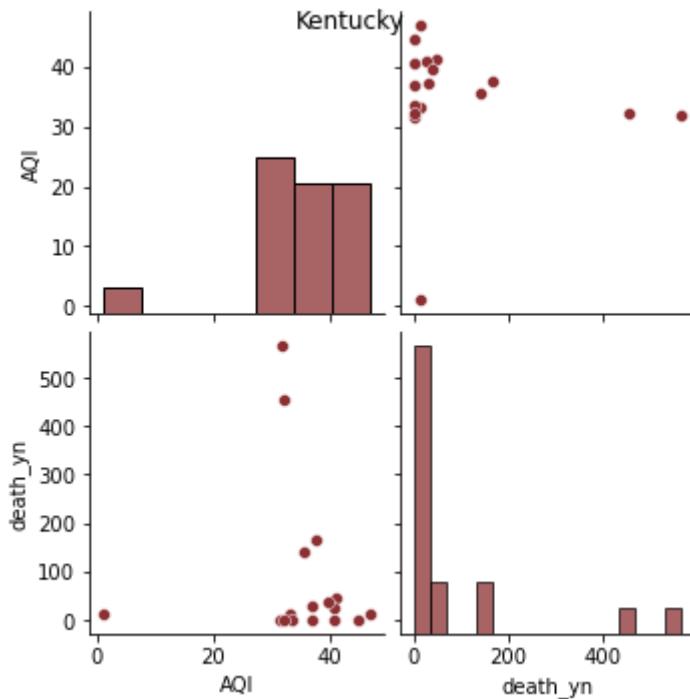
&lt;Figure size 1080x504 with 0 Axes&gt;



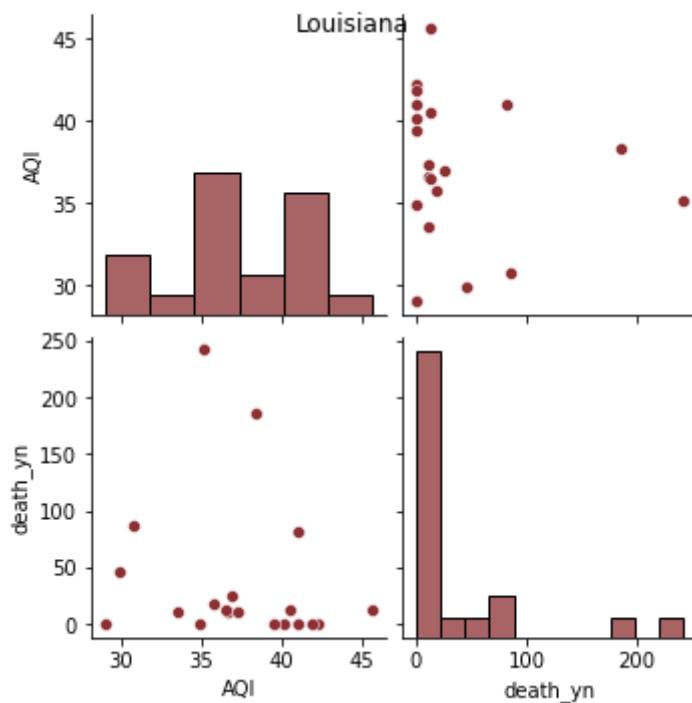
&lt;Figure size 1080x504 with 0 Axes&gt;



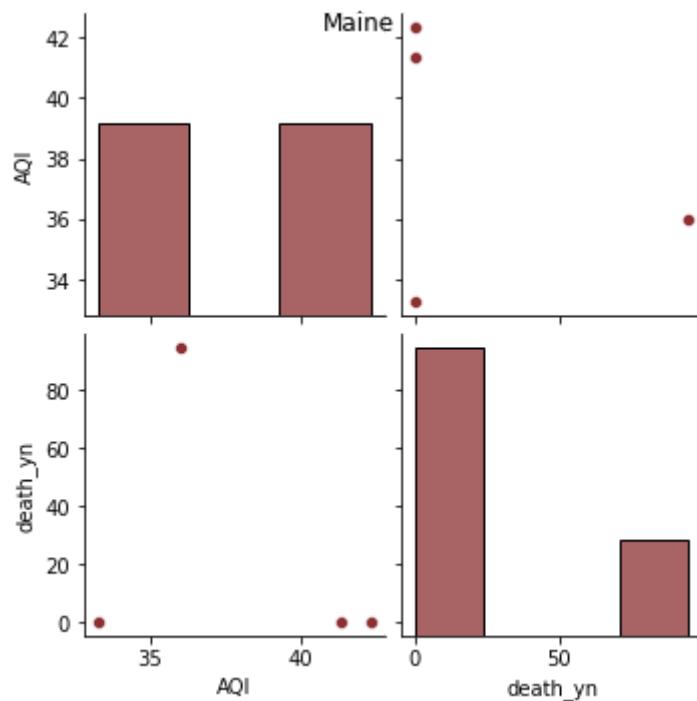
&lt;Figure size 1080x504 with 0 Axes&gt;



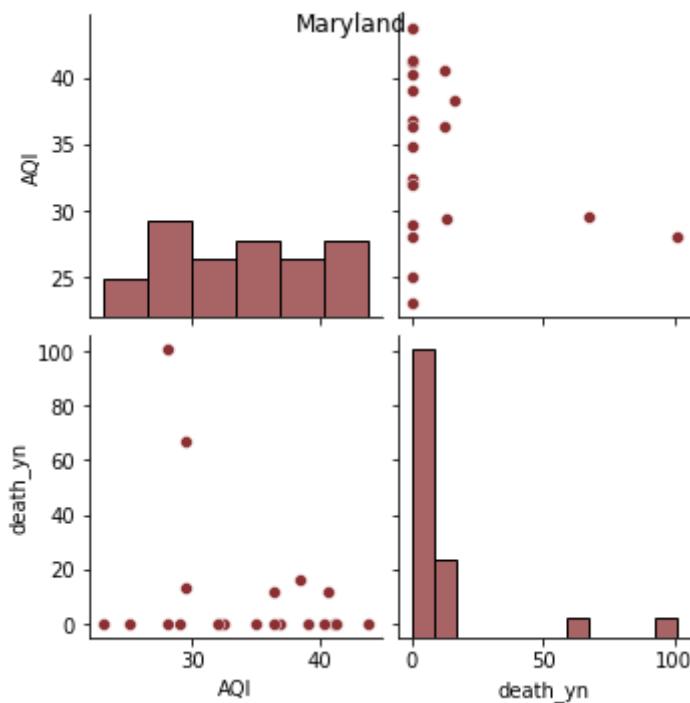
&lt;Figure size 1080x504 with 0 Axes&gt;



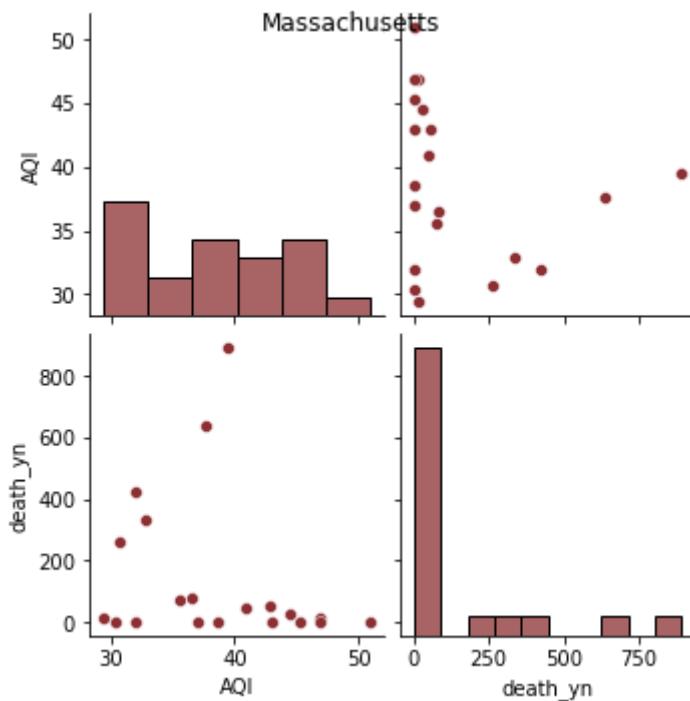
&lt;Figure size 1080x504 with 0 Axes&gt;



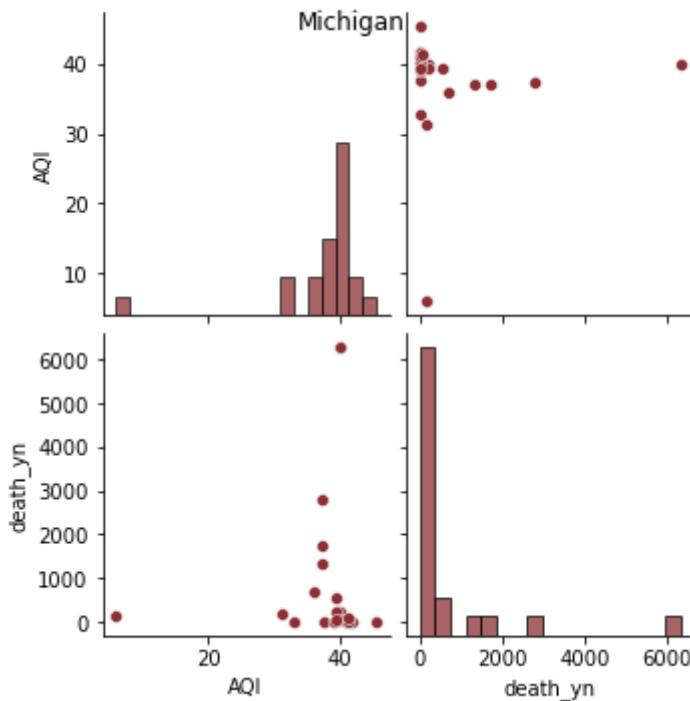
&lt;Figure size 1080x504 with 0 Axes&gt;



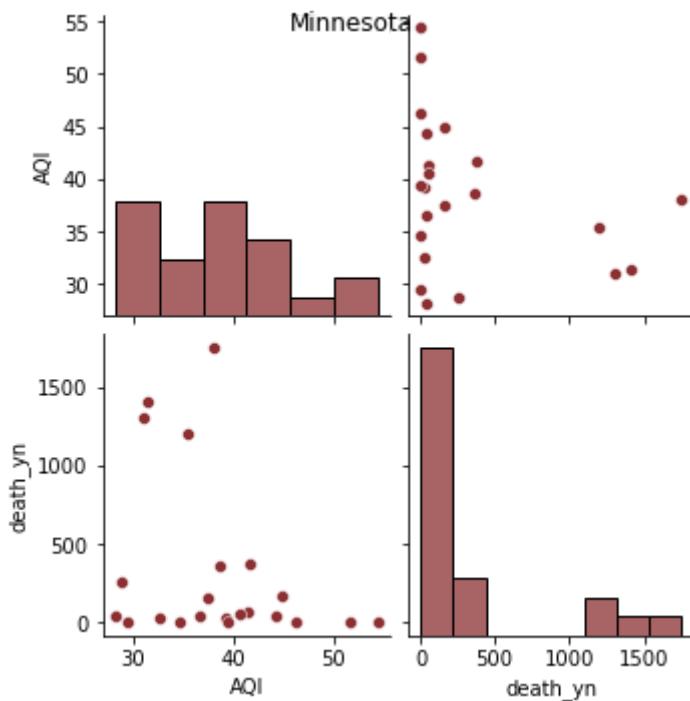
&lt;Figure size 1080x504 with 0 Axes&gt;



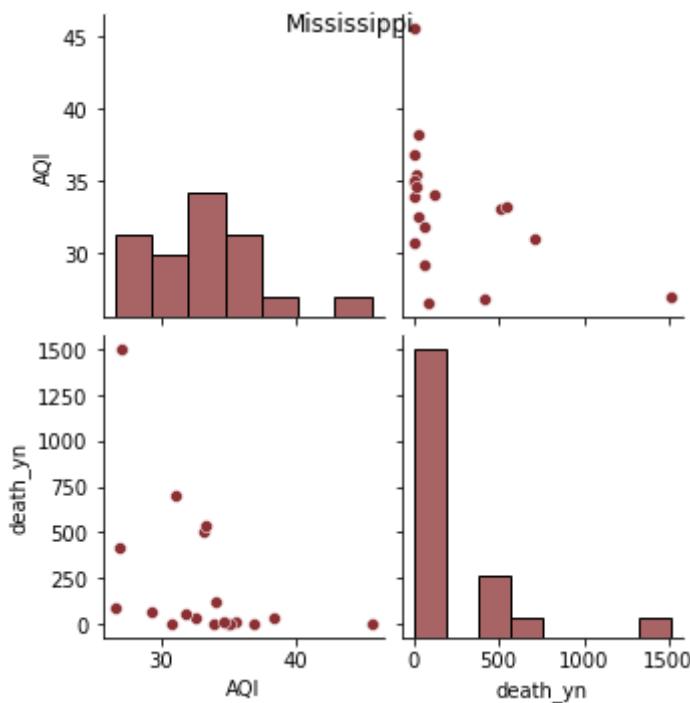
&lt;Figure size 1080x504 with 0 Axes&gt;



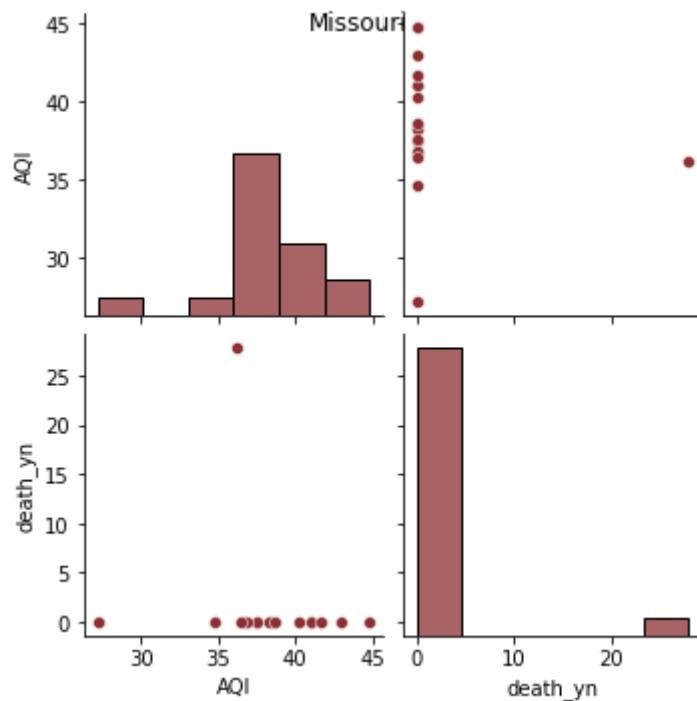
&lt;Figure size 1080x504 with 0 Axes&gt;



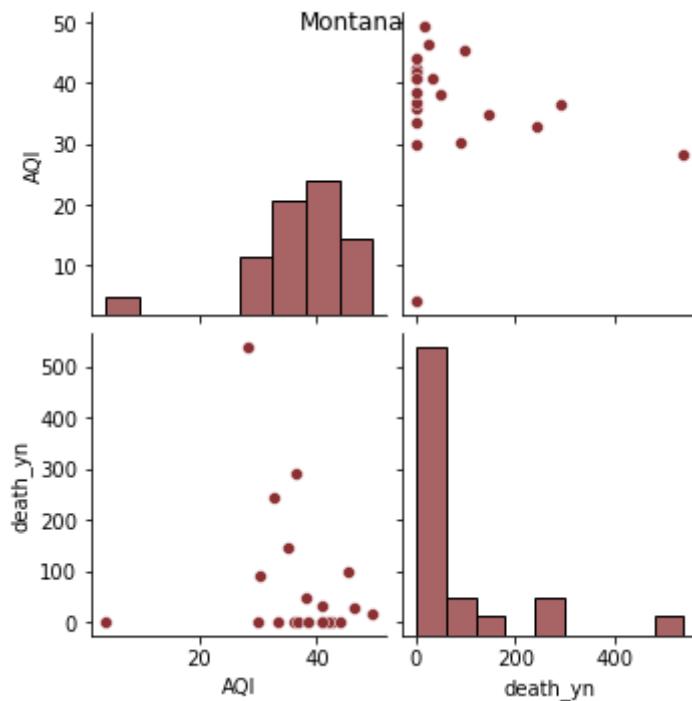
&lt;Figure size 1080x504 with 0 Axes&gt;



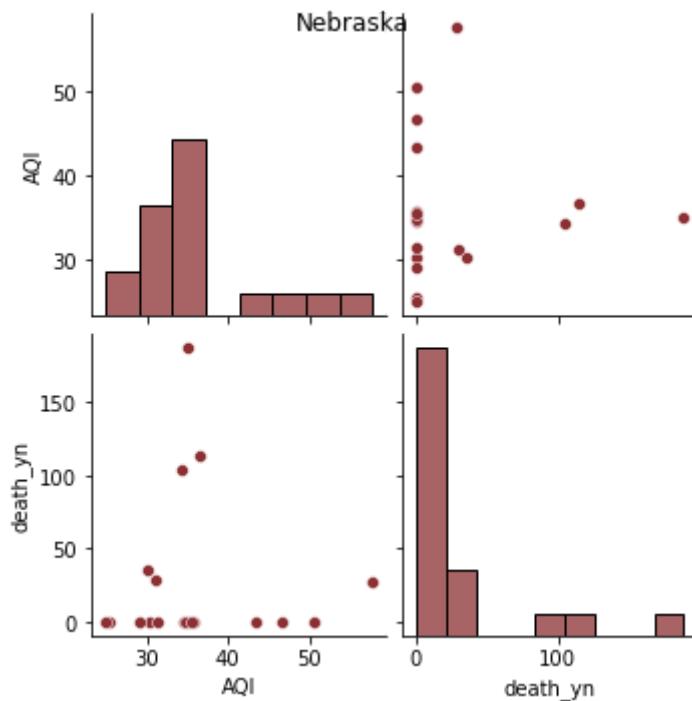
&lt;Figure size 1080x504 with 0 Axes&gt;



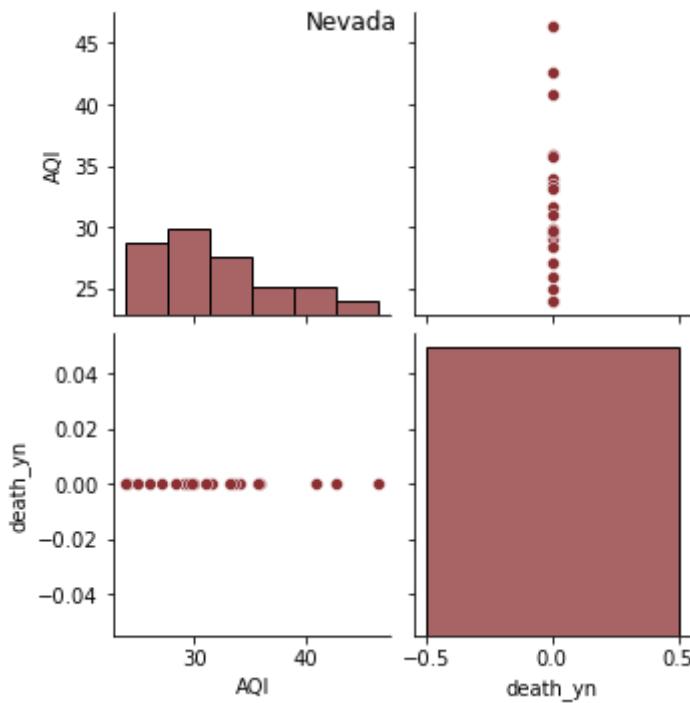
&lt;Figure size 1080x504 with 0 Axes&gt;



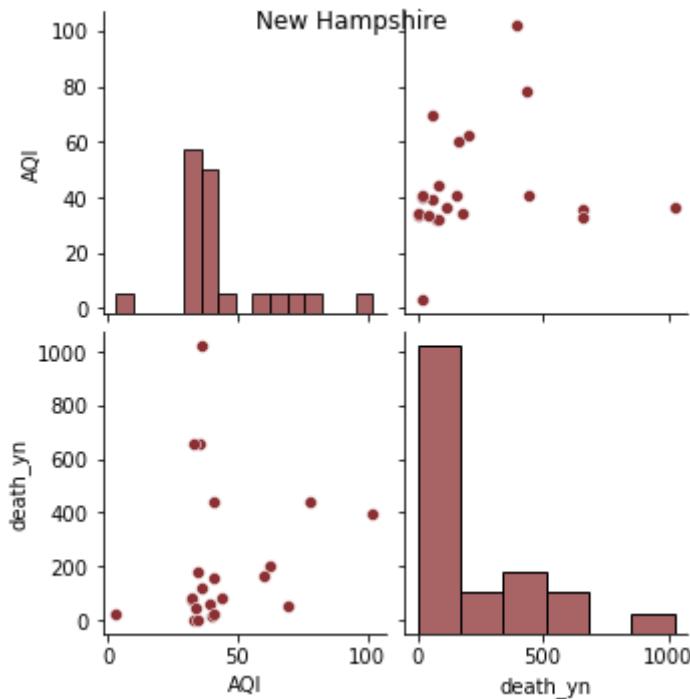
&lt;Figure size 1080x504 with 0 Axes&gt;



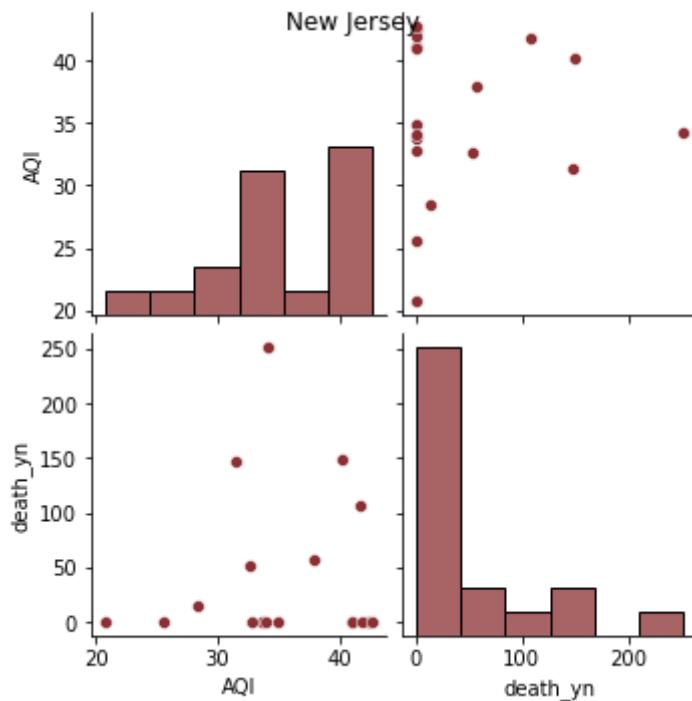
&lt;Figure size 1080x504 with 0 Axes&gt;



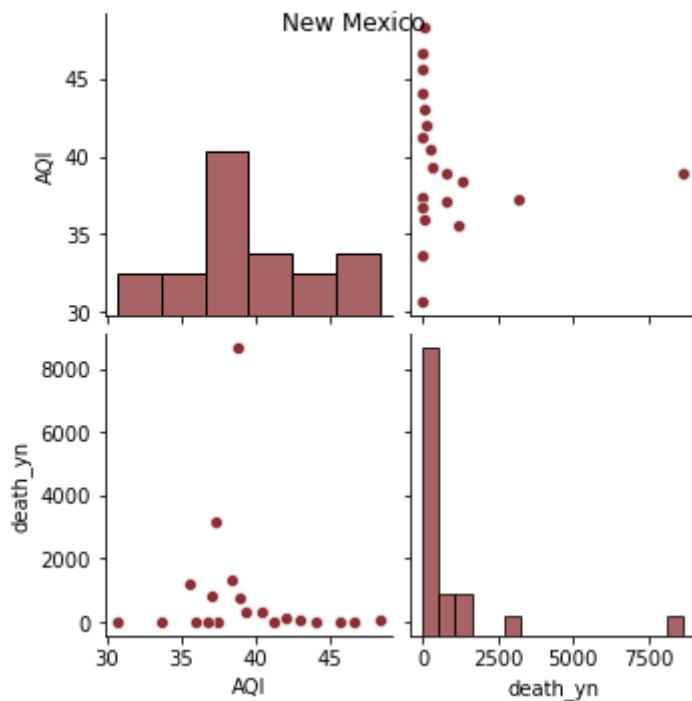
&lt;Figure size 1080x504 with 0 Axes&gt;



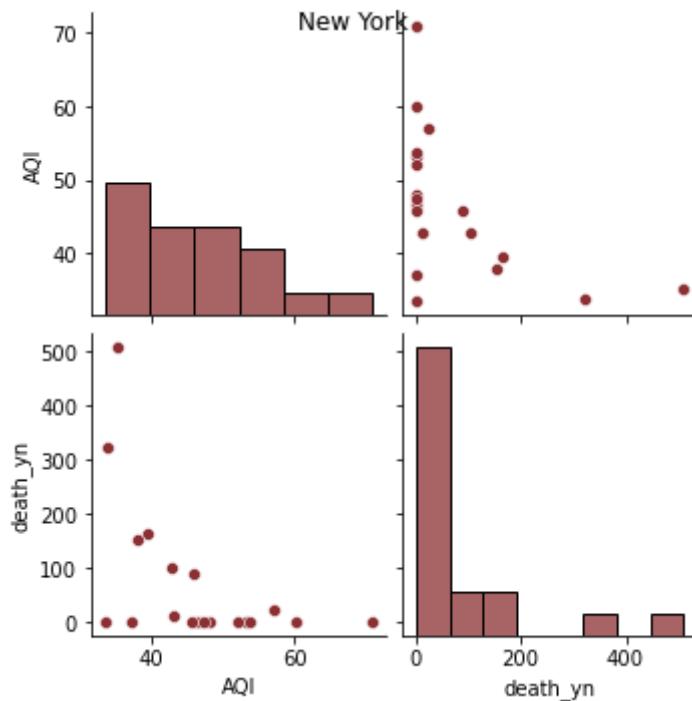
&lt;Figure size 1080x504 with 0 Axes&gt;



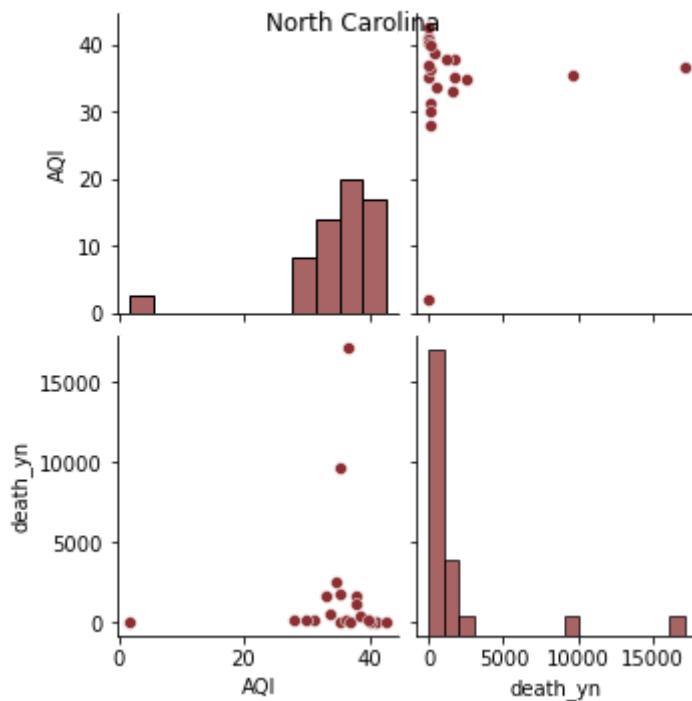
&lt;Figure size 1080x504 with 0 Axes&gt;



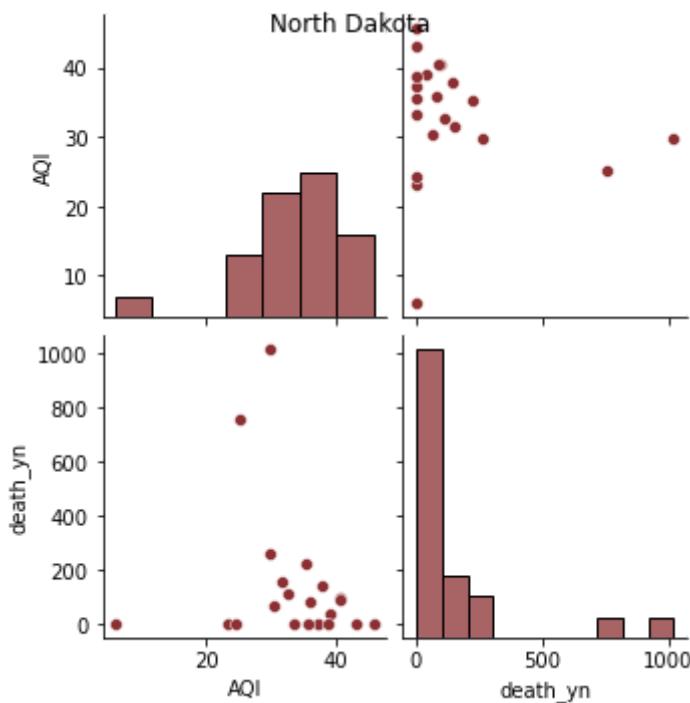
&lt;Figure size 1080x504 with 0 Axes&gt;



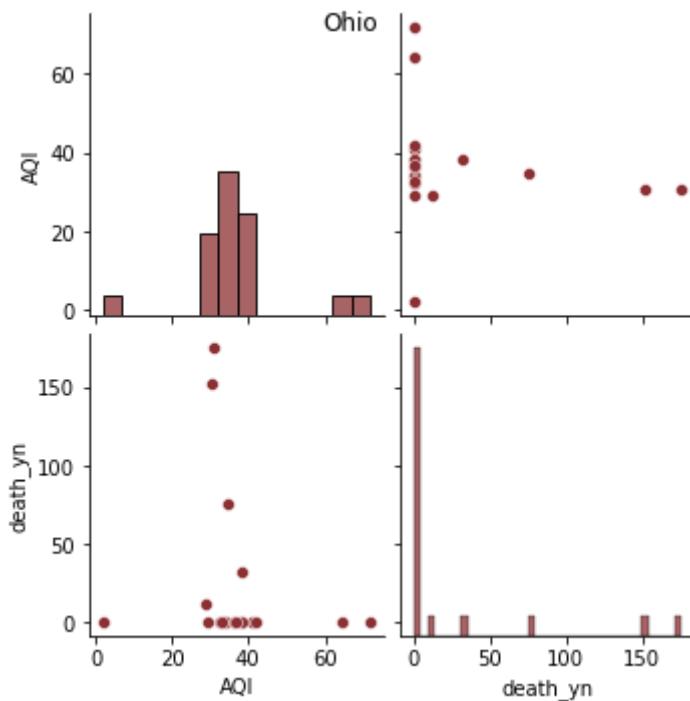
&lt;Figure size 1080x504 with 0 Axes&gt;



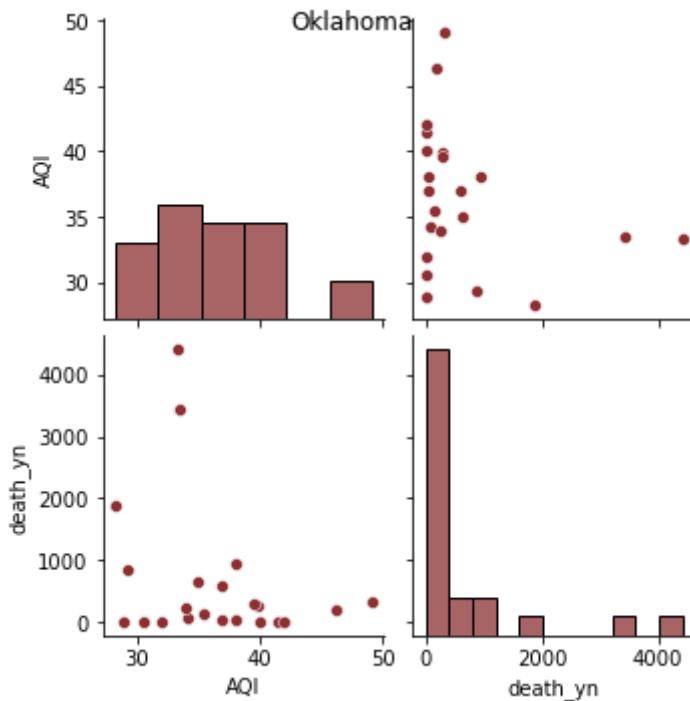
&lt;Figure size 1080x504 with 0 Axes&gt;



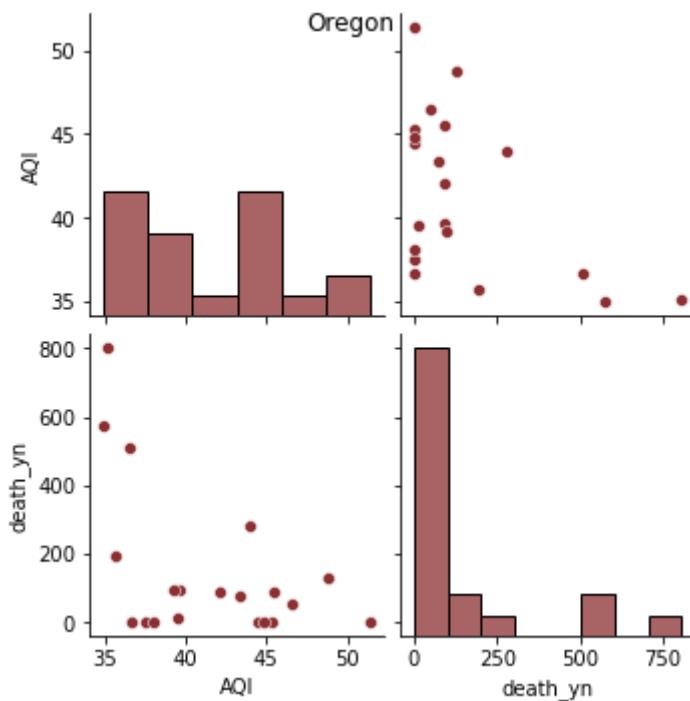
&lt;Figure size 1080x504 with 0 Axes&gt;



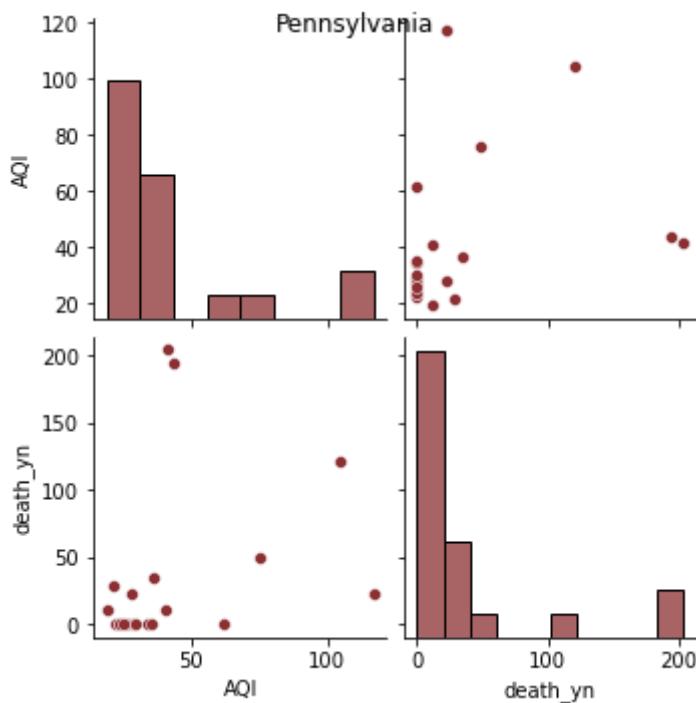
&lt;Figure size 1080x504 with 0 Axes&gt;



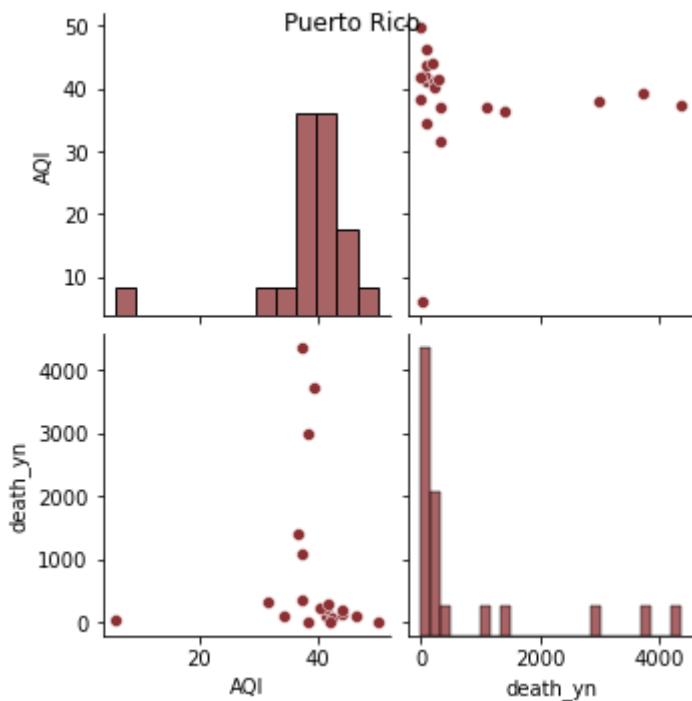
&lt;Figure size 1080x504 with 0 Axes&gt;



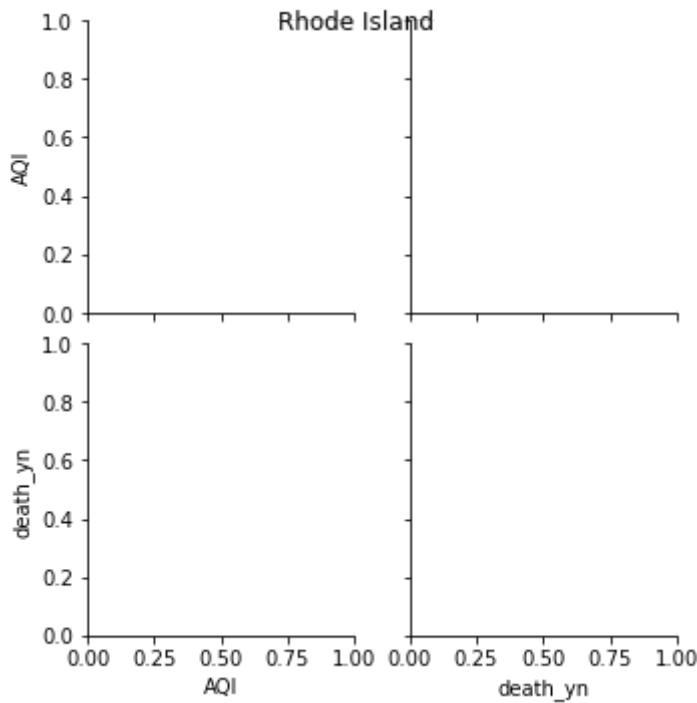
&lt;Figure size 1080x504 with 0 Axes&gt;



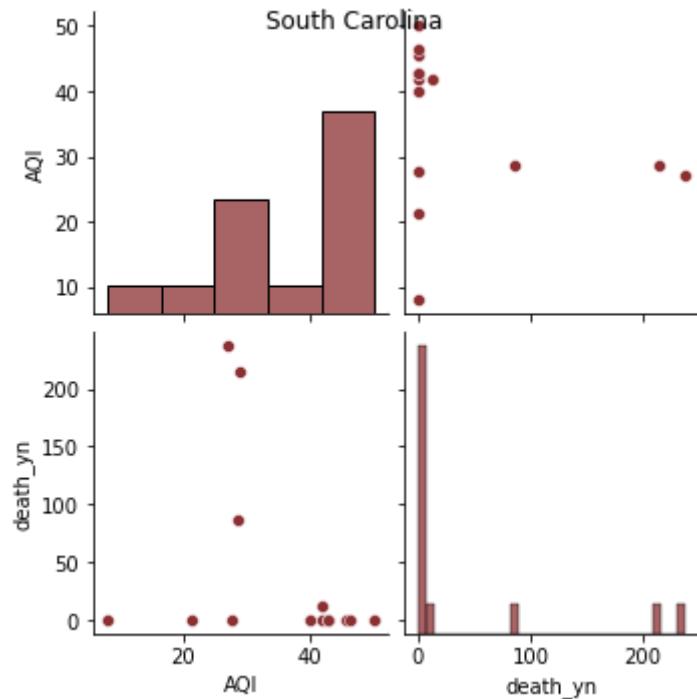
&lt;Figure size 1080x504 with 0 Axes&gt;



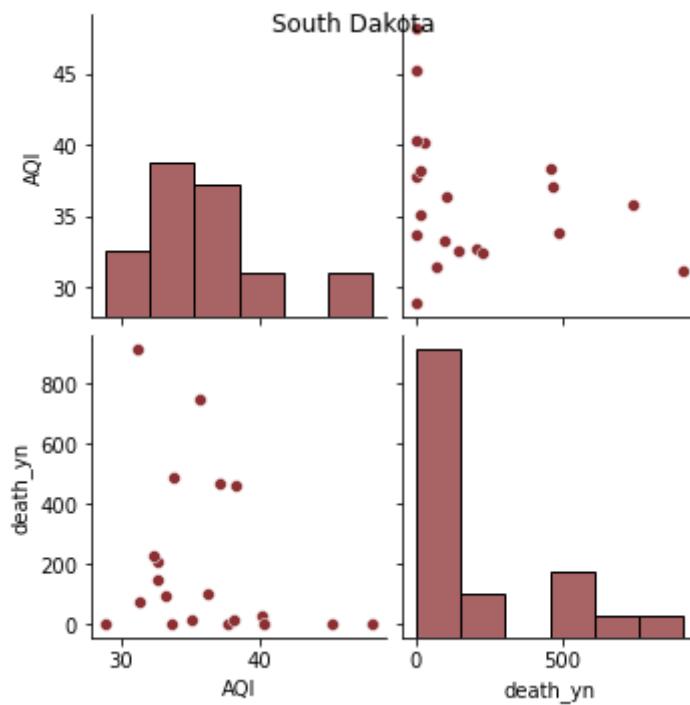
&lt;Figure size 1080x504 with 0 Axes&gt;



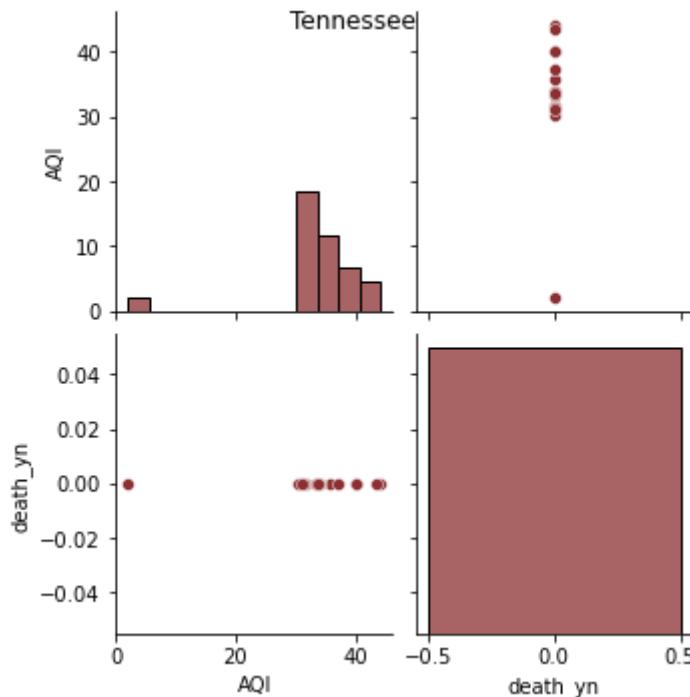
&lt;Figure size 1080x504 with 0 Axes&gt;



