df = pd.read\_csv("C:\Users\inees\Downloads\Data.csv")

df.head()

34/2:

import pandas as pd

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

df.head()

34/3:

import pandas as pd

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

df.groupby( by = "state").mean()

34/4:

import pandas as pd

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

df.groupby( by = "State")["PM2.5"]".mean()

34/5:

import pandas as pd

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

df.groupby( by = "State")["PM2.5"].mean()

34/6:

import pandas as pd

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

df.groupby( by = "state")["PM2.5"].mean()

34/7:

import pandas as pd

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

avg\_pm = df.groupby( by = "state")["PM2.5"].mean()

idm = avg\_pm.idmax()

idm

34/8:

import pandas as pd

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

avg\_pm = df.groupby( by = "state")["PM2.5"].mean()

idm = avg\_pm.idxmax()

idm

34/9:

import pandas as pd

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

print(df.head)

avg\_pm = df.groupby( by = "state")["PM2.5"].mean()

idm = avg\_pm.idxmax()

idm

34/10:

import pandas as pd

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

print(df.head)

avg\_pm = df.groupby( by = "state")["PM2.5"].mean()

idm = avg\_pm.idxmax()

idm

34/11:

import pandas as pd

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

print(df.head)

avg\_pm = df.groupby( by = "state")["PM2.5"].mean()

idm = avg\_pm.idxmax()

print(idm)

34/12:

import pandas as pd

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

print(df.head)

# avg\_pm = df.groupby( by = "state")["PM2.5"].mean()

# idm = avg\_pm.idxmax()

# print(idm)

34/13:

import pandas as pd

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

df.head

# avg\_pm = df.groupby( by = "state")["PM2.5"].mean()

# idm = avg\_pm.idxmax()

# print(idm)

34/14:

import pandas as pd

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

df.head()

# avg\_pm = df.groupby( by = "state")["PM2.5"].mean()

# idm = avg\_pm.idxmax()

# print(idm)

34/15:

import pandas as pd

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

where?

# avg\_pm = df.groupby( by = "state")["PM2.5"].mean()

# idm = avg\_pm.idxmax()

# print(idm)

34/16:

import pandas as pd

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

pd.where?

# avg\_pm = df.groupby( by = "state")["PM2.5"].mean()

# idm = avg\_pm.idxmax()

# print(idm)

34/17:

import pandas as pd

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

avg\_pm = df.groupby( by = "state")["PM2.5"].mean()

idxm = avg\_pm.idxmax()

m = avg\_pm.max()

print(f"{idxm} has the highest average PM2.5 concentration (of {max} )across all stations and across all years.")

34/18:

import pandas as pd

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

avg\_pm = df.groupby( by = "state")["PM2.5"].mean()

idxm = avg\_pm.idxmax()

m = avg\_pm.max()

print(f"{idxm} has the highest average PM2.5 concentration (of {m} )across all stations and across all years.")

34/19:

import pandas as pd

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

avg\_pm = df.groupby( by = "state")["PM2.5"].mean()

idxm = avg\_pm.idxmax()

m = avg\_pm.max()

print(f"{idxm} has the highest average PM2.5 concentration (of {m}.3f )across all stations and across all years.")

34/20:

import pandas as pd

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

avg\_pm = df.groupby( by = "state")["PM2.5"].mean()

idxm = avg\_pm.idxmax()

m = avg\_pm.max()

print(f"{idxm} has the highest average PM2.5 concentration (of {m:.3f} )across all stations and across all years.")

34/21:

import pandas as pd

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

avg\_pm = df.groupby( by = "state")["PM2.5"].mean()

idxm = avg\_pm.idxmax()

m = avg\_pm.max()

print(f"{idxm} has the highest average PM2.5 concentration (of {m:.3f})across all stations and across all years.")

34/22:

import pandas as pd

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

avg\_pm = df.groupby( by = "state")["PM2.5"].mean()

idxm = avg\_pm.idxmax()

m = avg\_pm.max()

print(f"{idxm} has the highest average PM2.5 concentration (of {m:.3f}) across all stations and across all years.")

34/23:

import pandas as pd

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

avg\_pm = df.groupby( by = "state")["PM2.5"].mean()

idxm = avg\_pm.idxmax()

m = avg\_pm.max()

print(f"{idxm} has the highest average PM2.5 concentration (of {m:.3f}) across all stations and across all years.")

34/24:

import pandas as pd

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

avg\_pm = df.groupby( by = "state")["PM2.5"].mean()

idxm = avg\_pm.idxmax()

m = avg\_pm.max()

print(f"{idxm} has the highest average PM2.5 concentration (of {m:.3f}) across all stations and across all years.")

34/25:

import pandas as pd

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

avg\_pm = df.groupby( by = "state")["PM2.5"].mean()

idxm = avg\_pm.idxmax() # Get the state w highest avg

m = avg\_pm.max() # Get the value of the hight avg PM2.5 Conc

print(f"{idxm} has the highest average PM2.5 concentration (of {m:.3f}) across all stations and across all years.")

34/26:

import pandas as pd

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

# Avg PM2.5 for each state

avg\_pm = df.groupby( by = "state")["PM2.5"].mean()

idxm = avg\_pm.idxmax() # Get the state w highest avg

m = avg\_pm.max() # Get the value of the hight avg PM2.5 Conc

print(f"{idxm} has the highest average PM2.5 concentration (of {m:.3f}) across all stations and across all years.")

34/27:

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

df.head()

34/28:

import pandas as pd

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

# Avg PM2.5 for each state

df\_dropped = df.dropna(subset=["PM2.5"]) # Drop rows w PM2.5 missing

avg\_pm = df\_dropped.groupby( by = "state")["PM2.5"].mean()

idxm = avg\_pm.idxmax() # Get the state w highest avg

m = avg\_pm.max() # Get the value of the hight avg PM2.5 Conc

print(f"{idxm} has the highest average PM2.5 concentration (of {m:.3f}) across all stations and across all years.")

34/29:

import pandas as pd

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

# Get rows for year 2023

df\_2023 = df[df["Timestamp"].dt.year == 2023]

# Drop rows w PM2.5 missing

df\_2023 = df\_2023.dropna(subset=["PM2.5"])

# hazardous PM2.5 levels

hazardous\_data = df\_2023[df\_2023["PM2.5"] > 300]

hazardous\_days\_per\_state = hazardous\_data.groupby("State")["Timestamp"].count()

# Find the state with the highest number of hazardous days

most\_hazardous\_state = hazardous\_days\_per\_state.idxmax()

most\_hazardous\_days = hazardous\_days\_per\_state.max()

print(f"The state with the most days having hazardous PM2.5 levels in 2023 is {most\_hazardous\_state} with {most\_hazardous\_days} days.")

34/30:

import pandas as pd

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

# Get rows for year 2023

df\_2023 = df[df["Timestamp"].dt.year == 2023]

# Drop rows w PM2.5 missing

df\_2023 = df\_2023.dropna(subset=["PM2.5"])

# hazardous PM2.5 levels

hazardous\_data = df\_2023[df\_2023["PM2.5"] > 300]

hazardous\_days\_per\_state = hazardous\_data.groupby("State")["Timestamp"].count()

# Find the state with the highest number of hazardous days

most\_hazardous\_state = hazardous\_days\_per\_state.idxmax()

most\_hazardous\_days = hazardous\_days\_per\_state.max()

print(f"The state with the most days having hazardous PM2.5 levels in 2023 is {most\_hazardous\_state} with {most\_hazardous\_days} days.")

34/31:

import pandas as pd

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

# Get rows for year 2023

# Convert "Timestamp" column to datetime format

df["Timestamp"] = pd.to\_datetime(df["Timestamp"], format="%y-%m-%d")

df\_2023 = df[df["Timestamp"].dt.year == 2023]

# Drop rows w PM2.5 missing

df\_2023 = df\_2023.dropna(subset=["PM2.5"])

# hazardous PM2.5 levels

hazardous\_data = df\_2023[df\_2023["PM2.5"] > 300]

hazardous\_days\_per\_state = hazardous\_data.groupby("State")["Timestamp"].count()

# Find the state with the highest number of hazardous days

most\_hazardous\_state = hazardous\_days\_per\_state.idxmax()

most\_hazardous\_days = hazardous\_days\_per\_state.max()

print(f"The state with the most days having hazardous PM2.5 levels in 2023 is {most\_hazardous\_state} with {most\_hazardous\_days} days.")

34/32:

import pandas as pd

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

# Get rows for year 2023

# Convert "Timestamp" column to datetime format

df["Timestamp"] = pd.to\_datetime(df["Timestamp"], format="%yyyy-%mm-%dd")

df\_2023 = df[df["Timestamp"].dt.year == 2023]

# Drop rows w PM2.5 missing

df\_2023 = df\_2023.dropna(subset=["PM2.5"])

# hazardous PM2.5 levels

hazardous\_data = df\_2023[df\_2023["PM2.5"] > 300]

hazardous\_days\_per\_state = hazardous\_data.groupby("State")["Timestamp"].count()

# Find the state with the highest number of hazardous days

most\_hazardous\_state = hazardous\_days\_per\_state.idxmax()

most\_hazardous\_days = hazardous\_days\_per\_state.max()

print(f"The state with the most days having hazardous PM2.5 levels in 2023 is {most\_hazardous\_state} with {most\_hazardous\_days} days.")

34/33:

import pandas as pd

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

# Get rows for year 2023

# Convert "Timestamp" column to datetime format

df["Timestamp"] = pd.to\_datetime(df["Timestamp"], format="%Y-%m-%d")

df\_2023 = df[df["Timestamp"].dt.year == 2023]

# Drop rows w PM2.5 missing

df\_2023 = df\_2023.dropna(subset=["PM2.5"])

# hazardous PM2.5 levels

hazardous\_data = df\_2023[df\_2023["PM2.5"] > 300]

hazardous\_days\_per\_state = hazardous\_data.groupby("State")["Timestamp"].count()

# Find the state with the highest number of hazardous days

most\_hazardous\_state = hazardous\_days\_per\_state.idxmax()

most\_hazardous\_days = hazardous\_days\_per\_state.max()

print(f"The state with the most days having hazardous PM2.5 levels in 2023 is {most\_hazardous\_state} with {most\_hazardous\_days} days.")

