CIVE 202 – Civil Engineering Analysis 2

Spring 2026

Johnathan Sevick

Project 1 Annotated Coding Document

Python Code	Annotations
import pandas as pd air_data = pd.read_csv('AirQuality_Daily_StudentVersion.csv')	These lines of code imports “pandas” using the import function. Then a CSV document is uploaded to the environment using the pd.read_csv function

air_data = pd.DataFrame(air_data)
air_data.head()
	These lines of code set the imported CSV document as a data frame using the pd.DataFrame function before it is printed to see what column headers exist within with use of the .head function.

voc_data1=air_data.groupby('sensor.name').agg(Mean_voc =('voc','mean'))	This line groups the data by sensor name using the .groupby function. Then the .agg function is used to create a new column “Mean_voc” with the mean voc values.
task1_voc_mean=voc_data1.sort_values(by = 'Mean_voc', ascending = False).head(5) task1_voc_mean	This section then takes the data from the last and sorts it in descending order with the .sort_values function and only shows the first 5 lines with the .head function. It is then printed out to show the top 5 sensor locations with the highest mean voc values.
voc_data2=air_data.groupby('sensor.name').agg(Median_voc =('voc','median'))	This line groups the data by sensor name using the .groupby function. Then the .agg function is used to create a new column “Median_voc” with the median voc values.
task1_voc_median=voc_data2.sort_values(by = 'Median_voc', ascending = False).head(5) task1_voc_median	This section then takes the data from the last and sorts it in descending order with the .sort_values function and only shows the first 5 lines with the .head function. It is then printed out to show the top 5 sensor locations with the highest median voc values.

pm25_data1=air_data.groupby('sensor.name').agg(Mean_pm25 =('pm2.5_atm','mean'))	This line groups the data by sensor name using the .groupby function. Then the .agg function is used to create a new column "Mean_pm25" with the mean pm2.5 values.
task1_pm25_mean=pm25_data1.sort_values(by = 'Mean_pm25', ascending = False).head(5) task1_pm25_mean	This section then takes the data from the last and sorts it in descending order with the .sort_values function and only shows the first 5 lines with the .head function. It is then printed out to show the top 5 sensor locations with the highest mean pm2.5 values.
pm25_data2=air_data.groupby('sensor.name').agg(Median_pm25 =('pm2.5_atm','median'))	This line groups the data by sensor name using the .groupby function. Then the .agg function is used to create a new column "Median_pm25" with the median pm2.5 values.
task1_pm25_median=pm25_data2.sort_values(by = 'Median_pm25', ascending = False).head(5) task1_pm25_median	This section then takes the data from the last and sorts it in descending order with the .sort_values function and only shows the first 5 lines with the .head function. It is then printed out to show the top 5 sensor locations with the highest median pm2.5 values.
pm10_data1=air_data.groupby('sensor.name').agg(Mean_pm10 =('pm10.0_atm','mean'))	This line groups the data by sensor name using the .groupby function. Then the .agg function is used to create a new column "Mean_pm10" with the mean pm10.0 values.
task1_pm10_mean=pm10_data1.sort_values(by = 'Mean_pm10', ascending = False).head(5) task1_pm10_mean	This section then takes the data from the last and sorts it in descending order with the .sort_values function and only shows the first 5 lines with the .head function. It is then printed out to show the top 5 sensor locations with the highest mean pm10.0 values.
pm10_data2=air_data.groupby('sensor.name').agg(Median_pm10 =('pm10.0_atm','median'))	This line groups the data by sensor name using the .groupby function. Then the .agg function is used to create a new column

	“Median_pm10” with the median pm10.0 values.
task1_pm10_median=pm10_data2.sort_values(by = 'Median_pm10', ascending = False).head(5) task1_pm10_median	This section then takes the data from the last and sorts it in descending order with the .sort_values function and only shows the first 5 lines with the .head function. It is then printed out to show the top 5 sensor locations with the highest median pm10.0 values.
voc_data3 = air_data.loc[air_data.groupby('date')['voc'].idxmax()]	This next line takes the data and groups it by the date with the .groupby function. The .idxmax function is then used to grab the ID of the max voc values
task2_voc_max=voc_data3.sort_values(by ='voc',ascending = False).head(5) task2_voc_max[['date', 'sensor.name', 'voc']]	Using the data from the last line, the .sort_values function sorts the max voc values in descending order and takes the top 5 highest maxes using the .head function. After that it prints it out to show the date, sensor name and max voc values in addition to the id.
pm25_data3 = air_data.loc[air_data.groupby('date')['pm2.5_atm'].idxmax()]	This next line takes the data and groups it by the date with the .groupby function. The .idxmax function is then used to grab the ID of the max pm2.5 values
task2_pm25_max = pm25_data3.sort_values(by='pm2.5_atm', ascending=False).head(5) task2_pm25_max[['date', 'sensor.name', 'pm2.5_atm']]	Using the data from the last line, the .sort_values function sorts the max pm2.5 values in descending order and takes the top 5 highest maxes using the .head function. After that it prints it out to show the date, sensor name and max pm2.5 values in addition to the id.
pm10_data3 = air_data.loc[air_data.groupby('date')['pm10.0_atm'].idxmax()]	This next line takes the data and groups it by the date with the .groupby function. The .idxmax function is then used to grab the ID of the max pm10.0 values
task2_pm10_max = pm10_data3.sort_values(by='pm10.0_atm', ascending=False).head(5)	Using the data from the last line, the .sort_values function sorts the max pm10.0 values in descending order and takes the top