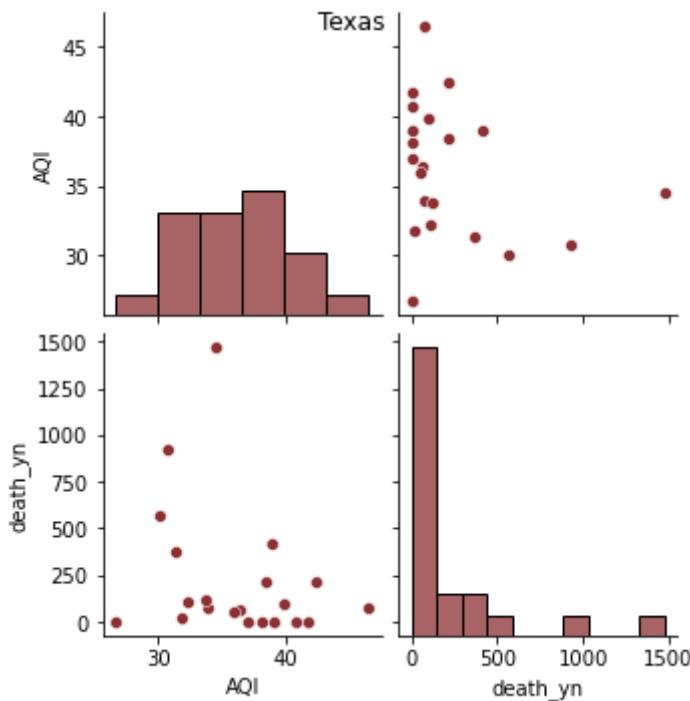
&lt;Figure size 1080x504 with 0 Axes&gt;



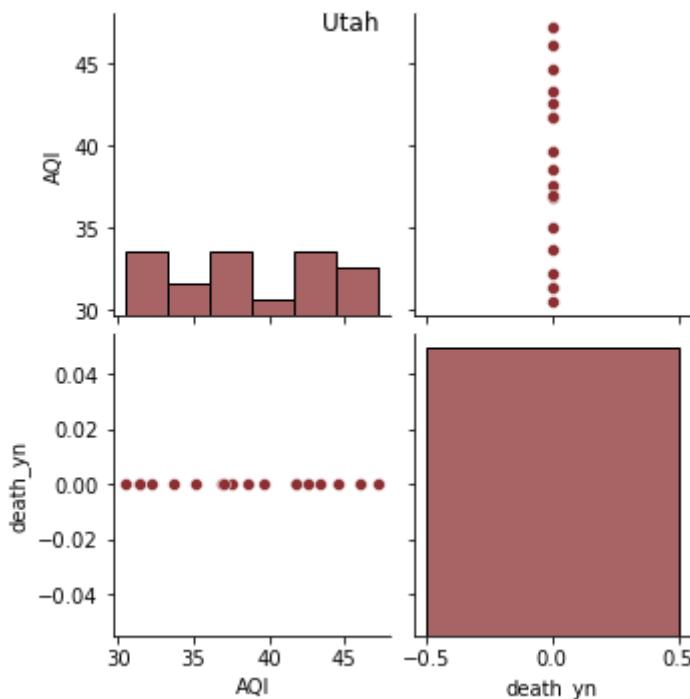
&lt;Figure size 1080x504 with 0 Axes&gt;



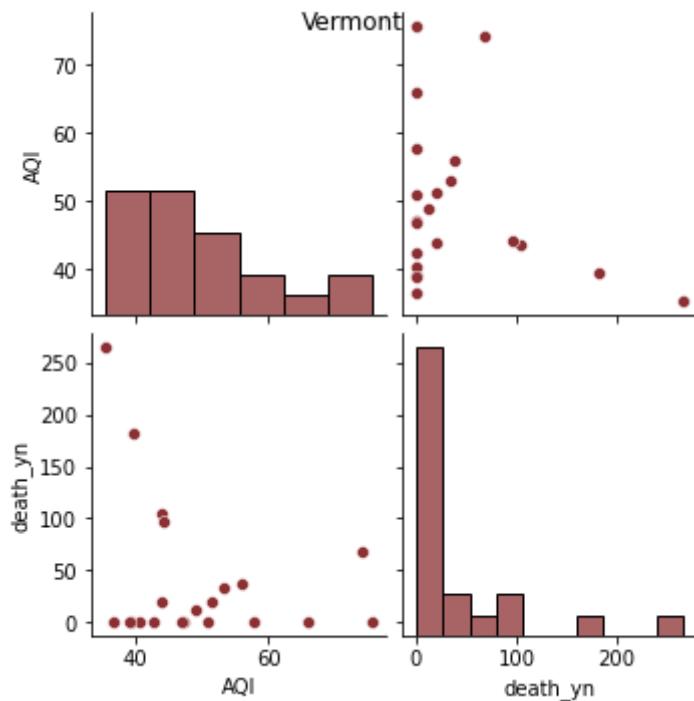
&lt;Figure size 1080x504 with 0 Axes&gt;



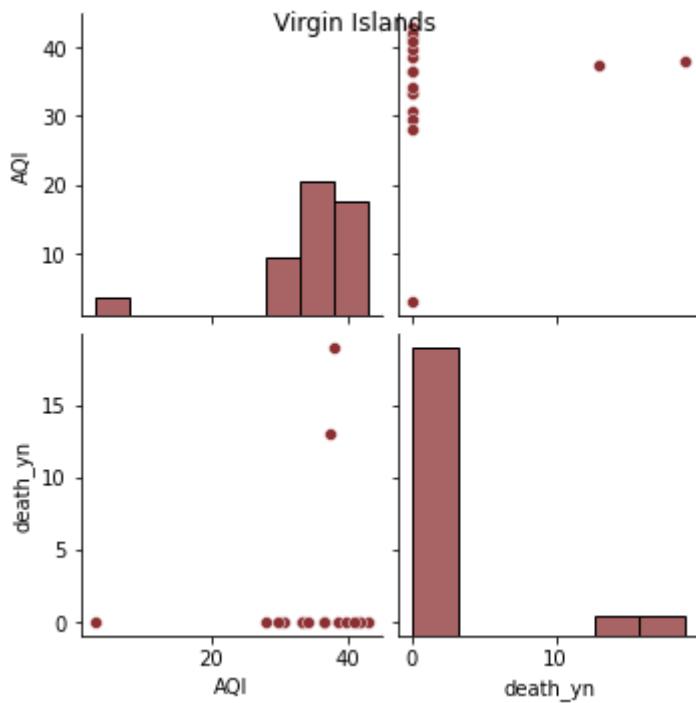
&lt;Figure size 1080x504 with 0 Axes&gt;



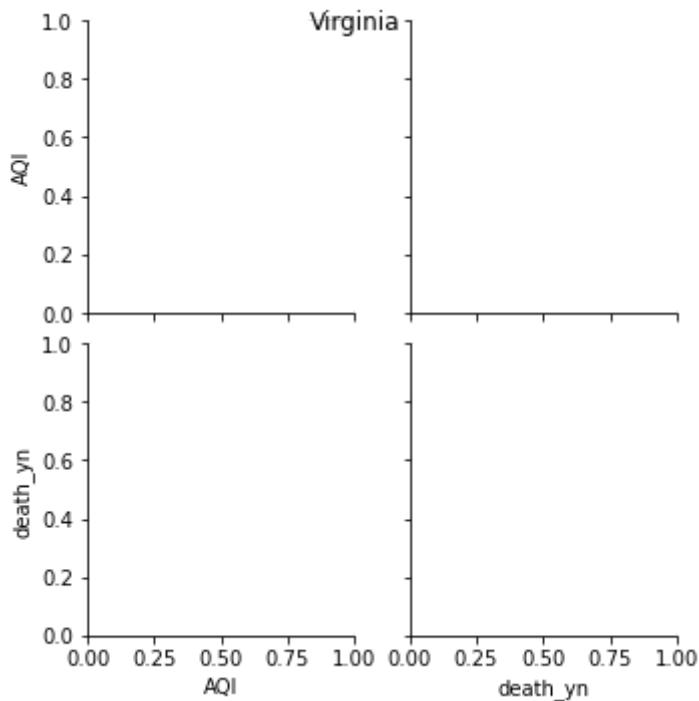
&lt;Figure size 1080x504 with 0 Axes&gt;



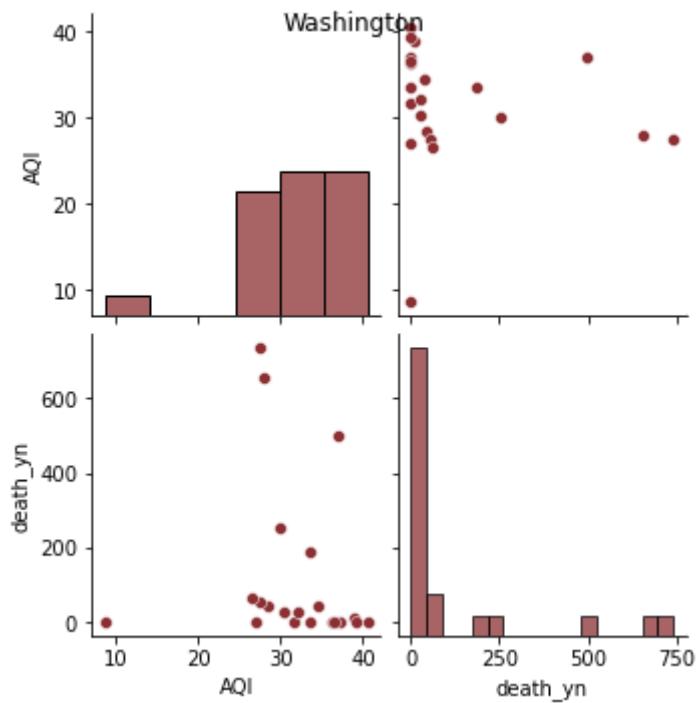
&lt;Figure size 1080x504 with 0 Axes&gt;



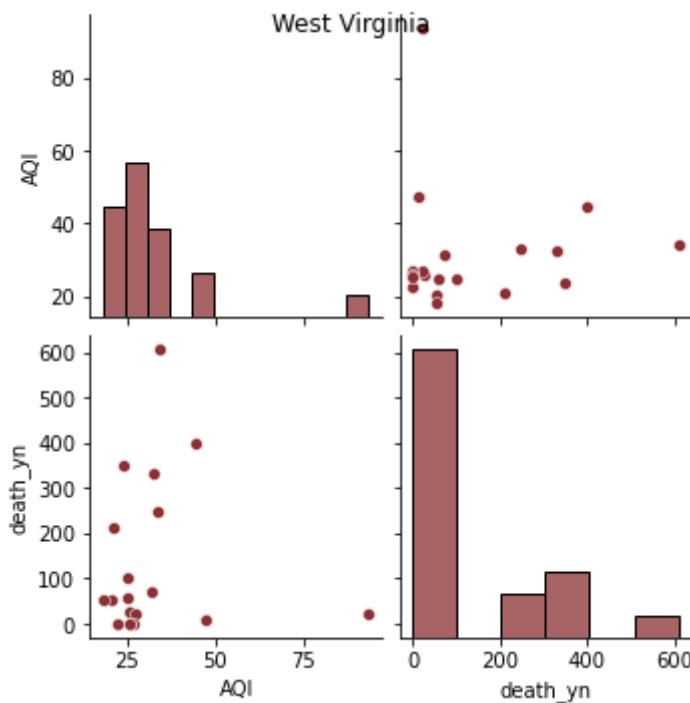
&lt;Figure size 1080x504 with 0 Axes&gt;



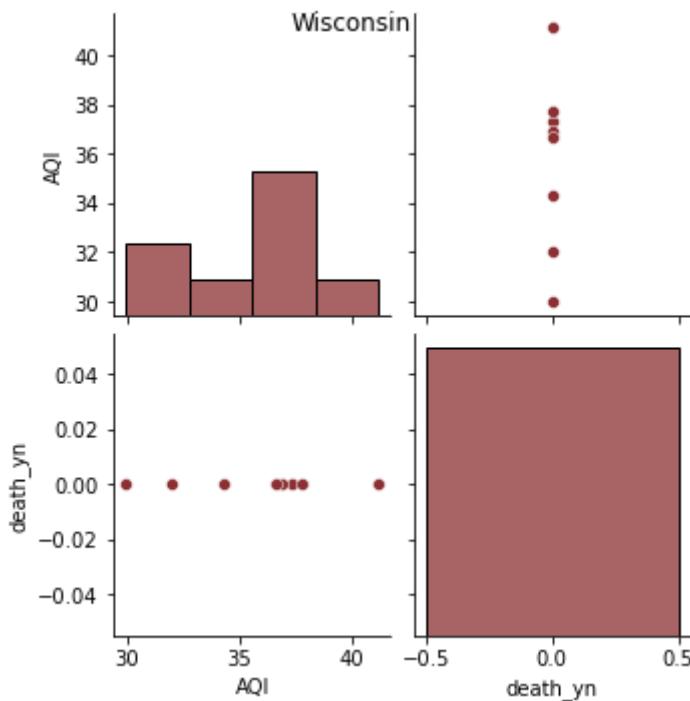
<Figure size 1080x504 with 0 Axes>



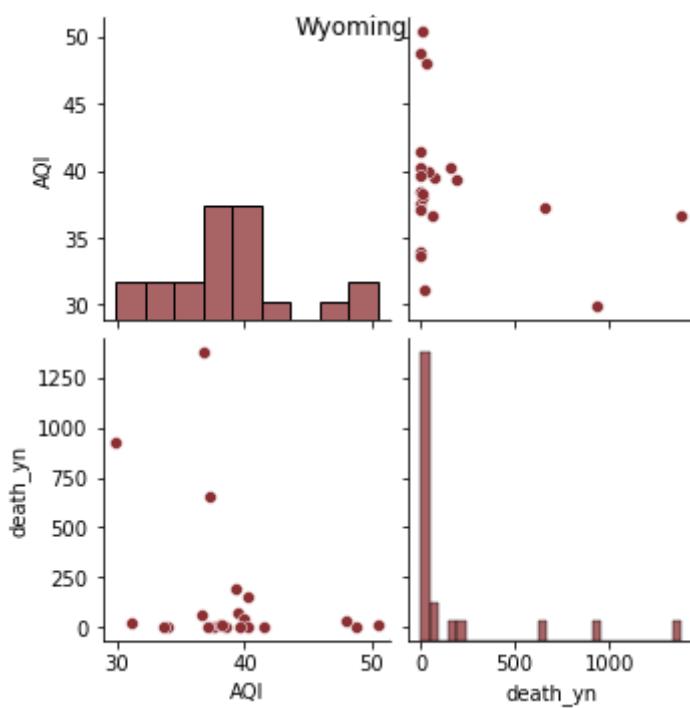
<Figure size 1080x504 with 0 Axes>



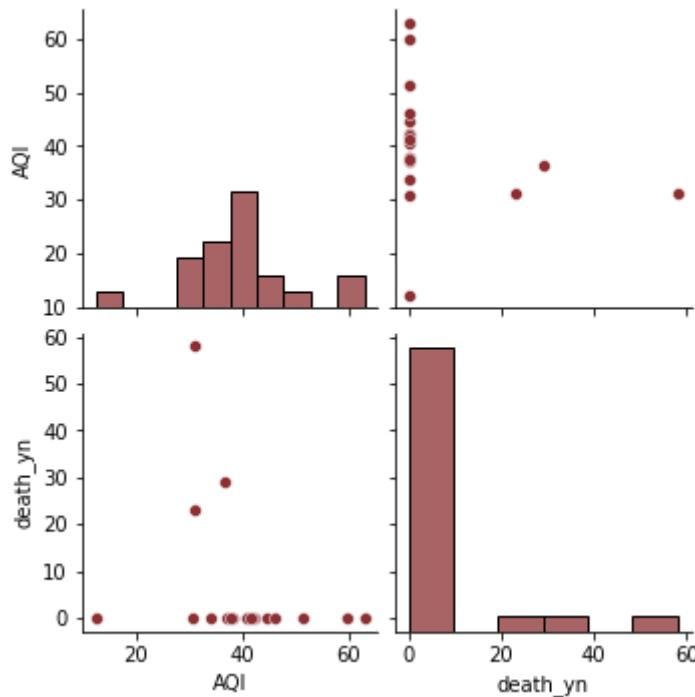
&lt;Figure size 1080x504 with 0 Axes&gt;



&lt;Figure size 1080x504 with 0 Axes&gt;



&lt;Figure size 1080x504 with 0 Axes&gt;



```
In [70]: ##Generating Correlation Heat Map AQI and Death_yn
```

```
i=0
j=24
for code in sc:

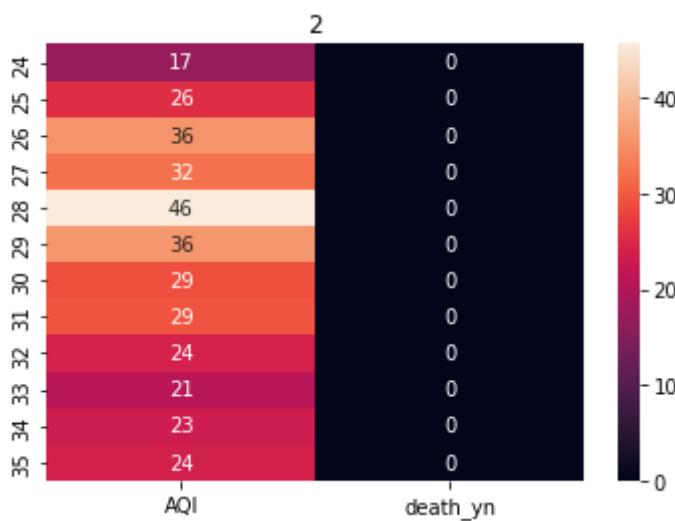
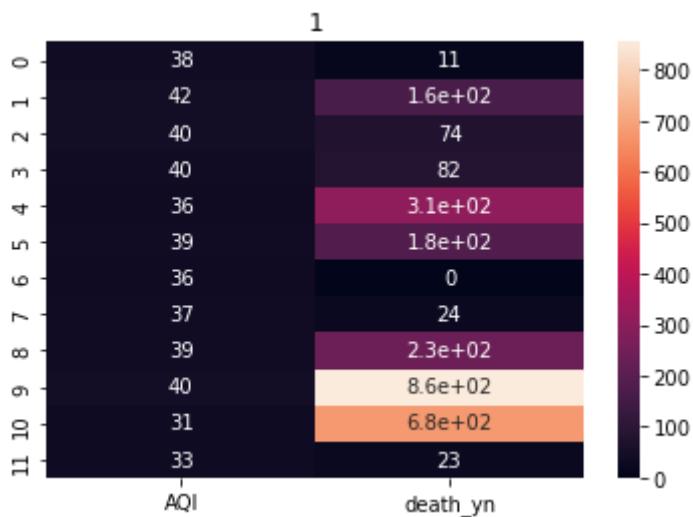
    df_plotter = pd.DataFrame()
    df_plotter=merged_df2.iloc[i:j,:]
    df_plotter = df_plotter.drop(['State_Name','year_month','State Code_x','ID',''
    df_plotter_2020=pd.DataFrame()
    df_plotter_2019=pd.DataFrame()
    df_plotter_2019=df_plotter.iloc[:12,:]
    df_plotter_2020=df_plotter.iloc[12:,:]
```

```

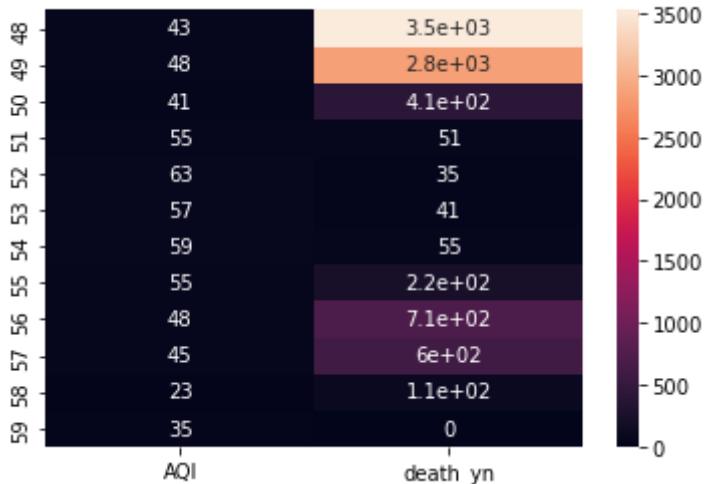
if not df_plotter.empty:
    try:
        ax = plt.axes()
        sns.heatmap(df_plotter_2019, annot=True, ax=ax)
        ax.set_title(code)
        plt.show()
        print('-----')
        p2 = sns.heatmap(df_plotter_2020, annot=True, ax=ax)
        ax.set_title(code)
        plt.show()
        print('-----')
    except ValueError:
        print("Warning: zero-size array")

```

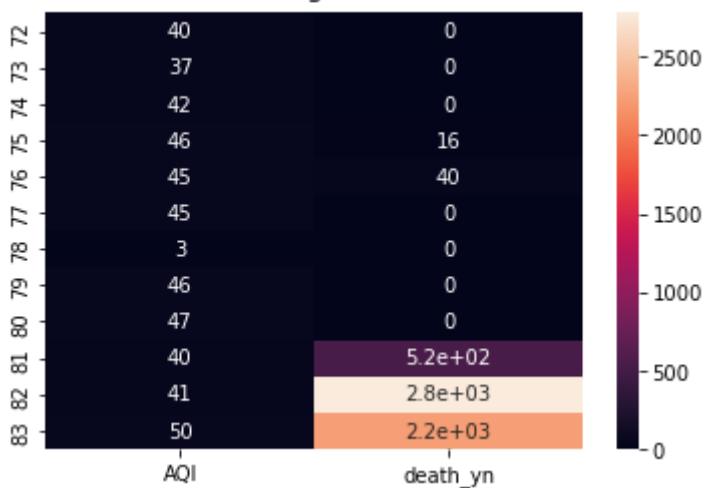
i+=24  
j+=24



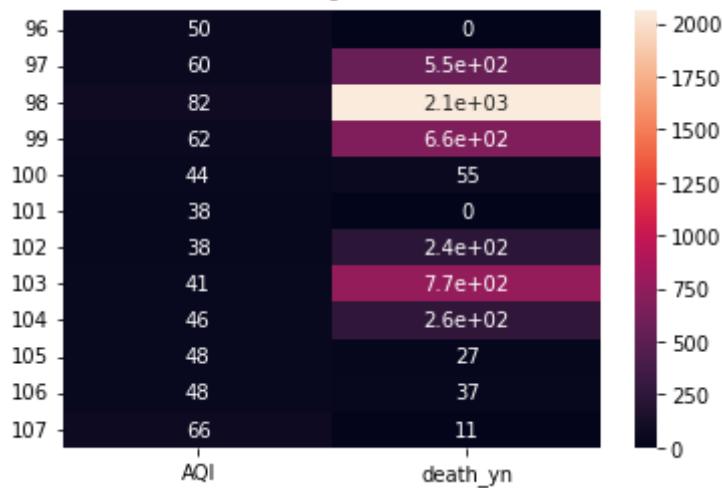
4



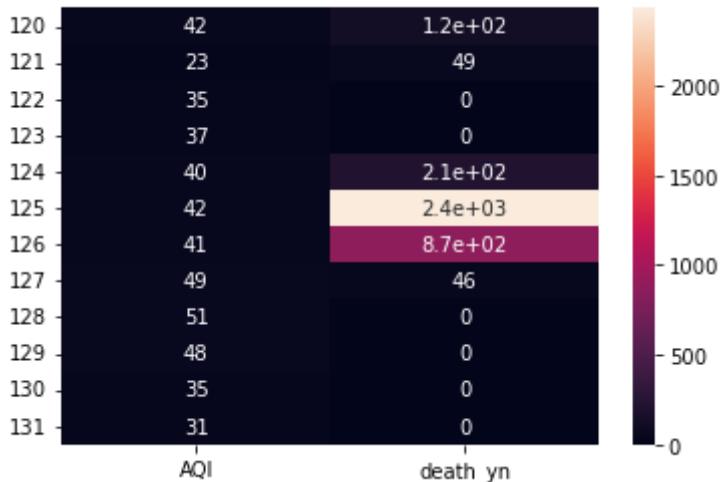
5



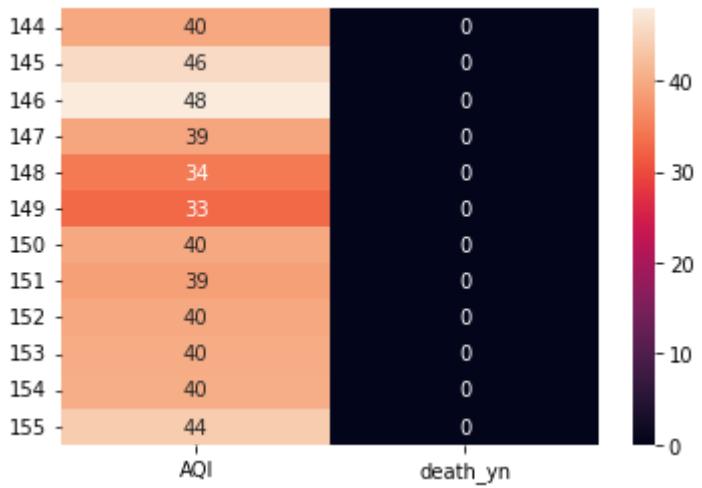
6



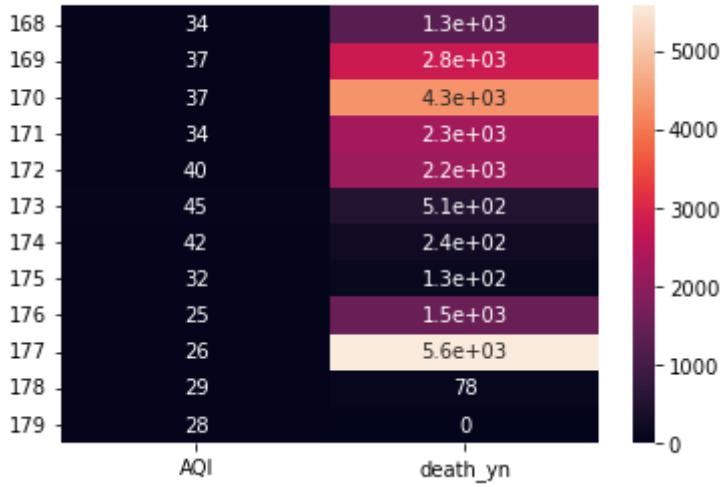
8

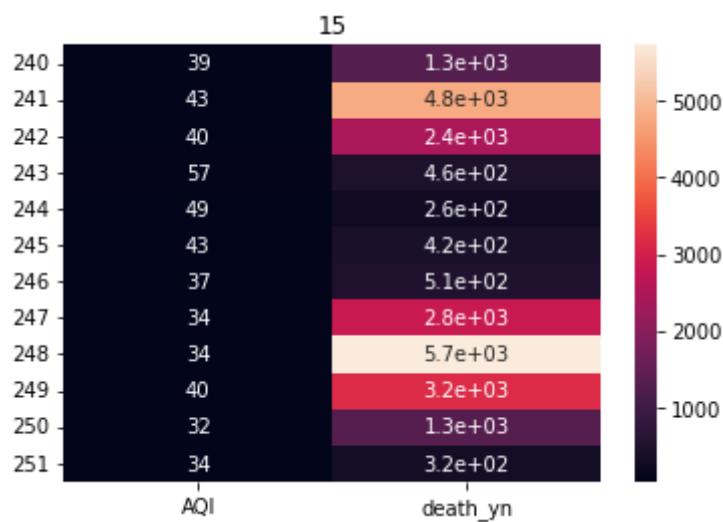
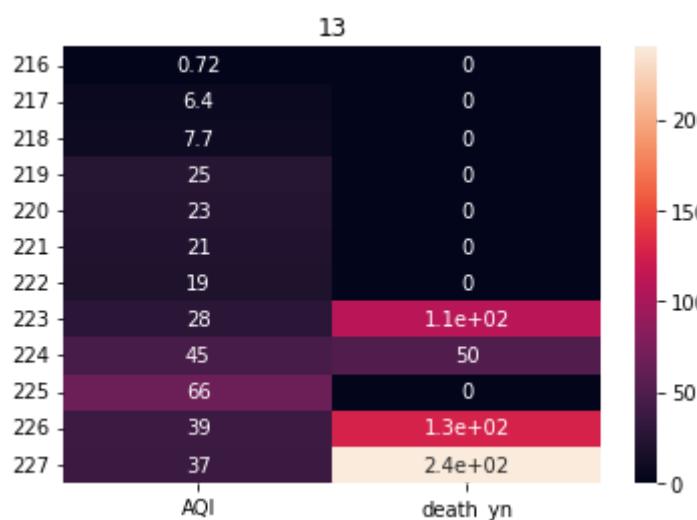
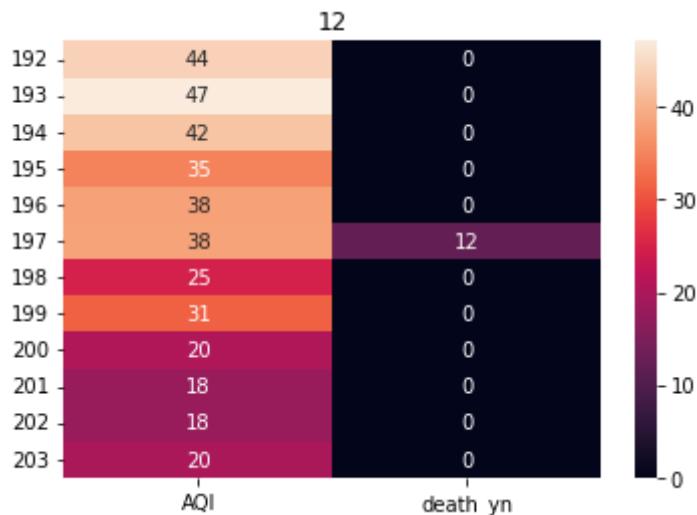


