Air Quality and Health Impact (Pranav Chauhan)

Using Random Forest Regression to predict the Health Impact Score based on air quality and pollution data.

Importing libraries

In [25]:
```
import seaborn as sns
import pandas as pd
import matplotlib.pyplot as plt


import sqlite3

from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score
```

Reading dataset

In [26]:
```
data = pd.read_csv("projectdata.csv")
```

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

Out[27]:

	RecordID	AQI	PM10	PM2_5	NO2	SO2	O3	Temperature	Humidity	Wind
0	1	187.270059	295.853039	13.038560	6.639263	66.161150	54.624280	5.150335	84.424344	6.1
1	2	475.357153	246.254703	9.984497	16.318326	90.499523	169.621728	1.543378	46.851415	4.5
2	3	365.996971	84.443191	23.111340	96.317811	17.875850	9.006794	1.169483	17.806977	11.1
3	4	299.329242	21.020609	14.273403	81.234403	48.323616	93.161033	21.925276	99.473373	15.3
4	5	78.009320	16.987667	152.111623	121.235461	90.866167	241.795138	9.217517	24.906837	14.5

In [28]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5811 entries, 0 to 5810
Data columns (total 14 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   RecordID             5811 non-null   int64  
 1   AQI                  5808 non-null   float64
 2   PM10                 5811 non-null   float64
 3   PM2_5                5811 non-null   float64
 4   NO2                  5811 non-null   float64
 5   SO2                  5808 non-null   float64
 6   O3                   5810 non-null   float64
 7   Temperature          5809 non-null   float64
 8   Humidity             5810 non-null   float64
 9   WindSpeed            5811 non-null   float64
 10  RespiratoryCases     5806 non-null   float64
 11  CardiovascularCases  5811 non-null   int64  
 12  HospitalAdmissions   5811 non-null   int64  
 13  HealthImpactScore    5811 non-null   float64
dtypes: float64(11), int64(3)
memory usage: 635.7 KB
```

Data Cleaning

In [29]:
```
#  Check for null values in each column

null_values = data.isnull().sum()

# Display the result:

print(f"Null values in each column:\n\n{null_values}")
```

```
Null values in each column:

RecordID              0
AQI                   3
PM10                  0
PM2_5                 0
NO2                   0
SO2                   3
O3                    1
Temperature           2
Humidity              1
WindSpeed             0
RespiratoryCases      5
CardiovascularCases   0
HospitalAdmissions    0
HealthImpactScore     0
dtype: int64
```

In [30]:
```
# replacing empty values with median

data = data.fillna(data.median())

#  Check for null values in each column

null_values = data.isnull().sum()

# Display the result:

print(f"Null values in each column:\n\n{null_values}")
```

```
Null values in each column:

RecordID               0
AQI                    0
PM10                   0
PM2_5                  0
NO2                    0
SO2                    0
O3                     0
Temperature            0
Humidity               0
WindSpeed              0
RespiratoryCases       0
CardiovascularCases    0
HospitalAdmissions     0
HealthImpactScore      0
dtype: int64
```

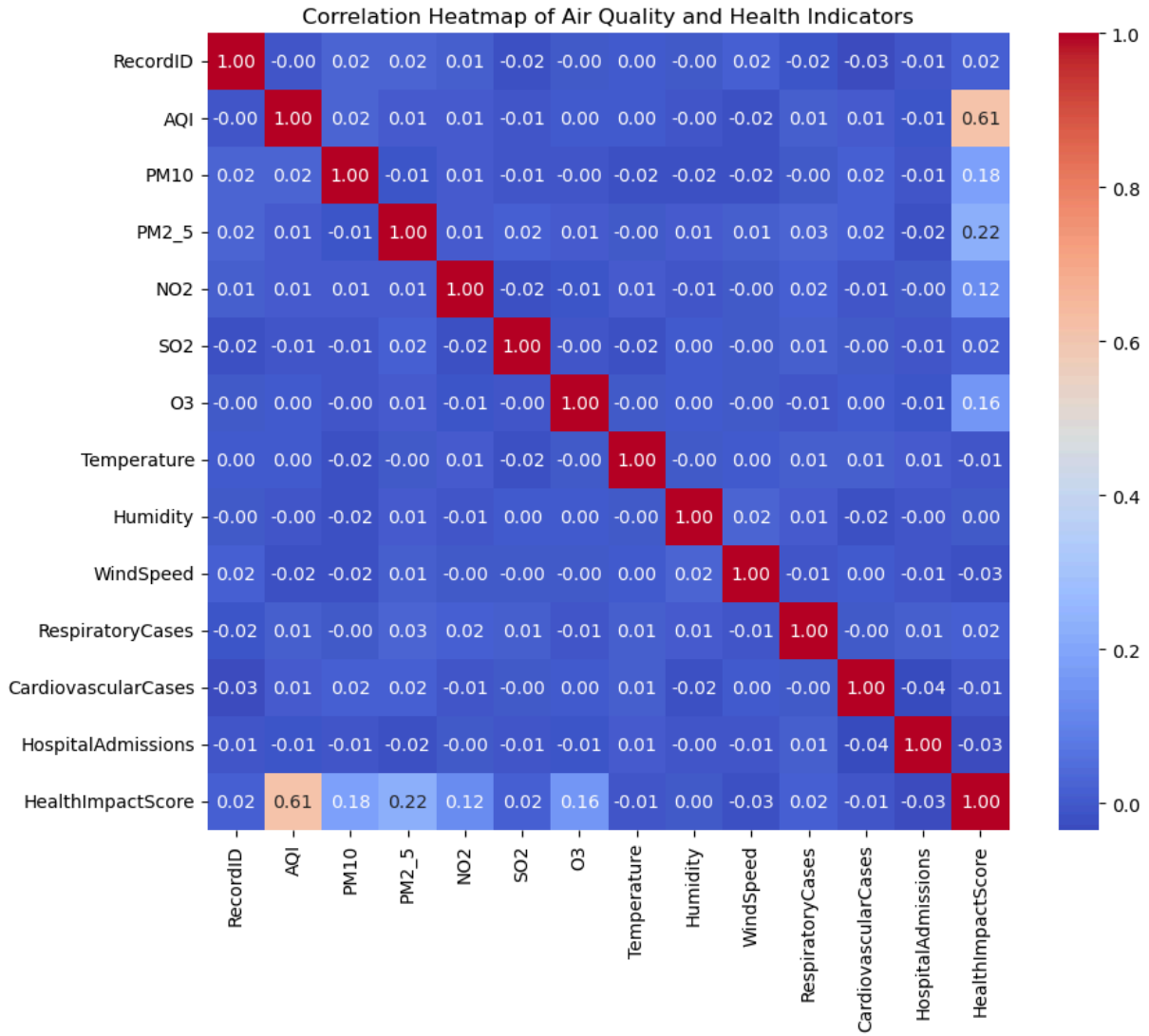
In [31]:
```
# print(data.dtypes)