task2_pm10_max[['date', 'sensor.name', 'pm10.0_atm']]	5 highest maxes using the .head function. After that it prints it out to show the date, sensor name and max pm10.0 values in addition to the id.

def temp_cat(t):
  if t < 32:
    return 'Below Freezeing'
  elif 32 <= t <= 50:
    return 'Cool'
  elif 51 <= t <= 70:
    return 'Warm'
  else:
    return 'Hot'
	This section defines a variable “temp_cat” using def to define a function, it looks at the input and groups it into one of the following temperature levels: below freezing, cool, warm, or hot.

def humid_cat(h):
  if h < 50:
    return "Low Humidity"
  elif h < 80:
    return "High Humidity"
  else:
    return "Very High Humidity"
	This section defines a variable “humid_cat” using def to define a function, it looks at the input and groups it into one of the following humidity levels: low humidity, high humidity, or very high humidity.

air_data['temperature.level'] = air_data['temperature'].apply(temp_cat)	This line uses the .apply function to apply the previously defined temp_cat function. It looks at the values in the “temperature” column and puts the assigned category in a new column called “temperature.level”.
air_data['humidity.level'] = air_data['humidity'].apply(humid_cat)	This line uses the .apply function to apply the previously defined humid_cat function. It looks at the values in the “humidity” column and puts the assigned category in a new column called “humidity.level”.

temp_data1=air_data.groupby('temperature.level').agg(voc_mean
= ('voc','mean'),
                        pm25_mean = ('pm2.5_atm','mean'),
                        pm10_mean = ('pm10.0_atm','mean'))
	This line uses the .groupby function to group by “temperature.level”. Then using the .agg function it creates a new column for the mean values of voc, pm2.5, and pm10.0 of each temperature level.

temp_data1	After the previous line, this line prints out the data showing the mean values of voc, pm2.5, and pm10.0 of each temperature level.

humid_data1=air_data.groupby('humidity.level').agg(voc_mean = ('voc','mean'),
pm25_mean = ('pm2.5_atm','mean'),
pm10_mean = ('pm10.0_atm','mean'))
	This line uses the .groupby function to group by “humidity.level”. Then using the .agg function it creates a new column for the mean values of voc, pm2.5, and pm10.0 of each temperature level.mean values of voc, pm2.5, and pm10.0 of each humidity level.

humid_data1	After the previous line, this line prints out the data showing the mean values of voc, pm2.5, and pm10.0 of each humidity level.
unhealthy_check_pm25 = (air_data[(air_data['pm2.5_atm'] >= 35.5)&(air_data['pm2.5_atm'] <= 55.4)].groupby('sensor.name').agg(unhealthy_dates=('date', list), unhealthy_count=('date', 'count')))	This single line first filters the pm2.5 values to find the ones in the “unhealthy for sensitive groups’ category as defined by the EPA’s AQI ratings (greater than or equal to 35.5 and less than or equal to 55.4). Then using the .groupby function, it groups them by the sensor name. Finally, is uses the .agg function to create a column called “unhealth_dates” to show all the dates where the pm2.5 values fell in range and creates column “unhealthy_count” to show how many time the pm2.5 values fell in range.
unhealthy_check_pm25	Following up the prior line, this line prints the data displaying the sensor name, how many time the pm2.5 values fell in range and the dates those occurred on.

unhealthy_check_pm10 = (air_data[(air_data['pm10.0_atm'] >= 155)&(air_data['pm10.0_atm'] <= 254)].groupby('sensor.name').agg(unhealthy_dates=('date', list),
 	
unhealthy_count=('date', 'count')))
	This single line first filters the pm10.0 values to find the ones in the “unhealthy for sensitive groups’ category as defined by the EPA’s AQI ratings (greater than or equal to 155 and less than or equal to 254). Then using the .groupby function, it groups them by the sensor name. Finally, is uses the .agg function to create a column called “unhealth_dates” to show all the dates where the pm10.0 values fell in range and creates column “unhealthy_count” to show how many time the pm10.0 values fell in range.

unhealthy_check_pm10	Following up the prior line, this line prints the data displaying the sensor name, how many time the pm10.0 values fell in range and the dates those occurred on.