This project employs a data-driven methodology to analyse Delhi's Air Quality Index (AQI) using Python. In this project, time-series approaches have been employed to forecast air quality, and it’s effectiveness has been assessed using a variety of metrics. The dataset includes hourly AQI readings, pollutant concentrations (PM2.5, PM10, NO_2, CO, SO_2, and O_3), and timestamps. The project provides a comprehensive understanding of the dynamics of the city's air quality by combining a number of Python modules and tools for data collection, preprocessing, analysis, and visualisation.

Dataset

The dataset consists of timestamped AQI values and pollutant concentrations recorded throughout January 2025. It includes key parameters such as pollutant levels, AQI category, and daily variations. The data will be processed using Pandas for cleaning, transformation, and analysis, allowing for the identification of high-risk pollution periods and potential improvements in air quality over time.

Objective:

- Data on pollutant concentrations should be used to calculate and analyse the Air Quality Index (AQI).
- Analyse the AQI category distribution to determine the frequency of various air quality conditions.
- Examine the hourly trends in the AQI to find trends and times when pollution is at its highest.
- Examine correlations among various contaminants to evaluate their interrelationships.
- To assess the severity of Delhi's air quality, compare the recommended air quality measures with the derived AQI metrics.

AQI calculation

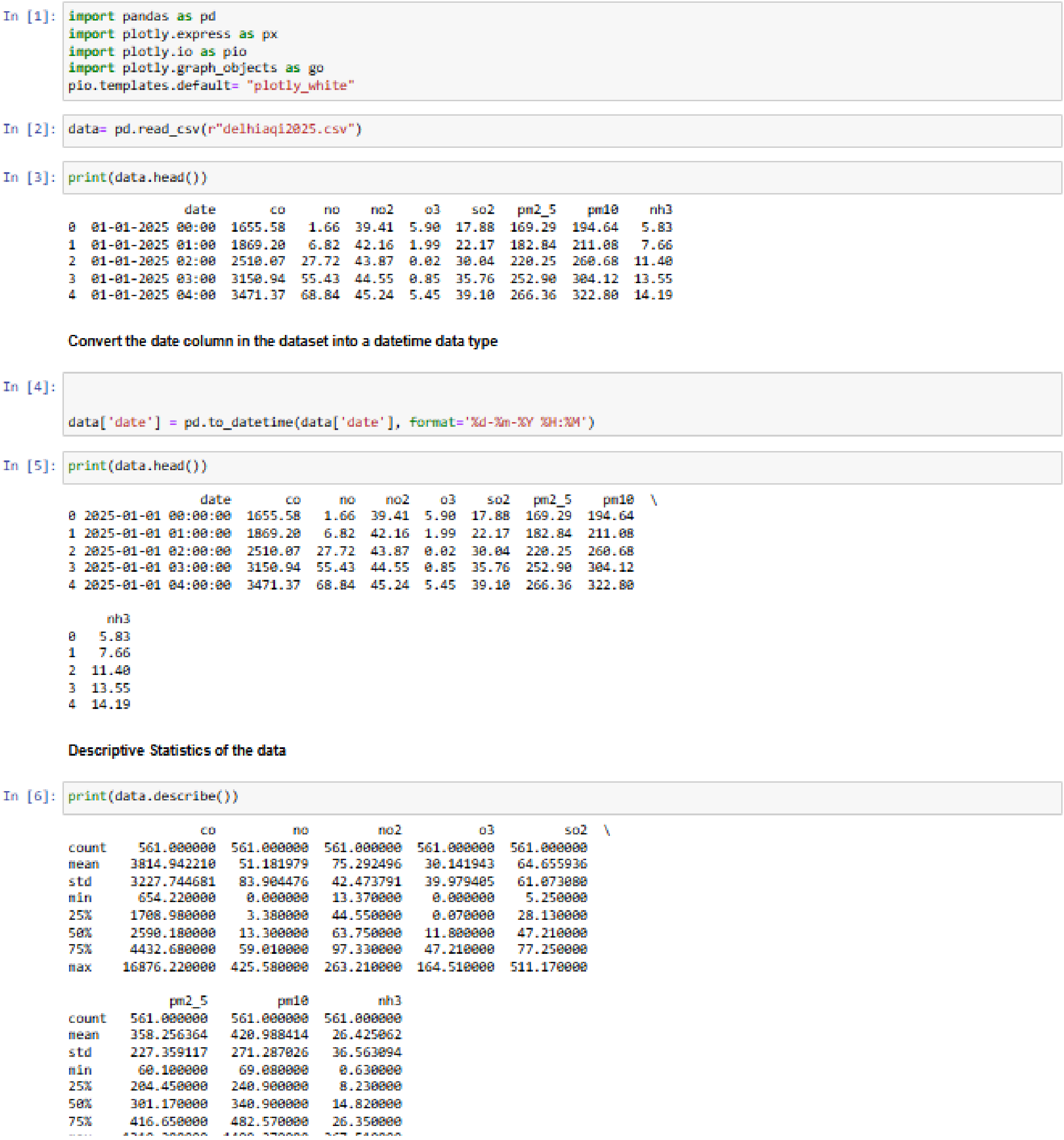
The purpose of the Air Quality Index (AQI) is to inform the public about the current and projected levels of air pollution. Research findings suggest that the Air Quality Index (AQI) forecast serves as a valuable instrument for raising public awareness regarding air quality. The AQI shows the state of the air and its impact on health. The following equation was used in the study to convert concentration measurements ($\mu g/Ncm$) to AQI values.

