DAT490 Capstone Project: Asthma and Pollution

Ainsley Atherton, Nikki La, Jordan Ledbetter, Palaniappan Vijay Sithambaram

Libraries

In [ ]:
```
# Data manipulation
import pandas as pd
import numpy as np
import scipy

# Data visualization
import matplotlib.pyplot as plt
import seaborn as sns

# Geospatial data
import geopandas as gpd

# Stats
import scipy.spatial.distance as ssd
import scipy.stats as ss
from scipy import stats

# Machine learning
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.preprocessing import MinMaxScaler
from sklearn.cluster import KMeans
from sklearn.linear_model import LinearRegression
import statsmodels.api as sm

# Ignoring warnings
import warnings
warnings.filterwarnings('ignore')
```

About the Datasets

- The CDC National Environmental Public Health Tracking Network is a collection of data comprising of census data, the Behavioral Risk Factor Surveillance System (BRFSS), and other epidemiological surveys. Querying asthma data by county and year for emergency room visits, hospitalizations, and prevalence.
  - Emergency room data includes: StateFIPS, State, CountyFIPS, County, Year, Value
  - Hospitalizations data includes: StateFIPS, State, CountyFIPS, County, Year, Value
  - Prevalence data includes: StateFIPS, State, CountyFIPS, County, Year, Value, Data Comment, 95% Confidence Interval, Confidence Interval Low, Confidence Interval High

- Environmental Protection Agency (EPA) data is utilized for this project, specifically, annual Air Quality Index (AQI) data by county spanning the years 2000 to 2021.
  - The CSV file encompasses the following variables: State, County, Year, Days with AQI, Good Days, Moderate Days, Unhealthy for Sensitive Groups Days, Unhealthy Days, Very Unhealthy Days, Hazardous Days, Max AQI, 90th Percentile AQI, Median AQI, Days CO, Days NO2, Days Ozone, Days PM2.5, Days PM10.

- Note: It's important to acknowledge that not all counties may report data; however, the analysis captures trends across urban and highly populated areas.

In [ ]:
```
#Read in the data
#Asthma
folder_path = 'CDC_Asthma_Data'
asthma_prev = pd.read_csv(folder_path + '/Adult_Asthma_prev2018-2021/data_1807
asthma_hosp = pd.read_csv(folder_path + '/Asthma_Hospitalizations2000-2021/dat
asthma_er = pd.read_csv(folder_path + '/ER_visits_Asthma2000-2021/data_181211.
```

In [ ]:
```
#EPA Data 2000 - 2021
AQI_folder_path = 'CDC_Asthma_Data/AQI_data'
AQI2000 = pd.read_csv(AQI_folder_path + '/annual_aqi_by_county_2000.csv')
AQI2001 = pd.read_csv(AQI_folder_path + '/annual_aqi_by_county_2001.csv')
AQI2002 = pd.read_csv(AQI_folder_path + '/annual_aqi_by_county_2002.csv')
AQI2003 = pd.read_csv(AQI_folder_path + '/annual_aqi_by_county_2003.csv')
AQI2004 = pd.read_csv(AQI_folder_path + '/annual_aqi_by_county_2004.csv')
AQI2005 = pd.read_csv(AQI_folder_path + '/annual_aqi_by_county_2005.csv')
AQI2006 = pd.read_csv(AQI_folder_path + '/annual_aqi_by_county_2006.csv')
AQI2007 = pd.read_csv(AQI_folder_path + '/annual_aqi_by_county_2007.csv')
AQI2008 = pd.read_csv(AQI_folder_path + '/annual_aqi_by_county_2008.csv')
AQI2009 = pd.read_csv(AQI_folder_path + '/annual_aqi_by_county_2009.csv')
AQI2010 = pd.read_csv(AQI_folder_path + '/annual_aqi_by_county_2010.csv')
AQI2011 = pd.read_csv(AQI_folder_path + '/annual_aqi_by_county_2011.csv')
AQI2012 = pd.read_csv(AQI_folder_path + '/annual_aqi_by_county_2012.csv')
AQI2013 = pd.read_csv(AQI_folder_path + '/annual_aqi_by_county_2013.csv')
AQI2014 = pd.read_csv(AQI_folder_path + '/annual_aqi_by_county_2014.csv')
AQI2015 = pd.read_csv(AQI_folder_path + '/annual_aqi_by_county_2015.csv')
AQI2016 = pd.read_csv(AQI_folder_path + '/annual_aqi_by_county_2016.csv')
AQI2017 = pd.read_csv(AQI_folder_path + '/annual_aqi_by_county_2017.csv')
AQI2018 = pd.read_csv(AQI_folder_path + '/annual_aqi_by_county_2018.csv')
AQI2019 = pd.read_csv(AQI_folder_path + '/annual_aqi_by_county_2019.csv')
AQI2020 = pd.read_csv(AQI_folder_path + '/annual_aqi_by_county_2020.csv')
AQI2021 = pd.read_csv(AQI_folder_path + '/annual_aqi_by_county_2021.csv')
```

In [ ]:
```
### Load shapefiles
# Counties
url_county = 'https://www2.census.gov/geo/tiger/GENZ2021/shp/cb_2021_us_county_
county_gdf = gpd.read_file(url_county)

# States
url_county = 'https://www2.census.gov/geo/tiger/GENZ2021/shp/cb_2021_us_state_5
state_gdf = gpd.read_file(url_county)

# Exclude everthing outside the contiguous US
exclude_list = [15, 72, 2, 60, 66, 69, 78]
county_gdf = county_gdf.loc[~county_gdf['STATEFP'].astype(int).isin(exclude_lis
state_gdf = state_gdf.loc[~state_gdf['STATEFP'].astype(int).isin(exclude_list)]
```

```
# Convert to 5070
county_gdf = county_gdf.to_crs(5070)
state_gdf = state_gdf.to_crs(5070)

# Create a FIPS code
county_gdf['FIPS'] = county_gdf.STATEFP + county_gdf.COUNTYFP
```

Data Pre-Processing

Dataset cleaning and merging with air quality data and shapefiles. Grouping the data into like groups of A, B, and C for increasing, decreasing, and neutral trends respectively for each asthma dataset.

The years 2005-2019 were used based on the completeness of the data for these years. Asthma prevalence was dropped due to insufficient data.

Models will be constructed on groups A, B, and C of ER visits and group B for hospitalizations due to an insufficient number of counties for the other two hospitalization groups.

Asthma Prevalence

In [ ]:
```
asthma_prev.rename(columns = {'Value':'Prevalence %'}, inplace = True)

#Remove null columns and unneded columns
asthma_prev = asthma_prev.drop(['Data Comment', 'Unnamed: 10', '95% Confidence

## Merge asthma prevalence data with geospatial data
# Convert county FIPS code columns to strings
asthma_prev['CountyFIPS'] = asthma_prev['CountyFIPS'].astype(str)

# Add leading zeros to FIPS code where needed
asthma_prev['CountyFIPS'] = asthma_prev['CountyFIPS'].apply(lambda x: x.zfill(5

# Groupy values by FIPS
#grouped_hosp = asthma_hosp.groupby('CountyFIPS')['Hospitalizations (rate per 1

prev18 = asthma_prev[asthma_prev['Year'] == 2018]
prev19 = asthma_prev[asthma_prev['Year'] == 2019]
prev20 = asthma_prev[asthma_prev['Year'] == 2020]
prev21 = asthma_prev[asthma_prev['Year'] == 2021]

# Merge the datasets on their FIPS code

merged_prev18 = county_gdf.merge(prev18, left_on='FIPS', right_on='CountyFIPS'
merged_prev19 = county_gdf.merge(prev19, left_on='FIPS', right_on='CountyFIPS'
merged_prev20 = county_gdf.merge(prev20, left_on='FIPS', right_on='CountyFIPS'
merged_prev21 = county_gdf.merge(prev21, left_on='FIPS', right_on='CountyFIPS'
```

Hospitalizations due to Asthma

- Age adjusted rate of hospitalization for asthma per 10,000

In [ ]:
```
# Rename Value column
asthma_hosp.rename(columns = {'Value':'Hospitalizations (rate per 10,000)'}, i

# Remove all-null columns
asthma_hosp = asthma_hosp.drop(columns = 'Unnamed: 7')

# Calculate percentage of suppressed data
is_suppressed = asthma_hosp[asthma_hosp['Hospitalizations (rate per 10,000)'] =
#print('Percentage of Suppressed Data:', (len(is_suppressed) / len(asthma_hosp)

# Calculate percentage of unstability
is_unstable = asthma_hosp[asthma_hosp['Data Comment'] == 'Unstable']
#print('Percentage of Unstability:', (len(is_unstable) / len(asthma_hosp)) * 1

# Asthma Hospitalizations: 2000-2021
asthma_hosp['Hospitalizations (rate per 10,000)'] = asthma_hosp['Hospitalizati
asthma_hosp['Hospitalizations (rate per 10,000)'] = pd.to_numeric(asthma_hosp[

# Fix columns for successful merge
asthma_hosp = asthma_hosp.drop(columns='Data Comment')
asthma_hosp['CountyFIPS'] = asthma_hosp['CountyFIPS'].astype(int)
```

In [ ]:
```
# BEGIN: group asthma_er dataset by year and sum the last column in new datafra
asthma_hosp_grouped = asthma_er.groupby('Year').sum()
asthma_hosp_grouped = asthma_hosp_grouped.drop(columns = {'CountyFIPS', 'State
#asthma_hosp_grouped.plot(kind='bar', figsize=(10, 6), title='Total Hospitaliza
```

Hospitalization Grouping

In [ ]:
```
ahosp = asthma_hosp[asthma_hosp['Year'] == 2005]
#print(aer.shape)
asthma_hosp19 = asthma_hosp[asthma_hosp['Year'] <= 2019]
asthma_hosp19 = asthma_hosp19[asthma_hosp19['Year'] >= 2005]
asthma_hosp19 = asthma_hosp19.dropna()
#print(asthma_er19.head())
asthma_hosp19.shape

unique_counties_df = asthma_hosp19.groupby(['CountyFIPS', 'State', 'County']).
print(f"before removal: {unique_counties_df.shape}")

unique_counties_df = unique_counties_df[unique_counties_df['Counts'] == 15]
print(f"after incomplete row removal: {unique_counties_df.shape}")
print(unique_counties_df.head(5))

states = unique_counties_df['State'].unique()
#states
```

```
before removal: (1752, 4)
after incomplete row removal: (689, 4)
   CountyFIPS    State    County  Counts
0        4001  Arizona    Apache      15
1        4003  Arizona   Cochise      15
2        4005  Arizona  Coconino      15
3        4007  Arizona      Gila      15
7        4013  Arizona  Maricopa      15
```

In [ ]:
```
#checking for null values
nan_values = asthma_hosp19.isna().sum()
```

```
print(nan_values)
```

```
StateFIPS                              0
State                                  0
CountyFIPS                             0
County                                 0
Year                                   0
Hospitalizations (rate per 10,000)     0
dtype: int64
```

Group sorting for Hospitalizations:

In [ ]:
```
plt.style.use("seaborn-whitegrid")
plt.rc(
    "figure",
    autolayout=True,
    figsize=(11, 4),
    titlesize=18,
    titleweight='bold',
)
plt.rc(
    "axes",
    labelweight="bold",
    labelsize="large",
    titleweight="bold",
    titlesize=16,
    titlepad=10,
)
%config InlineBackend.figure_format = 'retina'

hosp_IncA = []
hosp_DecB = []
hosp_neutralC = []

for county in unique_counties_df['CountyFIPS']:  #[:30]


    state = unique_counties_df[unique_counties_df['CountyFIPS'] == county]['Sta
    cname = unique_counties_df[unique_counties_df['CountyFIPS'] == county]['Cou
    x = asthma_hosp19[asthma_hosp19['CountyFIPS'] == county]['Year']
    y = asthma_hosp19[asthma_hosp19['CountyFIPS'] == county]['Hospitalizations

    slope, intercept, r, p, sterr = scipy.stats.linregress(x, y)


    #All plots
    """
    fig, ax = plt.subplots()
    ax.plot(x, y, label=county)
    p = sns.regplot(x='Year', y='Hospitalizations (rate per 10,000)', data=astl
    slope, intercept, r, p, sterr = scipy.stats.linregress(x=p.get_lines()[0].g
                                                           y=p.get_lines()[0].get_y
    ax.set_title(f'Hospitalizations for Asthma {state}, {cname} ({county})    S
    plt.show()
    print(f"FIPS: {county}, County: {cname}, Slope: {slope}, Intercept: {inter
    #df.append(x)
    #df.append(y)

    #Just stats
    #p = sns.regplot(x='Year', y='ER visits for Asthma (rate per 10,000)', data
```

```
    #slope, intercept, r, p, sterr = scipy.stats.linregress(x=p.get_lines()[0]
    #                                                       y=p.get_lines()[0].get_

    if slope > 0.1:
        hosp_IncA.append([county, slope])

    if slope < -0.1:
        hosp_DecB.append([county, slope])

    if slope < 0.1 and slope > -0.1:
        hosp_neutralC.append([county, slope])
```

Only 3 counties in group A and 15 in group B for hospitalizations. Dropping these groups as there is an insufficient amount of data to base a proper model on.

In [ ]:
```
A = pd.DataFrame(hosp_IncA, columns=['CountyFIPS', 'Slope'])
B = pd.DataFrame(hosp_DecB, columns=['CountyFIPS', 'Slope'])
C = pd.DataFrame(hosp_neutralC, columns=['CountyFIPS', 'Slope'])

print(A.shape)
print(B.shape)
print(C.shape)
```

```
(3, 2)
(671, 2)
(15, 2)
```

Merge groups with original dataset to get by year data

In [ ]:
```
hosp_IncA = pd.DataFrame(hosp_IncA)
hosp_IncA.shape
hosp_IncA_full = hosp_IncA.merge(asthma_hosp19, left_on=0, right_on='CountyFIPS
hosp_IncA_full = hosp_IncA_full.rename(columns={0: 'FIPS', 1: 'slope'})
#print(hosp_IncA_full.shape)
#print(hosp_IncA_full.head(10))

hosp_DecB = pd.DataFrame(hosp_DecB)
hosp_DecB.shape
hosp_DecB_full = hosp_DecB.merge(asthma_hosp19, left_on=0, right_on='CountyFIPS
hosp_DecB_full = hosp_DecB_full.rename(columns={0: 'FIPS', 1: 'slope'})
#print(hosp_DecB_full.shape)
#print(hosp_DecB_full.head(10))

hosp_neutralC = pd.DataFrame(hosp_neutralC)
hosp_neutralC.shape
hosp_neutralC_full = hosp_neutralC.merge(asthma_hosp19, left_on=0, right_on='C
hosp_neutralC_full = hosp_neutralC_full.rename(columns={0: 'FIPS', 1: 'slope'}
#print(hosp_neutralC_full.shape)
hosp_neutralC_full.head(10)
```

Out[ ]:

	FIPS	slope	StateFIPS	State	CountyFIPS	County	Year	Hospitalizations (rate per 10,000)
0	6045	-0.0825	6	California	6045	Mendocino	2005	6.7
1	6045	-0.0825	6	California	6045	Mendocino	2006	6.2
2	6045	-0.0825	6	California	6045	Mendocino	2007	3.2
3	6045	-0.0825	6	California	6045	Mendocino	2008	2.6
4	6045	-0.0825	6	California	6045	Mendocino	2009	4.9
5	6045	-0.0825	6	California	6045	Mendocino	2010	5.9
6	6045	-0.0825	6	California	6045	Mendocino	2011	6.5
7	6045	-0.0825	6	California	6045	Mendocino	2012	5.6
8	6045	-0.0825	6	California	6045	Mendocino	2013	3.7
9	6045	-0.0825	6	California	6045	Mendocino	2014	4.2

ER visits due to Asthma

In [ ]:
```
asthma_er.rename(columns = {'Value':'ER visits for Asthma (rate per 10,000)'},
#asthma_er.info()

asthma_er = asthma_er.drop(['Unnamed: 7', 'Data Comment'], axis=1)

# Calculate percentage of suppressed data
er_suppressed = asthma_er[asthma_er['ER visits for Asthma (rate per 10,000)'] =
#print('Percentage of Suppressed Data:', (len(er_suppressed) / len(asthma_er))

# Asthma ER visits: 2000-2021
asthma_er['ER visits for Asthma (rate per 10,000)'] = asthma_er['ER visits for
asthma_er['ER visits for Asthma (rate per 10,000)'] = pd.to_numeric(asthma_er[
#asthma_er['ER visits for Asthma (rate per 10,000)'].describe()

county_gdf.rename(columns = {'GEOID':'CountyFIPS'}, inplace = True)
```

In [ ]:
```
asthma_er = asthma_er.dropna()
asthma_er.isnull().sum()
```

Out[ ]:
```
StateFIPS                                  0
State                                      0
CountyFIPS                                 0
County                                     0
Year                                       0
ER visits for Asthma (rate per 10,000)     0
dtype: int64
```

In [ ]:
```
aer = asthma_er[asthma_er['Year'] == 2005]
#print(aer.shape)
asthma_er19 = asthma_er[asthma_er['Year'] <= 2019]
asthma_er19 = asthma_er19[asthma_er19['Year'] >= 2005]
#print(asthma_er19.head())
asthma_er19.shape

unique_counties_df = asthma_er19.groupby(['State', 'County']).size().reset_ind
print(f"before 14 ", {unique_counties_df.shape})
```

```
unique_counties_df = unique_counties_df[unique_counties_df['Counts'] == 15]
print(unique_counties_df.shape)
print(unique_counties_df.head(50))

states = unique_counties_df['State'].unique()
states
```

```
before 14  {(1459, 3)}
(725, 3)
```

	State	County	Counts
0	Arizona	Apache	15
1	Arizona	Cochise	15
2	Arizona	Coconino	15
3	Arizona	Gila	15
4	Arizona	Graham	15
5	Arizona	Greenlee	15
6	Arizona	La Paz	15
7	Arizona	Maricopa	15
8	Arizona	Mohave	15
9	Arizona	Navajo	15
10	Arizona	Pima	15
11	Arizona	Pinal	15
12	Arizona	Santa Cruz	15
13	Arizona	Yavapai	15
14	Arizona	Yuma	15
15	California	Alameda	15
17	California	Amador	15
18	California	Butte	15
19	California	Calaveras	15
20	California	Colusa	15
21	California	Contra Costa	15
22	California	Del Norte	15
23	California	El Dorado	15
24	California	Fresno	15
25	California	Glenn	15
26	California	Humboldt	15
27	California	Imperial	15
28	California	Inyo	15
29	California	Kern	15
30	California	Kings	15
31	California	Lake	15
32	California	Lassen	15
33	California	Los Angeles	15
34	California	Madera	15
35	California	Marin	15
36	California	Mariposa	15
37	California	Mendocino	15
38	California	Merced	15
39	California	Modoc	15
40	California	Mono	15
41	California	Monterey	15
42	California	Napa	15
43	California	Nevada	15
44	California	Orange	15
45	California	Placer	15
46	California	Plumas	15
47	California	Riverside	15
48	California	Sacramento	15
49	California	San Benito	15
50	California	San Bernardino	15

Out[ ]:
```
array(['Arizona', 'California', 'Connecticut', 'Florida', 'Iowa', 'Maine',
       'Massachusetts', 'Minnesota', 'Missouri', 'New Jersey', 'New York',
       'Rhode Island', 'Tennessee', 'Utah', 'Vermont', 'Wisconsin'],
      dtype=object)
```

In [ ]:
```
for county in unique_counties_df['County']:
    #state = asthma_er[asthma_er['County'] == county]['State'].values[0]
```

```
    state = unique_counties_df[unique_counties_df['County'] == county]['State']
    #state = unique_counties_df[unique_counties_df['County'] == county]['State
    #state = asthma_er[asthma_er['CountyFIPS'] == county]['State'].values[0]
    #print(f"County: {county}, State: {state}")

    #c1a = asthma_er[asthma_er['County'] == 'Maricopa']
    c1a = asthma_er19[asthma_er19["County"] == 'Maricopa']
    c1 = np.array(c1a['ER visits for Asthma (rate per 10,000)'])
    #print("This is c1:", c1)
    #c2a = asthma_er[asthma_er['County'] == county]
    #c2a = asthma_er[asthma_er['State'] == state]
    #c2a = asthma_er19[asthma_er19["County"] == county]
    c2a = asthma_er19[asthma_er19["County"] == county]
    c2a = asthma_er19[asthma_er19['State'] == state]
    c2a = c2a[c2a['County'] == county]
    c2 = np.array(c2a['ER visits for Asthma (rate per 10,000)'])
    #print("this is c2:", c2)
    #print(c2a)
    #print(c2a.shape)

    ssd.correlation(c1, c2) # Computes a distance measure based on correlation
    ccor = np.correlate(c1, c2, mode='valid') # Does a cross-correlation of sa
    mcor = np.corrcoef(c1, c2) # Gives back the correlation matrix for the two
    spear = ss.spearmanr(c1, c2) # Gives the spearman correlation for the two
    pearson = ss.pearsonr(c1, c2)
    #print(f"{county} Correlation: {ssd.correlation(c1, c2)}, spearman correla
    # Plot the data
    #df = [asthma_er[asthma_er['County'] == 'Maricopa'], asthma_er[asthma_er['
    #df = pd.concat(df)
    #df = c1a.append(c2a, ignore_index=True)
    #df.plot(x='Year', y='ER visits for Asthma (rate per 10,000)', marker='o')
    #plt.plot(c1a, c2a, marker='o')
    """ax = c1a.plot(x='Year', y='ER visits for Asthma (rate per 10,000)', mar
    c2a.plot(x='Year', y='ER visits for Asthma (rate per 10,000)', marker='o',

    """# Set the plot title and labels
    plt.title('ER Visits for Asthma ' + county)
    plt.xlabel('Year')
    plt.ylabel('ER visits (rate per 10,000)')

    # Show the plot
    plt.show()"""
    """
    if ssd.correlation(c1, c2) > 0.5:
        print(f"County: {county}, State: {state}, Correlation: {ssd.correlatio
```

In [ ]:

```
# Perform linear regression for each county
for county in unique_counties_df['County']:
    x = asthma_er19['Year']
    x = x.values.reshape(-1, 1)
    y = asthma_er19['ER visits for Asthma (rate per 10,000)']

    model = LinearRegression()
    model.fit(x, y)
    model = LinearRegression().fit(x, y)

    plt

    # Print the results
    r_sq = model.score(x, y)
```

```
    asthma_er19.append({'Slope': model.coef_, 'Intercept': model.intercept_, '
    #print(f"coefficient of determination: {r_sq}")

asthma_er19
```

Out[ ]:

	StateFIPS	State	CountyFIPS	County	Year	ER visits for Asthma (rate per 10,000)
0	4	Arizona	4001	Apache	2005	31.0
1	4	Arizona	4001	Apache	2006	21.1
2	4	Arizona	4001	Apache	2007	21.1
3	4	Arizona	4001	Apache	2008	20.2
4	4	Arizona	4001	Apache	2009	44.0
...	...	...	...	...	...	...
22052	55	Wisconsin	55141	Wood	2015	37.5
22053	55	Wisconsin	55141	Wood	2016	38.3
22054	55	Wisconsin	55141	Wood	2017	35.4
22055	55	Wisconsin	55141	Wood	2018	28.8
22056	55	Wisconsin	55141	Wood	2019	28.0

17488 rows × 6 columns

ER Grouping

In [ ]:

```
plt.style.use("seaborn-whitegrid")
plt.rc(
    "figure",
    autolayout=True,
    figsize=(11, 4),
    titlesize=18,
    titleweight='bold',
)
plt.rc(
    "axes",
    labelweight="bold",
    labelsize="large",
    titleweight="bold",
    titlesize=16,
    titlepad=10,
)
%config InlineBackend.figure_format = 'retina'

ER_IncA = []
ER_DecB = []
ER_neutralC = []

for county in unique_counties_df['County']:  #[:30]


    state = unique_counties_df[unique_counties_df['County'] == county]['State']
    x = asthma_er19[asthma_er19['County'] == county]['Year']
    y = asthma_er19[asthma_er19['County'] == county]['ER visits for Asthma (ra
```

```
    slope, intercept, r, p, sterr = scipy.stats.linregress(x, y)

    """
    #All plots
    fig, ax = plt.subplots()
    ax.plot(x, y, label=county)
    p = sns.regplot(x='Year', y='ER visits for Asthma (rate per 10,000)', data=
    slope, intercept, r, p, sterr = scipy.stats.linregress(x=p.get_lines()[0].g
                                                           y=p.get_lines()[0].get_y
    ax.set_title(f'ER Visits for Asthma {state}, {county}    Slope: {slope:.2f}
    plt.show()
    print(f"County: {county}, Slope: {slope}, Intercept: {intercept}, R: {r}, P
    #df.append(x)
    #df.append(y)

    #Just stats
    #p = sns.regplot(x='Year', y='ER visits for Asthma (rate per 10,000)', data
    #slope, intercept, r, p, sterr = scipy.stats.linregress(x=p.get_lines()[0]
    #                                                       y=p.get_lines()[0].get_

    if slope > 0.1:
        ER_IncA.append([county, slope])

    if slope < -0.1:
        ER_DecB.append([county, slope])

    if slope < 0.1 and slope > -0.1:
        ER_neutralC.append([county, slope])
```

In [ ]:
```
A = pd.DataFrame(ER_IncA, columns=['CountyFIPS', 'Slope'])
B = pd.DataFrame(ER_DecB, columns=['CountyFIPS', 'Slope'])
C = pd.DataFrame(ER_neutralC, columns=['CountyFIPS', 'Slope'])

print(A.shape)
print(B.shape)
print(C.shape)
```

```
(113, 2)
(582, 2)
(30, 2)
```

In [ ]:
```
ER_IncA = pd.DataFrame(ER_IncA)
ER_IncA_full = ER_IncA.merge(asthma_er19, left_on=0, right_on='County')
ER_IncA_full = ER_IncA_full.rename(columns={0: 'cname', 1: 'slope'})
print(ER_IncA_full.shape)
print(ER_IncA_full.head(10))

ER_DecB = pd.DataFrame(ER_DecB)
ER_DecB_full = ER_DecB.merge(asthma_er19, left_on=0, right_on='County')
ER_DecB_full = ER_DecB_full.rename(columns={0: 'cname', 1: 'slope'})
print(ER_DecB_full.shape)
print(ER_DecB_full.head(10))

ER_neutralC = pd.DataFrame(ER_neutralC)
ER_neutralC_full = ER_neutralC.merge(asthma_er19, left_on=0, right_on='County'
ER_neutralC_full = ER_neutralC_full.rename(columns={0: 'cname', 1: 'slope'})
print(ER_neutralC_full.shape)
ER_neutralC_full.head(10)
```

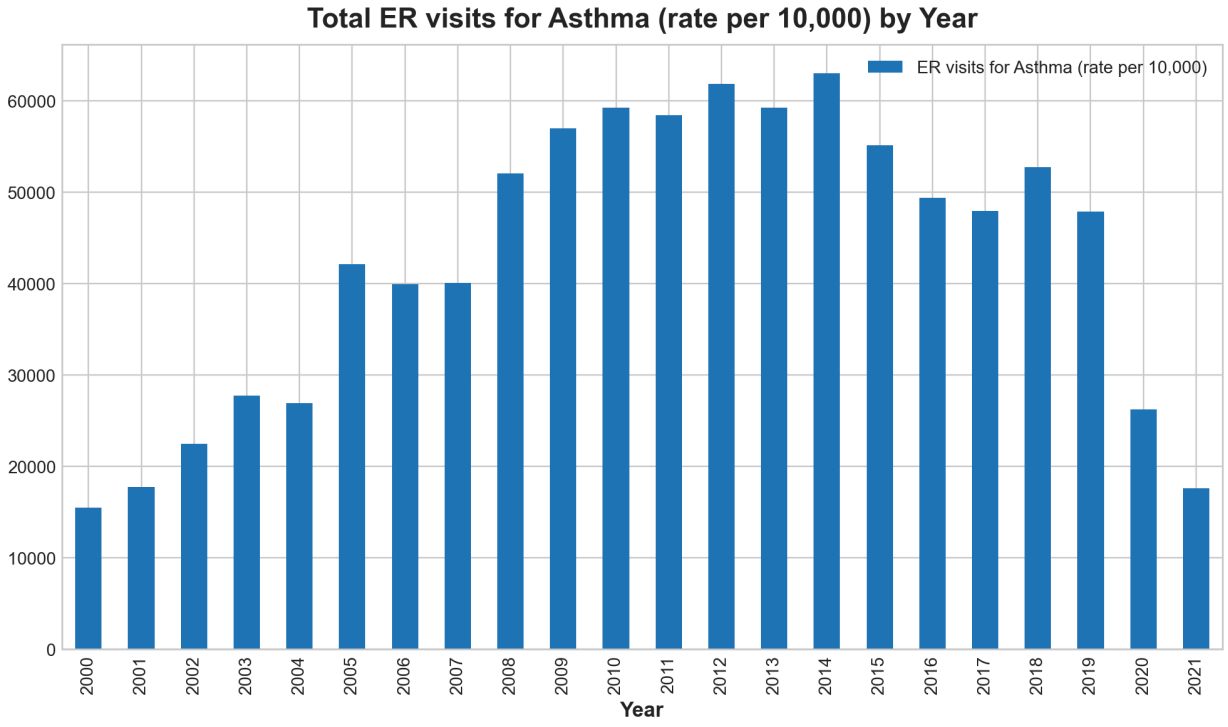
```
(3142, 8)
    cname     slope  StateFIPS    State  CountyFIPS  County  Year  \
0  Apache  0.130357          4  Arizona        4001  Apache  2005
1  Apache  0.130357          4  Arizona        4001  Apache  2006
2  Apache  0.130357          4  Arizona        4001  Apache  2007
3  Apache  0.130357          4  Arizona        4001  Apache  2008
4  Apache  0.130357          4  Arizona        4001  Apache  2009
5  Apache  0.130357          4  Arizona        4001  Apache  2010
6  Apache  0.130357          4  Arizona        4001  Apache  2011
7  Apache  0.130357          4  Arizona        4001  Apache  2012
8  Apache  0.130357          4  Arizona        4001  Apache  2013
9  Apache  0.130357          4  Arizona        4001  Apache  2014

   ER visits for Asthma (rate per 10,000)
0                                    31.0
1                                    21.1
2                                    21.1
3                                    20.2
4                                    44.0
5                                    45.0
6                                    57.8
7                                    37.1
8                                    41.0
9                                    37.7
(23343, 8)
     cname     slope  StateFIPS    State  CountyFIPS   County  Year  \
0  Cochise -1.376786          4  Arizona        4003  Cochise  2005
1  Cochise -1.376786          4  Arizona        4003  Cochise  2006
2  Cochise -1.376786          4  Arizona        4003  Cochise  2007
3  Cochise -1.376786          4  Arizona        4003  Cochise  2008
4  Cochise -1.376786          4  Arizona        4003  Cochise  2009
5  Cochise -1.376786          4  Arizona        4003  Cochise  2010
6  Cochise -1.376786          4  Arizona        4003  Cochise  2011
7  Cochise -1.376786          4  Arizona        4003  Cochise  2012
8  Cochise -1.376786          4  Arizona        4003  Cochise  2013
9  Cochise -1.376786          4  Arizona        4003  Cochise  2014

   ER visits for Asthma (rate per 10,000)
0                                    58.4
1                                    59.7
2                                    51.3
3                                    53.5
4                                    58.7
5                                    51.7
6                                    42.3
7                                    45.2
8                                    45.8
9                                    44.3
(665, 8)
```

Out[ ]:

	cname	slope	StateFIPS	State	CountyFIPS	County	Year	ER visits for Asthma (rate per 10,000)
0	Gila	-0.079286	4	Arizona	4007	Gila	2005	58.1
1	Gila	-0.079286	4	Arizona	4007	Gila	2006	53.1
2	Gila	-0.079286	4	Arizona	4007	Gila	2007	60.0
3	Gila	-0.079286	4	Arizona	4007	Gila	2008	44.8
4	Gila	-0.079286	4	Arizona	4007	Gila	2009	50.9
5	Gila	-0.079286	4	Arizona	4007	Gila	2010	60.9
6	Gila	-0.079286	4	Arizona	4007	Gila	2011	70.2
7	Gila	-0.079286	4	Arizona	4007	Gila	2012	59.6
8	Gila	-0.079286	4	Arizona	4007	Gila	2013	53.4
9	Gila	-0.079286	4	Arizona	4007	Gila	2014	53.0

In [ ]:
```
## Editing years
asthma_er

# BEGIN: group asthma_er dataset by year and sum the last column in new datafra
asthma_er_grouped = asthma_er.groupby('Year').sum()
asthma_er_grouped = asthma_er_grouped.drop(columns = {'CountyFIPS', 'StateFIPS
asthma_er_grouped.plot(kind='bar', figsize=(10, 6), title='Total ER visits for
```

Out[ ]:
```
<Axes: title={'center': 'Total ER visits for Asthma (rate per 10,000) by Yea
r'}, xlabel='Year'>
```



