```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
```

```
a = pd.read_csv("airquality.csv")
a
```

	Unnamed: 0	Ozone	Solar.R	Wind	Temp	Month	Day	Humidity
0	1	41.0	190.0	7.4	67	5	1	high
1	2	36.0	118.0	8.0	72	5	2	medium
2	3	12.0	149.0	12.6	74	5	3	low
3	4	18.0	313.0	11.5	62	5	4	medium
4	5	NaN	NaN	14.3	56	5	5	low
...	...	...	...	...	...	...	...	...
148	149	30.0	193.0	6.9	70	9	26	low
149	150	NaN	145.0	13.2	77	9	27	NaN
150	151	14.0	191.0	14.3	75	9	28	low
151	152	18.0	131.0	8.0	76	9	29	NaN
152	153	20.0	223.0	11.5	68	9	30	low

153 rows × 8 columns

Next steps: Generate code with a | View recommended plots | New interactive sheet

```
a.isnull().sum()
```

	0
Unnamed: 0	0
Ozone	37
Solar.R	7
Wind	0
Temp	0
Month	0
Day	0
Humidity	72

dtype: int64

```
# Fill missing numeric values with mean
a['Ozone'] = a['Ozone'].fillna(a['Ozone'].mean())
a['Solar.R'] = a['Solar.R'].fillna(a['Solar.R'].mean())
a['Temp'] = a['Temp'].fillna(a['Temp'].mean())
```

```
# Fill missing categorical values in 'Humidity' using mode
a['Humidity'] = a['Humidity'].fillna(a['Humidity'].mode()[0])
```

```
# Convert categorical 'Humidity' to numeric
encoder = LabelEncoder()
a['Humidity'] = encoder.fit_transform(a['Humidity'])
```

```
# Remove duplicate records
a.drop_duplicates(inplace=True)
```

```
a.head()
```

	Unnamed: 0	Ozone	Solar.R	Wind	Temp	Month	Day	Humidity
0	1	41.00000	190.000000	7.4	67	5	1	1
1	2	36.00000	118.000000	8.0	72	5	2	3
2	3	12.00000	149.000000	12.6	74	5	3	2
3	4	18.00000	313.000000	11.5	62	5	4	3
4	5	42.12931	185.931507	14.3	56	5	5	2

Next steps: Generate code with a | View recommended plots | New interactive sheet

```
# Removing invalid values
a = a[a['Wind'] >= 0]
a = a[(a['Month'] >= 1) & (a['Month'] <= 12)]
a = a[(a['Day'] >= 1) & (a['Day'] <= 31)]
```

```
a.head()
```

	Unnamed: 0	Ozone	Solar.R	Wind	Temp	Month	Day	Humidity
0	1	41.00000	190.000000	7.4	67	5	1	1
1	2	36.00000	118.000000	8.0	72	5	2	3
2	3	12.00000	149.000000	12.6	74	5	3	2
3	4	18.00000	313.000000	11.5	62	5	4	3
4	5	42.12931	185.931507	14.3	56	5	5	2

Next steps: Generate code with a | View recommended plots | New interactive sheet

```
# List of numerical columns to check for outliers
num_cols = ['Ozone', 'Solar.R', 'Wind', 'Temp']
```

```
for col in num_cols:
  Q1 = a[col].quantile(0.25)
  Q3 = a[col].quantile(0.75)
  IQR = Q3 - Q1
  lower_bound = Q1 - 1.5 * IQR
  upper_bound = Q3 + 1.5 * IQR
# Cap outliers
a[col] = a[col].clip(lower_bound, upper_bound)
a.head()
```

	Unnamed: 0	Ozone	Solar.R	Wind	Temp	Month	Day	Humidity
0	1	41.00000	190.000000	7.4	67	5	1	1
1	2	36.00000	118.000000	8.0	72	5	2	3
2	3	12.00000	149.000000	12.6	74	5	3	2
3	4	18.00000	313.000000	11.5	62	5	4	3
4	5	42.12931	185.931507	14.3	56	5	5	2

Next steps: Generate code with a | View recommended plots | New interactive sheet

```
# Define Features (X) and Target Variable (y)
X = a[['Solar.R', 'Wind', 'Temp', 'Humidity']]
y = a['Ozone']
```

