Climate Data Analysis

Task:-

- Undertake a comprehensive climate data analysis project to explore and understand historical climate patterns and trends. The objective is to derive valuable insights from climate data, enabling a better understanding of weather conditions over time.

Importing necessary libraries

In [1]:

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import datetime
import warnings
warnings.filterwarnings('ignore')
sns.set()
```

Loading and Exploring the Dataset

In [2]:

```python
df = pd.read_csv('climate_change_data.csv')
df.head()
```

Out[2]:

	Date	Location	Country	Temperature	CO2 Emissions	Sea Level Rise	Precipitation	Humidity	Wind Speed
0	2000-01-01 00:00:00.000000000	New Williamtown	Latvia	10.688986	403.118903	0.717506	13.835237	23.631256	18.492026
1	2000-01-01 20:09:43.258325832	North Rachel	South Africa	13.814430	396.663499	1.205715	40.974084	43.982946	34.249300
2	2000-01-02 16:19:26.516651665	West Williamland	French Guiana	27.323718	451.553155	-0.160783	42.697931	96.652600	34.124261
3	2000-01-03 12:29:09.774977497	South David	Vietnam	12.309581	422.404983	-0.475931	5.193341	47.467938	8.554563
4	2000-01-04 08:38:53.033303330	New Scottburgh	Moldova	13.210885	410.472999	1.135757	78.695280	61.789672	8.001164

In [3]:

```python
df['Date'] = pd.to_datetime(df['Date'])
df['Month'] = df['Date'].dt.month_name()
```

In [4]:

```python
df.set_index('Date', inplace=True)
df.head()
```

Out[4]:

Date	Location	Country	Temperature	CO2 Emissions	Sea Level Rise	Precipitation	Humidity	Wind Speed	Month
2000-01-01 00:00:00.000000000	New Williamtown	Latvia	10.688986	403.118903	0.717506	13.835237	23.631256	18.492026	January
2000-01-01 20:09:43.258325832	North Rachel	South Africa	13.814430	396.663499	1.205715	40.974084	43.982946	34.249300	January
2000-01-02 16:19:26.516651665	West Williamland	French Guiana	27.323718	451.553155	-0.160783	42.697931	96.652600	34.124261	January
2000-01-03 12:29:09.774977497	South David	Vietnam	12.309581	422.404983	-0.475931	5.193341	47.467938	8.554563	January
2000-01-04 08:38:53.033303330	New Scottburgh	Moldova	13.210885	410.472999	1.135757	78.695280	61.789672	8.001164	January

In [5]:

```python
df.shape
```

Out[5]: (10000, 9)

In [6]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 10000 entries, 2000-01-01 00:00:00 to 2022-12-31 00:00:00
Data columns (total 9 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   Location        10000 non-null  object 
 1   Country         10000 non-null  object 
 2   Temperature     10000 non-null  float64
 3   CO2 Emissions   10000 non-null  float64
 4   Sea Level Rise  10000 non-null  float64
 5   Precipitation   10000 non-null  float64
 6   Humidity        10000 non-null  float64
 7   Wind Speed      10000 non-null  float64
 8   Month           10000 non-null  object 
dtypes: float64(6), object(3)
memory usage: 781.2+ KB
```

In [7]:

```
df.describe()
```

Out[7]:

	Temperature	CO2 Emissions	Sea Level Rise	Precipitation	Humidity	Wind Speed
count	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000
mean	14.936034	400.220469	-0.003152	49.881208	49.771302	25.082066
std	5.030616	49.696933	0.991349	28.862417	28.929320	14.466648
min	-3.803589	182.131220	-4.092155	0.010143	0.018998	0.001732
25%	11.577991	367.109330	-0.673809	24.497516	24.713250	12.539733
50%	14.981136	400.821324	0.002332	49.818967	49.678412	24.910787
75%	18.305826	433.307905	0.675723	74.524991	75.206390	37.670260
max	33.976956	582.899701	4.116559	99.991900	99.959665	49.997664

In [8]:

```
df.describe(include=object)
```

Out[8]:

	Location	Country	Month
count	10000	10000	10000
unique	7764	243	12
top	North David	Congo	January
freq	12	94	849

In [9]:

```
df.isnull().sum()
```

Out[9]:

```
Location          0
Country           0
Temperature       0
CO2 Emissions     0
Sea Level Rise    0
Precipitation     0
Humidity          0
Wind Speed        0
Month             0
dtype: int64
```

In [10]:

```
df.columns
```

Out[10]:

```
Index(['Location', 'Country', 'Temperature', 'CO2 Emissions', 'Sea Level Rise',
       'Precipitation', 'Humidity', 'Wind Speed', 'Month'],
      dtype='object')
```

