```
In [1]:
#Importing Libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from scipy import stats
import seaborn as sns
```

```
In [2]:
Github_link : "https://github.com/Saqlainshaikh05/Weather-Analysis"
```

```
In [3]:
#Load the csv file
dataframe = pd.read_csv(r"C:\Users\HP\Desktop\Nexus\Task 1\weather - weather.csv")
```

```
In [4]:
#First Look of the data
dataframe
```

Out[4]:

	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	WindGustSpeed	WindDir9am	WindDir3pm	WindSpeed9am	...	Humidity3pm	Pressure9am	Pressure3pm	Cloud9am	Cloud3pm	Temp9am	Temp3pm	Rain
0	8.0	24.3	0.0	3.4	6.3	NW	30.0	SW	NW	6.0	...	29	1019.7	1015.0	7	7	14.4	23.6	
1	14.0	26.9	3.6	4.4	9.7	ENE	39.0	E	W	4.0	...	36	1012.4	1008.4	5	3	17.5	25.7	
2	13.7	23.4	3.6	5.8	3.3	NW	85.0	N	NNE	6.0	...	69	1009.5	1007.2	8	7	15.4	20.2	
3	13.3	15.5	39.8	7.2	9.1	NW	54.0	WNW	W	30.0	...	56	1005.5	1007.0	2	7	13.5	14.1	
4	7.6	16.1	2.8	5.6	10.6	SSE	50.0	SSE	ESE	20.0	...	49	1018.3	1018.5	7	7	11.1	15.4	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
361	9.0	30.7	0.0	7.6	12.1	NNW	76.0	SSE	NW	7.0	...	15	1016.1	1010.8	1	3	20.4	30.0	
362	7.1	28.4	0.0	11.6	12.7	N	48.0	NNW	NNW	2.0	...	22	1020.0	1016.9	0	1	17.2	28.2	
363	12.5	19.9	0.0	8.4	5.3	ESE	43.0	ENE	ENE	11.0	...	47	1024.0	1022.8	3	2	14.5	18.3	
364	12.5	26.9	0.0	5.0	7.1	NW	46.0	SSW	WNW	6.0	...	39	1021.0	1016.2	6	7	15.8	25.9	
365	12.3	30.2	0.0	6.0	12.6	NW	78.0	NW	WNW	31.0	...	13	1009.6	1009.2	1	1	23.8	28.6	

366 rows × 22 columns

```
In [5]:
# Shape of the Dataframe
rows,columns = dataframe.shape
```

```
In [6]:
print('The Shape of the datafarme is: ',dataframe.shape)
print('The Number of Rows are: ',rows)
print('The Number of Columns are: ',columns)
```

```
The Shape of the datafarme is:  (366, 22)
The Number of Rows are:  366
The Number of Columns are:  22
```

```
In [7]:
#Information about the dataframe columns
dataframe.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 366 entries, 0 to 365
Data columns (total 22 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   MinTemp        366 non-null    float64
 1   MaxTemp        366 non-null    float64
 2   Rainfall       366 non-null    float64
 3   Evaporation    366 non-null    float64
 4   Sunshine       363 non-null    float64
 5   WindGustDir    363 non-null    object
 6   WindGustSpeed  364 non-null    float64
 7   WindDir9am     335 non-null    object
 8   WindDir3pm     365 non-null    object
 9   WindSpeed9am   359 non-null    float64
 10  WindSpeed3pm   366 non-null    int64
 11  Humidity9am    366 non-null    int64
 12  Humidity3pm    366 non-null    int64
 13  Pressure9am    366 non-null    float64
 14  Pressure3pm    366 non-null    float64
 15  Cloud9am       366 non-null    int64
 16  Cloud3pm       366 non-null    int64
 17  Temp9am        366 non-null    float64
 18  Temp3pm        366 non-null    float64
 19  RainToday      366 non-null    object
 20  RISK_MM        366 non-null    float64
 21  RainTomorrow   366 non-null    object
dtypes: float64(12), int64(5), object(5)
memory usage: 63.0+ KB
```

```
In [8]:
#Describing the dataframe numerical columns.
dataframe.describe()
```

