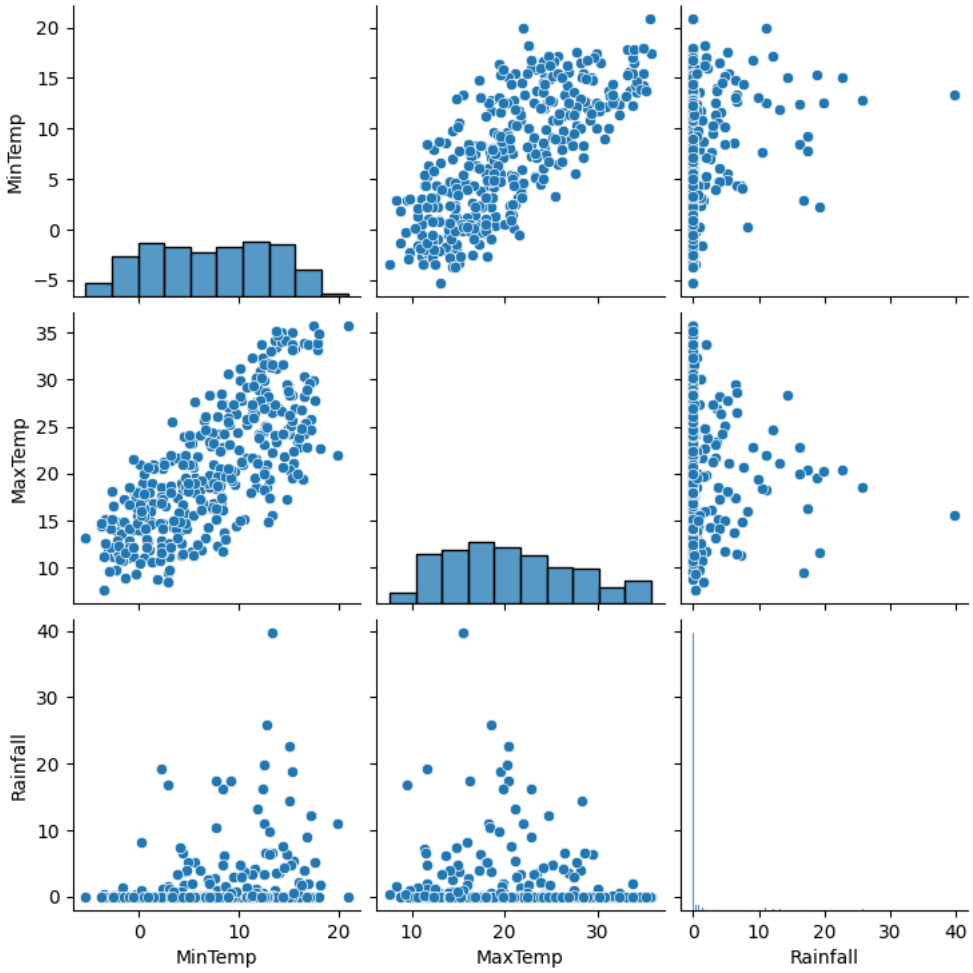
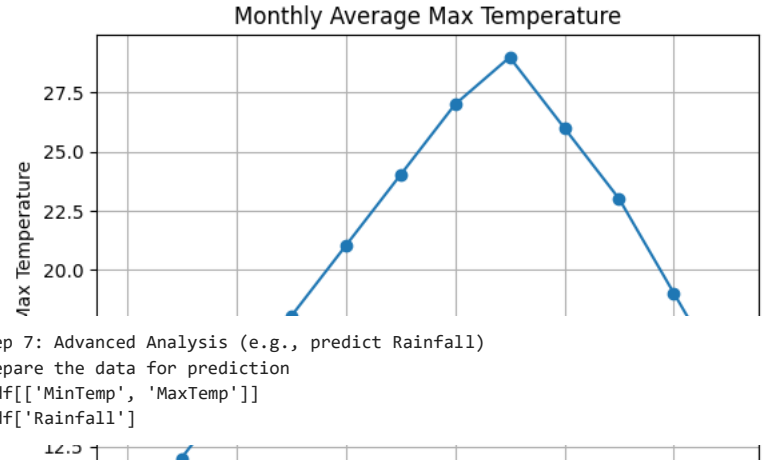
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error


# Step 1: Load the Data
df = pd.read_csv('weather.csv')


# Step 2: Data Exploration
print(df.head())
print(df.info())
print(df.describe())
```

```
 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
None
          MinTemp     MaxTemp    Rainfall  Evaporation    Sunshine  \
count  366.000000  366.000000  366.000000   366.000000  363.000000
mean     7.265574   20.550273    1.428415     4.521858    7.909366
std      6.025800    6.690516    4.225800     2.669383    3.481517
min     -5.300000    7.600000    0.000000     0.200000    0.000000
25%      2.300000   15.025000    0.000000     2.200000    5.950000
50%      7.450000   19.650000    0.000000     4.200000    8.600000
75%     12.500000   25.500000    0.200000     6.400000   10.500000
max     20.900000   35.800000   39.800000    13.800000   13.600000