data['AQI'] = pd.to_numeric(data['AQI'], errors='coerce')
print()

print(data.dtypes)
```

```
RecordID                 int64
AQI                    float64
PM10                   float64
PM2_5                  float64
NO2                    float64
SO2                    float64
O3                     float64
Temperature            float64
Humidity               float64
WindSpeed              float64
RespiratoryCases       float64
CardiovascularCases      int64
HospitalAdmissions       int64
HealthImpactScore      float64
dtype: object
```

Ploting Correaltion Heatmap

In [32]:

```
correlation_matrix = data.corr()

# Plot the heatmap

plt.figure(figsize=(10, 8))

sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")

plt.title("Correlation Heatmap of Air Quality and Health Indicators")
plt.show()
```



Droping the unnecessary columns

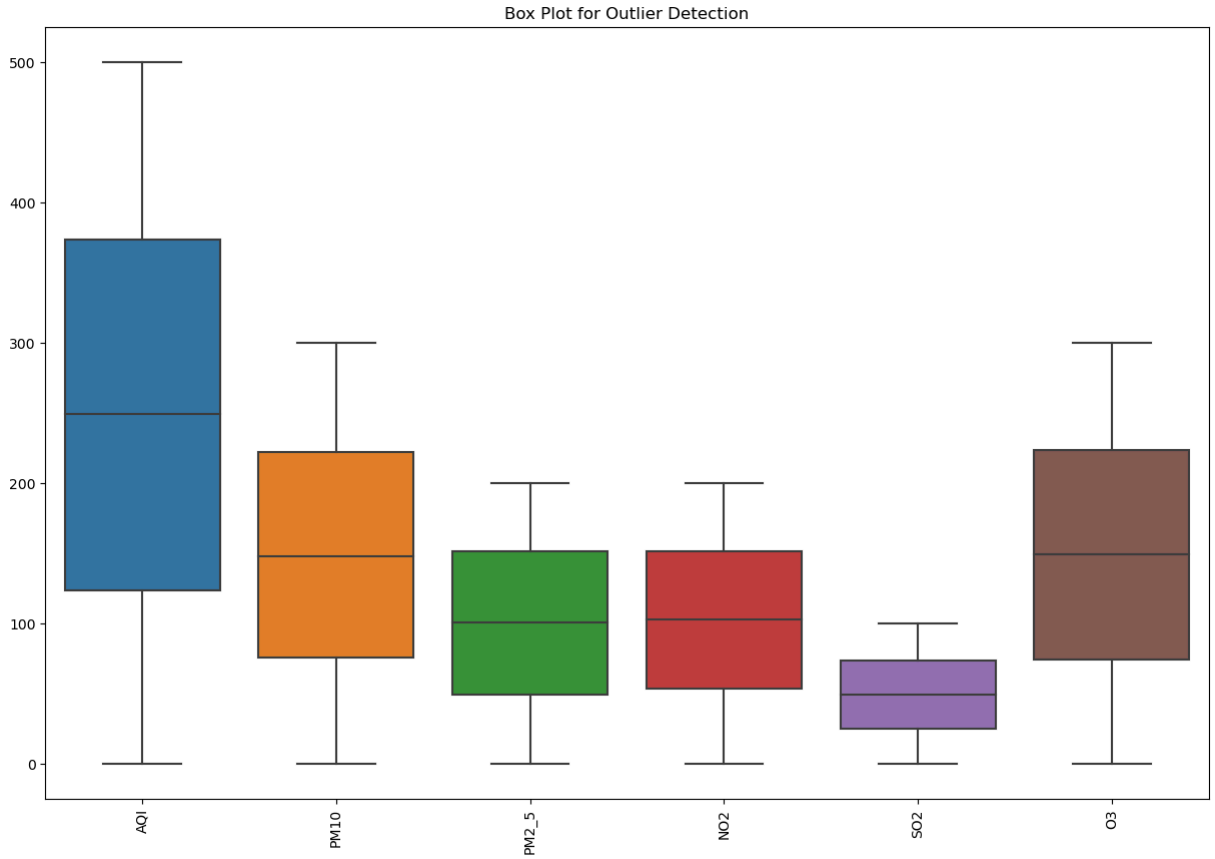
In [33]:

```
# Drop the columns


data = data.drop(columns=['Humidity'])
```

Boxplot for outliers detection

In [34]:

```
class graph:

    def create_boxplot(data, figsize=(15, 10), title="Box Plot for Outlier Detection", rotati
        data = data[['AQI','PM10','PM2_5','NO2','SO2','O3']]
        plt.figure(figsize=figsize)
        sns.boxplot(data=data)
        plt.xticks(rotation=rotation)
        plt.title(title)
        plt.show()


graph.create_boxplot(data)
```



Ploting graphs

In [35]:

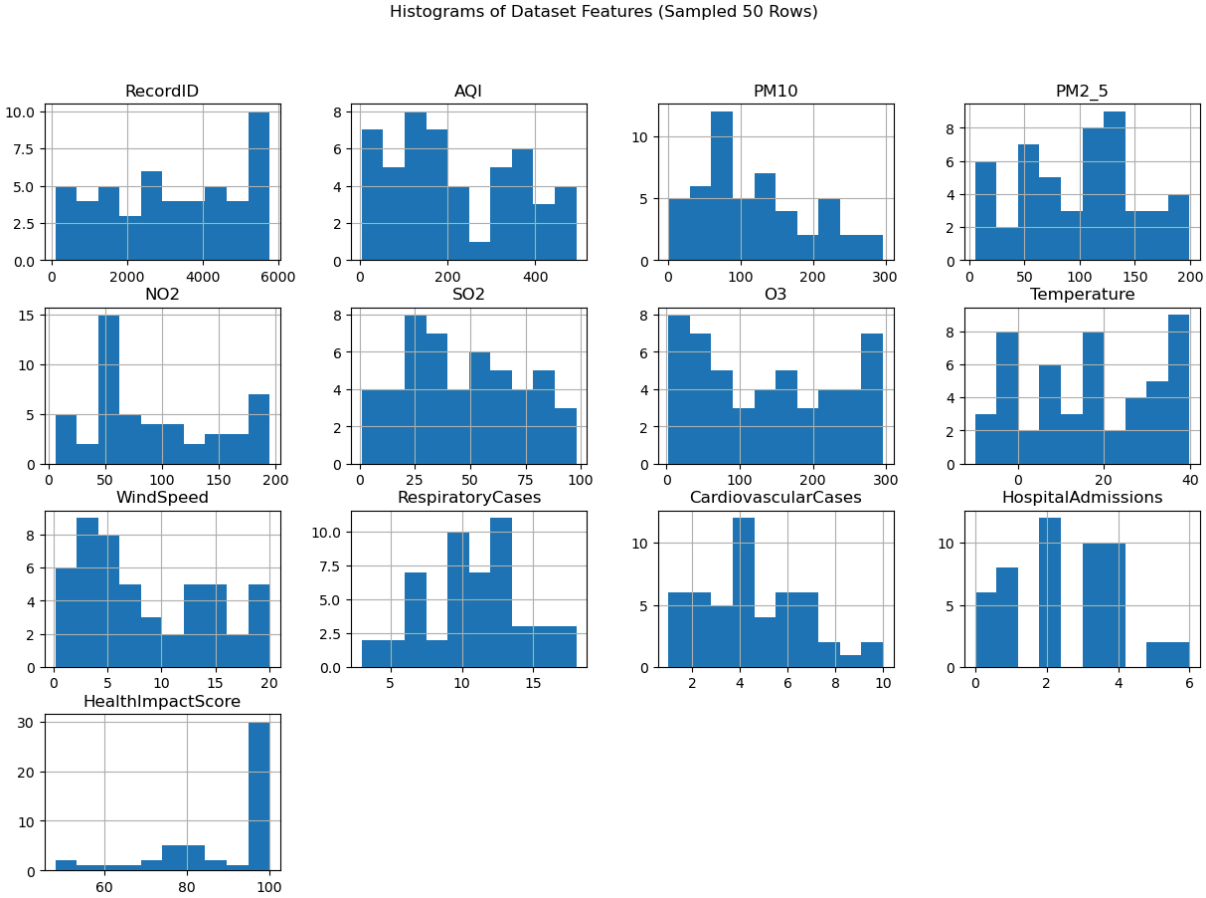
```
class Graph:

    def create_histogram(self, data):

        data_subset = data.sample(n=50, random_state=42)
        data_subset.hist(figsize=(15, 10))
        plt.suptitle('Histograms of Dataset Features (Sampled 50 Rows)')
        plt.show()

    def create_pairplot(self, data):

        data_subset = data.sample(n=20, random_state=42)
        sns.pairplot(data_subset)
        plt.show()


    def create_piechart(self, data):
        data_subset = data.sample(n=50, random_state=42)
        pollutants_sum = data_subset[['PM10', 'PM2_5', 'NO2', 'SO2']].sum()

        plt.figure(figsize=(8, 8))
        plt.pie(pollutants_sum, labels=pollutants_sum.index, autopct='%1.1f%%', startangle=14
        plt.title('Proportion of Pollutants')
        plt.show()

    def stacked_barchart(self,data):

        data_subset = data.sample(n=10)
        data_subset[['PM10', 'PM2_5', 'NO2', 'SO2']].plot(kind='bar', stacked=True, figsize=(
        plt.title('Pollutant Composition by Record ID')
        plt.xlabel('Record ID')
        plt.ylabel('Pollutant Level')
        plt.show()

    def scatterplot(self,data):
        data_subset = data.sample(n=50, random_state=42)
        plt.scatter(data_subset['AQI'], data_subset['RespiratoryCases'], alpha=0.5)
        plt.xlabel('AQI')
        plt.ylabel('Respiratory Cases')
        plt.title('AQI vs Respiratory Cases')
        plt.show()