Out[8]:

	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustSpeed	WindSpeed9am	WindSpeed3pm	Humidity9am	Humidity3pm	Pressure9am	Pressure3pm	Cloud9am	Cloud3pm	Temp9am	Temp3pm	RI
count	366.000000	366.000000	366.000000	366.000000	363.000000	364.000000	359.000000	366.000000	366.000000	366.000000	366.000000	366.000000	366.000000	366.000000	366.000000	366.000000	366
mean	7.265574	20.550273	1.428415	4.521858	7.909366	39.840659	9.651811	17.986339	72.035519	44.519126	1019.709016	1016.810383	3.890710	4.024590	12.358470	19.230874	1
std	6.025800	6.690516	4.225800	2.669383	3.481517	13.059807	7.951929	8.856997	13.137058	16.850947	6.686212	6.469422	2.956131	2.666268	5.630832	6.640346	4
min	-5.300000	7.600000	0.000000	0.200000	0.000000	13.000000	0.000000	0.000000	36.000000	13.000000	996.500000	996.800000	0.000000	0.000000	0.100000	5.100000	0
25%	2.300000	15.025000	0.000000	2.200000	5.950000	31.000000	6.000000	11.000000	64.000000	32.250000	1015.350000	1012.800000	1.000000	1.000000	7.625000	14.150000	0
50%	7.450000	19.650000	0.000000	4.200000	8.600000	39.000000	7.000000	17.000000	72.000000	43.000000	1020.150000	1017.400000	3.500000	4.000000	12.550000	18.550000	0
75%	12.500000	25.500000	0.200000	6.400000	10.500000	46.000000	13.000000	24.000000	81.000000	55.000000	1024.475000	1021.475000	7.000000	7.000000	17.000000	24.000000	0
max	20.900000	35.800000	39.800000	13.800000	13.600000	98.000000	41.000000	52.000000	99.000000	96.000000	1035.700000	1033.200000	8.000000	8.000000	24.700000	34.500000	39

```
In [9]:
#Dataframe columns
dataframe.columns
```

```
Out[9]:
Index(['MinTemp', 'MaxTemp', 'Rainfall', 'Evaporation', 'Sunshine',
       'WindGustDir', 'WindGustSpeed', 'WindDir9am', 'WindDir3pm',
       'WindSpeed9am', 'WindSpeed3pm', 'Humidity9am', 'Humidity3pm',
       'Pressure9am', 'Pressure3pm', 'Cloud9am', 'Cloud3pm', 'Temp9am',
       'Temp3pm', 'RainToday', 'RISK_MM', 'RainTomorrow'],
      dtype='object')
```

```
In [10]:
# Task 1
# 1. Descriptive Statistics:
#Compute and present basic statistics (mean, median, standard deviation) for MinTemp, MaxTemp, Rainfall, and Evaporation.

dataframe[['MinTemp','MaxTemp','Rainfall','Evaporation']].describe()
```

Out[10]:

	MinTemp	MaxTemp	Rainfall	Evaporation
count	366.000000	366.000000	366.000000	366.000000
mean	7.265574	20.550273	1.428415	4.521858
std	6.025800	6.690516	4.225800	2.669383
min	-5.300000	7.600000	0.000000	0.200000
25%	2.300000	15.025000	0.000000	2.200000
50%	7.450000	19.650000	0.000000	4.200000
75%	12.500000	25.500000	0.200000	6.400000
max	20.900000	35.800000	39.800000	13.800000

```
In [11]:
MinTemp_mean = round(dataframe['MinTemp'].mean(),2)
MaxTemp_mean = round(dataframe['MaxTemp'].mean(),2)
Rainfall_mean = round(dataframe['Rainfall'].mean(),2)
Evaporation_mean = round(dataframe['Evaporation'].mean(),2)
```

```
In [12]:
# Mean for the following are
print('Mean for Mininum Temperature is :',MinTemp_mean)
print('Mean for Maximum Temperature is :',MaxTemp_mean)
print('Mean for Rainfall Temperature is :',Rainfall_mean)
print('Mean for Evaporation Temperature is :',Evaporation_mean)
```

```
Mean for Mininum Temperature is : 7.27
Mean for Maximum Temperature is : 20.55
Mean for Rainfall Temperature is : 1.43
Mean for Evaporation Temperature is : 4.52
```

```
In [13]:
MinTemp_median = round(dataframe['MinTemp'].median(),2)
MaxTemp_median = round(dataframe['MaxTemp'].median(),2)
Rainfall_median = round(dataframe['Rainfall'].median(),2)
Evaporation_median = round(dataframe['Evaporation'].median(),2)
```

```
In [14]:
# Median for the following are
print('Median for Mininum Temperature is :',MinTemp_median)
print('Median for Maximum Temperature is :',MaxTemp_median)
print('Median for Rainfall Temperature is :',Rainfall_median)
print('Median for Evaporation Temperature is :',Evaporation_median)
```