$$I_P = \frac{I_{Hi} - I_{Lo}}{BP_{Hi} - BP_{Lo}}(C_P - BP_{Lo}) + I_{Lo} \qquad (1)$$

Where Ip is AQI value for the pollutant, Cp is Pollutant concentration, $BPHi$ *is Breakpoint* ≥ Cp, $BPLo$ *is Breakpoint* ≤ Cp, IHi *is AQI value corresponding to* $BPHi$, ILo is *AQI value corresponding to* $BPLo$.

Results and Discussion

Data analysis on Delhi's Air Quality Index (AQI) has produced insightful findings about the dynamics of the city's air quality.



```
In [1]: import pandas as pd
        import plotly.express as px
        import plotly.io as pio
        import plotly.graph_objects as go
        pio.templates.default= "plotly_white"
```

```
In [2]: data= pd.read_csv(r"delhiaqi2025.csv")
```

```
In [3]: print(data.head())
```

```
               date       co     no    no2    o3    so2   pm2_5    pm10    nh3
0  01-01-2025 00:00  1655.58   1.66  39.41  5.90  17.88  169.29  194.64   5.83
1  01-01-2025 01:00  1869.20   6.82  42.16  1.99  22.17  182.84  211.08   7.66
2  01-01-2025 02:00  2510.07  27.72  43.87  0.02  30.04  220.25  260.68  11.40
3  01-01-2025 03:00  3150.94  55.43  44.55  0.85  35.76  252.90  304.12  13.55
4  01-01-2025 04:00  3471.37  68.84  45.24  5.45  39.10  266.36  322.80  14.19
```

Convert the date column in the dataset into a datetime data type

```
In [4]: 

        data['date'] = pd.to_datetime(data['date'], format='%d-%m-%Y %H:%M')
```

```
In [5]: print(data.head())
```

```
                 date       co     no    no2    o3    so2   pm2_5    pm10  \
0 2025-01-01 00:00:00  1655.58   1.66  39.41  5.90  17.88  169.29  194.64   
1 2025-01-01 01:00:00  1869.20   6.82  42.16  1.99  22.17  182.84  211.08   
2 2025-01-01 02:00:00  2510.07  27.72  43.87  0.02  30.04  220.25  260.68   
3 2025-01-01 03:00:00  3150.94  55.43  44.55  0.85  35.76  252.90  304.12   
4 2025-01-01 04:00:00  3471.37  68.84  45.24  5.45  39.10  266.36  322.80   

     nh3  
0   5.83  
1   7.66  
2  11.40  
3  13.55  
4  14.19  
```