In [ ]:
```
asthma_er_xy = asthma_er[["Year", "ER visits for Asthma (rate per 10,000)"]]
asthma_er_xy
```

Out[ ]:

	Year	ER visits for Asthma (rate per 10,000)
0	2005	31.0
1	2006	21.1
2	2007	21.1
3	2008	20.2
4	2009	44.0
...	...	...
22054	2017	35.4
22055	2018	28.8
22056	2019	28.0
22057	2020	16.3
22058	2021	16.9

21358 rows × 2 columns

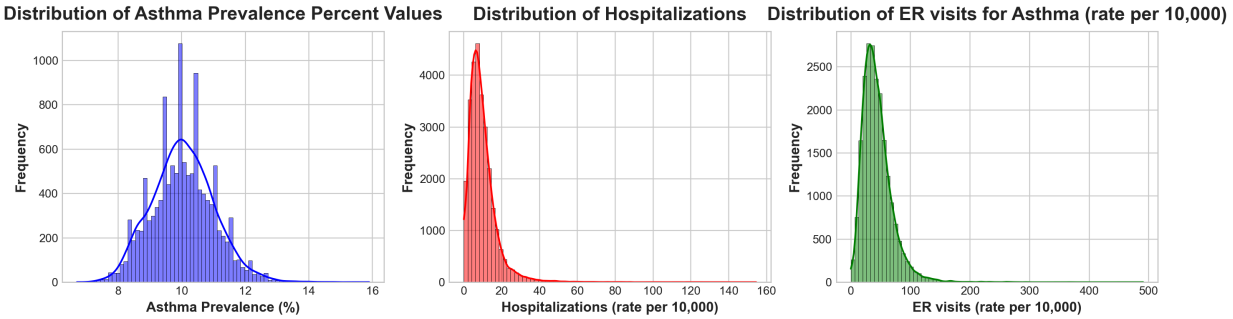
In [ ]:
```
# Create a MinMaxScaler object
scaler = MinMaxScaler()

for county in asthma_er['County'].unique():
    county_data = asthma_er[asthma_er['County'] == county]
    #print(county_data)
    normalized_data = scaler.fit_transform(county_data[['ER visits for Asthma
    normalized_data = pd.DataFrame(normalized_data, columns=['ER visits for As
    normalized_data[['County', 'Year']] = county_data[['County', 'Year']]
    #normalized_data.plot(x='Year', y='ER visits for Asthma (rate per 10,000)'
```

EDA Highlights

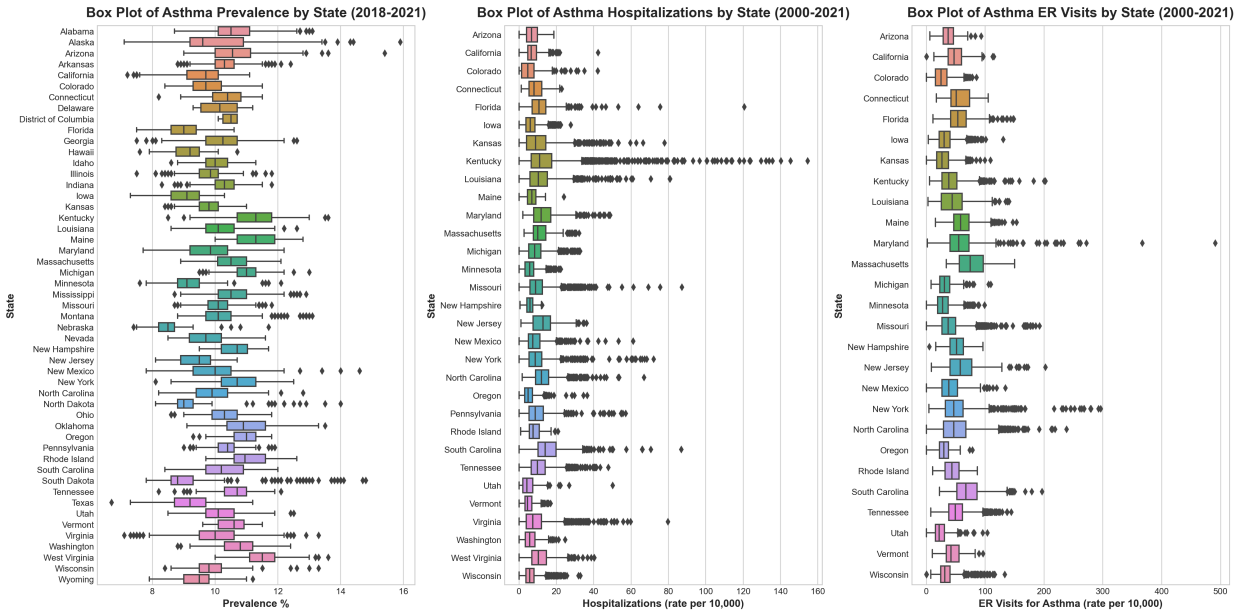
Asthma Data

In [ ]:
```
# Create a figure and three subplots
fig, axs = plt.subplots(1, 3, figsize=(14, 4))

# Plot the histograms
sns.histplot(ax=axs[0], data=asthma_prev, x='Prevalence %', bins=75, kde=True,
axs[0].set_title('Distribution of Asthma Prevalence Percent Values')
axs[0].set_xlabel('Asthma Prevalence (%)')
axs[0].set_ylabel('Frequency')

sns.histplot(ax=axs[1], data=asthma_hosp, x='Hospitalizations (rate per 10,000
axs[1].set_title('Distribution of Hospitalizations')
axs[1].set_xlabel('Hospitalizations (rate per 10,000)')
axs[1].set_ylabel('Frequency')

sns.histplot(ax=axs[2], data=asthma_er, x='ER visits for Asthma (rate per 10,0
axs[2].set_title('Distribution of ER visits for Asthma (rate per 10,000)')
axs[2].set_xlabel('ER visits (rate per 10,000)')
axs[2].set_ylabel('Frequency')
```

```
# Adjust the spacing between subplots
plt.tight_layout()

# Show the figure
plt.show()
```



In [ ]:
```
# Box Plot of Asthma ER Visits by State
fig, axs = plt.subplots(1, 3, figsize=(20, 10))

sns.boxplot(ax=axs[0], data=asthma_prev[(asthma_prev['Year'] >= 2018) & (asthma
            x='Prevalence %', y='State', orient='h')
axs[0].set_title('Box Plot of Asthma Prevalence by State (2018-2021)')
axs[0].set_xlabel('Prevalence %')
axs[0].set_ylabel('State')

sns.boxplot(ax=axs[1], data=asthma_hosp[(asthma_hosp['Year'] >= 2000) & (asthma
            x='Hospitalizations (rate per 10,000)', y='State', orient='h')
axs[1].set_title('Box Plot of Asthma Hospitalizations by State (2000-2021)')
axs[1].set_xlabel('Hospitalizations (rate per 10,000)')
axs[1].set_ylabel('State')

sns.boxplot(ax=axs[2], data=asthma_er[(asthma_er['Year'] >= 2000) & (asthma_er
            x='ER visits for Asthma (rate per 10,000)', y='State', orient='h')
axs[2].set_title('Box Plot of Asthma ER Visits by State (2000-2021)')
axs[2].set_xlabel('ER Visits for Asthma (rate per 10,000)')
axs[2].set_ylabel('State')

plt.tight_layout()
plt.show()
#axs[0].set_xticklabels(axs[0].get_xticklabels(), rotation=45)
```



Air Quality

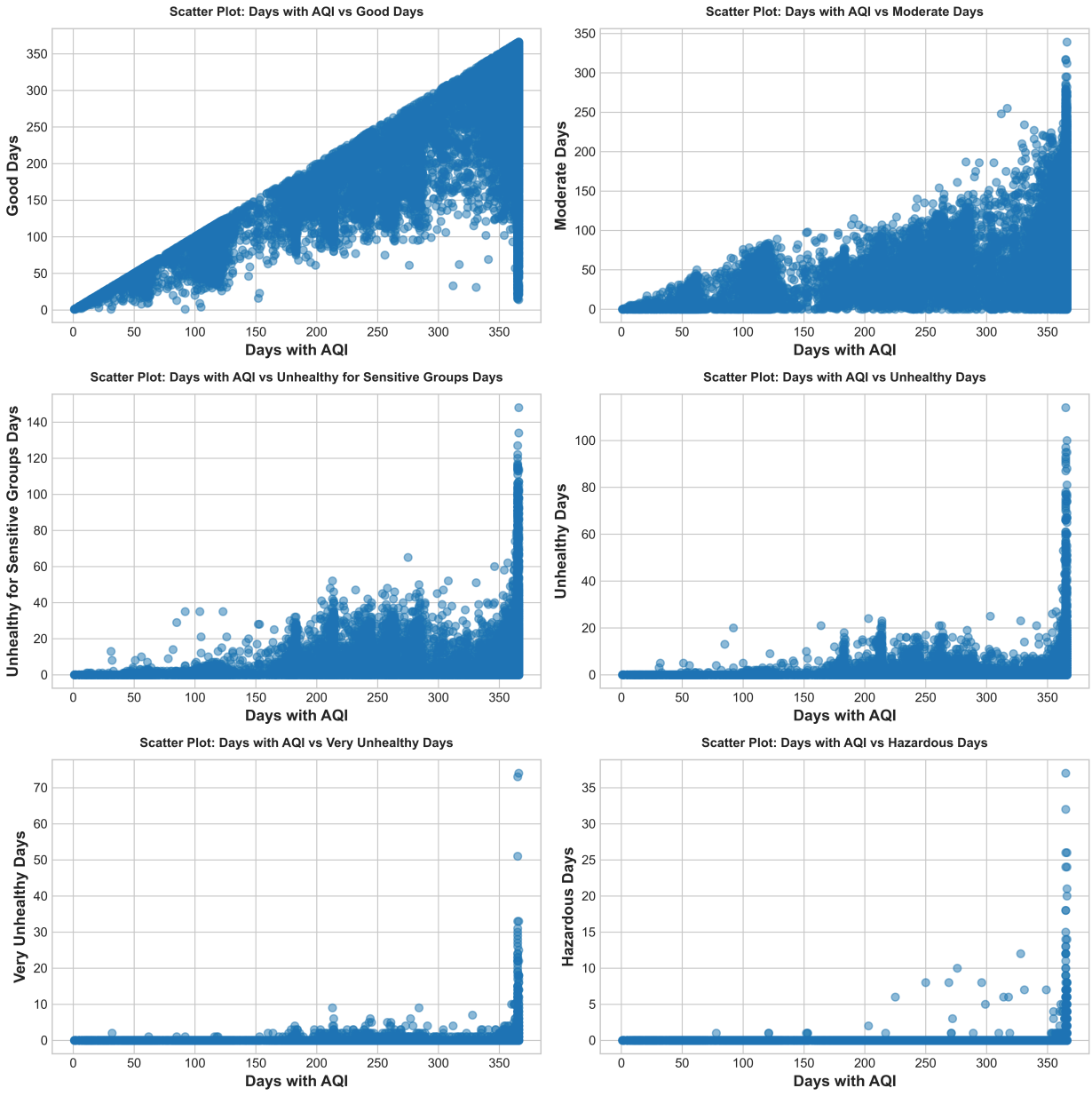
In [ ]:
```
# Merge AQI data from 2000 - 2019
all_aqi_data = pd.concat([AQI2000, AQI2001, AQI2002, AQI2003, AQI2004, AQI2005
                          AQI2011, AQI2012, AQI2013, AQI2014, AQI2015, AQI2016
```

In [ ]:
```
# Find NaN values
nan_values = all_aqi_data.isna().sum()
#print(nan_values) # 0 NaN values
#print(all_aqi_data.shape)

all_aqi_data
all_aqi_data = all_aqi_data.sort_values(by = ['County', 'Year'], ascending = T
#all_aqi_data

#columns_to_keep = ['State','FIPS', 'County', 'geometry']
columns_to_keep = ['STATE_NAME', "FIPS", 'NAME', 'geometry']
gdf_extra = county_gdf[columns_to_keep]

gdf_extra.rename(columns = {'STATE_NAME':'State', 'NAME':'County'}, inplace =

#merging with geospatial data
all_aqi_data = all_aqi_data.merge(gdf_extra, left_on = ['State', 'County'], ri
#all_aqi_data.head(3)
```

In [ ]:
```
geo_aqi_list = []

# Loop through years from 2000 to 2019
for year in range(2000, 2022):
    # Create a mask for the current year
    year_mask = (all_aqi_data['Year'] == year)

    # Filter data for the current year
    aqi_year = all_aqi_data[year_mask]

    # Merge AQI data with the county
    geo_aqi_year = county_gdf.merge(aqi_year)
```

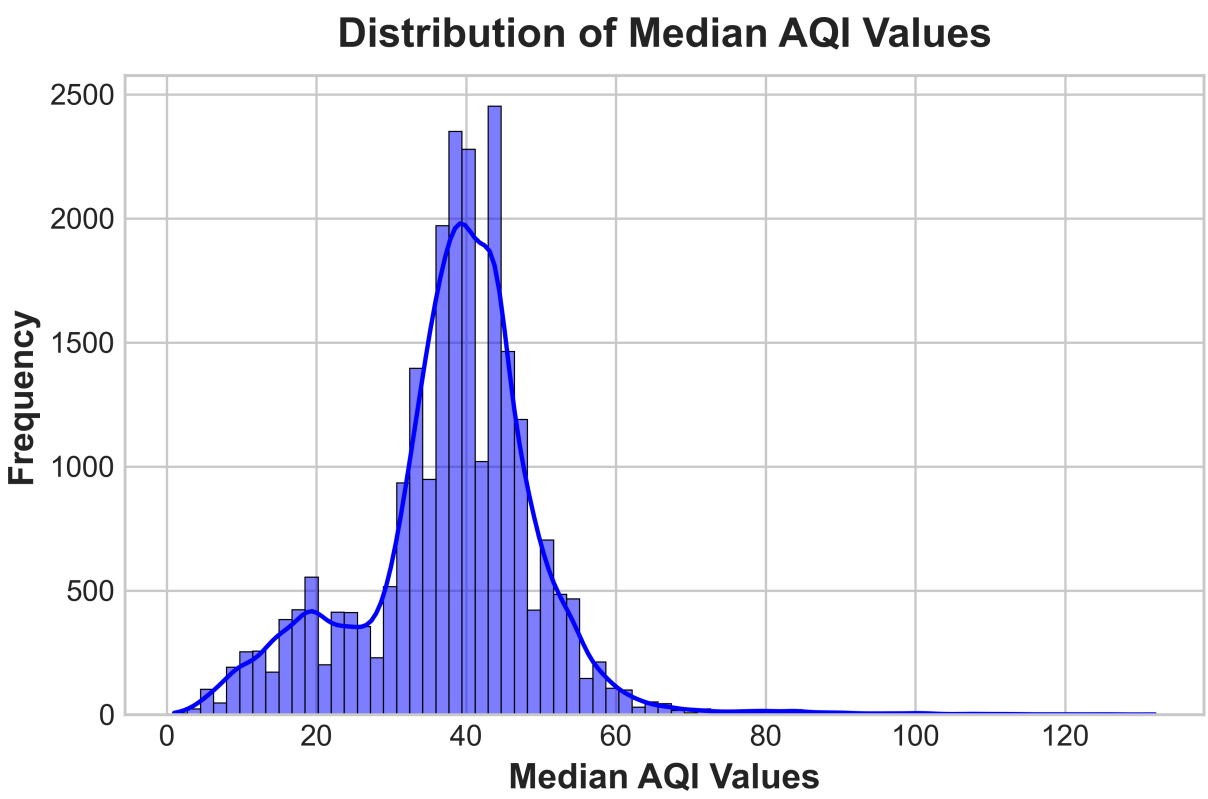
```
    geo_aqi_list.append(geo_aqi_year)
```

In [ ]:
```
categories = ['Good Days', 'Moderate Days', 'Unhealthy for Sensitive Groups Da

fig, axes = plt.subplots(3, 2, figsize=(12, 12), dpi=300)

for i, category in enumerate(categories):
    row, col = divmod(i, 2)
    axes[row, col].scatter(all_aqi_data['Days with AQI'], all_aqi_data[categor
    axes[row, col].set_title(f'Scatter Plot: Days with AQI vs {category}', fon
    axes[row, col].set_xlabel('Days with AQI')
    axes[row, col].set_ylabel(category)

plt.tight_layout()
plt.show()
```



In [ ]:
```
plt.figure(figsize=(6, 4), dpi=300)
sns.histplot(data=all_aqi_data, x='Median AQI', bins=75, kde=True, color='blue
plt.title('Distribution of Median AQI Values', fontsize=14, fontweight='bold')
```

```
plt.xlabel('Median AQI Values', fontsize=12, fontweight='bold')
plt.ylabel('Frequency', fontsize=12, fontweight='bold')
plt.show()
```



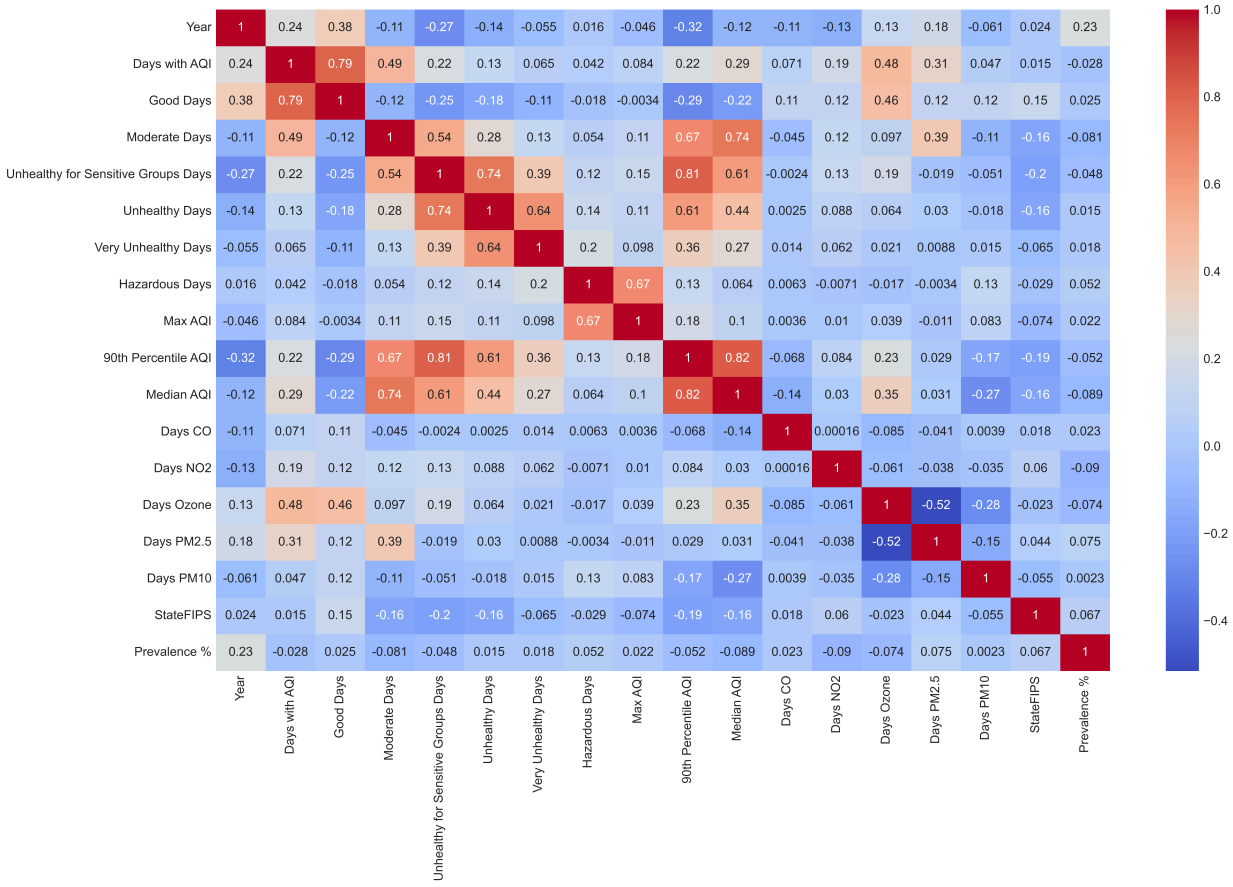
Asthma Prevalence and Air Quality

In [ ]:
```
aqi_prev = pd.merge(all_aqi_data, asthma_prev, on = ['State', 'County', 'Year']
```

In [ ]:
```
aqi_prev18 = aqi_prev[aqi_prev['Year'] == 2018]
geo_aqiprev18 = county_gdf.merge(aqi_prev18)

aqi_prev19 = aqi_prev[aqi_prev['Year'] == 2019]
geo_aqiprev19 = county_gdf.merge(aqi_prev19)

aqi_prev20 = aqi_prev[aqi_prev['Year'] == 2020]
geo_aqiprev20 = county_gdf.merge(aqi_prev20)

aqi_prev21 = aqi_prev[aqi_prev['Year'] == 2021]
geo_aqiprev21 = county_gdf.merge(aqi_prev21)
```

In [ ]:
```
aqi_prev_adjusted = aqi_prev.drop(columns=['geometry', 'State', 'County'])
fig, axes = plt.subplots(1, 1, figsize=(15, 10), dpi=300)
sns.heatmap(aqi_prev_adjusted.corr(), annot=True, cmap='coolwarm')
```

Out[ ]:
```
<Axes: >
```



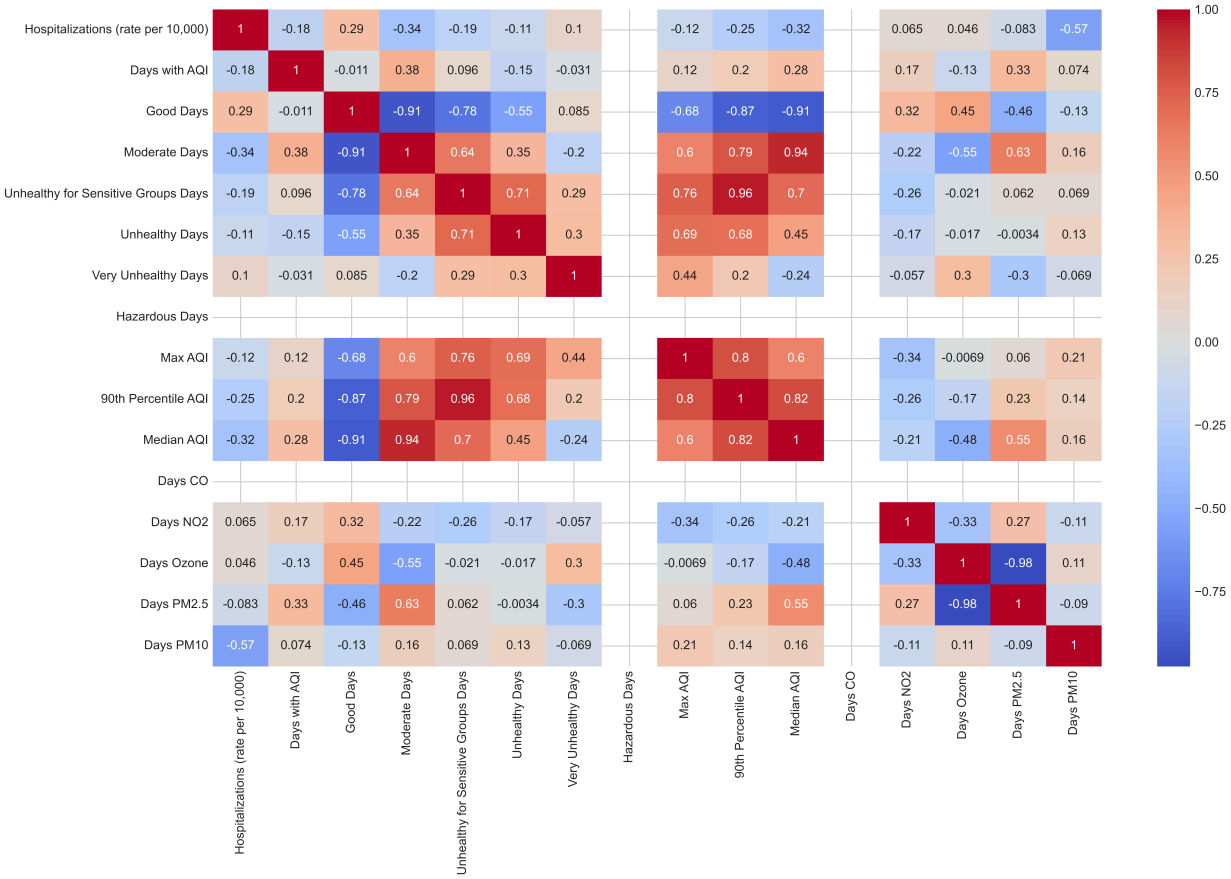
Asthma Hospitalizations and Air Quality

In [ ]:
```
hosp_IncA_full['FIPS'] = hosp_IncA_full['FIPS'].astype(float)
all_aqi_data["FIPS"] = all_aqi_data["FIPS"].astype("float64")
#all_aqi_data['FIPS'] = all_aqi_data['FIPS'].astype('int64')
#all_aqi_data.info()
```

In [ ]:
```
# New Group A - Increasing ER visits for Asthma
print(hosp_IncA_full.shape)
aqi_hospA = pd.merge(hosp_IncA_full, all_aqi_data, on = ['State', 'County', 'Y
print(aqi_hospA.shape)

aqi_hospA_nonan = aqi_hospA.dropna()
print(aqi_hospA_nonan.shape)

aqi_hospA_hm = aqi_hospA_nonan.drop(columns=['geometry', 'State', 'County', 's
fig, axes = plt.subplots(1, 1, figsize=(15, 10), dpi=300)
sns.heatmap(aqi_hospA_hm.corr(), annot=True, cmap='coolwarm')
```

```
(45, 8)
(45, 24)
(30, 24)
```

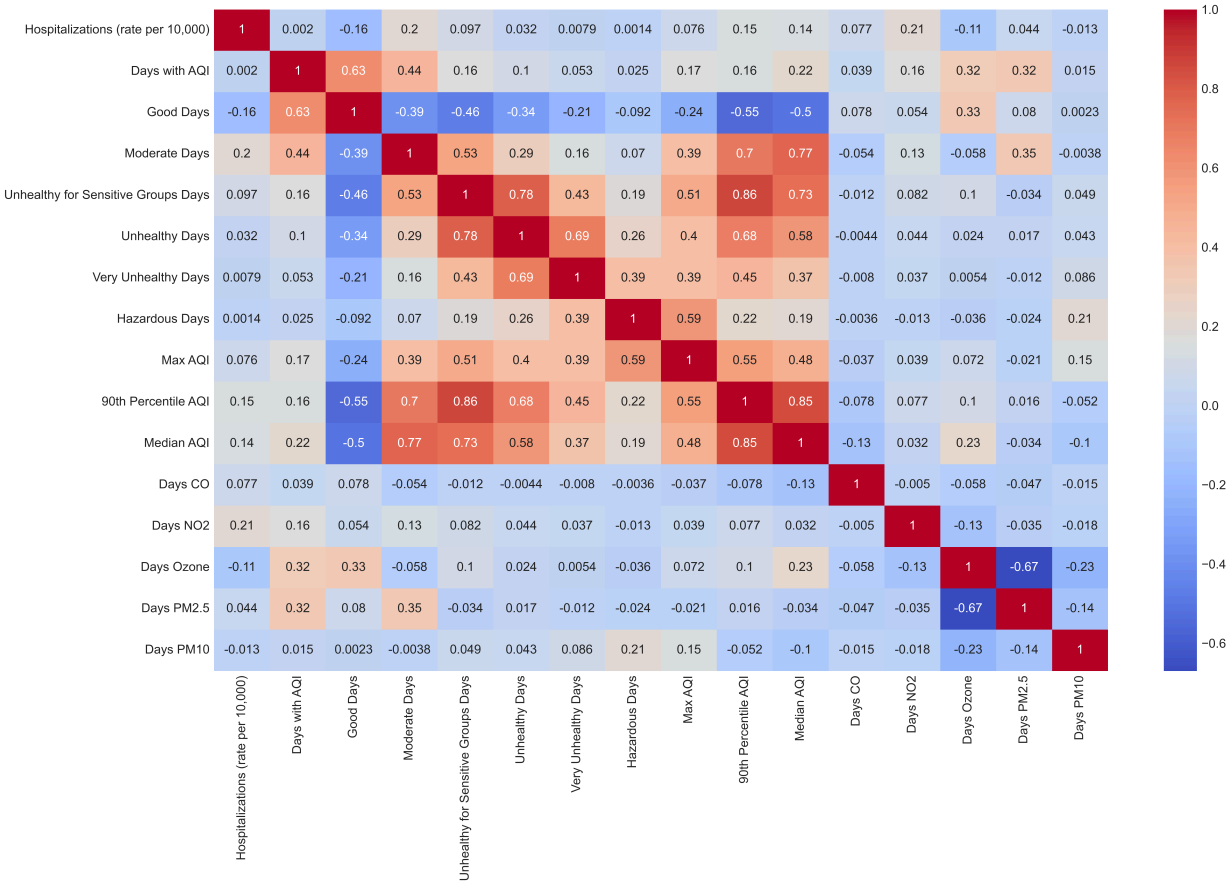
Out[ ]:
```
<Axes: >
```



In [ ]:
```
## Hospitalizations Group B
hosp_DecB_full['FIPS'] = hosp_DecB_full['FIPS'].astype(float)
#all_aqi_data["FIPS"] = all_aqi_data["FIPS"].astype("float64")

# New Group A - Increasing ER visits for Asthma
print(hosp_DecB_full.shape)
aqi_hospB = pd.merge(hosp_DecB_full, all_aqi_data, on = ['State', 'County', 'Y
print(aqi_hospB.shape)

aqi_hospB_nonan = aqi_hospB.dropna()
print(aqi_hospB_nonan.shape)

aqi_hospB_hm = aqi_hospB_nonan.drop(columns=['geometry', 'State', 'County', 's
fig, axes = plt.subplots(1, 1, figsize=(15, 10), dpi=300)
sns.heatmap(aqi_hospB_hm.corr(), annot=True, cmap='coolwarm')
```

```
(10065, 8)
(10065, 24)
(6311, 24)
```

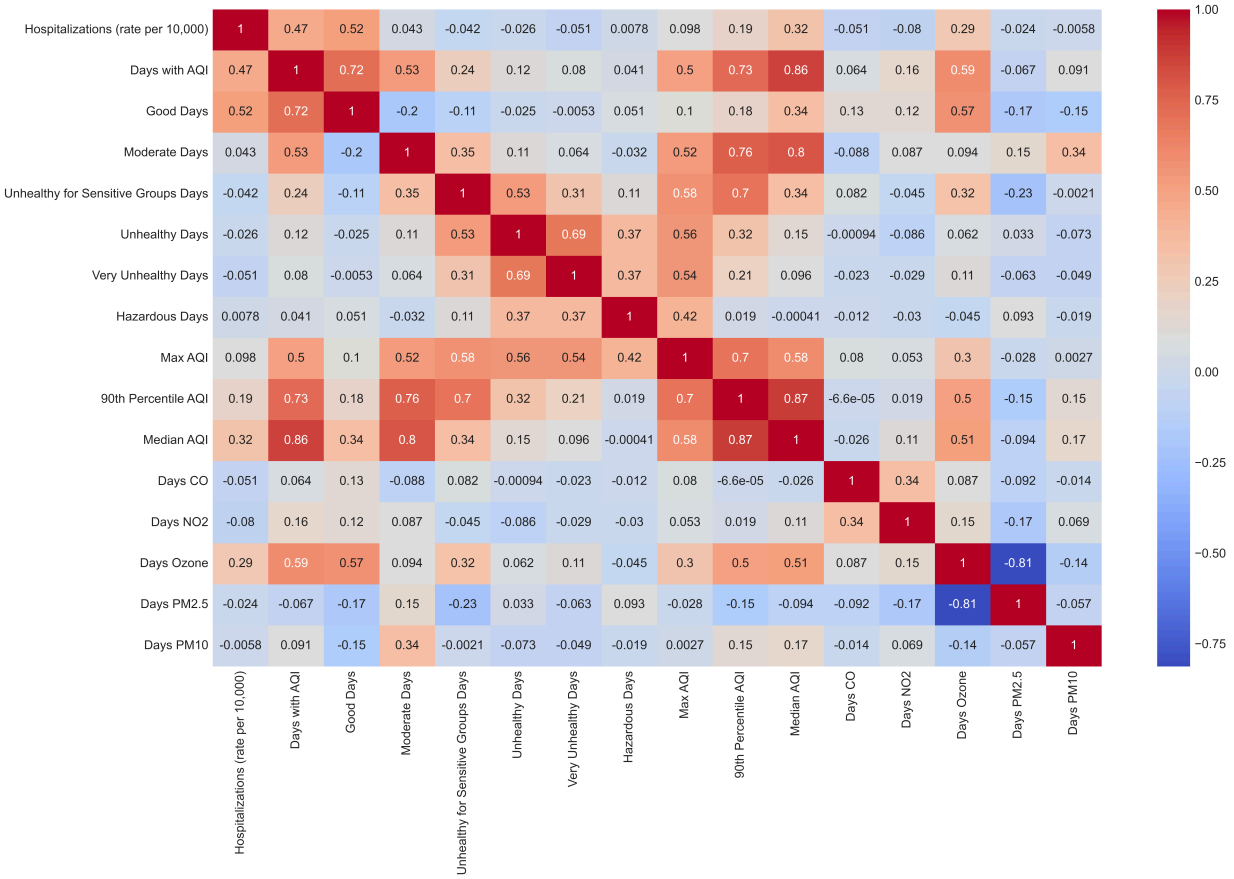
Out[ ]:
```
<Axes: >
```



In [ ]:
```
## Hospitalizations Group C
hosp_neutralC_full['FIPS'] = hosp_neutralC_full['FIPS'].astype(float)
#all_aqi_data["FIPS"] = all_aqi_data["FIPS"].astype("float64")

# New Group A - Increasing ER visits for Asthma
print(hosp_neutralC_full.shape)
aqi_hospC = pd.merge(hosp_neutralC_full, all_aqi_data, on = ['State', 'County'
print(aqi_hospC.shape)

aqi_hospC_nonan = aqi_hospC.dropna()
print(aqi_hospC_nonan.shape)

aqi_hospC_hm = aqi_hospC_nonan.drop(columns=['geometry', 'State', 'County', 's
fig, axes = plt.subplots(1, 1, figsize=(15, 10), dpi=300)
sns.heatmap(aqi_hospC_hm.corr(), annot=True, cmap='coolwarm')
```

```
(225, 8)
(225, 24)
(126, 24)
```