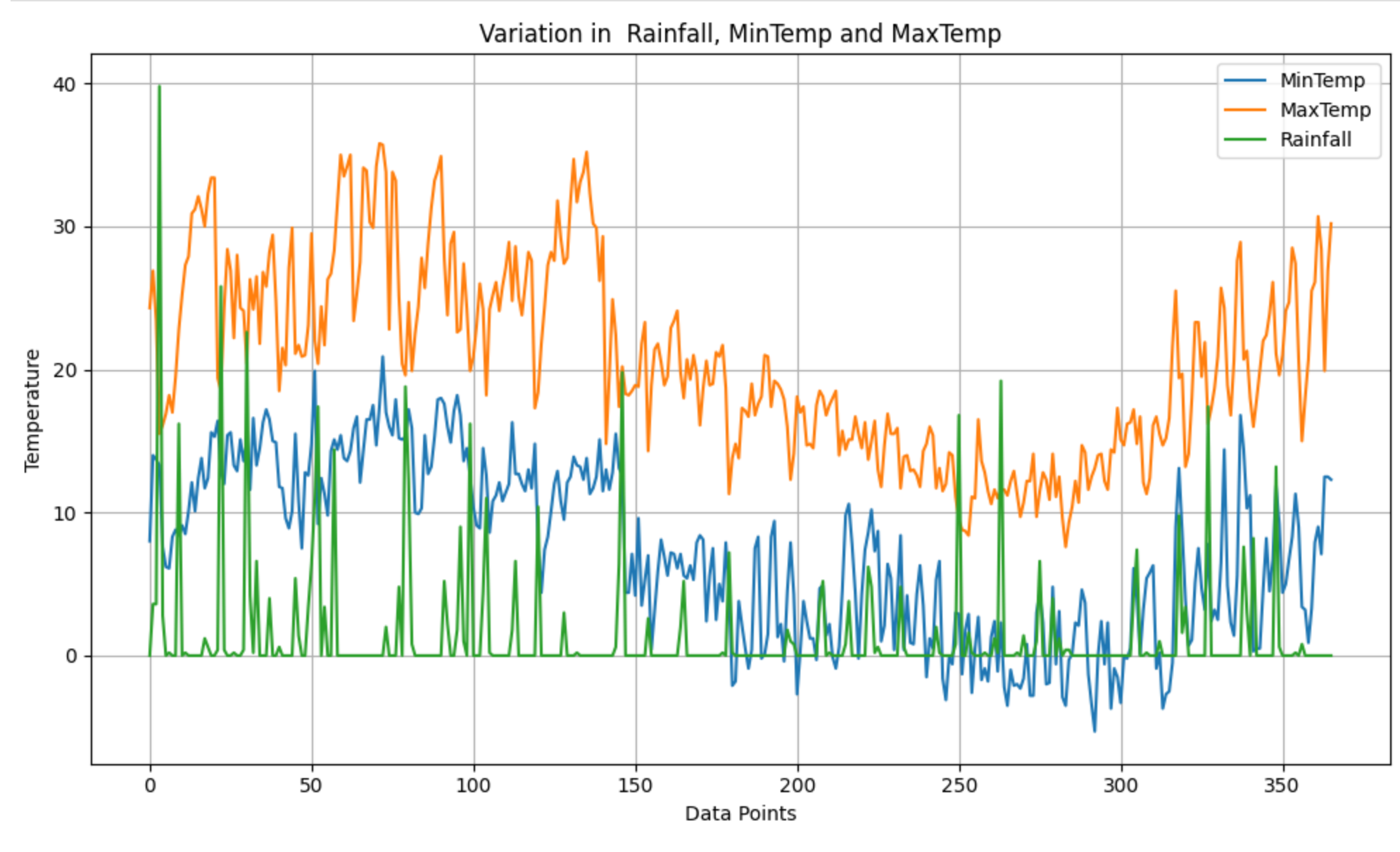
```
Median for Mininum Temperature is : 7.45
Median for Maximum Temperature is : 19.65
Median for Rainfall Temperature is : 0.0
Median for Evaporation Temperature is : 4.2
```

```
In [15]:
MinTemp_std = round(dataframe['MinTemp'].std(),2)
MaxTemp_std = round(dataframe['MaxTemp'].std(),2)
Rainfall_std = round(dataframe['Rainfall'].std(),2)
Evaporation_std = round(dataframe['Evaporation'].std(),2)
```