9

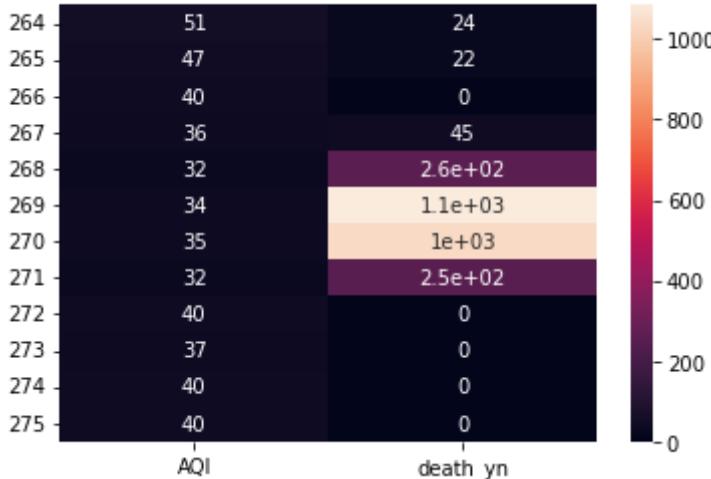


10

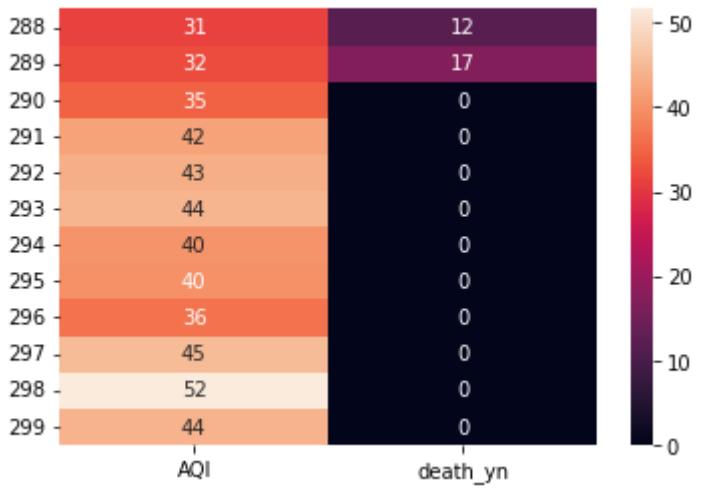




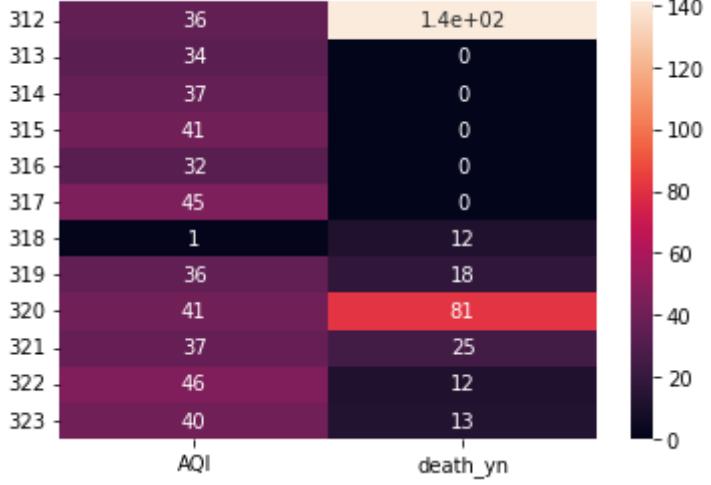
16



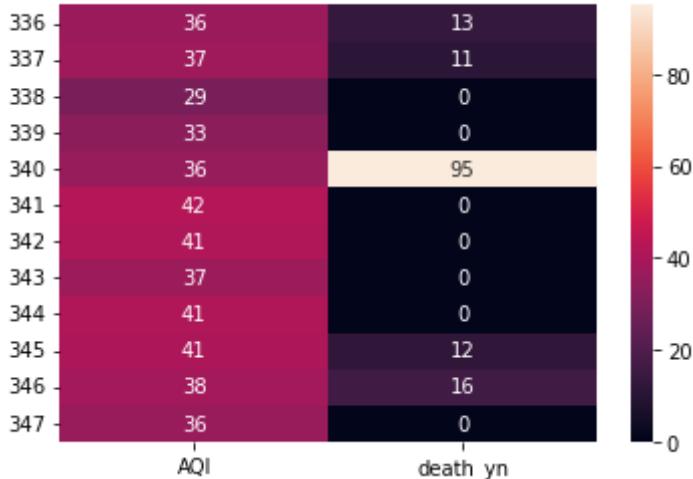
17



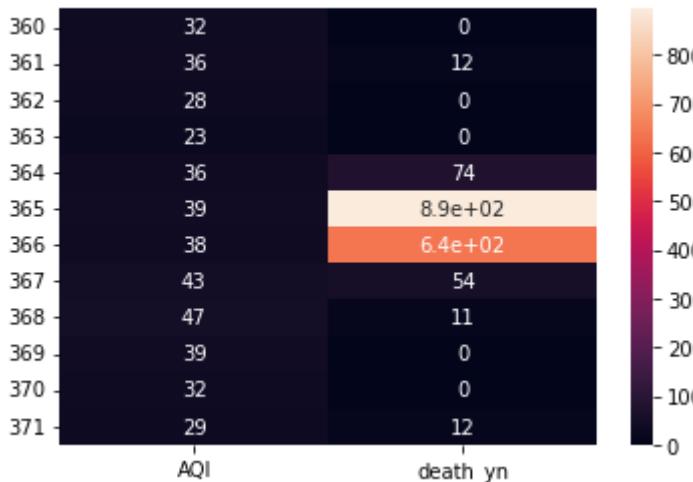
18



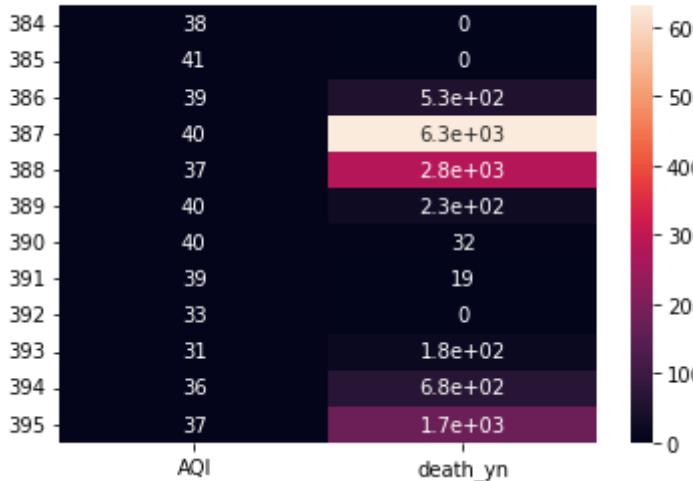
19



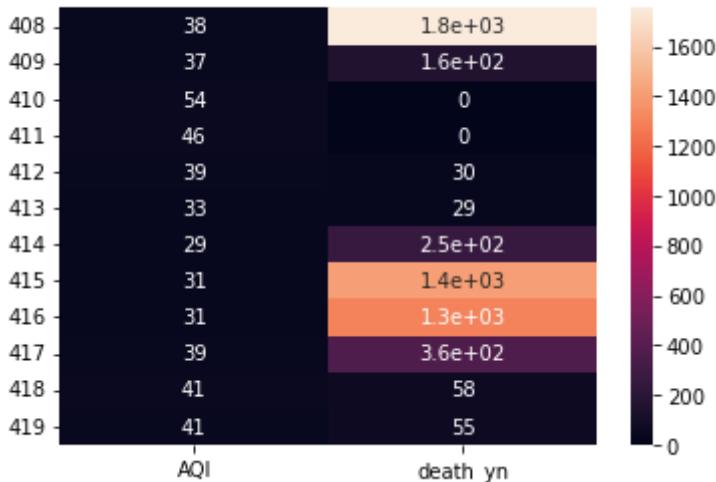
20



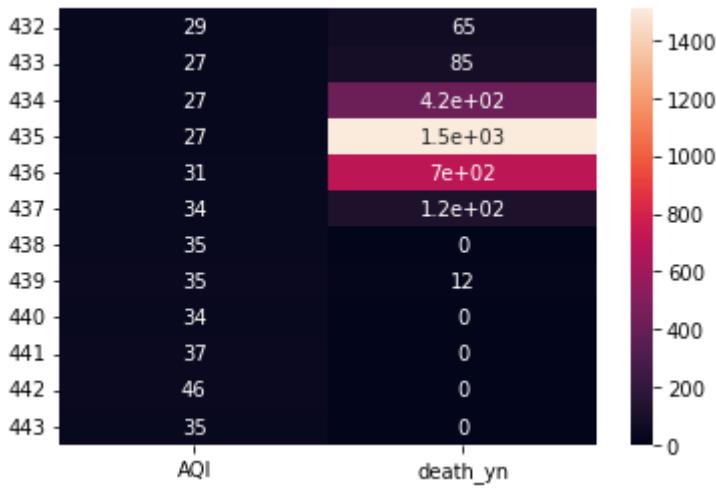
21



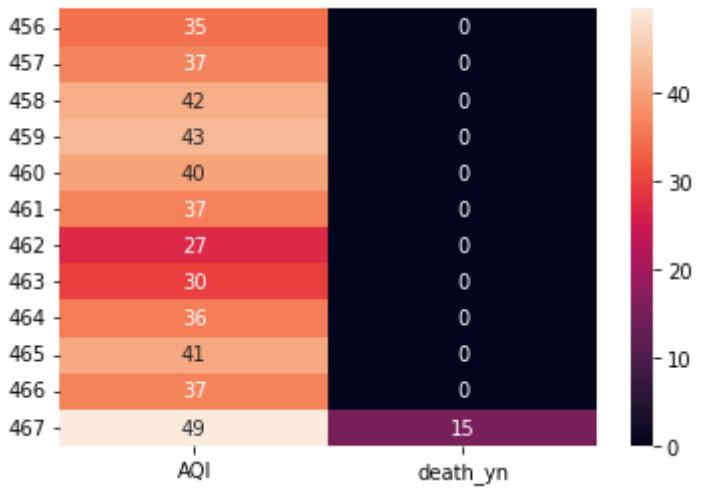
22



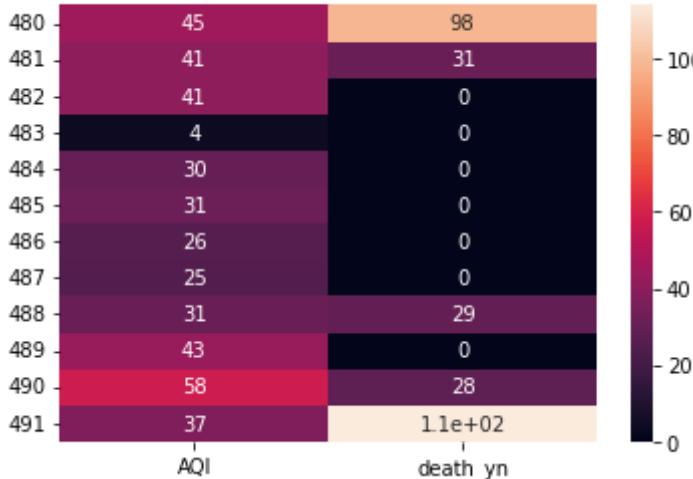
23



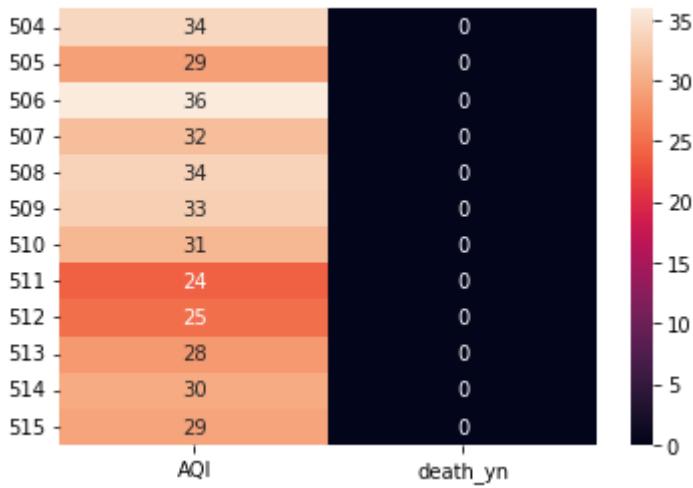
24



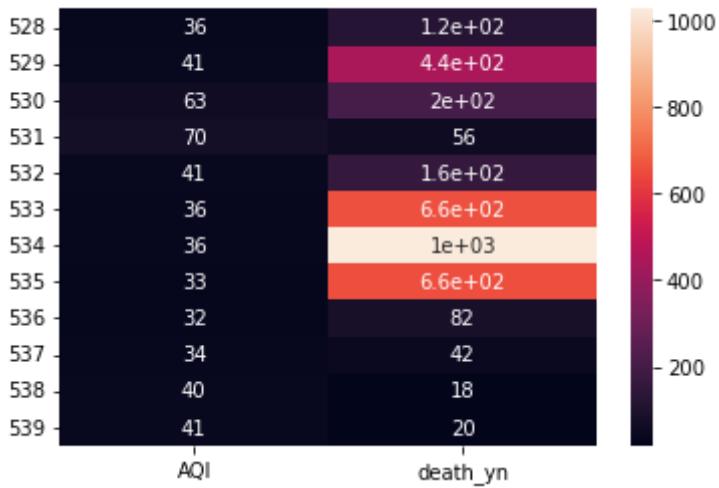
25



26



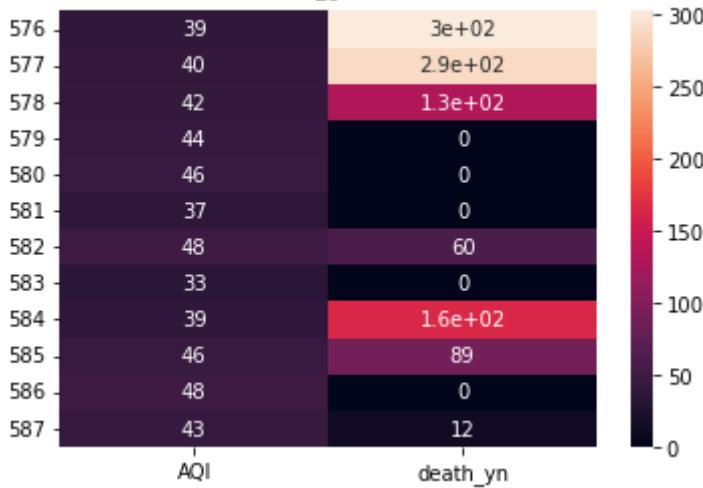
27



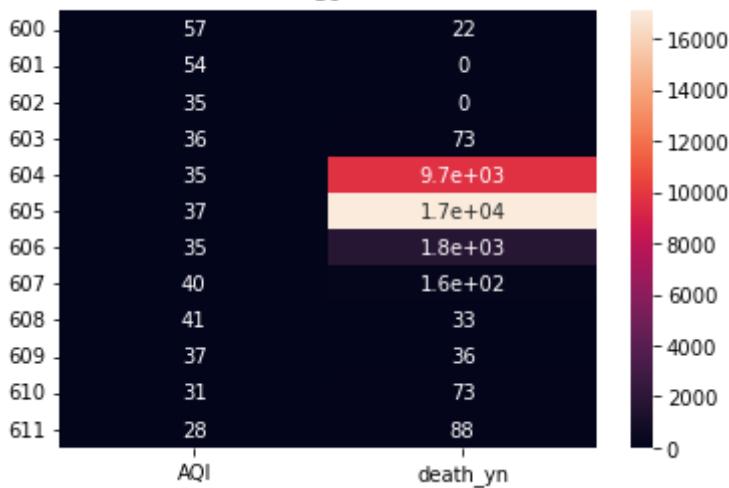
28



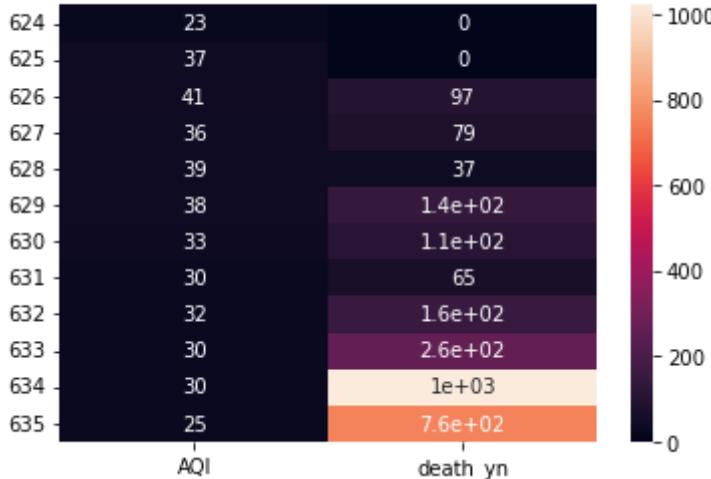
29



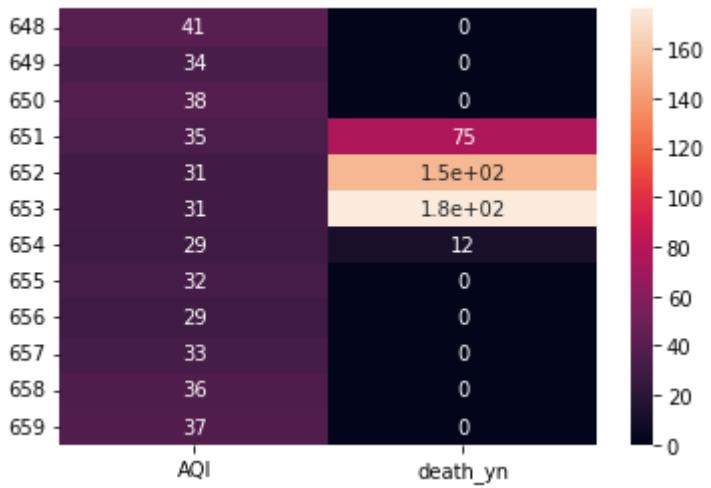
30



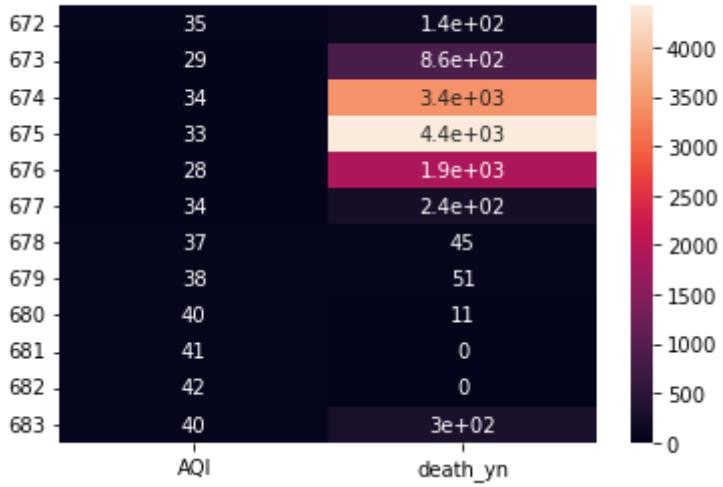
31



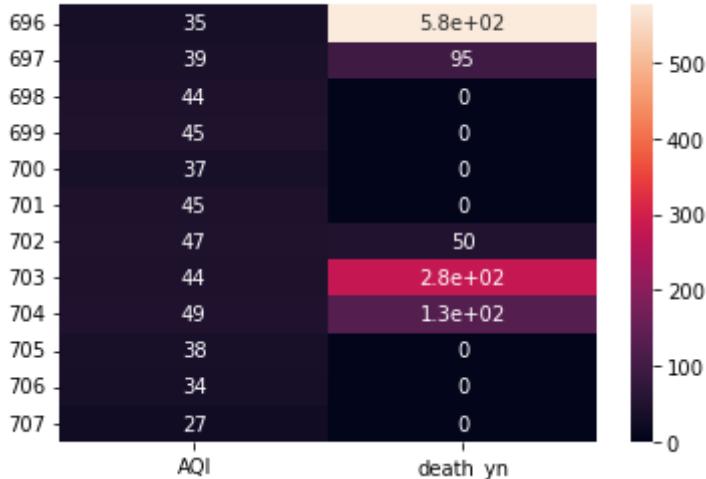
32



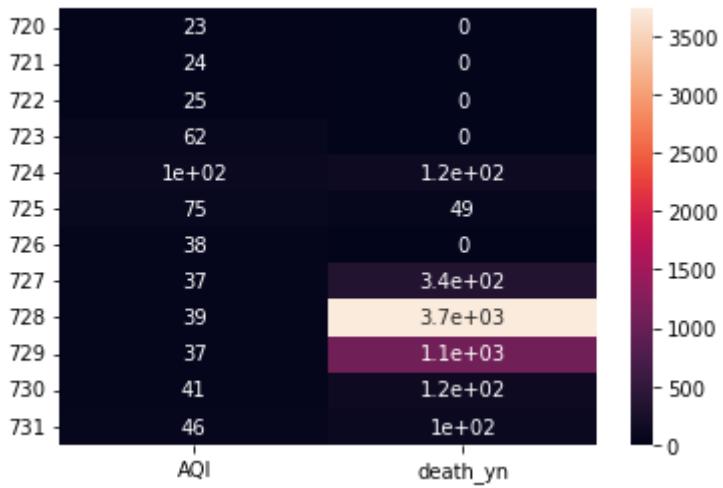
33



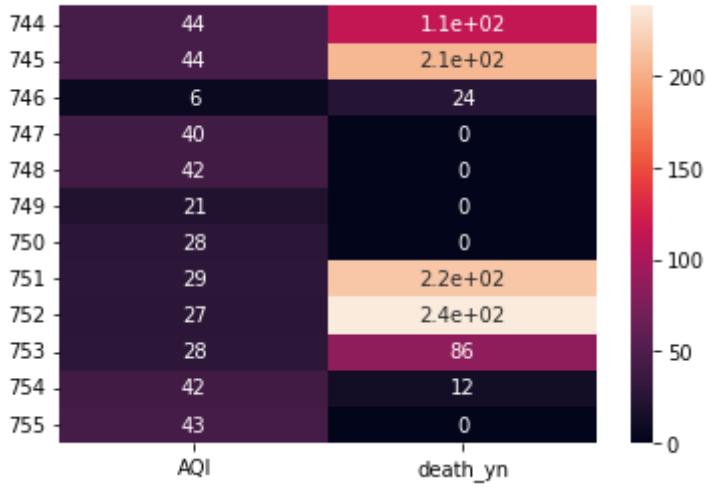
34



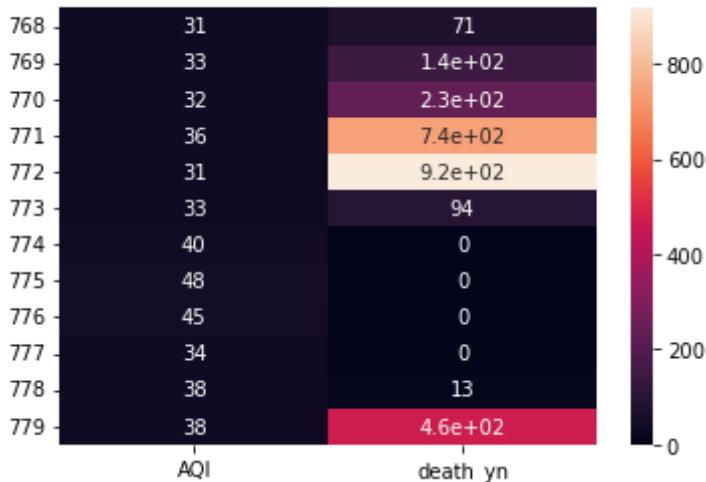
35



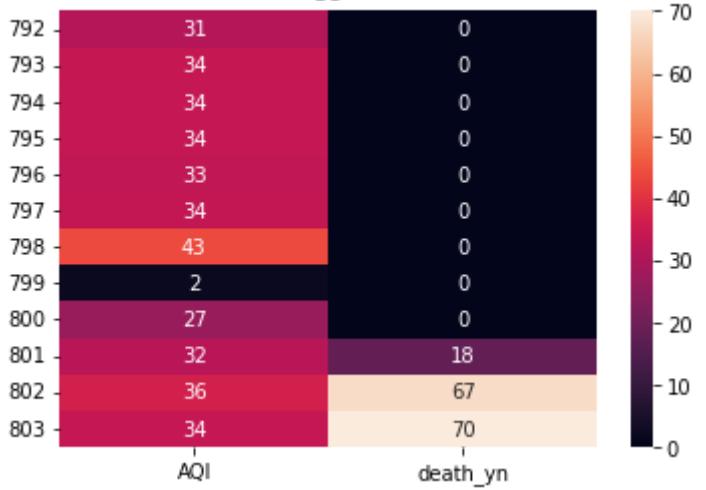
36



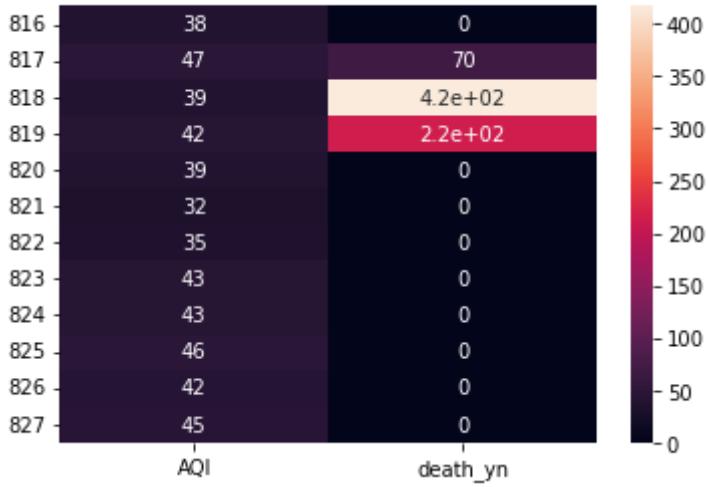
37

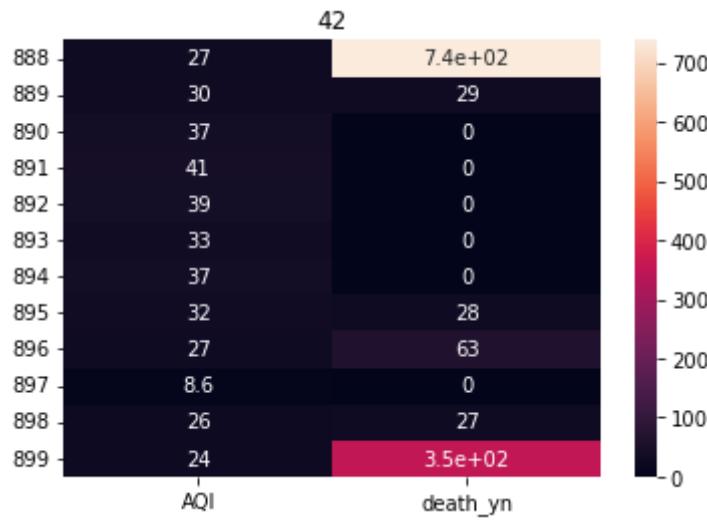
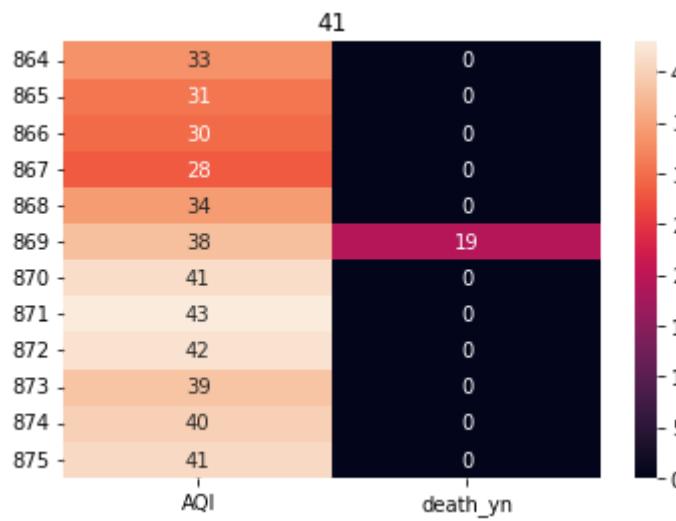
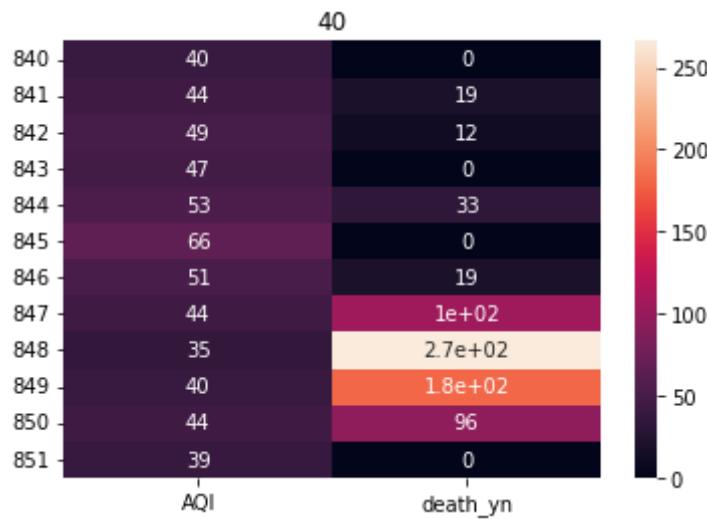


38

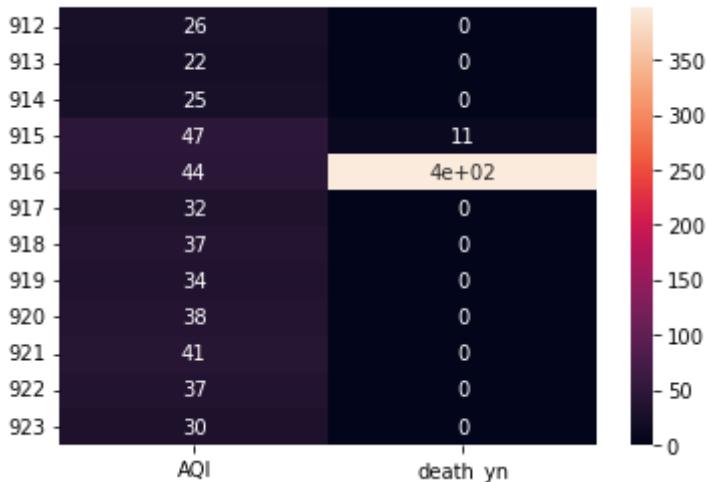


39

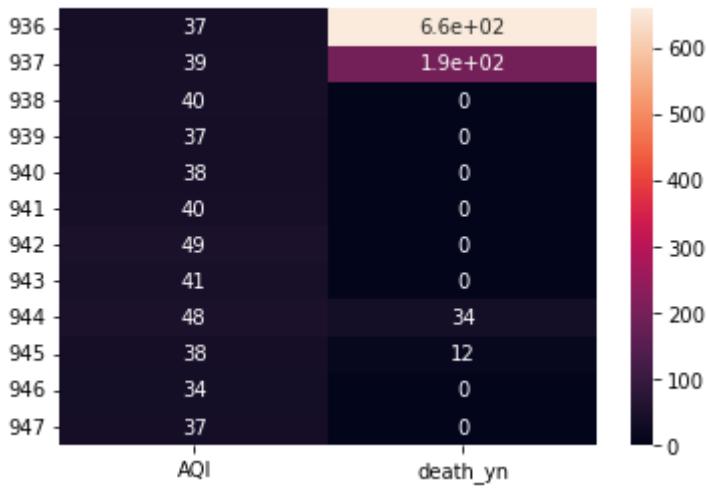




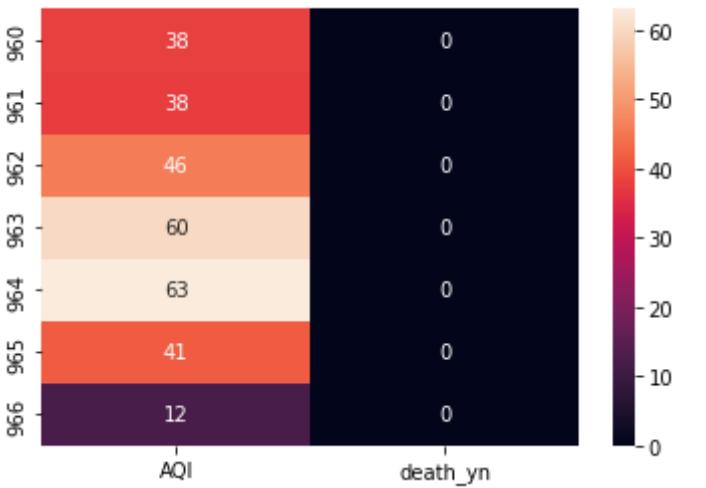
44



45



46



Warning: zero-size array

After Analysing the death conditions of the Dataframe: death\_yn states that the patient die as a result of this illness, which is

specified in terms of [Yes; No; Unknown; Missing] .

Based on the Above heat maps, the AQI has no correlation with the death\_yn Heatmaps as many States of them has 0 death rates between the AQI .

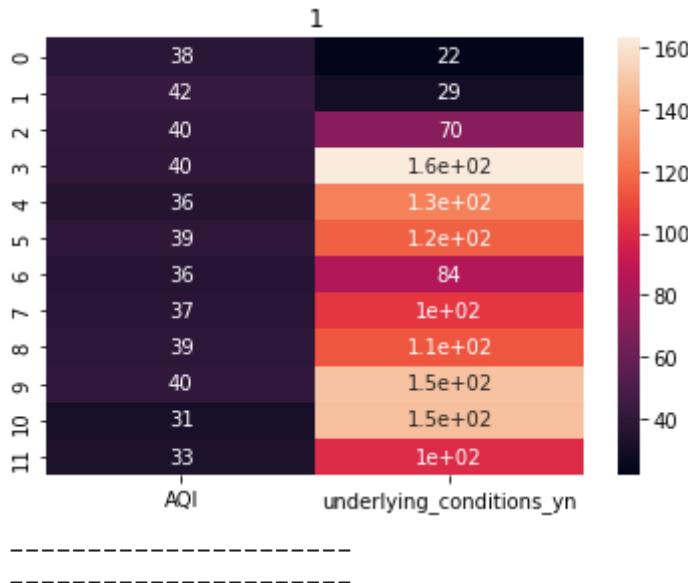
```
In [71]: ##Generating Correlation Heat Map of AQI and Underlying_Conditions

i=0
j=24
for code in sc:

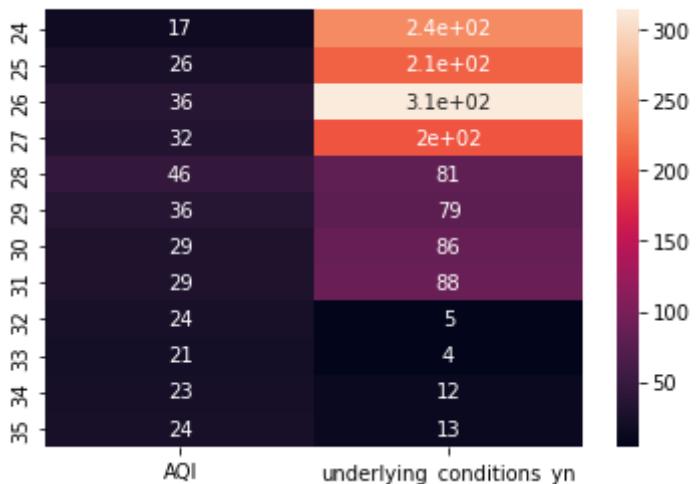
    df_plotter = pd.DataFrame()
    df_plotter=merged_df2.iloc[i:j,:]
    df_plotter = df_plotter.drop(['State_Name','year_month','State Code_x','ID',''
    df_plotter_2020=pd.DataFrame()
    df_plotter_2019=pd.DataFrame()
    df_plotter_2019=df_plotter.iloc[:12,:]
    df_plotter_2020=df_plotter.iloc[12:,:]

    if not df_plotter.empty:
        try:
            ax = plt.axes()
            sns.heatmap(df_plotter_2019, annot=True, ax=ax)
            ax.set_title(code)
            plt.show()
            print('-----')
            p2 = sns.heatmap(df_plotter_2020, annot=True, ax=ax)
            ax.set_title(code)
            plt.show()
            print('-----')
        except ValueError:
            print("Warning: zero-size array")

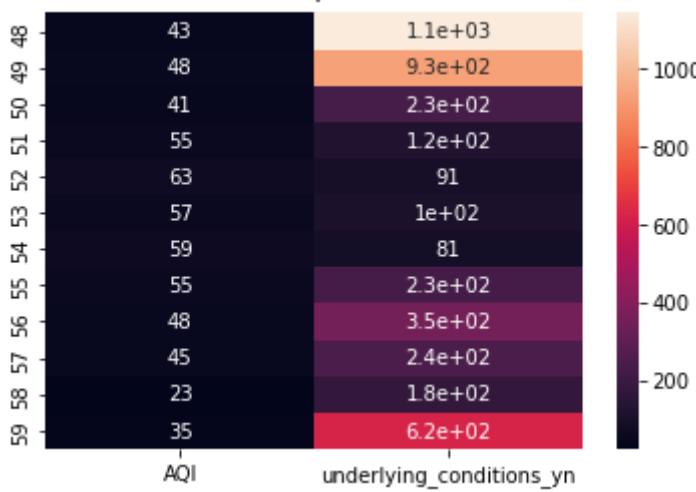
    i+=24
    j+=24
```



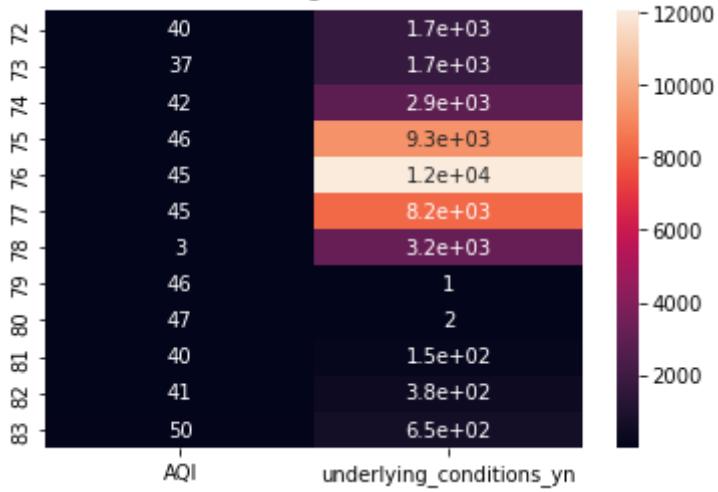
2

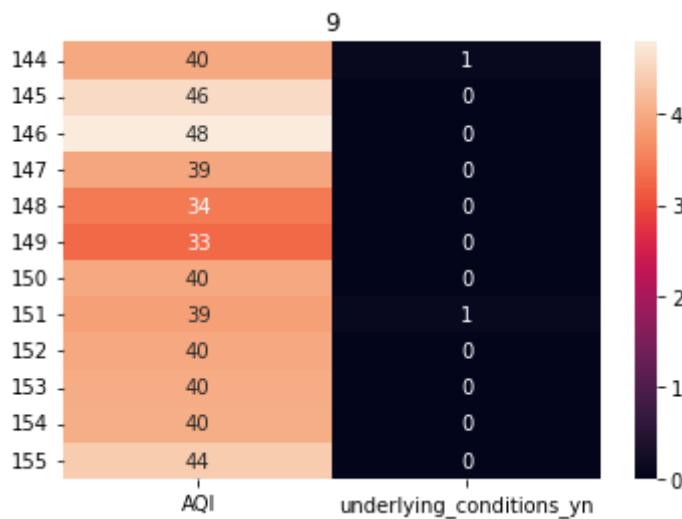
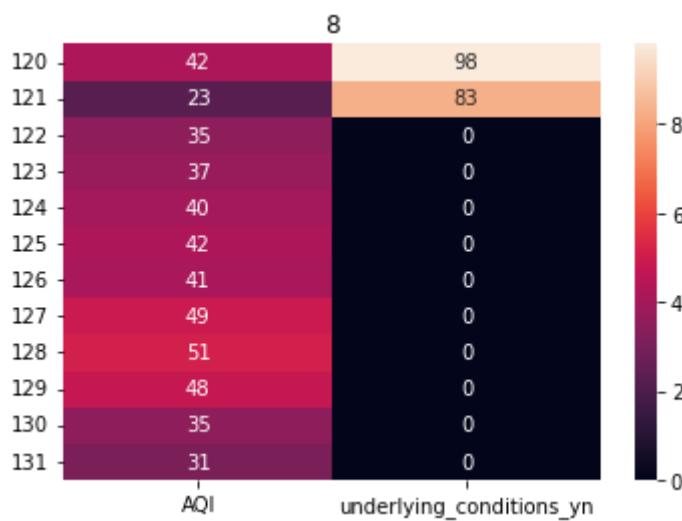
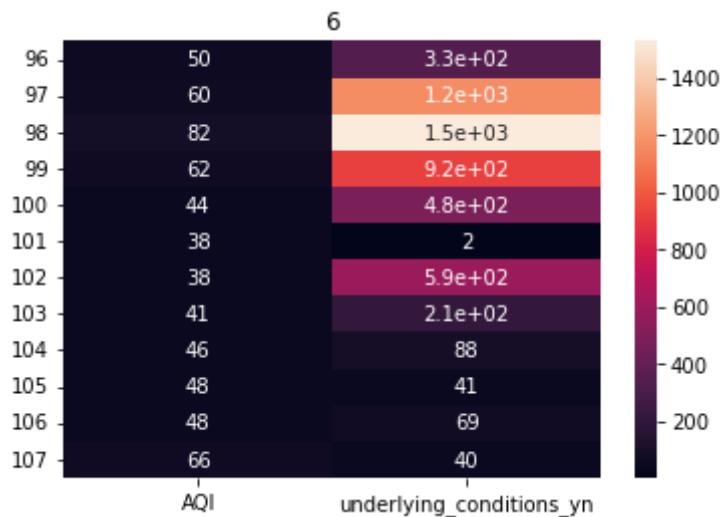


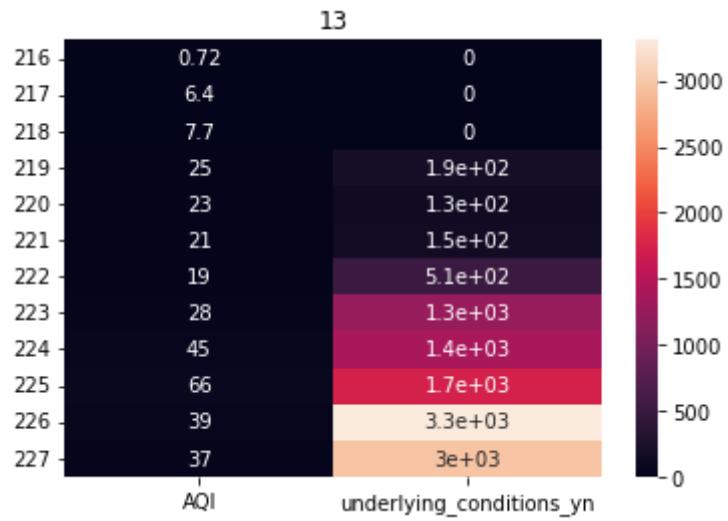
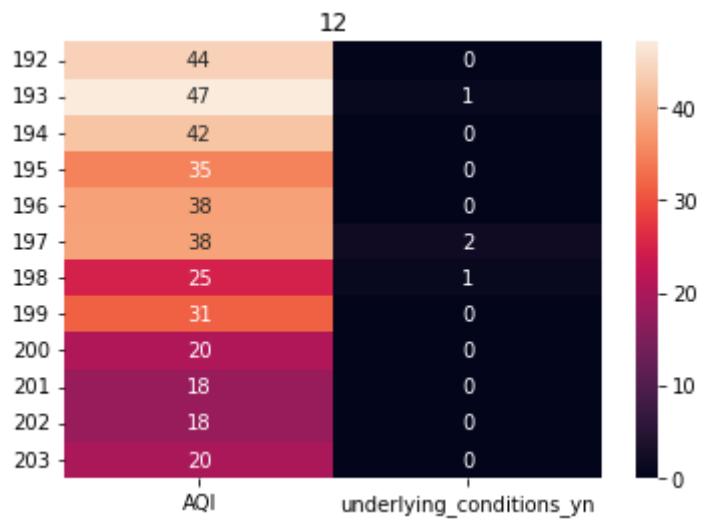
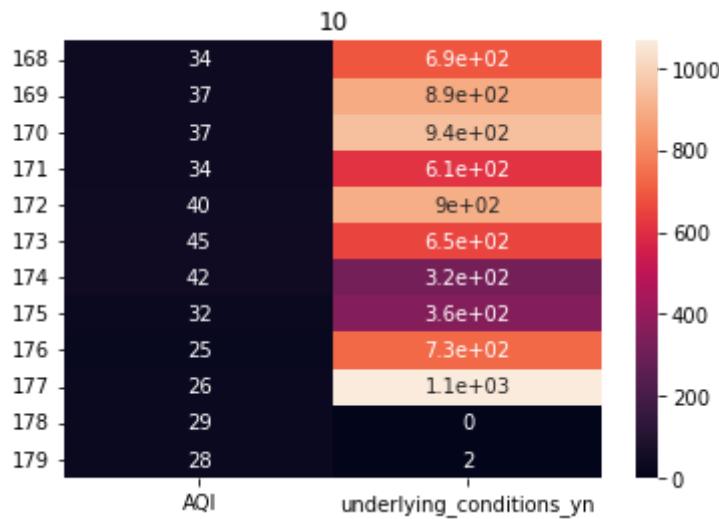
4