    def aqi_histogram(self,data):
        data_subset = data.sample(n=50, random_state=42)
        plt.hist(data_subset['AQI'], bins=20, edgecolor='black')
        plt.xlabel('AQI')
        plt.ylabel('Frequency')
        plt.title('Distribution of AQI')
        plt.show()
```

In [36]:

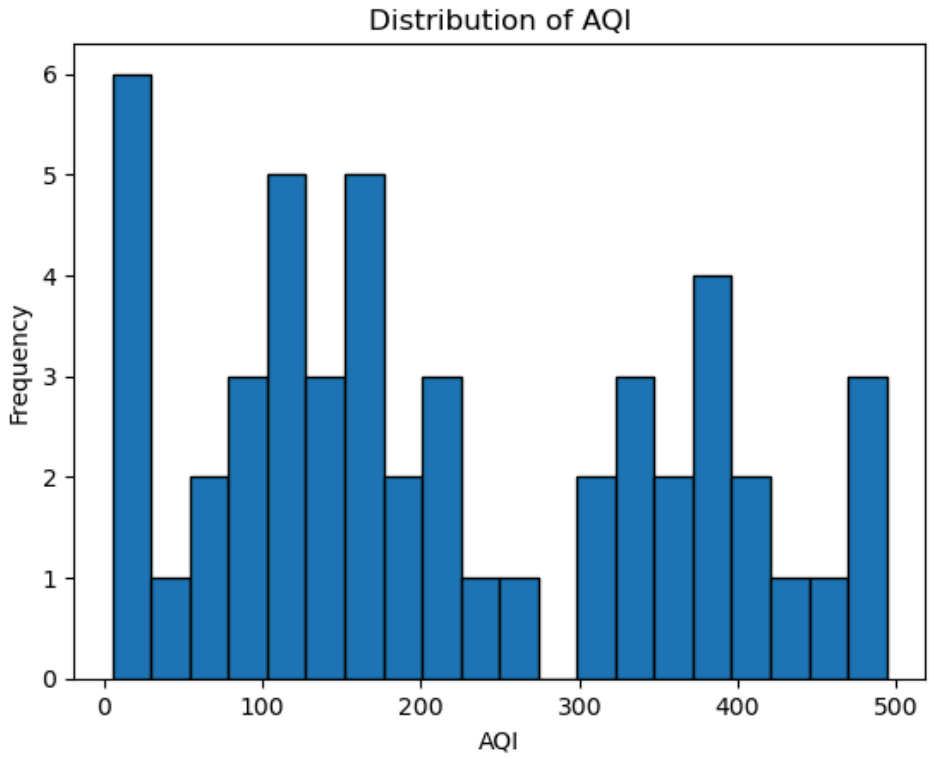
```
graph = Graph()
```

Histogram for each column

In [37]:
```
graph.create_histogram(data)
```

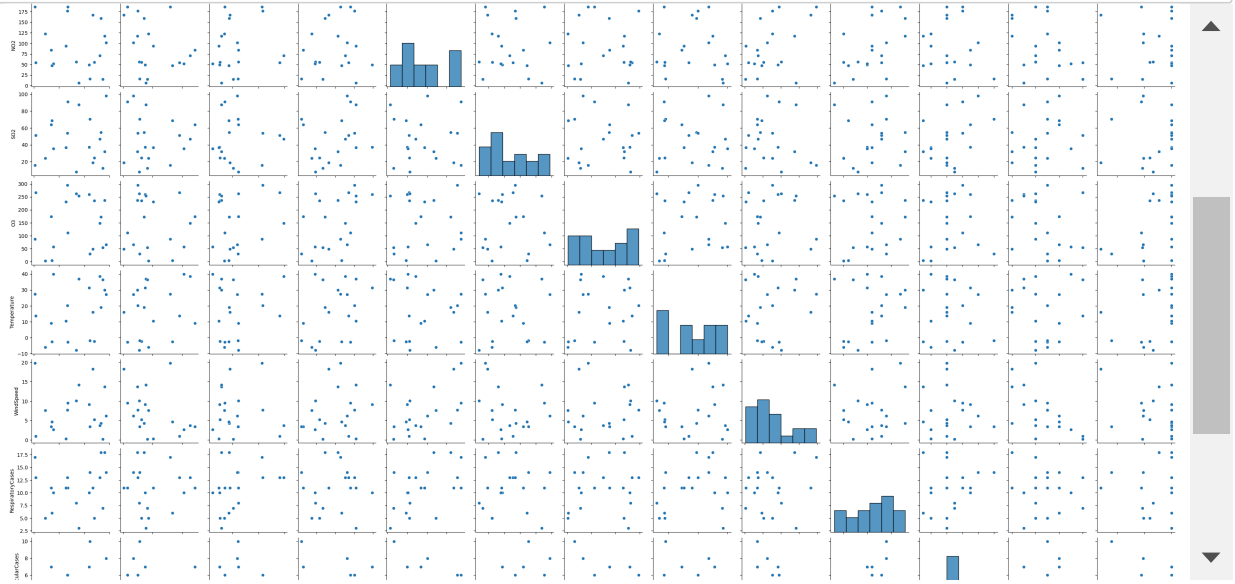