In [11]:

```
df.nunique()
```

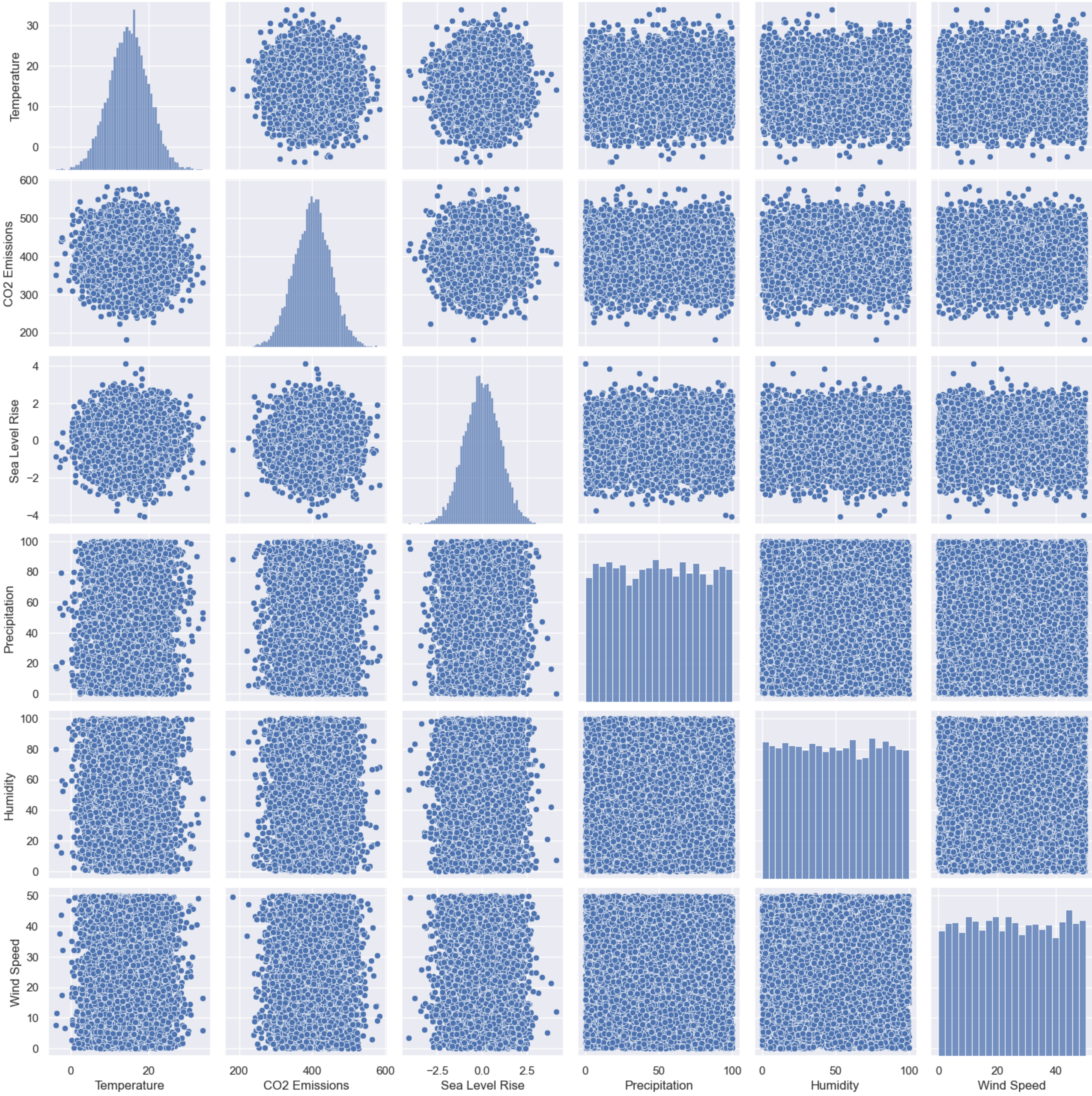
Out[11]:

```
Location           7764
Country             243
Temperature       10000
CO2 Emissions     10000
Sea Level Rise    10000
Precipitation     10000
Humidity          10000
Wind Speed        10000
Month                12
dtype: int64
```