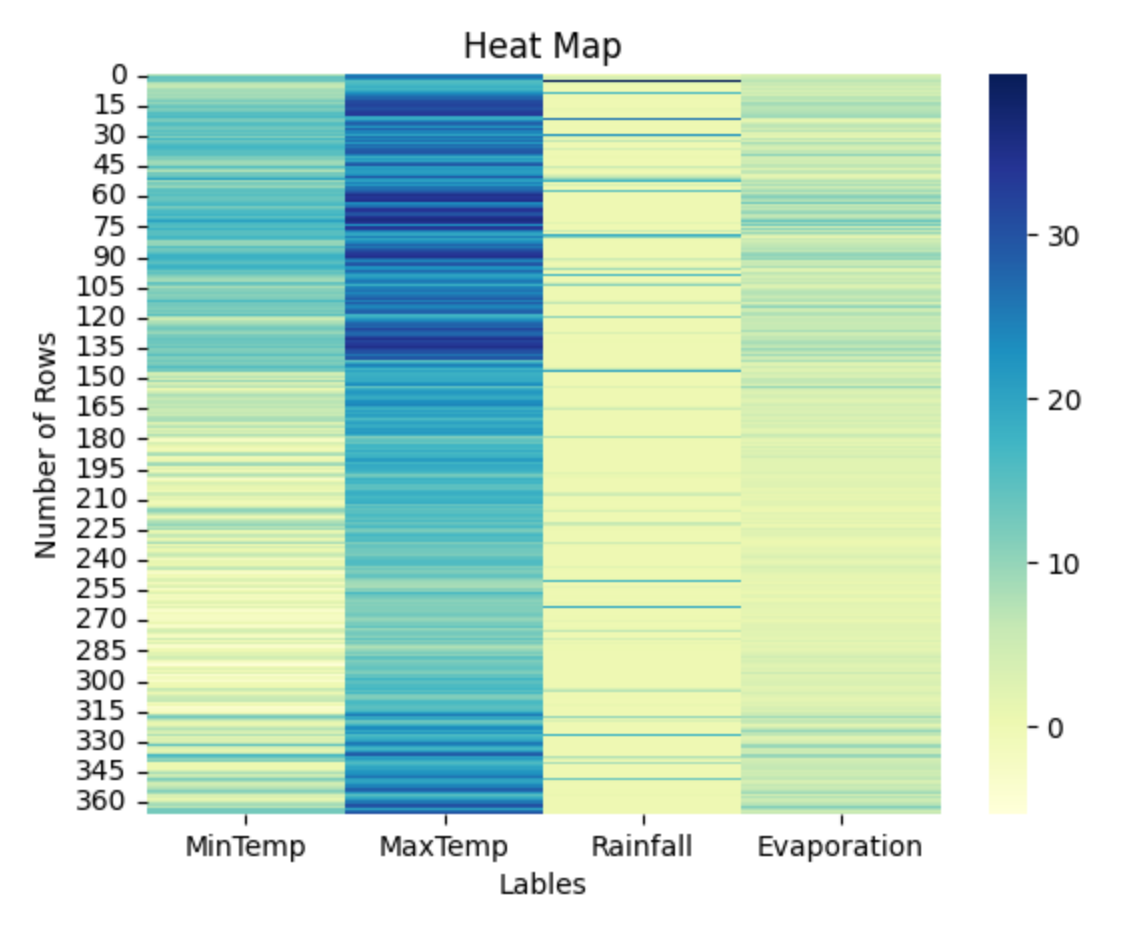
```
In [16]:
# Standard Deviation for the following are
print('Standard Deviation for Mininum Temperature is :',MinTemp_std)
print('Standard Deviation for Maximum Temperature is :',MaxTemp_std)
print('Standard Deviation for Rainfall Temperature is :',Rainfall_std)
print('Standard Deviation for Evaporation Temperature is :',Evaporation_std)
```

```
Standard Deviation for Mininum Temperature is : 6.03
Standard Deviation for Maximum Temperature is : 6.69
Standard Deviation for Rainfall Temperature is : 4.23
Standard Deviation for Evaporation Temperature is : 2.67
```

```
In [21]:
# 2.Time Series Visualization
# Create a line chart to show the variations in weather variables over time, identifying trends or patterns.
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np

# Assuming you have a DataFrame named 'dataframe' with columns including 'MinTemp' and 'MaxTemp'
# Replace 'dataframe' with the name of your DataFrame

# Creating an index to represent data points
index = np.arange(len(dataframe))

# Plotting MinTemp and MaxTemp
plt.figure(figsize=(10, 6))