34/34:

import pandas as pd

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

# Get rows for year 2023

# Convert "Timestamp" column to datetime format

df["Timestamp"] = pd.to\_datetime(df["Timestamp"], format="%Y-%m-%d")

df\_2023 = df[df["Timestamp"].dt.year == 2023]

# Drop rows w PM2.5 missing

df\_2023 = df\_2023.dropna(subset=["PM2.5"])

# hazardous PM2.5 levels

hazardous\_data = df\_2023[df\_2023["PM2.5"] > 300]

hazardous\_days\_per\_state = hazardous\_data.groupby("state")["Timestamp"].count()

# Find the state with the highest number of hazardous days

most\_hazardous\_state = hazardous\_days\_per\_state.idxmax()

most\_hazardous\_days = hazardous\_days\_per\_state.max()

print(f"The state with the most days having hazardous PM2.5 levels in 2023 is {most\_hazardous\_state} with {most\_hazardous\_days} days.")

34/35:

import pandas as pd

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

# Get rows for year 2023

# Convert Timestamp col to datetime format

df["Timestamp"] = pd.to\_datetime(df["Timestamp"], format="%Y-%m-%d")

df\_2023 = df[df["Timestamp"].dt.year == 2023]

# Drop rows w PM2.5 missing

df\_2023 = df\_2023.dropna(subset=["PM2.5"])

# hazardous PM2.5 levels

hazardous\_data = df\_2023[df\_2023["PM2.5"] > 300]

hazardous\_days\_per\_state = hazardous\_data.groupby("state")["Timestamp"].count()

# Find the state with the highest number of hazardous days

most\_hazardous\_state = hazardous\_days\_per\_state.idxmax()

most\_hazardous\_days = hazardous\_days\_per\_state.max()

print(f"The state with the most days having hazardous PM2.5 levels in 2023 is {most\_hazardous\_state} with {most\_hazardous\_days} days.")

34/36:

import pandas as pd

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

# Get rows for year 2023

# Convert Timestamp col to datetime format

df["Timestamp"] = pd.to\_datetime(df["Timestamp"], format="%Y-%m-%d")

df\_2023 = df[df["Timestamp"].dt.year == 2023]

# Drop rows w PM2.5 missing

df\_2023 = df\_2023.dropna(subset=["PM2.5"])

# Calc variation (std dev) of PM2.5 per state

var\_per\_state = df\_2023.groupby("State")["PM2.5"].std()

# Find the state with the highest variability

most\_var\_state = var\_per\_state.idxmax()

highest\_var = var\_per\_state.max()

print(f"The state with the highest variability in PM2.5 levels in 2023 is {most\_var\_state} with a standard deviation of {highest\_var:.3f}.")

34/37:

import pandas as pd

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

# Get rows for year 2023

# Convert Timestamp col to datetime format

df["Timestamp"] = pd.to\_datetime(df["Timestamp"], format="%Y-%m-%d")

df\_2023 = df[df["Timestamp"].dt.year == 2023]

# Drop rows w PM2.5 missing

df\_2023 = df\_2023.dropna(subset=["PM2.5"])

# Calc variation (std dev) of PM2.5 per state

var\_per\_state = df\_2023.groupby("state")["PM2.5"].std()

# Find the state with the highest variability

most\_var\_state = var\_per\_state.idxmax()

highest\_var = var\_per\_state.max()

print(f"The state with the highest variability in PM2.5 levels in 2023 is {most\_var\_state} with a standard deviation of {highest\_var:.3f}.")

34/38:

import pandas as pd

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

# Get rows for year 2020-21

# Convert Timestamp col to datetime format

df["Timestamp"] = pd.to\_datetime(df["Timestamp"], format="%Y-%m-%d")

df\_covid = df[(df["Timestamp"].dt.year >= 2020) & (df["Timestamp"].dt.year <= 2021)]

# Avg PM2.5 for each state

df\_dropped = df\_covid.dropna(subset=["PM2.5"]) # Drop rows w PM2.5 missing

avg\_pm = df\_dropped.groupby( by = "state")["PM2.5"].mean()

idxm = avg\_pm.idxmin() # Get the state w min avg

m = avg\_pm.min() # Get the value of the min avg PM2.5 Conc

print(f"{idxm} has the lowest average PM2.5 concentration (of {m:.3f}) across all stations and across years 2020-2021.")

34/39:

import pandas as pd

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

df["Timestamp"] = pd.to\_datetime(df["Timestamp"], format = "%Y-%m-%d") # Convert Timestamp to datetime format

df.loc?

df\_aug

34/40:

import pandas as pd

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

df["Timestamp"] = pd.to\_datetime(df["Timestamp"], format = "%Y-%m-%d") # Convert Timestamp to datetime format

df.loc?

34/41:

import pandas as pd

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

# Get rows for year 2023

df["Timestamp"] = pd.to\_datetime(df["Timestamp"], format="%Y-%m-%d") # Convert Timestamp col to datetime format

df\_2023 = df[df["Timestamp"].dt.year == 2023]

# Drop rows w PM2.5 missing

df\_2023 = df\_2023.dropna(subset=["PM2.5"])

# Calc variation (std dev) of PM2.5 per state

var\_per\_state = df\_2023.groupby("state")["PM2.5"].std()

# Find the state with the highest variability

most\_var\_state = var\_per\_state.idxmax()

highest\_var = var\_per\_state.max()

print(f"The state with the highest variability in PM2.5 levels in 2023 is {most\_var\_state} with a standard deviation of {highest\_var:.3f}.")

34/42:

import pandas as pd

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

# Get data for Aug 2020

df["Timestamp"] = pd.to\_datetime(df["Timestamp"], format = "%Y-%m-%d") # Convert Timestamp to datetime format

df\_aug\_2020 = df[(df["Timestamp"].dt.year == 2020) & (df["Timestamp"].dt.month == 8)]

df\_aug\_2020 = df\_aug\_2020.dropna(subset=["PM2.5"]) # Drop rows w PM2.5 missing

df\_aug\_2020["PM2.5"].idxmax

34/43:

import pandas as pd

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

# Get data for Aug 2020

df["Timestamp"] = pd.to\_datetime(df["Timestamp"], format = "%Y-%m-%d") # Convert Timestamp to datetime format

df\_aug\_2020 = df[(df["Timestamp"].dt.year == 2020) & (df["Timestamp"].dt.month == 8)]

df\_aug\_2020 = df\_aug\_2020.dropna(subset=["PM2.5"]) # Drop rows w PM2.5 missing

print(df\_aug\_2020["PM2.5"].idxmax)

34/44:

import pandas as pd

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

# Get data for Aug 2020

df["Timestamp"] = pd.to\_datetime(df["Timestamp"], format = "%Y-%m-%d") # Convert Timestamp to datetime format

df\_aug\_2020 = df[(df["Timestamp"].dt.year == 2020) & (df["Timestamp"].dt.month == 8)]

df\_aug\_2020 = df\_aug\_2020.dropna(subset=["PM2.5"]) # Drop rows w PM2.5 missing

print(df\_aug\_2020["PM2.5"].idxmax)[0]

34/45:

import pandas as pd

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

# Get data for Aug 2020

df["Timestamp"] = pd.to\_datetime(df["Timestamp"], format = "%Y-%m-%d") # Convert Timestamp to datetime format

df\_aug\_2020 = df[(df["Timestamp"].dt.year == 2020) & (df["Timestamp"].dt.month == 8)]

df\_aug\_2020 = df\_aug\_2020.dropna(subset=["PM2.5"]) # Drop rows w PM2.5 missing

print(df\_aug\_2020["PM2.5"].idxmax[0])

34/46:

import pandas as pd

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

# Get data for Aug 2020

df["Timestamp"] = pd.to\_datetime(df["Timestamp"], format = "%Y-%m-%d") # Convert Timestamp to datetime format

df\_aug\_2020 = df[(df["Timestamp"].dt.year == 2020) & (df["Timestamp"].dt.month == 8)]

df\_aug\_2020 = df\_aug\_2020.dropna(subset=["PM2.5"]) # Drop rows w PM2.5 missing

print(df\_aug\_2020["PM2.5"].idxmax())

34/47:

import pandas as pd

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

# Get data for Aug 2020

df["Timestamp"] = pd.to\_datetime(df["Timestamp"], format = "%Y-%m-%d") # Convert Timestamp to datetime format

df\_aug\_2020 = df[(df["Timestamp"].dt.year == 2020) & (df["Timestamp"].dt.month == 8)]

df\_aug\_2020 = df\_aug\_2020.dropna(subset=["PM2.5"]) # Drop rows w PM2.5 missing

idxm = df\_aug\_2020["PM2.5"].idxmax()

print(df[idxm])

34/48:

import pandas as pd

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

# Get data for Aug 2020

df["Timestamp"] = pd.to\_datetime(df["Timestamp"], format = "%Y-%m-%d") # Convert Timestamp to datetime format

df\_aug\_2020 = df[(df["Timestamp"].dt.year == 2020) & (df["Timestamp"].dt.month == 8)]

df\_aug\_2020 = df\_aug\_2020.dropna(subset=["PM2.5"]) # Drop rows w PM2.5 missing

idxm = df\_aug\_2020["PM2.5"].idxmax()

print(df.iloc(idxm))

34/49:

import pandas as pd

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

# Get data for Aug 2020

df["Timestamp"] = pd.to\_datetime(df["Timestamp"], format = "%Y-%m-%d") # Convert Timestamp to datetime format

df\_aug\_2020 = df[(df["Timestamp"].dt.year == 2020) & (df["Timestamp"].dt.month == 8)]

df\_aug\_2020 = df\_aug\_2020.dropna(subset=["PM2.5"]) # Drop rows w PM2.5 missing

idxm = df\_aug\_2020["PM2.5"].idxmax()

print(df.iloc(0, idxm))