Data Visualization

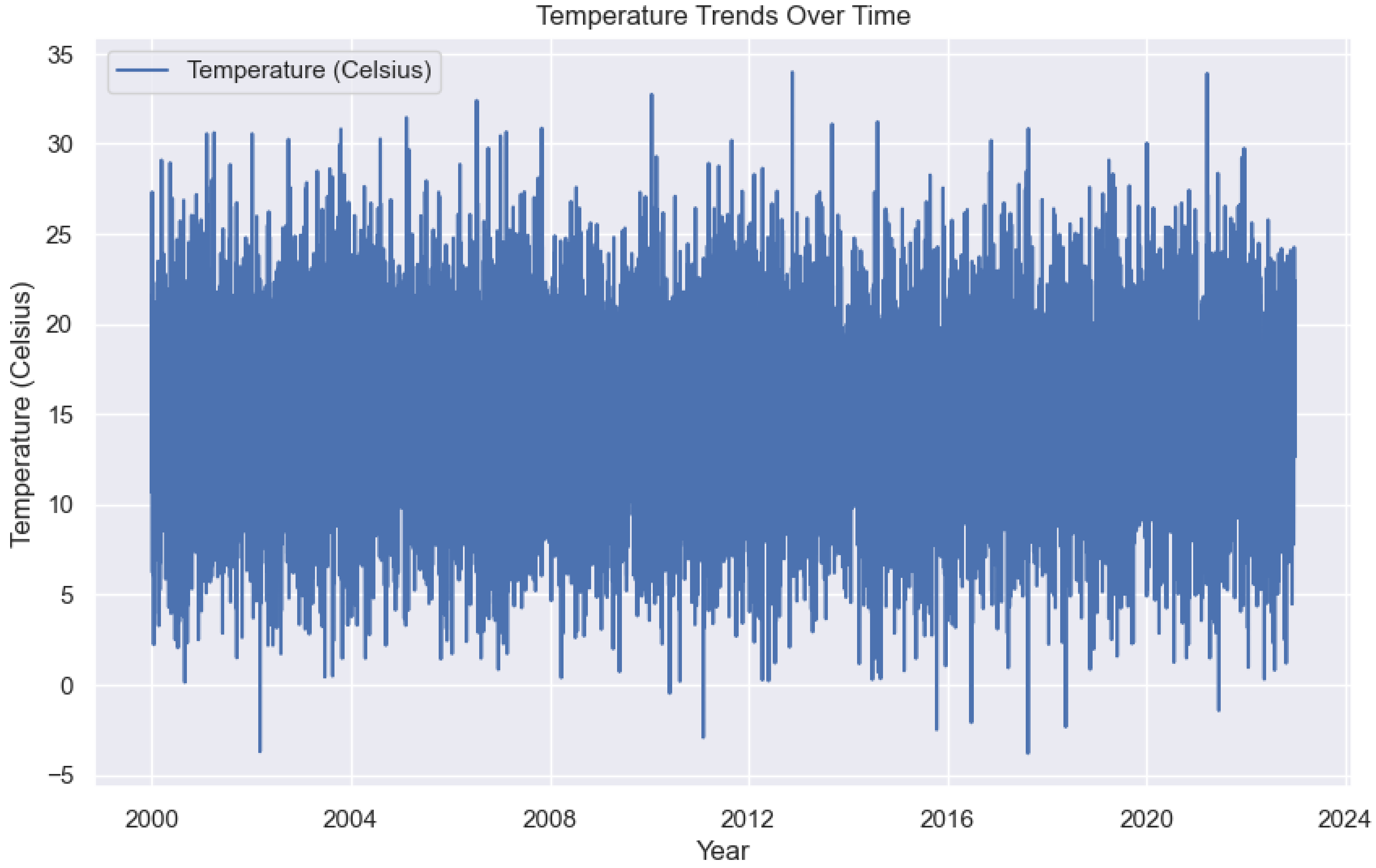
In [12]:
```
sns.pairplot(df)
```

Out[12]:
```
<seaborn.axisgrid.PairGrid at 0x2670961c880>
```



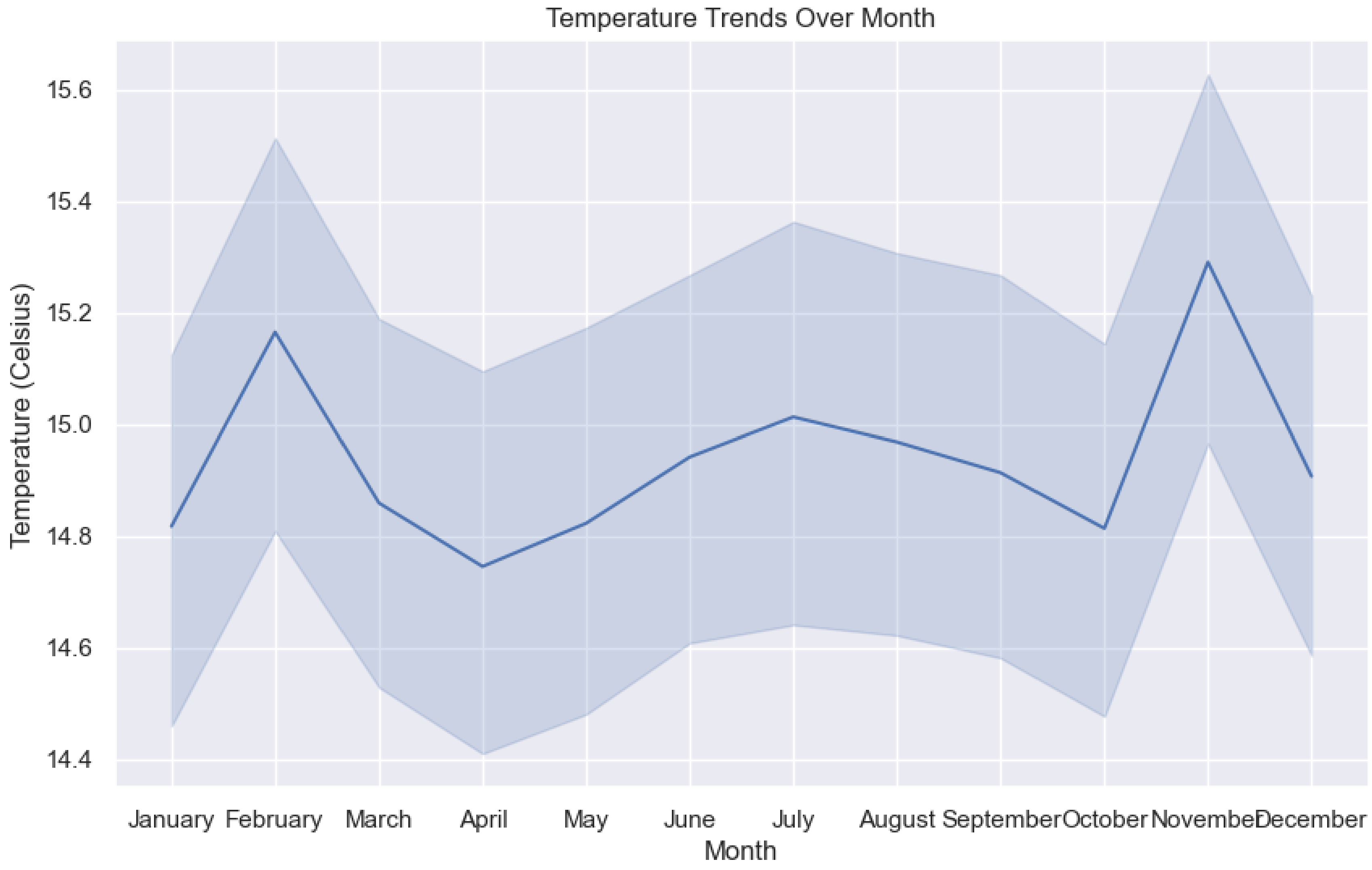
In [13]:

```
plt.figure(figsize=(10, 6))
plt.plot(df['Temperature'], label='Temperature (Celsius)')
plt.xlabel('Year')
plt.ylabel('Temperature (Celsius)')
plt.title('Temperature Trends Over Time')
plt.legend()
plt.show()
```