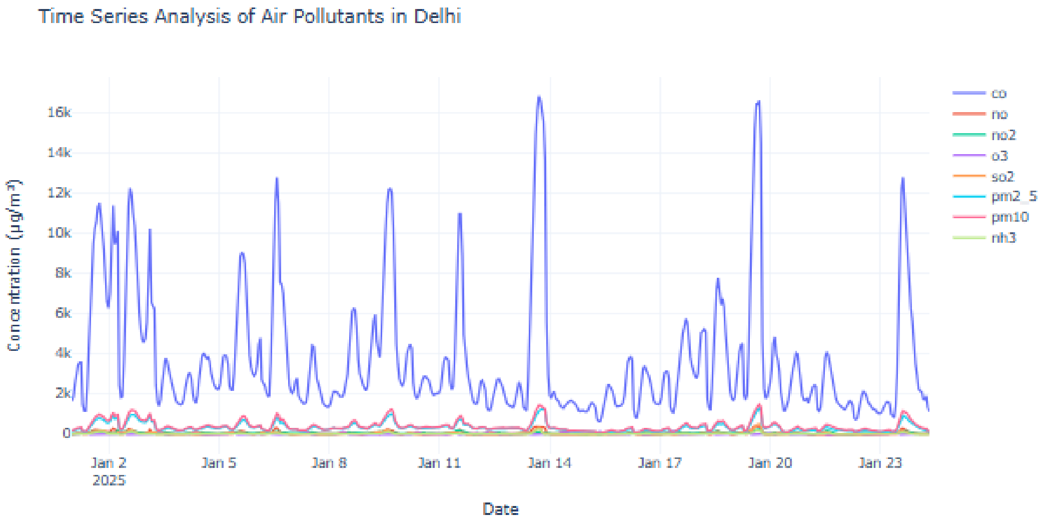
Descriptive Statistics of the data

```
In [6]: print(data.describe())
```

```
                 co          no         no2          o3         so2  \
count    561.000000  561.000000  561.000000  561.000000  561.000000   
mean    3814.942210   51.181979   75.292496   30.141943   64.655936   
std     3227.744681   83.904476   42.473791   39.979405   61.073080   
min      654.220000    0.000000   13.370000    0.000000    5.250000   
25%     1708.980000    3.380000   44.550000    0.070000   28.130000   
50%     2590.180000   13.300000   63.750000   11.800000   47.210000   
75%     4432.680000   59.010000   97.330000   47.210000   77.250000   
max    16876.220000  425.580000  263.210000  164.510000  511.170000   

            pm2_5         pm10         nh3  
count  561.000000   561.000000  561.000000  
mean   358.256364   420.988414   26.425062  
std    227.359117   271.287026   36.563094  
min     60.100000    69.080000    0.630000  
25%    204.450000   240.900000    8.230000  
50%    301.170000   340.900000   14.820000  
75%    416.650000   482.570000   26.350000  
```

Time Series plot for each pollutant

```
In [7]: fig = go.Figure()

        for pollutant in ['co', 'no', 'no2', 'o3', 'so2', 'pm2_5', 'pm10', 'nh3']:
            fig.add_trace(go.Scatter(x=data['date'], y=data[pollutant], mode='lines',
                                     name=pollutant))

        fig.update_layout(title='Time Series Analysis of Air Pollutants in Delhi',
                          xaxis_title='Date', yaxis_title='Concentration (µg/m³)')
        fig.show()
```



In the above code, we are creating a time series plot for each air pollutant in the dataset. It helps analyze the intensity of air pollutants over time