Out[ ]:
```
<Axes: >
```

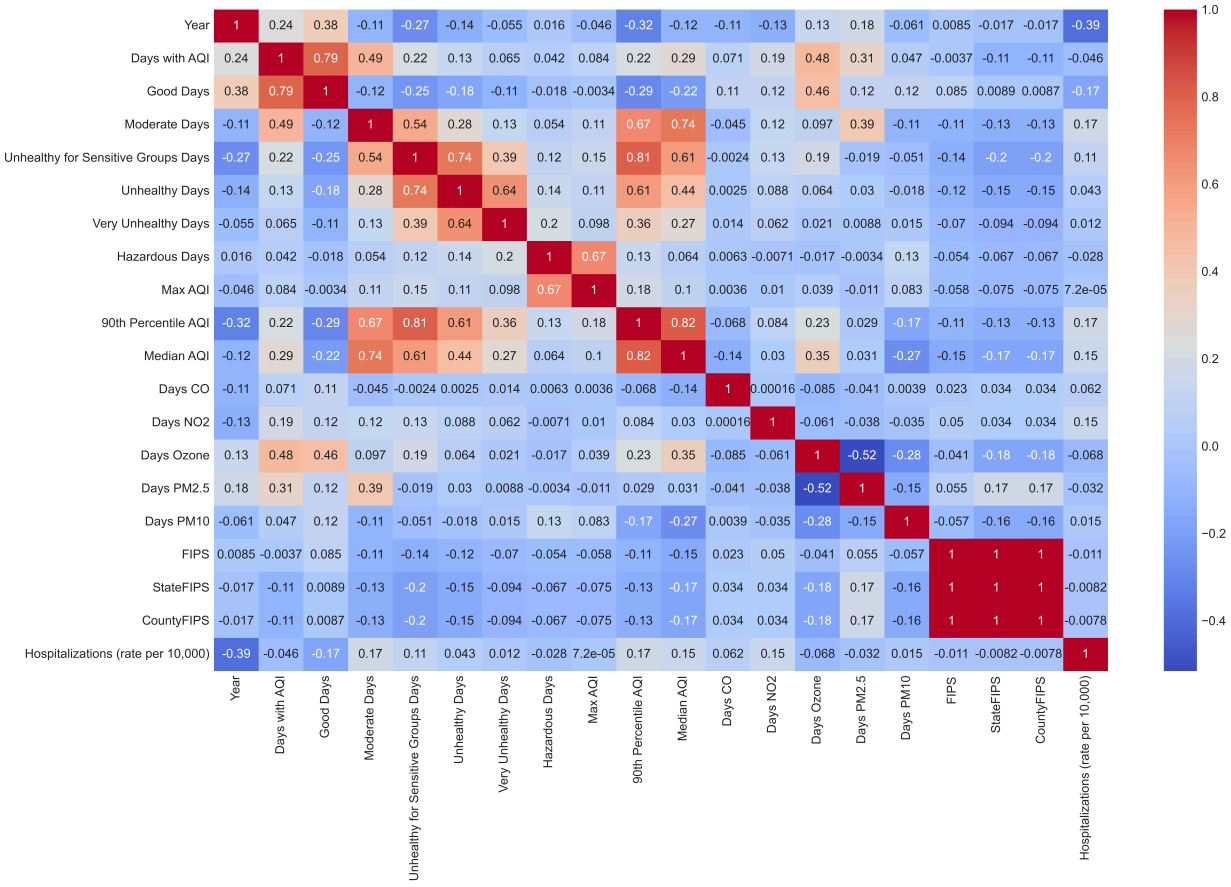


In [ ]:
```
aqi_hosp = pd.merge(all_aqi_data, asthma_hosp, on = ['State', 'County', 'Year']
```

In [ ]:
```
measures = ['Good Days',
       'Moderate Days', 'Unhealthy for Sensitive Groups Days',
       'Unhealthy Days', 'Very Unhealthy Days', 'Hazardous Days', 'Max AQI', 'M
       'Days Ozone', 'Days PM2.5', 'Days PM10', 'Year', 'Hospitalizations (rat
```

In [ ]:
```
aqi_hosp_adjusted = aqi_hosp.drop(columns=['geometry', 'State', 'County'])
fig, axes = plt.subplots(1, 1, figsize=(15, 10), dpi=300)
sns.heatmap(aqi_hosp_adjusted.corr(), annot=True, cmap='coolwarm')
```

Out[ ]:
```
<Axes: >
```



In [ ]:
```
counties = aqi_hosp['County'].unique()

for county in counties:

    # Filter the data for the current county
    county_data = aqi_hosp[aqi_hosp['County'] == county]

    # Exclude non-numeric columns
    numeric_columns = county_data.select_dtypes(include='number')

    corr_matrix = numeric_columns.corr()

    # Tail of the correlation matrix
    tail_corr_values = corr_matrix.iloc[-1]

    #print(f"Correlation values for {county} County:")
    #print(tail_corr_values)
```

In [ ]:
```
counties = aqi_hosp['County'].unique()

# Dict with correlation values
correlation_values = {}
for county in counties:
    county_data = aqi_hosp[aqi_hosp['County'] == county]
    numeric_columns = county_data.select_dtypes(include='number')
    corr_matrix = numeric_columns.corr()

    # Correlation values for pollutants
    relevant_corr_values = corr_matrix.loc[['Days CO', 'Days NO2', 'Days Ozone
```

```
    avg_corr = relevant_corr_values.mean().mean()

    correlation_values[county] = avg_corr

# Sort counties based on the average correlation with pollutants
sorted_counties = sorted(correlation_values, key=correlation_values.get, revers

for county in sorted_counties:
    county_data = aqi_hosp[aqi_hosp['County'] == county]
    numeric_columns = county_data.select_dtypes(include='number')
    corr_matrix = numeric_columns.corr()
    tail_corr_values = corr_matrix.iloc[-1]

    #print(f"Correlation values for {county} County:")
    #print(tail_corr_values)
    #print()
```

In [ ]:

```
## Note: This is with NaN values removed for the pollutants


# Dict with correlation values
correlation_values = {}
for county in counties:
    county_data = aqi_hosp[aqi_hosp['County'] == county]
    numeric_columns = county_data.select_dtypes(include='number')
    corr_matrix = numeric_columns.corr()

    # Correlation values for pollutants
    relevant_corr_values = corr_matrix.loc[['Days CO', 'Days NO2', 'Days Ozone

    # Exclude counties with all NaN values for the specified columns
    if not relevant_corr_values.isnull().all().any():
        avg_corr = relevant_corr_values.mean().mean()

        correlation_values[county] = avg_corr

# Sort counties based on the average correlation with pollutants
sorted_counties = sorted(correlation_values, key=correlation_values.get, revers

for county in sorted_counties:
    county_data = aqi_hosp[aqi_hosp['County'] == county]
    numeric_columns = county_data.select_dtypes(include='number')
    corr_matrix = numeric_columns.corr()
    tail_corr_values = corr_matrix.iloc[-1]

    print(f"Correlation values for {county} County:")
    print(tail_corr_values)
    print()
```

```
Correlation values for Jackson County:
Year                                -0.428444
Days with AQI                        0.359284
Good Days                            0.000079
Moderate Days                        0.708113
Unhealthy for Sensitive Groups Days  0.128348
Unhealthy Days                      -0.199151
Very Unhealthy Days                 -0.235855
Hazardous Days                      -0.141590
Max AQI                              0.001332
90th Percentile AQI                  0.403581
Median AQI                           0.601133
Days CO                              0.055724
Days NO2                             0.662789
Days Ozone                          -0.346061
Days PM2.5                           0.435953
Days PM10                            0.189569
FIPS                                 0.254988
StateFIPS                            0.253138
CountyFIPS                           0.254988
Hospitalizations (rate per 10,000)   1.000000
Name: Hospitalizations (rate per 10,000), dtype: float64

Correlation values for Marion County:
Year                                -0.430530
Days with AQI                       -0.493409
Good Days                           -0.589334
Moderate Days                        0.282418
Unhealthy for Sensitive Groups Days  0.208313
Unhealthy Days                      -0.006940
Very Unhealthy Days                       NaN
Hazardous Days                      -0.184122
Max AQI                             -0.171721
90th Percentile AQI                  0.552884
Median AQI                           0.786528
Days CO                             -0.242183
Days NO2                                  NaN
Days Ozone                          -0.014884
Days PM2.5                          -0.598803
Days PM10                                 NaN
FIPS                                -0.114286
StateFIPS                           -0.114529
CountyFIPS                          -0.114286
Hospitalizations (rate per 10,000)   1.000000
Name: Hospitalizations (rate per 10,000), dtype: float64

Correlation values for Orange County:
Year                                -0.276754
Days with AQI                        0.150443
Good Days                            0.368552
Moderate Days                       -0.280517
Unhealthy for Sensitive Groups Days -0.167531
Unhealthy Days                      -0.036168
Very Unhealthy Days                  0.006128
Hazardous Days                      -0.085089
Max AQI                              0.106586
90th Percentile AQI                 -0.168401
Median AQI                          -0.370868
Days CO                             -0.153420
Days NO2                            -0.382673
```

```
Days Ozone                              0.281019
Days PM2.5                             -0.058239
Days PM10                              -0.382966
FIPS                                    0.139976
StateFIPS                               0.140059
CountyFIPS                              0.139976
Hospitalizations (rate per 10,000)      1.000000
Name: Hospitalizations (rate per 10,000), dtype: float64

Correlation values for Jefferson County:
Year                                   -0.575546
Days with AQI                           0.163179
Good Days                              -0.386955
Moderate Days                           0.631421
Unhealthy for Sensitive Groups Days     0.413553
Unhealthy Days                          0.398060
Very Unhealthy Days                     0.110684
Hazardous Days                         -0.056431
Max AQI                                 0.344619
90th Percentile AQI                     0.501848
Median AQI                              0.513160
Days CO                                 0.323804
Days NO2                                0.428528
Days Ozone                             -0.159910
Days PM2.5                              0.212129
Days PM10                               0.025852
FIPS                                   -0.304234
StateFIPS                              -0.304910
CountyFIPS                             -0.304234
Hospitalizations (rate per 10,000)      1.000000
Name: Hospitalizations (rate per 10,000), dtype: float64

Correlation values for Linn County:
Year                                   -0.551875
Days with AQI                          -0.123725
Good Days                              -0.173001
Moderate Days                           0.019864
Unhealthy for Sensitive Groups Days     0.302350
Unhealthy Days                         -0.233499
Very Unhealthy Days                    -0.185141
Hazardous Days                         -0.185141
Max AQI                                -0.183800
90th Percentile AQI                     0.249879
Median AQI                              0.302159
Days CO                                 0.019840
Days NO2                                0.474442
Days Ozone                              0.558097
Days PM2.5                             -0.620055
Days PM10                              -0.071393
FIPS                                   -0.349633
StateFIPS                              -0.349579
CountyFIPS                             -0.349633
Hospitalizations (rate per 10,000)      1.000000
Name: Hospitalizations (rate per 10,000), dtype: float64

Correlation values for Santa Cruz County:
Year                                   -0.566759
Days with AQI                          -0.069326
Good Days                              -0.234857
Moderate Days                           0.239980
```

```
Unhealthy for Sensitive Groups Days    0.266353
Unhealthy Days                        -0.011304
Very Unhealthy Days                         NaN
Hazardous Days                              NaN
Max AQI                               -0.022408
90th Percentile AQI                    0.207943
Median AQI                             0.514203
Days CO                                0.009961
Days NO2                               0.200364
Days Ozone                            -0.055059
Days PM2.5                            -0.570230
Days PM10                              0.397494
FIPS                                  -0.152190
StateFIPS                             -0.152190
CountyFIPS                            -0.152190
Hospitalizations (rate per 10,000)     1.000000
Name: Hospitalizations (rate per 10,000), dtype: float64

Correlation values for Kings County:
Year                                  -0.436214
Days with AQI                         -0.064350
Good Days                              0.558897
Moderate Days                         -0.425212
Unhealthy for Sensitive Groups Days   -0.660799
Unhealthy Days                        -0.498877
Very Unhealthy Days                   -0.193112
Hazardous Days                        -0.195273
Max AQI                               -0.424403
90th Percentile AQI                   -0.597814
Median AQI                            -0.655864
Days CO                                0.672825
Days NO2                              -0.343567
Days Ozone                            -0.715256
Days PM2.5                             0.269817
Days PM10                             -0.595886
FIPS                                   0.731770
StateFIPS                              0.731770
CountyFIPS                             0.731770
Hospitalizations (rate per 10,000)     1.000000
Name: Hospitalizations (rate per 10,000), dtype: float64
```

In [ ]:
```
#correlation by variable for all counties
aqi_hosp.corr().tail(1)
```

Out[ ]:

	Year	Days with AQI	Good Days	Moderate Days	Unhealthy for Sensitive Groups Days	Unhealthy Days	Very Unhealthy Days
Hospitalizations (rate per 10,000)	-0.392238	-0.045975	-0.173514	0.168859	0.113351	0.04337	0.011988

In [ ]:
```
# Count the NaN values in the 'Days CO' column of the 'aqi_hosp' DataFrame
nan_count = aqi_hosp['Hospitalizations (rate per 10,000)'].isna().sum()
print(nan_count)

aqi_hosp_nonans = aqi_hosp['Hospitalizations (rate per 10,000)'].dropna
```

```
11156
```

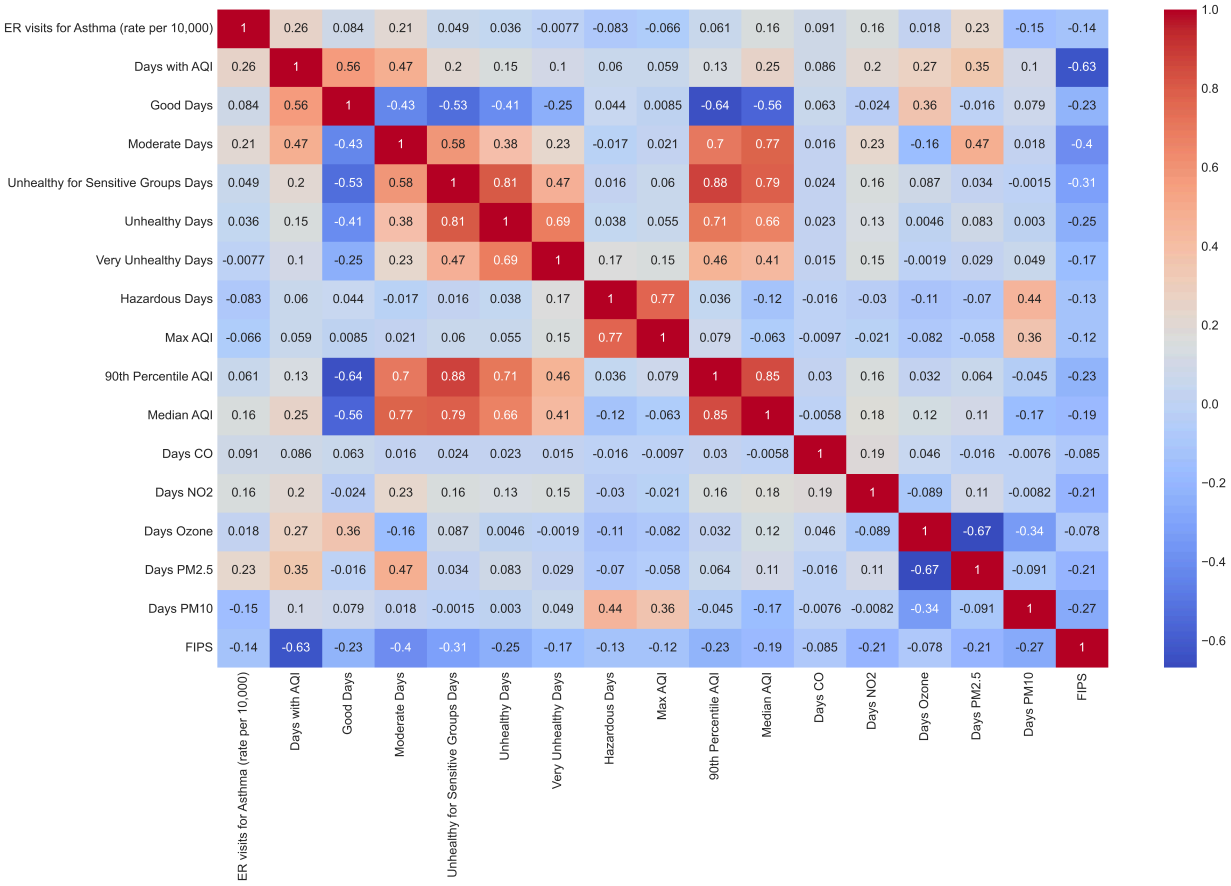
Asthma ER Visits and Air Quality

In [ ]:
```
aqi_er = pd.merge(all_aqi_data, asthma_er, on = ['State', 'County', 'Year'], h
```

In [ ]:
```
# New Group A - Increasing ER visits for Asthma
print(ER_IncA_full.shape)
aqi_erA = pd.merge(ER_IncA_full, all_aqi_data, on = ['State', 'County', 'Year'
print(aqi_erA.shape)

aqi_erA_nonan = aqi_erA.dropna()
print(aqi_erA_nonan.shape)

aqi_erA_hm = aqi_erA_nonan.drop(columns=['geometry', 'State', 'County', 'cname
fig, axes = plt.subplots(1, 1, figsize=(15, 10), dpi=300)
sns.heatmap(aqi_erA_hm.corr(), annot=True, cmap='coolwarm')
```

```
(3142, 8)
(3142, 25)
(1595, 25)
```

Out[ ]:
```
<Axes: >
```

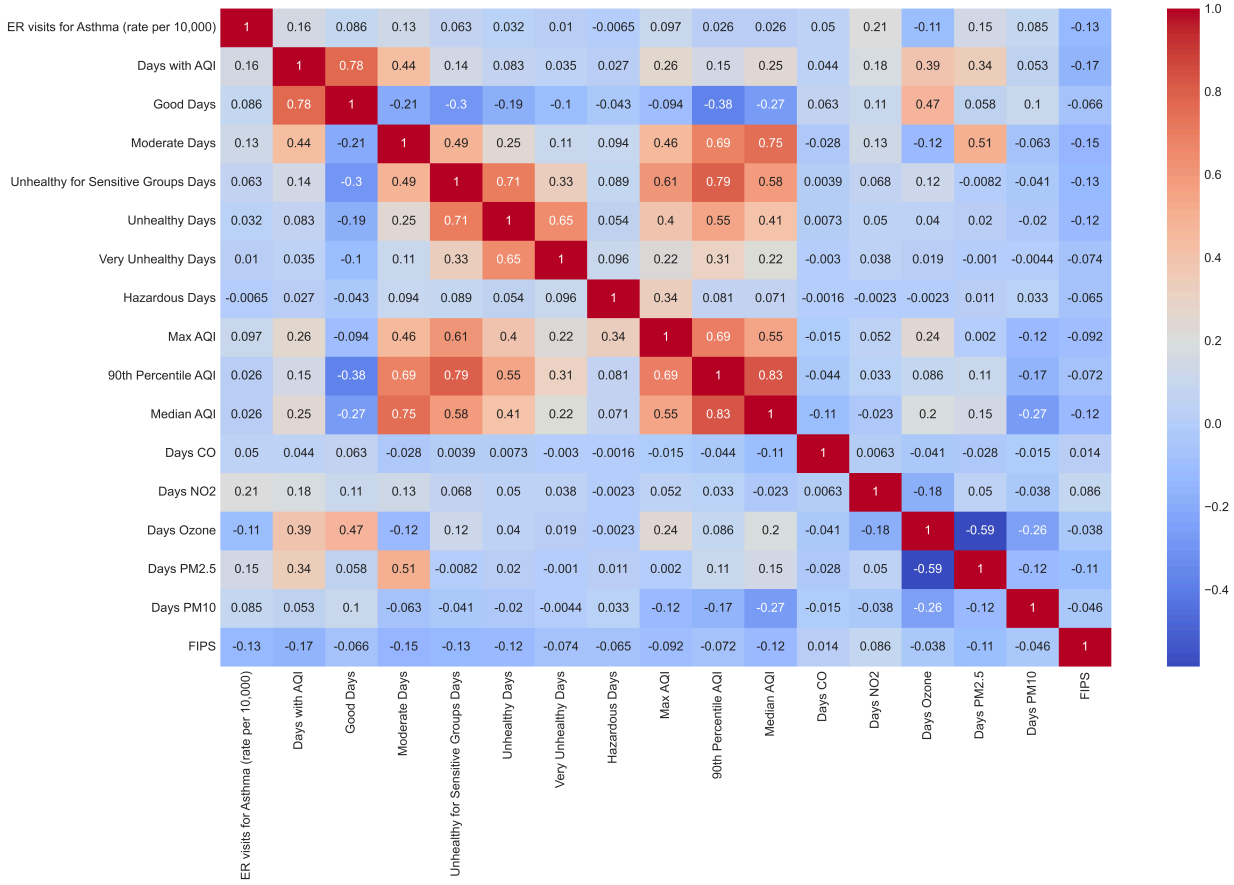


In [ ]:
```
# New Group B - Decreasing ER visits for Asthma

aqi_erB = pd.merge(ER_DecB_full, all_aqi_data, on = ['State', 'County', 'Year'
print(aqi_erB.shape)

aqi_erB_nonan = aqi_erB.dropna()
print(aqi_erB_nonan.shape)
```

```
aqi_erB_hm = aqi_erB_nonan.drop(columns=['geometry', 'State', 'County', 'cname
fig, axes = plt.subplots(1, 1, figsize=(15, 10), dpi=300)
sns.heatmap(aqi_erB_hm.corr(), annot=True, cmap='coolwarm')
```

```
(23343, 25)
(8472, 25)
<Axes: >
```

Out[ ]:

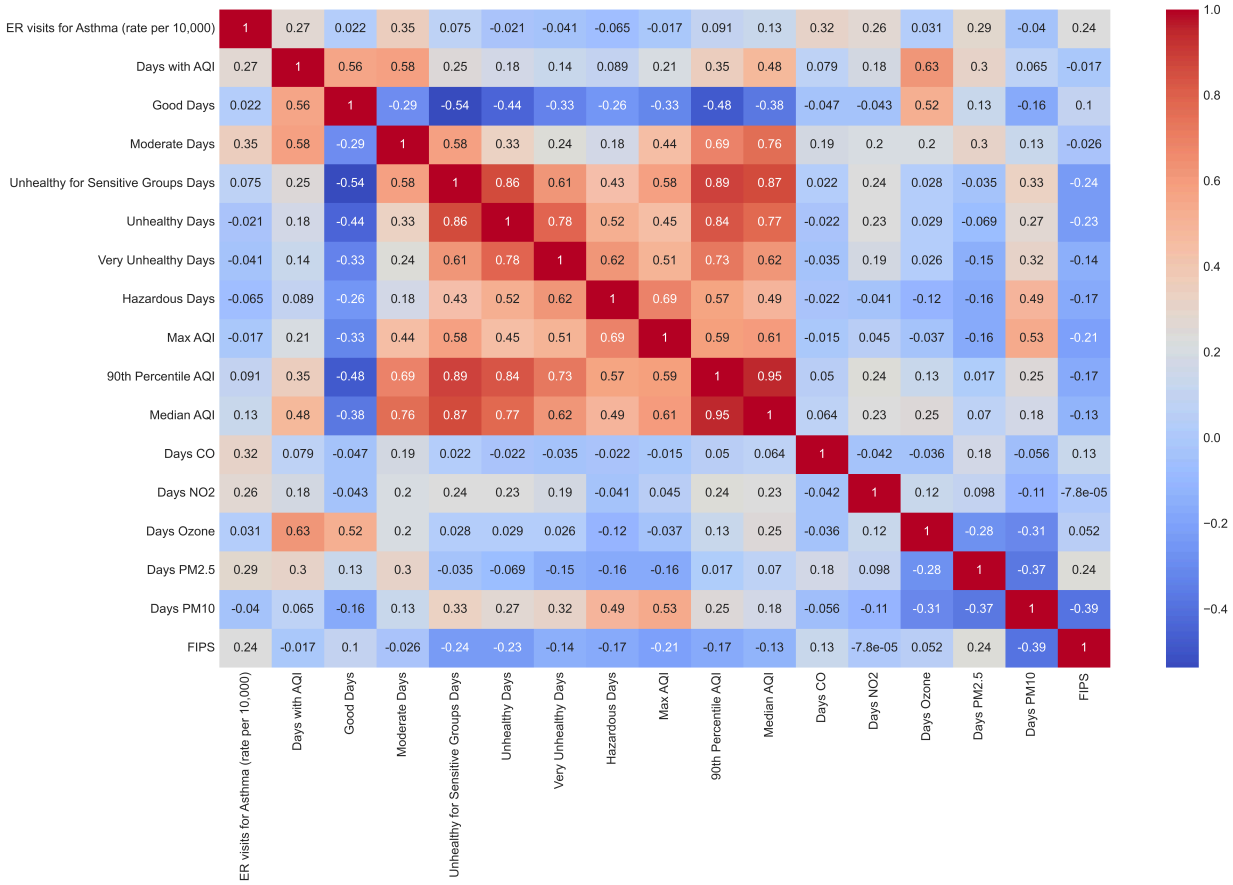


In [ ]:

```
# New Group C - Neutral ER visits for Asthma

aqi_erC = pd.merge(ER_neutralC_full, all_aqi_data, on = ['State', 'County', 'Y
print(aqi_erC.shape)

aqi_erC_nonan = aqi_erC.dropna()
print(aqi_erC_nonan.shape)

aqi_erC_hm = aqi_erC_nonan.drop(columns=['geometry', 'State', 'County', 'cname
fig, axes = plt.subplots(1, 1, figsize=(15, 10), dpi=300)
sns.heatmap(aqi_erC_hm.corr(), annot=True, cmap='coolwarm')
```

```
(665, 25)
(329, 25)
<Axes: >
```

Out[ ]:

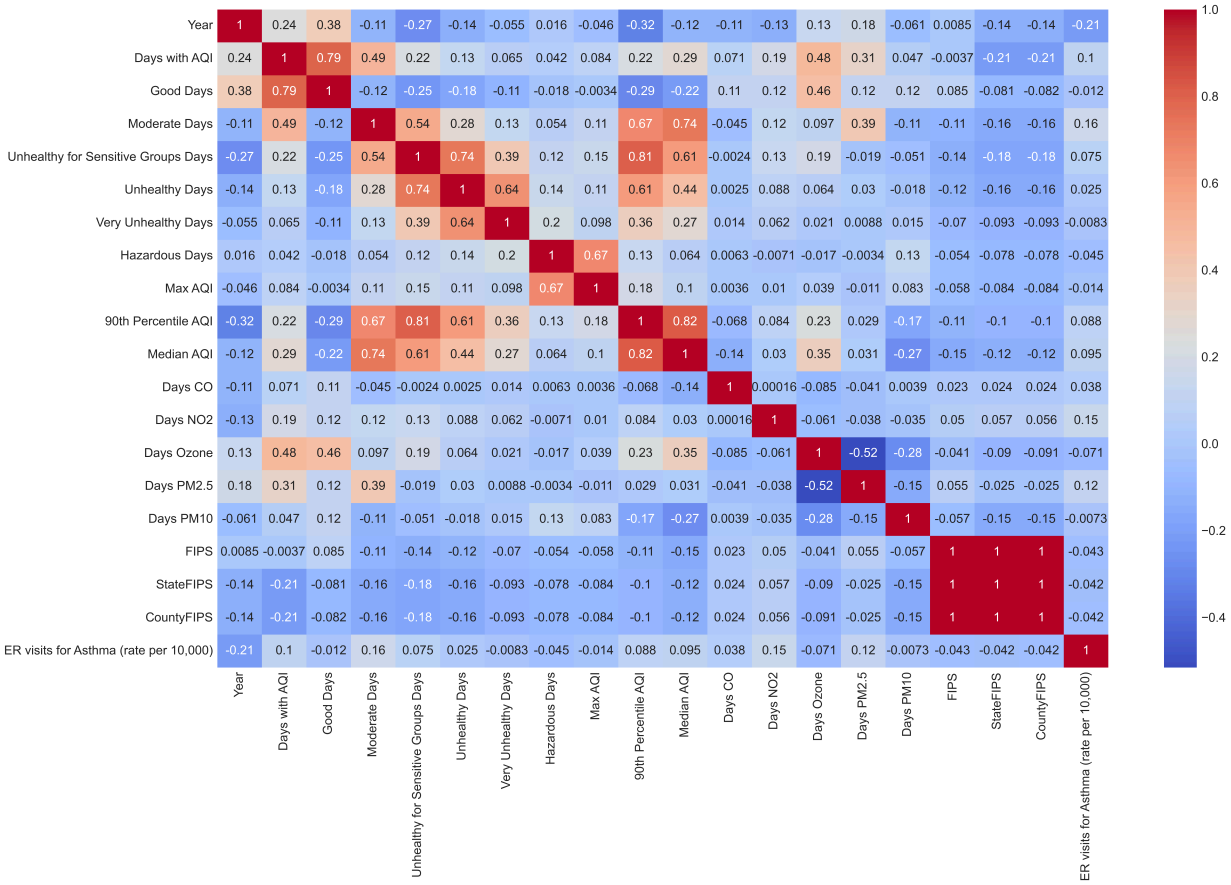


In [ ]:
```
aqi_er.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 23542 entries, 0 to 23541
Data columns (total 23 columns):
 #   Column                                    Non-Null Count  Dtype
---  ------                                    --------------  -----
 0   State                                     23542 non-null  object
 1   County                                    23542 non-null  object
 2   Year                                      23542 non-null  int64
 3   Days with AQI                             23542 non-null  int64
 4   Good Days                                 23542 non-null  int64
 5   Moderate Days                             23542 non-null  int64
 6   Unhealthy for Sensitive Groups Days       23542 non-null  int64
 7   Unhealthy Days                            23542 non-null  int64
 8   Very Unhealthy Days                       23542 non-null  int64
 9   Hazardous Days                            23542 non-null  int64
 10  Max AQI                                   23542 non-null  int64
 11  90th Percentile AQI                       23542 non-null  int64
 12  Median AQI                                23542 non-null  int64
 13  Days CO                                   23542 non-null  int64
 14  Days NO2                                  23542 non-null  int64
 15  Days Ozone                                23542 non-null  int64
 16  Days PM2.5                                23542 non-null  int64
 17  Days PM10                                 23542 non-null  int64
 18  FIPS                                      22369 non-null  float64
 19  geometry                                  22369 non-null  geometry
 20  StateFIPS                                 8366 non-null   float64
 21  CountyFIPS                                8366 non-null   float64
 22  ER visits for Asthma (rate per 10,000)  8366 non-null   float64
dtypes: float64(4), geometry(1), int64(16), object(2)
memory usage: 4.3+ MB
```

In [ ]:
```
aqi_er_adjusted= aqi_er.drop(columns=['geometry', 'State', 'County'])
fig, axes = plt.subplots(1, 1, figsize=(15, 10), dpi=300)
sns.heatmap(aqi_er_adjusted.corr(), annot=True, cmap='coolwarm')
```