In [14]:

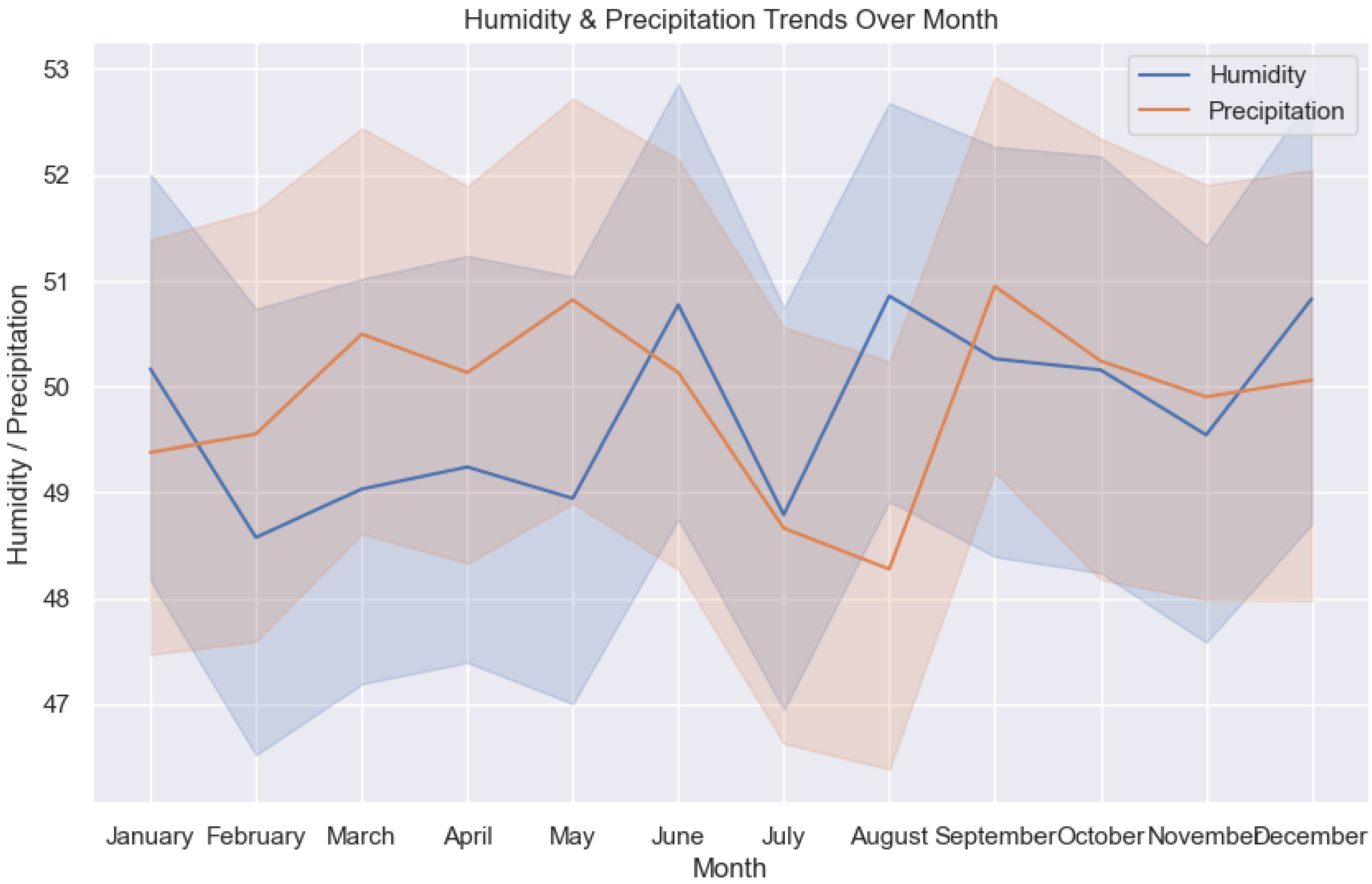
```
plt.figure(figsize=(10, 6))
sns.lineplot(x=df['Month'], y=df['Temperature'])
plt.xlabel('Month')
plt.ylabel('Temperature (Celsius)')
plt.title('Temperature Trends Over Month')
plt.show()
```



In [15]:

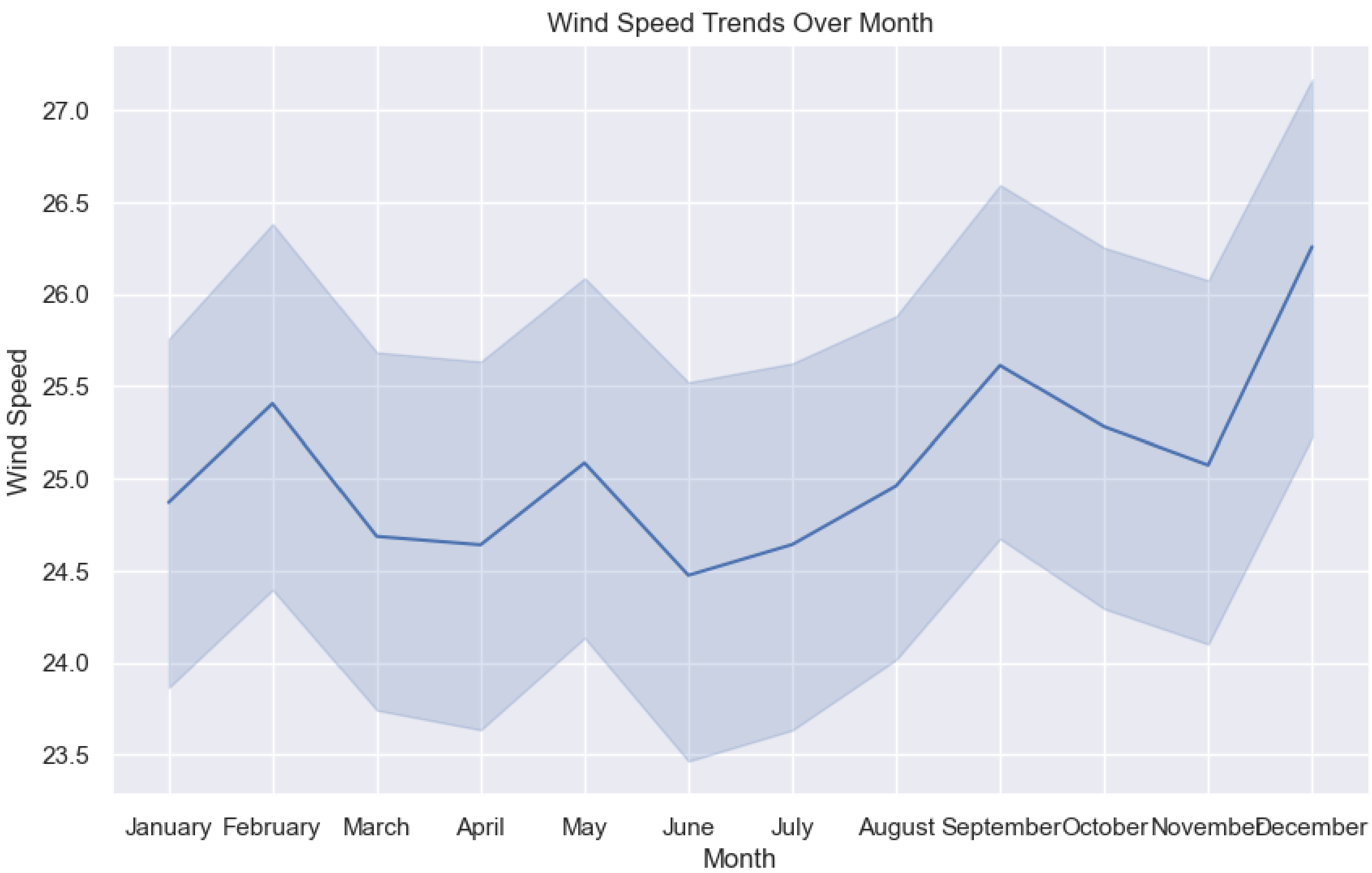
```
plt.figure(figsize=(10, 6))
sns.lineplot(x=df.Month, y=df['Humidity'], label='Humidity')
sns.lineplot(x=df.Month, y=df['Precipitation'], label='Precipitation')

plt.xlabel('Month')
plt.ylabel('Humidity / Precipitation')
plt.title('Humidity & Precipitation Trends Over Month')
plt.show()
```