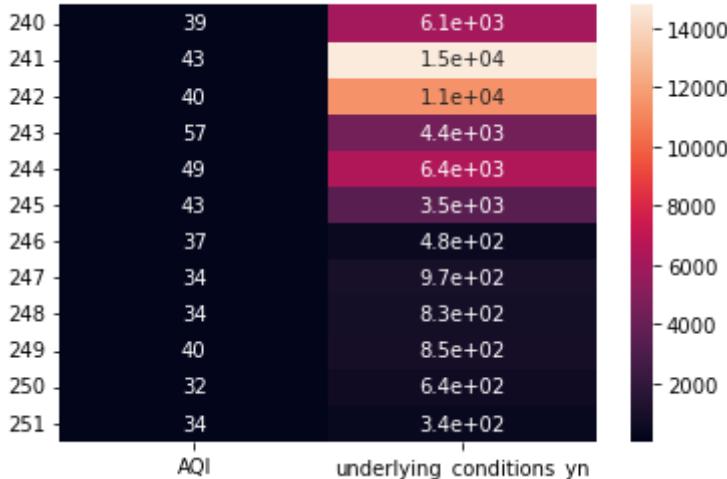
5



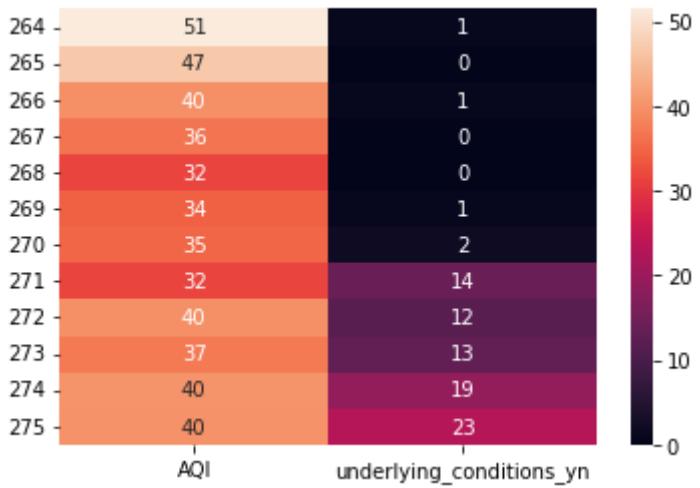




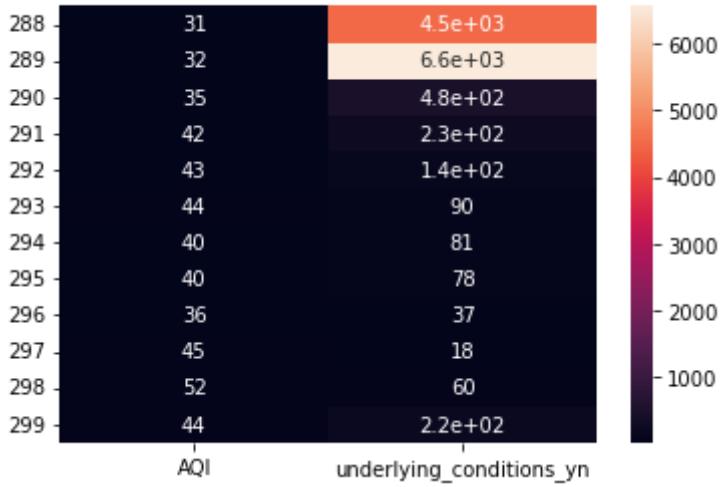
15



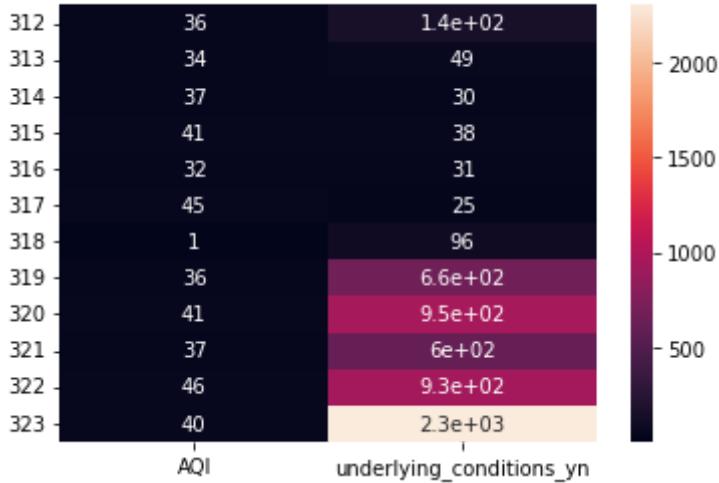
16



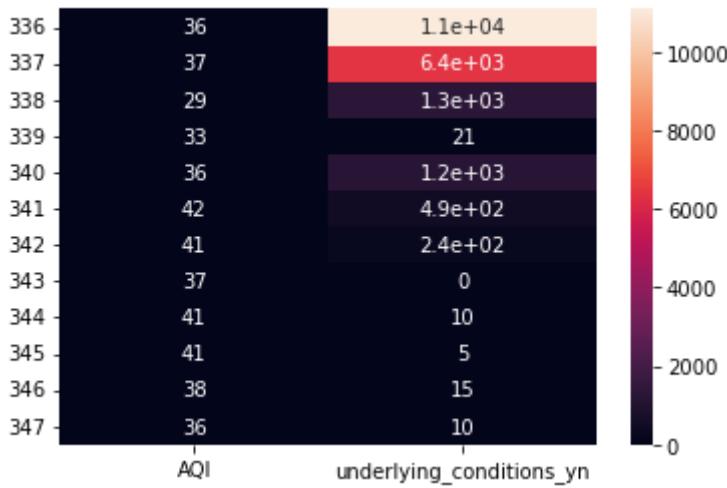
17



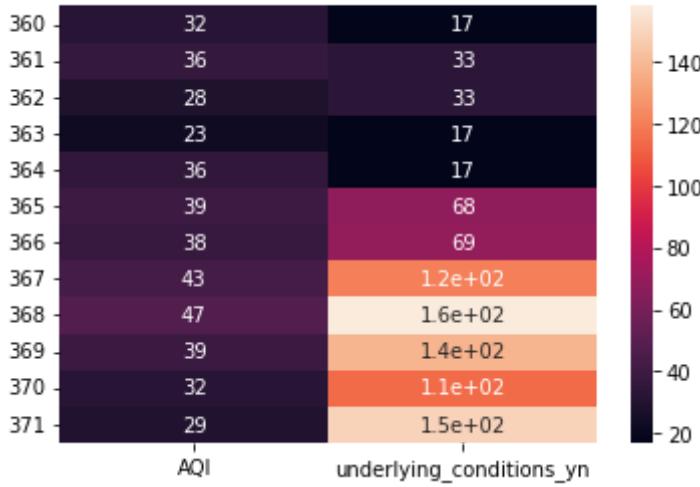
18



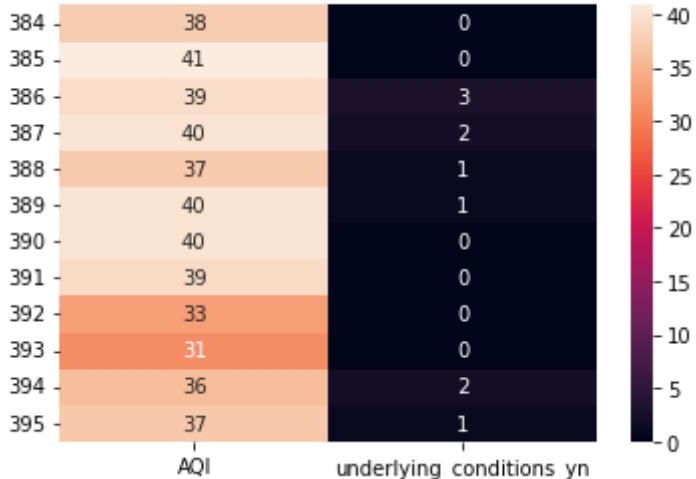
19



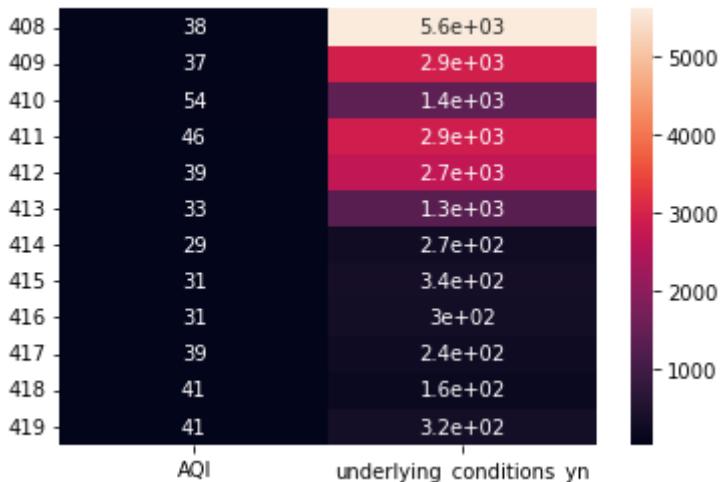
20



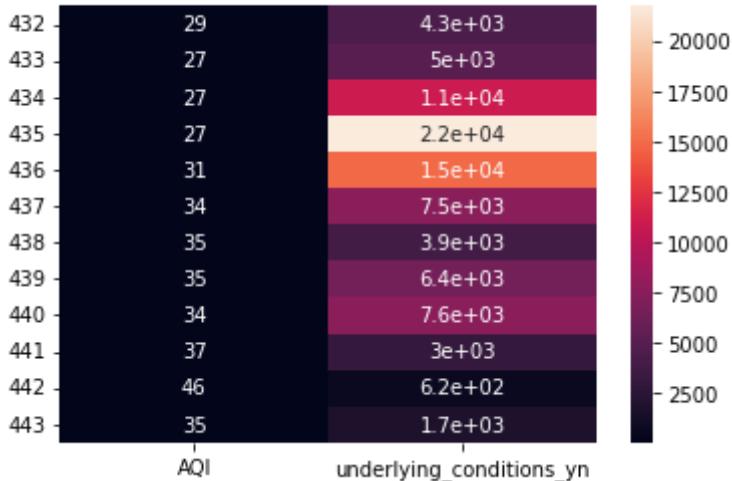
21



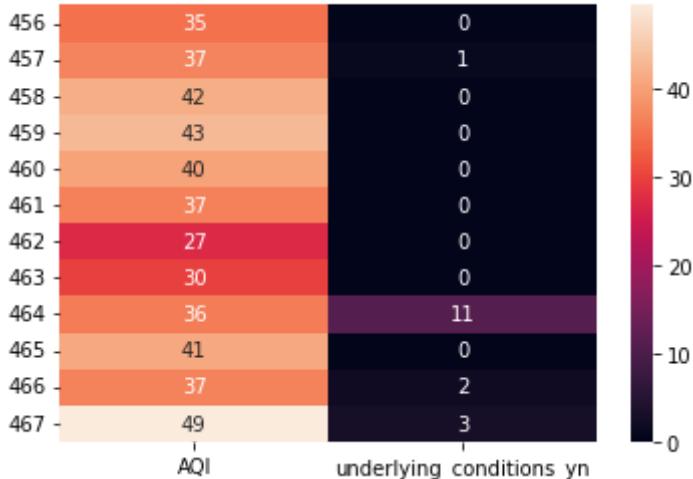
22



23



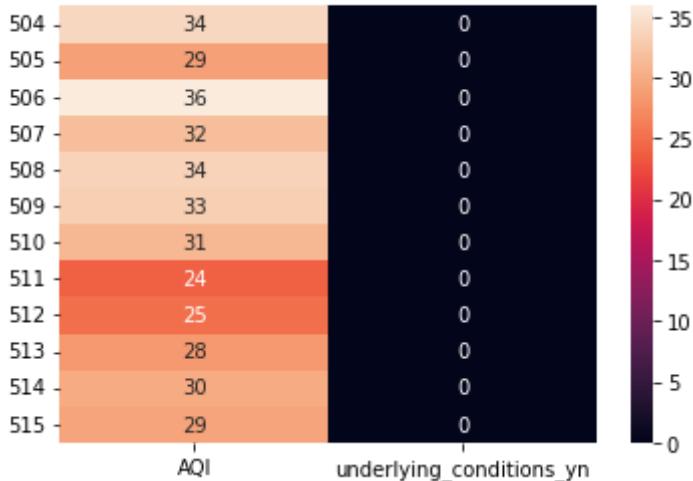
24



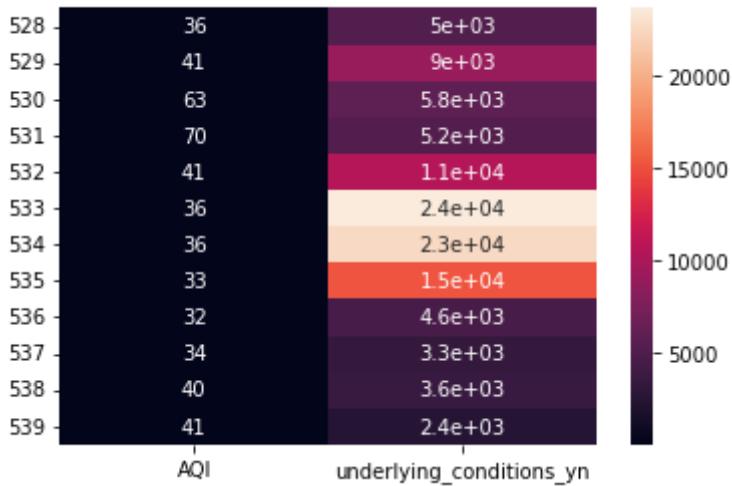
25



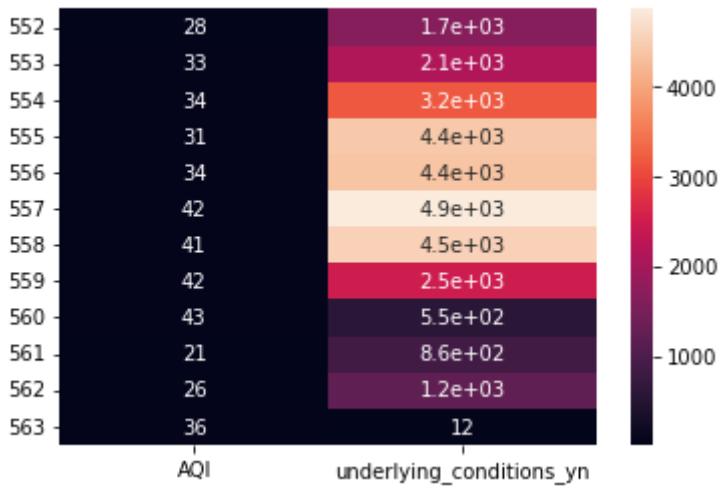
26



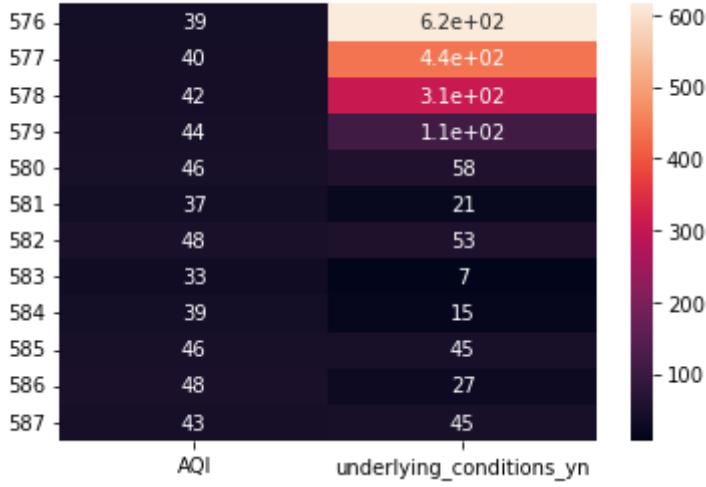
27



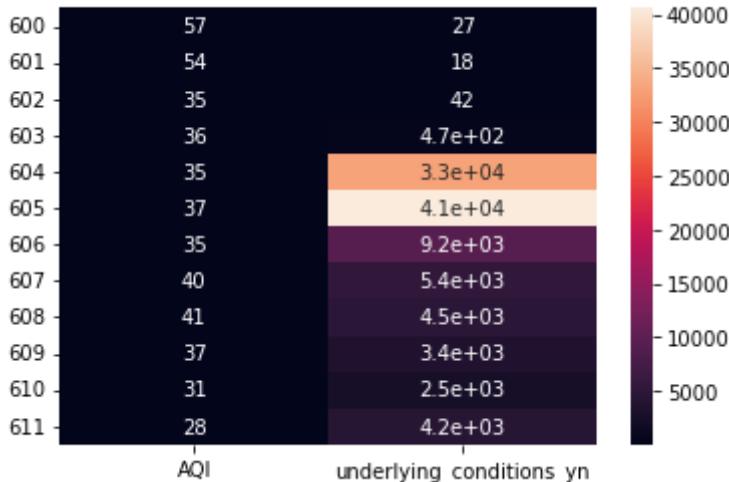
28



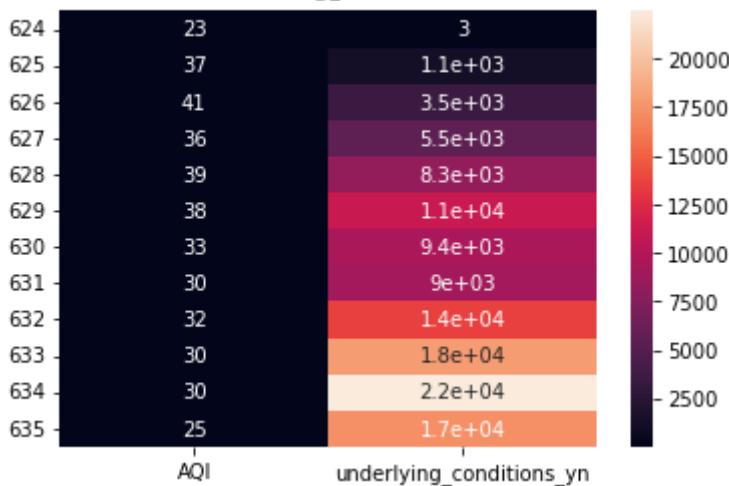
29



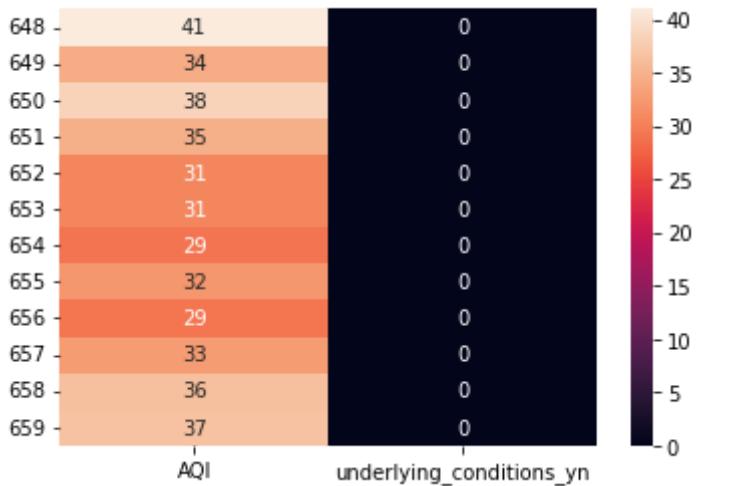
30



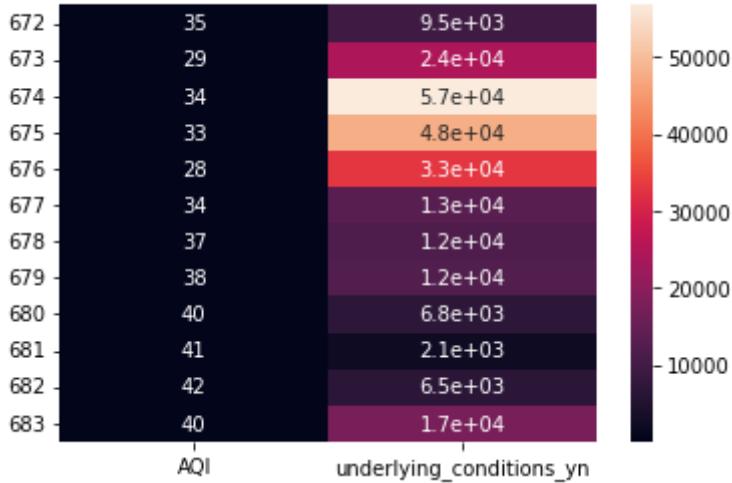
31



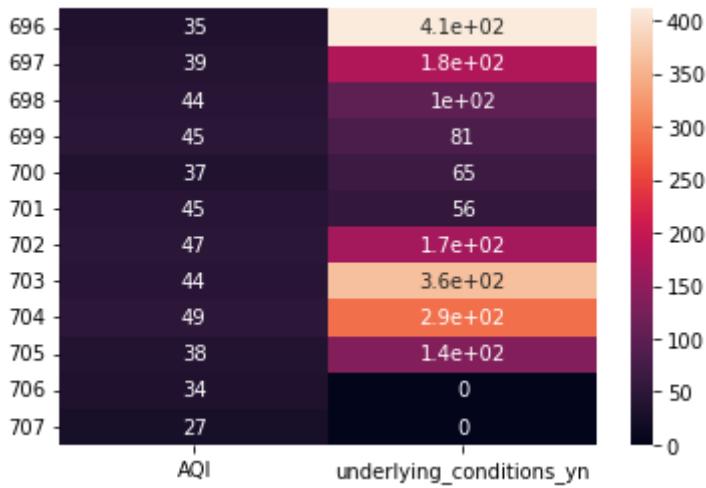
32



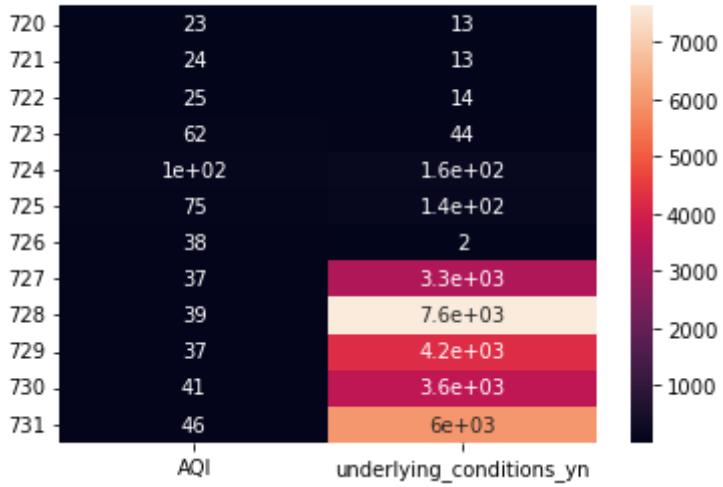
33



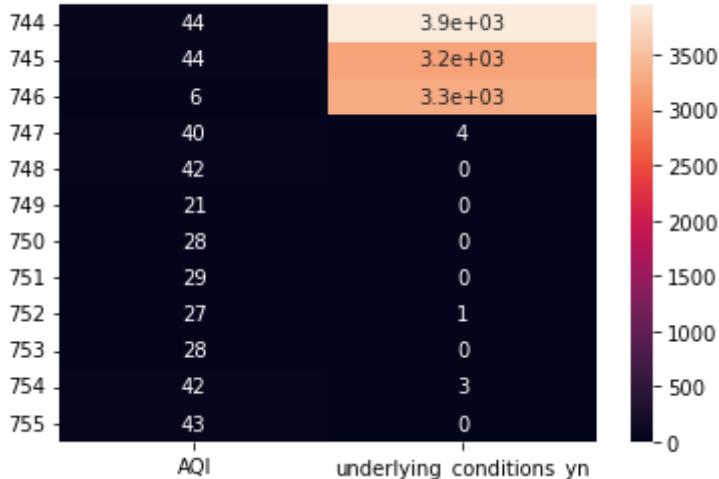
34



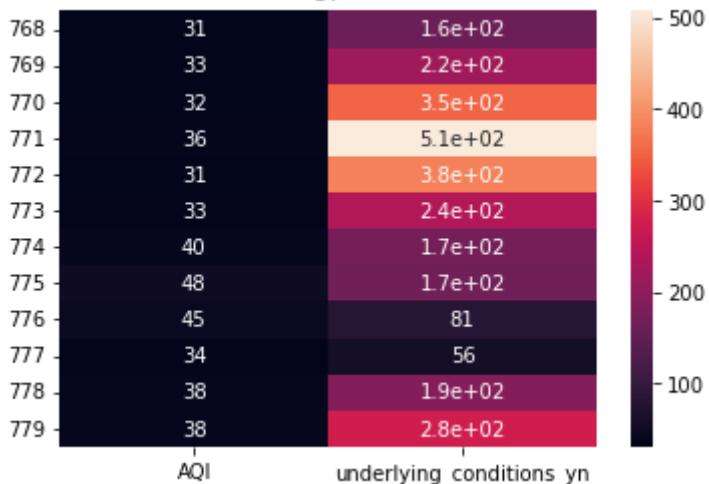
35



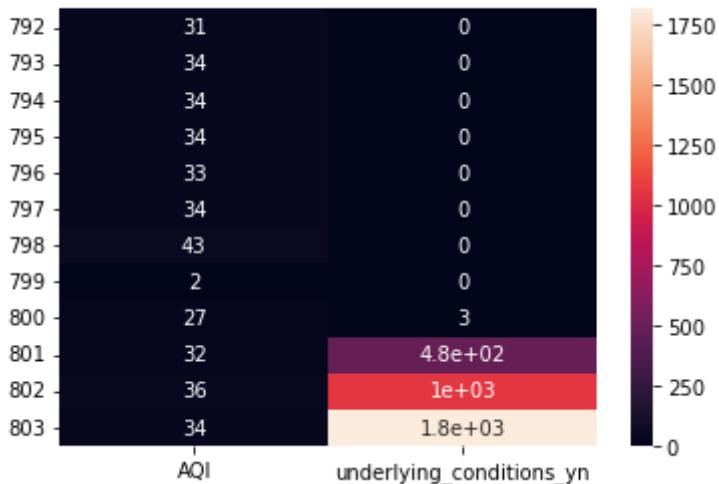
36



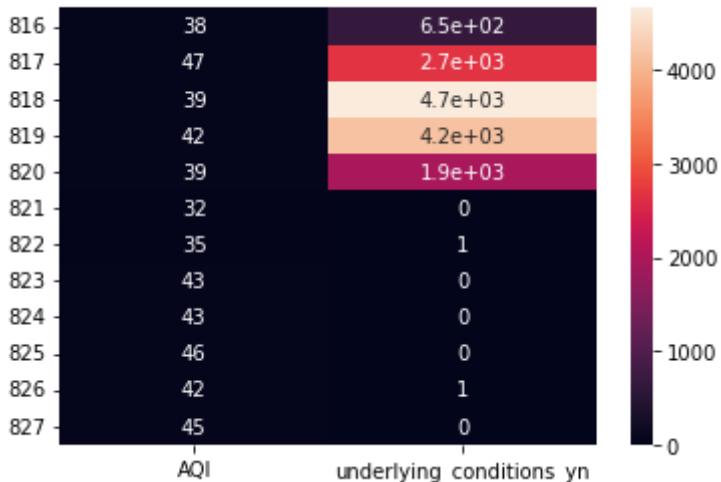
37



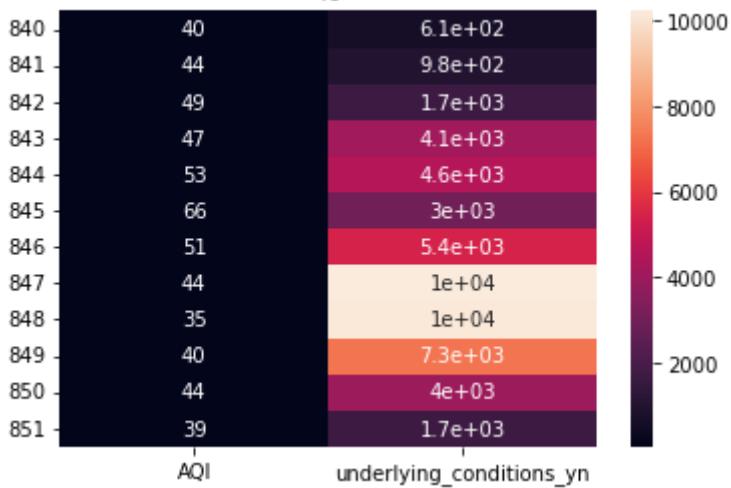
38



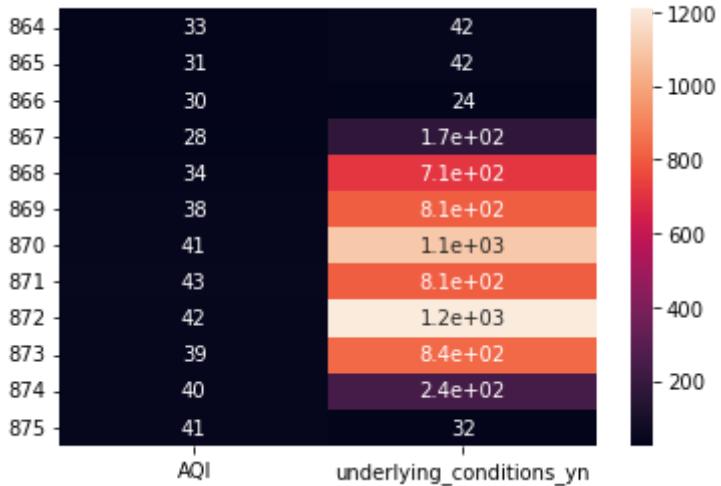
39



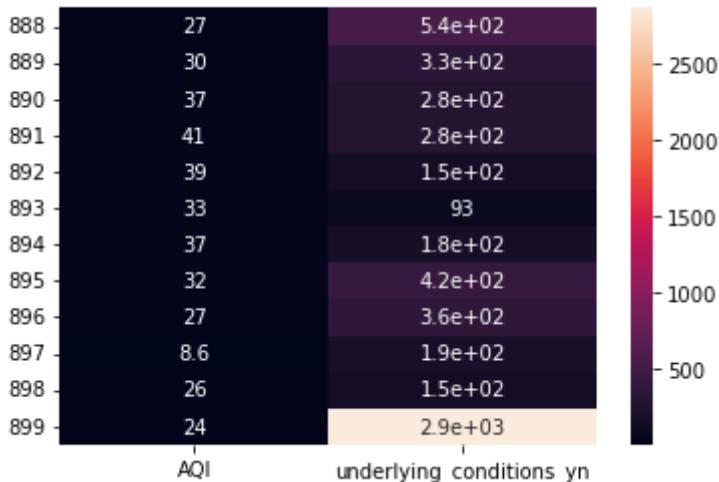
40



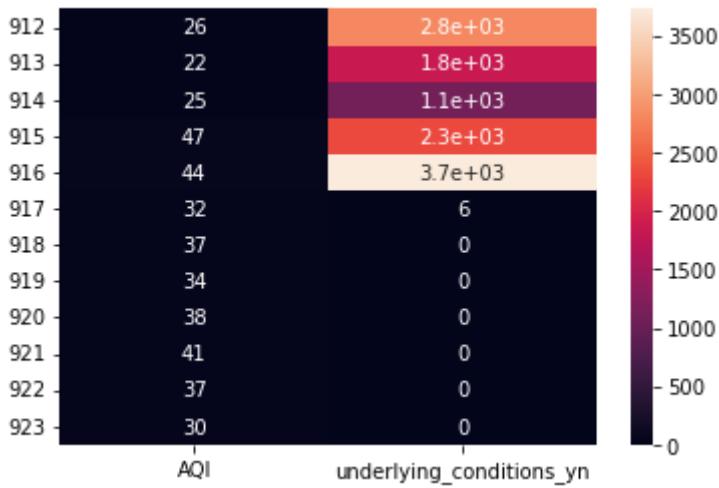
41



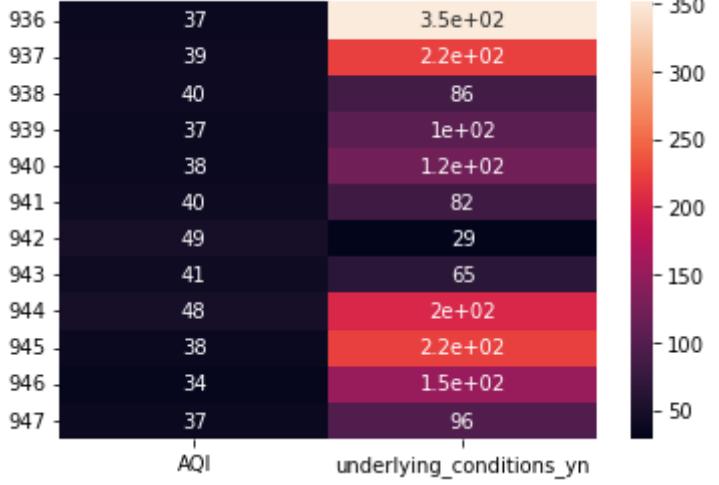
42

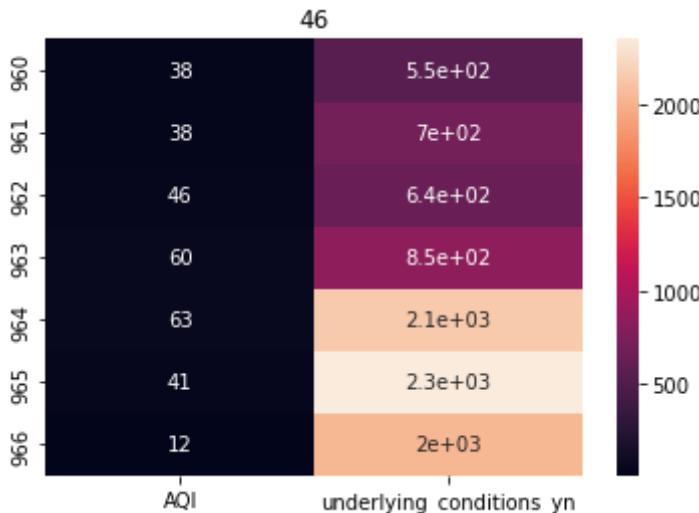


44



45





-----  
Warning: zero-size array

After Analysing the Underlying conditions of the Dataframe:  
Underlying\_Conditions states that the patient have one or more of the underlying medical conditions and risk behaviors: diabetes mellitus, hypertension, severe obesity ( $BMI > 40$ ), cardiovascular disease, chronic renal disease, chronic liver disease, chronic lung disease, other chronic diseases, immunosuppressive condition, autoimmune condition, current smoker, former smoker, substance abuse or misuse, disability, psychological/psychiatric, pregnancy, other. Which is specified in terms of [Yes, No, Unknown]

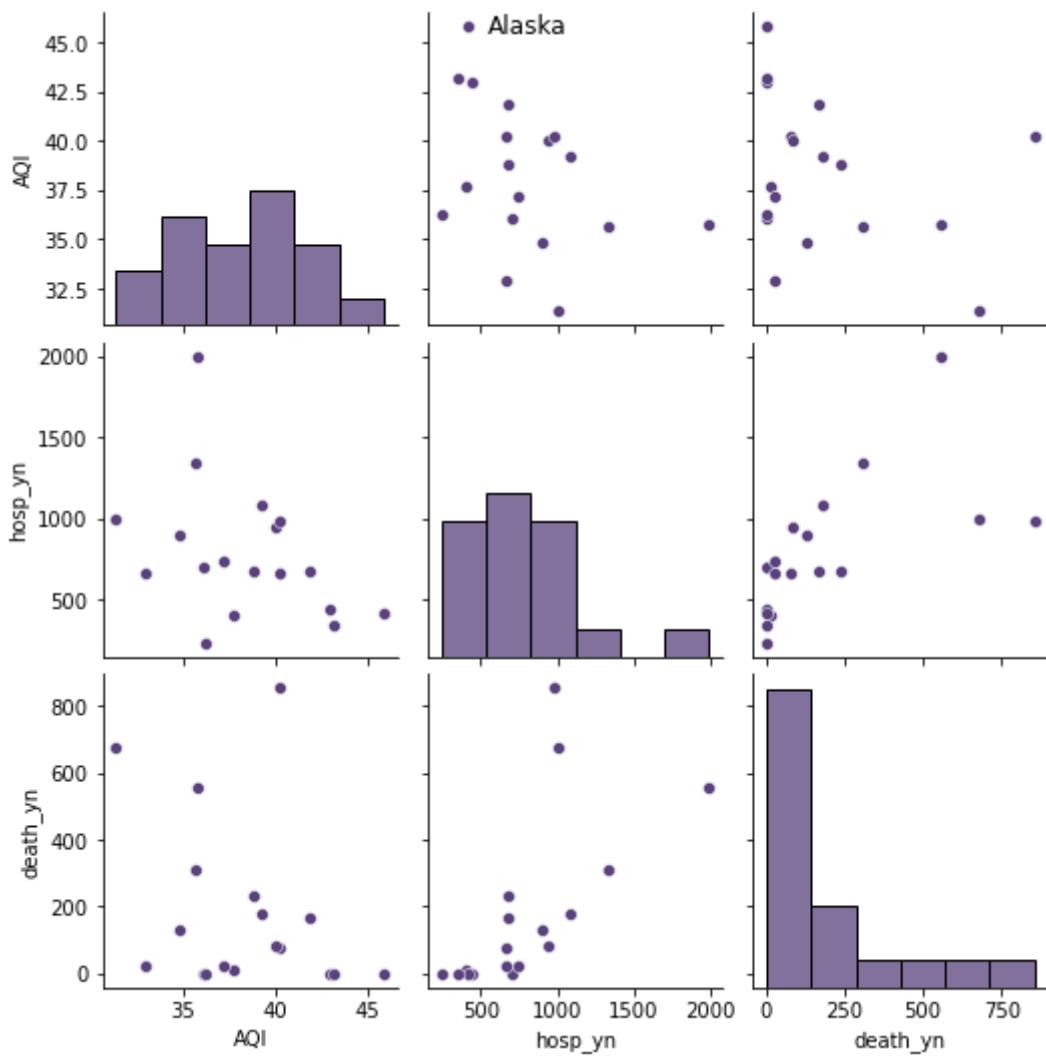
Based on the Above heat maps, the AQI has no correlation with the Underlying\_Heatmaps.

-----  
-----

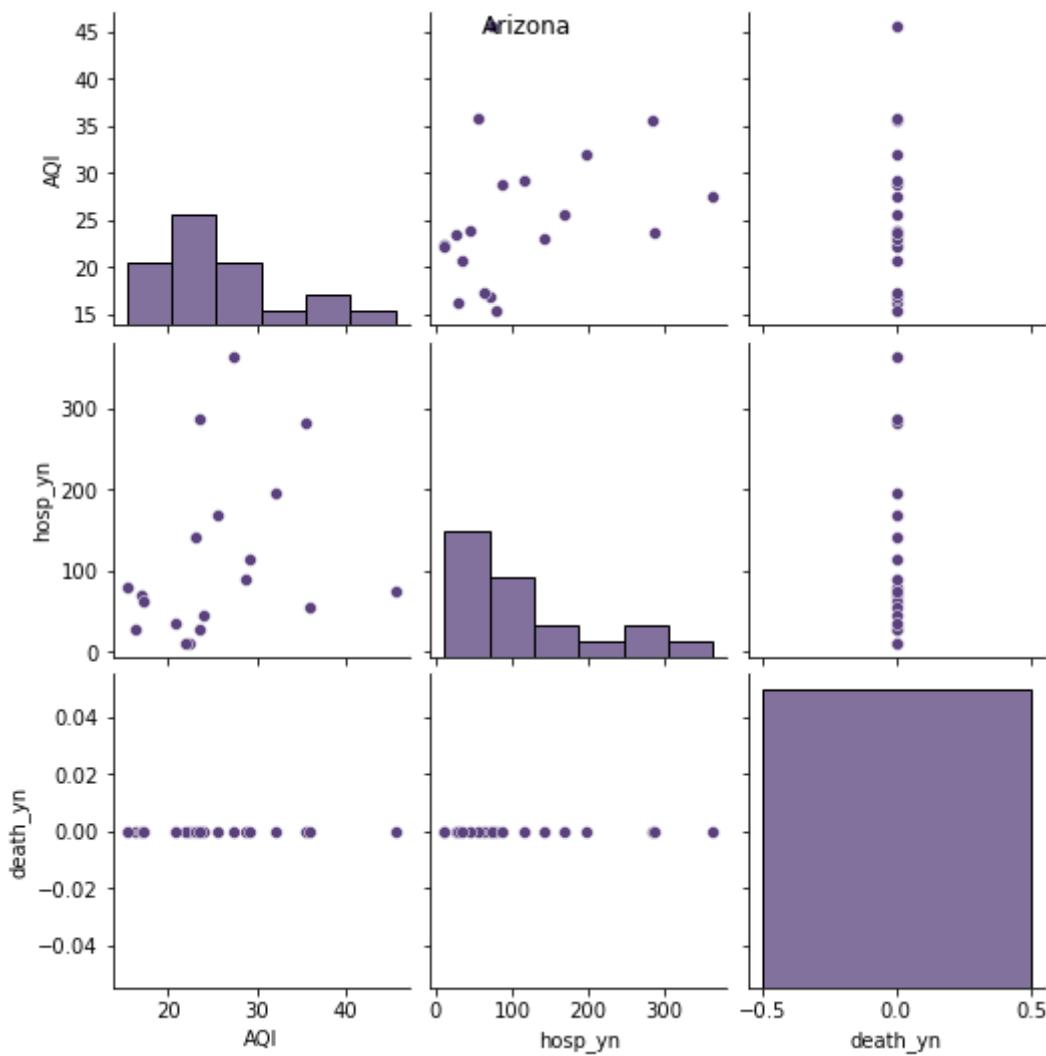
```
In [72]: for country in country_list:
    cn=[ ]
    cn.append(country)
    df_plotter = pd.DataFrame()
    df_plotter=merged_df2[merged_df2['State_Name'].isin(cn)]
    t1 = df_plotter[['AQI', 'hosp_yn', 'death_yn']]
    plt.suptitle(country)
    sns.set_palette("Purples_r", desat=0.6)
    plt.figure(figsize=(15,7))
    sns.pairplot(t1,diag_kind='hist')
```

<Figure size 432x288 with 0 Axes>

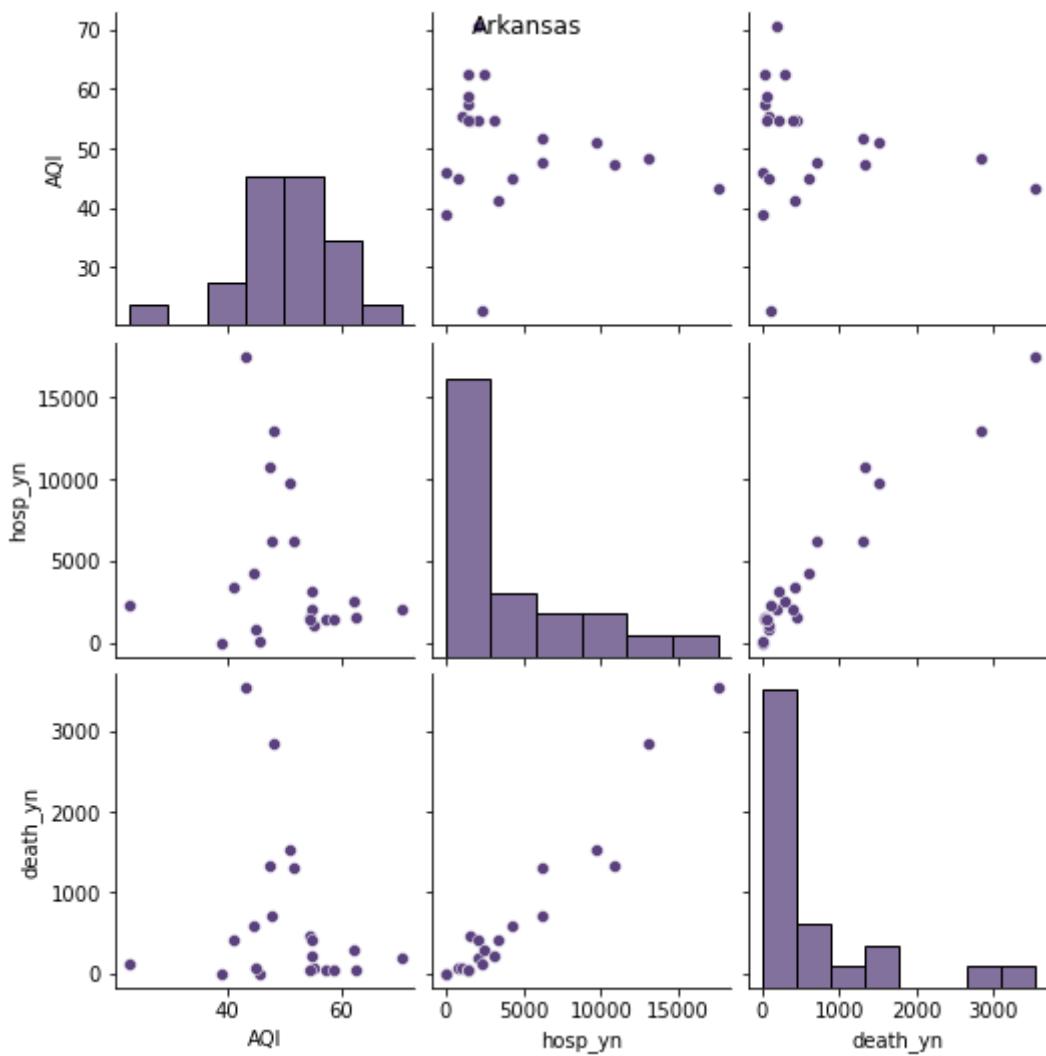
<Figure size 1080x504 with 0 Axes>



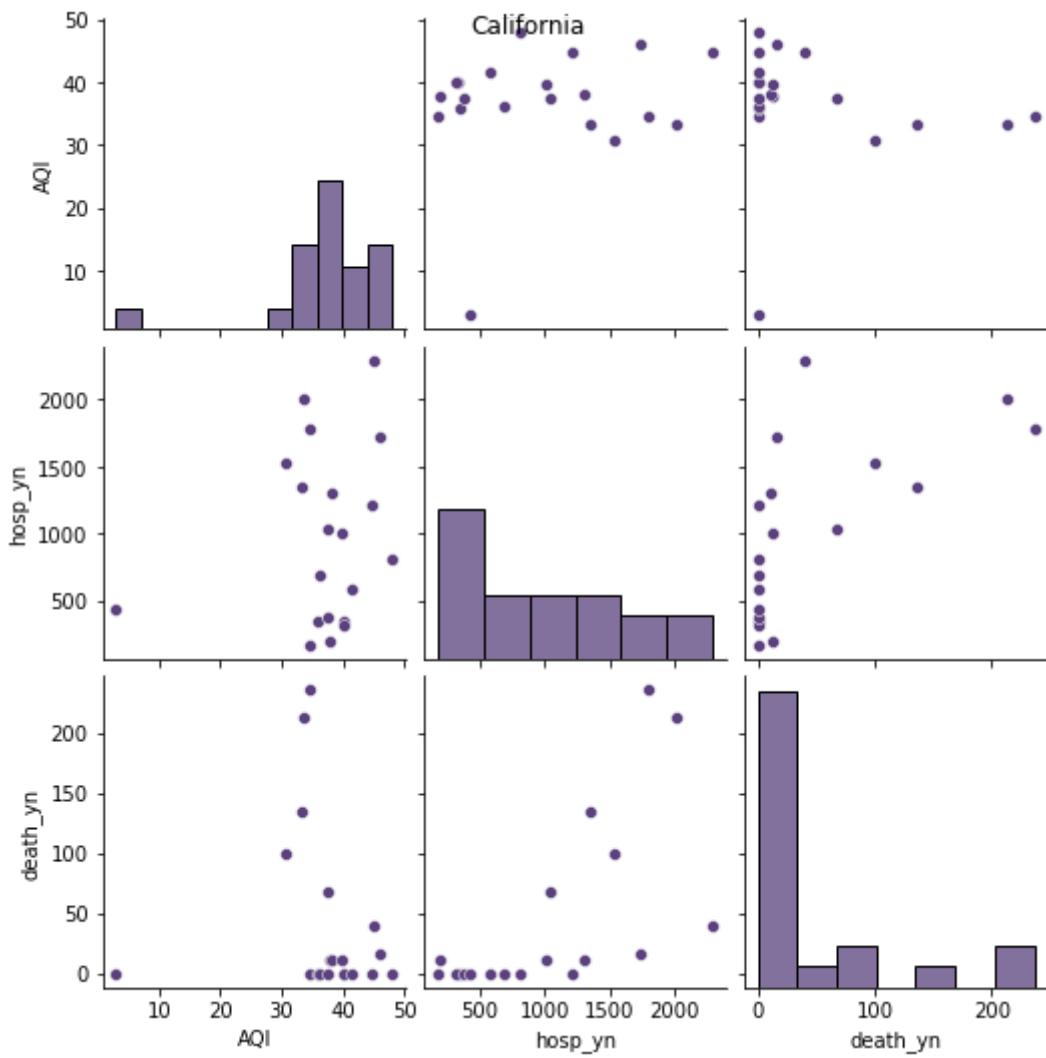
<Figure size 1080x504 with 0 Axes>



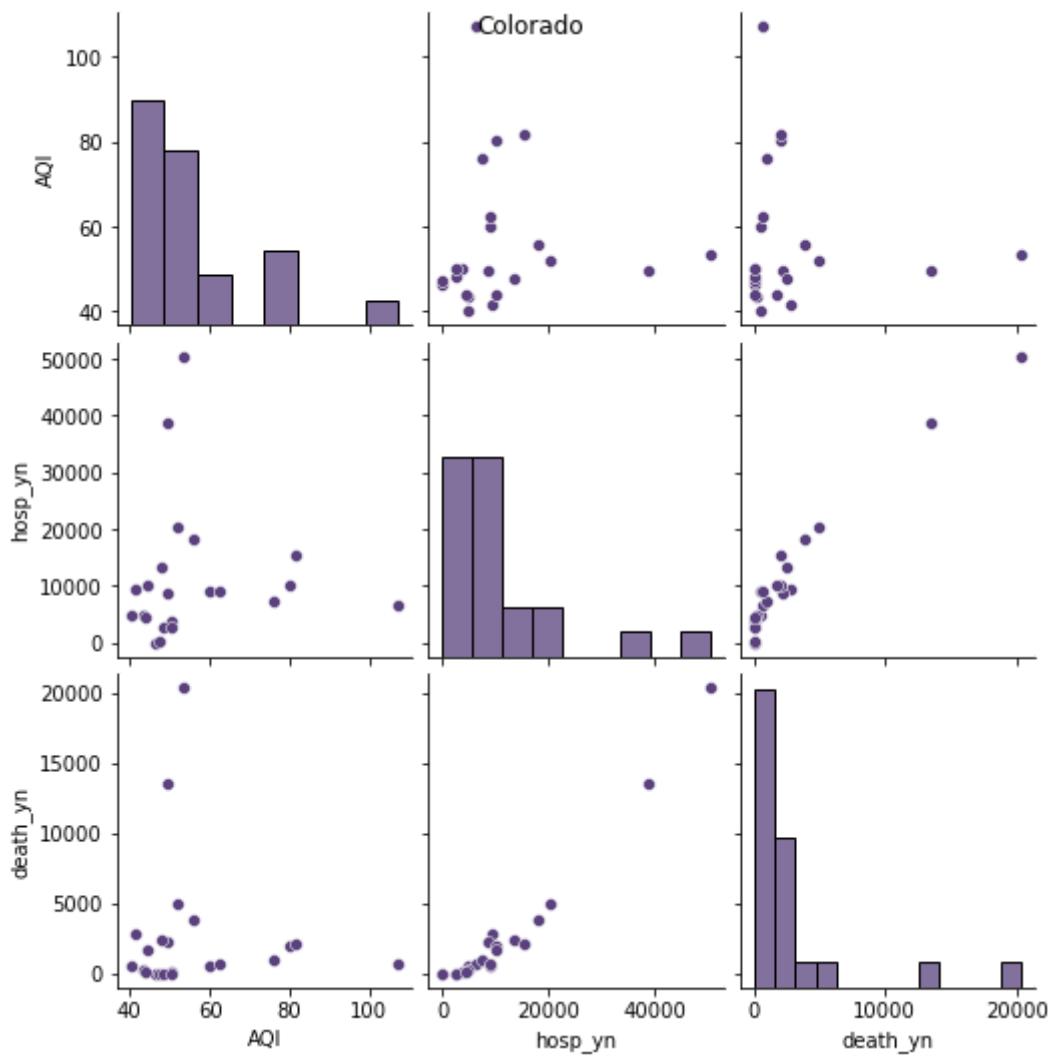
<Figure size 1080x504 with 0 Axes>



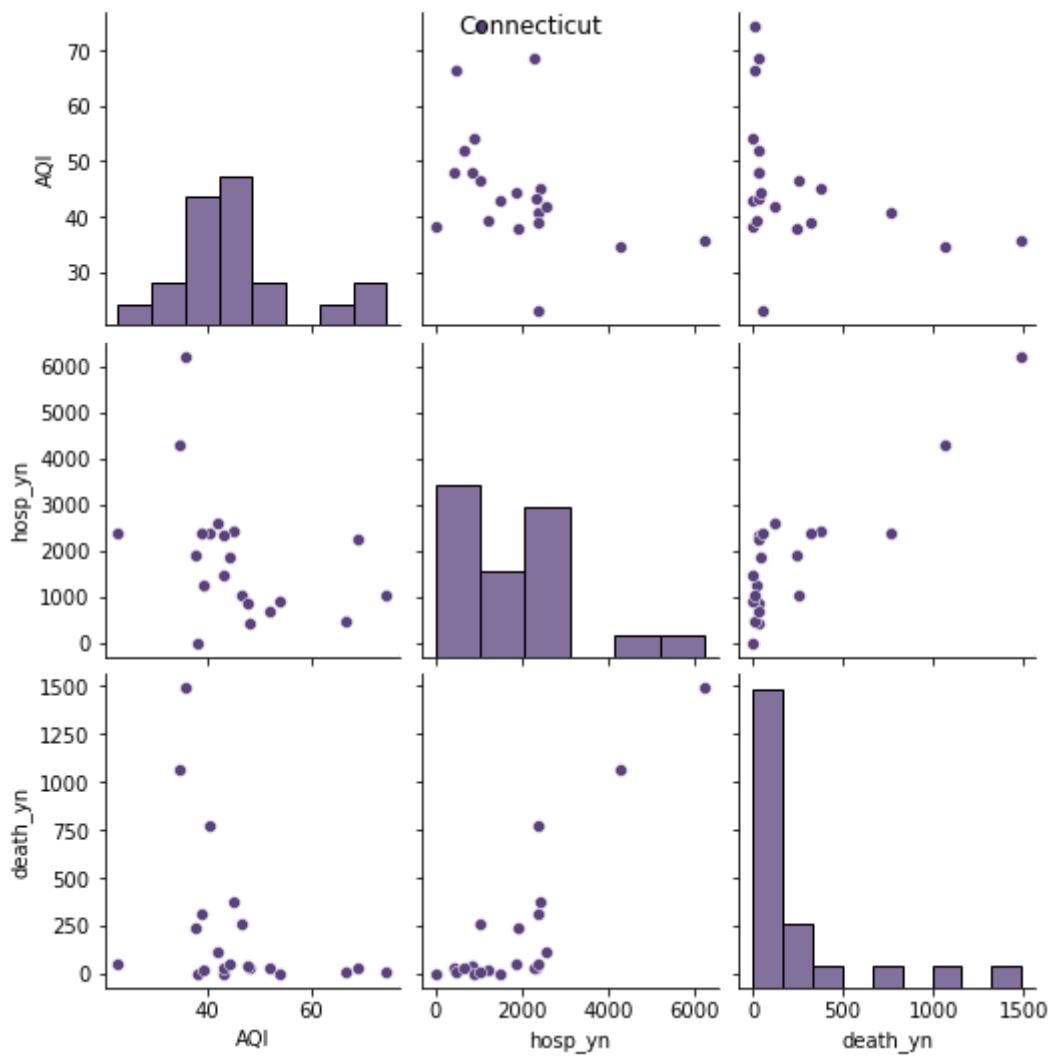
&lt;Figure size 1080x504 with 0 Axes&gt;