34/50:

import pandas as pd

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

# Get data for Aug 2020

df["Timestamp"] = pd.to\_datetime(df["Timestamp"], format = "%Y-%m-%d") # Convert Timestamp to datetime format

df\_aug\_2020 = df[(df["Timestamp"].dt.year == 2020) & (df["Timestamp"].dt.month == 8)]

df\_aug\_2020 = df\_aug\_2020.dropna(subset=["PM2.5"]) # Drop rows w PM2.5 missing

idxm = df\_aug\_2020["PM2.5"].idxmax()

print(df\_aug\_2020.iloc[idxm])

34/51:

import pandas as pd

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

# Get data for Aug 2020

df["Timestamp"] = pd.to\_datetime(df["Timestamp"], format = "%Y-%m-%d") # Convert Timestamp to datetime format

df\_aug\_2020 = df[(df["Timestamp"].dt.year == 2020) & (df["Timestamp"].dt.month == 8)]

df\_aug\_2020 = df\_aug\_2020.dropna(subset=["PM2.5"]) # Drop rows w PM2.5 missing

idxm = df\_aug\_2020["PM2.5"].idxmax()

print(df.iloc[idxm])

34/52:

import pandas as pd

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

# Get data for Aug 2020

df["Timestamp"] = pd.to\_datetime(df["Timestamp"], format = "%Y-%m-%d") # Convert Timestamp to datetime format

df\_aug\_2020 = df[(df["Timestamp"].dt.year == 2020) & (df["Timestamp"].dt.month == 8)]

df\_aug\_2020 = df\_aug\_2020.dropna(subset=["PM2.5"]) # Drop rows w PM2.5 missing

idxm = df\_aug\_2020["PM2.5"].idxmax()

print(df.iloc[idxm]["state"])

34/53:

import pandas as pd

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

# Get data for Aug 2020

df["Timestamp"] = pd.to\_datetime(df["Timestamp"], format = "%Y-%m-%d") # Convert Timestamp to datetime format

df\_aug\_2020 = df[(df["Timestamp"].dt.year == 2020) & (df["Timestamp"].dt.month == 8)]

df\_aug\_2020 = df\_aug\_2020.dropna(subset=["PM2.5"]) # Drop rows w PM2.5 missing

idxm = df\_aug\_2020["PM2.5"].idxmax()

print(df.iloc[idxm](["state"], ["PM2.5"]))

34/54:

import pandas as pd

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

# Get data for Aug 2020

df["Timestamp"] = pd.to\_datetime(df["Timestamp"], format = "%Y-%m-%d") # Convert Timestamp to datetime format

df\_aug\_2020 = df[(df["Timestamp"].dt.year == 2020) & (df["Timestamp"].dt.month == 8)]

df\_aug\_2020 = df\_aug\_2020.dropna(subset=["PM2.5"]) # Drop rows w PM2.5 missing

idxm = df\_aug\_2020["PM2.5"].idxmax()

print(df.iloc[idxm].loc(["state"], ["PM2.5"]))

34/55:

import pandas as pd

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

# Get data for Aug 2020

df["Timestamp"] = pd.to\_datetime(df["Timestamp"], format = "%Y-%m-%d") # Convert Timestamp to datetime format

df\_aug\_2020 = df[(df["Timestamp"].dt.year == 2020) & (df["Timestamp"].dt.month == 8)]

df\_aug\_2020 = df\_aug\_2020.dropna(subset=["PM2.5"]) # Drop rows w PM2.5 missing

idxm = df\_aug\_2020["PM2.5"].idxmax()

print(df.iloc[idxm].loc[["state"], ["PM2.5"]])

34/56:

import pandas as pd

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

# Get data for Aug 2020

df["Timestamp"] = pd.to\_datetime(df["Timestamp"], format = "%Y-%m-%d") # Convert Timestamp to datetime format

df\_aug\_2020 = df[(df["Timestamp"].dt.year == 2020) & (df["Timestamp"].dt.month == 8)]

df\_aug\_2020 = df\_aug\_2020.dropna(subset=["PM2.5"]) # Drop rows w PM2.5 missing

idxm = df\_aug\_2020["PM2.5"].idxmax()

print(df.iloc[idxm].loc["state", "PM2.5"])

34/57:

import pandas as pd

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

# Get data for Aug 2020

df["Timestamp"] = pd.to\_datetime(df["Timestamp"], format = "%Y-%m-%d") # Convert Timestamp to datetime format

df\_aug\_2020 = df[(df["Timestamp"].dt.year == 2020) & (df["Timestamp"].dt.month == 8)]

df\_aug\_2020 = df\_aug\_2020.dropna(subset=["PM2.5"]) # Drop rows w PM2.5 missing

idxm = df\_aug\_2020["PM2.5"].idxmax()

print(df.iloc[idxm].loc[["state", "PM2.5"]])

34/58:

import pandas as pd

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

# Get data for Aug 2020

df["Timestamp"] = pd.to\_datetime(df["Timestamp"], format = "%Y-%m-%d") # Convert Timestamp to datetime format

df\_aug\_2020 = df[(df["Timestamp"].dt.year == 2020) & (df["Timestamp"].dt.month == 8)]

df\_aug\_2020 = df\_aug\_2020.dropna(subset=["PM2.5"]) # Drop rows w PM2.5 missing

idxm = df\_aug\_2020["PM2.5"].idxmax()

a, b = df.iloc[idxm].loc[["state", "PM2.5"]]

print(a, b)

34/59:

import pandas as pd

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

# Get data for Aug 2020

df["Timestamp"] = pd.to\_datetime(df["Timestamp"], format = "%Y-%m-%d") # Convert Timestamp to datetime format

df\_aug\_2020 = df[(df["Timestamp"].dt.year == 2020) & (df["Timestamp"].dt.month == 8)]

df\_aug\_2020 = df\_aug\_2020.dropna(subset=["PM2.5"]) # Drop rows w PM2.5 missing

idxm = df\_aug\_2020["PM2.5"].idxmax()

# max\_state, max\_pm = df.iloc[idxm].loc[["state", "PM2.5"]]

df.loc?

# max\_state, max\_pm = df.loc[["state", "PM2.5"]]

# print(

34/60:

import pandas as pd

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

# Get data for Aug 2020

df["Timestamp"] = pd.to\_datetime(df["Timestamp"], format = "%Y-%m-%d") # Convert Timestamp to datetime format

df\_aug\_2020 = df[(df["Timestamp"].dt.year == 2020) & (df["Timestamp"].dt.month == 8)]

df\_aug\_2020 = df\_aug\_2020.dropna(subset=["PM2.5"]) # Drop rows w PM2.5 missing

idxm = df\_aug\_2020["PM2.5"].idxmax()

# max\_state, max\_pm = df.iloc[idxm].loc[["state", "PM2.5"]]

max\_state, max\_pm = df.loc[idxm, ["state", "PM2.5"]]

print(max\_state, max\_pm)

34/61:

import pandas as pd

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

df["Timestamp"] = pd.to\_datetime(df["Timestamp"], format = "%Y-%m-%d") # Convert Timestamp to datetime format

# Get data for Aug 2020

df\_aug\_2020 = df[(df["Timestamp"].dt.year == 2020) & (df["Timestamp"].dt.month == 8)]

df\_aug\_2020 = df\_aug\_2020.dropna(subset=["PM2.5"]) # Drop rows w PM2.5 missing

# Find max of pm2.5

idxm = df\_aug\_2020["PM2.5"].idxmax()

max\_station, max\_state, max\_pm = df.loc[idxm, ["station", "state", "PM2.5"]]

print(f"{max\_station} in the state of {max\_state} recorded the highest PM2.5 value (of {max\_pm}) in August 2020.")

34/62:

import pandas as pd

import numpy as np

# Load the dataset

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

df["Timestamp"] = pd.to\_datetime(df["Timestamp"], format = "%Y-%m-%d") # Convert Timestamp to datetime format

# Get rows for year 2018

df\_2018 = df[df["Timestamp"].dt.year == 2018]

# Mask for the given stations

df\_stations = df\_2018[df\_2018["station"] == "Lal Bahadur Shastri Nagar", "Kalaburagi - KSPCB"]

df\_stations = df\_stations.dropna(subset = ["PM2.5"]) # Drop rows w PM2.5 missing

# Define seasons

get\_season = [

(df\_stations["Timestamp"].dt.month.isin([12, 1, 2])), # winter

(df\_stations["Timestamp"].dt.month.isin([3, 4, 5])), # Summer

(df\_stations["Timestamp"].dt.month.isin([6, 7, 8, 9])) # monsoon

]

# Define the corresponding seasons for each condition

seasons = ["Winter", "Summer", "Monsoon"]

# Assign a season to each row in df\_stations w np.select()

df\_stations["Season"] = np.select(conditions, seasons, default="Other")

# Calculate average PM2.5 for each season

avg\_pm25\_per\_season = df\_stations.groupby("season")["PM2.5"].mean()

# Get season w the highest pollution

most\_polluted\_season = avg\_pm25\_per\_season.idxmax()

highest\_avg\_pm25 = avg\_pm25\_per\_season.max()

print(f"The average PM2.5 concentration for each season for stations Lal Bahadur Shastri Nagar, Kalaburagi - KSPCBavg\_pm25\_per\_season is {avg\_pm25\_per\_season}")

print(f"The most polluted season in 2018 was {most\_polluted\_season} with an average PM2.5 concentration of {highest\_avg\_pm25:.2f} µg/m³.")

34/63:

import pandas as pd

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

df["Timestamp"] = pd.to\_datetime(df["Timestamp"], format = "%Y-%m-%d") # Convert Timestamp to datetime format

# Get data for Aug 2020

df\_aug\_2020 = df[(df["Timestamp"].dt.year == 2020) & (df["Timestamp"].dt.month == 8)]

df\_aug\_2020 = df\_aug\_2020.dropna(subset =["PM2.5"]) # Drop rows w PM2.5 missing

# Find max of pm2.5

idxm = df\_aug\_2020["PM2.5"].idxmax()

max\_station, max\_state, max\_pm = df.loc[idxm, ["station", "state", "PM2.5"]]

print(f"{max\_station} in the state of {max\_state} recorded the highest PM2.5 value (of {max\_pm}) in August 2020.")