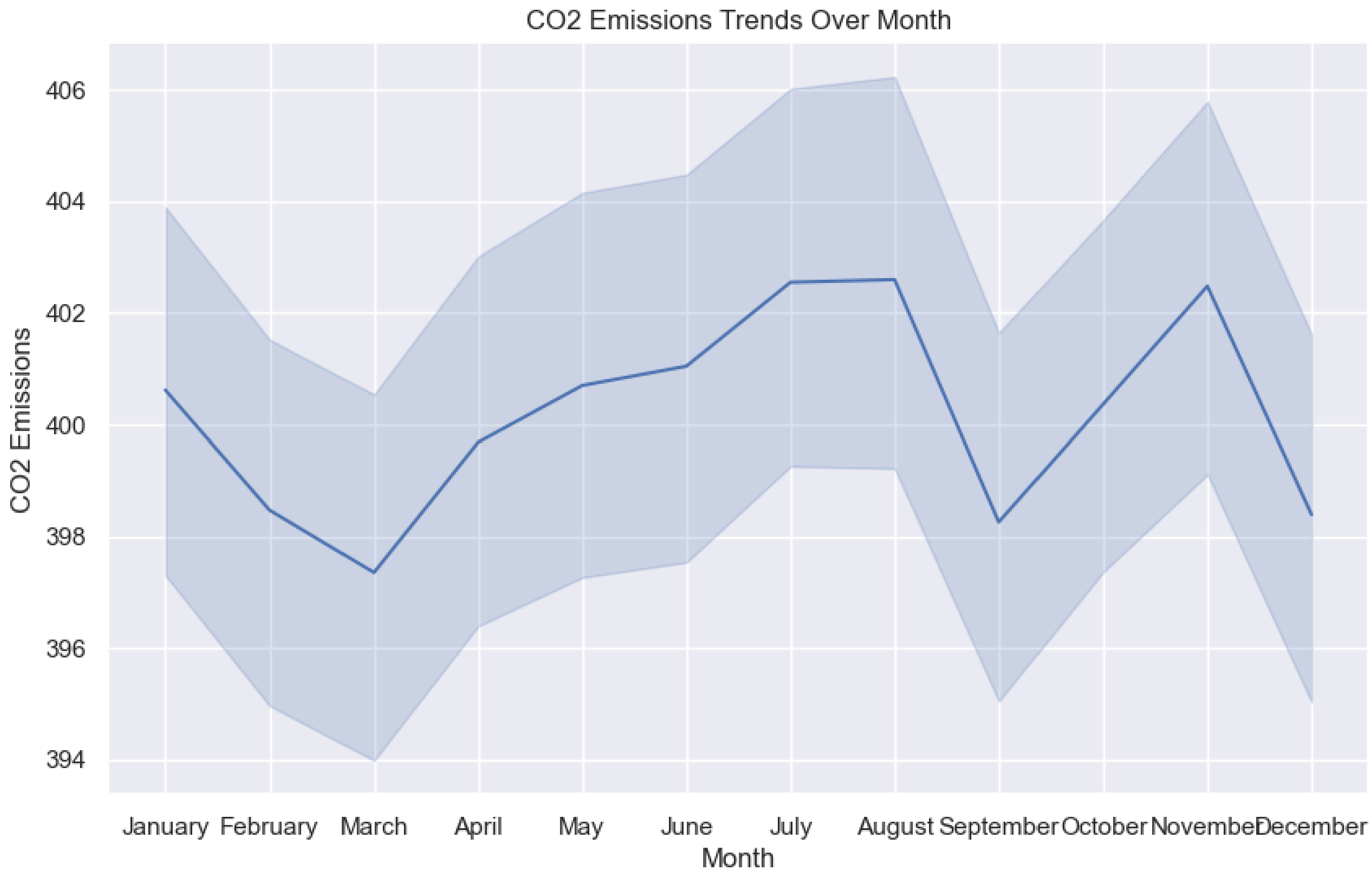
In [16]:

```
plt.figure(figsize=(10, 6))
sns.lineplot(x=df.Month, y=df['Wind Speed'])
plt.xlabel('Month')
plt.ylabel('Wind Speed')
plt.title('Wind Speed Trends Over Month')
plt.show()
```



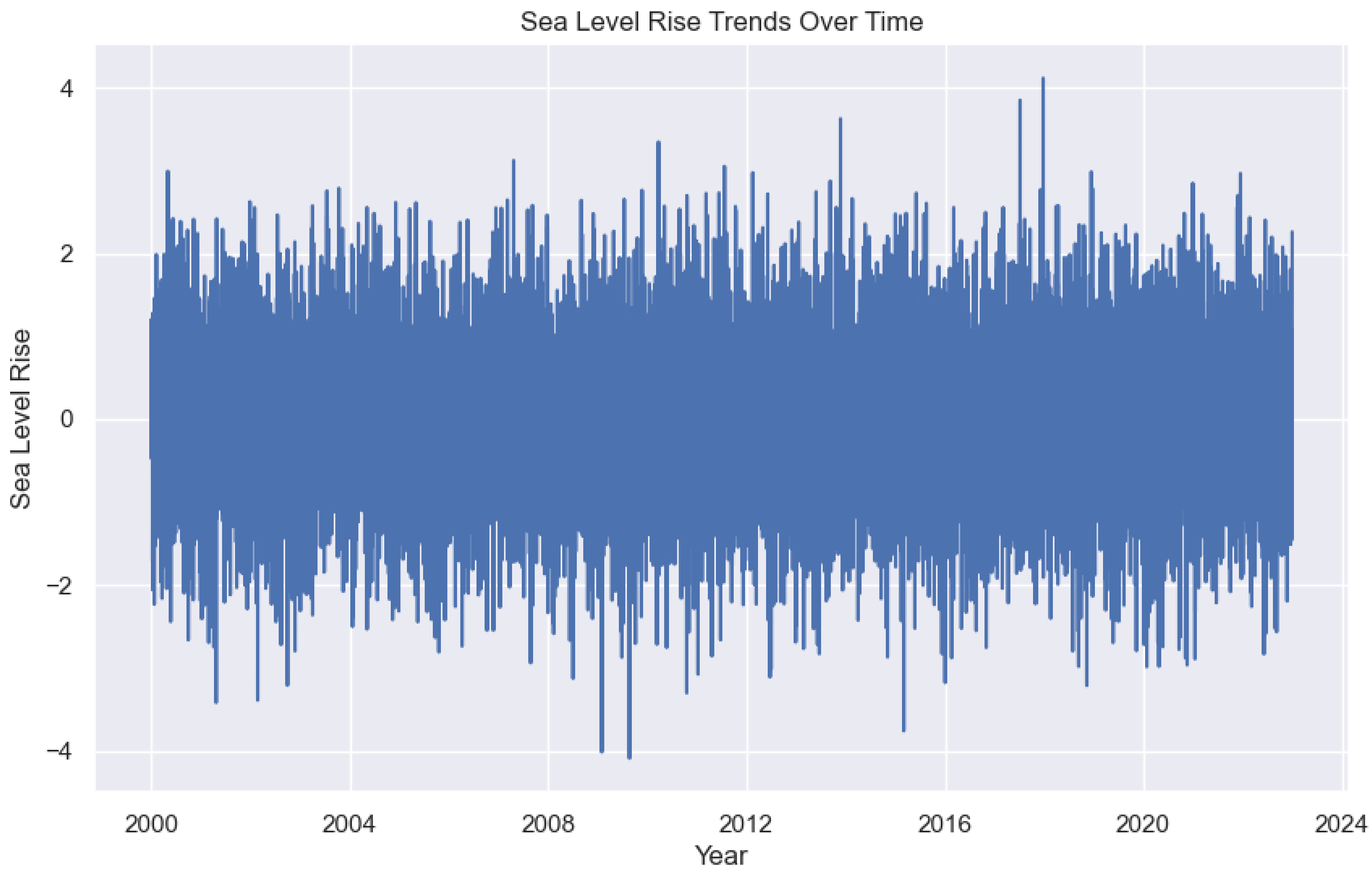
In [17]:

```
plt.figure(figsize=(10, 6))
sns.lineplot(x=df.Month, y=df['CO2 Emissions'])
plt.xlabel('Month')
plt.ylabel('CO2 Emissions')
plt.title('CO2 Emissions Trends Over Month')
plt.show()
```