plt.plot(index, dataframe['MinTemp'], label='MinTemp')
plt.plot(index, dataframe['MaxTemp'], label='MaxTemp')
plt.plot(index,dataframe['Rainfall'],label = 'Rainfall')

plt.title('Variation in  Rainfall, MinTemp and MaxTemp')
plt.xlabel('Data Points')
plt.ylabel('Temperature')
plt.legend()
plt.grid(True)
plt.tight_layout()

plt.show()
```




```
In [22]:
# 3. Correlation Analysis.
# Calculate and visualize correlations between MinTemp, MaxTemp, Rainfall, and Evaporation using a heatmap.
dataframe_heatmap = sns.heatmap(data = dataframe[['MinTemp','MaxTemp','Rainfall','Evaporation']],cmap="YlGnBu")
plt.title('Heat Map')
plt.xlabel('Lables')
plt.ylabel('Number of Rows')
plt.show()
```




```
In [23]:
# 4.Rainfall Distribution:
# Illustrate the distribution of rainfall through a histogram or kernel density plot, highlighting common levels and outliers.
```

```
In [24]:
dataframe[['Rainfall']].max()
```

```
Out[24]:
Rainfall    39.8
dtype: float64
```

```
In [25]:
dataframe[['Rainfall']].min()
```

```
Out[25]:
Rainfall    0.0
dtype: float64
```