34/64:

import pandas as pd

import numpy as np

# Load the dataset

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

df["Timestamp"] = pd.to\_datetime(df["Timestamp"], format = "%Y-%m-%d") # Convert Timestamp to datetime format

# Get rows for year 2018

df\_2018 = df[df["Timestamp"].dt.year == 2018]

# Mask for the given stations

df\_stations = df\_2018[df\_2018["station"] == "Lal Bahadur Shastri Nagar, Kalaburagi - KSPCB"]

df\_stations = df\_stations.dropna(subset = ["PM2.5"]) # Drop rows w PM2.5 missing

# Define seasons

get\_season = [

(df\_stations["Timestamp"].dt.month.isin([12, 1, 2])), # winter

(df\_stations["Timestamp"].dt.month.isin([3, 4, 5])), # Summer

(df\_stations["Timestamp"].dt.month.isin([6, 7, 8, 9])) # monsoon

]

# Define the corresponding seasons for each condition

seasons = ["Winter", "Summer", "Monsoon"]

# Assign a season to each row in df\_stations w np.select()

df\_stations["Season"] = np.select(conditions, seasons, default="Other")

# Calculate average PM2.5 for each season

avg\_pm25\_per\_season = df\_stations.groupby("season")["PM2.5"].mean()

# Get season w the highest pollution

most\_polluted\_season = avg\_pm25\_per\_season.idxmax()

highest\_avg\_pm25 = avg\_pm25\_per\_season.max()

print(f"The average PM2.5 concentration for each season for stations Lal Bahadur Shastri Nagar, Kalaburagi - KSPCBavg\_pm25\_per\_season is {avg\_pm25\_per\_season}")

print(f"The most polluted season in 2018 was {most\_polluted\_season} with an average PM2.5 concentration of {highest\_avg\_pm25:.2f}.")

34/65:

import pandas as pd

import numpy as np

# Load the dataset

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

df["Timestamp"] = pd.to\_datetime(df["Timestamp"], format = "%Y-%m-%d") # Convert Timestamp to datetime format

# Get rows for year 2018

df\_2018 = df[df["Timestamp"].dt.year == 2018]

# Mask for the given stations

df\_stations = df\_2018[df\_2018["station"] == "Lal Bahadur Shastri Nagar, Kalaburagi - KSPCB"]

df\_stations = df\_stations.dropna(subset = ["PM2.5"]) # Drop rows w PM2.5 missing

# Define seasons

get\_season = [

(df\_stations["Timestamp"].dt.month.isin([12, 1, 2])), # winter

(df\_stations["Timestamp"].dt.month.isin([3, 4, 5])), # Summer

(df\_stations["Timestamp"].dt.month.isin([6, 7, 8, 9])) # monsoon

]

# Corresponding seasons for each condition

seasons = ["Winter", "Summer", "Monsoon"]

# Assign a season to each row in df\_stations w np.select()

df\_stations["Season"] = np.select(get\_season, seasons, default="Other")

# Calculate average PM2.5 for each season

avg\_pm25\_per\_season = df\_stations.groupby("season")["PM2.5"].mean()

# Get season w the highest pollution

most\_polluted\_season = avg\_pm25\_per\_season.idxmax()

highest\_avg\_pm25 = avg\_pm25\_per\_season.max()

print(f"The average PM2.5 concentration for each season for stations Lal Bahadur Shastri Nagar, Kalaburagi - KSPCBavg\_pm25\_per\_season is {avg\_pm25\_per\_season}")

print(f"The most polluted season in 2018 was {most\_polluted\_season} with an average PM2.5 concentration of {highest\_avg\_pm25:.2f}.")

34/66:

import pandas as pd

import numpy as np

# Load the dataset

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

df["Timestamp"] = pd.to\_datetime(df["Timestamp"], format = "%Y-%m-%d") # Convert Timestamp to datetime format

# Get rows for year 2018

df\_2018 = df[df["Timestamp"].dt.year == 2018]

# Mask for the given stations

df\_stations = df\_2018[df\_2018["station"] == "Lal Bahadur Shastri Nagar, Kalaburagi - KSPCB"]

df\_stations = df\_stations.dropna(subset = ["PM2.5"]) # Drop rows w PM2.5 missing

# Define seasons

get\_season = [

(df\_stations["Timestamp"].dt.month.isin([12, 1, 2])), # winter

(df\_stations["Timestamp"].dt.month.isin([3, 4, 5])), # Summer

(df\_stations["Timestamp"].dt.month.isin([6, 7, 8, 9])) # monsoon

]

# Corresponding seasons for each condition

seasons = ["Winter", "Summer", "Monsoon"]

# Assign a season to each row in df\_stations w np.select()

df\_stations["Season"] = np.select(get\_season, seasons, default="Other")

# Calculate average PM2.5 for each season

avg\_pm25\_per\_season = df\_stations.groupby("Season")["PM2.5"].mean()

# Get season w the highest pollution

most\_polluted\_season = avg\_pm25\_per\_season.idxmax()

highest\_avg\_pm25 = avg\_pm25\_per\_season.max()

print(f"The average PM2.5 concentration for each season for stations Lal Bahadur Shastri Nagar, Kalaburagi - KSPCBavg\_pm25\_per\_season is {avg\_pm25\_per\_season}")

print(f"The most polluted season in 2018 was {most\_polluted\_season} with an average PM2.5 concentration of {highest\_avg\_pm25:.2f}.")

34/67:

import pandas as pd

import numpy as np

# Load the dataset

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

df["Timestamp"] = pd.to\_datetime(df["Timestamp"], format = "%Y-%m-%d") # Convert Timestamp to datetime format

# Get rows for year 2018

df\_2018 = df[df["Timestamp"].dt.year == 2018]

# Mask for the given stations

df\_stations = df\_2018[df\_2018["station"] == "Lal Bahadur Shastri Nagar, Kalaburagi - KSPCB"]

df\_stations = df\_stations.dropna(subset = ["PM2.5"]) # Drop rows w PM2.5 missing

# Define seasons

get\_season = [

(df\_stations["Timestamp"].dt.month.isin([12, 1, 2])), # winter

(df\_stations["Timestamp"].dt.month.isin([3, 4, 5])), # Summer

(df\_stations["Timestamp"].dt.month.isin([6, 7, 8, 9])) # monsoon

]

# Corresponding seasons for each condition

seasons = ["Winter", "Summer", "Monsoon"]

# Assign a season to each row in df\_stations w np.select()

df\_stations["Season"] = np.select(get\_season, seasons, default="Other")

# Calculate average PM2.5 for each season

avg\_pm25\_per\_season = df\_stations.groupby("Season")["PM2.5"].mean()

# Get season w the highest pollution

most\_polluted\_season = avg\_pm25\_per\_season.idxmax()

highest\_avg\_pm25 = avg\_pm25\_per\_season.max()

print(f"The average PM2.5 concentration for each season for stations Lal Bahadur Shastri Nagar, Kalaburagi - KSPCBavg\_pm25\_per\_season is:\n {avg\_pm25\_per\_season}")

print(f"\nThe most polluted season in 2018 was {most\_polluted\_season} with an average PM2.5 concentration of {highest\_avg\_pm25:.2f}.")

34/68:

import pandas as pd

import numpy as np

# Load the dataset

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

df["Timestamp"] = pd.to\_datetime(df["Timestamp"], format = "%Y-%m-%d") # Convert Timestamp to datetime format

# Get rows for year 2018

df\_2018 = df[df["Timestamp"].dt.year == 2018]

# Mask for the given stations

df\_stations = df\_2018[df\_2018["station"] == "Lal Bahadur Shastri Nagar, Kalaburagi - KSPCB"]

df\_stations = df\_stations.dropna(subset = ["PM2.5"]) # Drop rows w PM2.5 missing

# Define seasons

get\_season = [

(df\_stations["Timestamp"].dt.month.isin([12, 1, 2])), # winter

(df\_stations["Timestamp"].dt.month.isin([3, 4, 5])), # Summer

(df\_stations["Timestamp"].dt.month.isin([6, 7, 8, 9])) # monsoon

]

# Corresponding seasons for each condition

seasons = ["Winter", "Summer", "Monsoon"]

# Assign a season to each row in df\_stations w np.select()

df\_stations["Season"] = np.select(get\_season, seasons, default="Other")

# Calculate average PM2.5 for each season

avg\_pm25\_per\_season = df\_stations.groupby("Season")["PM2.5"].mean()

# Get season w the highest pollution

most\_polluted\_season = avg\_pm25\_per\_season.idxmax()

highest\_avg\_pm25 = avg\_pm25\_per\_season.max()

print(f"The average PM2.5 concentration for each season for stations Lal Bahadur Shastri Nagar, Kalaburagi - KSPCBavg\_pm25\_per\_season is:\n {avg\_pm25\_per\_season}")

print(f"\nThe most polluted season in 2018 was {most\_polluted\_season} with an average PM2.5 concentration of {highest\_avg\_pm25:.3f}")

34/69:

import pandas as pd

import numpy as np

# Load the dataset

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

df["Timestamp"] = pd.to\_datetime(df["Timestamp"], format = "%Y-%m-%d") # Convert Timestamp to datetime format

# Get rows for year 2018

df\_2018 = df[df["Timestamp"].dt.year == 2018]

# Mask for the given stations

df\_stations = df\_2018[df\_2018["station"] == "Lal Bahadur Shastri Nagar, Kalaburagi - KSPCB"]

df\_stations = df\_stations.dropna(subset = ["PM2.5"]) # Drop rows w PM2.5 missing

# Define seasons

get\_season = [

(df\_stations["Timestamp"].dt.month.isin([12, 1, 2])), # winter

(df\_stations["Timestamp"].dt.month.isin([3, 4, 5])), # Summer

(df\_stations["Timestamp"].dt.month.isin([6, 7, 8, 9])) # monsoon

]

# Corresponding seasons for each condition

seasons = ["Winter", "Summer", "Monsoon"]

# Assign a season to each row in df\_stations w np.select()

df\_stations["Season"] = np.select(get\_season, seasons, default="Other")

# Calculate average PM2.5 for each season

avg\_pm25\_per\_season = df\_stations.groupby("Season")["PM2.5"].mean()