Calculation of Air Quality Index

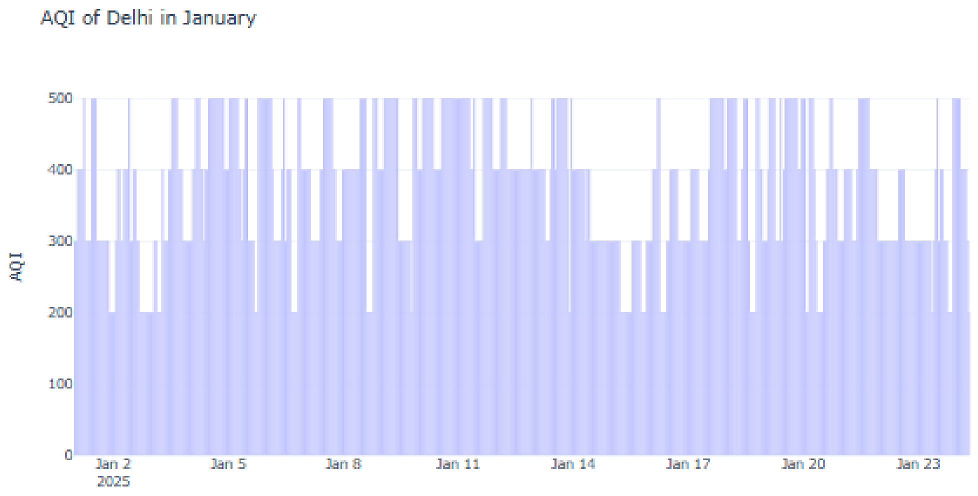
```
In [8]: aqi_breakpoints = [
            (0, 12.0, 50), (12.1, 35.4, 100), (35.5, 55.4, 150),
            (55.5, 150.4, 200), (150.5, 250.4, 300), (250.5, 350.4, 400),
            (350.5, 500.4, 500)
        ]

        def calculate_aqi(pollutant_name, concentration):
            for low, high, aqi in aqi_breakpoints:
                if low <= concentration <= high:
                    return aqi
            return None

        def calculate_overall_aqi(row):
            aqi_values = []
            pollutants = ['co', 'no', 'no2', 'o3', 'so2', 'pm2_5', 'pm10', 'nh3']
            for pollutant in pollutants:
                aqi = calculate_aqi(pollutant, row[pollutant])
                if aqi is not None:
                    aqi_values.append(aqi)
            return max(aqi_values)
        if aqi is not None:
            aqi_values.append(aqi)
    return max(aqi_values)

# Calculate AQI for each row
data['AQI'] = data.apply(calculate_overall_aqi, axis=1)

# Define AQI categories
aqi_categories = [
    (0, 50, 'Good'), (51, 100, 'Moderate'), (101, 150, 'Unhealthy for Sensitive Groups'),
    (151, 200, 'Unhealthy'), (201, 300, 'Very Unhealthy'), (301, 500, 'Hazardous')
]

def categorize_aqi(aqi_value):
    for low, high, category in aqi_categories:
        if low <= aqi_value <= high:
            return category
    return None

# Categorize AQI
data['AQI Category'] = data['AQI'].apply(categorize_aqi)
print(data.head())
```

```
                 date       co     no    no2    o3    so2    pm2_5    pm10  \
0 2025-01-01 00:00:00  1655.58   1.66  39.41  5.90  17.88  169.29  194.64
1 2025-01-01 01:00:00  1869.20   6.82  42.16  1.99  22.17  182.84  211.08
2 2025-01-01 02:00:00  2510.07  27.72  43.87  0.02  30.04  220.25  260.68
3 2025-01-01 03:00:00  3150.94  55.43  44.55  0.85  35.76  252.90  304.12
4 2025-01-01 04:00:00  3471.37  68.84  45.24  5.45  39.10  266.36  322.80

     nh3  AQI    AQI Category
0   5.83  300  Very Unhealthy
1   7.66  300  Very Unhealthy
2  11.40  400       Hazardous
3  13.55  400       Hazardous
4  14.19  400       Hazardous
```

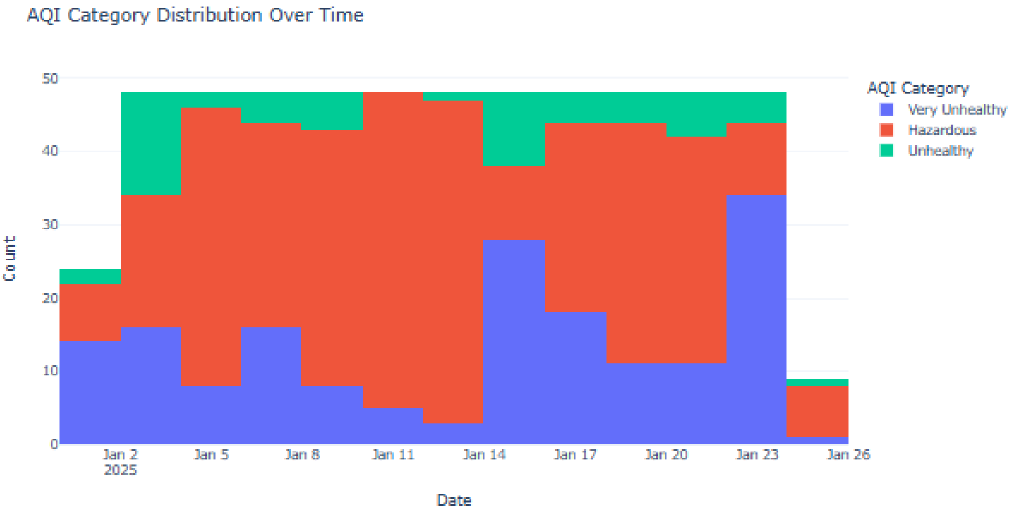
AQI of Delhi

In [9]:

```
fig = px.bar(data, x="date", y="AQI",
             title="AQI of Delhi in January")
fig.update_xaxes(title="Date")
fig.update_yaxes(title="AQI")
fig.show()
```