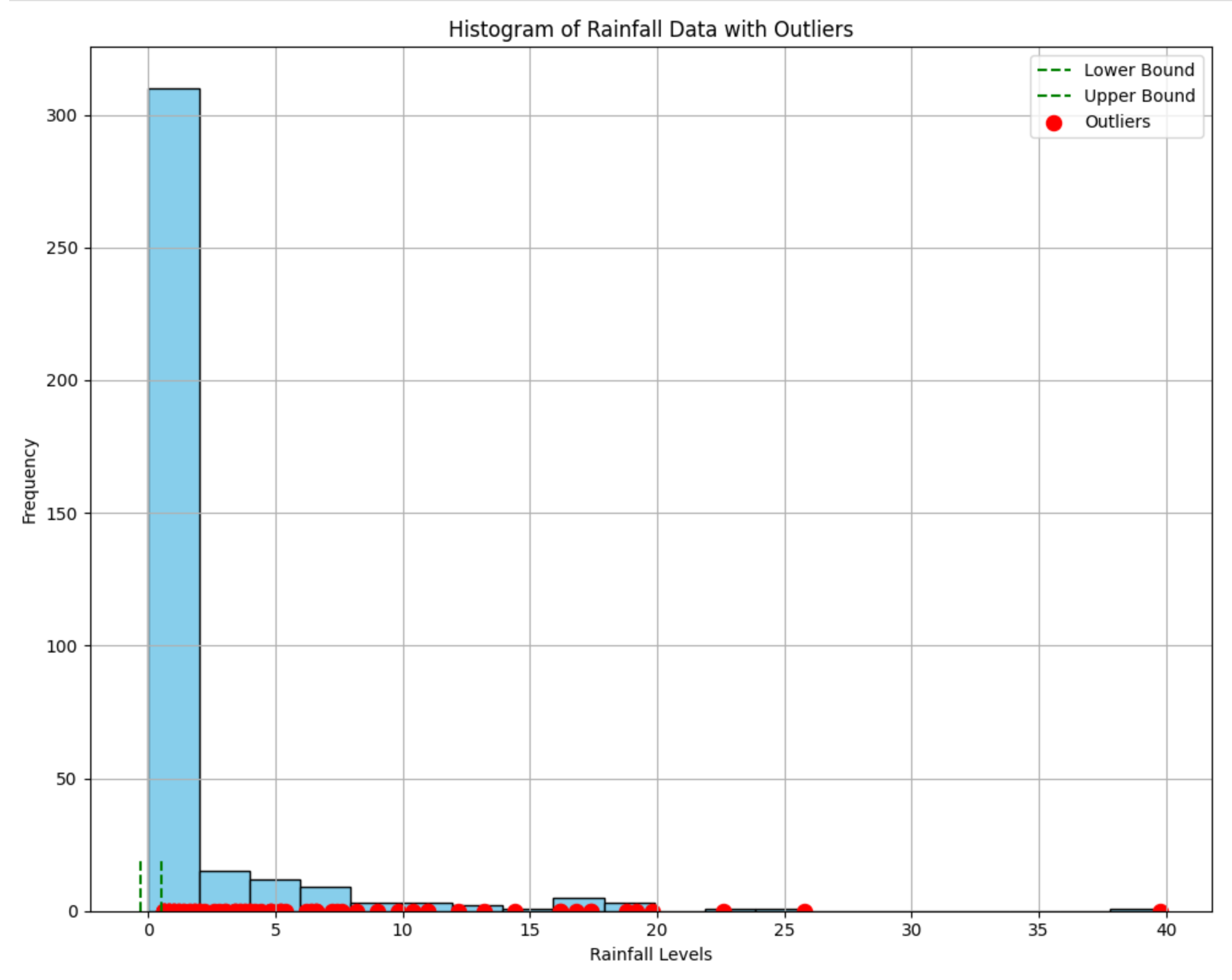
```
In [26]:
numeric_values_rainfall = dataframe[['Rainfall']].values
```

```
In [27]:
flat_numeric_rainfall = numeric_values_rainfall.flatten()
```

```
In [28]:
Q1 = np.percentile(flat_numeric_rainfall,25)
Q3 = np.percentile(flat_numeric_rainfall,75)

IQR = Q3 - Q1

lower_bound = Q1 - 1.5*IQR
upper_bound = Q3 + 1.5*IQR
outliers = flat_numeric_rainfall[(flat_numeric_rainfall < lower_bound) | (flat_numeric_rainfall > upper_bound)]
print("Outliers:", outliers)
```

```
Outliers: [ 3.6  3.6 39.8  2.8 16.2  1.2  0.6 25.8 22.6  4.2  6.6  4.   0.6  5.4
  1.4  3.4  6.4 11.  17.4  3.4 14.4  2.   4.8 18.8 12.2  0.8  5.2  2.2
  1.8  9.   1.  16.2  4.4 11.   1.8  6.6 10.4  3.   0.6  6.4 19.8  2.6
  2.   5.2  7.2  1.8  1.   0.8  3.8  5.2  0.8  3.8  6.2  4.8  0.6  4.8