Out[ ]:
```
<Axes: >
```



Correlation Dataframes

In [ ]:
```
hosp_aqi_corr = pd.DataFrame()

# Iterate over each county
for county in aqi_hosp['County'].unique():
    cname = county
    state = aqi_hosp[aqi_hosp['County'] == county]['State'].unique()[0]
    county_df = aqi_hosp[aqi_hosp['County'] == county].corr()
    county_df.insert(0, "County", cname, True)
    county_df.insert(1, "State", state, True)

    tail_end = county_df.tail(1)
    hosp_aqi_corr = pd.concat([hosp_aqi_corr, tail_end])

#hosp_aqi_corr = hosp_aqi_corr.drop(columns = ['Year']) #year corr values don'
hosp_aqi_corr.tail(3)
#aqi_hosp.corr().tail(1)
```

Out[ ]:

	County	State	Year	Days with AQI	Good Days	Moderate Days	Unhealthy for Sensitive Groups Days	Unhe
Hospitalizations (rate per 10,000)	York	Maine	-0.760312	-0.017328	-0.371828	0.233779	0.511511	0.32
Hospitalizations (rate per 10,000)	Yukon-Koyukuk	Alaska	NaN	NaN	NaN	NaN	NaN	
Hospitalizations (rate per 10,000)	Yuma	Arizona	-0.918819	-0.433850	-0.423845	0.110447	0.525274	0.0

3 rows × 22 columns

In [ ]:
```
prev_aqi_corr = pd.DataFrame()

# Iterate over each county
for county in aqi_prev['County'].unique():
    cname = county
    state = aqi_prev[aqi_prev['County'] == county]['State'].unique()[0]
    county_df = aqi_prev[aqi_prev['County'] == county].corr()
    county_df.insert(0, "County", cname, True)
    county_df.insert(1, "State", state, True)

    tail_end = county_df.tail(1)
    prev_aqi_corr = pd.concat([prev_aqi_corr, tail_end])

#hosp_aqi_corr = hosp_aqi_corr.drop(columns = ['Year']) #year corr values don'
prev_aqi_corr.tail(3)
#aqi_hosp.corr().tail(1)
```

Out[ ]:

	County	State	Year	Days with AQI	Good Days	Moderate Days	Unhealthy for Sensitive Groups Days	Unhealth Day
Prevalence %	York	Maine	2.389638e-01	-0.718811	-0.727756	-0.360480	-0.260328	Nal
Prevalence %	Yukon-Koyukuk	Alaska	NaN	NaN	NaN	NaN	NaN	Nal
Prevalence %	Yuma	Arizona	-1.621585e-15	0.471405	-0.510255	0.398194	0.434643	0.40824

In [ ]:
```
er_aqi_corr = pd.DataFrame()

# Iterate over each county
for county in aqi_er['County'].unique():
    cname = county
    state = aqi_er[aqi_er['County'] == county]['State'].unique()[0]
    county_df = aqi_er[aqi_er['County'] == county].corr()
    county_df.insert(0, "County", cname, True)
```

```
    county_df.insert(1, "State", state, True)

    tail_end = county_df.tail(1)
    er_aqi_corr = pd.concat([er_aqi_corr, tail_end])

#hosp_aqi_corr = hosp_aqi_corr.drop(columns = ['Year']) #year corr values don'
er_aqi_corr.tail(3)
#aqi_hosp.corr().tail(1)
```

Out[ ]:

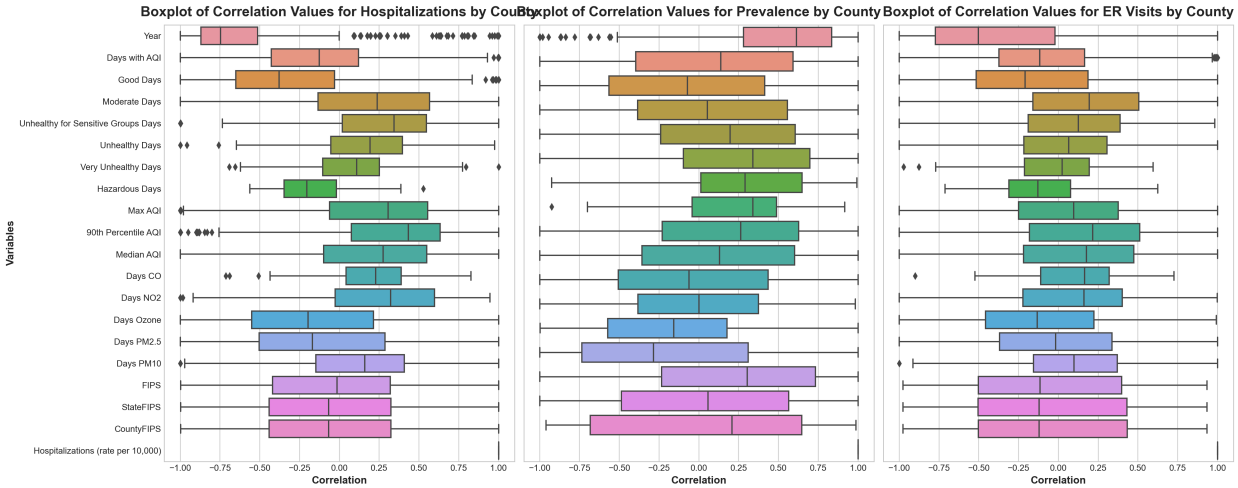
	County	State	Year	Days with AQI	Good Days	Moderate Days	Unhealthy for Sensitive Groups Days	Unhealthy Days	U
ER visits for Asthma (rate per 10,000)	York	Maine	-0.324796	0.301857	0.261300	-0.080697	0.204821	0.166744	
ER visits for Asthma (rate per 10,000)	Yukon-Koyukuk	Alaska	NaN	NaN	NaN	NaN	NaN	NaN	
ER visits for Asthma (rate per 10,000)	Yuma	Arizona	-0.847654	-0.217376	-0.232214	-0.009468	0.546024	0.062430	

3 rows × 22 columns

In [ ]:

```
fig, axs = plt.subplots(1, 3, figsize=(20, 8))

sns.boxplot(ax=axs[0], data=hosp_aqi_corr, orient='h')
axs[0].set_title("Boxplot of Correlation Values for Hospitalizations by County'
axs[0].set_xlabel('Correlation')
axs[0].set_ylabel('Variables')


sns.boxplot(ax=axs[1], data=prev_aqi_corr, orient='h')
axs[1].set_title("Boxplot of Correlation Values for Prevalence by County")
axs[1].set_xlabel('Correlation')
axs[1].set_yticklabels([])

sns.boxplot(ax=axs[2], data=er_aqi_corr, orient='h')
axs[2].set_title("Boxplot of Correlation Values for ER Visits by County")
axs[2].set_xlabel('Correlation')
axs[2].set_yticklabels([])
```

```
plt.show()
```



Results

Models for ER Group A (Increasing)

In [ ]:
```
#aqi_erA.info()
#aqi_erA.isnull().sum()
aqi_erA.tail(10)
aqi_erA_adj = aqi_erA.dropna()
aqi_erA_adj.shape
```

Out[ ]:
```
(1595, 25)
```

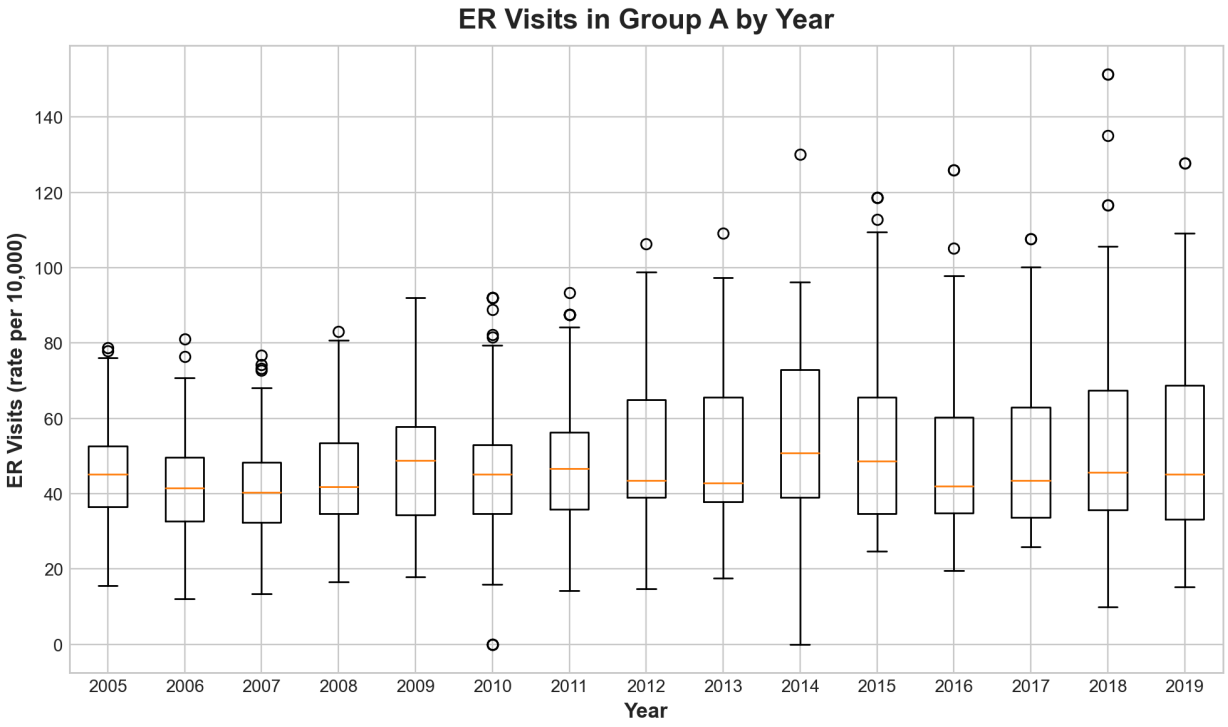
In [ ]:
```
aqi_erA_adj
```

Out[ ]:

	cname	slope	StateFIPS	State	CountyFIPS	County	Year	ER visits for Asthma (rate per 10,000)	Days with AQ
0	Apache	0.130357	4	Arizona	4001	Apache	2005	31.0	336.0
1	Apache	0.130357	4	Arizona	4001	Apache	2006	21.1	339.0
2	Apache	0.130357	4	Arizona	4001	Apache	2007	21.1	365.0
3	Apache	0.130357	4	Arizona	4001	Apache	2008	20.2	295.0
4	Apache	0.130357	4	Arizona	4001	Apache	2009	44.0	364.0
...	...	...	...	...	...	...	...	...	...
3122	Sheboygan	0.242857	55	Wisconsin	55117	Sheboygan	2015	33.1	198.0
3123	Sheboygan	0.242857	55	Wisconsin	55117	Sheboygan	2016	28.1	203.0
3124	Sheboygan	0.242857	55	Wisconsin	55117	Sheboygan	2017	26.3	245.0
3125	Sheboygan	0.242857	55	Wisconsin	55117	Sheboygan	2018	28.4	239.0
3126	Sheboygan	0.242857	55	Wisconsin	55117	Sheboygan	2019	27.8	206.0

1595 rows × 25 columns

In [ ]:
```
# Assuming `aqi_erA_adj` is the DataFrame containing ER visits data
# Assuming the column name for ER visits is 'ER visits for Asthma (rate per 10,
# Assuming the column name for year is 'Year'

plt.figure(figsize=(10, 6))  # Set the figure size

# Group the ER visits data by year and create a list of values for each year
data = [group['ER visits for Asthma (rate per 10,000)'].values for year, group

# Create the boxplot
plt.boxplot(data, labels=aqi_erA_adj['Year'].unique())

# Set the title and labels
plt.title('ER Visits in Group A by Year')
plt.xlabel('Year')
plt.ylabel('ER Visits (rate per 10,000)')

# Show the plot
plt.show()
```



I'ts messing things up here, making the era_adj a log

In [ ]:
```
A = aqi_erA_adj['ER visits for Asthma (rate per 10,000)']

aqi_erA_log = aqi_erA_adj.copy()
aqi_erA_log['ER visits for Asthma (rate per 10,000)'] = np.log(A)
```

In [ ]:
```
aqi_erA_adj
```

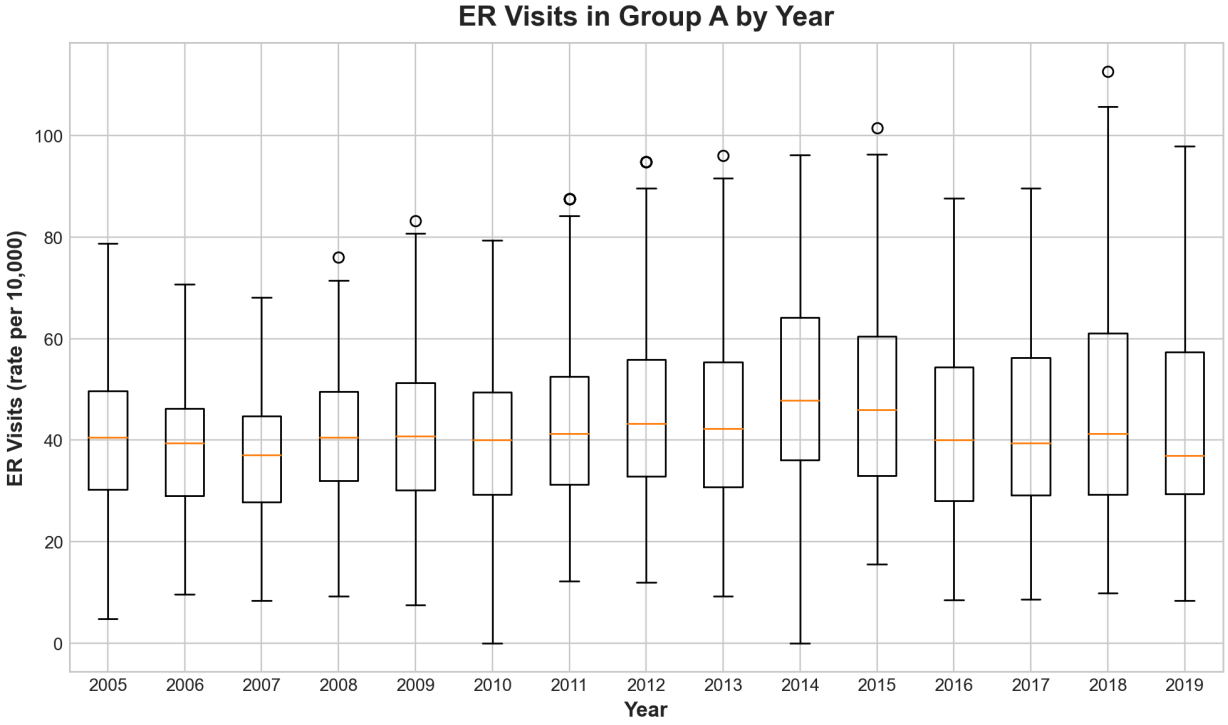
Out[ ]:

	cname	slope	StateFIPS	State	CountyFIPS	County	Year	ER visits for Asthma (rate per 10,000)	Days with AQ
0	Apache	0.130357	4	Arizona	4001	Apache	2005	31.0	336.0
1	Apache	0.130357	4	Arizona	4001	Apache	2006	21.1	339.0
2	Apache	0.130357	4	Arizona	4001	Apache	2007	21.1	365.0
3	Apache	0.130357	4	Arizona	4001	Apache	2008	20.2	295.0
4	Apache	0.130357	4	Arizona	4001	Apache	2009	44.0	364.0
...	...	...	...	...	...	...	...	...	...
3122	Sheboygan	0.242857	55	Wisconsin	55117	Sheboygan	2015	33.1	198.0
3123	Sheboygan	0.242857	55	Wisconsin	55117	Sheboygan	2016	28.1	203.0
3124	Sheboygan	0.242857	55	Wisconsin	55117	Sheboygan	2017	26.3	245.0
3125	Sheboygan	0.242857	55	Wisconsin	55117	Sheboygan	2018	28.4	239.0
3126	Sheboygan	0.242857	55	Wisconsin	55117	Sheboygan	2019	27.8	206.0

1595 rows × 25 columns

In [ ]:
```
# Assuming your dataset is stored in a variable called 'A'
years = aqi_erA['Year'].unique()  # Assuming the column name for year is 'Year
dfA_new = pd.DataFrame()
# Loop through each year and create a variable for it
for year in years:
    # Filter the dataset for the current year
    df_year = aqi_erA[aqi_erA['Year'] == year]
    #print(df_year)
    # Calculate the quartiles
    min_val = df_year['ER visits for Asthma (rate per 10,000)'].min()
    q1, q3 = df_year['ER visits for Asthma (rate per 10,000)'].quantile([0.25,
    median = df_year['ER visits for Asthma (rate per 10,000)'].median()
    max_val = df_year['ER visits for Asthma (rate per 10,000)'].max()
    print(f"Year: {year}, Minimum: {min_val:.2f}, Q1: {q1:.2f}, Median: {media

    iqr = q3 - q1
    lower_bound = q1 - 1.5 * iqr
    upper_bound = q3 + 1.5 * iqr

    outliers = df_year[(df_year['ER visits for Asthma (rate per 10,000)'] < lo
    #dfB_new['Hospitalizations (rate per 10,000)'].append(df_year[(df_year['Ho
    dfA_new = dfA_new.append(df_year[(df_year['ER visits for Asthma (rate per

plt.figure(figsize=(10, 6))  # Set the figure size

# Group the ER visits data by year and create a list of values for each year
data = [group['ER visits for Asthma (rate per 10,000)'].values for year, group

# Create the boxplot
plt.boxplot(data, labels=dfA_new['Year'].unique())

# Set the title and labels
plt.title('ER Visits in Group A by Year')
plt.xlabel('Year')
plt.ylabel('ER Visits (rate per 10,000)')

# Show the plot
plt.show()
```

```
Year: 2005, Minimum: 4.80, Q1: 30.30, Median: 40.60, Q3: 49.70, Maximum: 78.80
Year: 2006, Minimum: 9.60, Q1: 29.10, Median: 39.40, Q3: 46.20, Maximum: 81.00
Year: 2007, Minimum: 8.40, Q1: 29.15, Median: 37.60, Q3: 45.50, Maximum: 76.80
Year: 2008, Minimum: 9.20, Q1: 32.10, Median: 40.70, Q3: 50.10, Maximum: 83.10
Year: 2009, Minimum: 7.50, Q1: 30.40, Median: 41.30, Q3: 51.60, Maximum: 92.00
Year: 2010, Minimum: 0.00, Q1: 30.05, Median: 41.30, Q3: 50.20, Maximum: 128.4
0
Year: 2011, Minimum: 12.20, Q1: 31.30, Median: 41.30, Q3: 53.90, Maximum: 128.
00
Year: 2012, Minimum: 11.90, Q1: 33.18, Median: 43.20, Q3: 58.35, Maximum: 143.
30
Year: 2013, Minimum: 9.20, Q1: 31.20, Median: 42.30, Q3: 57.40, Maximum: 148.2
0
Year: 2014, Minimum: 0.00, Q1: 36.10, Median: 47.80, Q3: 64.60, Maximum: 139.4
0
Year: 2015, Minimum: 15.60, Q1: 33.10, Median: 48.20, Q3: 62.40, Maximum: 147.
30
Year: 2016, Minimum: 8.50, Q1: 29.20, Median: 41.10, Q3: 55.70, Maximum: 137.5
0
Year: 2017, Minimum: 8.60, Q1: 29.10, Median: 40.10, Q3: 56.40, Maximum: 130.1
0
Year: 2018, Minimum: 9.90, Q1: 29.60, Median: 42.00, Q3: 63.80, Maximum: 151.4
0
Year: 2019, Minimum: 8.40, Q1: 29.73, Median: 37.90, Q3: 59.70, Maximum: 127.7
0
```



Random Forest

In [ ]:
```
#aqi_erA_adj = aqi_erA_adj.dropna()
aqi_erA_adj = aqi_erA_adj.replace([np.inf, -np.inf], np.nan).dropna()

#feature_cols = ['Days with AQI', 'Moderate Days', 'Median AQI', 'Days CO', 'Da
feature_cols = ['Days with AQI', 'Good Days',
       'Moderate Days', 'Unhealthy for Sensitive Groups Days',
       'Unhealthy Days', 'Very Unhealthy Days', 'Hazardous Days', 'Max AQI',
       '90th Percentile AQI', 'Median AQI', 'Days CO', 'Days NO2',
```

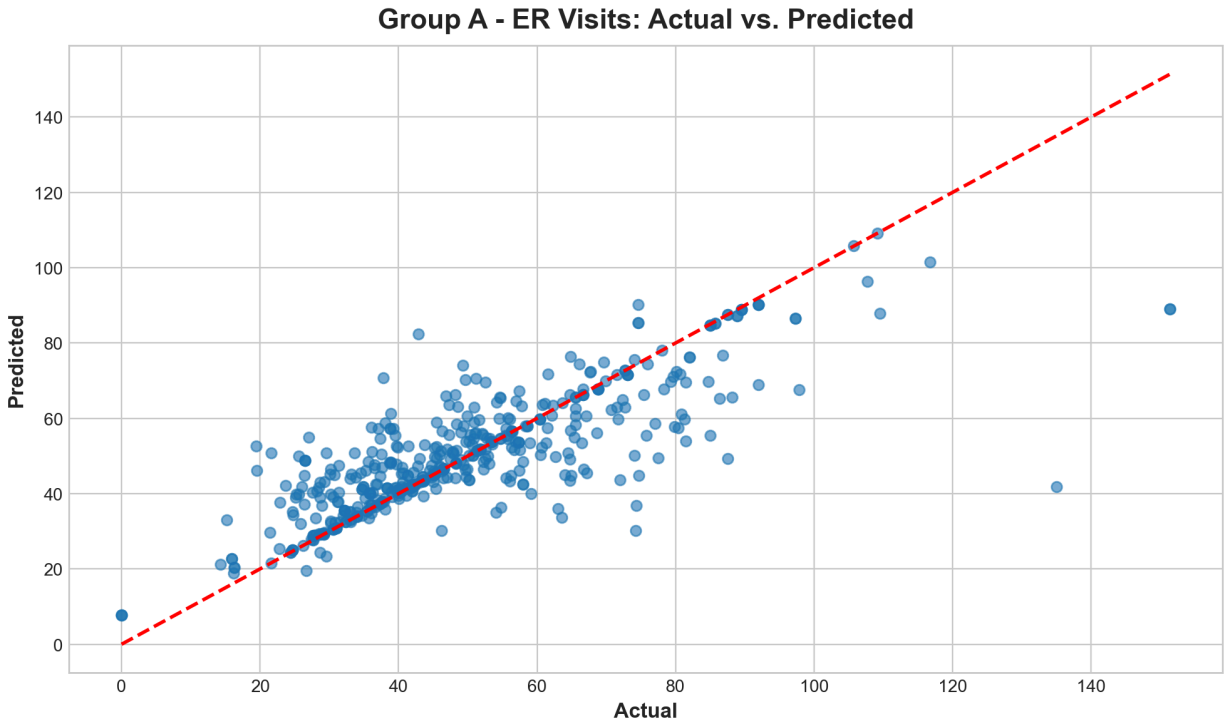
```
       'Days Ozone', 'Days PM2.5', 'Days PM10']

X_er = aqi_erA_adj[feature_cols]
y_er = aqi_erA_adj['ER visits for Asthma (rate per 10,000)']

#X_er = dfA_new[feature_cols]
#y_er = dfA_new['ER visits for Asthma (rate per 10,000)']

X_train_er, X_test_er, y_train_er, y_test_er = train_test_split(X_er, y_er, te
```

before outlier removal

In [ ]:

```
model_er = RandomForestRegressor(n_estimators = 100, random_state = 42)

model_er.fit(X_train_er, y_train_er)
predictions_er = model_er.predict(X_test_er)

mse_er = mean_squared_error(y_test_er, predictions_er)
r2_er = r2_score(y_test_er, predictions_er)

print('ER Visits')
print(f'MSE: {mse_er}')
print(f'R^2 Score: {r2_er}')
rmse = float(format(np.sqrt(mean_squared_error(y_test_er, predictions_er)),'.3
print("\nRMSE:\n",rmse)

# Function to plot actual vs. predicted values
def plot_actual_vs_predicted(y_actual, y_predicted, title):
    plt.figure(figsize=(10, 6))
    plt.scatter(y_actual, y_predicted, alpha=0.6)
    plt.plot([y_actual.min(), y_actual.max()], [y_actual.min(), y_actual.max()
    plt.xlabel('Actual')
    plt.ylabel('Predicted')
    plt.title(title)
    plt.show()

plot_actual_vs_predicted(y_test_er, predictions_er, 'Group A - ER Visits: Actua

# Feature Importance
feature_importances_er = model_er.feature_importances_
features_er = X_er.columns
feature_importances_df_er = pd.DataFrame({'feature': features_er, 'importance'
feature_importances_df_er = feature_importances_df_er.sort_values('importance'
feature_importances_df_er
```

```
ER Visits
MSE: 142.3785781837161
R^2 Score: 0.6584265559939564

RMSE:
 11.932
```



Out[ ]:

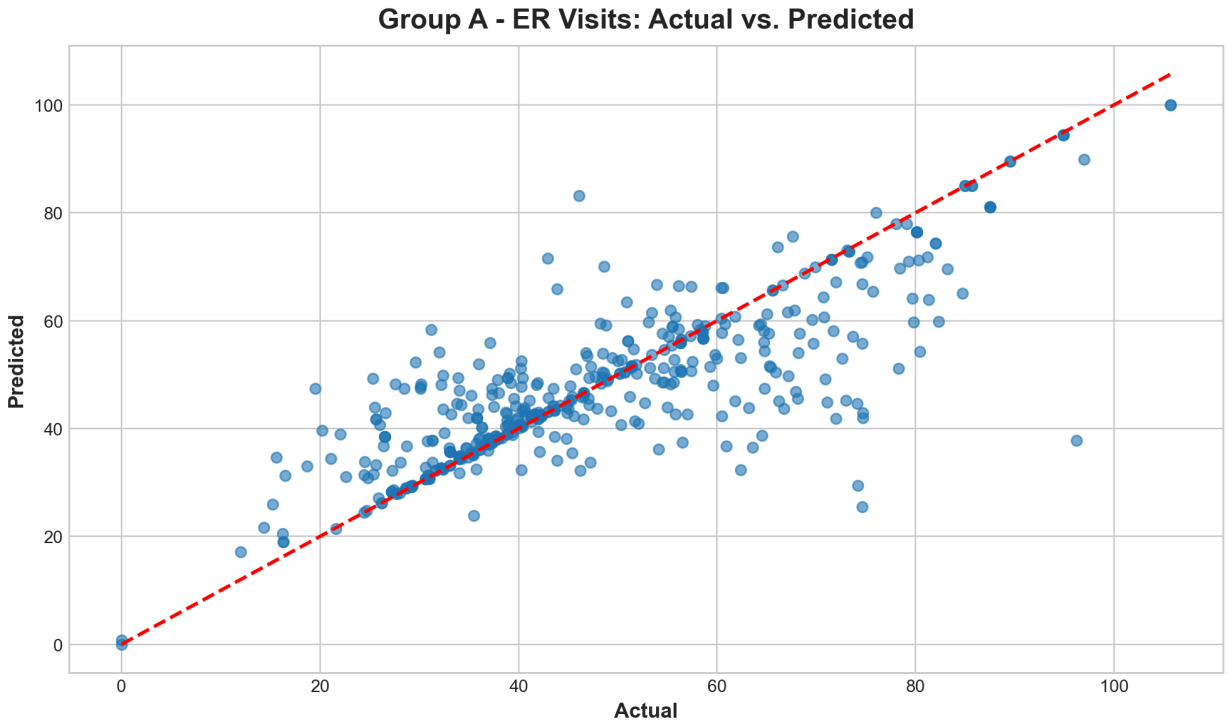
	feature	importance
13	Days PM2.5	0.172943
12	Days Ozone	0.129254
7	Max AQI	0.119284
1	Good Days	0.097031
2	Moderate Days	0.081829
11	Days NO2	0.073401
9	Median AQI	0.071211
8	90th Percentile AQI	0.069783
0	Days with AQI	0.063961
14	Days PM10	0.052390
3	Unhealthy for Sensitive Groups Days	0.033827
10	Days CO	0.021423
4	Unhealthy Days	0.010328
5	Very Unhealthy Days	0.002455
6	Hazardous Days	0.000879

with removed outliers

In [ ]:
```
feature_cols = ['Days with AQI', 'Good Days',
       'Moderate Days', 'Unhealthy for Sensitive Groups Days',
       'Unhealthy Days', 'Very Unhealthy Days', 'Hazardous Days', 'Max AQI',
       '90th Percentile AQI', 'Median AQI', 'Days CO', 'Days NO2',
       'Days Ozone', 'Days PM2.5', 'Days PM10']
dfA_new = dfA_new.replace([np.inf, -np.inf], np.nan).dropna()
```

```
X_er = dfA_new[feature_cols]
y_er = dfA_new['ER visits for Asthma (rate per 10,000)']


X_train_er, X_test_er, y_train_er, y_test_er = train_test_split(X_er, y_er, te

model_er = RandomForestRegressor(n_estimators = 100, random_state = 42, oob_sc

model_er.fit(X_train_er, y_train_er)
predictions_er = model_er.predict(X_test_er)

mse_er = mean_squared_error(y_test_er, predictions_er)
oob_score = model_er.oob_score_
r2_er = r2_score(y_test_er, predictions_er)

print('ER Visits')
print(f'MSE: {mse_er}')
print(f'R^2 Score: {r2_er}')

rmse = float(format(np.sqrt(mean_squared_error(y_test_er, predictions_er)),'.3
print("\nRMSE:\n",rmse)
#print(f'Out-of-Bag Score: {oob_score}')

# Function to plot actual vs. predicted values
def plot_actual_vs_predicted(y_actual, y_predicted, title):
    plt.figure(figsize=(10, 6))
    plt.scatter(y_actual, y_predicted, alpha=0.6)
    plt.plot([y_actual.min(), y_actual.max()], [y_actual.min(), y_actual.max()
    plt.xlabel('Actual')
    plt.ylabel('Predicted')
    plt.title(title)
    plt.show()

plot_actual_vs_predicted(y_test_er, predictions_er, 'Group A - ER Visits: Actu

# Feature Importance
feature_importances_er = model_er.feature_importances_
features_er = X_er.columns
feature_importances_df_er = pd.DataFrame({'feature': features_er, 'importance'
feature_importances_df_er = feature_importances_df_er.sort_values('importance'
feature_importances_df_er
```

```
ER Visits
MSE: 105.84496571086956
R^2 Score: 0.6739829708346019

RMSE:
 10.288
```



Out[ ]:

	feature	importance
11	Days NO2	0.166390
13	Days PM2.5	0.151932
12	Days Ozone	0.110927
7	Max AQI	0.099308
0	Days with AQI	0.085112
1	Good Days	0.078881
2	Moderate Days	0.072519
9	Median AQI	0.068260
8	90th Percentile AQI	0.054778
14	Days PM10	0.046236
3	Unhealthy for Sensitive Groups Days	0.041746
4	Unhealthy Days	0.011803
10	Days CO	0.007906
5	Very Unhealthy Days	0.002571
6	Hazardous Days	0.001631

Multiple Linear Regression

OLS

In [ ]:
```
# Create a DataFrame with all the independent variables
X = dfA_new[feature_cols]
```

```
# Add a constant to the DataFrame
X = sm.add_constant(X)

# Create a Series with the dependent variable
y = dfA_new['ER visits for Asthma (rate per 10,000)']

# Create a model
model = sm.OLS(y, X).fit()

# Get the summary of the model
print(model.summary())
print("Parameters: ", model.params)
print("R2: ", model.rsquared)
```

OLS Regression Results

Dep. Variable:	ER visits for Asthma (rate per 10,000)	R-squared:	0.166
Model:	OLS	Adj. R-squared:	0.159
Method:	Least Squares	F-statistic:	23.23
Date:	Mon, 29 Apr 2024	Prob (F-statistic):	2.01e-51
Time:	19:47:03	Log-Likelihood:	-6420.3
No. Observations:	1532	AIC:	1.287e+04
Df Residuals:	1518	BIC:	1.294e+04
Df Model:	13		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	15.2202	4.638	3.282	0.001	6.123	24.317
Days with AQI	-0.0843	0.133	-0.636	0.525	-0.344	0.176
Good Days	0.1529	0.110	1.390	0.165	-0.063	0.369
Moderate Days	0.1378	0.110	1.248	0.212	-0.079	0.354
Unhealthy for Sensitive Groups Days	0.0618	0.113	0.547	0.584	-0.160	0.283
Unhealthy Days	0.1236	0.205	0.604	0.546	-0.278	0.525
Very Unhealthy Days	-1.0141	0.573	-1.770	0.077	-2.138	0.110
Hazardous Days	0.4537	0.466	0.974	0.330	-0.460	1.368
Max AQI	0.0001	0.001	0.115	0.908	-0.002	0.002
90th Percentile AQI	-0.0138	0.048	-0.290	0.772	-0.108	0.080
Median AQI	0.4017	0.102	3.938	0.000	0.202	0.602
Days CO	-0.0851	0.551	-0.154	0.877	-1.166	0.996
Days NO2	0.0696	0.123	0.567	0.571	-0.171	0.310
Days Ozone	-0.0157	0.114	-0.138	0.890	-0.239	0.207
Days PM2.5	0.0099	0.114	0.087	0.931	-0.213	0.233
Days PM10	-0.0629	0.113	-0.555	0.579	-0.285	0.159