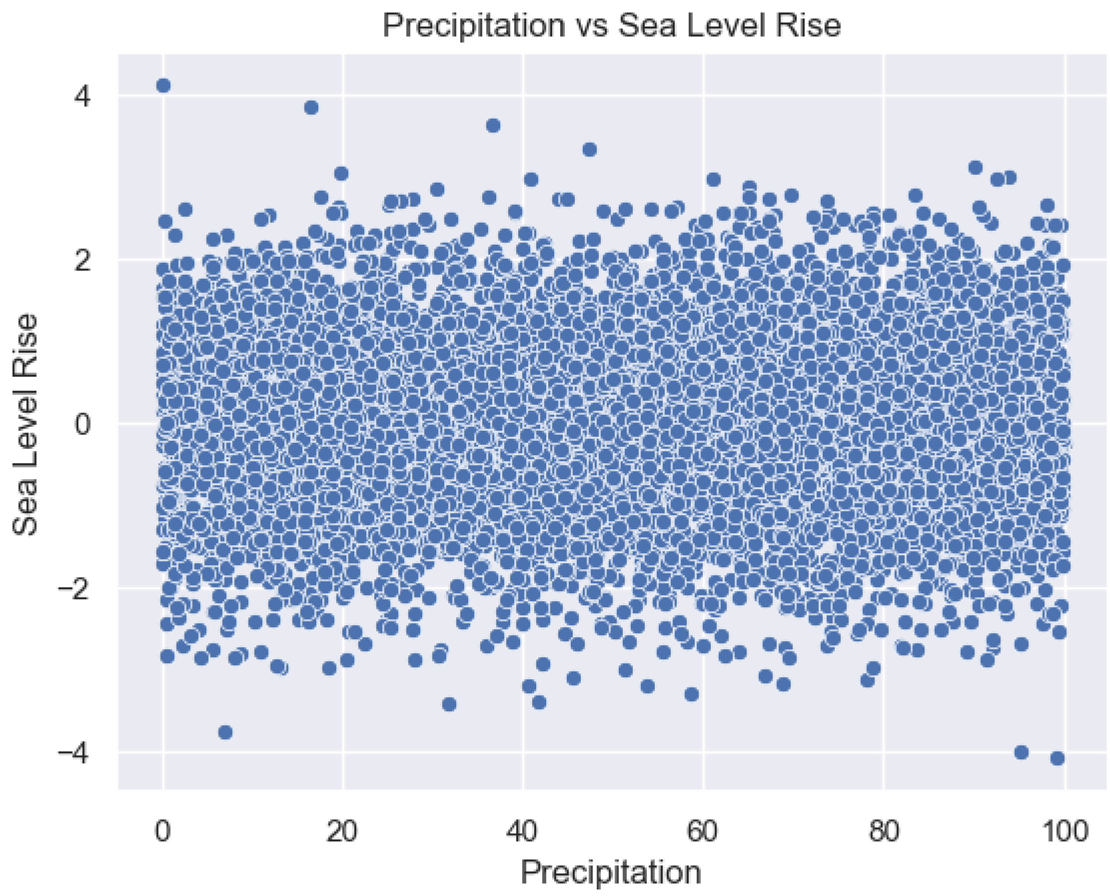
In [18]:

```
plt.figure(figsize=(10, 6))
sns.lineplot(x=df.index, y=df['Sea Level Rise'])
plt.xlabel('Year')
plt.ylabel('Sea Level Rise')
plt.title('Sea Level Rise Trends Over Time')
plt.show()
```