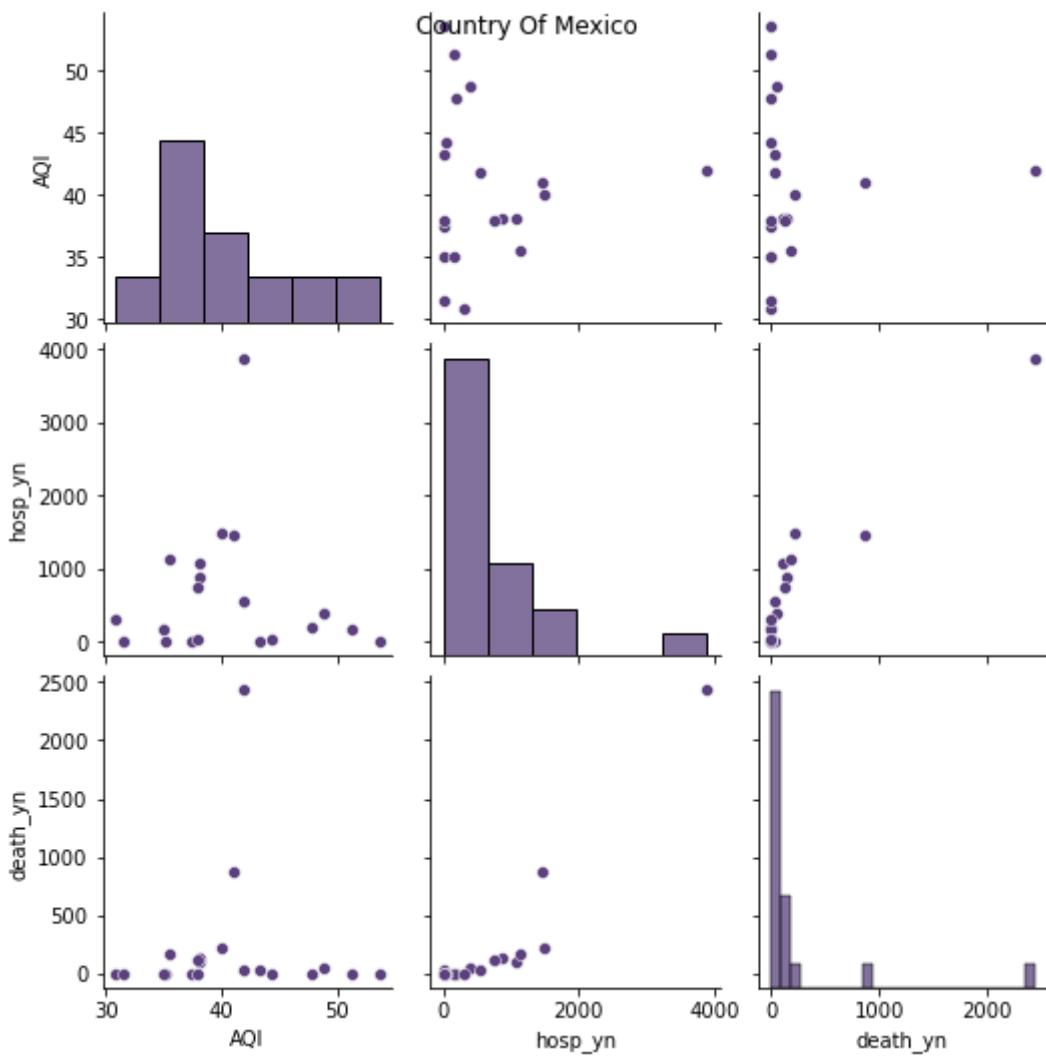
<Figure size 1080x504 with 0 Axes>



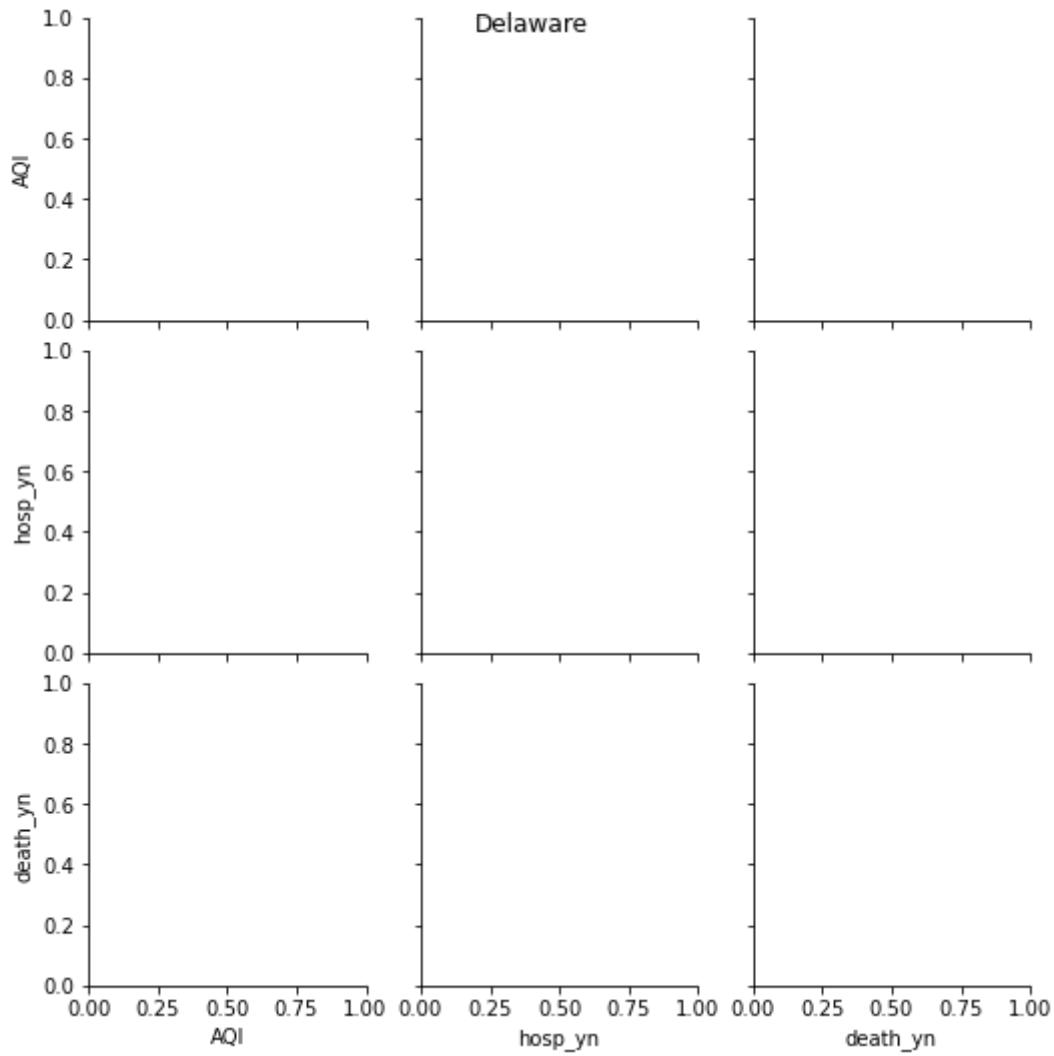
&lt;Figure size 1080x504 with 0 Axes&gt;



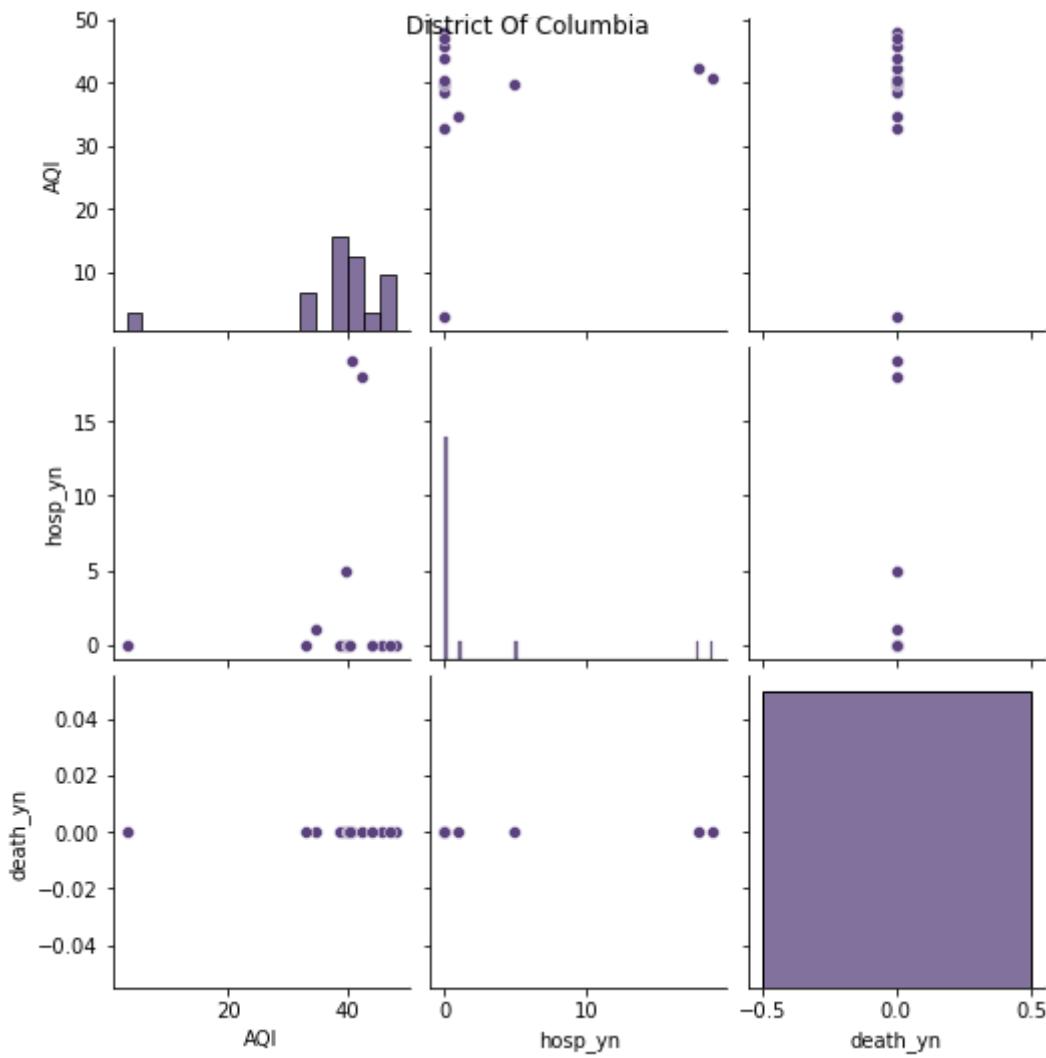
<Figure size 1080x504 with 0 Axes>



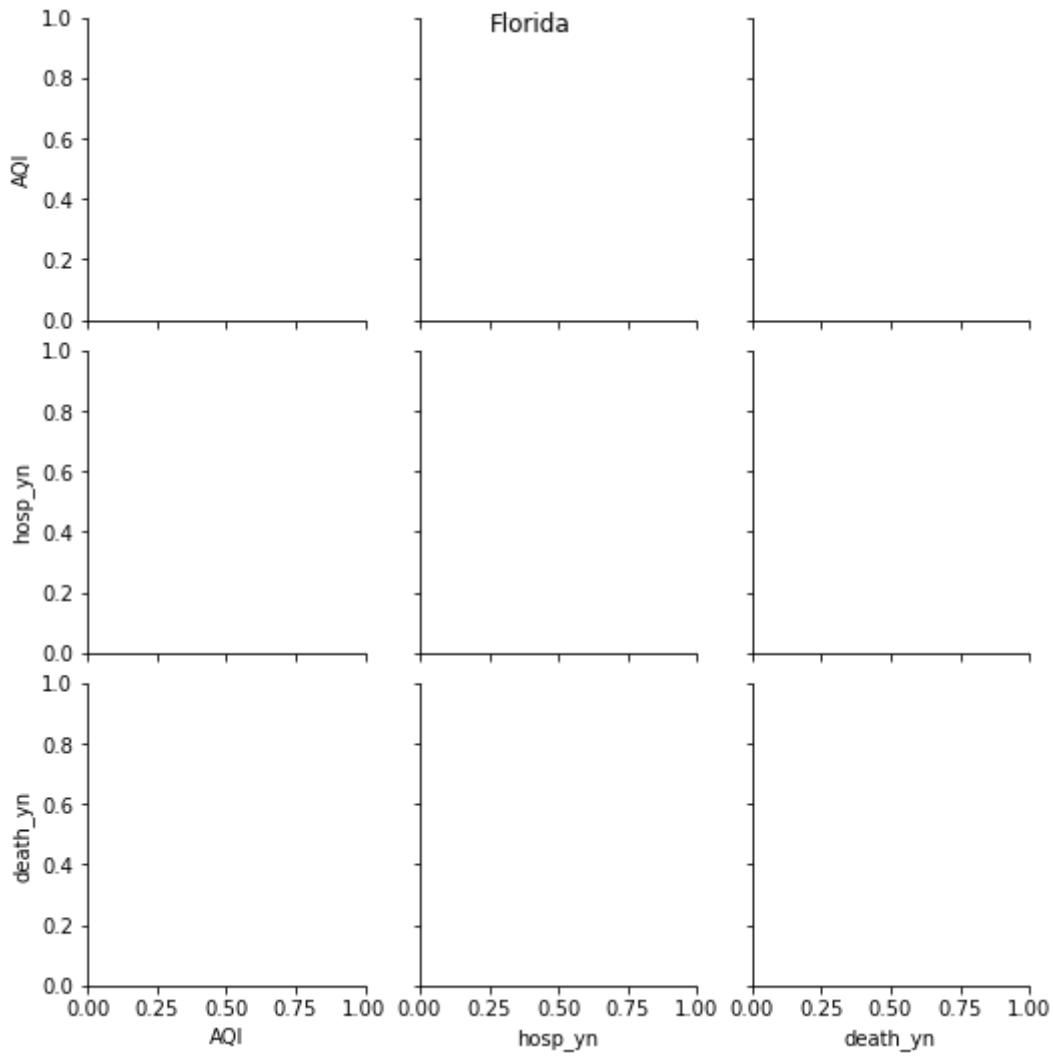
<Figure size 1080x504 with 0 Axes>



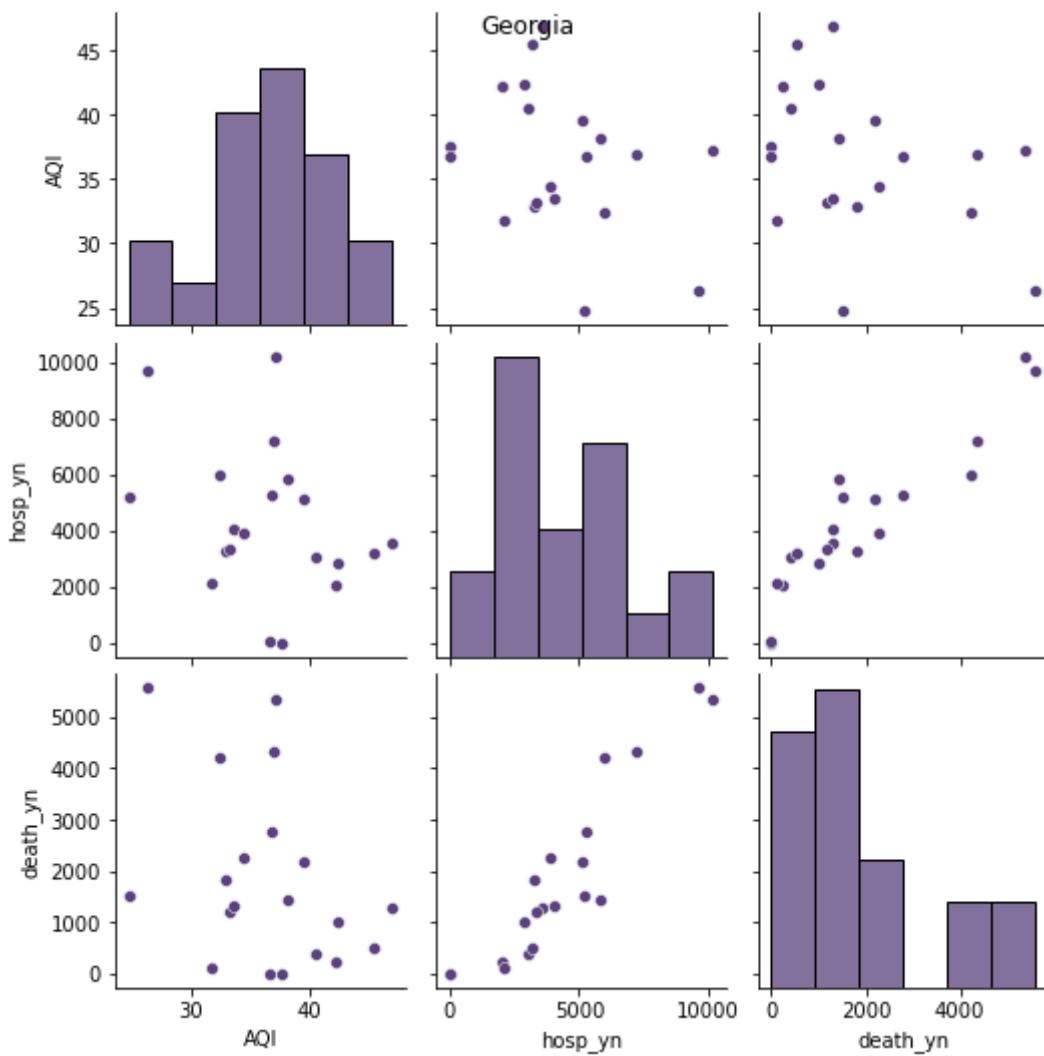
<Figure size 1080x504 with 0 Axes>



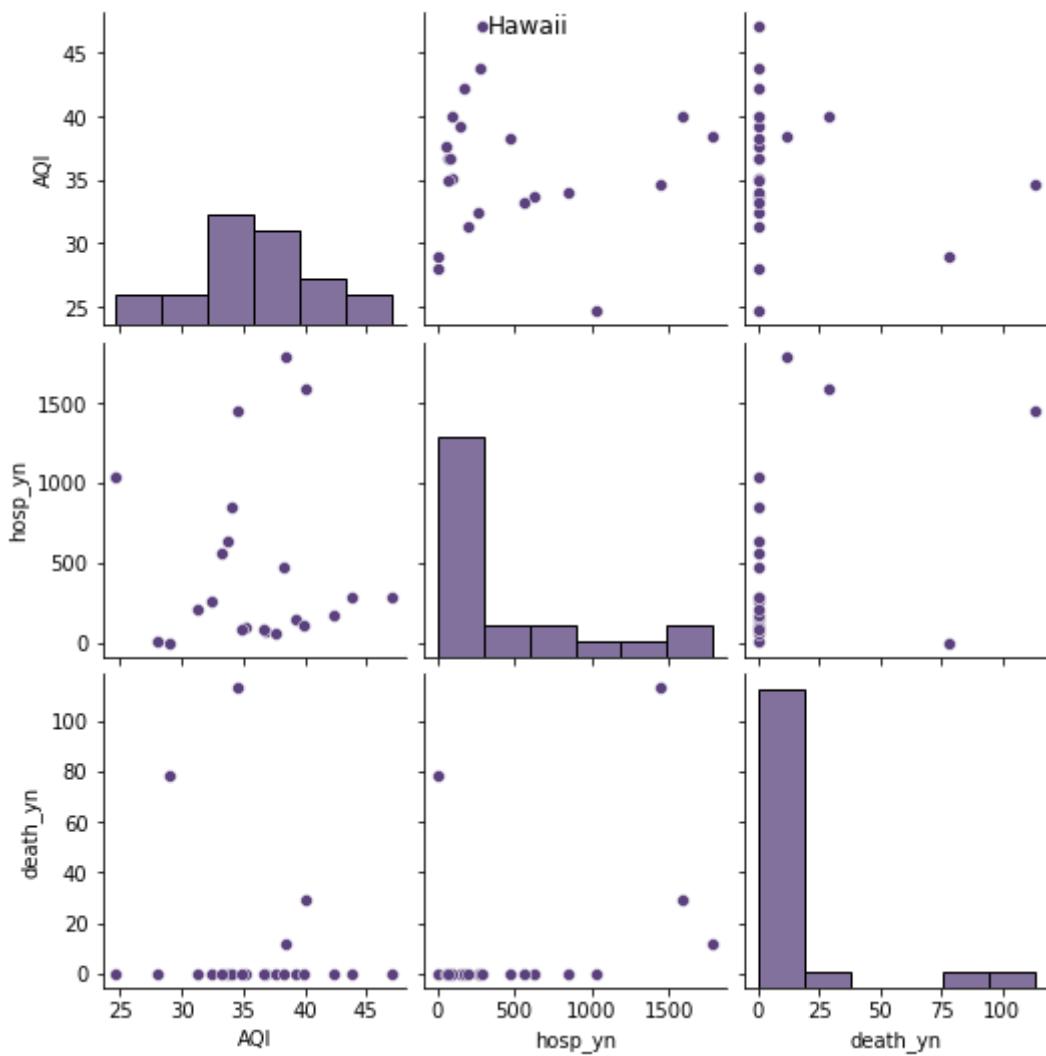
<Figure size 1080x504 with 0 Axes>



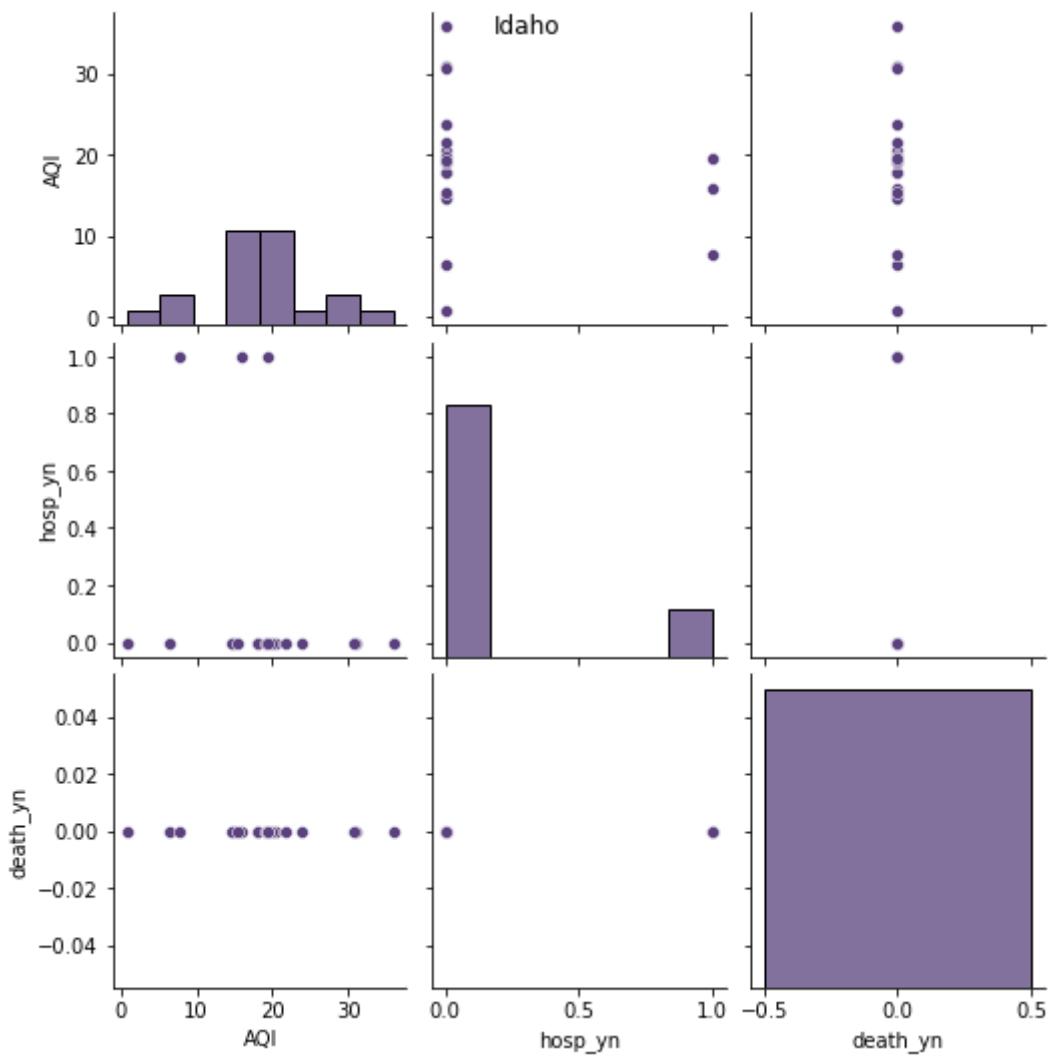
<Figure size 1080x504 with 0 Axes>



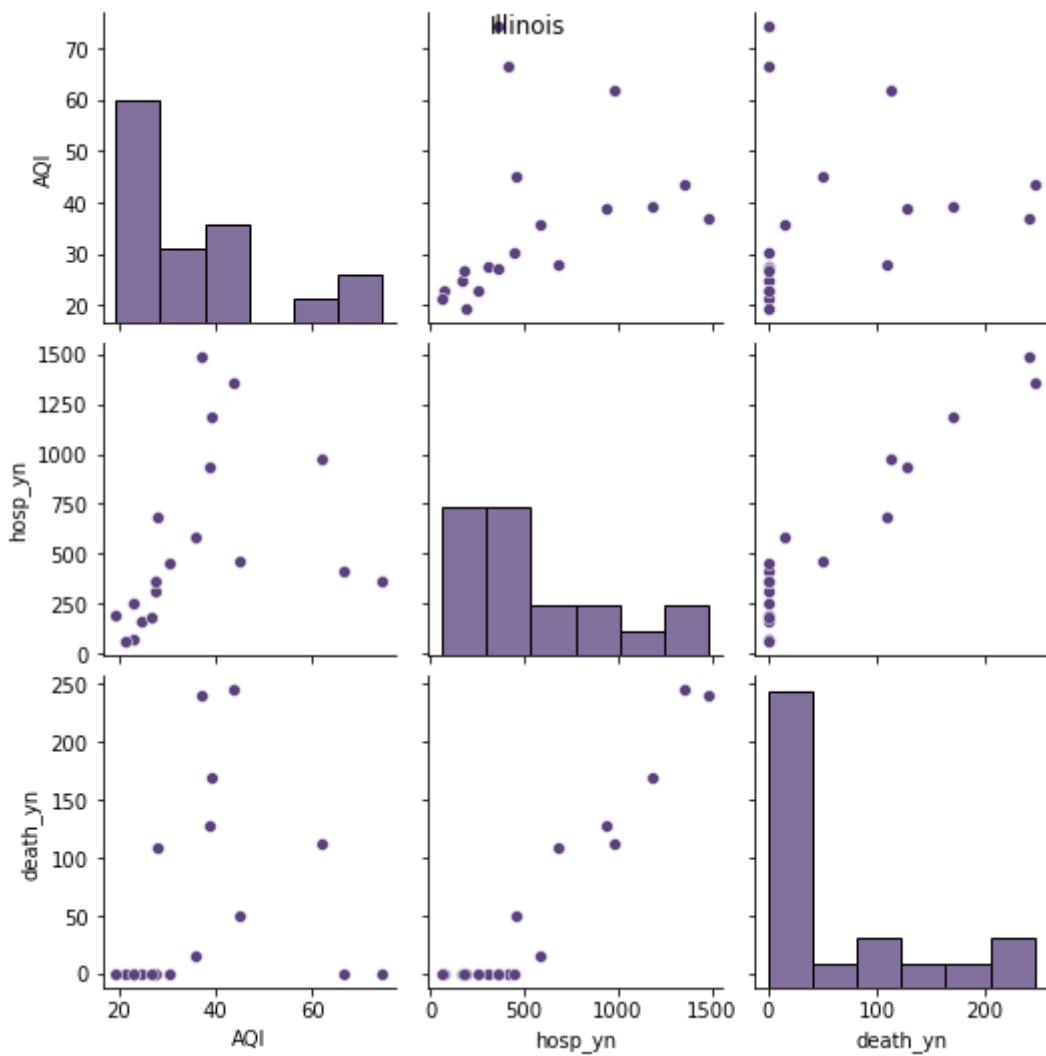
&lt;Figure size 1080x504 with 0 Axes&gt;



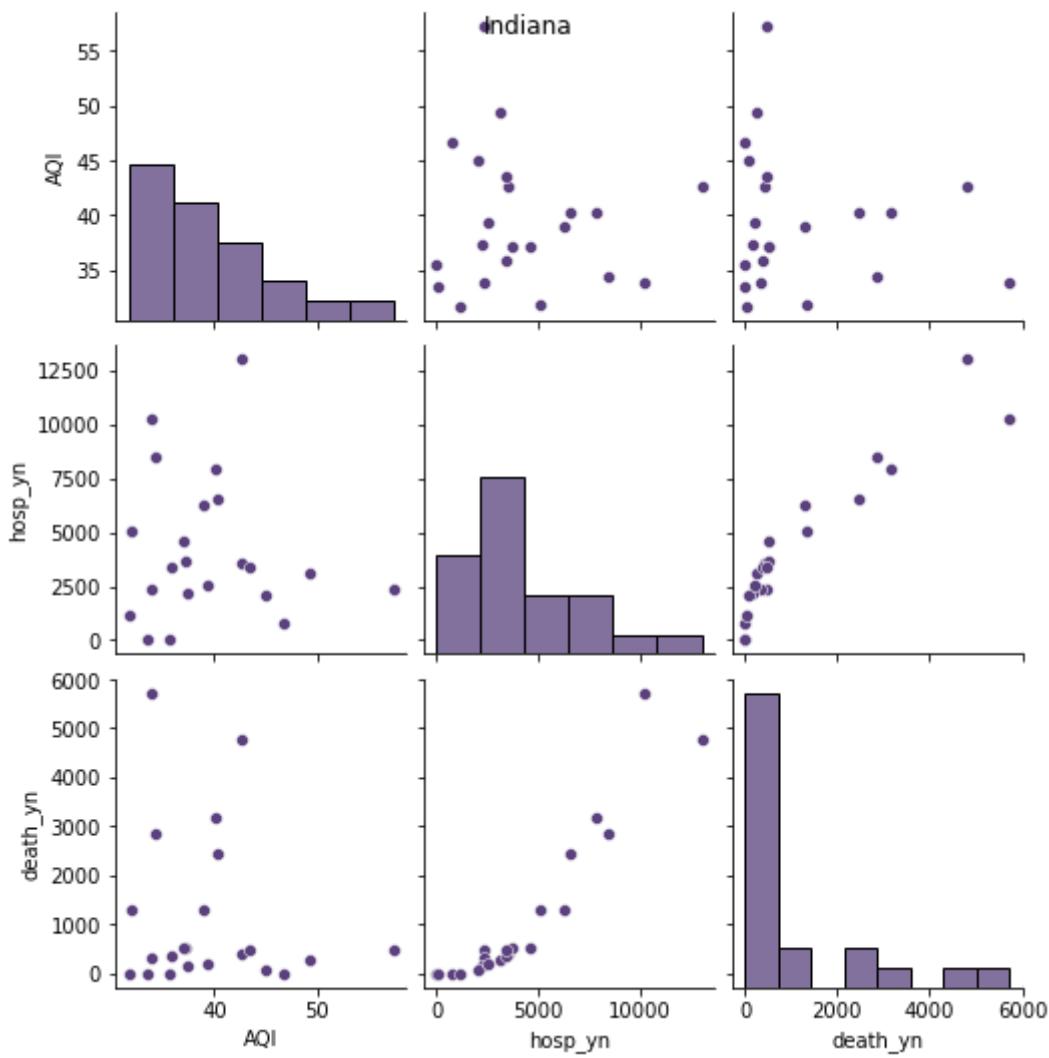
&lt;Figure size 1080x504 with 0 Axes&gt;



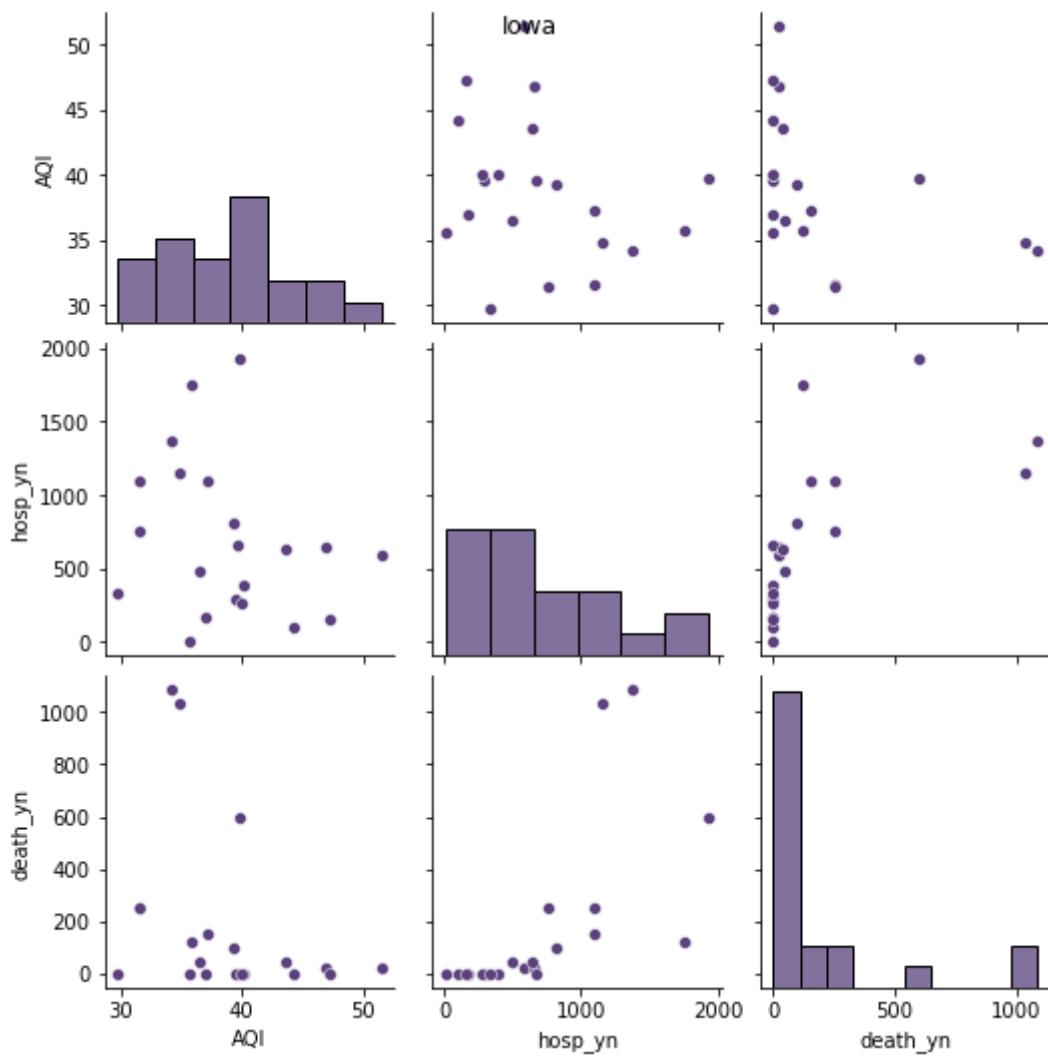
<Figure size 1080x504 with 0 Axes>



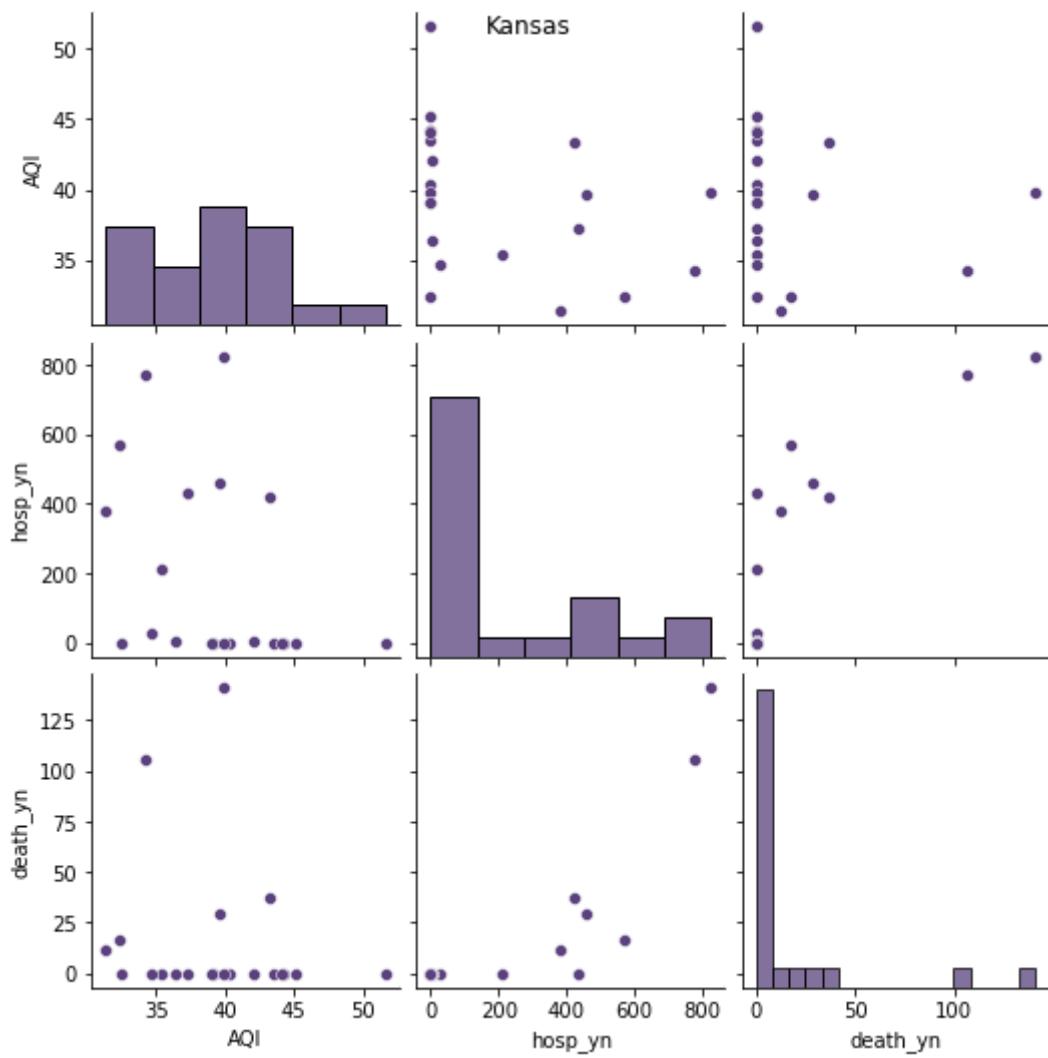
&lt;Figure size 1080x504 with 0 Axes&gt;