Omnibus:	45.094	Durbin-Watson:	2.104
Prob(Omnibus):	0.000	Jarque-Bera (JB):	48.550

```
Skew:                          0.436   Prob(JB):                     2.87e-11
Kurtosis:                      3.041   Cond. No.                     1.51e+16
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correct
ly specified.
[2] The smallest eigenvalue is 3.74e-24. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.
Parameters:  const                                15.220180
Days with AQI                            -0.084256
Good Days                                 0.152924
Moderate Days                             0.137786
Unhealthy for Sensitive Groups Days       0.061787
Unhealthy Days                            0.123626
Very Unhealthy Days                      -1.014105
Hazardous Days                            0.453725
Max AQI                                   0.000106
90th Percentile AQI                      -0.013840
Median AQI                                0.401709
Days CO                                  -0.085071
Days NO2                                  0.069553
Days Ozone                               -0.015716
Days PM2.5                                0.009858
Days PM10                                -0.062879
dtype: float64
R2:  0.16595201391094694
```

SARIMAX

In [ ]:

```
data = dfA_new.copy()
data = data.groupby('Year').mean()
data = data.drop(columns=['slope', 'StateFIPS',  'CountyFIPS',
        'Days with AQI', 'Good Days',
       'Moderate Days', 'Unhealthy for Sensitive Groups Days',
       'Unhealthy Days', 'Very Unhealthy Days', 'Hazardous Days', 'Max AQI',
       '90th Percentile AQI', 'Median AQI', 'Days CO', 'Days NO2',
         'Days Ozone', 'Days PM10', 'FIPS'])
data
```

Out[ ]:

Year	ER visits for Asthma (rate per 10,000)	Days PM2.5
2005	45.090361	66.795181
2006	40.804878	68.317073
2007	40.207595	75.506329
2008	44.183505	90.938144
2009	45.657143	89.329670
2010	42.190476	95.761905
2011	48.233028	105.000000
2012	52.024299	100.411215
2013	49.123636	104.390909
2014	54.247899	103.546218
2015	50.194393	99.252336
2016	46.628571	89.371429
2017	47.629245	100.783019
2018	50.362712	114.076271
2019	47.612281	101.429825

In [ ]:

```
en = data['ER visits for Asthma (rate per 10,000)']
ex = data['Days PM2.5']
order = (1, 1, 1)
seasonal_order = (1, 0, 0, 2)
exog = np.empty([14, 1])


model = sm.tsa.SARIMAX(endog=en, exog=ex, order=order, seasonal_order=seasonal_
results=model.fit()

data['forecast']=results.predict(start=1,end=14,dynamic=False)
data[['ER visits for Asthma (rate per 10,000)','forecast']].plot(figsize=(8,4)

results.summary()
```

```
RUNNING THE L-BFGS-B CODE

           * * *

Machine precision = 2.220D-16
 N =            5     M =           10

At X0         0 variables are exactly at the bounds

At iterate    0    f=  2.35337D+00    |proj g|=  6.50039D-02

At iterate    5    f=  2.34975D+00    |proj g|=  8.26371D-03

At iterate   10    f=  2.33927D+00    |proj g|=  8.15759D-02

At iterate   15    f=  2.32851D+00    |proj g|=  8.49759D-02

At iterate   20    f=  2.30752D+00    |proj g|=  1.31906D-02

At iterate   25    f=  2.30460D+00    |proj g|=  1.10831D-02

At iterate   30    f=  2.30418D+00    |proj g|=  1.10462D-02

At iterate   35    f=  2.30412D+00    |proj g|=  3.20032D-03

At iterate   40    f=  2.30411D+00    |proj g|=  1.14284D-04

           * * *

Tit   = total number of iterations
Tnf   = total number of function evaluations
Tnint = total number of segments explored during Cauchy searches
Skip  = number of BFGS updates skipped
Nact  = number of active bounds at final generalized Cauchy point
Projg = norm of the final projected gradient
F     = final function value

           * * *

   N    Tit     Tnf  Tnint  Skip  Nact     Projg        F
    5     43     58      1     0     0   2.559D-04   2.304D+00
  F =   2.3041119960406653

CONVERGENCE: REL_REDUCTION_OF_F_<=_FACTR*EPSMCH
 This problem is unconstrained.
```

Out[ ]:

SARIMAX Results

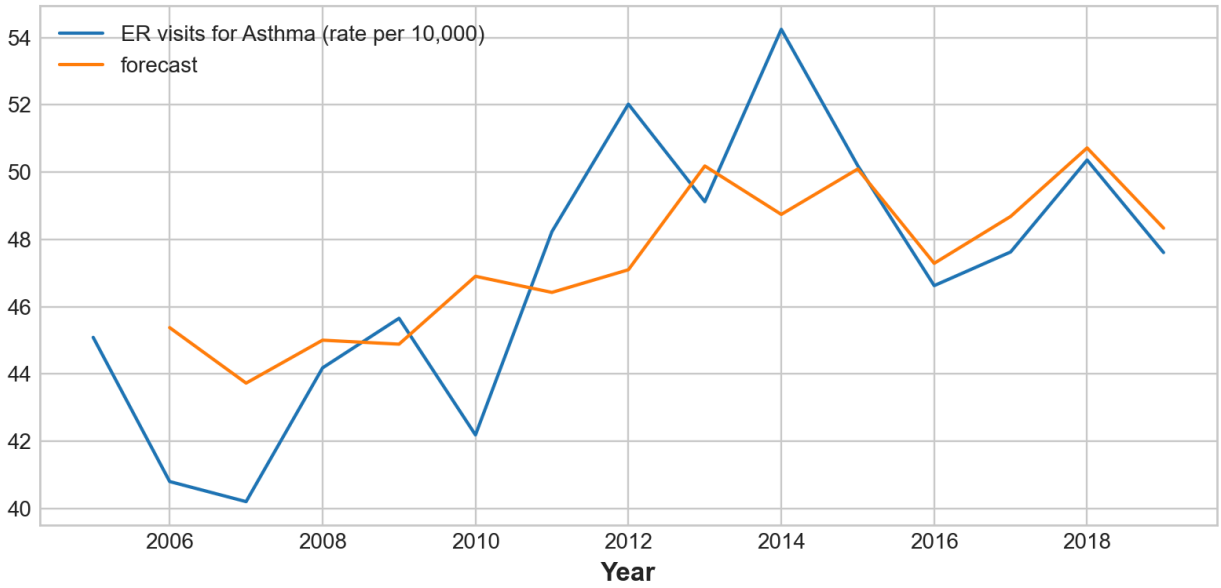
Dep. Variable:	ER visits for Asthma (rate per 10,000)	**No. Observations:**	15
Model:	SARIMAX(1, 1, 1)x(1, 0, [], 2)	**Log Likelihood**	-34.562
Date:	Mon, 29 Apr 2024	**AIC**	79.123
Time:	19:47:03	**BIC**	82.319
Sample:	0	**HQIC**	78.828
	- 15		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
Days PM2.5	0.1907	0.106	1.800	0.072	-0.017	0.398
ar.L1	0.3465	0.672	0.515	0.606	-0.971	1.664
ma.L1	-0.9992	136.555	-0.007	0.994	-268.641	266.643
ar.S.L2	0.0294	0.351	0.084	0.933	-0.659	0.718
sigma2	7.0747	961.165	0.007	0.994	-1876.774	1890.923

Ljung-Box (L1) (Q):	0.01	**Jarque-Bera (JB):**	0.78
Prob(Q):	0.91	**Prob(JB):**	0.68
Heteroskedasticity (H):	0.05	**Skew:**	0.58
Prob(H) (two-sided):	0.00	**Kurtosis:**	2.91

Warnings:

[1] Covariance matrix calculated using the outer product of gradients (complex-step).

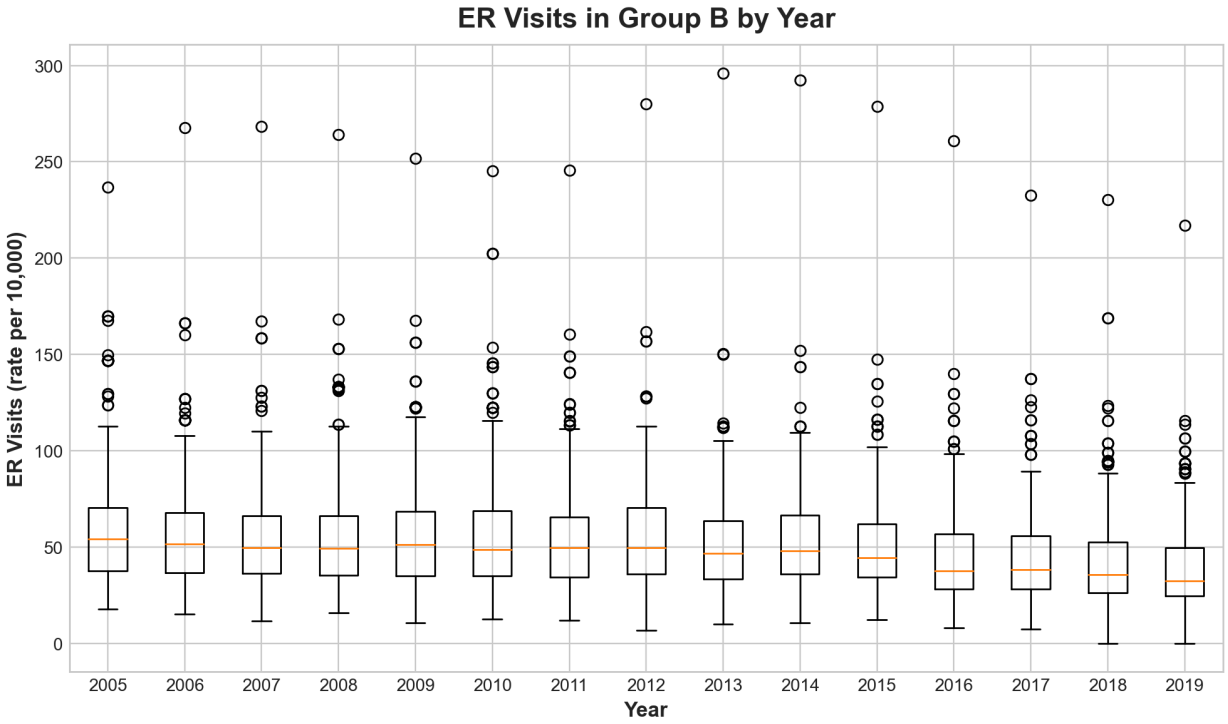


Models for ER Group B (Decreasing)

In [ ]:
```
#aqi_erB.info()
#aqi_erB.isnull().sum()
#print(aqi_erB.tail(10))
aqi_erB_adj = aqi_erB.dropna()
aqi_erB_adj.shape
```

Out[ ]:
```
(8472, 25)
```

In [ ]:
```
plt.figure(figsize=(10, 6))  # Set the figure size

# Group the ER visits data by year and create a list of values for each year
data = [group['ER visits for Asthma (rate per 10,000)'].values for year, group

# Create the boxplot
plt.boxplot(data, labels=aqi_erB_adj['Year'].unique())

# Set the title and labels
plt.title('ER Visits in Group B by Year')
plt.xlabel('Year')
plt.ylabel('ER Visits (rate per 10,000)')

# Show the plot
plt.show()
```

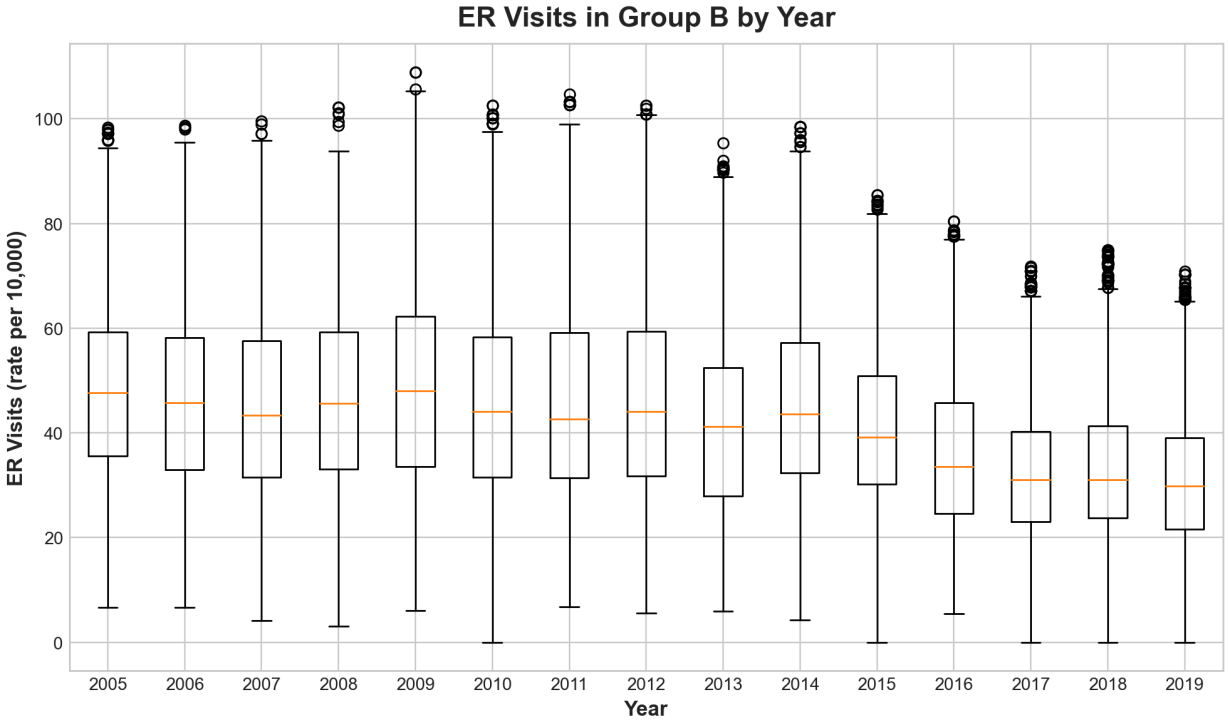


In [ ]:
```
B = aqi_erB_adj['ER visits for Asthma (rate per 10,000)']
aqi_erB_log = aqi_erB_adj.copy()
aqi_erB_log['ER visits for Asthma (rate per 10,000)'] = np.log(B)
```

In [ ]:
```
# Assuming your dataset is stored in a variable called 'A'
years = aqi_erB['Year'].unique()  # Assuming the column name for year is 'Year
dfB_new = pd.DataFrame()
# Loop through each year and create a variable for it
for year in years:
    # Filter the dataset for the current year
    df_year = aqi_erB[aqi_erB['Year'] == year]
```

```
    #print(df_year)
    # Calculate the quartiles
    min_val = df_year['ER visits for Asthma (rate per 10,000)'].min()
    q1, q3 = df_year['ER visits for Asthma (rate per 10,000)'].quantile([0.25,
    median = df_year['ER visits for Asthma (rate per 10,000)'].median()
    max_val = df_year['ER visits for Asthma (rate per 10,000)'].max()
    print(f"Year: {year}, Minimum: {min_val:.2f}, Q1: {q1:.2f}, Median: {media

    iqr = q3 - q1
    lower_bound = q1 - 1.5 * iqr
    upper_bound = q3 + 1.5 * iqr

    outliers = df_year[(df_year['ER visits for Asthma (rate per 10,000)'] < lo
    #dfB_new['Hospitalizations (rate per 10,000)'].append(df_year[(df_year['Ho
    dfB_new = dfB_new.append(df_year[(df_year['ER visits for Asthma (rate per

plt.figure(figsize=(10, 6))  # Set the figure size

# Group the ER visits data by year and create a list of values for each year
data = [group['ER visits for Asthma (rate per 10,000)'].values for year, group

# Create the boxplot
plt.boxplot(data, labels=dfB_new['Year'].unique())

# Set the title and labels
plt.title('ER Visits in Group B by Year')
plt.xlabel('Year')
plt.ylabel('ER Visits (rate per 10,000)')

# Show the plot
plt.show()
```

```
Year: 2005, Minimum: 6.60, Q1: 35.70, Median: 48.10, Q3: 61.15, Maximum: 236.8
0
Year: 2006, Minimum: 6.70, Q1: 33.20, Median: 46.40, Q3: 59.70, Maximum: 267.6
0
Year: 2007, Minimum: 4.10, Q1: 31.70, Median: 44.30, Q3: 59.35, Maximum: 268.4
0
Year: 2008, Minimum: 3.10, Q1: 33.40, Median: 46.50, Q3: 61.42, Maximum: 264.1
0
Year: 2009, Minimum: 6.10, Q1: 33.60, Median: 48.90, Q3: 64.00, Maximum: 251.6
0
Year: 2010, Minimum: 0.00, Q1: 31.70, Median: 45.20, Q3: 60.80, Maximum: 245.1
0
Year: 2011, Minimum: 6.80, Q1: 31.70, Median: 44.20, Q3: 61.10, Maximum: 245.5
0
Year: 2012, Minimum: 5.60, Q1: 32.40, Median: 45.20, Q3: 60.90, Maximum: 280.0
0
Year: 2013, Minimum: 5.90, Q1: 28.48, Median: 43.05, Q3: 55.30, Maximum: 296.0
0
Year: 2014, Minimum: 4.30, Q1: 32.80, Median: 44.20, Q3: 59.25, Maximum: 292.5
0
Year: 2015, Minimum: 0.00, Q1: 30.92, Median: 40.10, Q3: 53.20, Maximum: 278.7
0
Year: 2016, Minimum: 5.50, Q1: 25.00, Median: 34.10, Q3: 47.20, Maximum: 260.9
0
Year: 2017, Minimum: 0.00, Q1: 24.23, Median: 32.50, Q3: 43.40, Maximum: 232.5
0
Year: 2018, Minimum: 0.00, Q1: 24.10, Median: 31.90, Q3: 44.50, Maximum: 230.2
0
Year: 2019, Minimum: 0.00, Q1: 22.40, Median: 30.90, Q3: 41.80, Maximum: 216.9
0
```



Random Forest

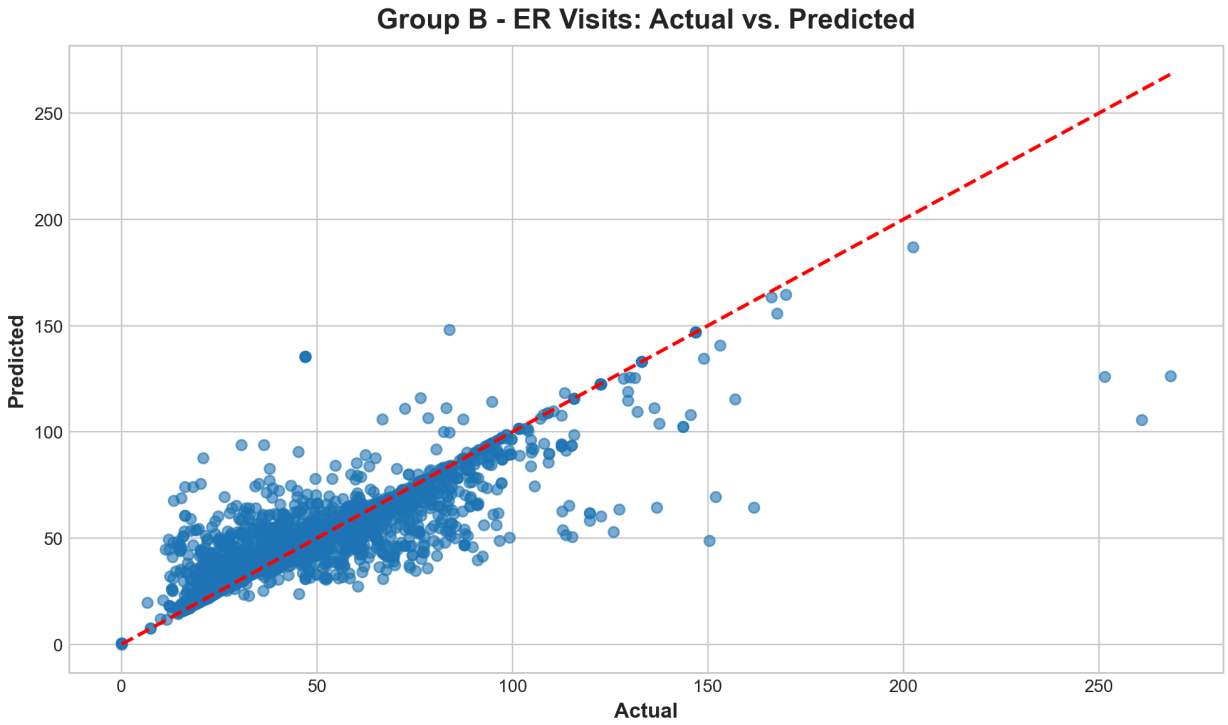
In [ ]:
```
#aqi_erA_adj = aqi_erA_adj.dropna()
aqi_erB_adj = aqi_erB_adj.replace([np.inf, -np.inf], np.nan).dropna()
```

```
#feature_cols = ['Days with AQI', 'Moderate Days', 'Median AQI', 'Days CO', 'Da
feature_cols = ['Days with AQI', 'Good Days',
       'Moderate Days', 'Unhealthy for Sensitive Groups Days',
       'Unhealthy Days', 'Very Unhealthy Days', 'Hazardous Days', 'Max AQI',
       '90th Percentile AQI', 'Median AQI', 'Days CO', 'Days NO2',
       'Days Ozone', 'Days PM2.5', 'Days PM10']

X_er = aqi_erB_adj[feature_cols]
y_er = aqi_erB_adj['ER visits for Asthma (rate per 10,000)']

#X_er = dfA_new[feature_cols]
#y_er = dfA_new['ER visits for Asthma (rate per 10,000)']

X_train_er, X_test_er, y_train_er, y_test_er = train_test_split(X_er, y_er, tes
```

In [ ]:

```
model_er = RandomForestRegressor(n_estimators = 100, random_state = 42)

model_er.fit(X_train_er, y_train_er)
predictions_er = model_er.predict(X_test_er)

mse_er = mean_squared_error(y_test_er, predictions_er)
r2_er = r2_score(y_test_er, predictions_er)

print('ER Visits')
print(f'MSE: {mse_er}')
print(f'R^2 Score: {r2_er}')
rmse = float(format(np.sqrt(mean_squared_error(y_test_er, predictions_er)),'.3f
print("\nRMSE:\n",rmse)

# Function to plot actual vs. predicted values
def plot_actual_vs_predicted(y_actual, y_predicted, title):
    plt.figure(figsize=(10, 6))
    plt.scatter(y_actual, y_predicted, alpha=0.6)
    plt.plot([y_actual.min(), y_actual.max()], [y_actual.min(), y_actual.max()]
    plt.xlabel('Actual')
    plt.ylabel('Predicted')
    plt.title(title)
    plt.show()

plot_actual_vs_predicted(y_test_er, predictions_er, 'Group B - ER Visits: Actua

# Feature Importance
feature_importances_er = model_er.feature_importances_
features_er = X_er.columns
feature_importances_df_er = pd.DataFrame({'feature': features_er, 'importance'
feature_importances_df_er = feature_importances_df_er.sort_values('importance'
feature_importances_df_er
```

```
ER Visits
MSE: 206.02930996205785
R^2 Score: 0.6801519454379699

RMSE:
 14.354
```



Out[ ]:

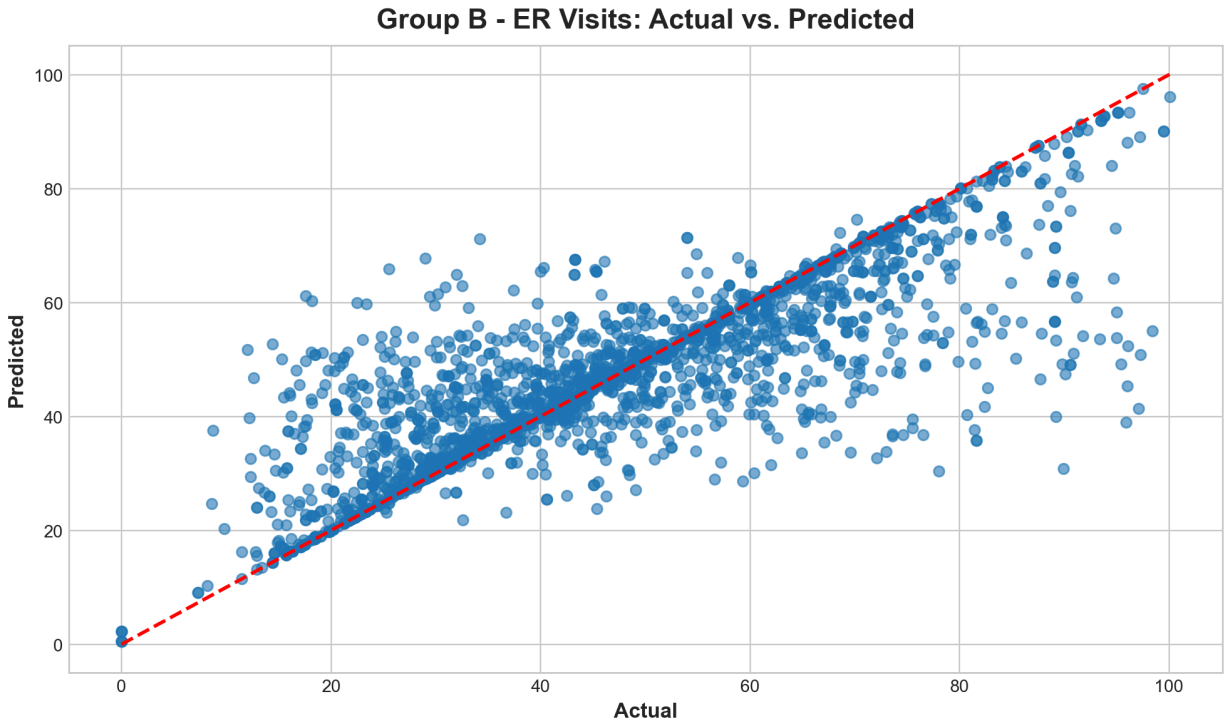
	feature	importance
12	Days Ozone	0.141913
13	Days PM2.5	0.125446
2	Moderate Days	0.121525
11	Days NO2	0.101096
9	Median AQI	0.090731
7	Max AQI	0.090128
1	Good Days	0.079757
0	Days with AQI	0.069521
8	90th Percentile AQI	0.066851
3	Unhealthy for Sensitive Groups Days	0.045779
14	Days PM10	0.045302
10	Days CO	0.010377
4	Unhealthy Days	0.009598
5	Very Unhealthy Days	0.001953
6	Hazardous Days	0.000021

with removed outliers

In [ ]:
```
dfB_new = dfB_new.replace([np.inf, -np.inf], np.nan).dropna()
X_er = dfB_new[feature_cols]
y_er = dfB_new['ER visits for Asthma (rate per 10,000)']

X_train_er, X_test_er, y_train_er, y_test_er = train_test_split(X_er, y_er, te
```

```
model_er = RandomForestRegressor(n_estimators = 100, random_state = 42)

model_er.fit(X_train_er, y_train_er)
predictions_er = model_er.predict(X_test_er)

mse_er = mean_squared_error(y_test_er, predictions_er)
r2_er = r2_score(y_test_er, predictions_er)

print('ER Visits')
print(f'MSE: {mse_er}')
print(f'R^2 Score: {r2_er}')
rmse = float(format(np.sqrt(mean_squared_error(y_test_er, predictions_er)),'.3
print("\nRMSE:\n",rmse)

# Function to plot actual vs. predicted values
def plot_actual_vs_predicted(y_actual, y_predicted, title):
    plt.figure(figsize=(10, 6))
    plt.scatter(y_actual, y_predicted, alpha=0.6)
    plt.plot([y_actual.min(), y_actual.max()], [y_actual.min(), y_actual.max()]
    plt.xlabel('Actual')
    plt.ylabel('Predicted')
    plt.title(title)
    plt.show()

plot_actual_vs_predicted(y_test_er, predictions_er, 'Group B - ER Visits: Actua

# Feature Importance
feature_importances_er = model_er.feature_importances_
features_er = X_er.columns
feature_importances_df_er = pd.DataFrame({'feature': features_er, 'importance'
feature_importances_df_er = feature_importances_df_er.sort_values('importance'
feature_importances_df_er
```

```
ER Visits
MSE: 119.76554724974014
R^2 Score: 0.6771196916764299

RMSE:
 10.944
```



Out[ ]:

	feature	importance
7	Max AQI	0.114436
13	Days PM2.5	0.114346
12	Days Ozone	0.112581
2	Moderate Days	0.111708
1	Good Days	0.110611
9	Median AQI	0.106154
0	Days with AQI	0.090137
8	90th Percentile AQI	0.080262
14	Days PM10	0.053151
11	Days NO2	0.046032
3	Unhealthy for Sensitive Groups Days	0.041389
4	Unhealthy Days	0.010956
10	Days CO	0.007151
5	Very Unhealthy Days	0.001067
6	Hazardous Days	0.000018

Multiple Linear Regression

OLS

In [ ]:
```
# Create a DataFrame with all the independent variables
X = dfB_new[feature_cols]
```

```
# Add a constant to the DataFrame
X = sm.add_constant(X)

# Create a Series with the dependent variable
y = dfB_new['ER visits for Asthma (rate per 10,000)']

# Create a model
model = sm.OLS(y, X).fit()

# Get the summary of the model
print(model.summary())
print("Parameters: ", model.params)
print("R2: ", model.rsquared)
```

OLS Regression Results

Dep. Variable:	ER visits for Asthma (rate per 10,000)	R-squared:	0.076
Model:	OLS	Adj. R-squared:	0.075
Method:	Least Squares	F-statistic:	46.77
Date:	Mon, 29 Apr 2024	Prob (F-statistic):	2.60e-125
Time:	19:47:07	Log-Likelihood:	-34603.
No. Observations:	7953	AIC:	6.924e+04
Df Residuals:	7938	BIC:	6.934e+04
Df Model:	14		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	39.7281	1.973	20.133	0.000	35.860	43.596
Days with AQI	-6.103e+10	6.34e+10	-0.962	0.336	-1.85e+11	6.33e+10
Good Days	2.729e+10	2.84e+10	0.962	0.336	-2.83e+10	8.29e+10
Moderate Days	2.729e+10	2.84e+10	0.962	0.336	-2.83e+10	8.29e+10
Unhealthy for Sensitive Groups Days	2.729e+10	2.84e+10	0.962	0.336	-2.83e+10	8.29e+10
Unhealthy Days	2.729e+10	2.84e+10	0.962	0.336	-2.83e+10	8.29e+10
Very Unhealthy Days	2.729e+10	2.84e+10	0.962	0.336	-2.83e+10	8.29e+10
Hazardous Days	2.729e+10	2.84e+10	0.962	0.336	-2.83e+10	8.29e+10
Max AQI	0.0938	0.009	10.823	0.000	0.077	0.111
90th Percentile AQI	-0.2481	0.033	-7.523	0.000	-0.313	-0.183
Median AQI	0.1529	0.055	2.778	0.005	0.045	0.261
Days CO	3.374e+10	3.51e+10	0.962	0.336	-3.5e+10	1.02e+11
Days NO2	3.374e+10	3.51e+10	0.962	0.336	-3.5e+10	1.02e+11
Days Ozone	3.374e+10	3.51e+10	0.962	0.336	-3.5e+10	1.02e+11
Days PM2.5	3.374e+10	3.51e+10	0.962	0.336	-3.5e+10	1.02e+11
Days PM10	3.374e+10	3.51e+10	0.962	0.336	-3.5e+10	1.02e+11

Omnibus:	259.831	Durbin-Watson:	1.569
Prob(Omnibus):	0.000	Jarque-Bera (JB):	266.799

```
Skew:                          0.423   Prob(JB):                     1.16e-58
Kurtosis:                      2.701   Cond. No.                     4.56e+15
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correct
ly specified.
[2] The smallest eigenvalue is 8.97e-23. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.
Parameters:  const                                3.972814e+01
Days with AQI                          -6.103301e+10
Good Days                               2.729190e+10
Moderate Days                           2.729190e+10
Unhealthy for Sensitive Groups Days     2.729190e+10
Unhealthy Days                          2.729190e+10
Very Unhealthy Days                     2.729190e+10
Hazardous Days                          2.729190e+10
Max AQI                                 9.378352e-02
90th Percentile AQI                    -2.481001e-01
Median AQI                              1.528565e-01
Days CO                                 3.374111e+10
Days NO2                                3.374111e+10
Days Ozone                              3.374111e+10
Days PM2.5                              3.374111e+10
Days PM10                               3.374111e+10
dtype: float64
R2:  0.07620028058628836
```