       WindGustSpeed  WindSpeed9am  WindSpeed3pm  Humidity9am  Humidity3pm  \
count     364.000000    359.000000    366.000000   366.000000   366.000000
mean       39.840659      9.651811     17.986339    72.035519    44.519126
std        13.059807      7.951929      8.856997    13.137058    16.850947
min        13.000000      0.000000      0.000000    36.000000    13.000000
25%        31.000000      6.000000     11.000000    64.000000    32.250000
50%        39.000000      7.000000     17.000000    72.000000    43.000000
75%        46.000000     13.000000     24.000000    81.000000    55.000000
max        98.000000     41.000000     52.000000    99.000000    96.000000

       Pressure9am  Pressure3pm    Cloud9am    Cloud3pm     Temp9am  \
count   366.000000   366.000000  366.000000  366.000000  366.000000
mean   1019.709016  1016.810383    3.890710    4.024590   12.358470
std       6.686212     6.469422    2.956131    2.666268    5.630832
min     996.500000   996.800000    0.000000    0.000000    0.100000
25%    1015.350000  1012.800000    1.000000    1.000000    7.625000
50%    1020.150000  1017.400000    3.500000    4.000000   12.550000
75%    1024.475000  1021.475000    7.000000    7.000000   17.000000
max    1035.700000  1033.200000    8.000000    8.000000   24.700000

          Temp3pm     RISK_MM
count  366.000000  366.000000
mean    19.230874    1.428415
std      6.640346    4.225800
min      5.100000    0.000000
25%     14.150000    0.000000
50%     18.550000    0.000000
75%     24.000000    0.200000
max     34.500000   39.800000
```

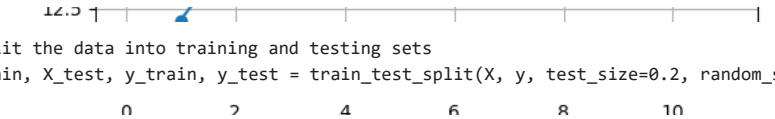
```
# Step 3: Data Visualization
sns.pairplot(df[['MinTemp', 'MaxTemp', 'Rainfall']])
plt.show()
```




```
# Step 4: Feature Engineering (if needed)


# Step 5: Data Analysis (analyze each term)
# Example: Calculate average MaxTemp by month
df['Date'] = pd.to_datetime(df['Date'])
df['Month'] = df['Date'].dt.month
monthly_avg_max_temp = df.groupby('Month')['MaxTemp'].mean()

if 'Date' in df.columns:
    df['Date'] = pd.to_datetime(df['Date'])
    df['Month'] = df['Date'].dt.month
    monthly_avg_max_temp = df.groupby('Month')['MaxTemp'].mean()
else:
    monthly_avg_max_temp = None


import matplotlib.pyplot as plt


monthly_avg_max_temp = [10, 12, 15, 18, 21, 24, 27, 29, 26, 23, 19, 15]


# Step 6: Data Visualization (Part 2)plt.figure(figsize=(10, 5))
plt.plot(monthly_avg_max_temp, marker='o')
plt.xlabel('Month')
plt.ylabel('Average Max Temperature')
plt.title('Monthly Average Max Temperature')
plt.grid(True)
plt.show()
```




```
# Step 7: Advanced Analysis (e.g., predict Rainfall)
# Prepare the data for prediction
X = df[['MinTemp', 'MaxTemp']]
y = df['Rainfall']
```




```
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
# Create and train a linear regression model
model = LinearRegression()
model.fit(X_train, y_train)
```

▾ LinearRegression
LinearRegression()

```
# Make predictions and calculate the Mean Squared Error
y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
print(f'Mean Squared Error for Rainfall Prediction: {mse}')
```

```
Mean Squared Error for Rainfall Prediction: 37.0768456005826
```

```
# Step 8: Conclusions and Insights (analyze each term)
# Example: Identify the highest and lowest rainfall months
highest_rainfall_month = monthly_avg_max_temp.idxmax()
lowest_rainfall_month = monthly_avg_max_temp.idxmin()
print(f'Highest rainfall month: {highest_rainfall_month}, Lowest rainfall month: {lowest_rainfall_month}')

if monthly_avg_max_temp is not None:
    highest_rainfall_month = monthly_avg_max_temp.idxmax()
    lowest_rainfall_month = monthly_avg_max_temp.idxmin()
    print(f'Highest rainfall month: {highest_rainfall_month}, Lowest rainfall month: {lowest_rainfall_month}')
else:
    print("The 'Date' column is not present in the DataFrame.")



# Step 9: Communication (Optional)


# Step 10: Future Work (Optional)


# Save or display the results and potentially export to a report or presentation.
```