In [38]:
```
graph.aqi_histogram(data)
```



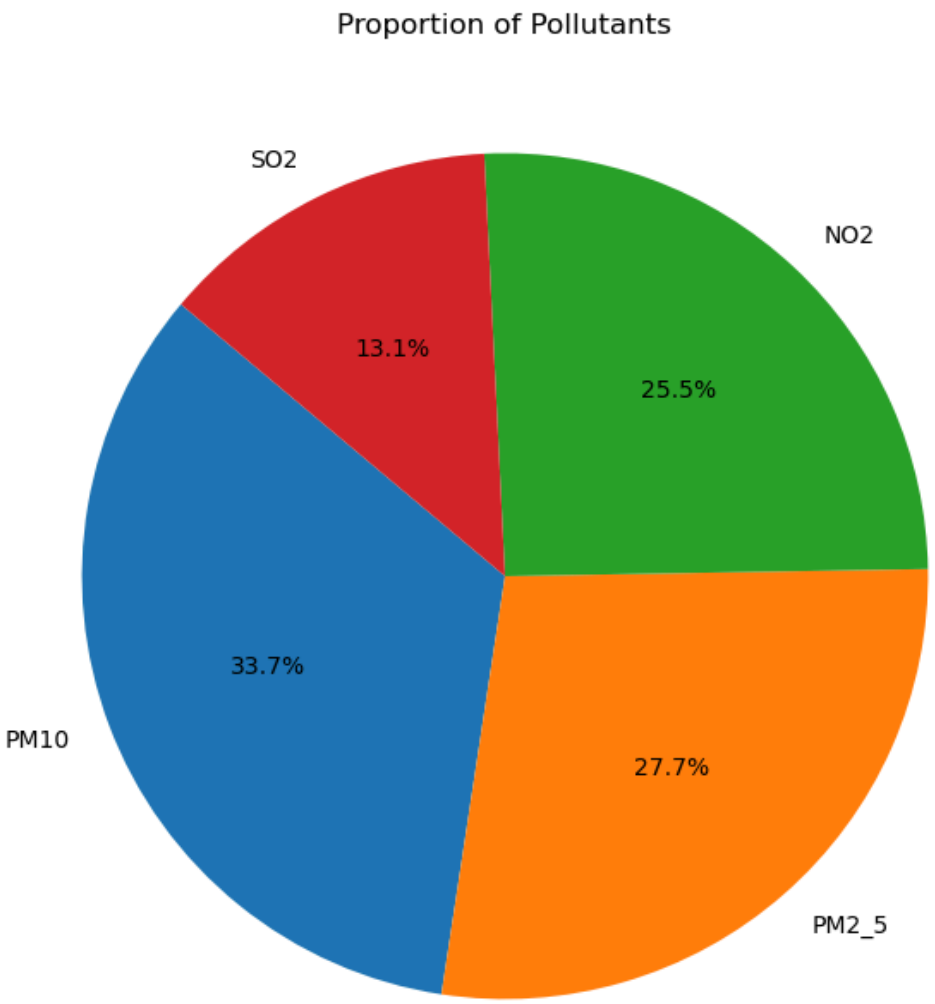
Pairplot

In [39]:
```
graph.create_pairplot(data)
```



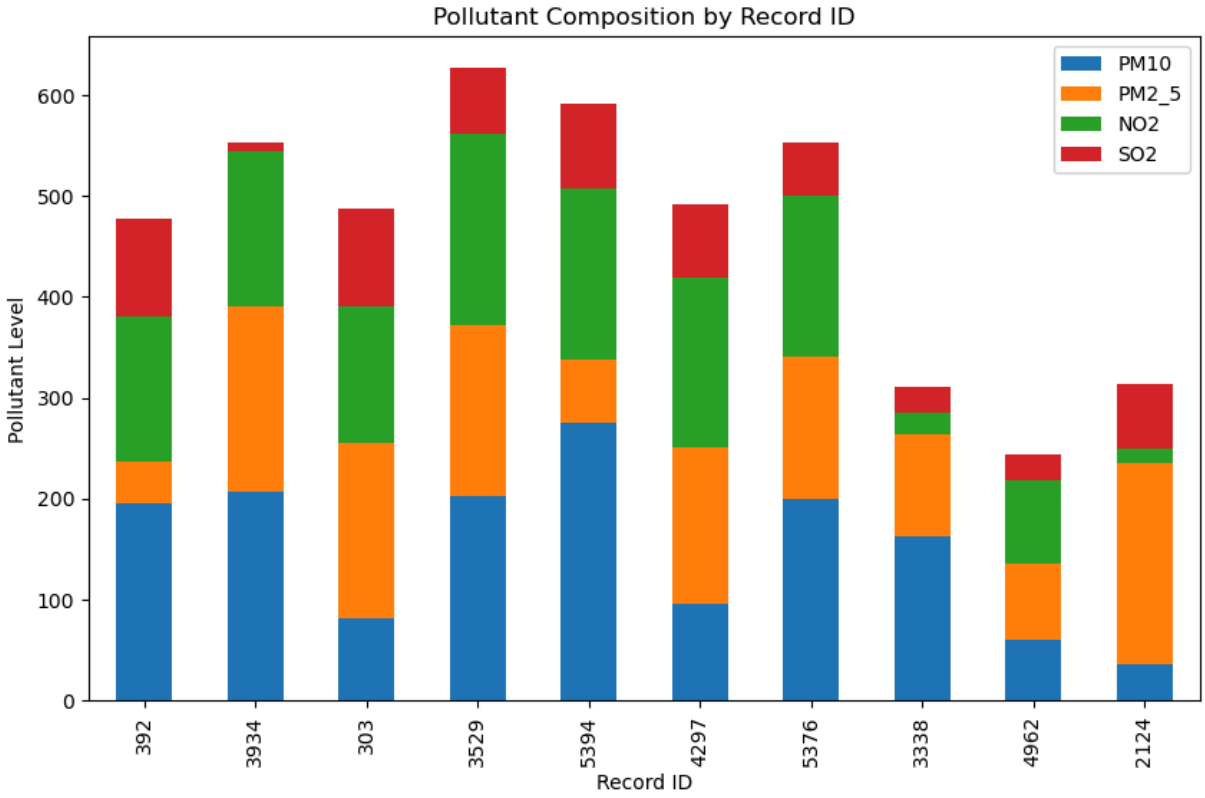
Pie Chart

In [40]:
```
graph.create_piechart(data)
```



Stacked Bar Chart

In [41]:
```
graph.stacked_barchart(data)
```



Scatter Plot

In [42]:
```
graph.scatterplot(data)
```

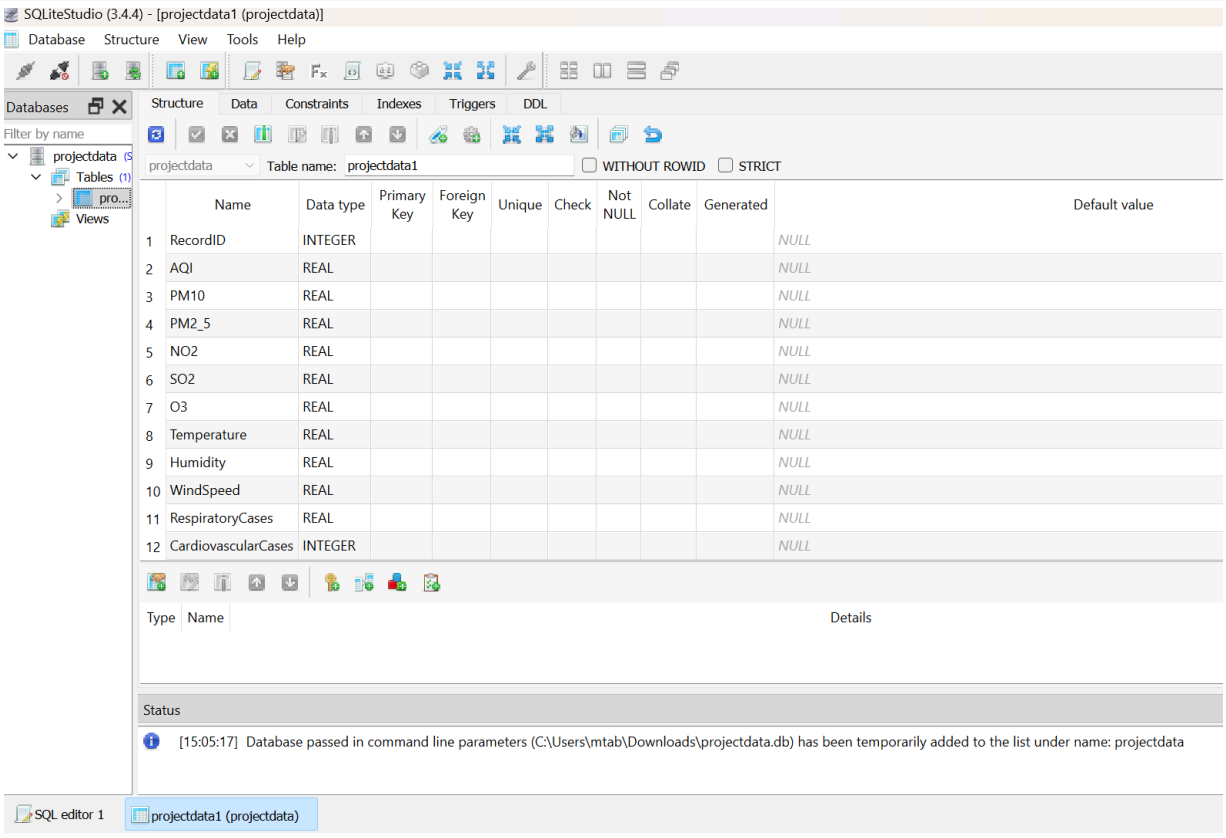
SQL Connectivity

In [86]:
```
try:

    conn = sqlite3.connect("projectdata.db")
    print("database created!")

    cursor = conn.cursor()

    df = pd.read_csv("projectdata.csv")

    df.to_sql("projectdata1", conn, if_exists='replace', index=False)
    print("table Created !")

except sqlite3.Error as error:
    print("Error is:", error)

finally:
    if conn:
        conn.close()
```

```
database created!
table Created !
```

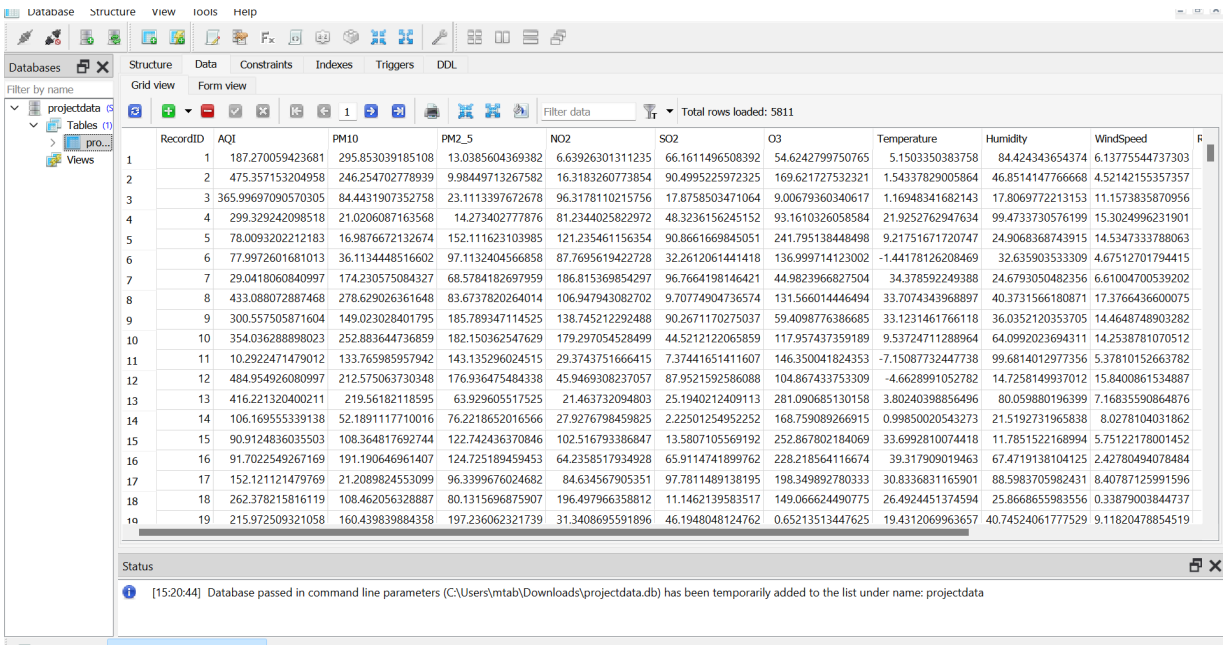
In [7]:
```
from IPython.display import Image