# Get season w the highest pollution

most\_polluted\_season = avg\_pm25\_per\_season.idxmax()

highest\_avg\_pm25 = avg\_pm25\_per\_season.max()

print(f"The average PM2.5 concentration for each season for stations Lal Bahadur Shastri Nagar, Kalaburagi - KSPCBavg\_pm25\_per\_season is:\n{avg\_pm25\_per\_season}")

print(f"\nThe most polluted season in 2018 was {most\_polluted\_season} with an average PM2.5 concentration of {highest\_avg\_pm25:.3f}")

34/70:

import pandas as pd

import numpy as np

# Load the dataset

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

df["Timestamp"] = pd.to\_datetime(df["Timestamp"], format = "%Y-%m-%d") # Convert Timestamp to datetime format

# Get rows for year 2018

df\_2018 = df[df["Timestamp"].dt.year == 2018]

# Mask for the given stations

df\_stations = df\_2018[df\_2018["station"] == "Lal Bahadur Shastri Nagar, Kalaburagi - KSPCB"]

df\_stations = df\_stations.dropna(subset = ["PM2.5"]) # Drop rows w PM2.5 missing

# Define seasons

get\_season = [

(df\_stations["Timestamp"].dt.month.isin([12, 1, 2])), # winter

(df\_stations["Timestamp"].dt.month.isin([3, 4, 5])), # Summer

(df\_stations["Timestamp"].dt.month.isin([6, 7, 8, 9])) # monsoon

]

# Corresponding seasons for each condition

seasons = ["Winter", "Summer", "Monsoon"]

# Assign a season to each row in df\_stations w np.select()

df\_stations["Season"] = np.select(get\_season, seasons, default="Other")

# Calculate average PM2.5 for each season

avg\_pm25\_per\_season = df\_stations.groupby("Season")["PM2.5"].agg("Average PM2.5" = mean)

# Get season w the highest pollution

most\_polluted\_season = avg\_pm25\_per\_season.idxmax()

highest\_avg\_pm25 = avg\_pm25\_per\_season.max()

print(f"The average PM2.5 concentration for each season for stations Lal Bahadur Shastri Nagar, Kalaburagi - KSPCBavg\_pm25\_per\_season is:\n{avg\_pm25\_per\_season}")

print(f"\nThe most polluted season in 2018 was {most\_polluted\_season} with an average PM2.5 concentration of {highest\_avg\_pm25:.3f}")

34/71:

import pandas as pd

import numpy as np

# Load the dataset

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

df["Timestamp"] = pd.to\_datetime(df["Timestamp"], format = "%Y-%m-%d") # Convert Timestamp to datetime format

# Get rows for year 2018

df\_2018 = df[df["Timestamp"].dt.year == 2018]

# Mask for the given stations

df\_stations = df\_2018[df\_2018["station"] == "Lal Bahadur Shastri Nagar, Kalaburagi - KSPCB"]

df\_stations = df\_stations.dropna(subset = ["PM2.5"]) # Drop rows w PM2.5 missing

# Define seasons

get\_season = [

(df\_stations["Timestamp"].dt.month.isin([12, 1, 2])), # winter

(df\_stations["Timestamp"].dt.month.isin([3, 4, 5])), # Summer

(df\_stations["Timestamp"].dt.month.isin([6, 7, 8, 9])) # monsoon

]

# Corresponding seasons for each condition

seasons = ["Winter", "Summer", "Monsoon"]

# Assign a season to each row in df\_stations w np.select()

df\_stations["Season"] = np.select(get\_season, seasons, default="Other")

# Calculate average PM2.5 for each season

avg\_pm25\_per\_season = df\_stations.groupby("Season")["PM2.5"].agg("Average PM2.5" = "mean")

# Get season w the highest pollution

most\_polluted\_season = avg\_pm25\_per\_season.idxmax()

highest\_avg\_pm25 = avg\_pm25\_per\_season.max()

print(f"The average PM2.5 concentration for each season for stations Lal Bahadur Shastri Nagar, Kalaburagi - KSPCBavg\_pm25\_per\_season is:\n{avg\_pm25\_per\_season}")

print(f"\nThe most polluted season in 2018 was {most\_polluted\_season} with an average PM2.5 concentration of {highest\_avg\_pm25:.3f}")

34/72:

import pandas as pd

import numpy as np

# Load the dataset

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

df["Timestamp"] = pd.to\_datetime(df["Timestamp"], format = "%Y-%m-%d") # Convert Timestamp to datetime format

# Get rows for year 2018

df\_2018 = df[df["Timestamp"].dt.year == 2018]

# Mask for the given stations

df\_stations = df\_2018[df\_2018["station"] == "Lal Bahadur Shastri Nagar, Kalaburagi - KSPCB"]

df\_stations = df\_stations.dropna(subset = ["PM2.5"]) # Drop rows w PM2.5 missing

# Define seasons

get\_season = [

(df\_stations["Timestamp"].dt.month.isin([12, 1, 2])), # winter

(df\_stations["Timestamp"].dt.month.isin([3, 4, 5])), # Summer

(df\_stations["Timestamp"].dt.month.isin([6, 7, 8, 9])) # monsoon

]

# Corresponding seasons for each condition

seasons = ["Winter", "Summer", "Monsoon"]

# Assign a season to each row in df\_stations w np.select()

df\_stations["Season"] = np.select(get\_season, seasons, default="Other")

# Calculate average PM2.5 for each season

avg\_pm25\_per\_season = df\_stations.groupby("Season")["PM2.5"]..agg("Average PM2.5" = "mean")

# Get season w the highest pollution

most\_polluted\_season = avg\_pm25\_per\_season.idxmax()

highest\_avg\_pm25 = avg\_pm25\_per\_season.max()

print(f"The average PM2.5 concentration for each season for stations Lal Bahadur Shastri Nagar, Kalaburagi - KSPCBavg\_pm25\_per\_season is:\n{avg\_pm25\_per\_season}")

print(f"\nThe most polluted season in 2018 was {most\_polluted\_season} with an average PM2.5 concentration of {highest\_avg\_pm25:.3f}")

34/73:

import pandas as pd

import numpy as np

# Load the dataset

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

df["Timestamp"] = pd.to\_datetime(df["Timestamp"], format = "%Y-%m-%d") # Convert Timestamp to datetime format

# Get rows for year 2018

df\_2018 = df[df["Timestamp"].dt.year == 2018]

# Mask for the given stations

df\_stations = df\_2018[df\_2018["station"] == "Lal Bahadur Shastri Nagar, Kalaburagi - KSPCB"]

df\_stations = df\_stations.dropna(subset = ["PM2.5"]) # Drop rows w PM2.5 missing

# Define seasons

get\_season = [

(df\_stations["Timestamp"].dt.month.isin([12, 1, 2])), # winter

(df\_stations["Timestamp"].dt.month.isin([3, 4, 5])), # Summer

(df\_stations["Timestamp"].dt.month.isin([6, 7, 8, 9])) # monsoon

]

# Corresponding seasons for each condition

seasons = ["Winter", "Summer", "Monsoon"]

# Assign a season to each row in df\_stations w np.select()

df\_stations["Season"] = np.select(get\_season, seasons, default="Other")

# Calculate average PM2.5 for each season

avg\_pm25\_per\_season = df\_stations.groupby("Season")["PM2.5"].agg("Average PM2.5" = "mean")

# Get season w the highest pollution

most\_polluted\_season = avg\_pm25\_per\_season.idxmax()

highest\_avg\_pm25 = avg\_pm25\_per\_season.max()

print(f"The average PM2.5 concentration for each season for stations Lal Bahadur Shastri Nagar, Kalaburagi - KSPCBavg\_pm25\_per\_season is:\n{avg\_pm25\_per\_season}")

print(f"\nThe most polluted season in 2018 was {most\_polluted\_season} with an average PM2.5 concentration of {highest\_avg\_pm25:.3f}")

34/74:

import pandas as pd

import numpy as np

# Load the dataset

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

df["Timestamp"] = pd.to\_datetime(df["Timestamp"], format = "%Y-%m-%d") # Convert Timestamp to datetime format

# Get rows for year 2018

df\_2018 = df[df["Timestamp"].dt.year == 2018]

# Mask for the given stations

df\_stations = df\_2018[df\_2018["station"] == "Lal Bahadur Shastri Nagar, Kalaburagi - KSPCB"]

df\_stations = df\_stations.dropna(subset = ["PM2.5"]) # Drop rows w PM2.5 missing

# Define seasons

get\_season = [

(df\_stations["Timestamp"].dt.month.isin([12, 1, 2])), # winter

(df\_stations["Timestamp"].dt.month.isin([3, 4, 5])), # Summer

(df\_stations["Timestamp"].dt.month.isin([6, 7, 8, 9])) # monsoon

]

# Corresponding seasons for each condition

seasons = ["Winter", "Summer", "Monsoon"]

# Assign a season to each row in df\_stations w np.select()

df\_stations["Season"] = np.select(get\_season, seasons, default="Other")

# Calculate average PM2.5 for each season

avg\_pm25\_per\_season = df\_stations.groupby("Season")["PM2.5"].agg(Average\_PM2.5 = "mean")

# Get season w the highest pollution

most\_polluted\_season = avg\_pm25\_per\_season.idxmax()

highest\_avg\_pm25 = avg\_pm25\_per\_season.max()

print(f"The average PM2.5 concentration for each season for stations Lal Bahadur Shastri Nagar, Kalaburagi - KSPCBavg\_pm25\_per\_season is:\n{avg\_pm25\_per\_season}")

print(f"\nThe most polluted season in 2018 was {most\_polluted\_season} with an average PM2.5 concentration of {highest\_avg\_pm25:.3f}")

34/75:

import pandas as pd

import numpy as np

# Load the dataset

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

df["Timestamp"] = pd.to\_datetime(df["Timestamp"], format = "%Y-%m-%d") # Convert Timestamp to datetime format

# Get rows for year 2018

df\_2018 = df[df["Timestamp"].dt.year == 2018]

# Mask for the given stations

df\_stations = df\_2018[df\_2018["station"] == "Lal Bahadur Shastri Nagar, Kalaburagi - KSPCB"]

df\_stations = df\_stations.dropna(subset = ["PM2.5"]) # Drop rows w PM2.5 missing