  0.6  2.   0.8 16.8  1.6  1.2 19.2  1.4  1.   6.6  4.   1.2  4.   7.4
  1.   9.8  1.6  3.4 17.4  7.6  3.   8.2 13.2  0.6  0.8]
```

```
In [29]:
plt.figure(figsize = (10,8))
plt.hist(flat_numeric_rainfall, bins=20, color='skyblue', edgecolor='black')

# Highlighting outliers on the histogram
plt.plot([lower_bound, lower_bound], [0, 20], color='green', linestyle='--', label='Lower Bound')
plt.plot([upper_bound, upper_bound], [0, 20], color='green', linestyle='--', label='Upper Bound')

plt.scatter(outliers, np.zeros_like(outliers), color='red', label='Outliers', s=75)

plt.title('Histogram of Rainfall Data with Outliers')
plt.xlabel('Rainfall Levels')
plt.ylabel('Frequency')
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```



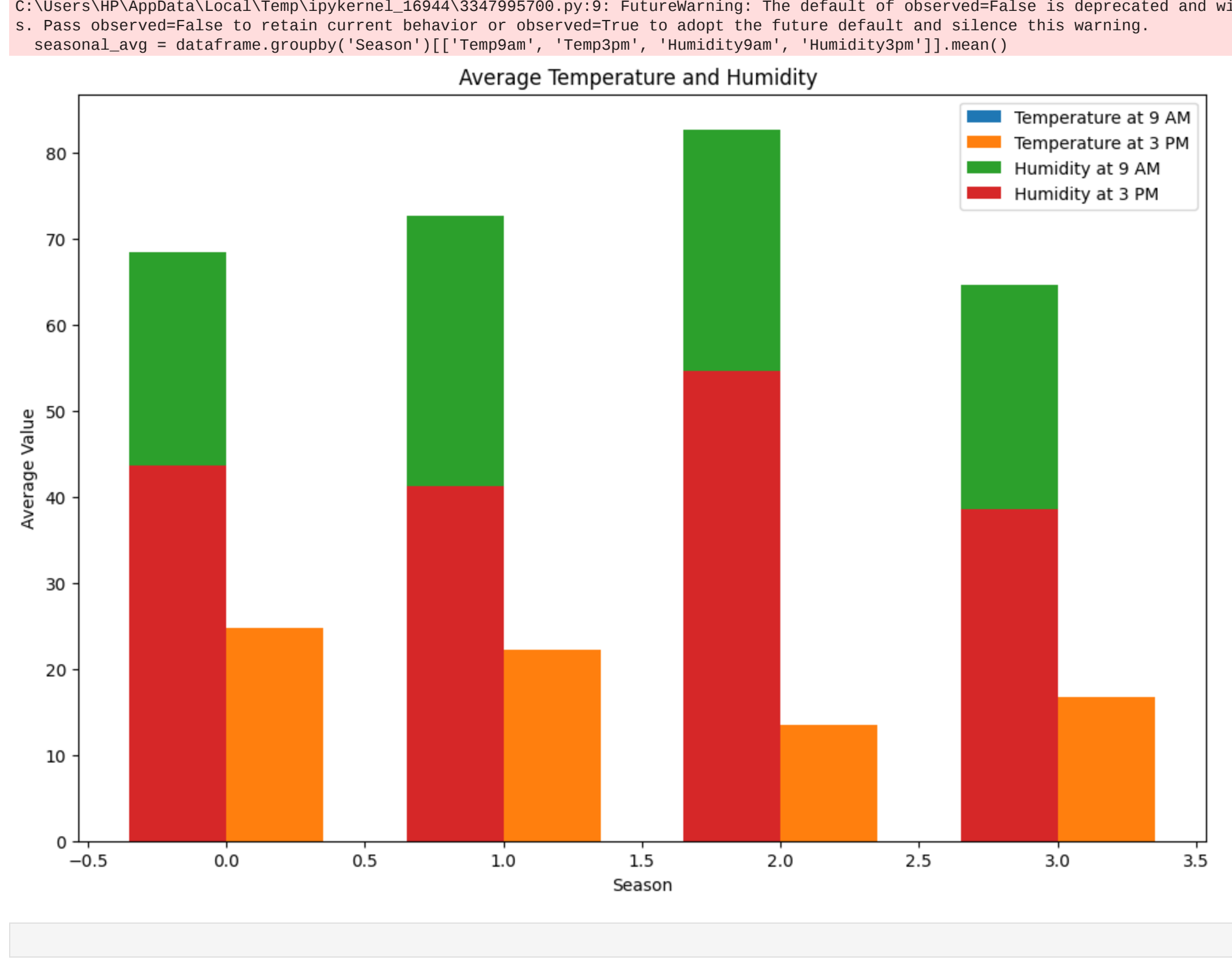