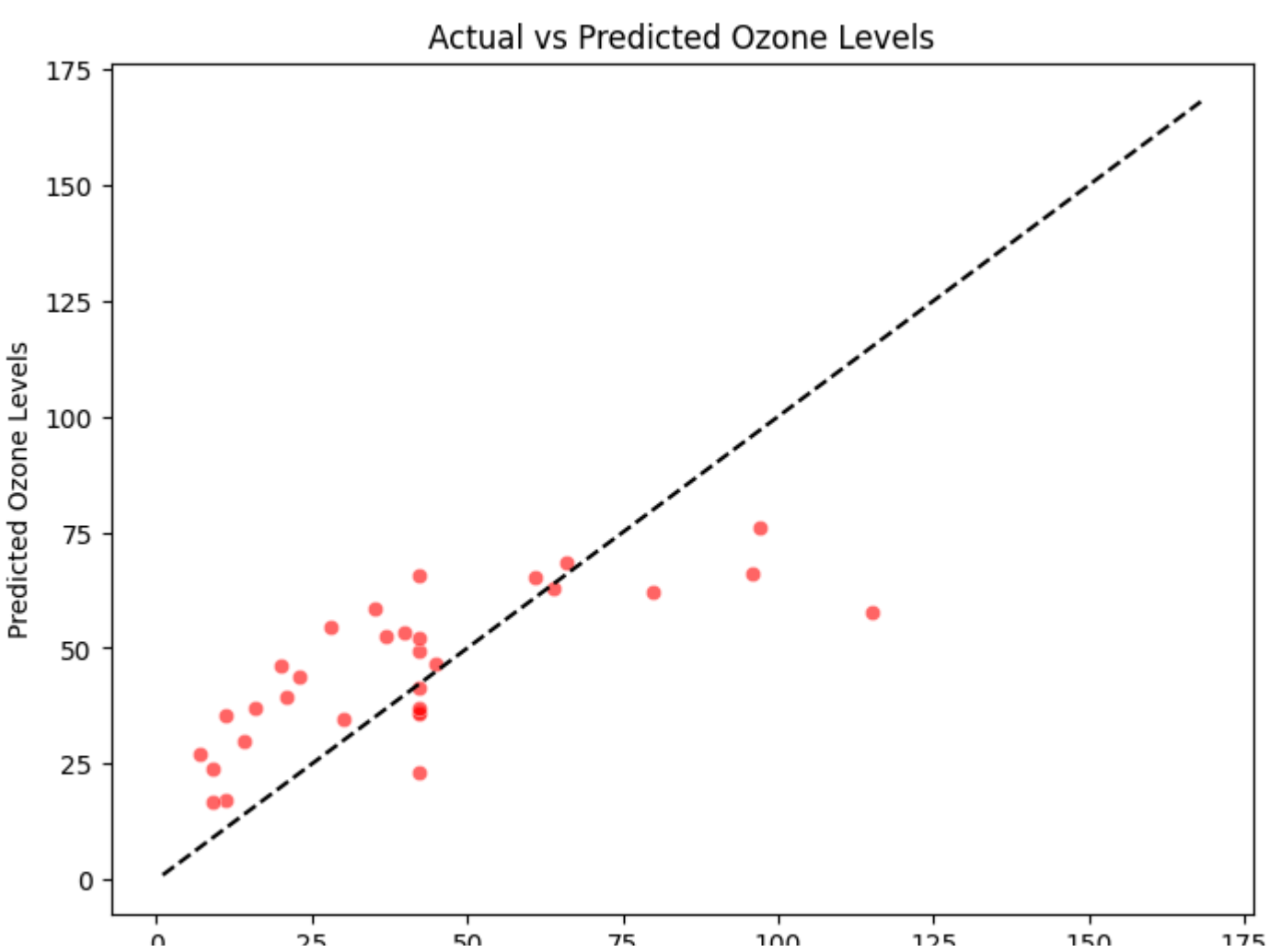
```
# Split Data into Training and Testing Sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
print( X_train.shape, "X_test shape:", X_test.shape)
```

```
(122, 4) X_test shape: (31, 4)
```

```
# Train a Linear Regression Model
model = LinearRegression()
model.fit(X_train, y_train)
# Make Predictions
y_pred = model.predict(X_test)
```

```
plt.figure(figsize=(8, 6))
sns.scatterplot(x=y_test, y=y_pred, color='red', alpha=0.6)
```

```
plt.plot([y.min(), y.max()], [y.min(), y.max()], color='black', linestyle='--')
plt.xlabel("Actual Ozone Levels")
plt.ylabel("Predicted Ozone Levels")
plt.title("Actual vs Predicted Ozone Levels")
plt.show()
```




```
#1. Model Evaluation
```

```
# Calculate errors
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_test, y_pred)
```

```
print(f"\n Model Performance:")
print(f" Mean Absolute Error (MAE): {mae:.2f}")
print(f" Mean Squared Error (MSE): {mse:.2f}")
print(f" Root Mean Squared Error (RMSE): {rmse:.2f}")
print(f" R² Score: {r2:.2f}")
```

```
 Model Performance:
 Mean Absolute Error (MAE): 15.20
 Mean Squared Error (MSE): 364.07
 Root Mean Squared Error (RMSE): 19.08
 R² Score: 0.50
```