# Define seasons

get\_season = [

(df\_stations["Timestamp"].dt.month.isin([12, 1, 2])), # winter

(df\_stations["Timestamp"].dt.month.isin([3, 4, 5])), # Summer

(df\_stations["Timestamp"].dt.month.isin([6, 7, 8, 9])) # monsoon

]

# Corresponding seasons for each condition

seasons = ["Winter", "Summer", "Monsoon"]

# Assign a season to each row in df\_stations w np.select()

df\_stations["Season"] = np.select(get\_season, seasons, default="Other")

# Calculate average PM2.5 for each season

avg\_pm25\_per\_season = df\_stations.groupby("Season")["PM2.5"].agg(Average\_PM2.5 = mean)

# Get season w the highest pollution

most\_polluted\_season = avg\_pm25\_per\_season.idxmax()

highest\_avg\_pm25 = avg\_pm25\_per\_season.max()

print(f"The average PM2.5 concentration for each season for stations Lal Bahadur Shastri Nagar, Kalaburagi - KSPCBavg\_pm25\_per\_season is:\n{avg\_pm25\_per\_season}")

print(f"\nThe most polluted season in 2018 was {most\_polluted\_season} with an average PM2.5 concentration of {highest\_avg\_pm25:.3f}")

34/76:

import pandas as pd

import numpy as np

# Load the dataset

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

df["Timestamp"] = pd.to\_datetime(df["Timestamp"], format = "%Y-%m-%d") # Convert Timestamp to datetime format

# Get rows for year 2018

df\_2018 = df[df["Timestamp"].dt.year == 2018]

# Mask for the given stations

df\_stations = df\_2018[df\_2018["station"] == "Lal Bahadur Shastri Nagar, Kalaburagi - KSPCB"]

df\_stations = df\_stations.dropna(subset = ["PM2.5"]) # Drop rows w PM2.5 missing

# Define seasons

get\_season = [

(df\_stations["Timestamp"].dt.month.isin([12, 1, 2])), # winter

(df\_stations["Timestamp"].dt.month.isin([3, 4, 5])), # Summer

(df\_stations["Timestamp"].dt.month.isin([6, 7, 8, 9])) # monsoon

]

# Corresponding seasons for each condition

seasons = ["Winter", "Summer", "Monsoon"]

# Assign a season to each row in df\_stations w np.select()

df\_stations["Season"] = np.select(get\_season, seasons, default="Other")

# Calculate average PM2.5 for each season

avg\_pm25\_per\_season = df\_stations.groupby("Season")["PM2.5"].agg(Average\_PM = mean)

# Get season w the highest pollution

most\_polluted\_season = avg\_pm25\_per\_season.idxmax()

highest\_avg\_pm25 = avg\_pm25\_per\_season.max()

print(f"The average PM2.5 concentration for each season for stations Lal Bahadur Shastri Nagar, Kalaburagi - KSPCBavg\_pm25\_per\_season is:\n{avg\_pm25\_per\_season}")

print(f"\nThe most polluted season in 2018 was {most\_polluted\_season} with an average PM2.5 concentration of {highest\_avg\_pm25:.3f}")

34/77:

import pandas as pd

import numpy as np

# Load the dataset

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

df["Timestamp"] = pd.to\_datetime(df["Timestamp"], format = "%Y-%m-%d") # Convert Timestamp to datetime format

# Get rows for year 2018

df\_2018 = df[df["Timestamp"].dt.year == 2018]

# Mask for the given stations

df\_stations = df\_2018[df\_2018["station"] == "Lal Bahadur Shastri Nagar, Kalaburagi - KSPCB"]

df\_stations = df\_stations.dropna(subset = ["PM2.5"]) # Drop rows w PM2.5 missing

# Define seasons

get\_season = [

(df\_stations["Timestamp"].dt.month.isin([12, 1, 2])), # winter

(df\_stations["Timestamp"].dt.month.isin([3, 4, 5])), # Summer

(df\_stations["Timestamp"].dt.month.isin([6, 7, 8, 9])) # monsoon

]

# Corresponding seasons for each condition

seasons = ["Winter", "Summer", "Monsoon"]

# Assign a season to each row in df\_stations w np.select()

df\_stations["Season"] = np.select(get\_season, seasons, default="Other")

# Calculate average PM2.5 for each season

avg\_pm25\_per\_season = df\_stations.groupby("Season")["PM2.5"].agg(Average\_PM = "mean")

# Get season w the highest pollution

most\_polluted\_season = avg\_pm25\_per\_season.idxmax()

highest\_avg\_pm25 = avg\_pm25\_per\_season.max()

print(f"The average PM2.5 concentration for each season for stations Lal Bahadur Shastri Nagar, Kalaburagi - KSPCBavg\_pm25\_per\_season is:\n{avg\_pm25\_per\_season}")

print(f"\nThe most polluted season in 2018 was {most\_polluted\_season} with an average PM2.5 concentration of {highest\_avg\_pm25:.3f}")

34/78:

import pandas as pd

import numpy as np

# Load the dataset

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

df["Timestamp"] = pd.to\_datetime(df["Timestamp"], format = "%Y-%m-%d") # Convert Timestamp to datetime format

# Get rows for year 2018

df\_2018 = df[df["Timestamp"].dt.year == 2018]

# Mask for the given stations

df\_stations = df\_2018[df\_2018["station"] == "Lal Bahadur Shastri Nagar, Kalaburagi - KSPCB"]

df\_stations = df\_stations.dropna(subset = ["PM2.5"]) # Drop rows w PM2.5 missing

# Define seasons

get\_season = [

(df\_stations["Timestamp"].dt.month.isin([12, 1, 2])), # winter

(df\_stations["Timestamp"].dt.month.isin([3, 4, 5])), # Summer

(df\_stations["Timestamp"].dt.month.isin([6, 7, 8, 9])) # monsoon

]

# Corresponding seasons for each condition

seasons = ["Winter", "Summer", "Monsoon"]

# Assign a season to each row in df\_stations w np.select()

df\_stations["Season"] = np.select(get\_season, seasons, default="Other")

# Calculate average PM2.5 for each season

avg\_pm25\_per\_season = df\_stations.groupby("Season")["PM2.5"].mean()

# Get season w the highest pollution

most\_polluted\_season = avg\_pm25\_per\_season.idxmax()

highest\_avg\_pm25 = avg\_pm25\_per\_season.max()

print(f"The average PM2.5 concentration for each season for stations Lal Bahadur Shastri Nagar, Kalaburagi - KSPCBavg\_pm25\_per\_season is:\n{avg\_pm25\_per\_season}")

print(f"\nThe most polluted season in 2018 was {most\_polluted\_season} with an average PM2.5 concentration of {highest\_avg\_pm25:.3f}")

34/79:

import pandas as pd

import numpy as np

# Load the dataset

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

df["Timestamp"] = pd.to\_datetime(df["Timestamp"], format = "%Y-%m-%d") # Convert Timestamp to datetime format

# Get rows for year 2018

df\_2018 = df[df["Timestamp"].dt.year == 2018]

# Mask for the given stations

df\_stations = df\_2018[df\_2018["station"] == "Lal Bahadur Shastri Nagar, Kalaburagi - KSPCB"]

df\_stations = df\_stations.dropna(subset = ["PM2.5"]) # Drop rows w PM2.5 missing

# Define seasons

get\_season = [

(df\_stations["Timestamp"].dt.month.isin([12, 1, 2])), # winter

(df\_stations["Timestamp"].dt.month.isin([3, 4, 5])), # Summer

(df\_stations["Timestamp"].dt.month.isin([6, 7, 8, 9])) # monsoon

]

# Corresponding seasons for each condition

seasons = ["Winter", "Summer", "Monsoon"]

# Assign a season to each row in df\_stations w np.select()

df\_stations["Season"] = np.select(get\_season, seasons, default="Other")

# Calculate average PM2.5 for each season

avg\_pm25\_per\_season = df\_stations.groupby("Season")["PM2.5"].mean()

# Get season w the highest pollution

most\_polluted\_season = avg\_pm25\_per\_season.idxmax()

highest\_avg\_pm25 = avg\_pm25\_per\_season.max()

print(f"The average PM2.5 concentration for each season for stations Lal Bahadur Shastri Nagar, Kalaburagi - KSPCB is:\n{avg\_pm25\_per\_season}")

print(f"\nThe most polluted season in 2018 was {most\_polluted\_season} with an average PM2.5 concentration of {highest\_avg\_pm25:.3f}")

34/80:

import pandas as pd

import numpy as np

# Load the dataset

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

df["Timestamp"] = pd.to\_datetime(df["Timestamp"], format = "%Y-%m-%d") # Convert Timestamp to datetime format

# Get rows for year 2018

df\_2018 = df[df["Timestamp"].dt.year == 2018]

# Mask for the given stations

df\_stations = df\_2018[df\_2018["station"] == "Lal Bahadur Shastri Nagar, Kalaburagi - KSPCB"]

df\_stations = df\_stations.dropna(subset = ["PM2.5"]) # Drop rows w PM2.5 missing

print(df\_stations)

# Define seasons

get\_season = [

(df\_stations["Timestamp"].dt.month.isin([12, 1, 2])), # winter

(df\_stations["Timestamp"].dt.month.isin([3, 4, 5])), # Summer

(df\_stations["Timestamp"].dt.month.isin([6, 7, 8, 9])) # monsoon

]

# Corresponding seasons for each condition

seasons = ["Winter", "Summer", "Monsoon"]

# Assign a season to each row in df\_stations w np.select()

df\_stations["Season"] = np.select(get\_season, seasons, default="Other")

# Calculate average PM2.5 for each season

avg\_pm25\_per\_season = df\_stations.groupby("Season")["PM2.5"].mean()

# Get season w the highest pollution

most\_polluted\_season = avg\_pm25\_per\_season.idxmax()

highest\_avg\_pm25 = avg\_pm25\_per\_season.max()

print(f"The average PM2.5 concentration for each season for stations Lal Bahadur Shastri Nagar, Kalaburagi - KSPCB is:\n{avg\_pm25\_per\_season}")

print(f"\nThe most polluted season in 2018 was {most\_polluted\_season} with an average PM2.5 concentration of {highest\_avg\_pm25:.3f}")