SARIMAX

In [ ]:

```
data = dfB_new.copy()
data = data.groupby('Year').mean()
data = data.drop(columns=['slope', 'StateFIPS',  'CountyFIPS',
        'Days with AQI', 'Good Days',
       'Moderate Days', 'Unhealthy for Sensitive Groups Days',
       'Unhealthy Days', 'Very Unhealthy Days', 'Hazardous Days', 'Max AQI',
       '90th Percentile AQI', 'Median AQI', 'Days CO', 'Days NO2',
         'Days Ozone', 'Days PM10', 'FIPS'])

en = data['ER visits for Asthma (rate per 10,000)']
ex = data['Days PM2.5']
order = (1, 1, 1)
seasonal_order = (1, 0, 0, 2)
exog = np.empty([14, 1])


model = sm.tsa.SARIMAX(endog=en, exog=ex, order=order, seasonal_order=seasonal_
results=model.fit()

data['forecast']=results.predict(start=1,end=14,dynamic=False)
data[['ER visits for Asthma (rate per 10,000)','forecast']].plot(figsize=(8,4)

results.summary()
```

```
RUNNING THE L-BFGS-B CODE

           * * *

Machine precision = 2.220D-16
 N =            5     M =           10

At X0         0 variables are exactly at the bounds

At iterate    0    f=  2.22335D+00    |proj g|=  2.25813D-01

At iterate    5    f=  2.21780D+00    |proj g|=  5.61931D-03

At iterate   10    f=  2.21779D+00    |proj g|=  3.54685D-04

At iterate   15    f=  2.21779D+00    |proj g|=  1.84675D-03

           * * *

Tit   = total number of iterations
Tnf   = total number of function evaluations
Tnint = total number of segments explored during Cauchy searches
Skip  = number of BFGS updates skipped
Nact  = number of active bounds at final generalized Cauchy point
Projg = norm of the final projected gradient
F     = final function value

           * * *

   N    Tit     Tnf  Tnint  Skip  Nact     Projg        F
    5     18     22      1     0     0   1.246D-05   2.218D+00
  F =   2.2177857579287248

CONVERGENCE: REL_REDUCTION_OF_F_<=_FACTR*EPSMCH
 This problem is unconstrained.
```

Out[ ]:

SARIMAX Results

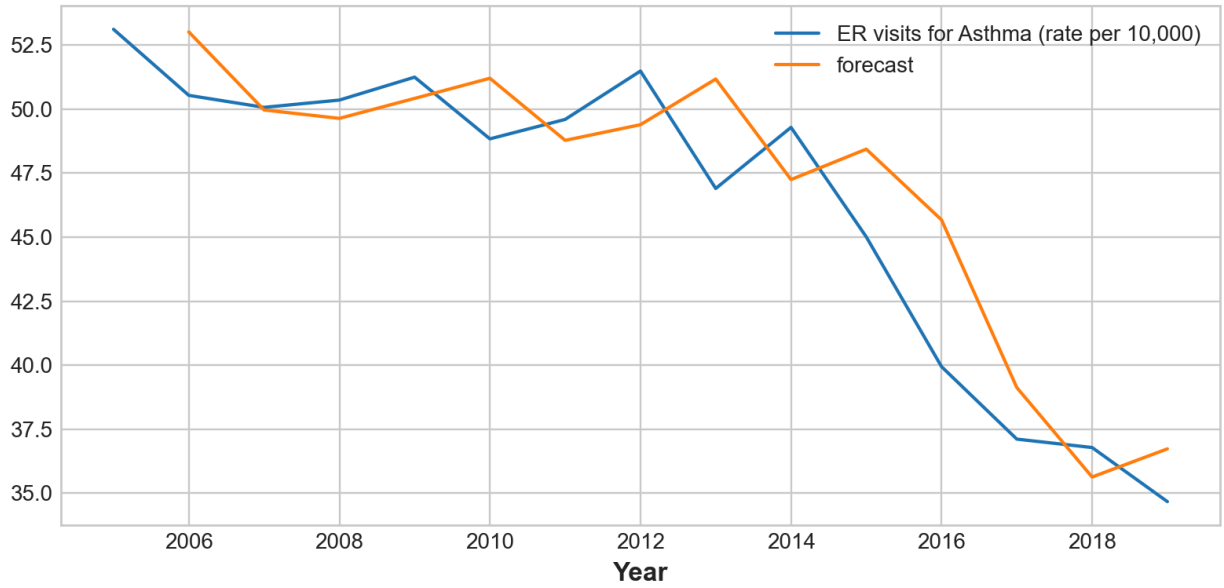
Dep. Variable:	ER visits for Asthma (rate per 10,000)	**No. Observations:**	15
Model:	SARIMAX(1, 1, 1)x(1, 0, [], 2)	**Log Likelihood**	-33.267
Date:	Mon, 29 Apr 2024	**AIC**	76.534
Time:	19:47:08	**BIC**	79.729
Sample:	0	**HQIC**	76.238
	- 15		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
Days PM2.5	-0.0326	0.101	-0.323	0.747	-0.230	0.165
ar.L1	-0.0005	127.269	-4.09e-06	1.000	-249.444	249.443
ma.L1	-0.0243	127.403	-0.000	1.000	-249.730	249.681
ar.S.L2	0.1766	3.032	0.058	0.954	-5.765	6.119
sigma2	6.7528	4.714	1.433	0.152	-2.486	15.991

Ljung-Box (L1) (Q):	0.36	**Jarque-Bera (JB):**	0.93
Prob(Q):	0.55	**Prob(JB):**	0.63
Heteroskedasticity (H):	4.25	**Skew:**	-0.36
Prob(H) (two-sided):	0.14	**Kurtosis:**	1.96

Warnings:

[1] Covariance matrix calculated using the outer product of gradients (complex-step).

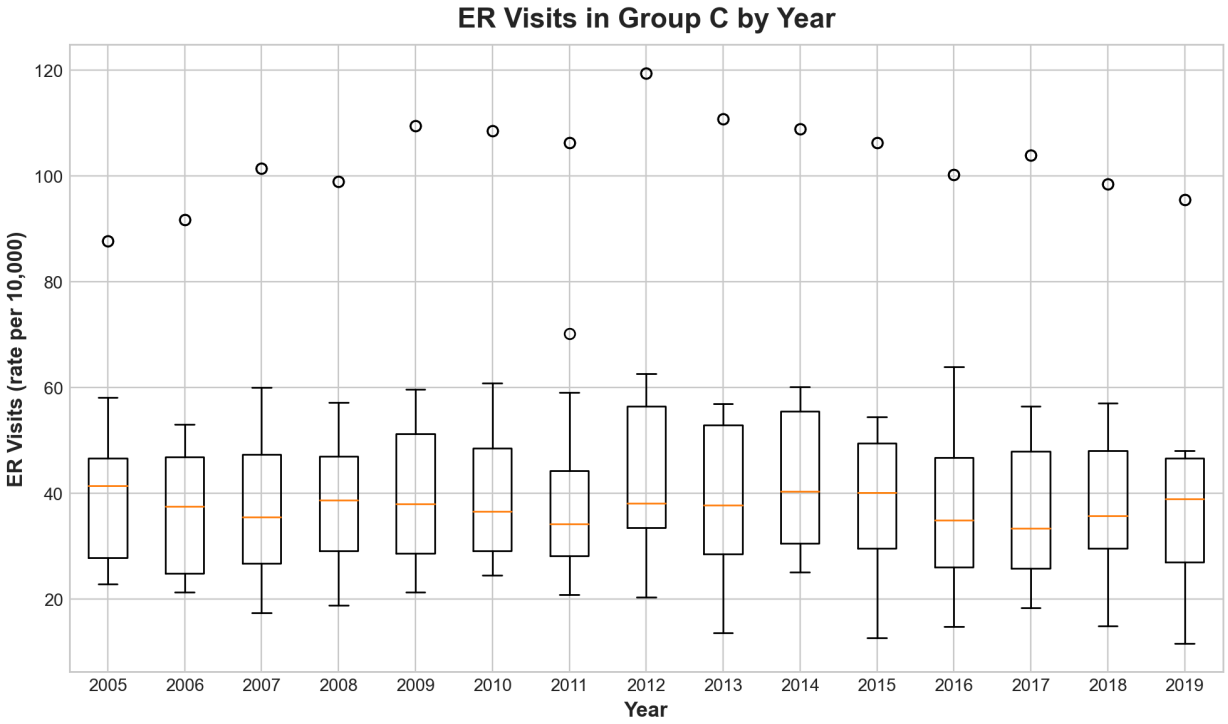


Models for ER Group C (Neutral)

In [ ]:
```
#aqi_erC.info()
#aqi_erC.isnull().sum()
#aqi_erC.tail(10)
aqi_erC_adj = aqi_erC.dropna()
aqi_erC_adj.shape
```

Out[ ]:
```
(329, 25)
```

In [ ]:
```
plt.figure(figsize=(10, 6))  # Set the figure size

# Group the ER visits data by year and create a list of values for each year
data = [group['ER visits for Asthma (rate per 10,000)'].values for year, group

# Create the boxplot
plt.boxplot(data, labels=aqi_erC_adj['Year'].unique())

# Set the title and labels
plt.title('ER Visits in Group C by Year')
plt.xlabel('Year')
plt.ylabel('ER Visits (rate per 10,000)')

# Show the plot
plt.show()
```

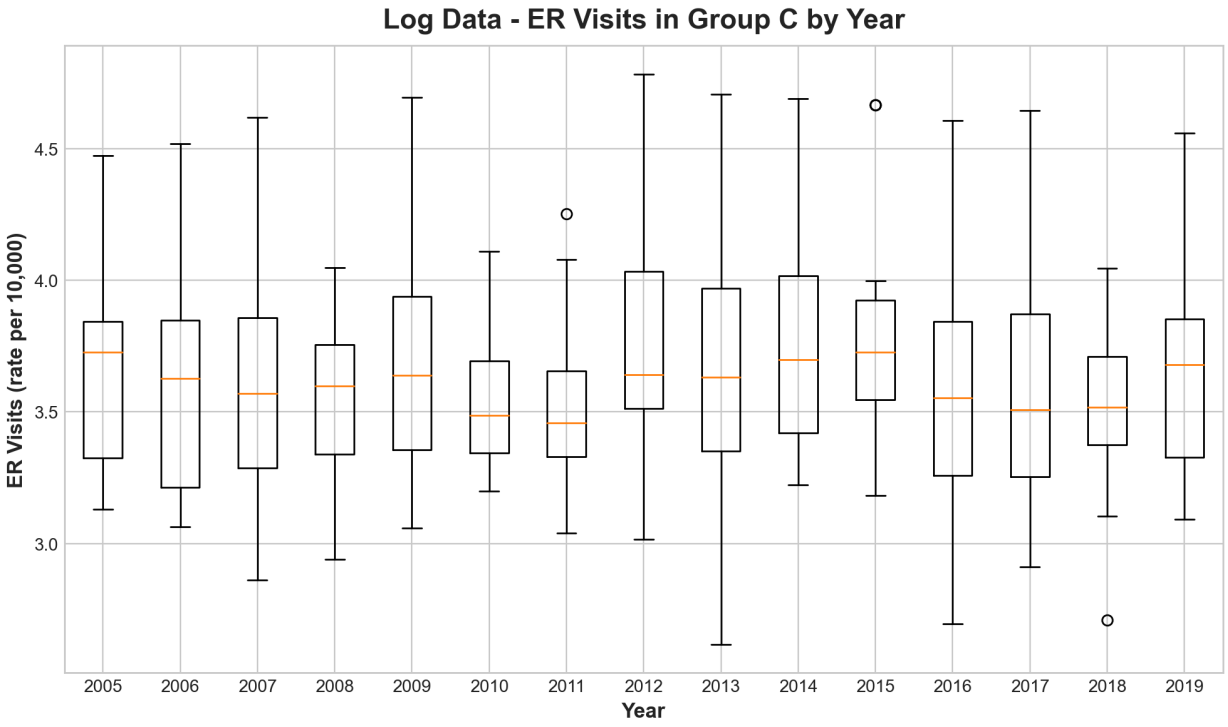


In [ ]:
```
C = aqi_erC_adj['ER visits for Asthma (rate per 10,000)']
aqi_erC_log = aqi_erC_adj.copy()
aqi_erC_log['ER visits for Asthma (rate per 10,000)'] = np.log(C)
```

In [ ]:
```
# Assuming your dataset is stored in a variable called 'A'
years = aqi_erC_log['Year'].unique()  # Assuming the column name for year is '
dfC_new = pd.DataFrame()
# Loop through each year and create a variable for it
for year in years:
    # Filter the dataset for the current year
    df_year = aqi_erC_log[aqi_erC_log['Year'] == year]
```

```
    #print(df_year)
    # Calculate the quartiles
    min_val = df_year['ER visits for Asthma (rate per 10,000)'].min()
    q1, q3 = df_year['ER visits for Asthma (rate per 10,000)'].quantile([0.25,
    median = df_year['ER visits for Asthma (rate per 10,000)'].median()
    max_val = df_year['ER visits for Asthma (rate per 10,000)'].max()
    print(f"Year: {year}, Minimum: {min_val:.2f}, Q1: {q1:.2f}, Median: {media

    iqr = q3 - q1
    lower_bound = q1 - 1.5 * iqr
    upper_bound = q3 + 1.5 * iqr

    outliers = df_year[(df_year['ER visits for Asthma (rate per 10,000)'] < lo
    #dfB_new['Hospitalizations (rate per 10,000)'].append(df_year[(df_year['Ho
    dfC_new = dfC_new.append(df_year[(df_year['ER visits for Asthma (rate per

plt.figure(figsize=(10, 6))  # Set the figure size

# Group the ER visits data by year and create a list of values for each year
data = [group['ER visits for Asthma (rate per 10,000)'].values for year, group

# Create the boxplot
plt.boxplot(data, labels=dfC_new['Year'].unique())

# Set the title and labels
plt.title('Log Data - ER Visits in Group C by Year')
plt.xlabel('Year')
plt.ylabel('ER Visits (rate per 10,000)')

# Show the plot
plt.show()
```

```
Year: 2005, Minimum: 3.13, Q1: 3.33, Median: 3.73, Q3: 3.84, Maximum: 4.47
Year: 2006, Minimum: 3.06, Q1: 3.21, Median: 3.63, Q3: 3.85, Maximum: 4.52
Year: 2007, Minimum: 2.86, Q1: 3.29, Median: 3.57, Q3: 3.86, Maximum: 4.62
Year: 2008, Minimum: 2.94, Q1: 3.37, Median: 3.66, Q3: 3.85, Maximum: 4.59
Year: 2009, Minimum: 3.06, Q1: 3.36, Median: 3.64, Q3: 3.94, Maximum: 4.70
Year: 2010, Minimum: 3.20, Q1: 3.37, Median: 3.60, Q3: 3.88, Maximum: 4.69
Year: 2011, Minimum: 3.04, Q1: 3.34, Median: 3.54, Q3: 3.79, Maximum: 4.67
Year: 2012, Minimum: 3.02, Q1: 3.51, Median: 3.64, Q3: 4.03, Maximum: 4.78
Year: 2013, Minimum: 2.62, Q1: 3.35, Median: 3.63, Q3: 3.97, Maximum: 4.71
Year: 2014, Minimum: 3.22, Q1: 3.42, Median: 3.70, Q3: 4.02, Maximum: 4.69
Year: 2015, Minimum: 2.54, Q1: 3.38, Median: 3.69, Q3: 3.90, Maximum: 4.67
Year: 2016, Minimum: 2.69, Q1: 3.26, Median: 3.55, Q3: 3.84, Maximum: 4.61
Year: 2017, Minimum: 2.91, Q1: 3.25, Median: 3.51, Q3: 3.87, Maximum: 4.64
Year: 2018, Minimum: 2.71, Q1: 3.39, Median: 3.58, Q3: 3.87, Maximum: 4.59
Year: 2019, Minimum: 2.46, Q1: 3.30, Median: 3.66, Q3: 3.84, Maximum: 4.56
```



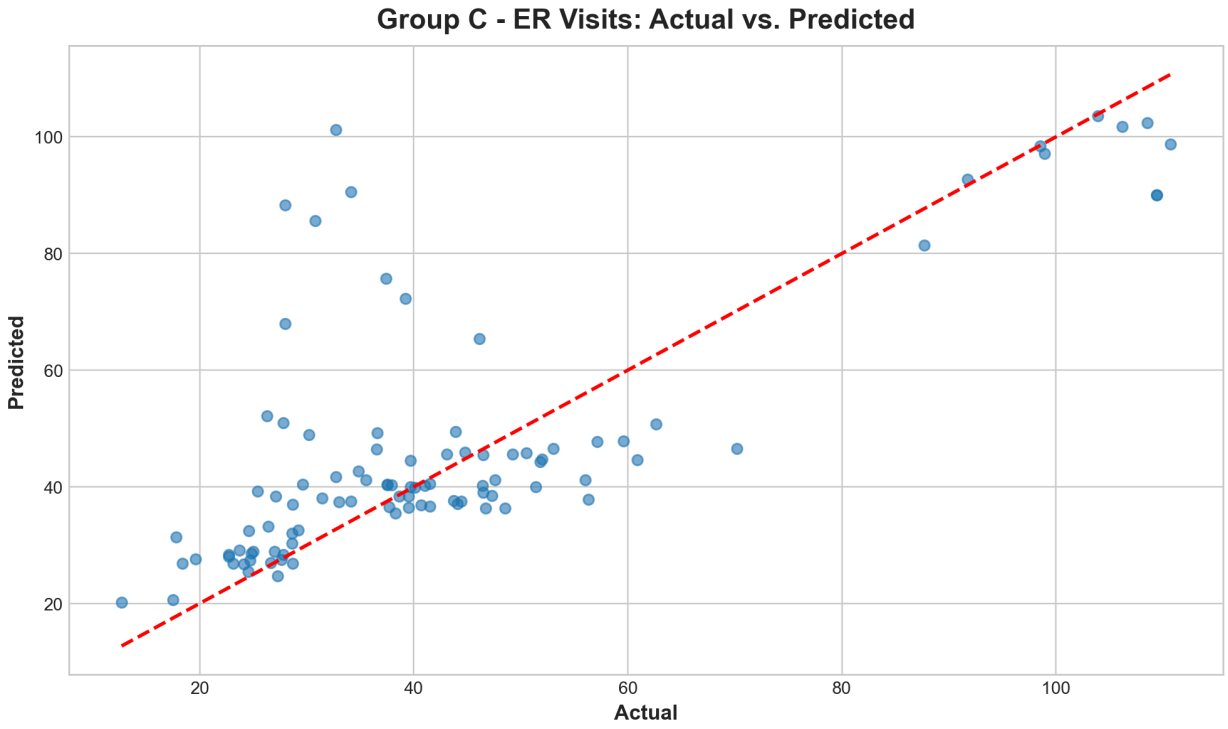
Random Forest

In [ ]:
```
#aqi_erA_adj = aqi_erA_adj.dropna()
aqi_erC_adj = aqi_erC_adj.replace([np.inf, -np.inf], np.nan).dropna()

#feature_cols = ['Days with AQI', 'Moderate Days', 'Median AQI', 'Days CO', 'Da
feature_cols = ['Days with AQI', 'Good Days',
       'Moderate Days', 'Unhealthy for Sensitive Groups Days',
       'Unhealthy Days', 'Very Unhealthy Days', 'Hazardous Days', 'Max AQI',
       '90th Percentile AQI', 'Median AQI', 'Days CO', 'Days NO2',
       'Days Ozone', 'Days PM2.5', 'Days PM10']

X_er = aqi_erC_adj[feature_cols]
y_er = aqi_erC_adj['ER visits for Asthma (rate per 10,000)']

#X_er = dfA_new[feature_cols]
#y_er = dfA_new['ER visits for Asthma (rate per 10,000)']

X_train_er, X_test_er, y_train_er, y_test_er = train_test_split(X_er, y_er, tes
```

In [ ]:
```
model_er = RandomForestRegressor(n_estimators = 100, random_state = 42)

model_er.fit(X_train_er, y_train_er)
predictions_er = model_er.predict(X_test_er)

mse_er = mean_squared_error(y_test_er, predictions_er)
r2_er = r2_score(y_test_er, predictions_er)

print('ER Visits')
print(f'MSE: {mse_er}')
print(f'R^2 Score: {r2_er}')
rmse = float(format(np.sqrt(mean_squared_error(y_test_er, predictions_er)),'.3
print("\nRMSE:\n",rmse)
```

```
# Function to plot actual vs. predicted values
def plot_actual_vs_predicted(y_actual, y_predicted, title):
    plt.figure(figsize=(10, 6))
    plt.scatter(y_actual, y_predicted, alpha=0.6)
    plt.plot([y_actual.min(), y_actual.max()], [y_actual.min(), y_actual.max()]
    plt.xlabel('Actual')
    plt.ylabel('Predicted')
    plt.title(title)
    plt.show()

plot_actual_vs_predicted(y_test_er, predictions_er, 'Group C - ER Visits: Actua

# Feature Importance
feature_importances_er = model_er.feature_importances_
features_er = X_er.columns
feature_importances_df_er = pd.DataFrame({'feature': features_er, 'importance'
feature_importances_df_er = feature_importances_df_er.sort_values('importance'
feature_importances_df_er
```

```
ER Visits
MSE: 263.83290968686845
R^2 Score: 0.49171371316816126

RMSE:
 16.243
```



Out[ ]:

	feature	importance
13	Days PM2.5	0.205870
11	Days NO2	0.190887
0	Days with AQI	0.097258
14	Days PM10	0.093428
7	Max AQI	0.086873
2	Moderate Days	0.081432
3	Unhealthy for Sensitive Groups Days	0.053241
1	Good Days	0.046181
9	Median AQI	0.043725
8	90th Percentile AQI	0.034508
12	Days Ozone	0.034261
10	Days CO	0.025763
4	Unhealthy Days	0.003895
5	Very Unhealthy Days	0.002597
6	Hazardous Days	0.000081

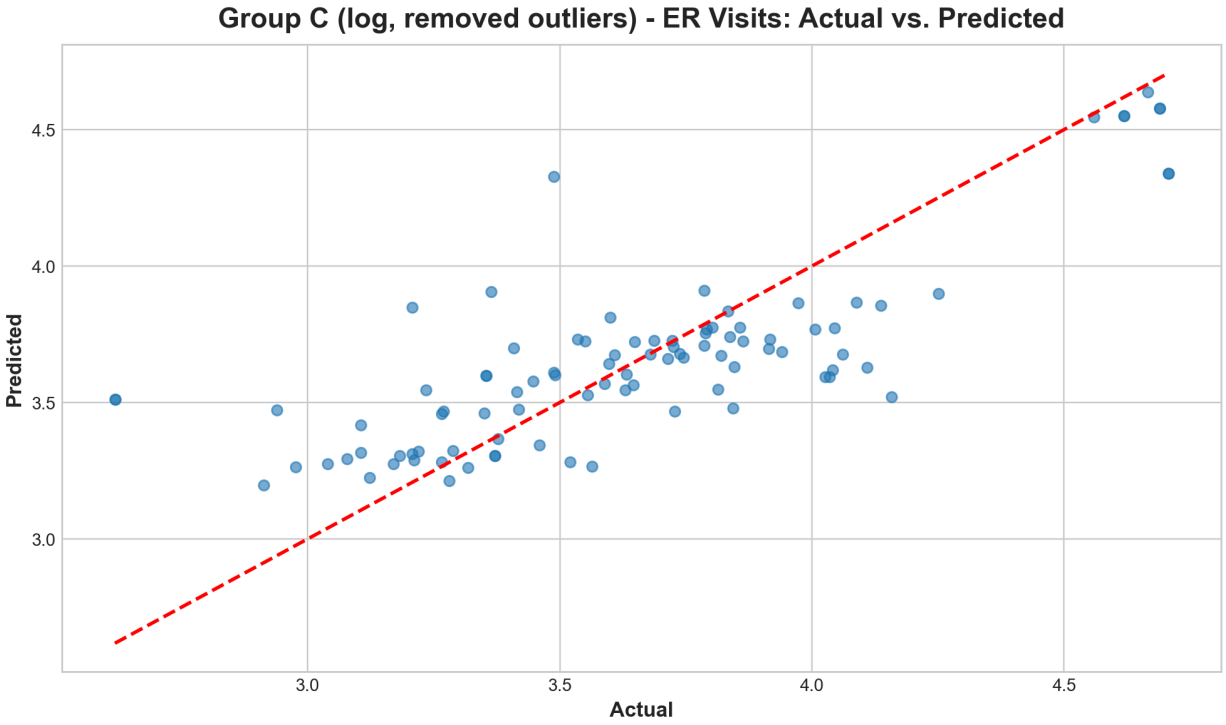
log transform with removed outliers

In [ ]:

```
X_er = dfC_new[feature_cols]
y_er = dfC_new['ER visits for Asthma (rate per 10,000)']

X_train_er, X_test_er, y_train_er, y_test_er = train_test_split(X_er, y_er, te

model_er = RandomForestRegressor(n_estimators = 100, random_state = 42)

model_er.fit(X_train_er, y_train_er)
predictions_er = model_er.predict(X_test_er)

mse_er = mean_squared_error(y_test_er, predictions_er)
r2_er = r2_score(y_test_er, predictions_er)

print('ER Visits')
print(f'MSE: {mse_er}')
print(f'R^2 Score: {r2_er}')
rmse = float(format(np.sqrt(mean_squared_error(y_test_er, predictions_er)),'.3
print("\nRMSE:\n",rmse)

# Function to plot actual vs. predicted values
def plot_actual_vs_predicted(y_actual, y_predicted, title):
    plt.figure(figsize=(10, 6))
    plt.scatter(y_actual, y_predicted, alpha=0.6)
    plt.plot([y_actual.min(), y_actual.max()], [y_actual.min(), y_actual.max()
    plt.xlabel('Actual')
    plt.ylabel('Predicted')
    plt.title(title)
    plt.show()
```

```
plot_actual_vs_predicted(y_test_er, predictions_er, 'Group C (log, removed out

# Feature Importance
feature_importances_er = model_er.feature_importances_
features_er = X_er.columns
feature_importances_df_er = pd.DataFrame({'feature': features_er, 'importance'
feature_importances_df_er = feature_importances_df_er.sort_values('importance'
feature_importances_df_er
```

```
ER Visits
MSE: 0.07515755116117233
R^2 Score: 0.6324981717081315

RMSE:
 0.274
```



Out[ ]:

	feature	importance
13	Days PM2.5	0.170319
11	Days NO2	0.147256
9	Median AQI	0.103050
2	Moderate Days	0.086485
14	Days PM10	0.080111
7	Max AQI	0.077482
12	Days Ozone	0.073484
0	Days with AQI	0.069254
8	90th Percentile AQI	0.064386
1	Good Days	0.063835
3	Unhealthy for Sensitive Groups Days	0.056220
4	Unhealthy Days	0.004664
5	Very Unhealthy Days	0.002750
6	Hazardous Days	0.000538
10	Days CO	0.000170

OLS

In [ ]:

```
# Create a DataFrame with all the independent variables
X = dfC_new[feature_cols]

# Add a constant to the DataFrame
X = sm.add_constant(X)

# Create a Series with the dependent variable
y = dfC_new['ER visits for Asthma (rate per 10,000)']

# Create a model
model = sm.OLS(y, X).fit()

# Get the summary of the model
print(model.summary())
print("Parameters: ", model.params)
print("R2: ", model.rsquared)
```

OLS Regression Results

Dep. Variable:	ER visits for Asthma (rate per 10,000)	R-squared:	0.196
Model:	OLS	Adj. R-squared:	0.161
Method:	Least Squares	F-statistic:	5.610
Date:	Mon, 29 Apr 2024	Prob (F-statistic):	3.59e-09
Time:	19:47:09	Log-Likelihood:	-162.47
No. Observations:	314	AIC:	352.9
Df Residuals:	300	BIC:	405.4
Df Model:	13		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	3.8386	0.181	21.252	0.000	3.483	4.194
Days with AQI	-0.0313	0.061	-0.511	0.609	-0.152	0.089
Good Days	-0.0160	0.011	-1.484	0.139	-0.037	0.005
Moderate Days	-0.0120	0.011	-1.129	0.260	-0.033	0.009
Unhealthy for Sensitive Groups Days	-0.0093	0.011	-0.857	0.392	-0.031	0.012
Unhealthy Days	-0.0168	0.013	-1.304	0.193	-0.042	0.009
Very Unhealthy Days	0.0094	0.018	0.513	0.609	-0.027	0.045
Hazardous Days	0.0134	0.017	0.791	0.429	-0.020	0.047
Max AQI	-0.0005	0.000	-2.586	0.010	-0.001	-0.000
90th Percentile AQI	-0.0084	0.003	-2.586	0.010	-0.015	-0.002
Median AQI	-0.0042	0.006	-0.664	0.507	-0.016	0.008
Days CO	-0.2437	0.347	-0.702	0.483	-0.927	0.439
Days NO2	0.0682	0.072	0.951	0.342	-0.073	0.209
Days Ozone	0.0480	0.071	0.672	0.502	-0.093	0.189
Days PM2.5	0.0479	0.071	0.671	0.503	-0.093	0.189
Days PM10	0.0483	0.071	0.676	0.500	-0.092	0.189

Omnibus:	8.884	Durbin-Watson:	1.714
Prob(Omnibus):	0.012	Jarque-Bera (JB):	8.357

```
Skew:                          0.348   Prob(JB):                     0.0153
Kurtosis:                      2.606   Cond. No.                     3.15e+16
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correct
ly specified.
[2] The smallest eigenvalue is 8.03e-26. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.
Parameters:  const                                3.838591
Days with AQI                          -0.031347
Good Days                              -0.016050
Moderate Days                          -0.012009
Unhealthy for Sensitive Groups Days    -0.009297
Unhealthy Days                         -0.016786
Very Unhealthy Days                     0.009402
Hazardous Days                          0.013393
Max AQI                                -0.000464
90th Percentile AQI                    -0.008366
Median AQI                             -0.004159
Days CO                                -0.243745
Days NO2                                0.068159
Days Ozone                              0.048022
Days PM2.5                              0.047938
Days PM10                               0.048279
dtype: float64
R2:  0.19554830455126415
```

SARIMAX

In [ ]:

```
data = dfC_new.copy()
data = data.groupby('Year').mean()
data = data.drop(columns=['slope', 'StateFIPS',  'CountyFIPS',
        'Days with AQI', 'Good Days',
       'Moderate Days', 'Unhealthy for Sensitive Groups Days',
       'Unhealthy Days', 'Very Unhealthy Days', 'Hazardous Days', 'Max AQI',
       '90th Percentile AQI', 'Median AQI', 'Days CO', 'Days NO2',
         'Days Ozone', 'Days PM10', 'FIPS'])

en = data['ER visits for Asthma (rate per 10,000)']
ex = data['Days PM2.5']
order = (1, 1, 1)
seasonal_order = (1, 0, 0, 2)



model = sm.tsa.SARIMAX(endog=en, exog=ex, order=order, seasonal_order=seasonal_
results=model.fit()

data['forecast']=results.predict(start=1,end=14,dynamic=False)
data[['ER visits for Asthma (rate per 10,000)','forecast']].plot(figsize=(8,4)

results.summary()
```

```
RUNNING THE L-BFGS-B CODE

           * * *

Machine precision = 2.220D-16
 N =            5     M =           10

At X0         0 variables are exactly at the bounds

At iterate    0    f= -7.68380D-01    |proj g|=  2.09595D+01

At iterate    5    f= -7.96523D-01    |proj g|=  7.13556D+00

At iterate   10    f= -8.01872D-01    |proj g|=  4.33642D-01

At iterate   15    f= -8.06518D-01    |proj g|=  5.07282D+00

At iterate   20    f= -8.09885D-01    |proj g|=  2.20079D-01

At iterate   25    f= -8.10832D-01    |proj g|=  1.54708D-02

At iterate   30    f= -8.10919D-01    |proj g|=  6.99061D-01

At iterate   35    f= -8.11298D-01    |proj g|=  9.10468D-03

At iterate   40    f= -8.11316D-01    |proj g|=  1.83627D-02

           * * *

Tit   = total number of iterations
Tnf   = total number of function evaluations
Tnint = total number of segments explored during Cauchy searches
Skip  = number of BFGS updates skipped
Nact  = number of active bounds at final generalized Cauchy point
Projg = norm of the final projected gradient
F     = final function value

           * * *

   N    Tit     Tnf  Tnint  Skip  Nact     Projg        F
    5     43     50      1     0     0   6.225D-06  -8.113D-01
  F = -0.81131597196503624

CONVERGENCE: NORM_OF_PROJECTED_GRADIENT_<=_PGTOL
 This problem is unconstrained.
```

Out[ ]:

SARIMAX Results

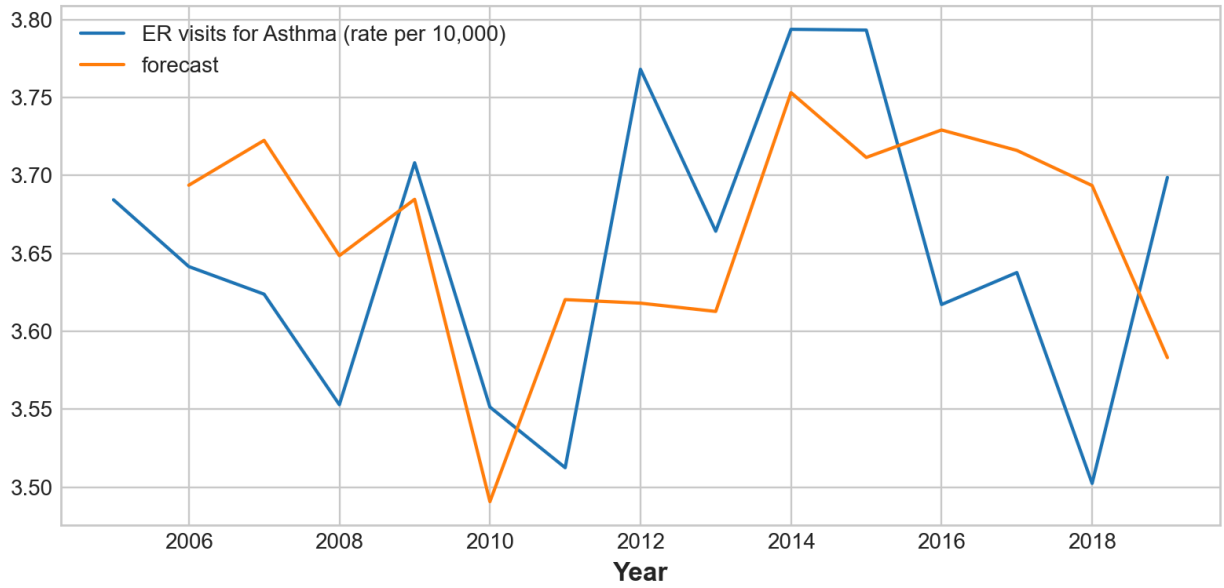
Dep. Variable:	ER visits for Asthma (rate per 10,000)	**No. Observations:**	15
Model:	SARIMAX(1, 1, 1)x(1, 0, [], 2)	**Log Likelihood**	12.170
Date:	Mon, 29 Apr 2024	**AIC**	-14.339
Time:	19:47:09	**BIC**	-11.144
Sample:	0	**HQIC**	-14.635
	- 15		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
Days PM2.5	0.0047	0.003	1.374	0.169	-0.002	0.011
ar.L1	-0.5580	1.397	-0.399	0.690	-3.297	2.181
ma.L1	-0.1218	1.519	-0.080	0.936	-3.099	2.855
ar.S.L2	0.0660	1.043	0.063	0.950	-1.978	2.110
sigma2	0.0099	0.009	1.056	0.291	-0.008	0.028

Ljung-Box (L1) (Q):	0.07	**Jarque-Bera (JB):**	0.78
Prob(Q):	0.79	**Prob(JB):**	0.68
Heteroskedasticity (H):	3.06	**Skew:**	-0.02
Prob(H) (two-sided):	0.25	**Kurtosis:**	1.84

Warnings:

[1] Covariance matrix calculated using the outer product of gradients (complex-step).