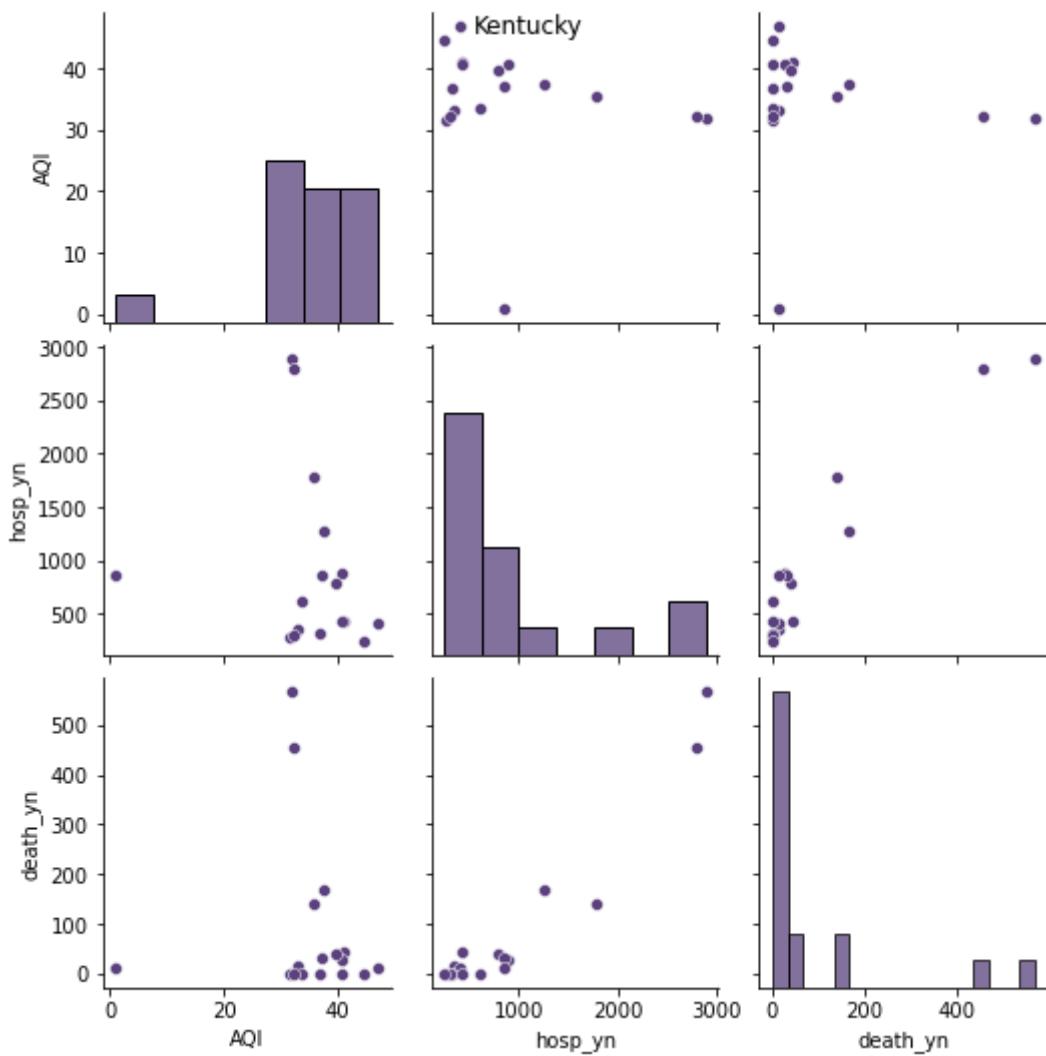
<Figure size 1080x504 with 0 Axes>



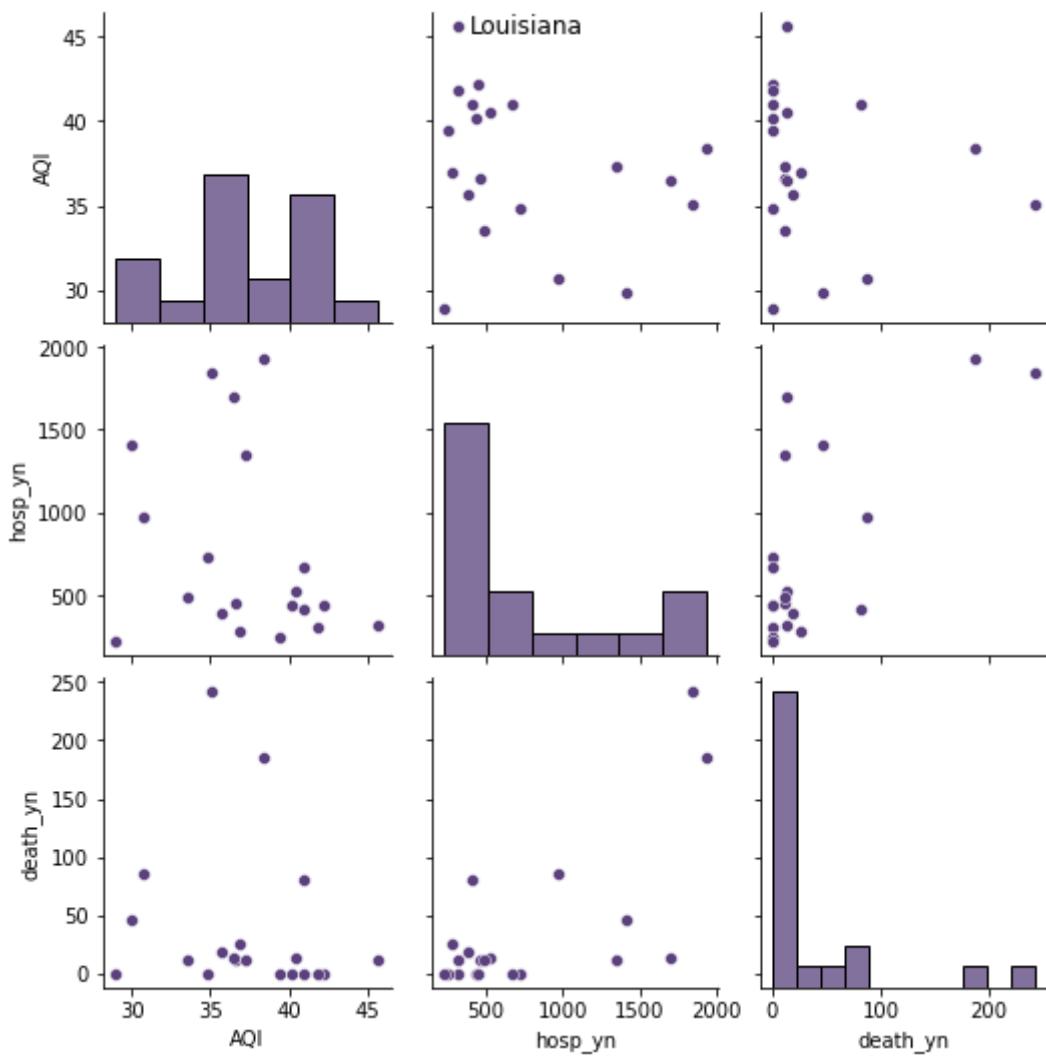
<Figure size 1080x504 with 0 Axes>



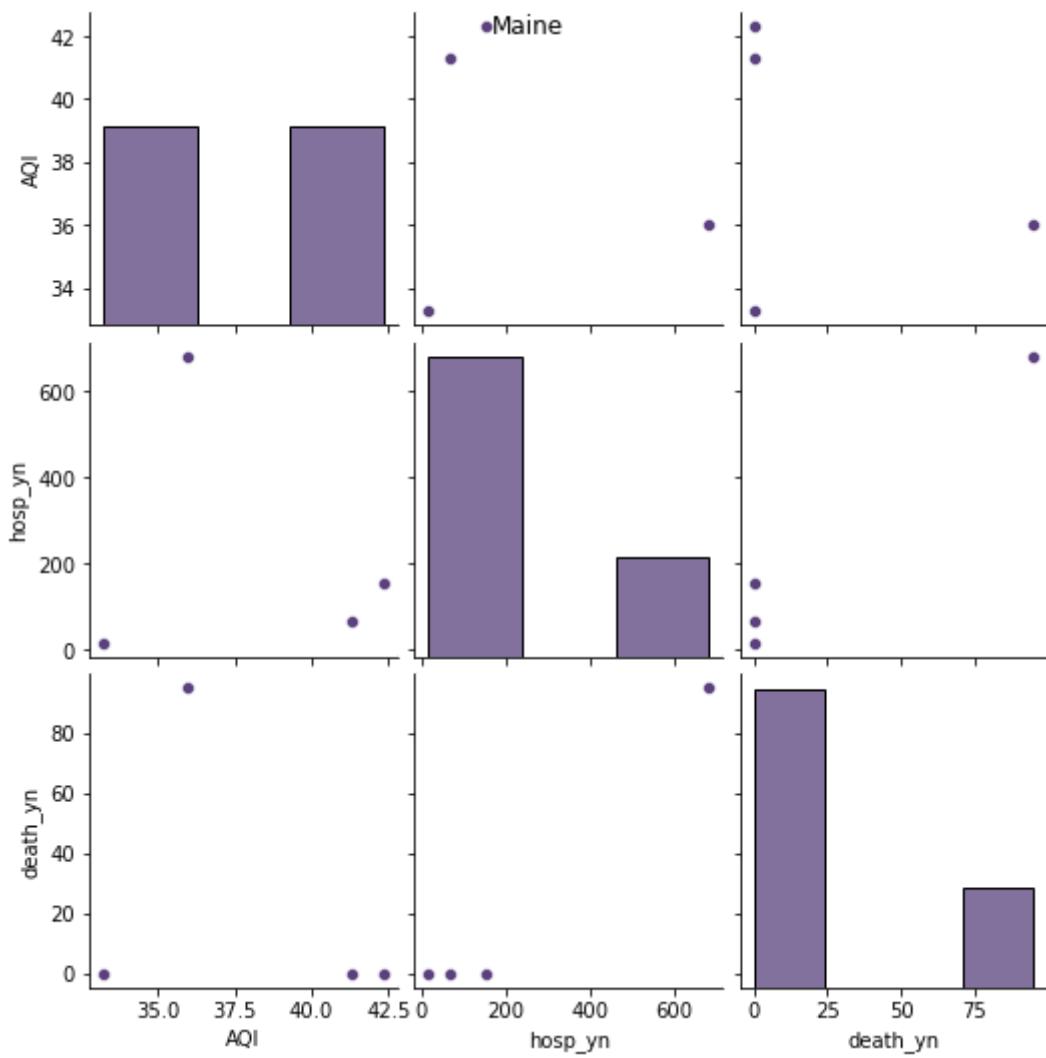
<Figure size 1080x504 with 0 Axes>



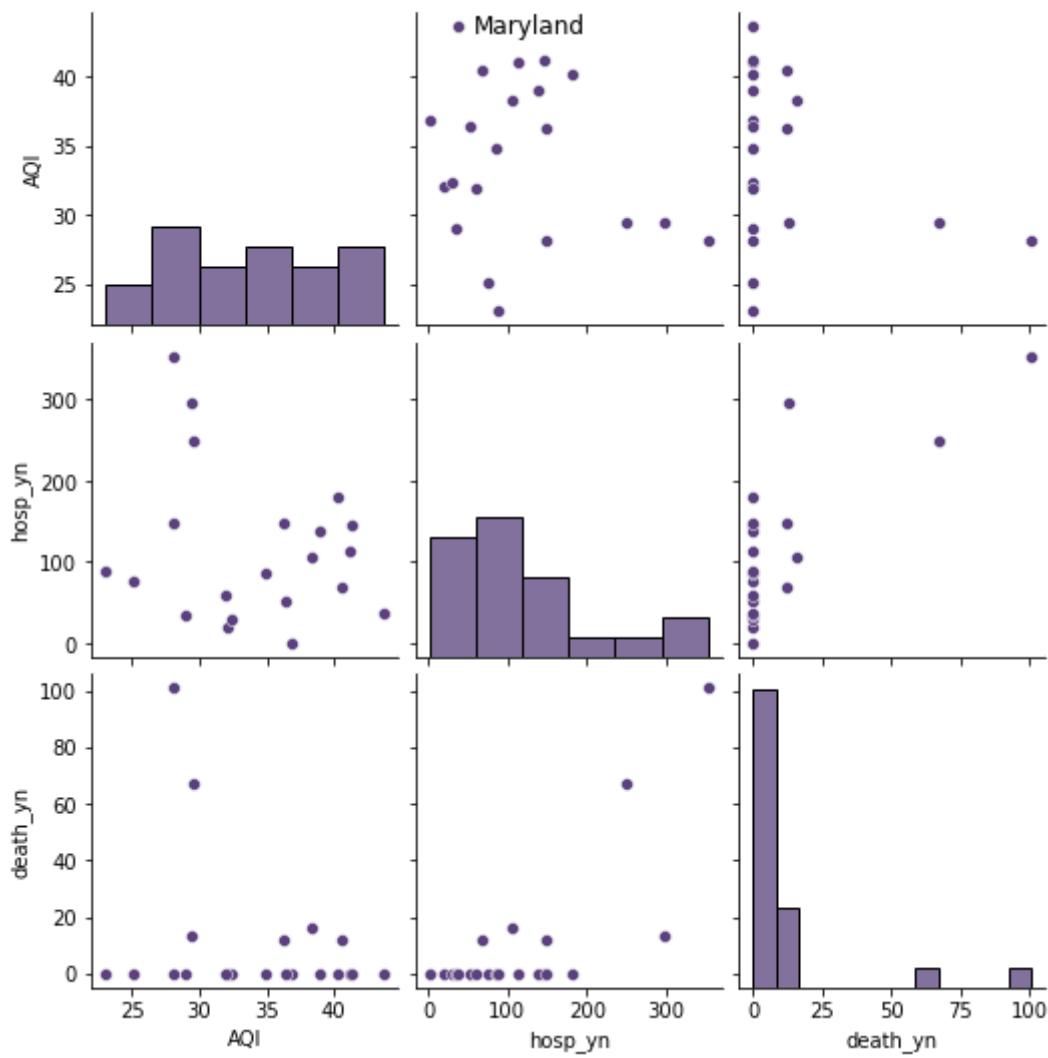
&lt;Figure size 1080x504 with 0 Axes&gt;



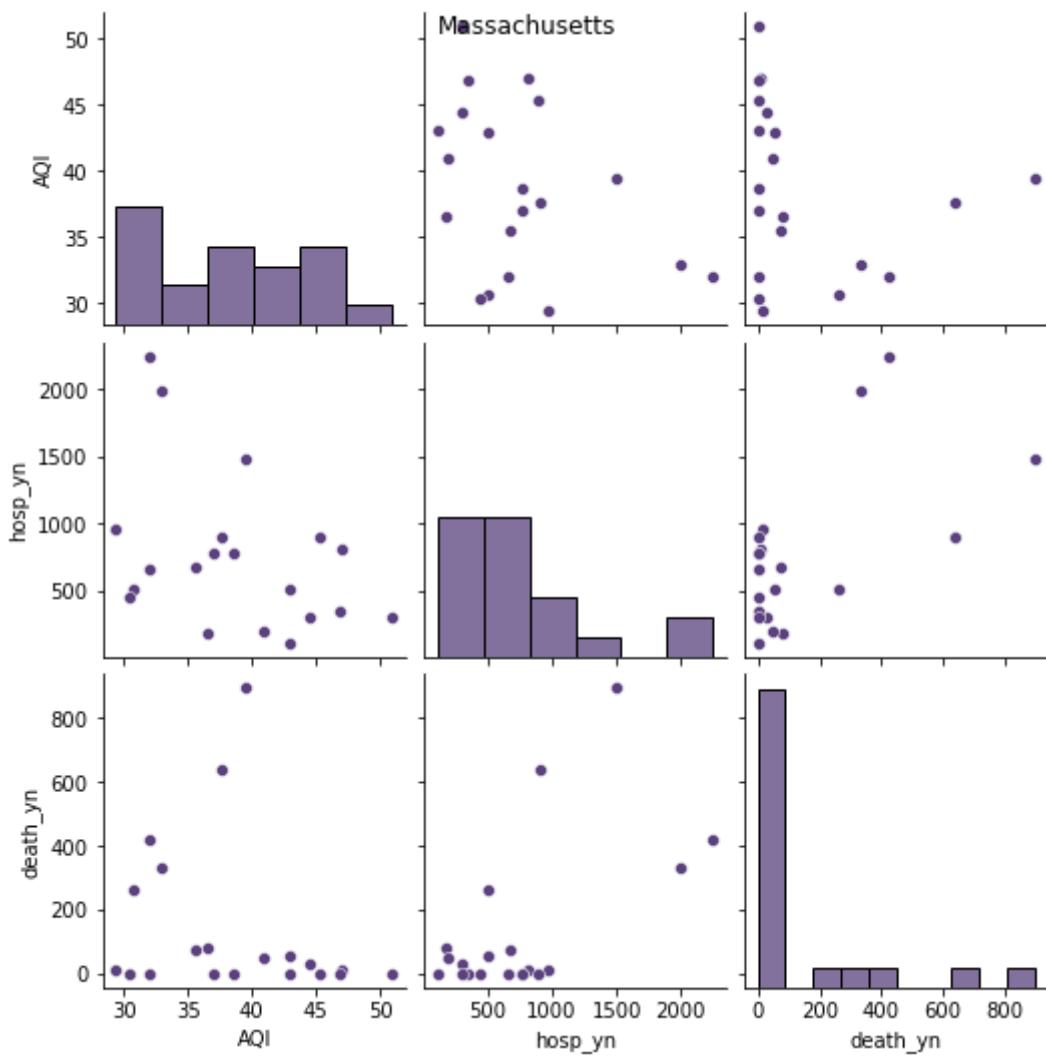
<Figure size 1080x504 with 0 Axes>



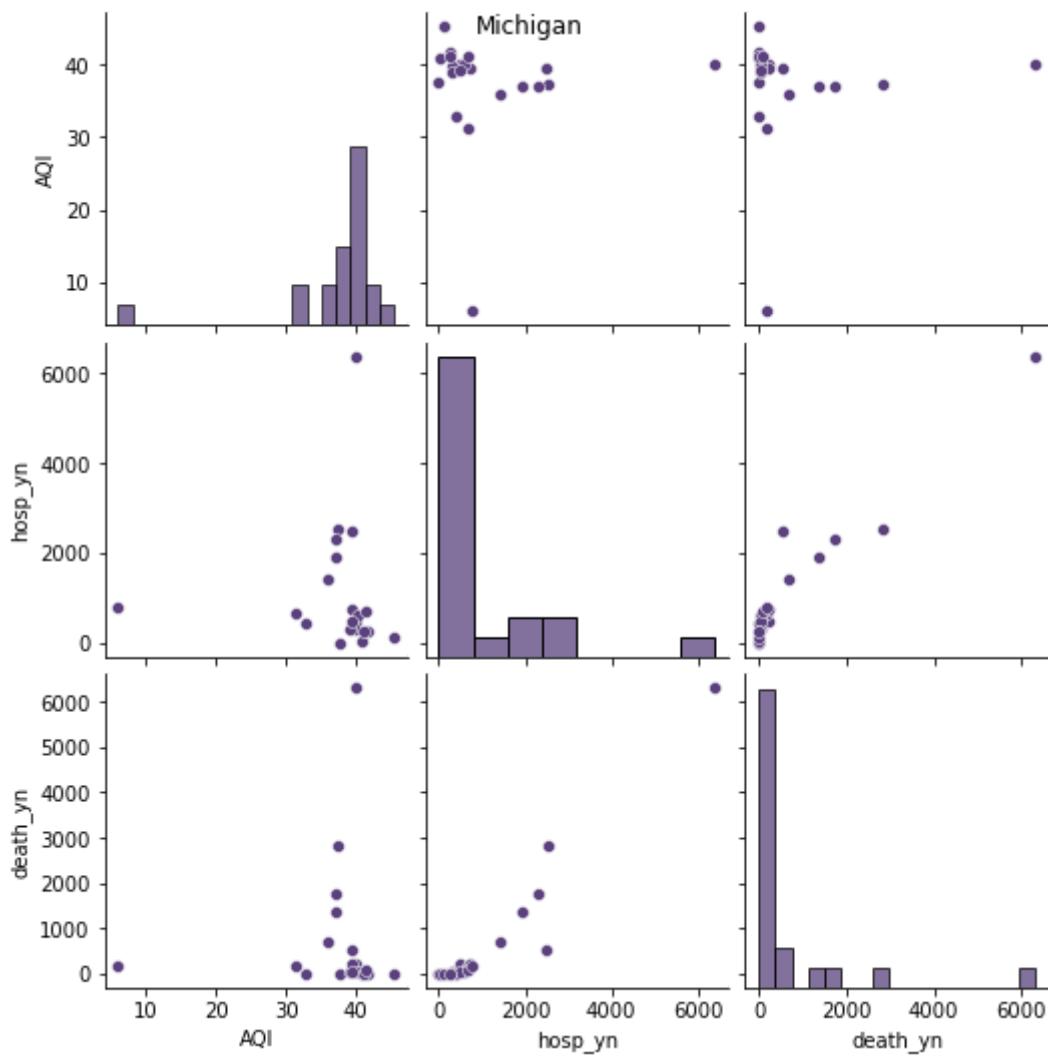
<Figure size 1080x504 with 0 Axes>



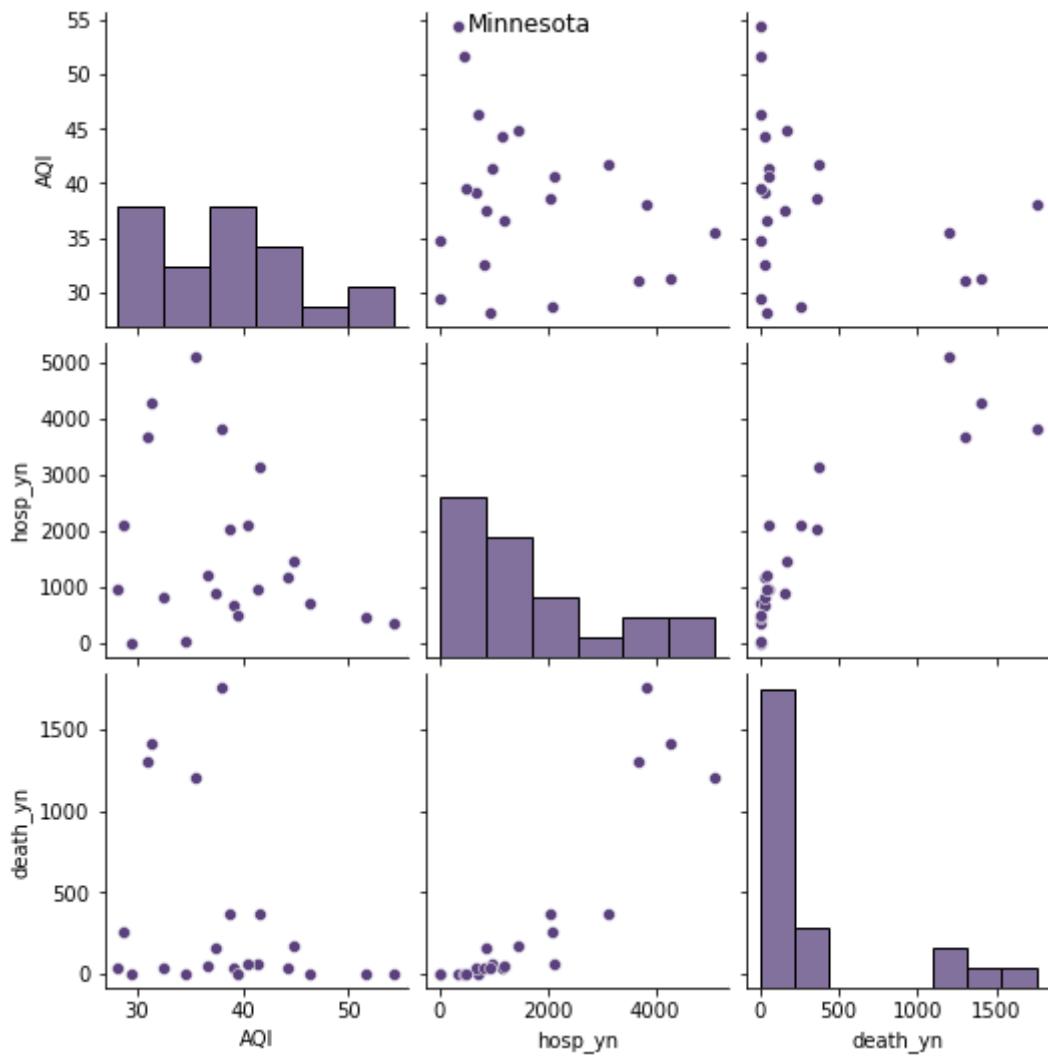
<Figure size 1080x504 with 0 Axes>



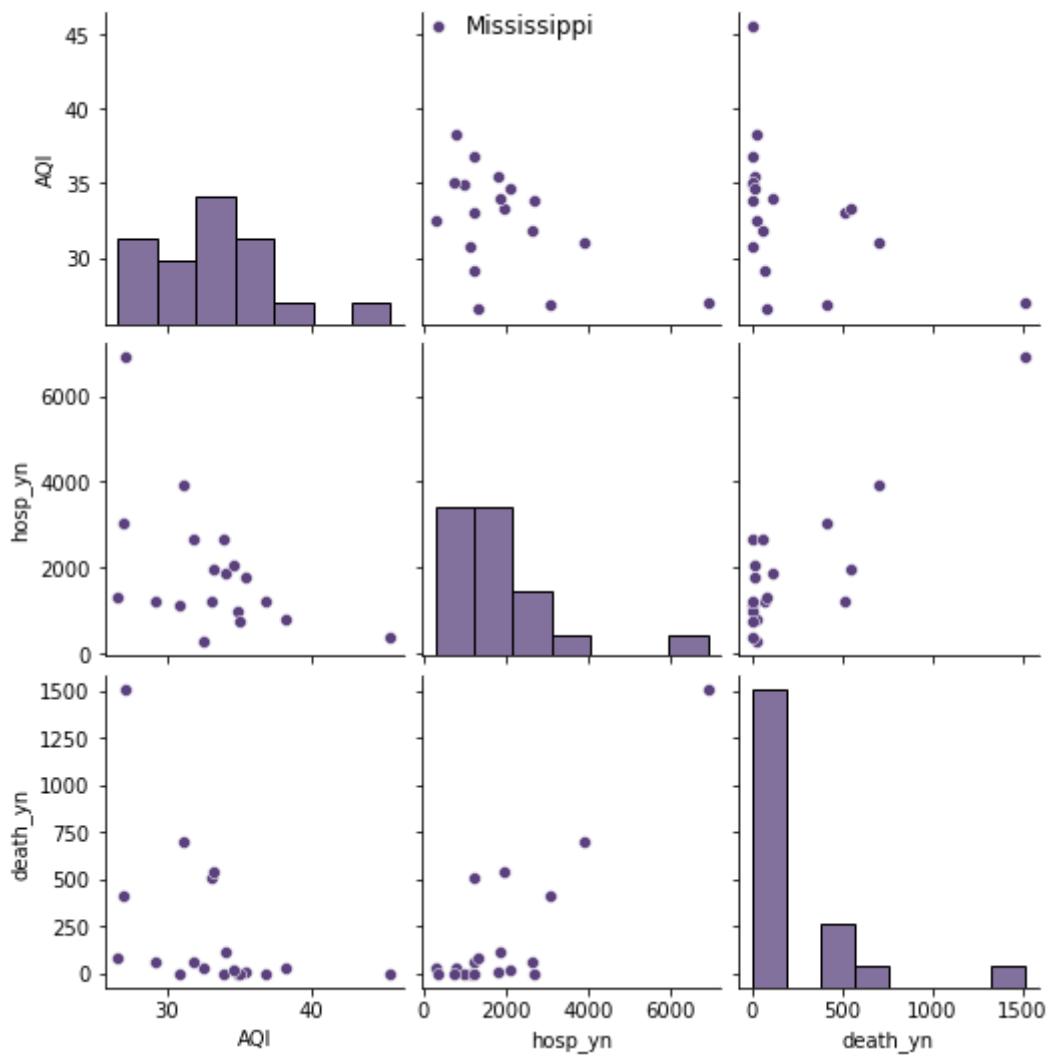
<Figure size 1080x504 with 0 Axes>



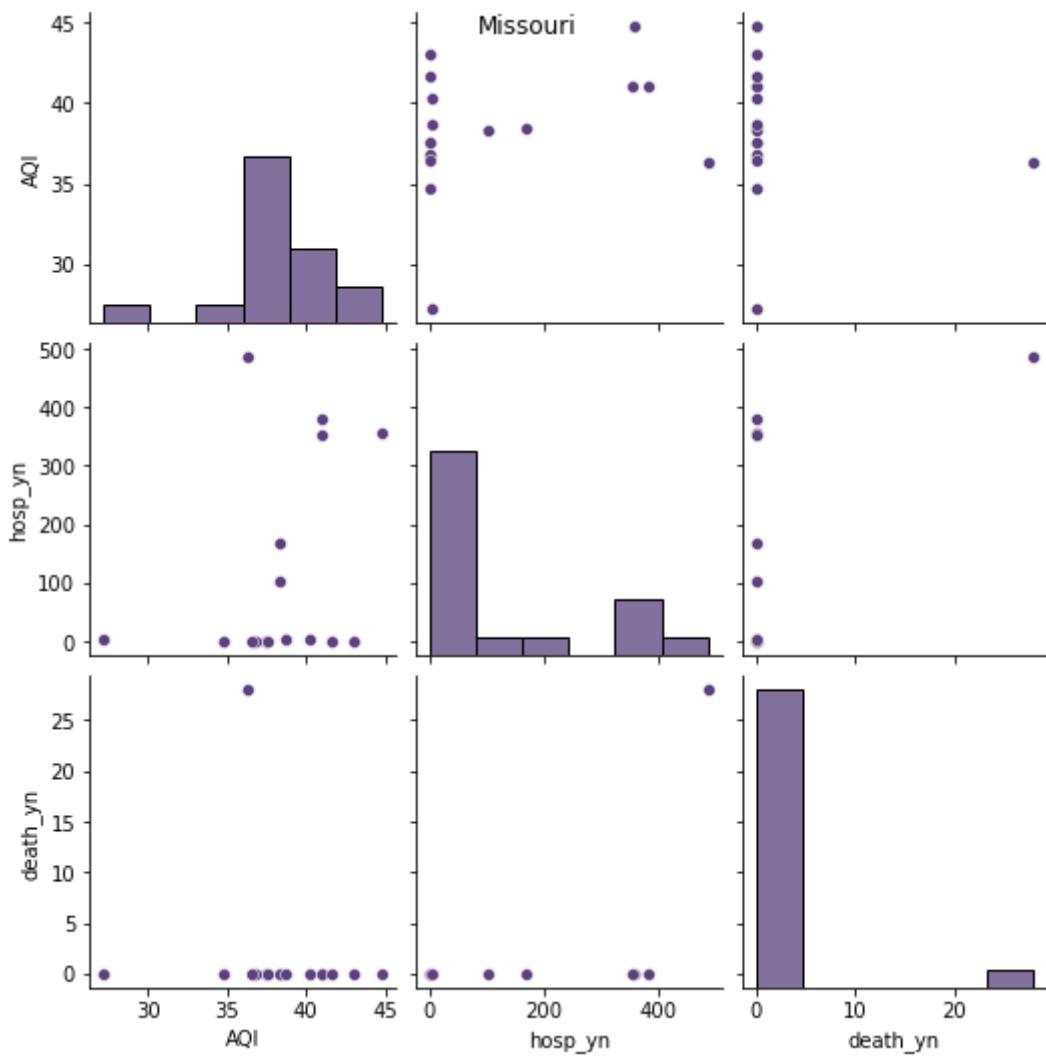
<Figure size 1080x504 with 0 Axes>



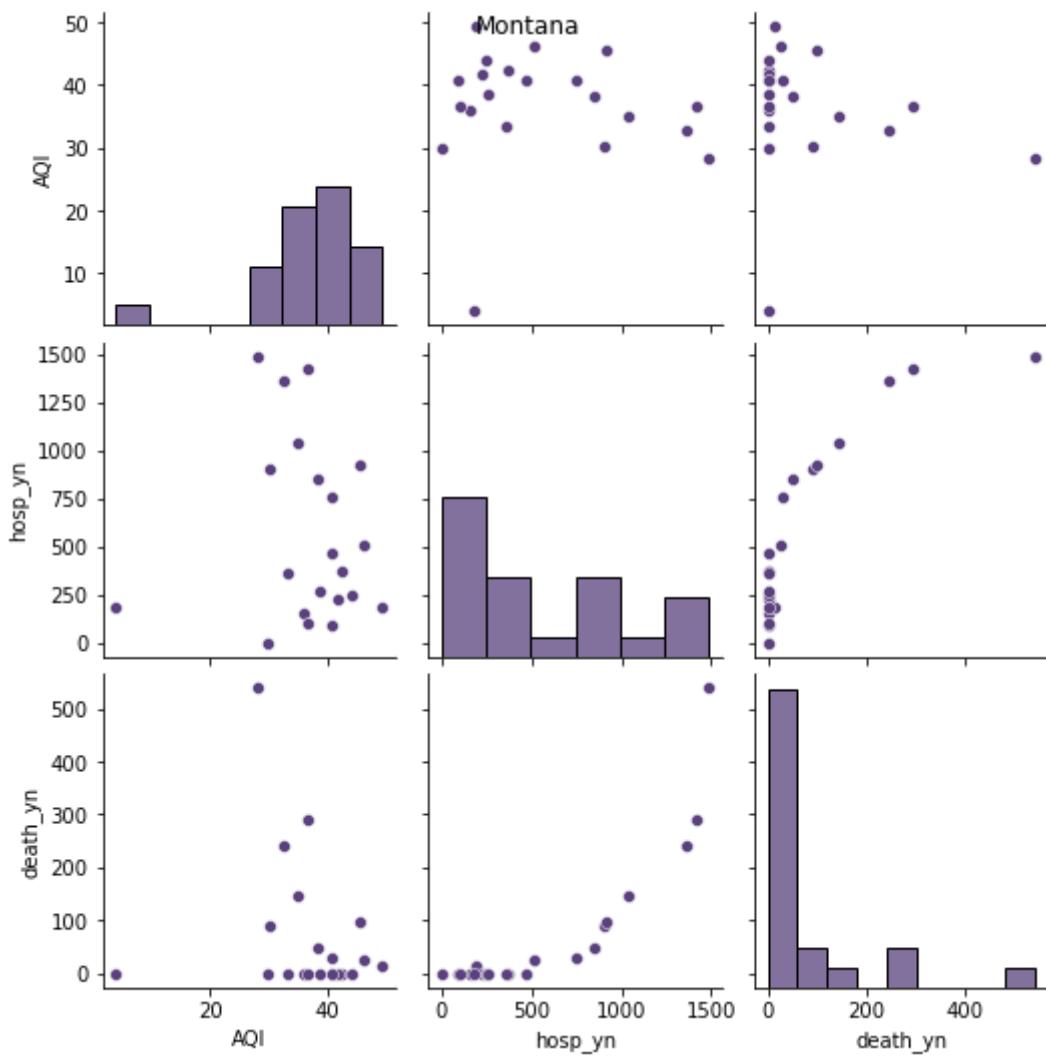
<Figure size 1080x504 with 0 Axes>



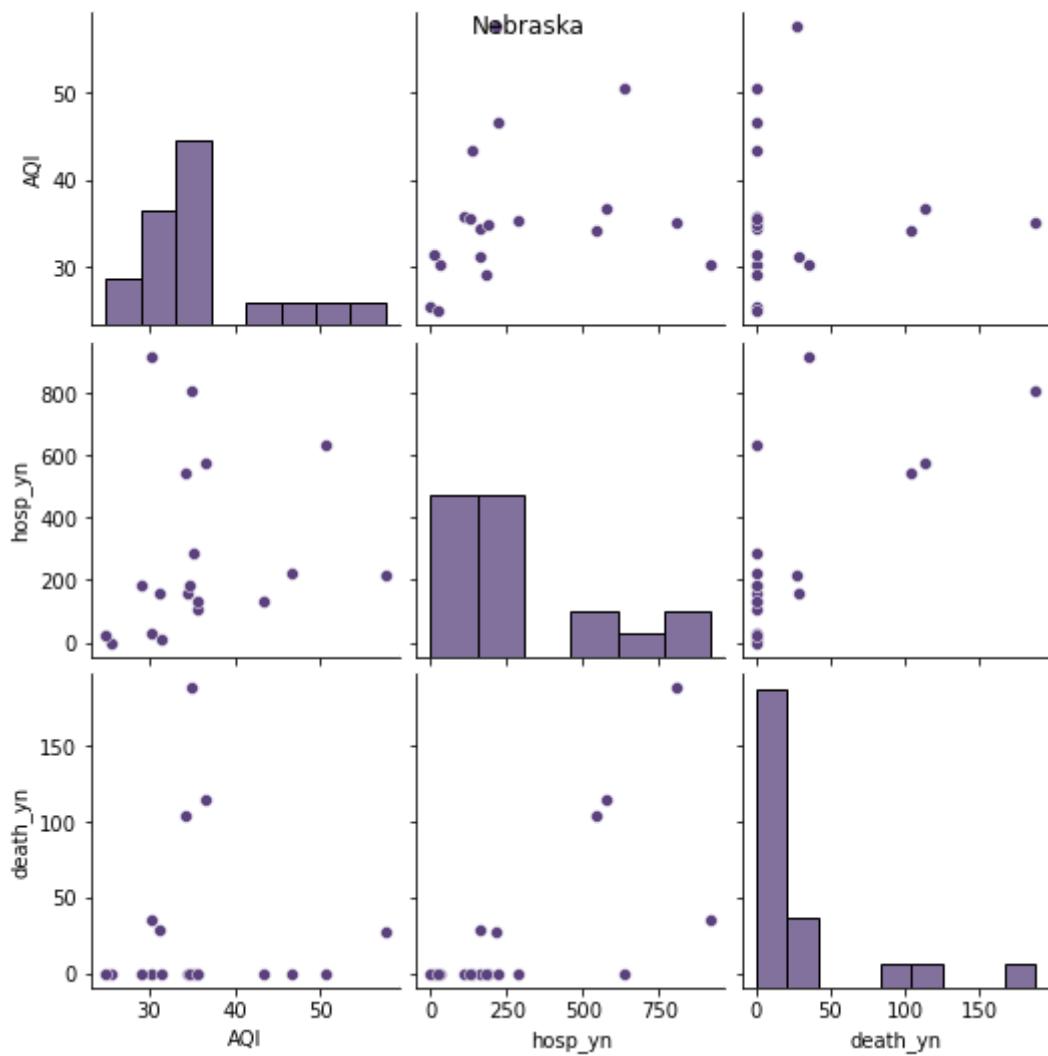
&lt;Figure size 1080x504 with 0 Axes&gt;