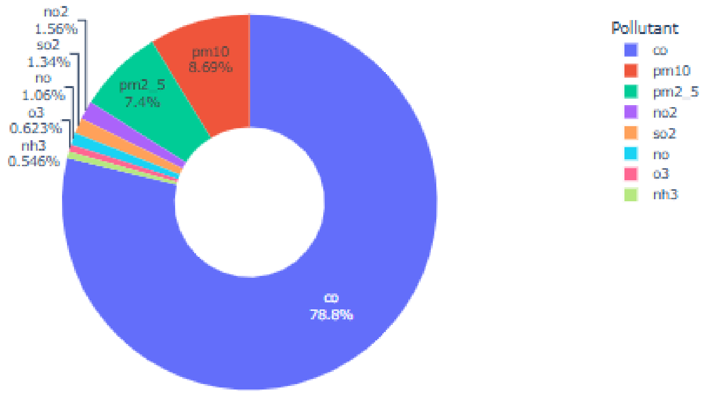
```
In [10]: fig = px.histogram(data, x="date",
                            color="AQI Category",
                            title="AQI Category Distribution Over Time")
         fig.update_xaxes(title="Date")
         fig.update_yaxes(title="Count")
         fig.show()
```



```
In [11]: # Define pollutants and their colors
         pollutants = ["co", "no", "no2", "o3", "so2", "pm2_5", "pm10", "nh3"]
         pollutant_colors = px.colors.qualitative.Plotly

         # Calculate the sum of pollutant concentrations
         total_concentrations = data[pollutants].sum()

         # Create a DataFrame for the concentrations
         concentration_data = pd.DataFrame({
             "Pollutant": pollutants,
             "Concentration": total_concentrations
         })
```

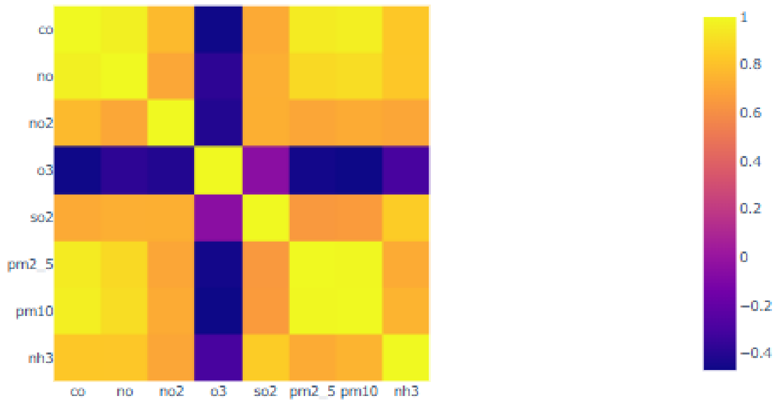
```
In [12]: # Create a donut plot for pollutant concentrations
         fig = px.pie(concentration_data, names="Pollutant", values="Concentration",
                      title="Pollutant Concentrations in Delhi",
                      hole=0.4, color_discrete_sequence=pollutant_colors)

         # Update layout for the donut plot
         fig.update_traces(textinfo="percent+label")
         fig.update_layout(legend_title="Pollutant")

         fig.show()
```

