```
import pandas as pd  # Import pandas so we can read the CSV and analyze data in tables

df = pd.read_csv("AirQuality_Daily_StudentVersion.csv")  # Load the client CSV into a dataframe called df

df['date'] = pd.to_datetime(df['date'], errors='coerce')  # Convert the date column into real datetime values safely
df = df.dropna()  # Remove rows with missing values so calculations work correctly

summary = df.groupby('sensor.name')[['voc', 'pm2.5_atm', 'pm10.0_atm']].agg(['mean', 'median'])  # Group by sensor and compute mean/median for VOC, PM2.5, PM10

top5_mean_voc = summary.sort_values(('voc', 'mean'), ascending=False).head(5)  # Get top 5 sensors by highest mean VOC
top5_median_voc = summary.sort_values(('voc', 'median'), ascending=False).head(5)  # Get top 5 sensors by highest median VOC

top5_mean_pm25 = summary.sort_values(('pm2.5_atm', 'mean'), ascending=False).head(5)  # Get top 5 sensors by highest mean PM2.5
top5_median_pm25 = summary.sort_values(('pm2.5_atm', 'median'), ascending=False).head(5)  # Get top 5 sensors by highest median PM2.5

top5_mean_pm10 = summary.sort_values(('pm10.0_atm', 'mean'), ascending=False).head(5)  # Get top 5 sensors by highest mean PM10
top5_median_pm10 = summary.sort_values(('pm10.0_atm', 'median'), ascending=False).head(5)  # Get top 5 sensors by highest median PM10

print("TOP 5 LOCATIONS (SENSORS) BY MEAN VOC\n")  # Print a label so the output is clear
print(top5_mean_voc)  # Print the top 5 mean VOC table

print("\nTOP 5 LOCATIONS (SENSORS) BY MEDIAN VOC\n")  # Print a label so the output is clear
print(top5_median_voc)  # Print the top 5 median VOC table

print("\nTOP 5 LOCATIONS (SENSORS) BY MEAN PM2.5\n")  # Print a label so the output is clear
print(top5_mean_pm25)  # Print the top 5 mean PM2.5 table

print("\nTOP 5 LOCATIONS (SENSORS) BY MEDIAN PM2.5\n")  # Print a label so the output is clear
print(top5_median_pm25)  # Print the top 5 median PM2.5 table

print("\nTOP 5 LOCATIONS (SENSORS) BY MEAN PM10\n")  # Print a label so the output is clear
print(top5_mean_pm10)  # Print the top 5 mean PM10 table

print("\nTOP 5 LOCATIONS (SENSORS) BY MEDIAN PM10\n")  # Print a label so the output is
clear
print(top5_median_pm10)  # Print the top 5 median PM10 table

max_pm25 = df.loc[df['pm2.5_atm'].idxmax()]  # Select the row that contains the maximum
PM2.5 value
max_pm10 = df.loc[df['pm10.0_atm'].idxmax()]  # Select the row that contains the maximum
PM10 value
max_voc = df.loc[df['voc'].idxmax()]  # Select the row that contains the maximum VOC value

print("\nMAXIMUM PM2.5 EVENT")  # Print a label for the max PM2.5 event
print(max_pm25[['sensor.name', 'date', 'pm2.5_atm']])  # Print the sensor name, date, and
maximum PM2.5 value

print("\nMAXIMUM PM10 EVENT")  # Print a label for the max PM10 event
print(max_pm10[['sensor.name', 'date', 'pm10.0_atm']])  # Print the sensor name, date, and
maximum PM10 value

print("\nMAXIMUM VOC EVENT")  # Print a label for the max VOC event
print(max_voc[['sensor.name', 'date', 'voc']])  # Print the sensor name, date, and maximum VOC
value

def humidity_category(h):  # Create a function to turn humidity numbers into categories
    if h < 50:  # Check if humidity is less than 50%
        return "Low"  # Return the category name Low
    elif h <= 80:  # Check if humidity is between 50% and 80%
        return "High"  # Return the category name High
    else:  # If humidity is above 80%
        return "Very High"  # Return the category name Very High

df['humidity_level'] = df['humidity'].apply(humidity_category)  # Apply the humidity categories to
every row

df['temperature_F'] = (df['temperature'] * 9/5) + 32  # Convert temperature from Celsius to
Fahrenheit to match the client categories

def temp_category(t):  # Create a function to turn Fahrenheit temperature into categories
    if t < 32:  # Check if temperature is below 32°F
        return "Below Freezing"  # Return the category Below Freezing
    elif t <= 50:  # Check if temperature is between 32°F and 50°F
        return "Cool"  # Return the category Cool
    elif t <= 70:  # Check if temperature is between 51°F and 70°F
```

```
        return "Warm"  # Return the category Warm
    else:  # If temperature is greater than 70°F
        return "Hot"  # Return the category Hot

df['temp_level'] = df['temperature_F'].apply(temp_category)  # Apply the temperature categories
to every row

humidity_summary = df.groupby('humidity_level')[['pm2.5_atm', 'pm10.0_atm', 'voc']].mean()  #
Compute average pollution values by humidity group
print("\nAVERAGE POLLUTION BY HUMIDITY CATEGORY\n")  # Print a label for the humidity
summary
print(humidity_summary)  # Print the humidity summary table

temp_summary = df.groupby('temp_level')[['pm2.5_atm', 'pm10.0_atm', 'voc']].mean()  #
Compute average pollution values by temperature group
print("\nAVERAGE POLLUTION BY TEMPERATURE CATEGORY\n")  # Print a label for the
temperature summary
print(temp_summary)  # Print the temperature summary table

unhealthy_pm25 = df[df['pm2.5_atm'] >= 35.5]  # Filter rows where PM2.5 is unhealthy for
sensitive groups (AQI-based threshold)
unhealthy_pm10 = df[df['pm10.0_atm'] >= 155]  # Filter rows where PM10 is unhealthy for
sensitive groups (AQI-based threshold)

aqi_events = pd.concat([unhealthy_pm25, unhealthy_pm10])  # Combine PM2.5 and PM10
unhealthy events into one table

print("\nAQI HEALTH RISK EVENTS (UNHEALTHY FOR SENSITIVE GROUPS)\n")  # Print a
label for AQI events
print(aqi_events[['sensor.name', 'date', 'pm2.5_atm', 'pm10.0_atm']])  # Print the sensor name,
date, and PM values for AQI events

altitude_corr = df[['sensor.altitude', 'pm2.5_atm', 'pm10.0_atm', 'voc']].corr()  # Calculate
correlation between altitude and pollutants
print("\nCORRELATION BETWEEN ALTITUDE AND POLLUTANTS\n")  # Print a label for
altitude correlation
print(altitude_corr)  # Print the correlation table

top5_mean_voc.to_csv("top5_mean_voc.csv")  # Save the top 5 mean VOC table to a CSV file
top5_median_voc.to_csv("top5_median_voc.csv")  # Save the top 5 median VOC table to a
CSV file

top5_mean_pm25.to_csv("top5_mean_pm25.csv")  # Save the top 5 mean PM2.5 table to a
CSV file
```

```
top5_median_pm25.to_csv("top5_median_pm25.csv")  # Save the top 5 median PM2.5 table to a CSV file

top5_mean_pm10.to_csv("top5_mean_pm10.csv")  # Save the top 5 mean PM10 table to a CSV file
top5_median_pm10.to_csv("top5_median_pm10.csv")  # Save the top 5 median PM10 table to a CSV file

aqi_events.to_csv("aqi_health_events.csv")  # Save AQI health risk events to a CSV file
```