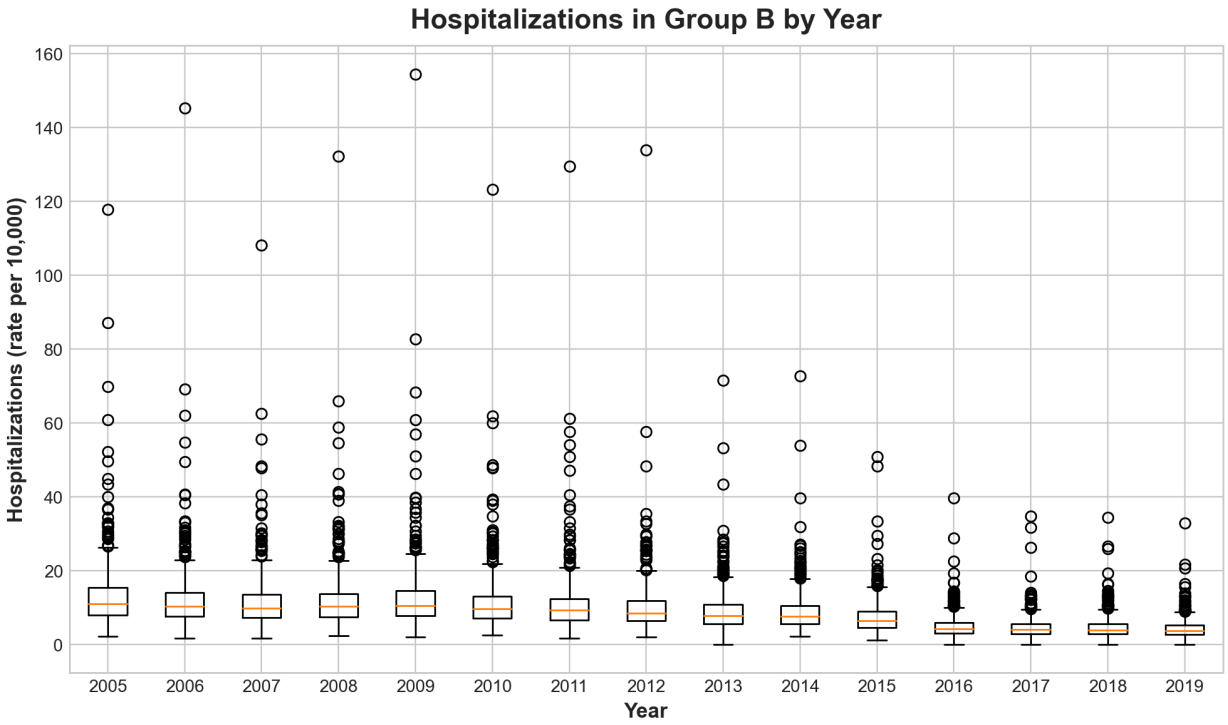


Models for Hospitalizations Group B

In [ ]:
```
#aqi_hospB.info()
#aqi_hospB.isnull().sum()
#aqi_hospB.tail(10)
aqi_hospB_adj = aqi_hospB.dropna()
aqi_hospB_adj.shape
```

Out[ ]:
```
(6311, 24)
```

In [ ]:
```
#boxplot group B
plt.figure(figsize=(10, 6))  # Set the figure size

# Group the ER visits data by year and create a list of values for each year
data = [group['Hospitalizations (rate per 10,000)'].values for year, group in a

# Create the boxplot
plt.boxplot(data, labels=aqi_hospB['Year'].unique())

# Set the title and labels
plt.title('Hospitalizations in Group B by Year')
plt.xlabel('Year')
plt.ylabel('Hospitalizations (rate per 10,000)')

# Show the plot
plt.show()
```



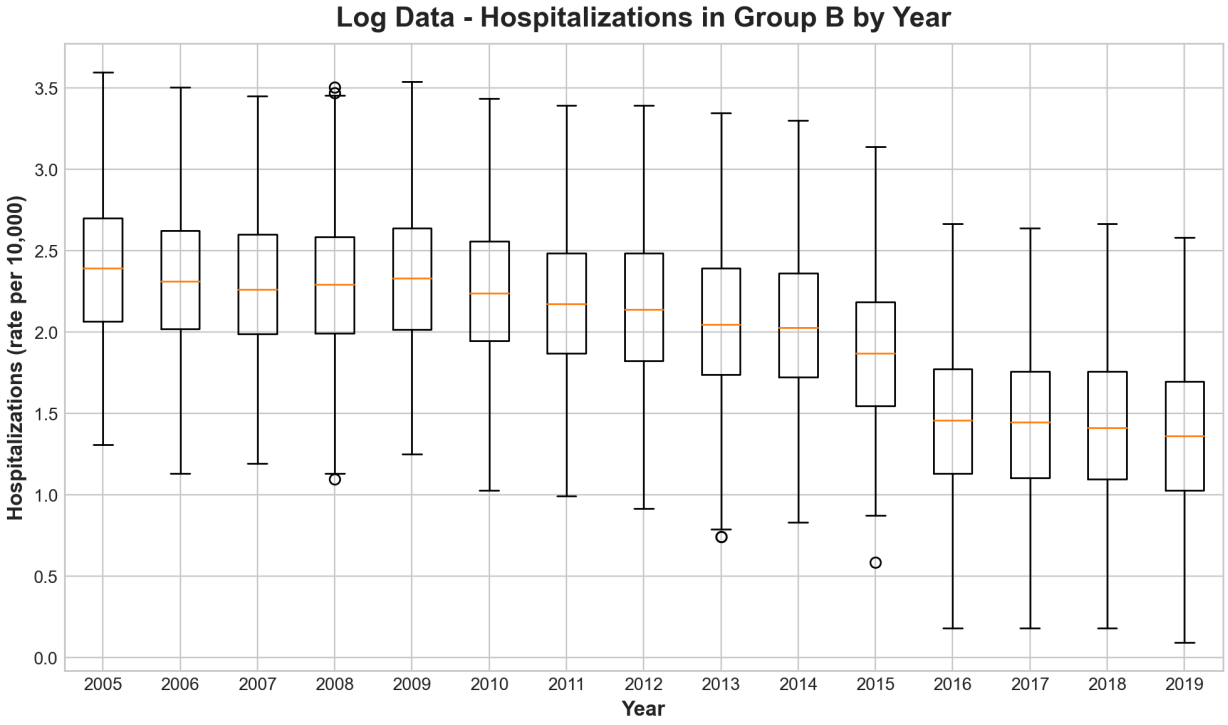
In [ ]:
```
column_names = aqi_hospB.columns
print(column_names)
```

```
Index(['FIPS', 'slope', 'StateFIPS', 'State', 'CountyFIPS', 'County', 'Year',
       'Hospitalizations (rate per 10,000)', 'Days with AQI', 'Good Days',
       'Moderate Days', 'Unhealthy for Sensitive Groups Days',
       'Unhealthy Days', 'Very Unhealthy Days', 'Hazardous Days', 'Max AQI',
       '90th Percentile AQI', 'Median AQI', 'Days CO', 'Days NO2',
       'Days Ozone', 'Days PM2.5', 'Days PM10', 'geometry'],
      dtype='object')
```

In [ ]:
```
B = aqi_hospB_adj['Hospitalizations (rate per 10,000)']
aqi_hospB_log = aqi_hospB_adj.copy()
aqi_hospB_log['Hospitalizations (rate per 10,000)'] = np.log(B)
```

In [ ]:
```
# Assuming your dataset is stored in a variable called 'A'
years = aqi_hospB_log['Year'].unique()  # Assuming the column name for year is
dfB_new = pd.DataFrame()
# Loop through each year and create a variable for it
for year in years:
    # Filter the dataset for the current year
    df_year = aqi_hospB_log[aqi_hospB_log['Year'] == year]
    #print(df_year)
    # Calculate the quartiles
    min_val = df_year['Hospitalizations (rate per 10,000)'].min()
    q1, q3 = df_year['Hospitalizations (rate per 10,000)'].quantile([0.25, 0.7
    median = df_year['Hospitalizations (rate per 10,000)'].median()
    max_val = df_year['Hospitalizations (rate per 10,000)'].max()
    print(f"Year: {year}, Minimum: {min_val:.2f}, Q1: {q1:.2f}, Median: {media

    iqr = q3 - q1
    lower_bound = q1 - 1.5 * iqr
    upper_bound = q3 + 1.5 * iqr

    outliers = df_year[(df_year['Hospitalizations (rate per 10,000)'] < lower_
    #dfB_new['Hospitalizations (rate per 10,000)'].append(df_year[(df_year['Hos
    dfB_new = dfB_new.append(df_year[(df_year['Hospitalizations (rate per 10,0

plt.figure(figsize=(10, 6))  # Set the figure size

# Group the ER visits data by year and create a list of values for each year
data = [group['Hospitalizations (rate per 10,000)'].values for year, group in

# Create the boxplot
plt.boxplot(data, labels=dfB_new['Year'].unique())

# Set the title and labels
plt.title('Log Data - Hospitalizations in Group B by Year')
plt.xlabel('Year')
plt.ylabel('Hospitalizations (rate per 10,000)')

# Show the plot
plt.show()
```

```
Year: 2005, Minimum: 0.74, Q1: 2.07, Median: 2.39, Q3: 2.70, Maximum: 4.11
Year: 2006, Minimum: 1.03, Q1: 2.01, Median: 2.32, Q3: 2.63, Maximum: 4.13
Year: 2007, Minimum: 0.83, Q1: 1.97, Median: 2.26, Q3: 2.60, Maximum: 4.14
Year: 2008, Minimum: 0.88, Q1: 1.99, Median: 2.29, Q3: 2.61, Maximum: 4.19
Year: 2009, Minimum: 0.96, Q1: 2.02, Median: 2.33, Q3: 2.65, Maximum: 4.22
Year: 2010, Minimum: 0.88, Q1: 1.94, Median: 2.24, Q3: 2.56, Maximum: 4.09
Year: 2011, Minimum: 0.74, Q1: 1.87, Median: 2.17, Q3: 2.50, Maximum: 4.05
Year: 2012, Minimum: 0.92, Q1: 1.82, Median: 2.15, Q3: 2.49, Maximum: 4.05
Year: 2013, Minimum: 0.34, Q1: 1.73, Median: 2.05, Q3: 2.40, Maximum: 3.97
Year: 2014, Minimum: 0.74, Q1: 1.72, Median: 2.03, Q3: 2.37, Maximum: 3.99
Year: 2015, Minimum: 0.18, Q1: 1.54, Median: 1.87, Q3: 2.19, Maximum: 3.88
Year: 2016, Minimum: -0.22, Q1: 1.13, Median: 1.46, Q3: 1.77, Maximum: 3.68
Year: 2017, Minimum: -0.11, Q1: 1.10, Median: 1.44, Q3: 1.76, Maximum: 3.55
Year: 2018, Minimum: -0.36, Q1: 1.10, Median: 1.39, Q3: 1.76, Maximum: 3.54
Year: 2019, Minimum: -0.36, Q1: 1.03, Median: 1.36, Q3: 1.70, Maximum: 3.49
```



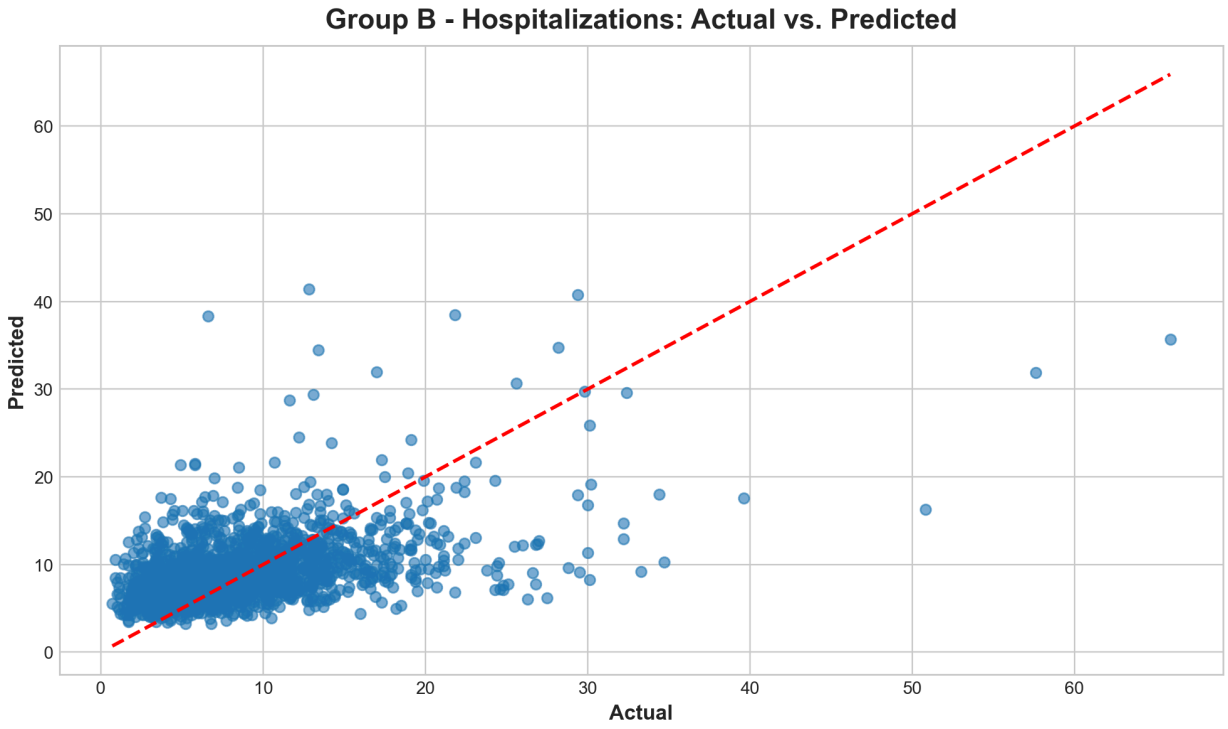
Random Forest

In [ ]:
```
#aqi_erA_adj = aqi_erA_adj.dropna()
aqi_hospB_adj = aqi_hospB_adj.replace([np.inf, -np.inf], np.nan).dropna()

#feature_cols = ['Days with AQI', 'Moderate Days', 'Median AQI', 'Days CO', 'Da
feature_cols = ['Days with AQI', 'Good Days',
       'Moderate Days', 'Unhealthy for Sensitive Groups Days',
       'Unhealthy Days', 'Very Unhealthy Days', 'Hazardous Days', 'Max AQI',
       '90th Percentile AQI', 'Median AQI', 'Days CO', 'Days NO2',
       'Days Ozone', 'Days PM2.5', 'Days PM10']

X_hosp = aqi_hospB_adj[feature_cols]
y_hosp = aqi_hospB_adj['Hospitalizations (rate per 10,000)']

#X_er = dfA_new[feature_cols]
#y_er = dfA_new['ER visits for Asthma (rate per 10,000)']

X_train_hosp, X_test_hosp, y_train_hosp, y_test_hosp = train_test_split(X_hosp
```

In [ ]:
```
model_hosp = RandomForestRegressor(n_estimators = 100, random_state = 42)

model_hosp.fit(X_train_hosp, y_train_hosp)
predictions_hosp = model_hosp.predict(X_test_hosp)

mse_hosp = mean_squared_error(y_test_hosp, predictions_hosp)
r2_hosp = r2_score(y_test_hosp, predictions_hosp)

print('Hospitalizations')
print(f'MSE: {mse_hosp}')
print(f'R^2 Score: {r2_hosp}')
rmse = float(format(np.sqrt(mean_squared_error(y_test_er, predictions_er)),'.3
print("\nRMSE:\n",rmse)
```

```
# Function to plot actual vs. predicted values
def plot_actual_vs_predicted(y_actual, y_predicted, title):
    plt.figure(figsize=(10, 6))
    plt.scatter(y_actual, y_predicted, alpha=0.6)
    plt.plot([y_actual.min(), y_actual.max()], [y_actual.min(), y_actual.max()]
    plt.xlabel('Actual')
    plt.ylabel('Predicted')
    plt.title(title)
    plt.show()

plot_actual_vs_predicted(y_test_hosp, predictions_hosp, 'Group B - Hospitaliza

# Feature Importance
feature_importances_hosp = model_hosp.feature_importances_
features_hosp = X_hosp.columns
feature_importances_df_hosp = pd.DataFrame({'feature': features_hosp, 'importa
feature_importances_df_hosp = feature_importances_df_hosp.sort_values('importa
feature_importances_df_hosp
```

```
Hospitalizations
MSE: 25.485789507919748
R^2 Score: 0.17297783279858758

RMSE:
 0.274
```



Out[ ]:

	feature	importance
12	Days Ozone	0.130591
8	90th Percentile AQI	0.110721
13	Days PM2.5	0.106270
7	Max AQI	0.104593
11	Days NO2	0.104464
1	Good Days	0.094820
2	Moderate Days	0.093283
9	Median AQI	0.065728
3	Unhealthy for Sensitive Groups Days	0.062394
0	Days with AQI	0.055157
14	Days PM10	0.037974
10	Days CO	0.016960
4	Unhealthy Days	0.015345
5	Very Unhealthy Days	0.001641
6	Hazardous Days	0.000059

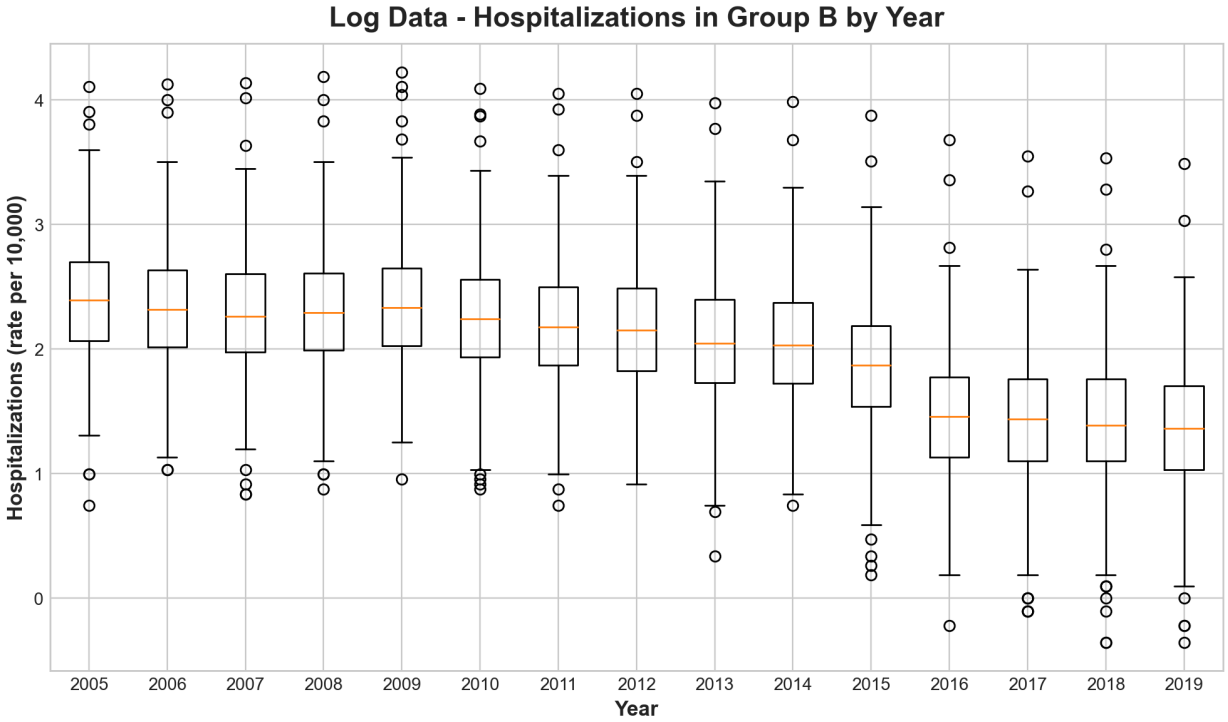
In [ ]:

```
# CAN WE REMOVE THIS?

"""model = RandomForestRegressor(n_estimators = 100, random_state = 42)

model.fit(X_train, y_train)
predictions = model.predict(X_test)

mse_hosp = mean_squared_error(y_test, predictions)
r2_hosp = r2_score(y_test, predictions)

print('Hospitalizations')
print(f'MSE: {mse_hosp}')
print(f'R^2 Score: {r2_hosp}')

# Function to plot actual vs. predicted values
def plot_actual_vs_predicted(y_actual, y_predicted, title):
    plt.figure(figsize=(10, 6))
    plt.scatter(y_actual, y_predicted, alpha=0.6)
    plt.plot([y_actual.min(), y_actual.max()], [y_actual.min(), y_actual.max()]
    plt.xlabel('Actual')
    plt.ylabel('Predicted')
    plt.title(title)
    plt.show()

plot_actual_vs_predicted(y_test, predictions, 'Group B - Hospitalizations: Actu
```

Out[ ]:
```
"model = RandomForestRegressor(n_estimators = 100, random_state = 42)\n\nmode
l.fit(X_train, y_train)\npredictions = model.predict(X_test)\n\nmse_hosp = mea
n_squared_error(y_test, predictions)\nr2_hosp = r2_score(y_test, predictions)
\n\nprint('Hospitalizations')\nprint(f'MSE: {mse_hosp}')\nprint(f'R^2 Score:
{r2_hosp}')\n\n# Function to plot actual vs. predicted values\ndef plot_actual
_vs_predicted(y_actual, y_predicted, title):\n    plt.figure(figsize=(10, 6))
\n    plt.scatter(y_actual, y_predicted, alpha=0.6)\n    plt.plot([y_actual.mi
n(), y_actual.max()], [y_actual.min(), y_actual.max()], 'r--', lw=2)\n    plt.
xlabel('Actual')\n    plt.ylabel('Predicted')\n    plt.title(title)\n    plt.s
how()\n\nplot_actual_vs_predicted(y_test, predictions, 'Group B - Hospitalizat
ions: Actual vs. Predicted')"
```

With Outliers

In [ ]:
```
#aqi_hospB_log
#boxplot group B
plt.figure(figsize=(10, 6))  # Set the figure size

# Group the Hospitalizations data by year and create a list of values for each
data = [group['Hospitalizations (rate per 10,000)'].values for year, group in a

# Create the boxplot
plt.boxplot(data, labels=aqi_hospB_log['Year'].unique())

# Set the title and labels
plt.title('Log Data - Hospitalizations in Group B by Year')
plt.xlabel('Year')
plt.ylabel('Hospitalizations (rate per 10,000)')

# Show the plot
plt.show()
```



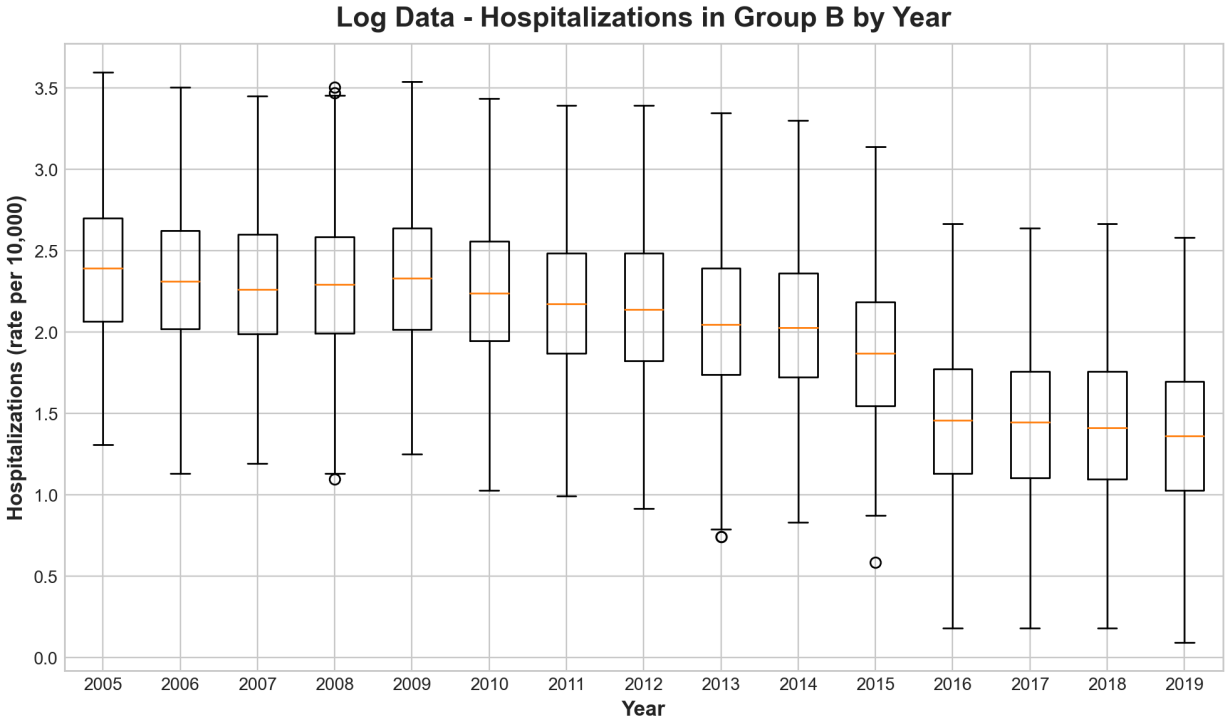
log transform with removed outliers

In [ ]:
```
"""min_val = aqi_hospB_log['Hospitalizations (rate per 10,000)'].min()
q1, q3 = np.percentile(aqi_hospB_log['Hospitalizations (rate per 10,000)'], [25
median = aqi_hospB_log['Hospitalizations (rate per 10,000)'].median()
max_val = aqi_hospB_log['Hospitalizations (rate per 10,000)'].max()
print(f"Minimum: {min_val:.2f}, Q1: {q1:.2f}, Median: {median:.2f}, Q3: {q3:.2
```

Out[ ]:
```
'min_val = aqi_hospB_log[\'Hospitalizations (rate per 10,000)\'].min()\nq1, q3
= np.percentile(aqi_hospB_log[\'Hospitalizations (rate per 10,000)\'], [25, 7
5])\nmedian = aqi_hospB_log[\'Hospitalizations (rate per 10,000)\'].median()\n
max_val = aqi_hospB_log[\'Hospitalizations (rate per 10,000)\'].max()\nprint
(f"Minimum: {min_val:.2f}, Q1: {q1:.2f}, Median: {median:.2f}, Q3: {q3:.2f}, M
aximum: {max_val:.2f}")'
```

In [ ]:
```
# Assuming your dataset is stored in a variable called 'A'
years = aqi_hospB_log['Year'].unique()  # Assuming the column name for year is
dfB_new = pd.DataFrame()
# Loop through each year and create a variable for it
for year in years:
    # Filter the dataset for the current year
    df_year = aqi_hospB_log[aqi_hospB_log['Year'] == year]
    #print(df_year)
    # Calculate the quartiles
    min_val = df_year['Hospitalizations (rate per 10,000)'].min()
    q1, q3 = df_year['Hospitalizations (rate per 10,000)'].quantile([0.25, 0.75
    median = df_year['Hospitalizations (rate per 10,000)'].median()
    max_val = df_year['Hospitalizations (rate per 10,000)'].max()
    print(f"Year: {year}, Minimum: {min_val:.2f}, Q1: {q1:.2f}, Median: {median

    iqr = q3 - q1
    lower_bound = q1 - 1.5 * iqr
    upper_bound = q3 + 1.5 * iqr

    outliers = df_year[(df_year['Hospitalizations (rate per 10,000)'] < lower_b
    #dfB_new['Hospitalizations (rate per 10,000)'].append(df_year[(df_year['Hos
    dfB_new = dfB_new.append(df_year[(df_year['Hospitalizations (rate per 10,00

plt.figure(figsize=(10, 6))  # Set the figure size

# Group the ER visits data by year and create a list of values for each year
data = [group['Hospitalizations (rate per 10,000)'].values for year, group in d

# Create the boxplot
plt.boxplot(data, labels=dfB_new['Year'].unique())

# Set the title and labels
plt.title('Log Data - Hospitalizations in Group B by Year')
plt.xlabel('Year')
plt.ylabel('Hospitalizations (rate per 10,000)')

# Show the plot
plt.show()
```

```
Year: 2005, Minimum: 0.74, Q1: 2.07, Median: 2.39, Q3: 2.70, Maximum: 4.11
Year: 2006, Minimum: 1.03, Q1: 2.01, Median: 2.32, Q3: 2.63, Maximum: 4.13
Year: 2007, Minimum: 0.83, Q1: 1.97, Median: 2.26, Q3: 2.60, Maximum: 4.14
Year: 2008, Minimum: 0.88, Q1: 1.99, Median: 2.29, Q3: 2.61, Maximum: 4.19
Year: 2009, Minimum: 0.96, Q1: 2.02, Median: 2.33, Q3: 2.65, Maximum: 4.22
Year: 2010, Minimum: 0.88, Q1: 1.94, Median: 2.24, Q3: 2.56, Maximum: 4.09
Year: 2011, Minimum: 0.74, Q1: 1.87, Median: 2.17, Q3: 2.50, Maximum: 4.05
Year: 2012, Minimum: 0.92, Q1: 1.82, Median: 2.15, Q3: 2.49, Maximum: 4.05
Year: 2013, Minimum: 0.34, Q1: 1.73, Median: 2.05, Q3: 2.40, Maximum: 3.97
Year: 2014, Minimum: 0.74, Q1: 1.72, Median: 2.03, Q3: 2.37, Maximum: 3.99
Year: 2015, Minimum: 0.18, Q1: 1.54, Median: 1.87, Q3: 2.19, Maximum: 3.88
Year: 2016, Minimum: -0.22, Q1: 1.13, Median: 1.46, Q3: 1.77, Maximum: 3.68
Year: 2017, Minimum: -0.11, Q1: 1.10, Median: 1.44, Q3: 1.76, Maximum: 3.55
Year: 2018, Minimum: -0.36, Q1: 1.10, Median: 1.39, Q3: 1.76, Maximum: 3.54
Year: 2019, Minimum: -0.36, Q1: 1.03, Median: 1.36, Q3: 1.70, Maximum: 3.49
```