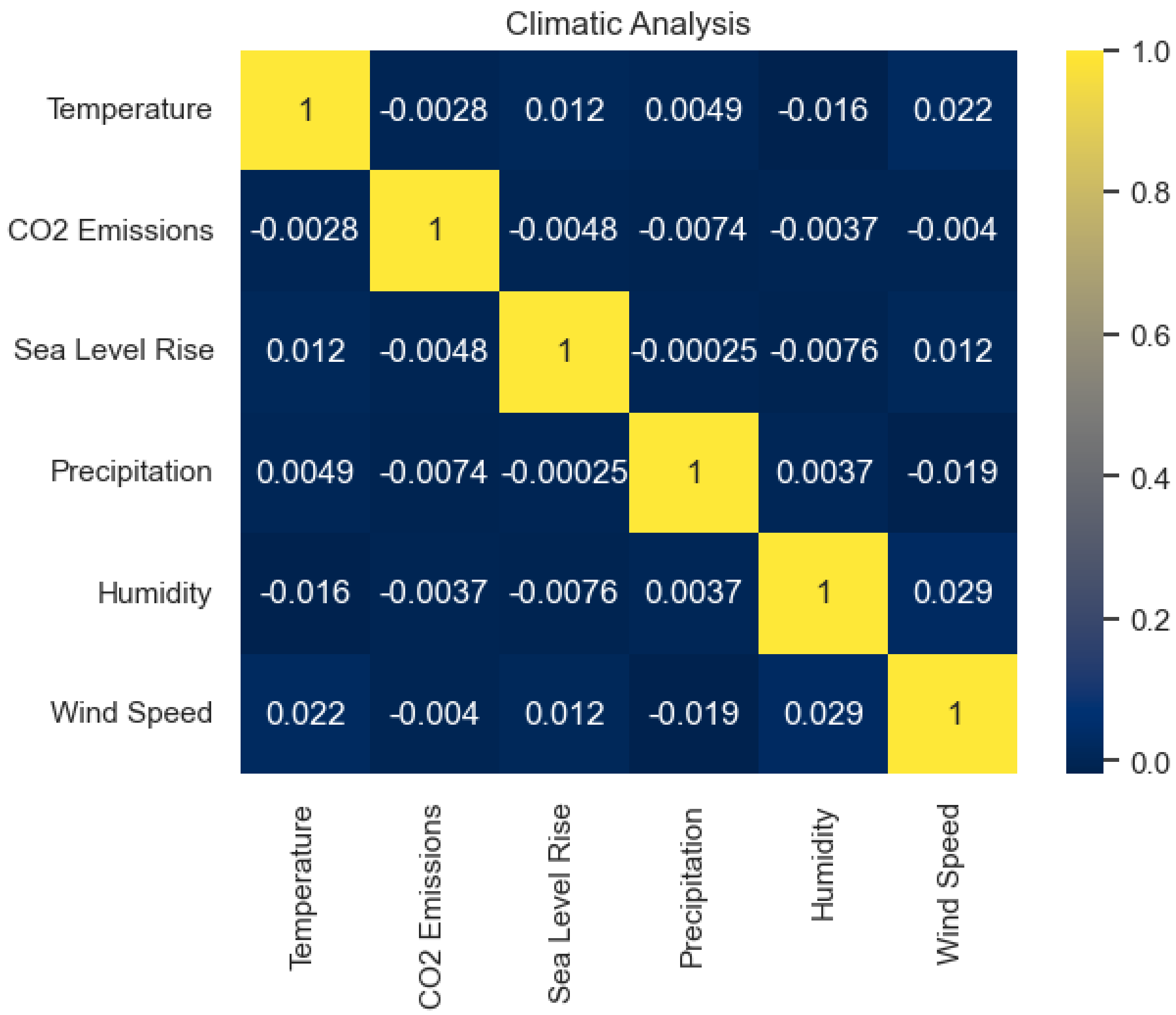
In [19]:

```
sns.scatterplot(x='Precipitation', y='Sea Level Rise', data=df)
plt.title('Precipitation vs Sea Level Rise')
plt.xlabel('Precipitation')
plt.ylabel('Sea Level Rise')
plt.show()
```



In [20]:

```
# Correlation Matrix
sns.heatmap(df.corr(), annot=True, cmap='cividis')
plt.title('Climatic Analysis')
plt.show()
```



Data Preprocessing

In [21]:

```
X = df[['Precipitation', 'Humidity', 'Wind Speed']]
y = df[['Temperature']]
```

In [22]:

```
# Splitting the Dataset
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```

Training the Model

- After training the model, the model can be able to predict the temperature according to the given climatic features (i.e., CO2 emissions, sea level rise, precipitation, wind speed, humidity, etc.)

In [23]:
```
from sklearn.linear_model import LinearRegression
model = LinearRegression()
```

In [24]:
```
model.fit(x_train, y_train)
```

Out[24]:
```
▾ LinearRegression
LinearRegression()
```

In [25]:
```
len(x_train), len(y_train)
```

Out[25]:
```
(8000, 8000)
```

In [26]:
```
len(x_test), len(y_test)
```

Out[26]:
```
(2000, 2000)
```

In [27]:
```
print('Accuracy of the model: ', model.score(x_test, y_test))
```

```
Accuracy of the model:  0.0010176356149836918
```

In [28]:
```
from sklearn.metrics import mean_absolute_error, mean_squared_error
y_pred = model.predict(x_test)
```

In [29]:
```
mae = mean_absolute_error(y_test, y_pred)
print(f'Mean Absolute Error: {mae}')
```

```
Mean Absolute Error: 3.999099614445476
```

In [30]:
```
mse = mean_squared_error(y_test, y_pred)
print(f'Mean Squared Error: {mse}')
```

```
Mean Squared Error: 25.185754765740786
```

In [31]:
```
rmse = np.sqrt(mse)
print(f'Root Mean Squared Error: {rmse}')
```

```
Root Mean Squared Error: 5.018541099337614
```

In [32]:
```
actual_predict= pd.DataFrame({
    'Actual values': y_test.values.flatten(),
    'Predicted values': y_pred.flatten()})
actual_predict.head()
```

Out[32]:

	Actual values	Predicted values
0	12.459229	15.095398
1	21.508186	14.960996
2	8.076335	14.936417
3	16.547310	14.933838
4	12.623254	14.809402

In [ ]:
```

```