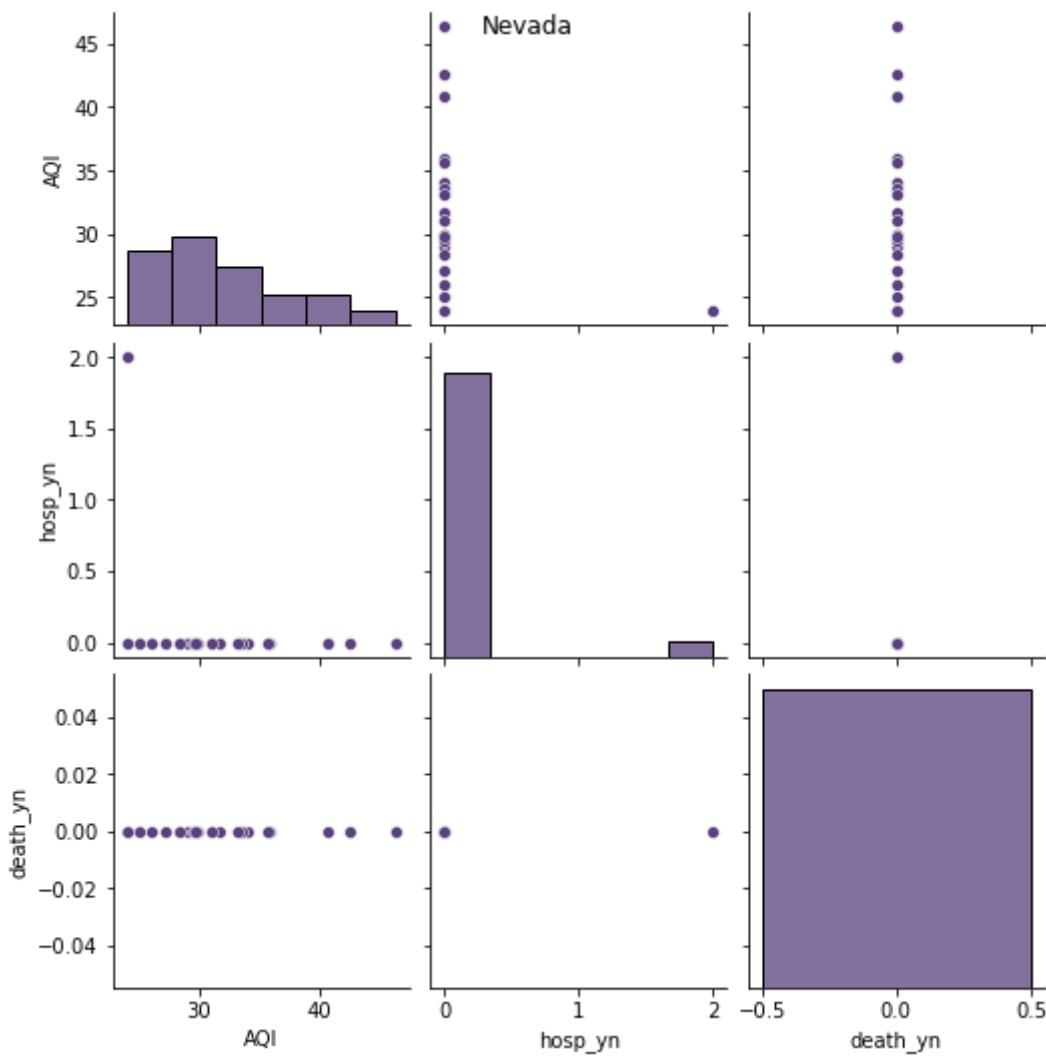
<Figure size 1080x504 with 0 Axes>



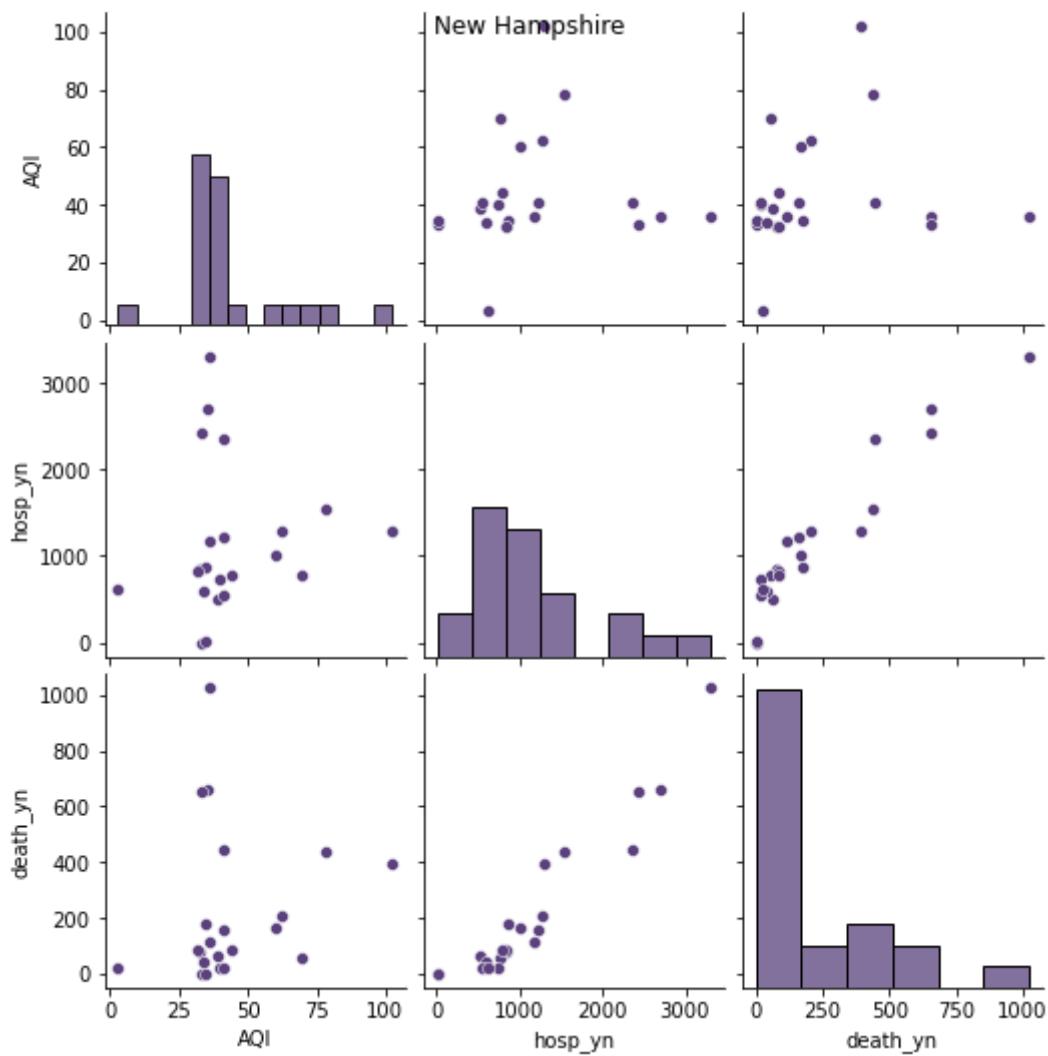
<Figure size 1080x504 with 0 Axes>



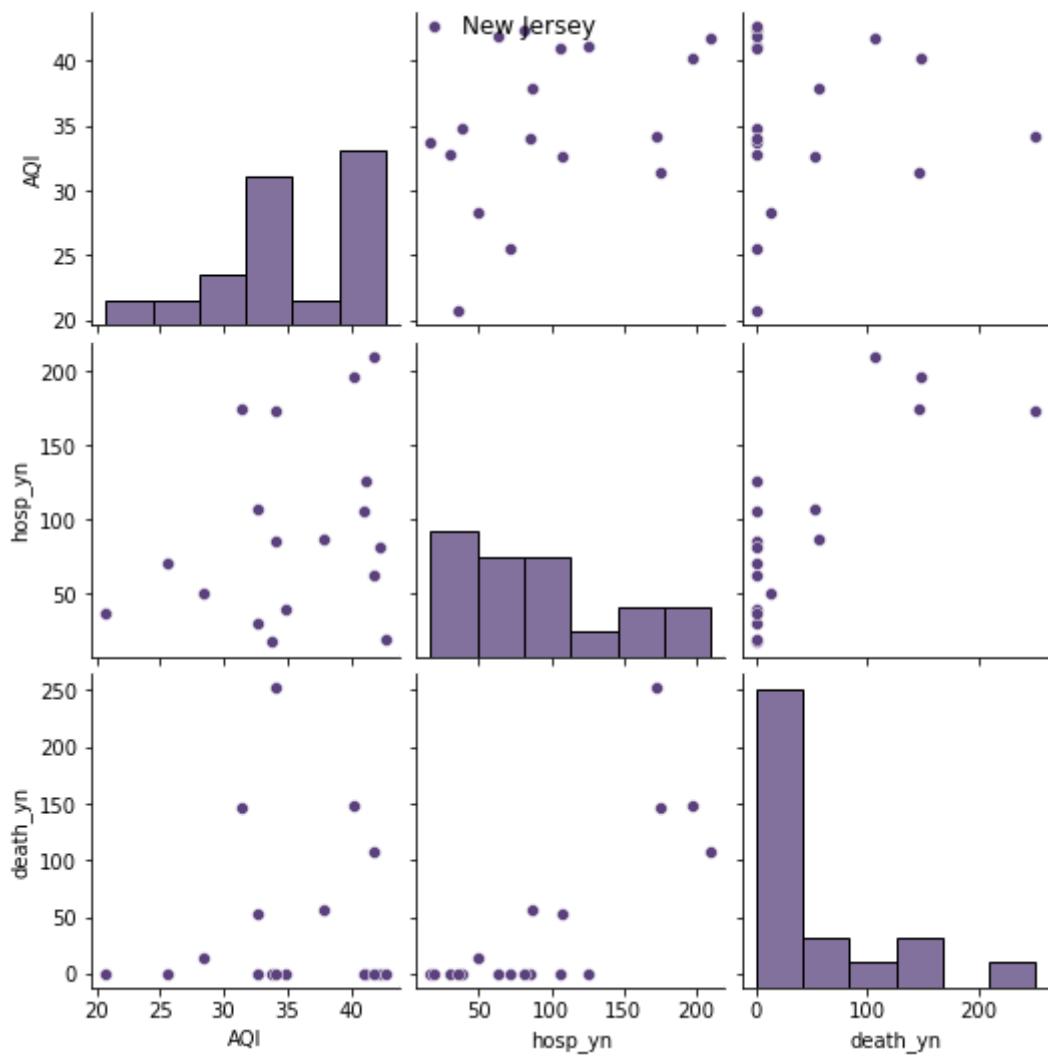
<Figure size 1080x504 with 0 Axes>



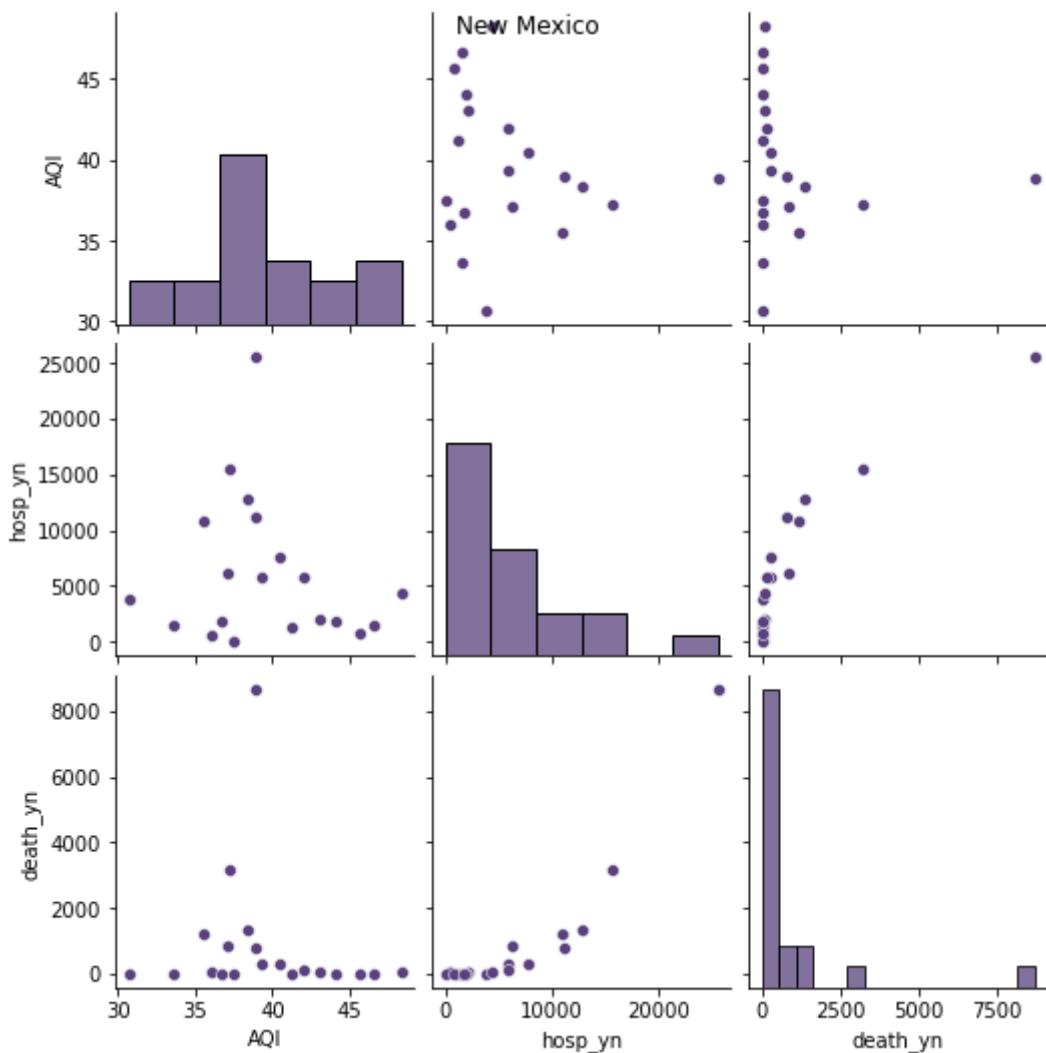
<Figure size 1080x504 with 0 Axes>



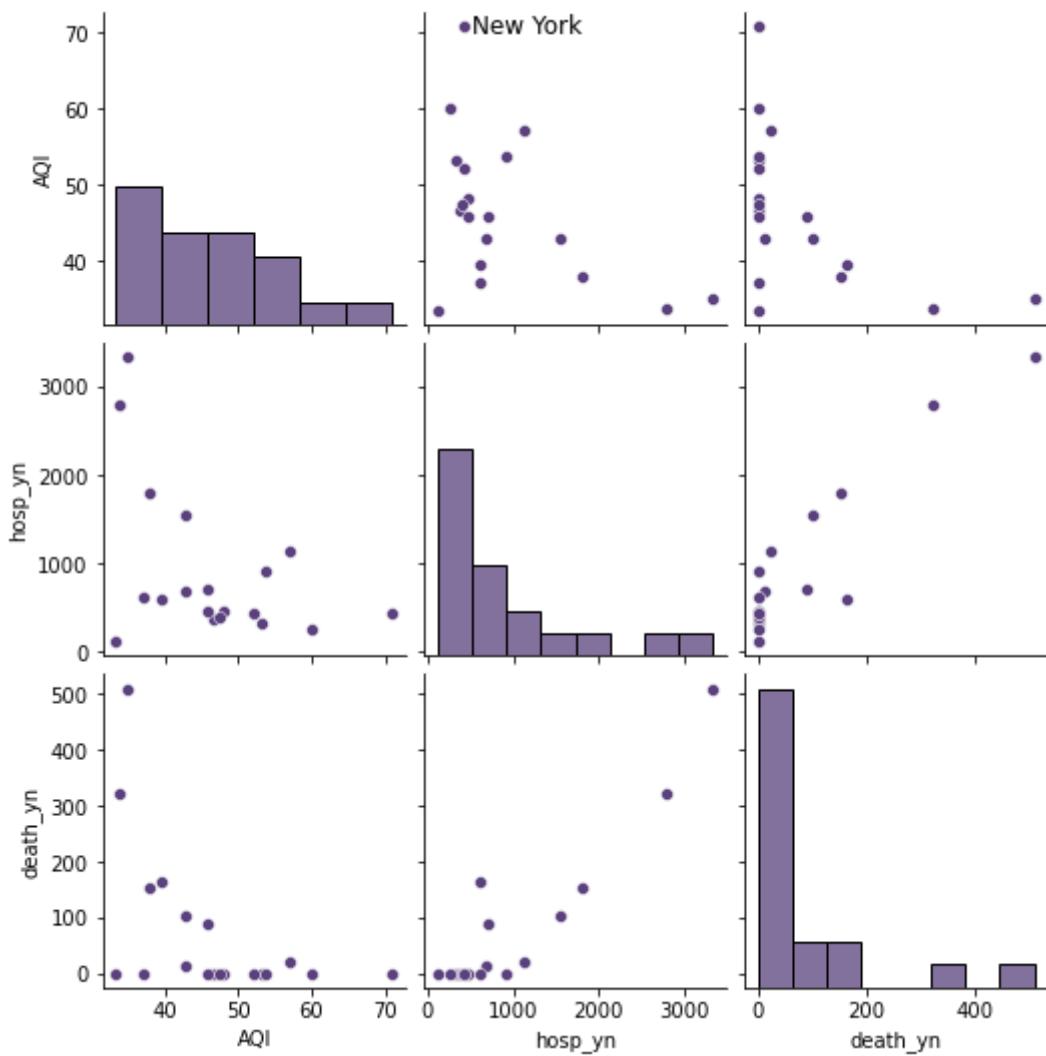
&lt;Figure size 1080x504 with 0 Axes&gt;



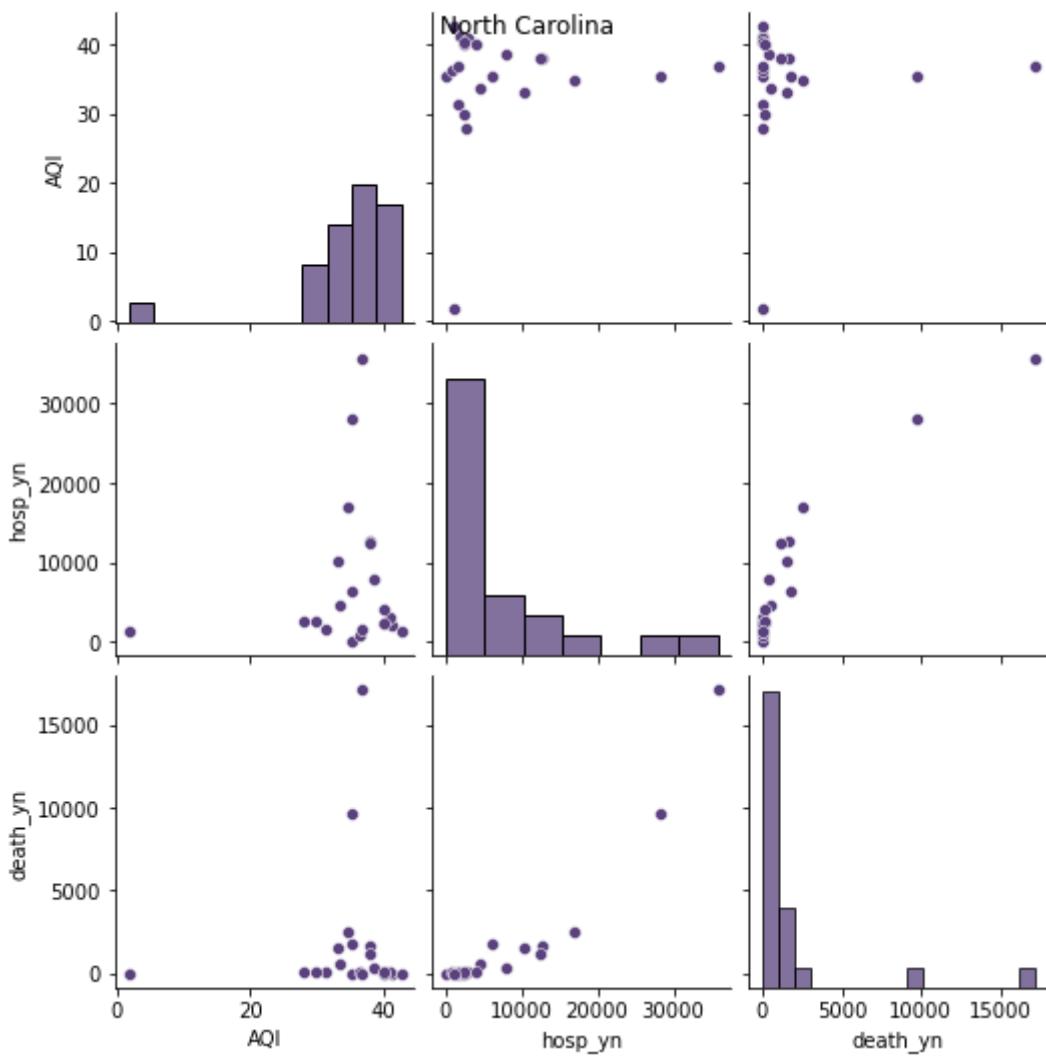
<Figure size 1080x504 with 0 Axes>



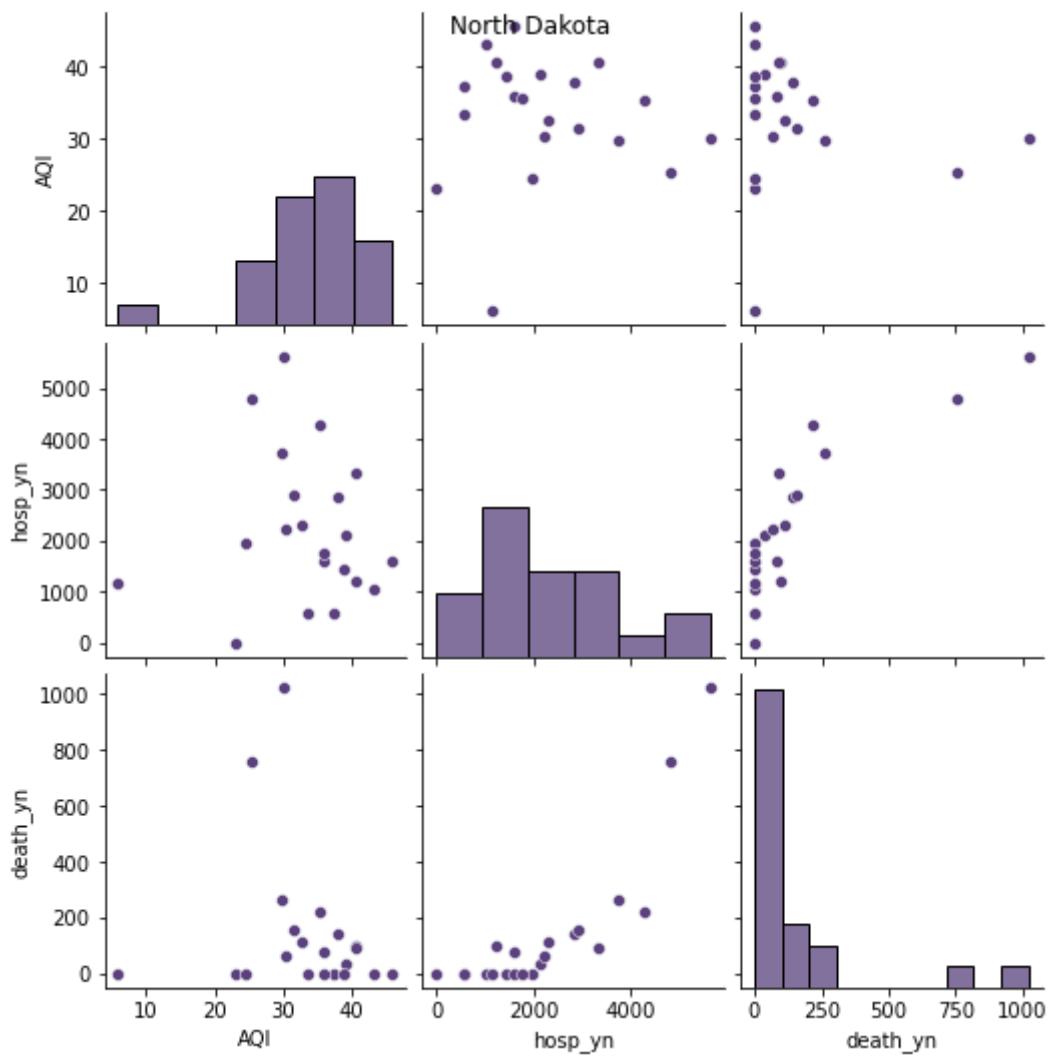
<Figure size 1080x504 with 0 Axes>



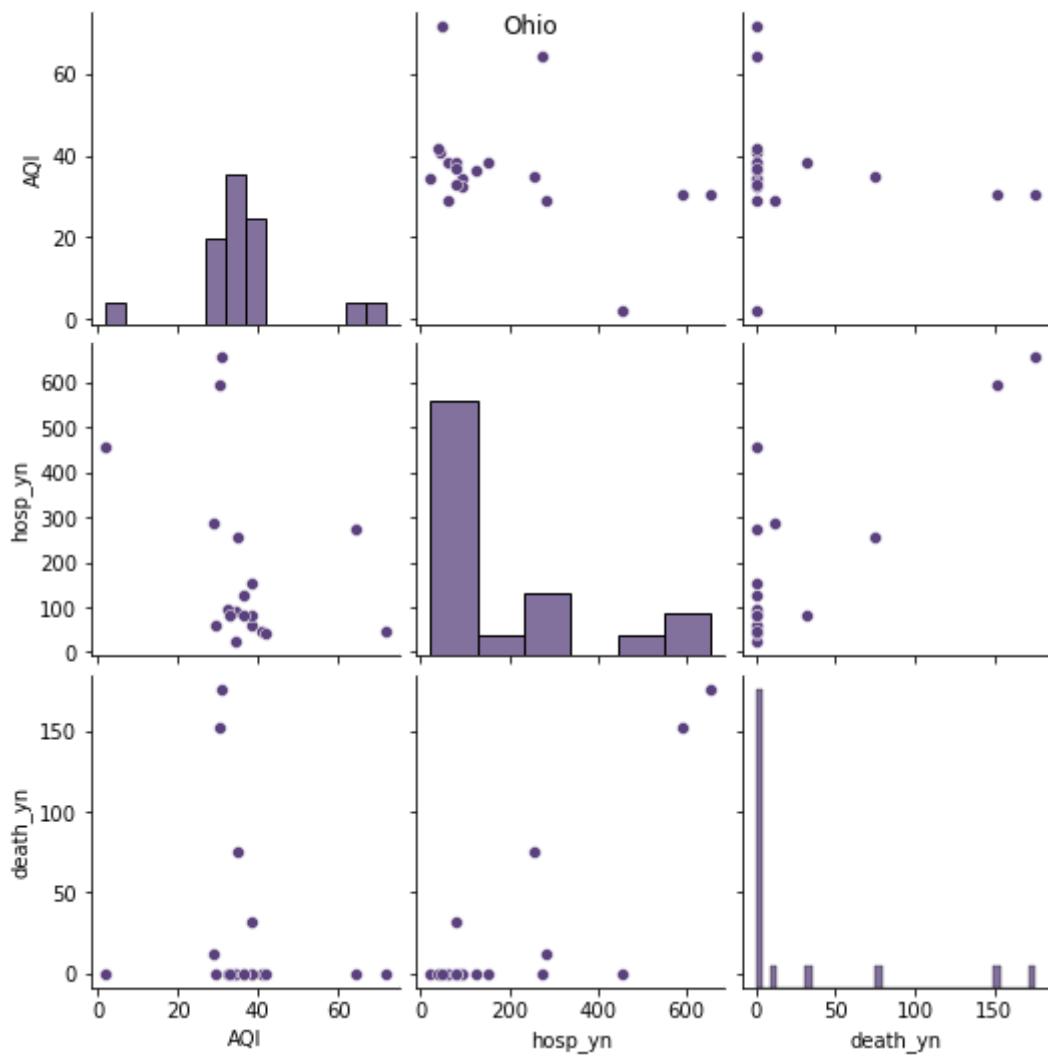
&lt;Figure size 1080x504 with 0 Axes&gt;



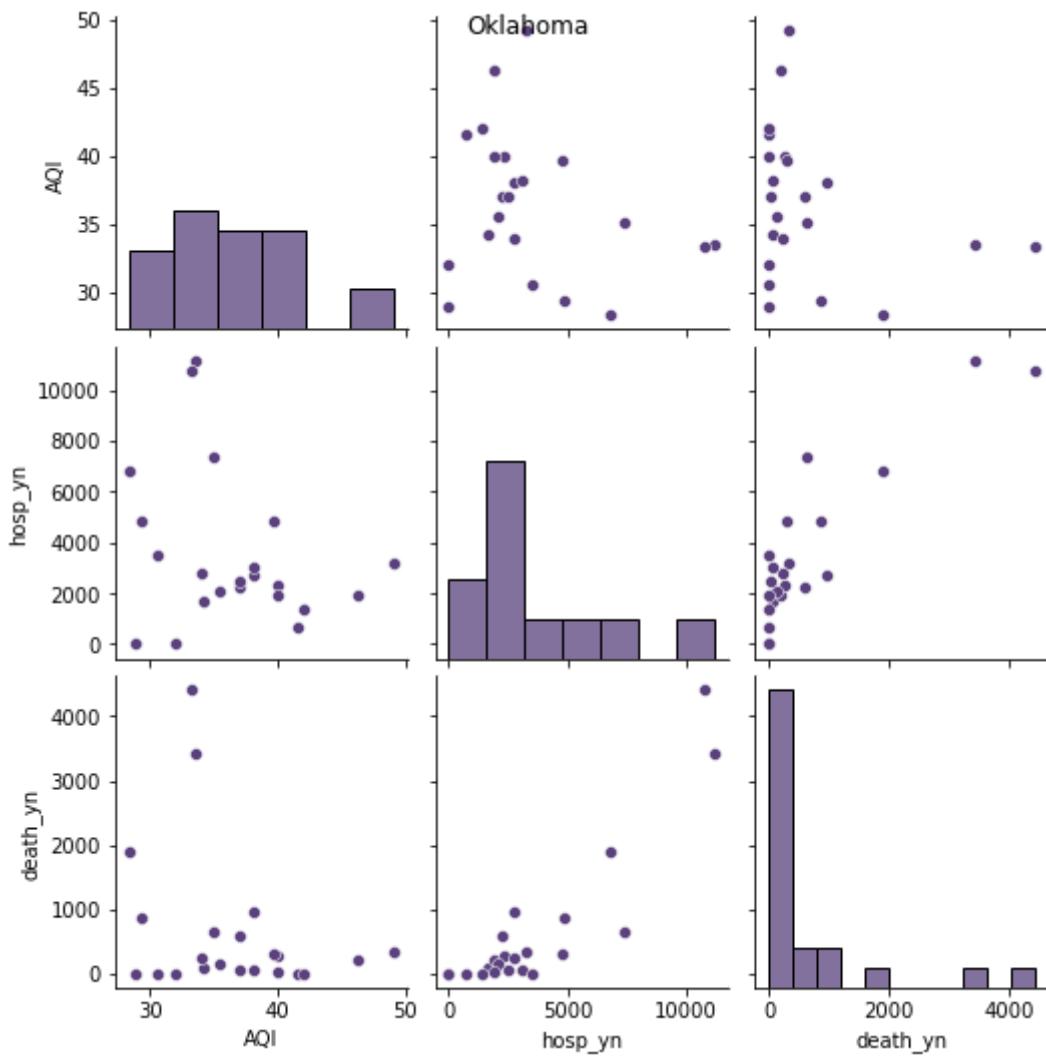
<Figure size 1080x504 with 0 Axes>



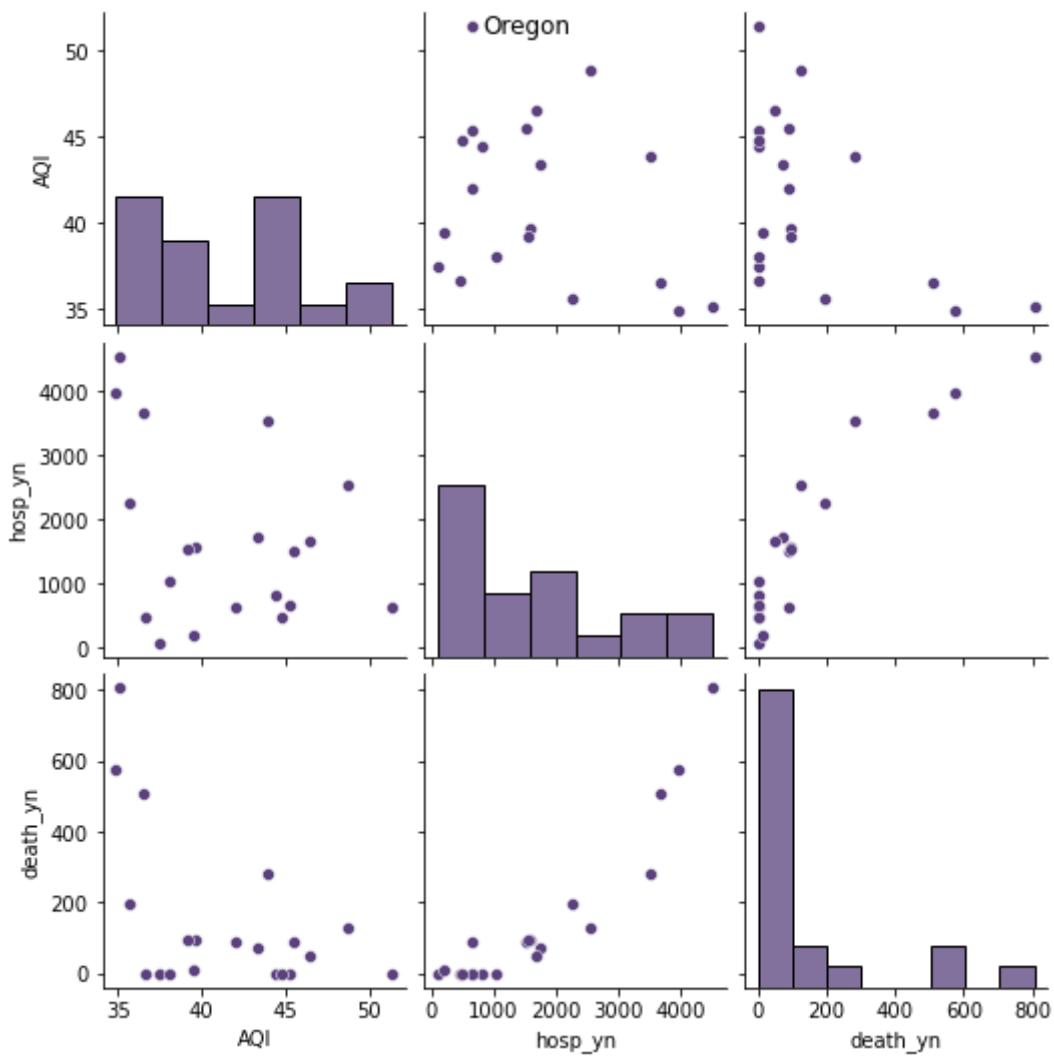
<Figure size 1080x504 with 0 Axes>



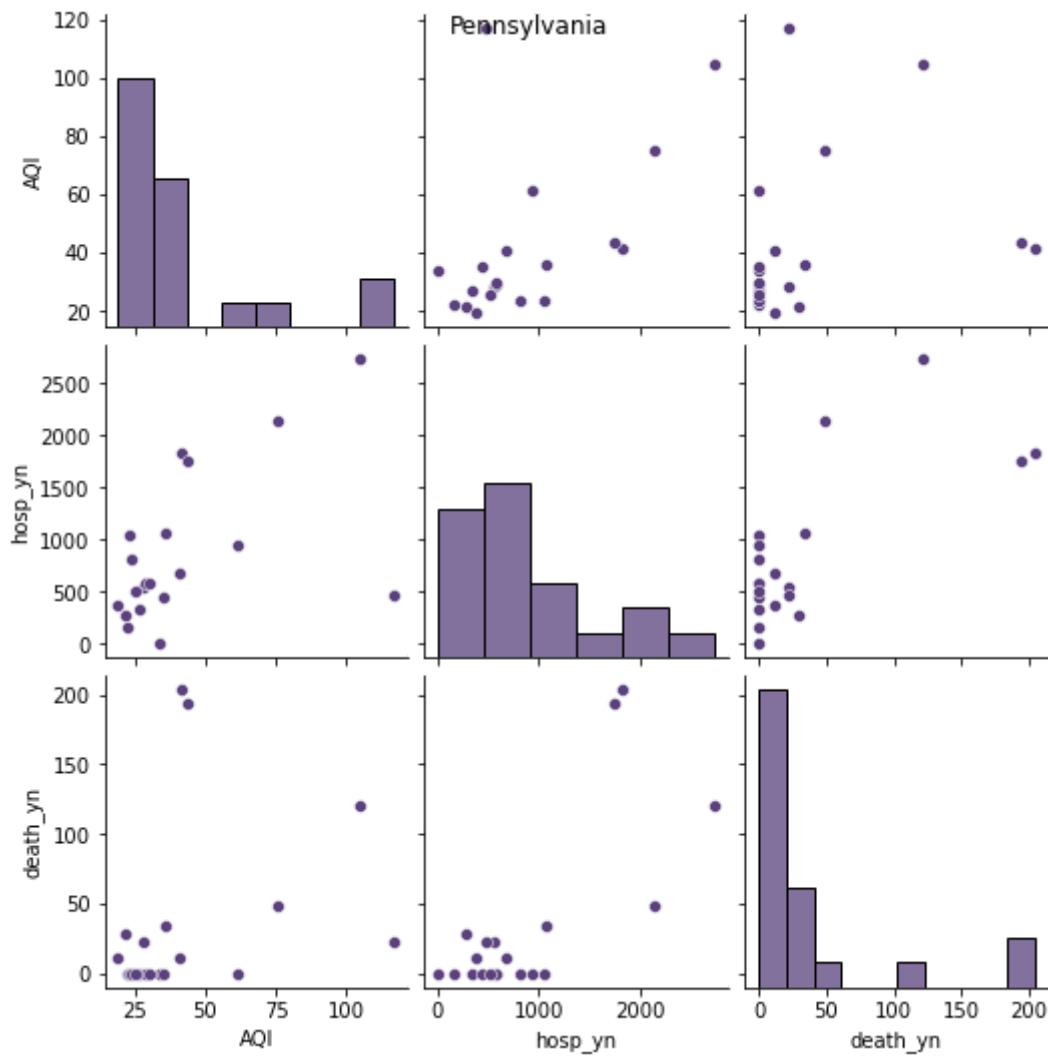
<Figure size 1080x504 with 0 Axes>



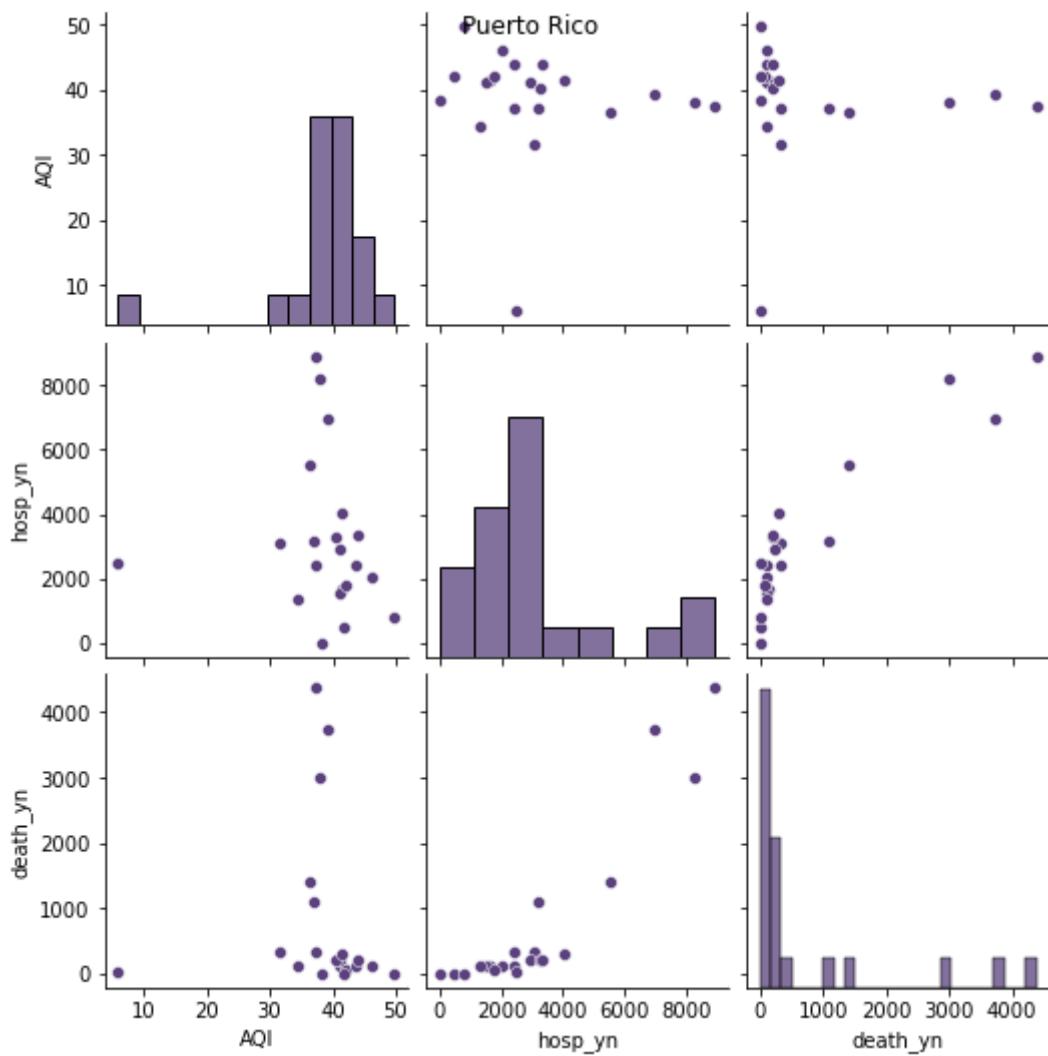
<Figure size 1080x504 with 0 Axes>



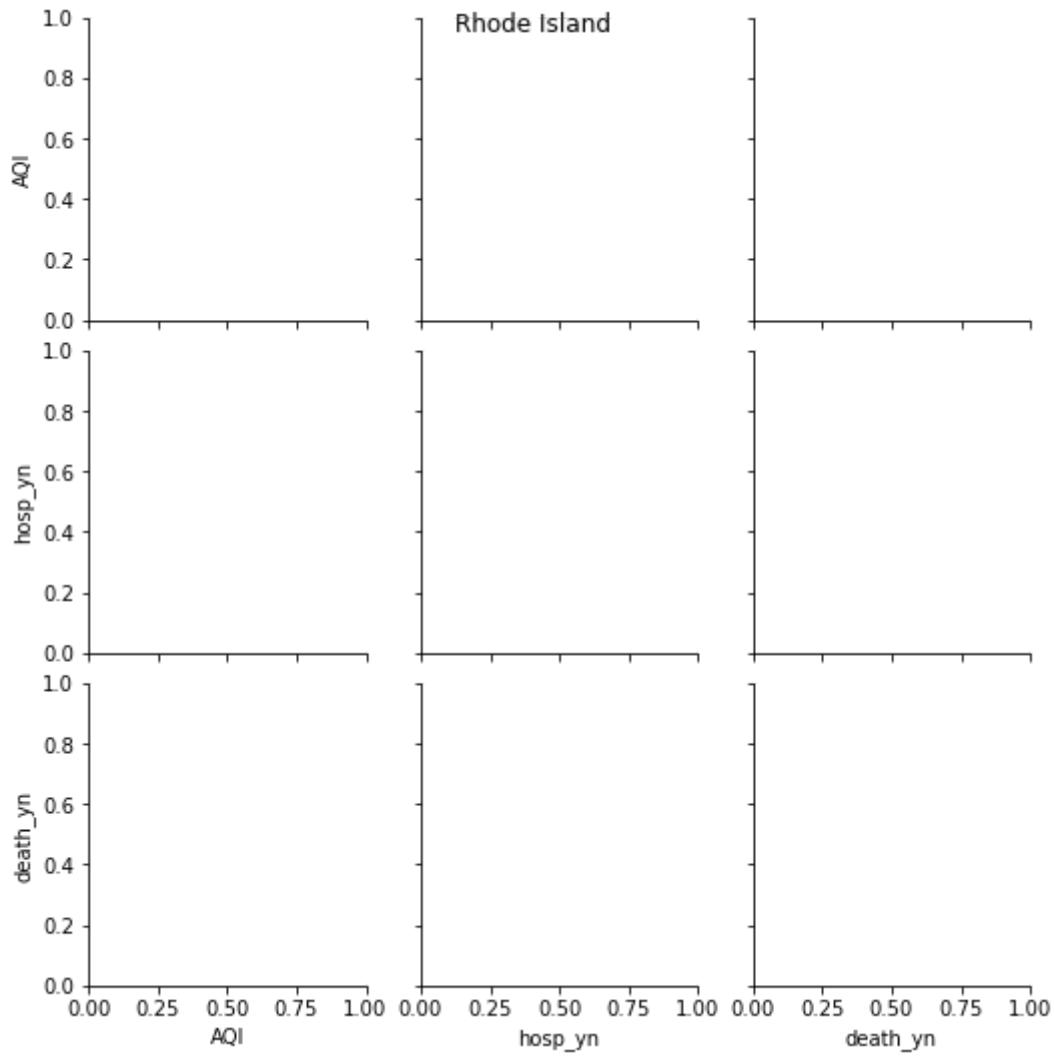
&lt;Figure size 1080x504 with 0 Axes&gt;