34/81:

import pandas as pd

import numpy as np

# Load the dataset

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

df["Timestamp"] = pd.to\_datetime(df["Timestamp"], format = "%Y-%m-%d") # Convert Timestamp to datetime format

# Get rows for year 2018

df\_2018 = df[df["Timestamp"].dt.year == 2018]

# Mask for the given stations

df\_stations = df\_2018[df\_2018["station"] == "Lal Bahadur Shastri Nagar, Kalaburagi - KSPCB"]

df\_stations = df\_stations.dropna(subset = ["PM2.5"]) # Drop rows w PM2.5 missing

# Define seasons

get\_season = [

(df\_stations["Timestamp"].dt.month.isin([12, 1, 2])), # winter

(df\_stations["Timestamp"].dt.month.isin([3, 4, 5])), # Summer

(df\_stations["Timestamp"].dt.month.isin([6, 7, 8, 9])) # monsoon

]

# Corresponding seasons for each condition

seasons = ["Winter", "Summer", "Monsoon"]

# Assign a season to each row in df\_stations w np.select()

df\_stations["Season"] = np.select(get\_season, seasons, default="Other")

print(df\_stations.loc["Season" == "Summer")

# Calculate average PM2.5 for each season

avg\_pm25\_per\_season = df\_stations.groupby("Season")["PM2.5"].mean()

# Get season w the highest pollution

most\_polluted\_season = avg\_pm25\_per\_season.idxmax()

highest\_avg\_pm25 = avg\_pm25\_per\_season.max()

print(f"The average PM2.5 concentration for each season for stations Lal Bahadur Shastri Nagar, Kalaburagi - KSPCB is:\n{avg\_pm25\_per\_season}")

print(f"\nThe most polluted season in 2018 was {most\_polluted\_season} with an average PM2.5 concentration of {highest\_avg\_pm25:.3f}")

34/82:

import pandas as pd

import numpy as np

# Load the dataset

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

df["Timestamp"] = pd.to\_datetime(df["Timestamp"], format = "%Y-%m-%d") # Convert Timestamp to datetime format

# Get rows for year 2018

df\_2018 = df[df["Timestamp"].dt.year == 2018]

# Mask for the given stations

df\_stations = df\_2018[df\_2018["station"] == "Lal Bahadur Shastri Nagar, Kalaburagi - KSPCB"]

df\_stations = df\_stations.dropna(subset = ["PM2.5"]) # Drop rows w PM2.5 missing

# Define seasons

get\_season = [

(df\_stations["Timestamp"].dt.month.isin([12, 1, 2])), # winter

(df\_stations["Timestamp"].dt.month.isin([3, 4, 5])), # Summer

(df\_stations["Timestamp"].dt.month.isin([6, 7, 8, 9])) # monsoon

]

# Corresponding seasons for each condition

seasons = ["Winter", "Summer", "Monsoon"]

# Assign a season to each row in df\_stations w np.select()

df\_stations["Season"] = np.select(get\_season, seasons, default="Other")

print(df\_stations.loc["Season" == "Summer"]

# Calculate average PM2.5 for each season

avg\_pm25\_per\_season = df\_stations.groupby("Season")["PM2.5"].mean()

# Get season w the highest pollution

most\_polluted\_season = avg\_pm25\_per\_season.idxmax()

highest\_avg\_pm25 = avg\_pm25\_per\_season.max()

print(f"The average PM2.5 concentration for each season for stations Lal Bahadur Shastri Nagar, Kalaburagi - KSPCB is:\n{avg\_pm25\_per\_season}")

print(f"\nThe most polluted season in 2018 was {most\_polluted\_season} with an average PM2.5 concentration of {highest\_avg\_pm25:.3f}")

34/83:

import pandas as pd

import numpy as np

# Load the dataset

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

df["Timestamp"] = pd.to\_datetime(df["Timestamp"], format = "%Y-%m-%d") # Convert Timestamp to datetime format

# Get rows for year 2018

df\_2018 = df[df["Timestamp"].dt.year == 2018]

# Mask for the given stations

df\_stations = df\_2018[df\_2018["station"] == "Lal Bahadur Shastri Nagar, Kalaburagi - KSPCB"]

df\_stations = df\_stations.dropna(subset = ["PM2.5"]) # Drop rows w PM2.5 missing

# Define seasons

get\_season = [

(df\_stations["Timestamp"].dt.month.isin([12, 1, 2])), # winter

(df\_stations["Timestamp"].dt.month.isin([3, 4, 5])), # Summer

(df\_stations["Timestamp"].dt.month.isin([6, 7, 8, 9])) # monsoon

]

# Corresponding seasons for each condition

seasons = ["Winter", "Summer", "Monsoon"]

# Assign a season to each row in df\_stations w np.select()

df\_stations["Season"] = np.select(get\_season, seasons, default="Other")

print(df\_stations.loc["Season" == "Summer"])

# Calculate average PM2.5 for each season

avg\_pm25\_per\_season = df\_stations.groupby("Season")["PM2.5"].mean()

# Get season w the highest pollution

most\_polluted\_season = avg\_pm25\_per\_season.idxmax()

highest\_avg\_pm25 = avg\_pm25\_per\_season.max()

print(f"The average PM2.5 concentration for each season for stations Lal Bahadur Shastri Nagar, Kalaburagi - KSPCB is:\n{avg\_pm25\_per\_season}")

print(f"\nThe most polluted season in 2018 was {most\_polluted\_season} with an average PM2.5 concentration of {highest\_avg\_pm25:.3f}")

34/84:

import pandas as pd

import numpy as np

# Load the dataset

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

df["Timestamp"] = pd.to\_datetime(df["Timestamp"], format = "%Y-%m-%d") # Convert Timestamp to datetime format

# Get rows for year 2018

df\_2018 = df[df["Timestamp"].dt.year == 2018]

# Mask for the given stations

df\_stations = df\_2018[df\_2018["station"] == "Lal Bahadur Shastri Nagar, Kalaburagi - KSPCB"]

df\_stations = df\_stations.dropna(subset = ["PM2.5"]) # Drop rows w PM2.5 missing

# Define seasons

get\_season = [

(df\_stations["Timestamp"].dt.month.isin([12, 1, 2])), # winter

(df\_stations["Timestamp"].dt.month.isin([3, 4, 5])), # Summer

(df\_stations["Timestamp"].dt.month.isin([6, 7, 8, 9])) # monsoon

]

# Corresponding seasons for each condition

seasons = ["Winter", "Summer", "Monsoon"]

# Assign a season to each row in df\_stations w np.select()

df\_stations["Season"] = np.select(get\_season, seasons, default="Other")

df\_stations[df\_stations["Season"] == "Summer"]

# Calculate average PM2.5 for each season

avg\_pm25\_per\_season = df\_stations.groupby("Season")["PM2.5"].mean()

# Get season w the highest pollution

most\_polluted\_season = avg\_pm25\_per\_season.idxmax()

highest\_avg\_pm25 = avg\_pm25\_per\_season.max()

print(f"The average PM2.5 concentration for each season for stations Lal Bahadur Shastri Nagar, Kalaburagi - KSPCB is:\n{avg\_pm25\_per\_season}")

print(f"\nThe most polluted season in 2018 was {most\_polluted\_season} with an average PM2.5 concentration of {highest\_avg\_pm25:.3f}")

34/85:

import pandas as pd

import numpy as np

# Load the dataset

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

df["Timestamp"] = pd.to\_datetime(df["Timestamp"], format = "%Y-%m-%d") # Convert Timestamp to datetime format

# Get rows for year 2018

df\_2018 = df[df["Timestamp"].dt.year == 2018]

# Mask for the given stations

df\_stations = df\_2018[df\_2018["station"] == "Lal Bahadur Shastri Nagar, Kalaburagi - KSPCB"]

df\_stations = df\_stations.dropna(subset = ["PM2.5"]) # Drop rows w PM2.5 missing

# Define seasons

get\_season = [

(df\_stations["Timestamp"].dt.month.isin([12, 1, 2])), # winter

(df\_stations["Timestamp"].dt.month.isin([3, 4, 5])), # Summer

(df\_stations["Timestamp"].dt.month.isin([6, 7, 8, 9])) # monsoon

]

# Corresponding seasons for each condition

seasons = ["Winter", "Summer", "Monsoon"]

# Assign a season to each row in df\_stations w np.select()

df\_stations["Season"] = np.select(get\_season, seasons, default="Other")

df\_stations["hi", df\_stations["Season"] == "Summer", "j"]

# Calculate average PM2.5 for each season

avg\_pm25\_per\_season = df\_stations.groupby("Season")["PM2.5"].mean()

# Get season w the highest pollution

most\_polluted\_season = avg\_pm25\_per\_season.idxmax()

highest\_avg\_pm25 = avg\_pm25\_per\_season.max()

print(f"The average PM2.5 concentration for each season for stations Lal Bahadur Shastri Nagar, Kalaburagi - KSPCB is:\n{avg\_pm25\_per\_season}")

print(f"\nThe most polluted season in 2018 was {most\_polluted\_season} with an average PM2.5 concentration of {highest\_avg\_pm25:.3f}")

34/86:

import pandas as pd

import numpy as np

# Load the dataset

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

df["Timestamp"] = pd.to\_datetime(df["Timestamp"], format = "%Y-%m-%d") # Convert Timestamp to datetime format

# Get rows for year 2018

df\_2018 = df[df["Timestamp"].dt.year == 2018]

# Mask for the given stations

df\_stations = df\_2018[df\_2018["station"] == "Lal Bahadur Shastri Nagar, Kalaburagi - KSPCB"]

df\_stations = df\_stations.dropna(subset = ["PM2.5"]) # Drop rows w PM2.5 missing

# Define seasons

get\_season = [

(df\_stations["Timestamp"].dt.month.isin([12, 1, 2])), # winter

(df\_stations["Timestamp"].dt.month.isin([3, 4, 5])), # Summer

(df\_stations["Timestamp"].dt.month.isin([6, 7, 8, 9])) # monsoon

]

# Corresponding seasons for each condition

seasons = ["Winter", "Summer", "Monsoon"]

# Assign a season to each row in df\_stations w np.select()

df\_stations["Season"] = np.select(get\_season, seasons, default="Other")

print(df\_stations[df\_stations["Season"] == "Summer"])