# Display the image
Image(filename='screenshott.png')
```

Out[7]:



In [8]:
```
Image(filename='screenshott_1.png')
```

Out[8]:



	RecordID	AQI	PM10	PM2_5	NO2	SO2	O3	Temperature	Humidity	WindSpeed
1	1	187.270059423681	295.853039185108	13.0385604369382	6.63926301311235	66.1611496508392	54.6242799750765	5.1503350383758	84.424343654374	6.13775544737303
2	2	475.357153204958	246.254702778939	9.98449713267582	16.3183260773854	90.4995225972325	169.621727532321	1.54337829005864	46.8514147766668	4.52142155357357
3	3	365.99697090570305	84.4431907352758	23.1113397672678	96.3178110215756	17.8758503471064	9.00679360340617	1.16948341682143	17.8069772213153	11.1573835870956
4	4	299.329242098518	21.0206087163568	14.273402777876	81.2344025822972	48.3236156245152	93.1610326058584	21.9252762947634	99.4733730576199	15.3024996231901
5	5	78.0093202212183	16.9876672132674	152.111623103985	121.235461156354	90.8661669845051	241.795138448498	9.21751671720747	24.9068368743915	14.5347333788063
6	6	77.9972601681013	36.1134448516602	97.1132404566858	87.7695619422728	32.2612061441418	136.999714123002	-1.44178126208469	32.6359033533309	4.67512701794415
7	7	29.0418060840997	174.230575084327	68.5784182697959	186.815369854297	96.7664198146421	44.9823966827504	34.378592249388	24.6793050482356	6.61004700539202
8	8	433.088072887468	278.629026361648	83.6737820264014	106.947943082702	9.70774904736574	131.566014446494	33.7074343968897	40.3731566180871	17.3766436600075
9	9	300.557505871604	149.023028401795	185.789347114525	138.745212292488	90.2671170275037	59.4098776386685	33.1231461766118	36.0352120353705	14.4648748903282
10	10	354.036288898023	252.883644736859	182.150362547629	179.297054528499	44.5212122065859	117.957437359189	9.53724711288964	64.0992023694311	14.2538781070512
11	11	10.2922471479012	133.765985957942	143.135296024515	29.3743751666415	7.37441651411607	146.350041824353	-7.15087732447738	99.6814012977356	5.37810152663782
12	12	484.954926080997	212.575063730348	176.936475484338	45.9469308237057	87.9521592586088	104.867433753309	-4.6628991052782	14.7258149937012	15.8400861534887
13	13	416.221320400211	219.56182118595	63.929605517525	21.463732094803	25.1940212409113	281.090685130158	3.80240398856496	80.059880196399	7.16835590864876
14	14	106.169555339138	52.1891117710016	76.2218652016566	27.9276798459825	2.22501254952252	168.759089266915	0.99850020543273	21.5192731965838	8.0278104031862
15	15	90.9124836035503	108.364817692744	122.742436370846	102.516793386847	13.5807105569192	252.867802184069	33.6992810074418	11.7851522168994	5.75122178001452
16	16	91.7022549267169	191.190646961407	124.725189459453	64.2358517934928	65.9114741899762	228.218564116674	39.317909019463	67.4719138104125	2.42780494078484
17	17	152.121121479769	21.2089824553099	96.3399676024682	84.634567905351	97.7811489138195	198.349892780333	30.8336831165901	88.5983705982431	8.40787125991596
18	18	262.378215816119	108.462056328887	80.1315696875907	196.497966358812	11.1462139583517	149.066624490775	26.4924451374594	25.8668655983556	0.33879003844737
19	19	215.972509321058	160.439839884358	197.236062321739	31.3408695591896	46.1948048124762	0.65213513447625	19.4312069963657	40.74524061777529	9.11820478854519

Spliting data into train and test data

In [43]:
```
x = data.drop(columns=['HealthImpactScore'])

y = data['HealthImpactScore']

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42)

print("Training data shape:", x_train.shape)
print("Testing data shape:", x_test.shape)

# print(x_train.dtypes)
```

```
Training data shape: (4648, 12)
Testing data shape: (1163, 12)
```

Using Random Forest Regression Model

In [44]:
```
# Initialize and train the model
rf_model = RandomForestRegressor()
rf_model.fit(x_train, y_train)

# Make predictions
y_pred = rf_model.predict(x_test)
```

Predicting for custom input

In [45]:

```
new_data = {
    'RecordID' : 889,
    'AQI': 85,
    'PM10': 55.0,
    'PM2_5': 35.2,
    'NO2': 20.3,
    'SO2': 12.5,
    'O3': 40.0,
    'Temperature': 25.3,
    'WindSpeed': 5.5,
    'RespiratoryCases': 10,
    'CardiovascularCases': 8,
    'HospitalAdmissions': 2
}


new_data = pd.DataFrame([new_data])

# Make prediction
prediction = rf_model.predict(new_data)

# Display the result
print(f"Predicted Health Impact Score: {prediction}")
```

```
Predicted Health Impact Score: [54.58507219]
```

Calculating Mean Sqaured Error and R2 score

In [46]:

```


# Calculate Mean Squared Error (MSE)
mse = mean_squared_error(y_test, y_pred)
print(f'Mean Squared Error: {mse}')

# Calculate R-squared (R²)
r2 = r2_score(y_test, y_pred)
print(f'R-squared: {r2}')
```

```
Mean Squared Error: 10.003289135006993
R-squared: 0.9466109674434999
```