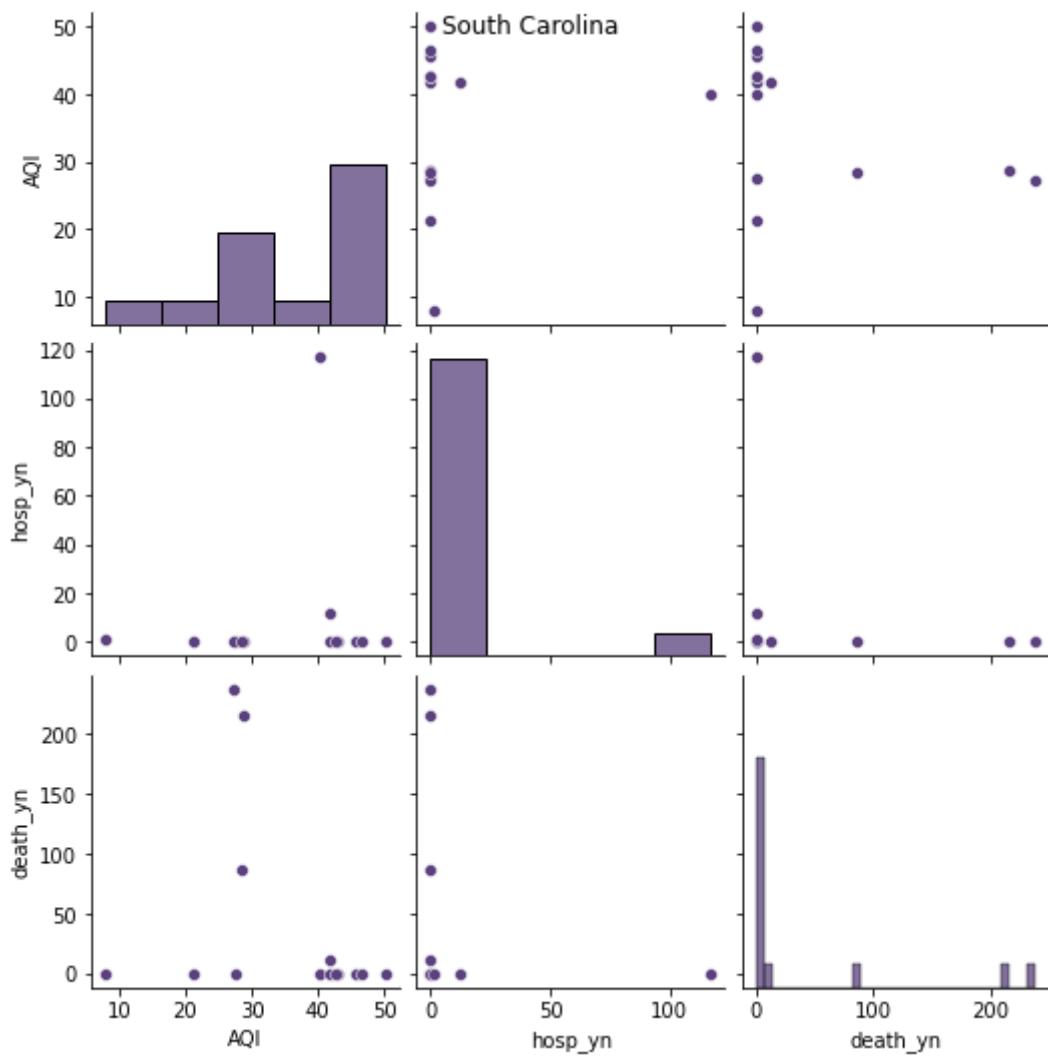
<Figure size 1080x504 with 0 Axes>



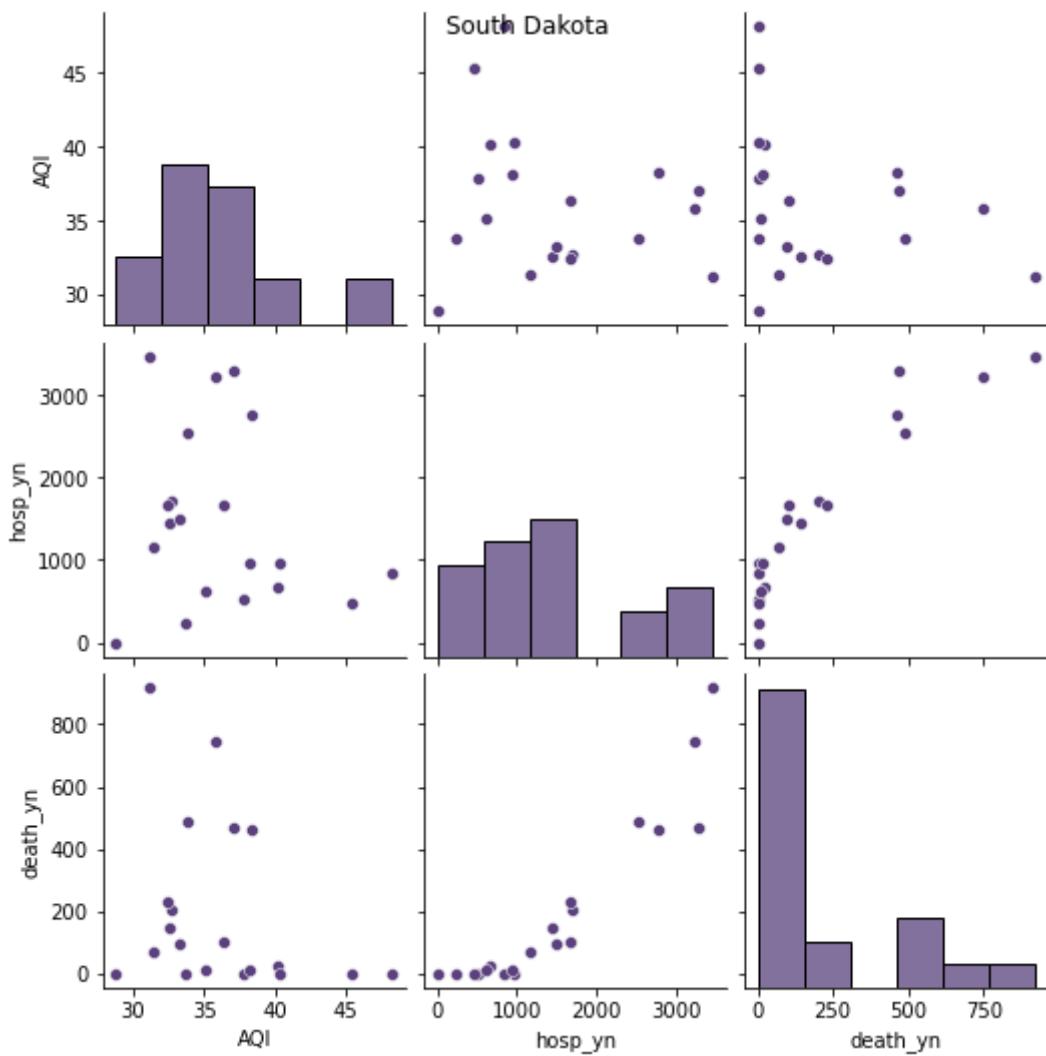
<Figure size 1080x504 with 0 Axes>



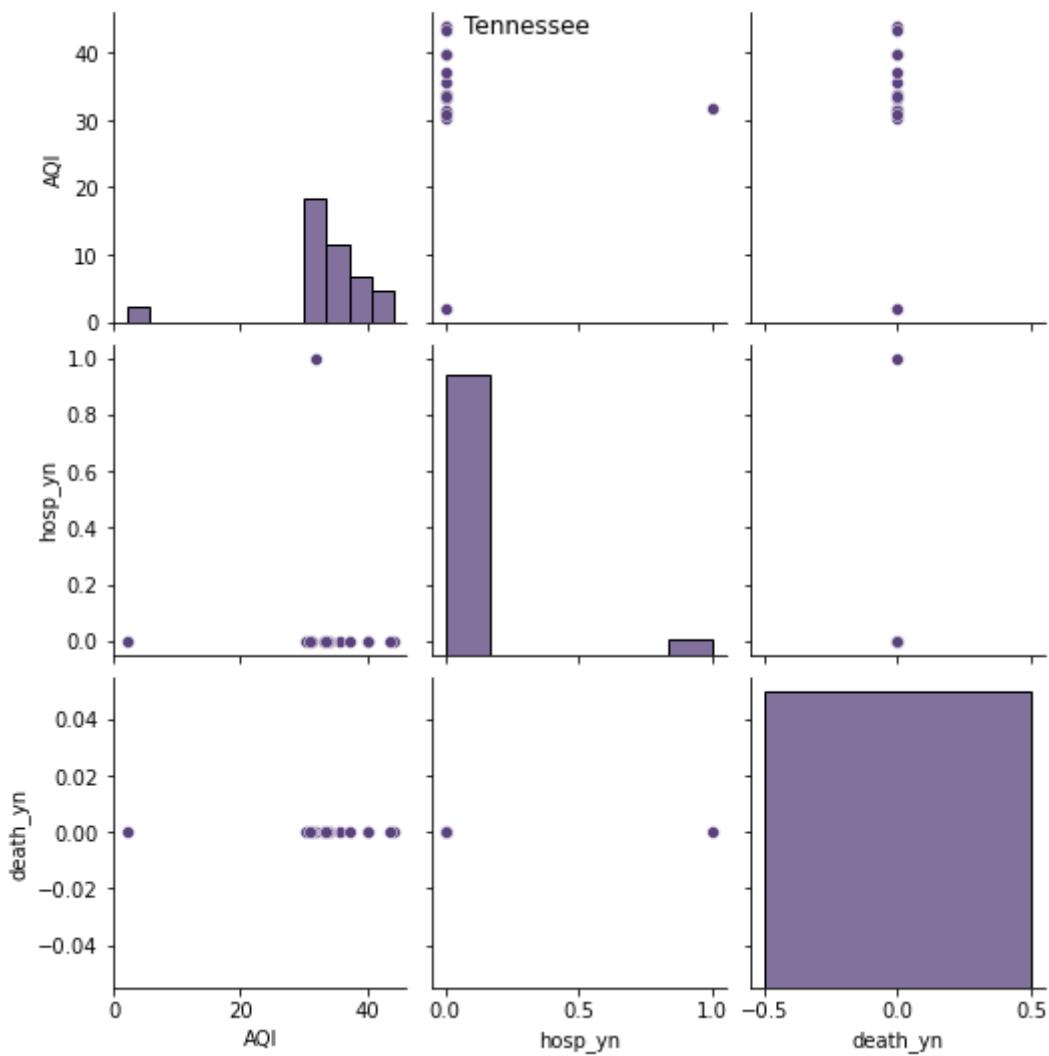
<Figure size 1080x504 with 0 Axes>



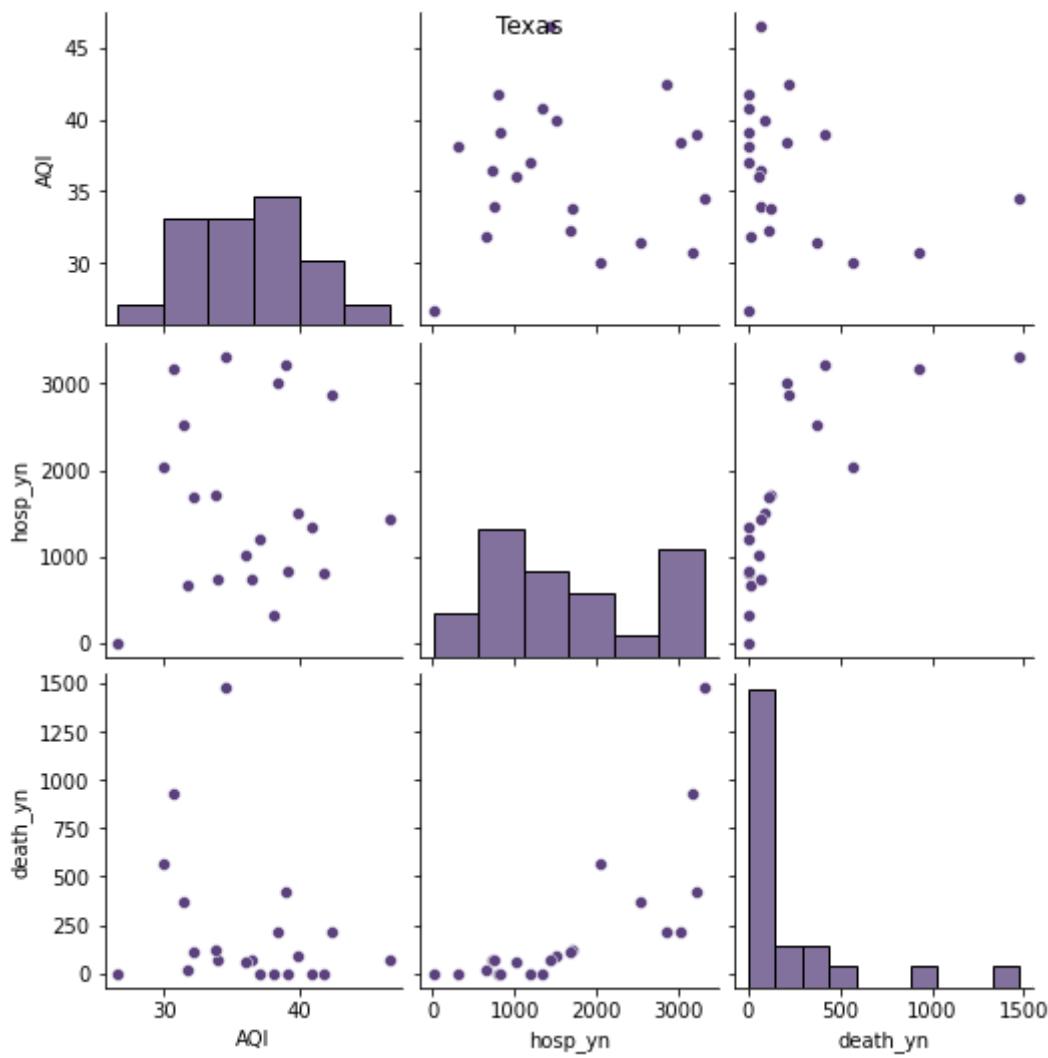
<Figure size 1080x504 with 0 Axes>



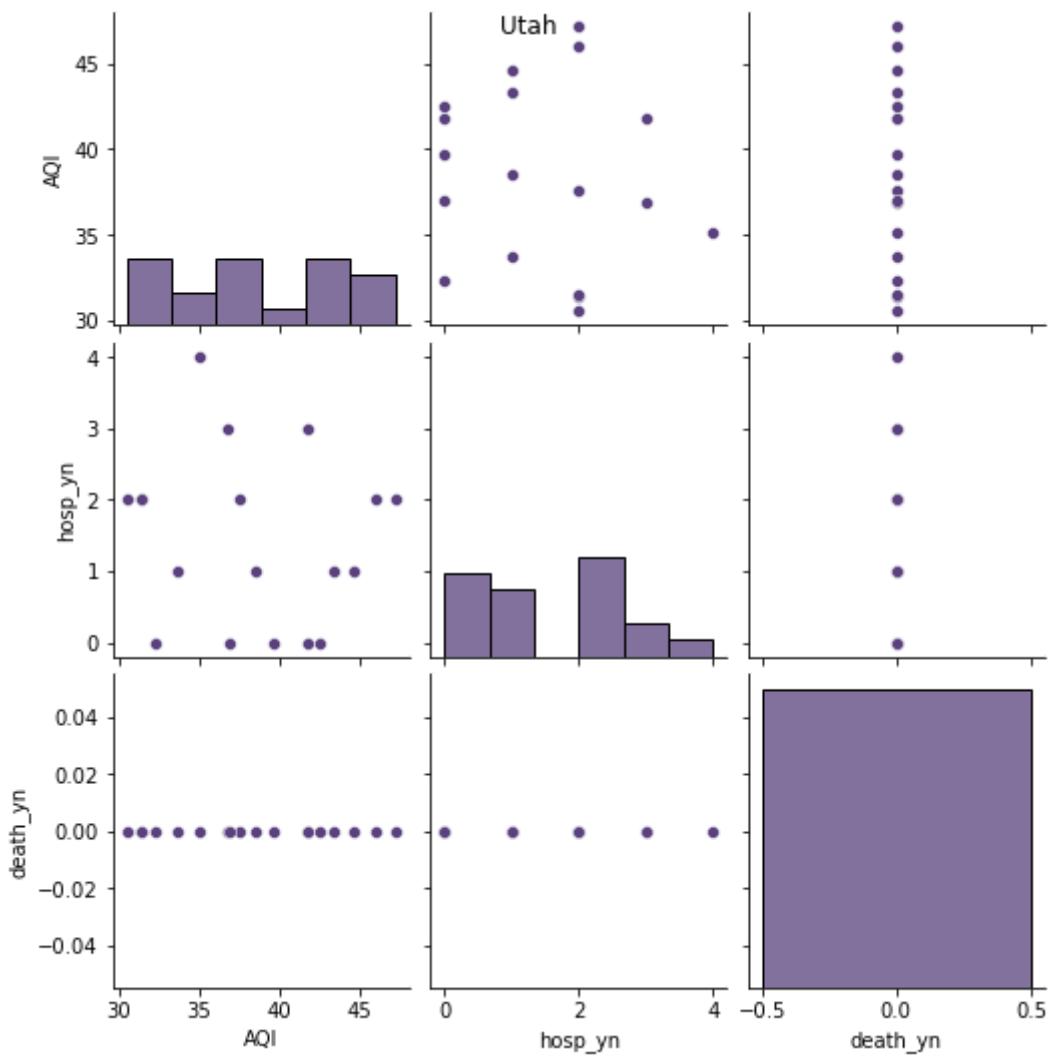
&lt;Figure size 1080x504 with 0 Axes&gt;



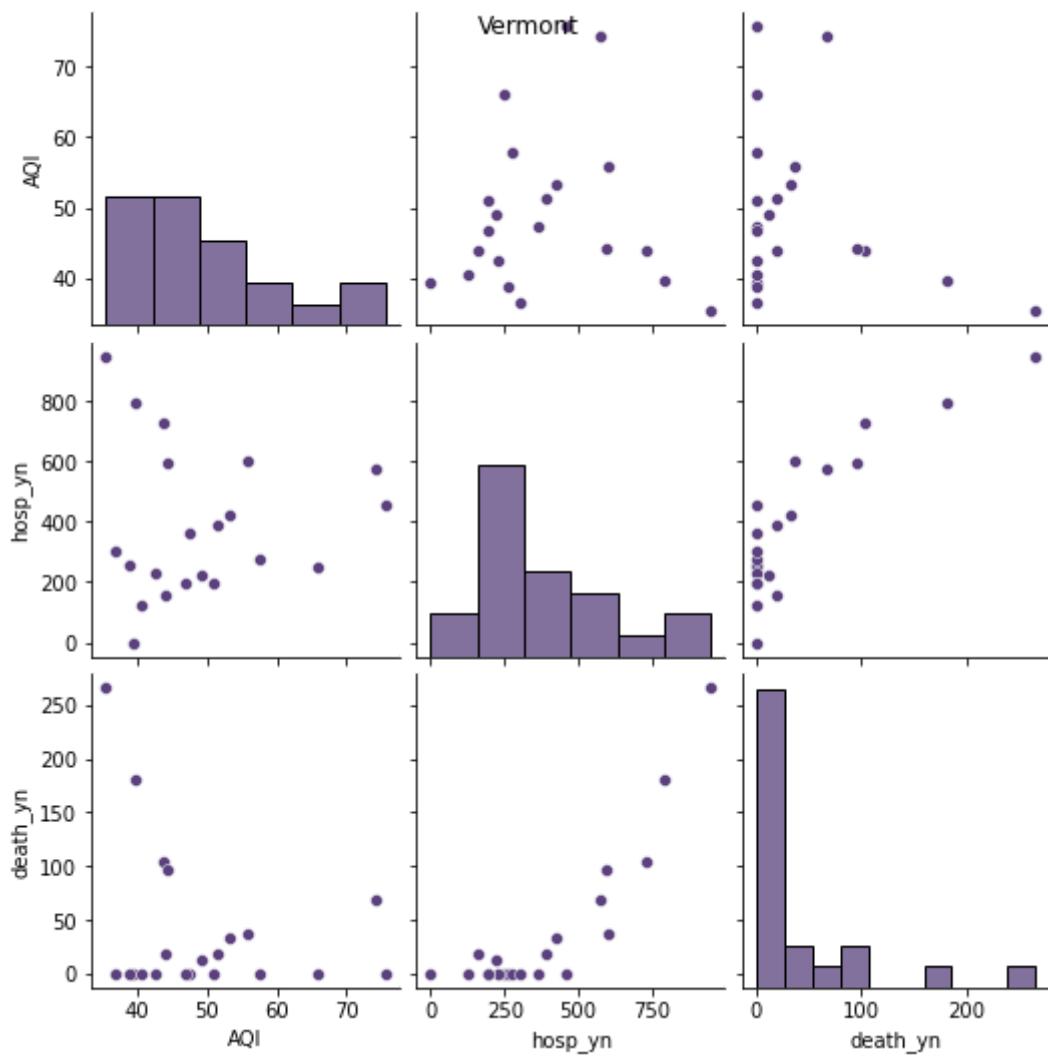
<Figure size 1080x504 with 0 Axes>



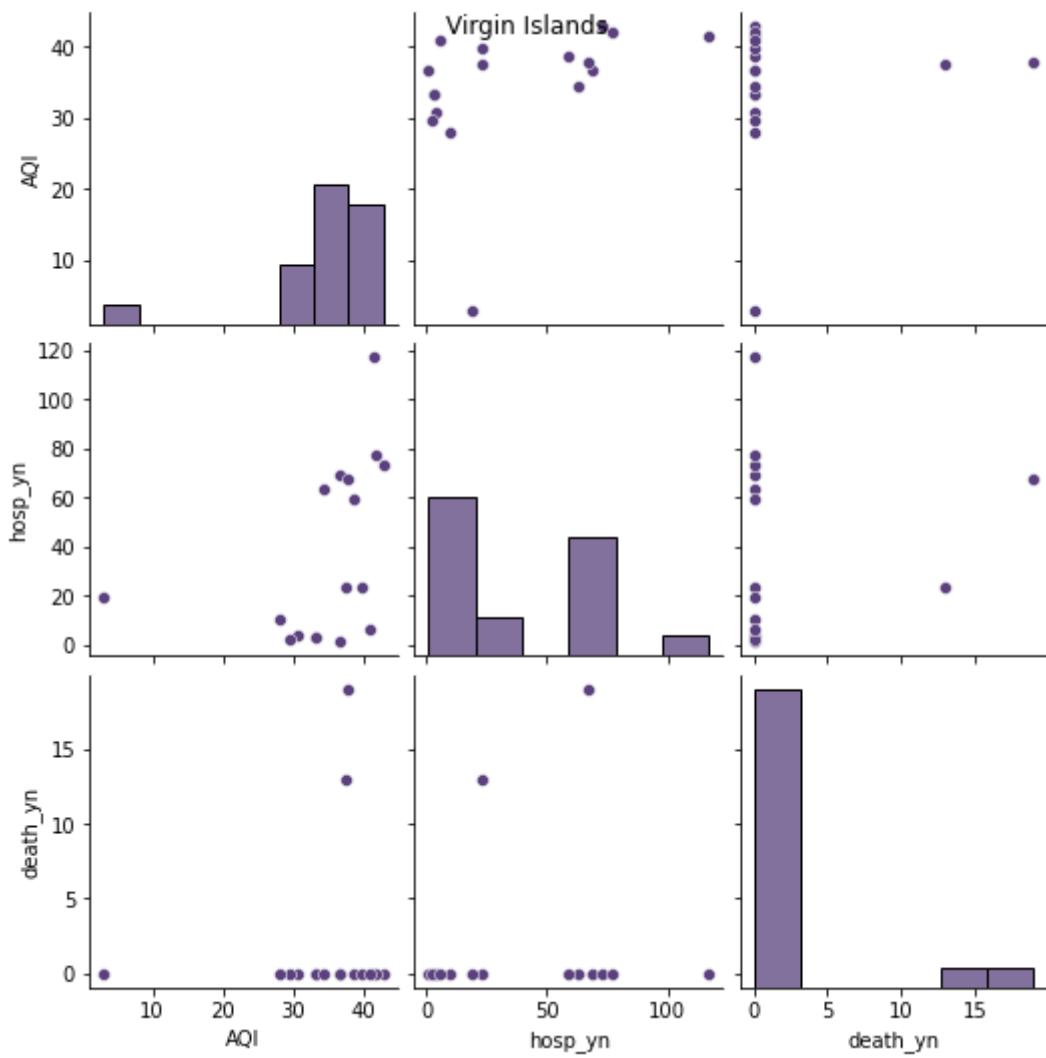
<Figure size 1080x504 with 0 Axes>



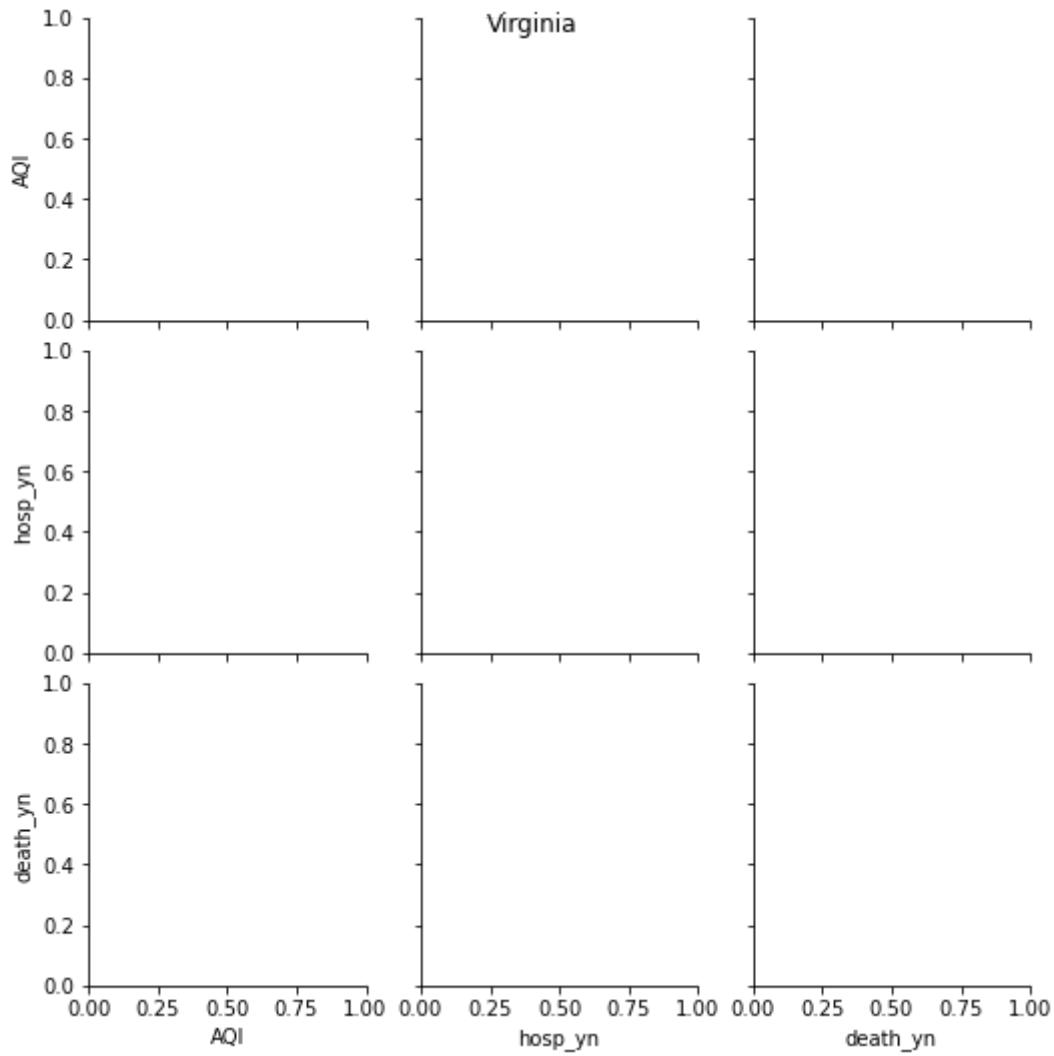
<Figure size 1080x504 with 0 Axes>



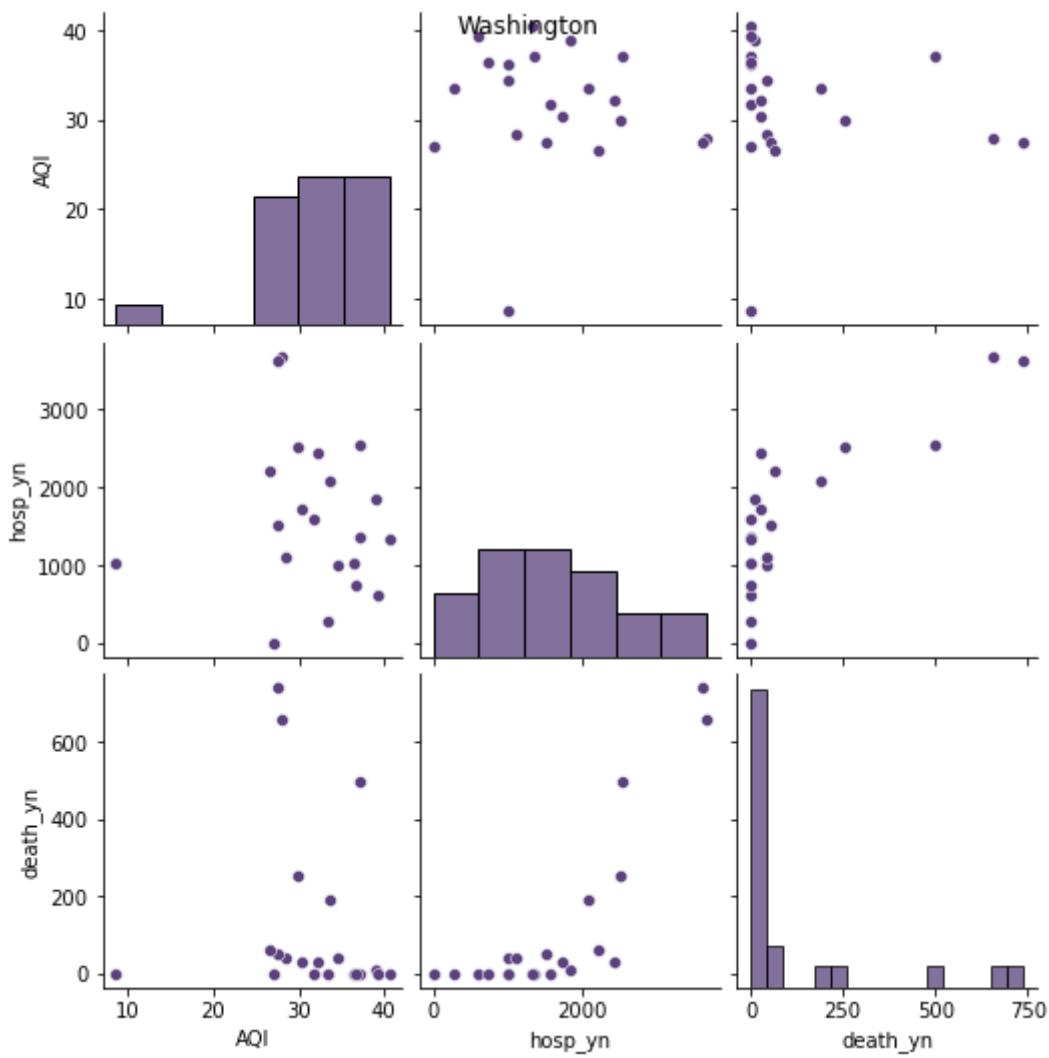
<Figure size 1080x504 with 0 Axes>



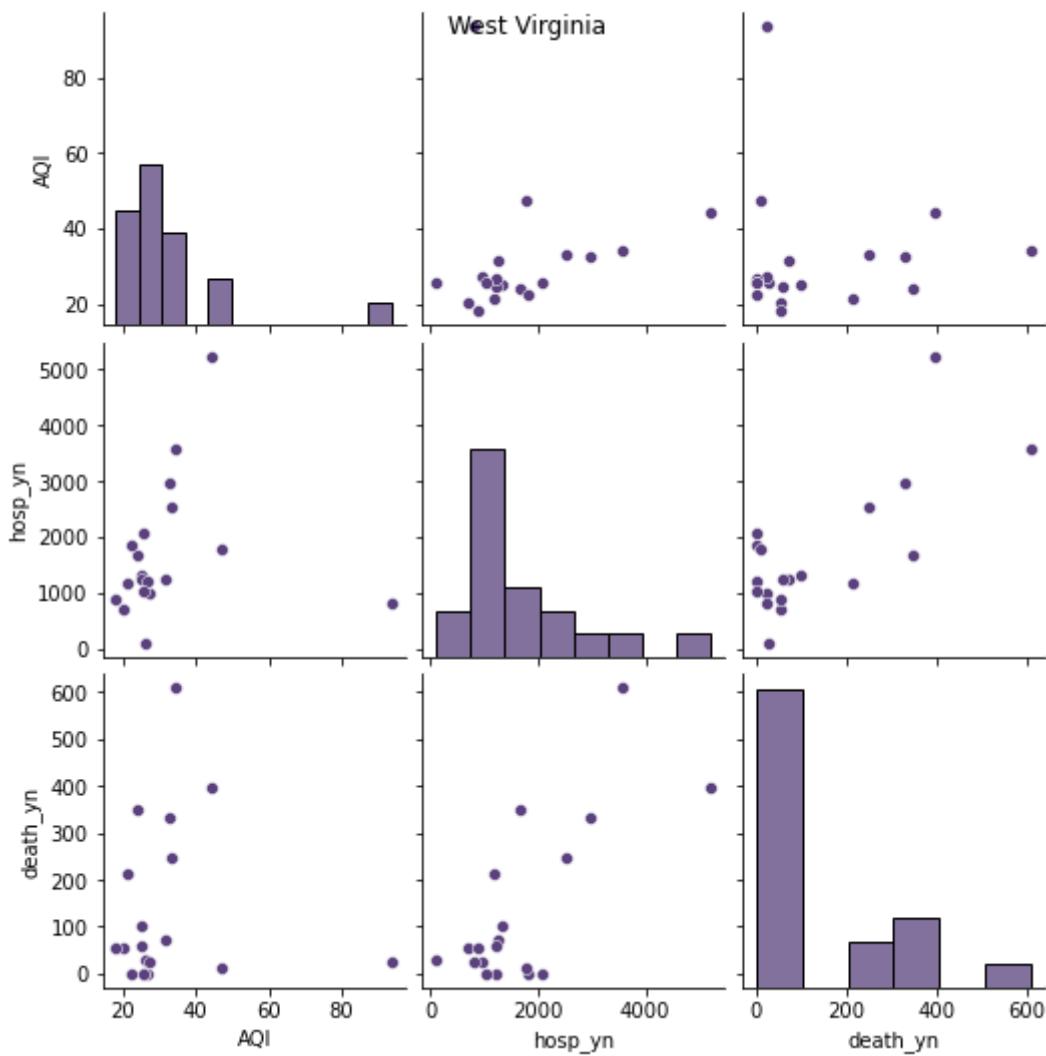
&lt;Figure size 1080x504 with 0 Axes&gt;



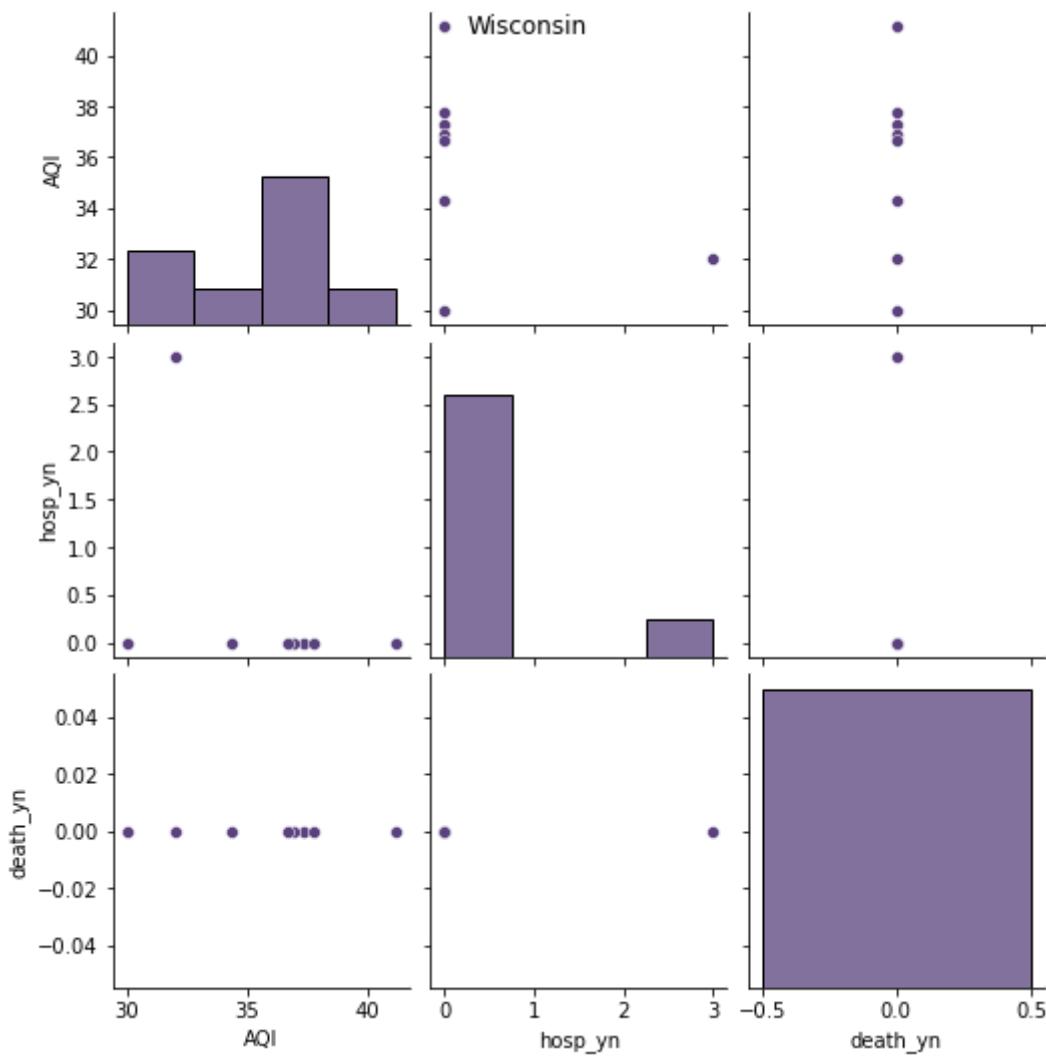
<Figure size 1080x504 with 0 Axes>



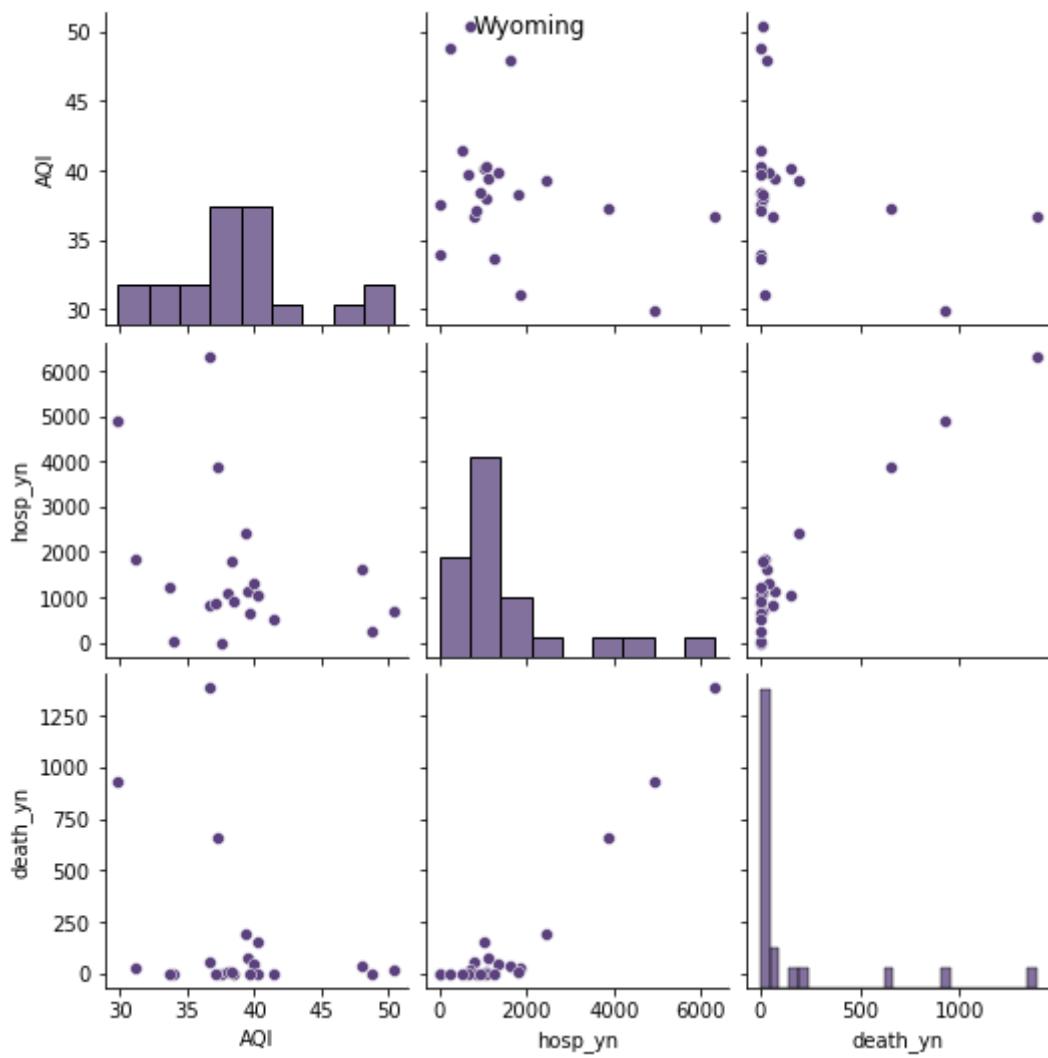
&lt;Figure size 1080x504 with 0 Axes&gt;



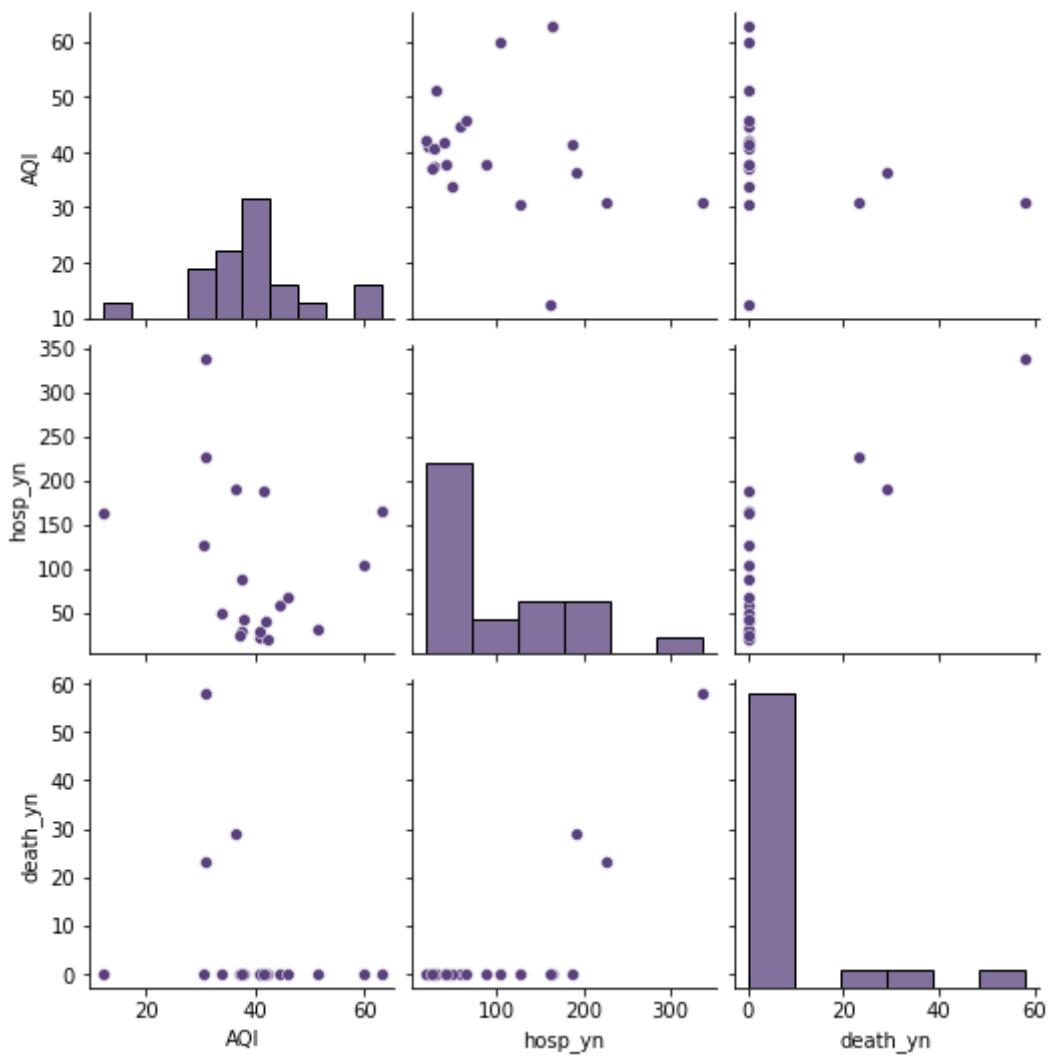
&lt;Figure size 1080x504 with 0 Axes&gt;



&lt;Figure size 1080x504 with 0 Axes&gt;



<Figure size 1080x504 with 0 Axes>



## Final Summary

Confounding factors are variables that are associated with both the exposure and the outcome, and can therefore distort the relationship between the exposure and outcome. In this case according to the heat graph of Death and underlying conditions, they are not directly associated with AQI: Hence, they are not considered confounding factors in the relationship between AQI and death or underlying conditions. Confounding factors should be controlled for in order to accurately assess the relationship between the exposure and outcome.