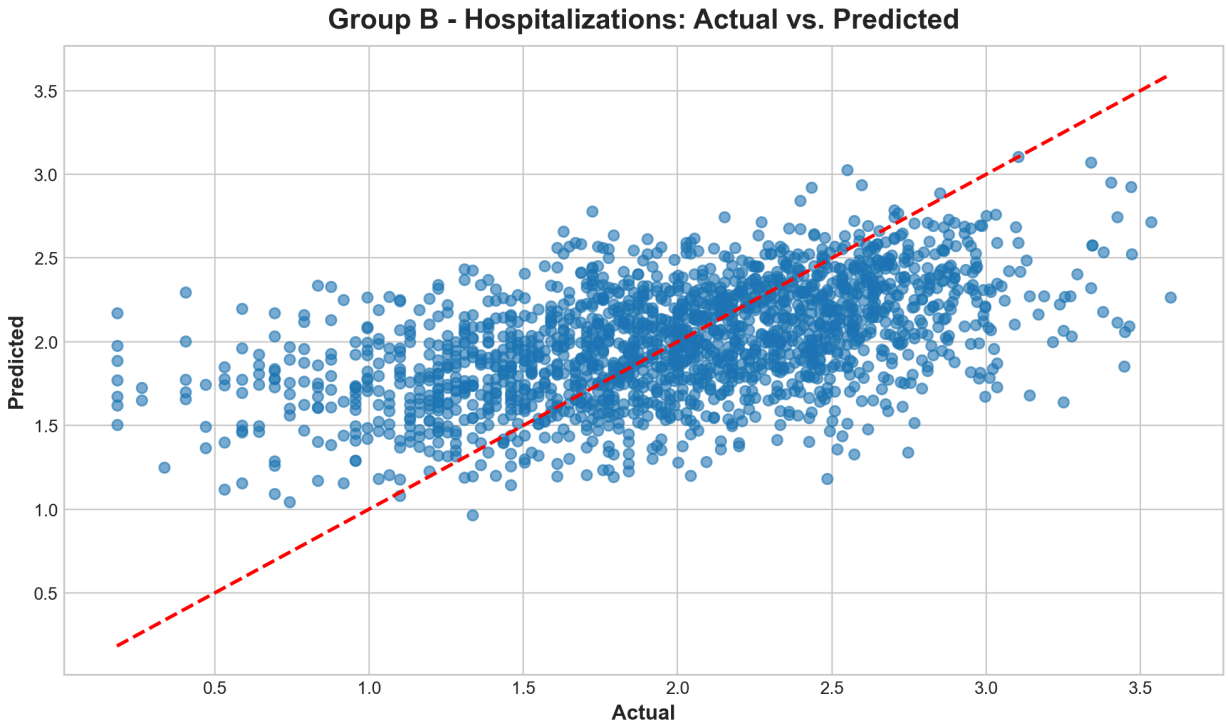


In [ ]:
```
#log transformed and outliers removed
X_hosp = dfB_new[feature_cols]
y_hosp = dfB_new['Hospitalizations (rate per 10,000)']

X_train, X_test, y_train, y_test = train_test_split(X_hosp, y_hosp, test_size =
```

In [ ]:
```
model = RandomForestRegressor(n_estimators = 100, random_state = 42)

model.fit(X_train, y_train)
predictions = model.predict(X_test)

mse_hosp = mean_squared_error(y_test, predictions)
r2_hosp = r2_score(y_test, predictions)

print('Hospitalizations')
print(f'MSE: {mse_hosp}')
print(f'R^2 Score: {r2_hosp}')
rmse = float(format(np.sqrt(mean_squared_error(y_test, predictions)),'.3f'))
print("\nRMSE:\n",rmse)

# Function to plot actual vs. predicted values
```

```
def plot_actual_vs_predicted(y_actual, y_predicted, title):
    plt.figure(figsize=(10, 6))
    plt.scatter(y_actual, y_predicted, alpha=0.6)
    plt.plot([y_actual.min(), y_actual.max()], [y_actual.min(), y_actual.max()]
    plt.xlabel('Actual')
    plt.ylabel('Predicted')
    plt.title(title)
    plt.show()

plot_actual_vs_predicted(y_test, predictions, 'Group B - Hospitalizations: Actu
```

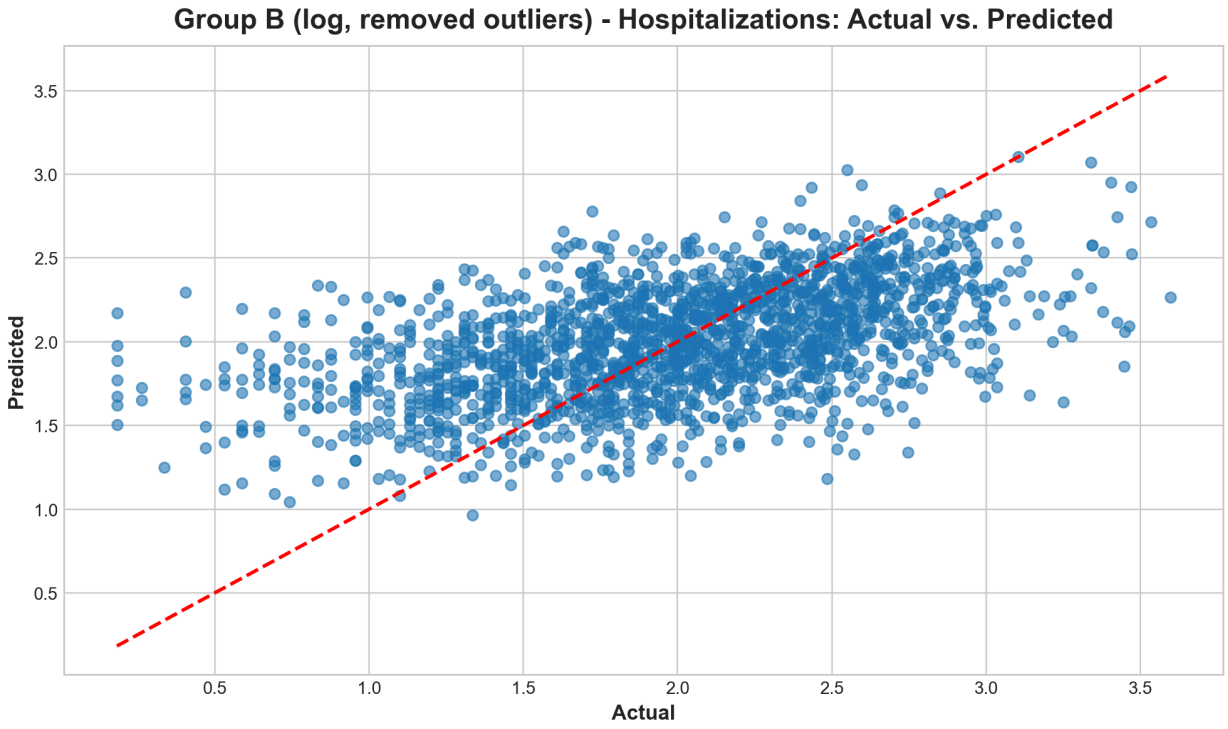
```
Hospitalizations
MSE: 0.2631599590816444
R^2 Score: 0.24858629759657236

RMSE:
 0.513
```



In [ ]:
```
X_hosp = dfB_new[feature_cols]
y_hosp = dfB_new['Hospitalizations (rate per 10,000)']

X_train_hosp, X_test_hosp, y_train_hosp, y_test_hosp = train_test_split(X_hosp

model_hosp = RandomForestRegressor(n_estimators = 100, random_state = 42)

model_hosp.fit(X_train_hosp, y_train_hosp)
predictions_hosp = model_hosp.predict(X_test_hosp)

mse_hosp = mean_squared_error(y_test_hosp, predictions_hosp)
r2_hosp = r2_score(y_test_hosp, predictions_hosp)

print('Hospitalizations')
print(f'MSE: {mse_hosp}')
print(f'R^2 Score: {r2_hosp}')
rmse = float(format(np.sqrt(mean_squared_error(y_test, predictions)),'.3f'))
print("\nRMSE:\n",rmse)
```

```
# Function to plot actual vs. predicted values
def plot_actual_vs_predicted(y_actual, y_predicted, title):
    plt.figure(figsize=(10, 6))
    plt.scatter(y_actual, y_predicted, alpha=0.6)
    plt.plot([y_actual.min(), y_actual.max()], [y_actual.min(), y_actual.max()]
    plt.xlabel('Actual')
    plt.ylabel('Predicted')
    plt.title(title)
    plt.show()

plot_actual_vs_predicted(y_test_hosp, predictions_hosp, 'Group B (log, removed

# Feature Importance
feature_importances_hosp = model_hosp.feature_importances_
features_hosp = X_hosp.columns
feature_importances_df_hosp = pd.DataFrame({'feature': features_hosp, 'importa
feature_importances_df_hosp = feature_importances_df_hosp.sort_values('importa
feature_importances_df_hosp
```

```
Hospitalizations
MSE: 0.2631599590816444
R^2 Score: 0.24858629759657236

RMSE:
 0.513
```



Out[ ]:

	feature	importance
8	90th Percentile AQI	0.124611
12	Days Ozone	0.112983
7	Max AQI	0.108735
13	Days PM2.5	0.105639
2	Moderate Days	0.103606
1	Good Days	0.098099
9	Median AQI	0.087986
0	Days with AQI	0.067340
11	Days NO2	0.064894
3	Unhealthy for Sensitive Groups Days	0.048497
14	Days PM10	0.041774
4	Unhealthy Days	0.018067
10	Days CO	0.015412
5	Very Unhealthy Days	0.002082
6	Hazardous Days	0.000276

OLS

In [ ]:

```
# Create a DataFrame with all the independent variables
X = dfB_new[feature_cols]

# Add a constant to the DataFrame
X = sm.add_constant(X)

# Create a Series with the dependent variable
y = dfB_new['Hospitalizations (rate per 10,000)']

# Create a model
model = sm.OLS(y, X).fit()

# Get the summary of the model
print(model.summary())
print("Parameters: ", model.params)
print("R2: ", model.rsquared)
```

OLS Regression Results

Dep. Variable:	Hospitalizations (rate per 10,000)	R-squared:	0.100
Model:	OLS	Adj. R-squared:	0.098
Method:	Least Squares	F-statistic:	52.87
Date:	Mon, 29 Apr 2024	Prob (F-statistic):	5.05e-131
Time:	19:47:15	Log-Likelihood:	-5223.0
No. Observations:	6224	AIC:	1.047e+04
Df Residuals:	6210	BIC:	1.057e+04
Df Model:	13		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	1.8777	0.084	22.233	0.000	1.712	2.043
Days with AQI	-2.874e+09	4.14e+09	-0.695	0.487	-1.1e+10	5.24e+09
Good Days	1.312e+09	1.89e+09	0.695	0.487	-2.39e+09	5.01e+09
Moderate Days	1.312e+09	1.89e+09	0.695	0.487	-2.39e+09	5.01e+09
Unhealthy for Sensitive Groups Days	1.312e+09	1.89e+09	0.695	0.487	-2.39e+09	5.01e+09
Unhealthy Days	1.312e+09	1.89e+09	0.695	0.487	-2.39e+09	5.01e+09
Very Unhealthy Days	1.312e+09	1.89e+09	0.695	0.487	-2.39e+09	5.01e+09
Hazardous Days	1.312e+09	1.89e+09	0.695	0.487	-2.39e+09	5.01e+09
Max AQI	7.426e-05	0.000	0.491	0.623	-0.000	0.000
90th Percentile AQI	0.0065	0.001	6.885	0.000	0.005	0.008
Median AQI	-0.0021	0.002	-1.134	0.257	-0.006	0.002
Days CO	1.562e+09	2.25e+09	0.695	0.487	-2.85e+09	5.97e+09
Days NO2	1.562e+09	2.25e+09	0.695	0.487	-2.85e+09	5.97e+09
Days Ozone	1.562e+09	2.25e+09	0.695	0.487	-2.85e+09	5.97e+09
Days PM2.5	1.562e+09	2.25e+09	0.695	0.487	-2.85e+09	5.97e+09
Days PM10	1.562e+09	2.25e+09	0.695	0.487	-2.85e+09	5.97e+09

Omnibus:	48.116	Durbin-Watson:	1.020
Prob(Omnibus):	0.000	Jarque-Bera (JB):	47.481

```
Skew:                          -0.196   Prob(JB):                     4.89e-11
Kurtosis:                       2.828   Cond. No.                     3.60e+15
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correct
ly specified.
[2] The smallest eigenvalue is 1.21e-22. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.
Parameters:  const                                1.877740e+00
Days with AQI                          -2.874374e+09
Good Days                               1.311890e+09
Moderate Days                           1.311890e+09
Unhealthy for Sensitive Groups Days     1.311890e+09
Unhealthy Days                          1.311890e+09
Very Unhealthy Days                     1.311890e+09
Hazardous Days                          1.311890e+09
Max AQI                                 7.425798e-05
90th Percentile AQI                     6.460680e-03
Median AQI                             -2.106478e-03
Days CO                                 1.562484e+09
Days NO2                                1.562484e+09
Days Ozone                              1.562484e+09
Days PM2.5                              1.562484e+09
Days PM10                               1.562484e+09
dtype: float64
R2:  0.09964448983556451
```

SARIMAX

In [ ]:

```
data = dfB_new.copy()
data = data.groupby('Year').mean()
data = data.drop(columns=['slope', 'StateFIPS',  'CountyFIPS',
        'Days with AQI', 'Good Days',
       'Moderate Days', 'Unhealthy for Sensitive Groups Days',
       'Unhealthy Days', 'Very Unhealthy Days', 'Hazardous Days', 'Max AQI',
       '90th Percentile AQI', 'Median AQI', 'Days CO', 'Days NO2',
         'Days Ozone', 'Days PM10', 'FIPS'])

en = data['Hospitalizations (rate per 10,000)']
ex = data['Days PM2.5']
order = (1, 1, 1)
seasonal_order = (1, 0, 0, 2)
exog = np.empty([14, 1])


model = sm.tsa.SARIMAX(endog=en, exog=ex, order=order, seasonal_order=seasonal_
results=model.fit()

data['forecast']=results.predict(start=1,end=14,dynamic=False)
data[['Hospitalizations (rate per 10,000)','forecast']].plot(figsize=(8,4))

results.summary()
```

```
RUNNING THE L-BFGS-B CODE

           * * *

Machine precision = 2.220D-16
 N =            5     M =           10

At X0         0 variables are exactly at the bounds

At iterate    0    f= -6.36229D-01    |proj g|=  4.53990D+00

At iterate    5    f= -6.48429D-01    |proj g|=  2.36644D+00

At iterate   10    f= -6.51150D-01    |proj g|=  2.75720D+00

At iterate   15    f= -6.84613D-01    |proj g|=  1.09936D+00

At iterate   20    f= -6.97292D-01    |proj g|=  4.66441D+00

At iterate   25    f= -7.04535D-01    |proj g|=  3.42394D-02

At iterate   30    f= -7.29869D-01    |proj g|=  1.80591D+00

At iterate   35    f= -7.50347D-01    |proj g|=  5.07449D-01

At iterate   40    f= -7.51034D-01    |proj g|=  4.59120D-02

           * * *

Tit   = total number of iterations
Tnf   = total number of function evaluations
Tnint = total number of segments explored during Cauchy searches
Skip  = number of BFGS updates skipped
Nact  = number of active bounds at final generalized Cauchy point
Projg = norm of the final projected gradient
F     = final function value

           * * *

   N    Tit     Tnf  Tnint  Skip  Nact     Projg        F
    5     43     52      1     0     0   2.203D-02  -7.510D-01
  F = -0.75103416824013070

CONVERGENCE: REL_REDUCTION_OF_F_<=_FACTR*EPSMCH
 This problem is unconstrained.
```

Out[ ]:

SARIMAX Results

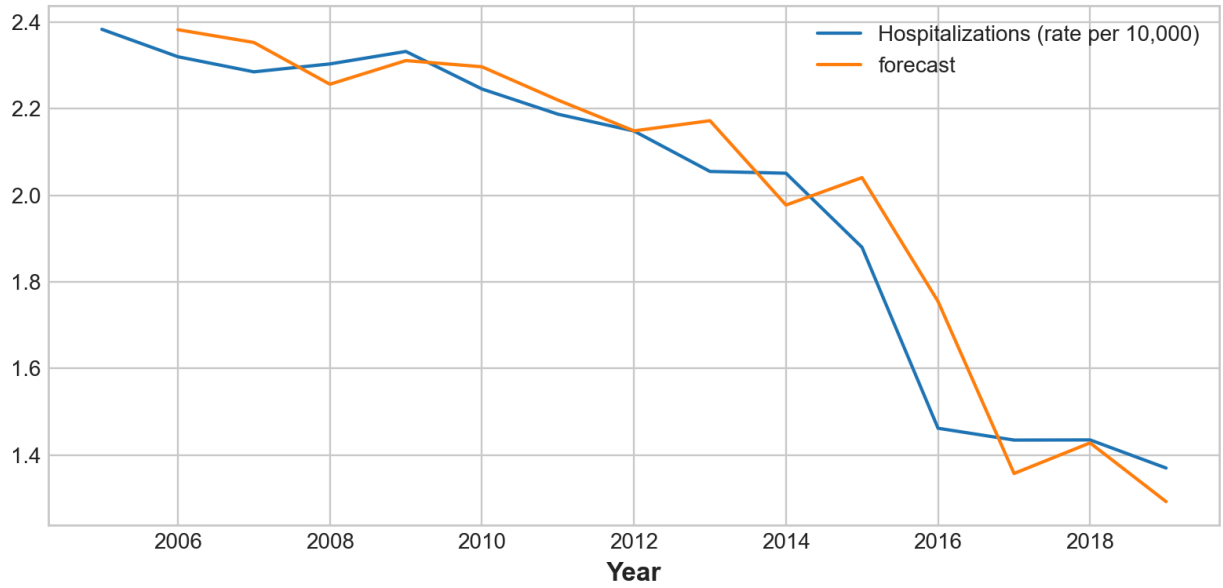
Dep. Variable:	Hospitalizations (rate per 10,000)	**No. Observations:**	15
Model:	SARIMAX(1, 1, 1)x(1, 0, [], 2)	**Log Likelihood**	11.266
Date:	Mon, 29 Apr 2024	**AIC**	-12.531
Time:	19:47:16	**BIC**	-9.336
Sample:	0	**HQIC**	-12.827
	- 15		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
Days PM2.5	0.0034	0.006	0.528	0.597	-0.009	0.016
ar.L1	0.9198	0.347	2.652	0.008	0.240	1.600
ma.L1	-0.5861	0.607	-0.966	0.334	-1.775	0.603
ar.S.L2	-0.3528	0.590	-0.598	0.550	-1.510	0.804
sigma2	0.0111	0.008	1.484	0.138	-0.004	0.026

Ljung-Box (L1) (Q):	0.10	**Jarque-Bera (JB):**	3.24
Prob(Q):	0.75	**Prob(JB):**	0.20
Heteroskedasticity (H):	10.64	**Skew:**	-1.11
Prob(H) (two-sided):	0.02	**Kurtosis:**	3.81

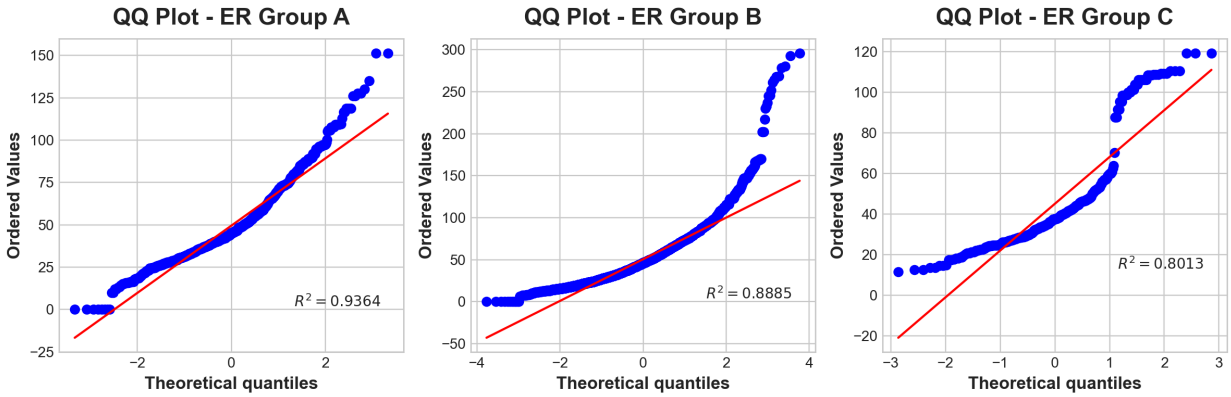
Warnings:

[1] Covariance matrix calculated using the outer product of gradients (complex-step).



QQ plots

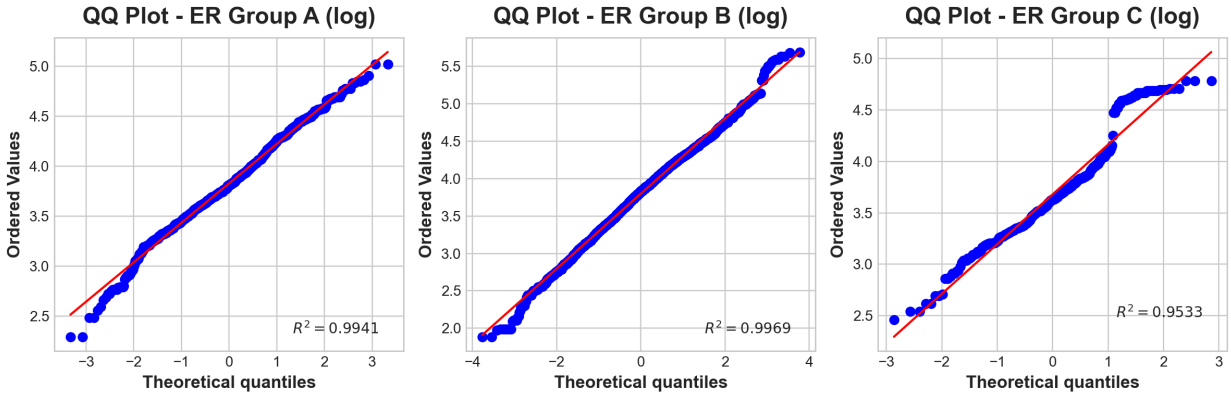
In [ ]:
```
dataA = aqi_erA_adj['ER visits for Asthma (rate per 10,000)']
dataB = aqi_erB_adj['ER visits for Asthma (rate per 10,000)']
dataC = aqi_erC_adj['ER visits for Asthma (rate per 10,000)']


dataB = dataB.replace([np.inf, -np.inf], np.nan).dropna()


# Create a figure and subplots
fig, axes = plt.subplots(1, 3, figsize=(12, 4))

# Plot QQ plots for each dataset
stats.probplot(dataA, dist="norm", plot=axes[0], rvalue=True)
axes[0].set_title('QQ Plot - ER Group A')

stats.probplot(dataB, dist="norm", plot=axes[1], rvalue=True)
axes[1].set_title('QQ Plot - ER Group B')

stats.probplot(dataC, dist="norm", plot=axes[2], rvalue=True)
axes[2].set_title('QQ Plot - ER Group C')

# Adjust spacing between subplots
plt.tight_layout()

# Show the figure
plt.show()
```



In [ ]:
```
transformed_dataA = np.log(dataA)
transformed_dataB = np.log(dataB)
transformed_dataC = np.log(dataC)

print(f'shape A: {transformed_dataA.shape}, shape B: {transformed_dataB.shape}

# Create a figure and subplots
fig, axes = plt.subplots(1, 3, figsize=(12, 4))

transformed_dataA = transformed_dataA.replace([np.inf, -np.inf], np.nan).dropna
transformed_dataB = transformed_dataB.replace([np.inf, -np.inf], np.nan).dropna
transformed_dataC = transformed_dataC.replace([np.inf, -np.inf], np.nan).dropna
print(f'shape A: {transformed_dataA.shape}, shape B: {transformed_dataB.shape}

# Plot QQ plots for each dataset
stats.probplot(transformed_dataA, dist="norm", plot=axes[0], rvalue=True)
axes[0].set_title('QQ Plot - ER Group A (log)')

stats.probplot(transformed_dataB, dist="norm", plot=axes[1], rvalue=True)
```

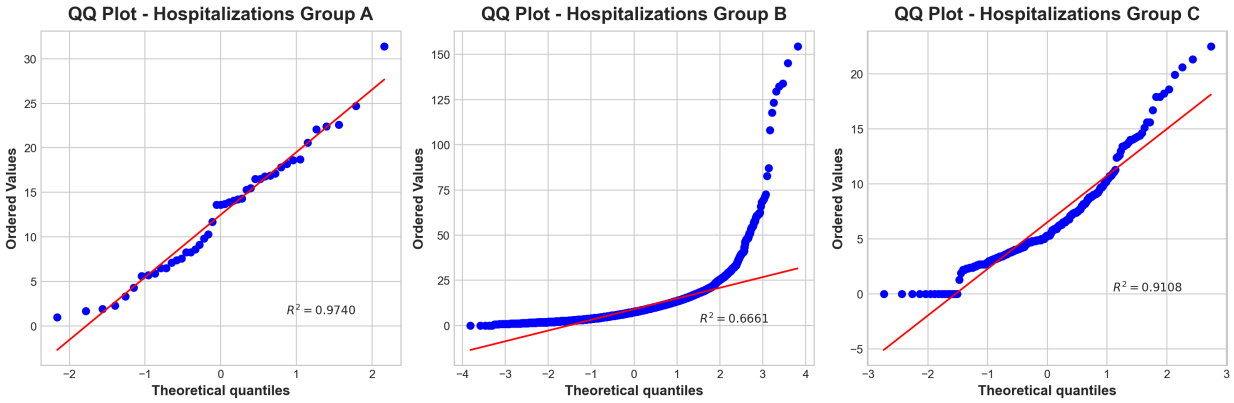
```
axes[1].set_title('QQ Plot - ER Group B (log)')

stats.probplot(transformed_dataC, dist="norm", plot=axes[2], rvalue=True)
axes[2].set_title('QQ Plot - ER Group C (log)')

plt.tight_layout()

# Show the figure
plt.show()
```

```
shape A: (1595,), shape B: (8472,), shape C: (329,)
shape A: (1587,), shape B: (8460,), shape C: (329,)
```



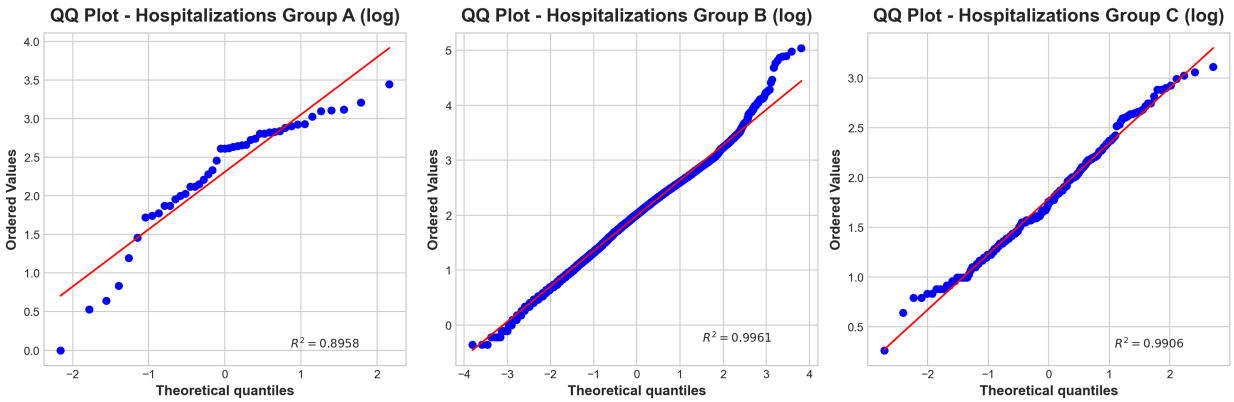
In [ ]:

```
## Hospitalizations

dataA = aqi_hospA['Hospitalizations (rate per 10,000)']
dataB = aqi_hospB['Hospitalizations (rate per 10,000)']
dataC = aqi_hospC['Hospitalizations (rate per 10,000)']

# Create a figure and subplots
fig, axes = plt.subplots(1, 3, figsize=(15,5))

# Plot QQ plots for each dataset
stats.probplot(dataA, dist="norm", plot=axes[0], rvalue=True)
axes[0].set_title('QQ Plot - Hospitalizations Group A')

stats.probplot(dataB, dist="norm", plot=axes[1], rvalue=True)
axes[1].set_title('QQ Plot - Hospitalizations Group B')

stats.probplot(dataC, dist="norm", plot=axes[2], rvalue=True)
axes[2].set_title('QQ Plot - Hospitalizations Group C')

# Adjust spacing between subplots
plt.tight_layout()

# Show the figure
plt.show()
```



In [ ]:
```
transformed_dataA = np.log(dataA)
transformed_dataB = np.log(dataB)
transformed_dataC = np.log(dataC)

print(f'shape A: {transformed_dataA.shape}, shape B: {transformed_dataB.shape}

# Create a figure and subplots
fig, axes = plt.subplots(1, 3, figsize=(15, 5))

transformed_dataA = transformed_dataA.replace([np.inf, -np.inf], np.nan).dropna
transformed_dataB = transformed_dataB.replace([np.inf, -np.inf], np.nan).dropna
transformed_dataC = transformed_dataC.replace([np.inf, -np.inf], np.nan).dropna
print(f'shape A: {transformed_dataA.shape}, shape B: {transformed_dataB.shape}

# Plot QQ plots for each dataset
stats.probplot(transformed_dataA, dist="norm", plot=axes[0], rvalue=True)
axes[0].set_title('QQ Plot - Hospitalizations Group A (log)')

stats.probplot(transformed_dataB, dist="norm", plot=axes[1], rvalue=True)
axes[1].set_title('QQ Plot - Hospitalizations Group B (log)')

stats.probplot(transformed_dataC, dist="norm", plot=axes[2], rvalue=True)
axes[2].set_title('QQ Plot - Hospitalizations Group C (log)')

plt.tight_layout()

# Show the figure
plt.show()
```

```
shape A: (45,), shape B: (10065,), shape C: (225,)
shape A: (45,), shape B: (10060,), shape C: (210,)
```



Conclusions

Random Forest

The Random Forest models demonstrated varying levels of performance across the groups. ER visits group B showed the highest performance with an R^2 score of 0.68. ER visits group A and C also performed relatively well with R^2 scores of 0.65 and 0.63, respectively. These models predict asthma ER visits relatively accurately based on the R^2 scores and RMSE values. For Hospitalizations Group B, the performance was lower compared to groups in ER visits, with an R^2 score of 0.25 and an RMSE of 0.51.

Random Forest Summary			
Group	R^2 Score	MSE	RMSE
ER Visits Group A	0.65	105.84	10.29
ER Visits Group B	0.68	119.77	10.94
ER Visits Group C	0.63	0.08	0.27
Hospitalizations Group B	0.25	0.26	0.51

Table 2: Performance metrics for Random Forest models

SARIMAX

The SARIMAX models were evaluated using various diagnostic statistics to assess their reliability. Notably, the Ljung-Box test indicated minimal autocorrelation in the residuals, with statistics ranging from 0.02 to 0.56 across models for ER visits and hospitalizations. Importantly, the associated p-values were above 0.05, indicating no significant autocorrelation. Furthermore, the Jarque-Bera test confirmed the normality of the residuals, with high p-values across all models, ranging from 0.60 to 3.24. However, regarding heteroskedasticity, the hospitalization group B model has considerable variability, with a statistically significant value of 10.64 in the heteroskedasticity test, while other models range from 0.22 to 4.01.

	ER Visits (A)	ER Visits (B)	ER Visits (C)	Hosp (B)
Ljung-Box (L1) (Q)	0.02	0.56	0.07	0.10
Prob(Q)	0.90	0.45	0.79	0.75
Heteroskedasticity (H)	0.22	4.01	3.06	10.64
Prob(H) (two-sided)	0.22	0.15	0.25	0.02
Jarque-Bera (JB)	0.60	1.01	0.78	3.24
Prob(JB)	0.74	0.60	0.68	0.20
Skew	0.44	-0.65	-0.02	-1.11
Kurtosis	2.51	3.11	1.84	3.81

Table 3: Performance metrics for SARIMAX Models