```
In [30]:
outliers
```

```
Out[30]:
array([ 3.6,  3.6, 39.8,  2.8, 16.2,  1.2,  0.6, 25.8, 22.6,  4.2,  6.6,
        4. ,  0.6,  5.4,  1.4,  3.4,  6.4, 11. , 17.4,  3.4, 14.4,  2. ,
        4.8, 18.8, 12.2,  0.8,  5.2,  2.2,  1.8,  9. ,  1. , 16.2,  4.4,
       11. ,  1.8,  6.6, 10.4,  3. ,  0.6,  6.4, 19.8,  2.6,  2. ,  5.2,
        7.2,  1.8,  1. ,  0.8,  3.8,  5.2,  0.8,  3.8,  6.2,  4.8,  0.6,
        4.8,  0.6,  2. ,  0.8, 16.8,  1.6,  1.2, 19.2,  1.4,  1. ,  6.6,
        4. ,  1.2,  4. ,  7.4,  1. ,  9.8,  1.6,  3.4, 17.4,  7.6,  3. ,
        8.2, 13.2,  0.6,  0.8])
```

```
In [ ]:
# 5. Seasonal Analysis:
# Analyze average values of weather variables across different seasons and visualize seasonal patterns with bar graphs.
```

```
In [36]:
start_date = '2020-01-01'
end_date = '2020-12-31'
dataframe['Date'] = pd.date_range(start=start_date, end=end_date)

dataframe['Season'] = pd.cut(dataframe['Date'].dt.month, bins=[0, 3, 6, 9, 12],
                             labels=['Winter', 'Spring', 'Summer', 'Fall'])

# Calculating average values of temperature and humidity per season
seasonal_avg = dataframe.groupby('Season')[['Temp9am', 'Temp3pm', 'Humidity9am', 'Humidity3pm']].mean()


import matplotlib.pyplot as plt


plt.figure(figsize=(12, 8))

bar_width = 0.35
index = range(len(seasonal_avg))

plt.bar(index, seasonal_avg['Temp9am'], width=bar_width, label='Temperature at 9 AM',align='edge')
plt.bar(index, seasonal_avg['Temp3pm'], width=bar_width, label='Temperature at 3 PM', align='edge')



plt.bar(index, seasonal_avg['Humidity9am'], width=-bar_width, label='Humidity at 9 AM', align='edge')
plt.bar(index, seasonal_avg['Humidity3pm'], width=-bar_width, label='Humidity at 3 PM', align='edge')

plt.xlabel('Season')
plt.ylabel('Average Value')
plt.title('Average Temperature and Humidity ')
plt.legend()
plt.show()
```

```
C:\Users\HP\AppData\Local\Temp\ipykernel_16944\3347995700.py:9: FutureWarning: The default of observed=False is deprecated and will be changed to True in a future version of panda
s. Pass observed=False to retain current behavior or observed=True to adopt the future default and silence this warning.
  seasonal_avg = dataframe.groupby('Season')[['Temp9am', 'Temp3pm', 'Humidity9am', 'Humidity3pm']].mean()
```




```
In [ ]:
```