Importing Libraries

```
import pandas as pd
import numpy as np
```

Reading the file

```
df = pd.read_csv('airquality.csv', encoding='cp1252')
```

```
df.head()
```

	Ozone	Solar.R	Wind	Temp	Month	Day	Humidity
0	41.0	190.0	7.4	67.0	5	1	High
1	36.0	118.0	8.0	72.0	5	2	Medium
2	12.0	149.0	12.6	74.0	5	3	low
3	18.0	313.0	11.5	62.0	5	4	Medium
4	NaN	NaN	14.3	56.0	5	5	Medium

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

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 153 entries, 0 to 152
Data columns (total 7 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Ozone     116 non-null    float64
 1   Solar.R   146 non-null    float64
 2   Wind      153 non-null    float64
 3   Temp      142 non-null    float64
 4   Month     153 non-null    int64
 5   Day       153 non-null    int64
 6   Humidity  141 non-null    object
dtypes: float64(4), int64(2), object(1)
memory usage: 8.5+ KB
```

```
df.columns
```

```
Index(['Ozone', 'Solar.R', 'Wind', 'Temp', 'Month', 'Day', 'Humidity'], dtype='object')
```

Data Cleaning

```
df=df.drop_duplicates()
```

```
df.isna().sum()
```

	0
Ozone	37
Solar.R	7
Wind	0
Temp	11
Month	0
Day	0
Humidity	12

dtype: int64

```
percent_missing = df.isnull().sum() * 100 / len(df)
```

```
percent_missing.sort_values(ascending=False)
```

	0
Ozone	24.183007
Humidity	7.843137
Temp	7.189542
Solar.R	4.575163
Wind	0.000000
Month	0.000000
Day	0.000000

dtype: float64

```
df.head()
```

	Ozone	Solar.R	Wind	Temp	Month	Day	Humidity
0	41.0	190.0	7.4	67.0	5	1	High
1	36.0	118.0	8.0	72.0	5	2	Medium
2	12.0	149.0	12.6	74.0	5	3	low
3	18.0	313.0	11.5	62.0	5	4	Medium
4	NaN	NaN	14.3	56.0	5	5	Medium

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

```
df.columns
```

```
Index(['Ozone', 'Solar.R', 'Wind', 'Temp', 'Month', 'Day', 'Humidity'], dtype='object')
```

```
for col in df.columns:
    if df[col].dtype == 'object' or df[col].dtype == 'string':
        df[col] = df[col].fillna(df[col].mode()[0])
    else:
        df[col] = df[col].fillna(df[col].mean())
```

```
df.isna().sum()
```

	0
Ozone	0
Solar.R	0
Wind	0
Temp	0
Month	0
Day	0
Humidity	0

dtype: int64

```
df
```

	Ozone	Solar.R	Wind	Temp	Month	Day	Humidity
0	41.00000	190.000000	7.4	67.0	5	1	High
1	36.00000	118.000000	8.0	72.0	5	2	Medium
2	12.00000	149.000000	12.6	74.0	5	3	low
3	18.00000	313.000000	11.5	62.0	5	4	Medium
4	42.12931	185.931507	14.3	56.0	5	5	Medium
...	...	...	...	...	...	...	...
148	30.00000	193.000000	6.9	70.0	9	26	low
149	42.12931	145.000000	13.2	77.0	9	27	low
150	14.00000	191.000000	14.3	75.0	9	28	low
151	18.00000	131.000000	8.0	76.0	9	29	Medium
152	20.00000	223.000000	11.5	-30.0	9	30	low

153 rows × 7 columns

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

```
df.isna().sum()
```

	0
Ozone	0
Solar.R	0
Wind	0
Temp	0
Month	0
Day	0
Humidity	0

dtype: int64

Data integration

```
subSet1 = df[['Ozone', 'Temp']]
subSet2 = df[['Month','Humidity']]
```

```
subSet1.head()
```

	Ozone	Temp
0	31	13
1	27	18
2	8	21
3	12	8
4	32	3

Next steps: Generate code with subSet1 | View recommended plots | New interactive sheet

```
subSet2.head()
```

	Month	Humidity
0	0	0
1	0	1
2	0	2
3	0	1
4	0	1

Next steps: Generate code with subSet2 View recommended plots New interactive sheet

```
concatenated_df = pd.concat([subSet1, subSet2], axis=1)
```

```
concatenated_df
```

	Ozone	Temp	Month	Humidity
0	31	13	0	0
1	27	18	0	1
2	8	21	0	2
3	12	8	0	1
4	32	3	0	1
...	...	...	...	...
148	22	16	4	2
149	32	24	4	2
150	10	22	4	2
151	12	23	4	1
152	14	2	4	2

153 rows × 4 columns

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

Error Correcting

```
def remove_outliers(column):
    Q1 = column.quantile(0.25)
    Q3 = column.quantile(0.75)
    IQR = Q3 - Q1
    threshold = 1.5 * IQR
    outlier_mask = (column < Q1 - threshold) | (column > Q3 + threshold)
    return column[~outlier_mask]
```

```
df.columns
```

```
Index(['Ozone', 'Solar.R', 'Wind', 'Temp', 'Month', 'Day', 'Humidity'], dtype='object')
```

```
import seaborn as sns
import matplotlib.pyplot as plt
```

Data Transform

```
from sklearn.preprocessing import LabelEncoder
```

```
col_label= ['Humidity','type']
```

```
encoder = LabelEncoder()
```

```
for col in df.columns:

        df[col] = encoder.fit_transform(df[col])
```

```
df
```

	Ozone	Solar.R	Wind	Temp	Month	Day	Humidity
0	31	57	11	13	0	0	0
1	27	35	12	18	0	1	1
2	8	45	20	21	0	2	2
3	12	111	18	8	0	3	1
4	32	52	23	3	0	4	1
...	...	...	...	...	...	...	...
148	22	60	10	16	4	25	2
149	32	43	21	24	4	26	2
150	10	58	23	22	4	27	2
151	12	38	12	23	4	28	1
152	14	71	18	2	4	29	2

153 rows × 7 columns

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

Model Building

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
```

```
type_index = df.columns[-1]

X = df.drop(columns=[type_index])
y = df[type_index]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
model = LogisticRegression()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"Model Accuracy: {accuracy}")
```