Pollutant Concentrations in Delhi



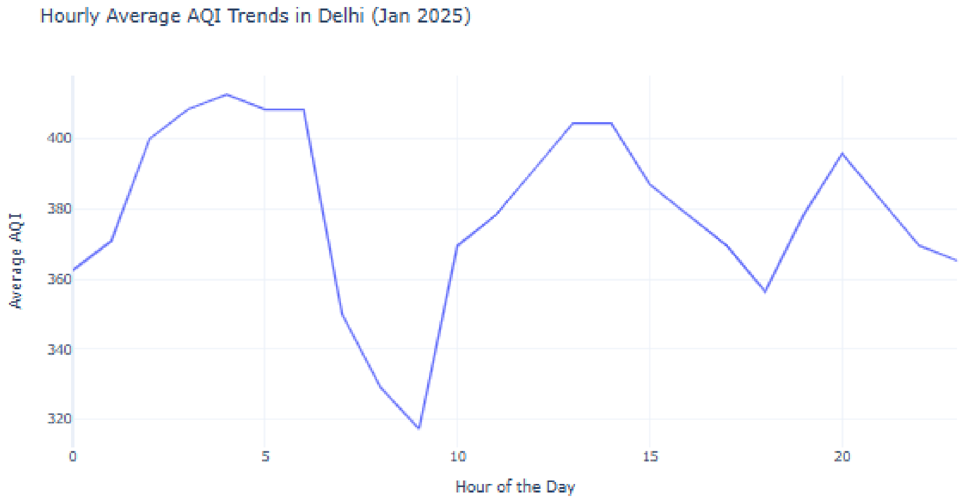
```
In [13]: # Correlation Between Pollutants
         correlation_matrix = data[pollutants].corr()
         fig = px.imshow(correlation_matrix, x=pollutants,
                         y=pollutants, title="Correlation Between Pollutants")
         fig.show()
```

Correlation Between Pollutants



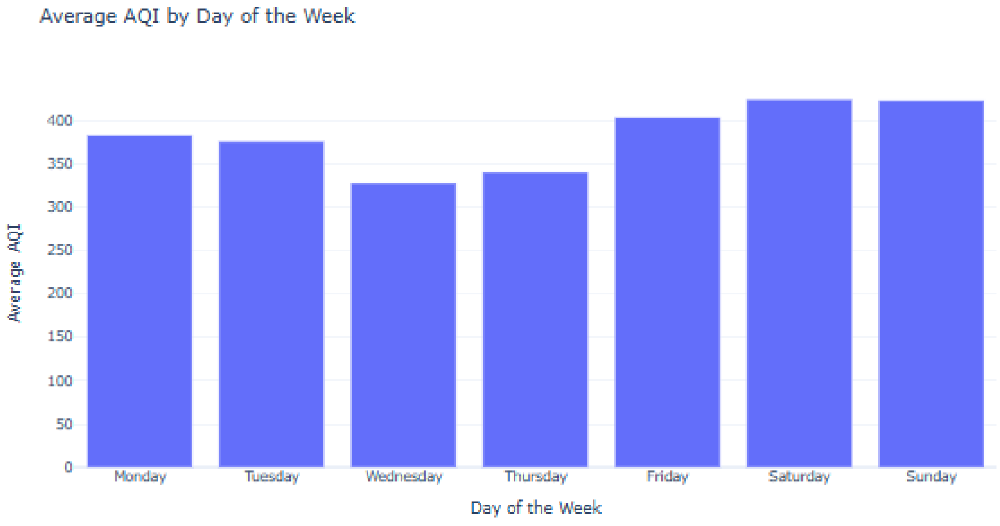
In [14]:

```
# Extract the hour from the date
data['Hour'] = pd.to_datetime(data['date']).dt.hour

# Calculate hourly average AQI
hourly_avg_aqi = data.groupby('Hour')['AQI'].mean().reset_index()

# Create a line plot for hourly trends in AQI
fig = px.line(hourly_avg_aqi, x='Hour', y='AQI',
              title='Hourly Average AQI Trends in Delhi (Jan 2025)')
fig.update_xaxes(title="Hour of the Day")
fig.update_yaxes(title="Average AQI")
fig.show()
```



In [15]:

```
# Average AQI by Day of the Week
data['Day_of_Week'] = data['date'].dt.day_name()
average_aqi_by_day = data.groupby('Day_of_Week')['AQI'].mean().reindex(['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday',
fig = px.bar(average_aqi_by_day, x=average_aqi_by_day.index, y='AQI',
              title='Average AQI by Day of the Week')
fig.update_xaxes(title="Day of the Week")
fig.update_yaxes(title="Average AQI")
fig.show()
```


Average AQI by Day of the Week
Average AQI
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday
Sunday
Day of the Week