# Calculate average PM2.5 for each season

avg\_pm25\_per\_season = df\_stations.groupby("Season")["PM2.5"].mean()

# Get season w the highest pollution

most\_polluted\_season = avg\_pm25\_per\_season.idxmax()

highest\_avg\_pm25 = avg\_pm25\_per\_season.max()

print(f"The average PM2.5 concentration for each season for stations Lal Bahadur Shastri Nagar, Kalaburagi - KSPCB is:\n{avg\_pm25\_per\_season}")

print(f"\nThe most polluted season in 2018 was {most\_polluted\_season} with an average PM2.5 concentration of {highest\_avg\_pm25:.3f}")

34/87:

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

34/88:

import pandas as pd

import matplotlib.pyplot as plt

# Load the dataset

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

df["Timestamp"] = pd.to\_datetime(df["Timestamp"], format = "%Y-%m-%d") # Convert Timestamp to datetime format

# Get rows for year 2021, given station

df\_2021 = df[(df["Timestamp"].dt.year == 2021) & (df\_2021["station"] == "Lal Bahadur Shastri Nagar, Kalaburagi - KSPCB")]

df\_2021 = df\_2021.dropna(subset = ["PM2.5"]) # Drop rows w PM2.5 missing

# Get month and day of the week

conditions = [df\_2021["Day\_of\_week"] < 5, df\_2021["Day\_of\_week"] >= 5]

choices = ["Weekday", "Weekend"]

# Assign weekday or weekend

df\_2021["Type"] = np.select(conditions, choices)

# Group by month and type, then calc the avg PM2.5

monthly\_avg\_pm25 = df\_2021.groupby(["Month", "Type"])["PM2.5"].mean().unstack() # turn multi indexed series to dataframe, for easy plotting

# Plot

plt.figure(figsize=(10, 5))

plt.plot(monthly\_avg\_pm25.index, monthly\_avg\_pm25["Weekday"], marker="o", linestyle="-", label="Weekdays", color="blue")

plt.plot(monthly\_avg\_pm25.index, monthly\_avg\_pm25["Weekend"], marker="x", linestyle="---", label="Weekends", color="red")

plt.xlabel("Month")

plt.ylabel("Average PM2.5 Levels")

plt.title("Monthly Average PM2.5 Levels: Weekdays vs. Weekends (2021)")

plt.xticks(range(1, 13), ["Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"])

plt.legend()

plt.grid(True)

plt.show()

34/89:

import pandas as pd

import matplotlib.pyplot as plt

# Load the dataset

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

df["Timestamp"] = pd.to\_datetime(df["Timestamp"], format = "%Y-%m-%d") # Convert Timestamp to datetime format

# Get rows for year 2021, given station

df\_2021 = df[(df["Timestamp"].dt.year == 2021) & (df["station"] == "Lal Bahadur Shastri Nagar, Kalaburagi - KSPCB")]

df\_2021 = df\_2021.dropna(subset = ["PM2.5"]) # Drop rows w PM2.5 missing

# Get month and day of the week

conditions = [df\_2021["Day\_of\_week"] < 5, df\_2021["Day\_of\_week"] >= 5]

choices = ["Weekday", "Weekend"]

# Assign weekday or weekend

df\_2021["Type"] = np.select(conditions, choices)

# Group by month and type, then calc the avg PM2.5

monthly\_avg\_pm25 = df\_2021.groupby(["Month", "Type"])["PM2.5"].mean().unstack() # turn multi indexed series to dataframe, for easy plotting

# Plot

plt.figure(figsize=(10, 5))

plt.plot(monthly\_avg\_pm25.index, monthly\_avg\_pm25["Weekday"], marker="o", linestyle="-", label="Weekdays", color="blue")

plt.plot(monthly\_avg\_pm25.index, monthly\_avg\_pm25["Weekend"], marker="x", linestyle="---", label="Weekends", color="red")

plt.xlabel("Month")

plt.ylabel("Average PM2.5 Levels")

plt.title("Monthly Average PM2.5 Levels: Weekdays vs. Weekends (2021)")

plt.xticks(range(1, 13), ["Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"])

plt.legend()

plt.grid(True)

plt.show()

34/90:

import pandas as pd

import matplotlib.pyplot as plt

# Load the dataset

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

df["Timestamp"] = pd.to\_datetime(df["Timestamp"], format = "%Y-%m-%d") # Convert Timestamp to datetime format

df\_2021["Day\_of\_week"] = df\_2021["Timestamp"].dt.dayofweek # Monday = 0, Sunday = 6

# Get rows for year 2021, given station

df\_2021 = df[(df["Timestamp"].dt.year == 2021) & (df["station"] == "Lal Bahadur Shastri Nagar, Kalaburagi - KSPCB")]

df\_2021 = df\_2021.dropna(subset = ["PM2.5"]) # Drop rows w PM2.5 missing

# Get month and day of the week

conditions = [df\_2021["Day\_of\_week"] < 5, df\_2021["Day\_of\_week"] >= 5]

choices = ["Weekday", "Weekend"]

# Assign weekday or weekend

df\_2021["Type"] = np.select(conditions, choices)

# Group by month and type, then calc the avg PM2.5

monthly\_avg\_pm25 = df\_2021.groupby(["Month", "Type"])["PM2.5"].mean().unstack() # turn multi indexed series to dataframe, for easy plotting

# Plot

plt.figure(figsize=(10, 5))

plt.plot(monthly\_avg\_pm25.index, monthly\_avg\_pm25["Weekday"], marker="o", linestyle="-", label="Weekdays", color="blue")

plt.plot(monthly\_avg\_pm25.index, monthly\_avg\_pm25["Weekend"], marker="x", linestyle="---", label="Weekends", color="red")

plt.xlabel("Month")

plt.ylabel("Average PM2.5 Levels")

plt.title("Monthly Average PM2.5 Levels: Weekdays vs. Weekends (2021)")

plt.xticks(range(1, 13), ["Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"])

plt.legend()

plt.grid(True)

plt.show()

34/91:

import pandas as pd

import matplotlib.pyplot as plt

# Load the dataset

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

df["Timestamp"] = pd.to\_datetime(df["Timestamp"], format = "%Y-%m-%d") # Convert Timestamp to datetime format

# Get rows for year 2021, given station, make a new col for days

df\_2021 = df[(df["Timestamp"].dt.year == 2021) & (df["station"] == "Lal Bahadur Shastri Nagar, Kalaburagi - KSPCB")]

df\_2021 = df\_2021.dropna(subset = ["PM2.5"]) # Drop rows w PM2.5 missing

df\_2021["Day\_of\_week"] = df\_2021["Timestamp"].dt.dayofweek # Monday = 0, Sunday = 6

# Get month and day of the week

conditions = [df\_2021["Day\_of\_week"] < 5, df\_2021["Day\_of\_week"] >= 5]

choices = ["Weekday", "Weekend"]

# Assign weekday or weekend

df\_2021["Type"] = np.select(conditions, choices)

# Group by month and type, then calc the avg PM2.5

monthly\_avg\_pm25 = df\_2021.groupby(["Month", "Type"])["PM2.5"].mean().unstack() # turn multi indexed series to dataframe, for easy plotting

# Plot

plt.figure(figsize=(10, 5))

plt.plot(monthly\_avg\_pm25.index, monthly\_avg\_pm25["Weekday"], marker="o", linestyle="-", label="Weekdays", color="blue")

plt.plot(monthly\_avg\_pm25.index, monthly\_avg\_pm25["Weekend"], marker="x", linestyle="---", label="Weekends", color="red")

plt.xlabel("Month")

plt.ylabel("Average PM2.5 Levels")

plt.title("Monthly Average PM2.5 Levels: Weekdays vs. Weekends (2021)")

plt.xticks(range(1, 13), ["Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"])

plt.legend()

plt.grid(True)

plt.show()

34/92:

import pandas as pd

import matplotlib.pyplot as plt

# Load the dataset

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

df["Timestamp"] = pd.to\_datetime(df["Timestamp"], format = "%Y-%m-%d") # Convert Timestamp to datetime format

# Get rows for year 2021, given station, make a new col for days

df\_2021 = df[(df["Timestamp"].dt.year == 2021) & (df["station"] == "Lal Bahadur Shastri Nagar, Kalaburagi - KSPCB")]

df\_2021 = df\_2021.dropna(subset = ["PM2.5"]) # Drop rows w PM2.5 missing

df\_2021["Day\_of\_week"] = df\_2021["Timestamp"].dt.dayofweek # Monday = 0, Sunday = 6

# Get month and day of the week

conditions = [df\_2021["Day\_of\_week"] < 5, df\_2021["Day\_of\_week"] >= 5]

choices = ["Weekday", "Weekend"]

# Assign weekday or weekend

df\_2021["Type"] = np.select(conditions, choices, default = "Weekday")

# Group by month and type, then calc the avg PM2.5

monthly\_avg\_pm25 = df\_2021.groupby(["Month", "Type"])["PM2.5"].mean().unstack() # turn multi indexed series to dataframe, for easy plotting

# Plot

plt.figure(figsize=(10, 5))

plt.plot(monthly\_avg\_pm25.index, monthly\_avg\_pm25["Weekday"], marker="o", linestyle="-", label="Weekdays", color="blue")

plt.plot(monthly\_avg\_pm25.index, monthly\_avg\_pm25["Weekend"], marker="x", linestyle="---", label="Weekends", color="red")

plt.xlabel("Month")

plt.ylabel("Average PM2.5 Levels")

plt.title("Monthly Average PM2.5 Levels: Weekdays vs. Weekends (2021)")

plt.xticks(range(1, 13), ["Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"])

plt.legend()

plt.grid(True)

plt.show()

34/93:

import pandas as pd

import matplotlib.pyplot as plt

# Load the dataset

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

df["Timestamp"] = pd.to\_datetime(df["Timestamp"], format = "%Y-%m-%d") # Convert Timestamp to datetime format

# Get rows for year 2021, given station, make a new col for days

df\_2021 = df[(df["Timestamp"].dt.year == 2021) & (df["station"] == "Lal Bahadur Shastri Nagar, Kalaburagi - KSPCB")]

df\_2021 = df\_2021.dropna(subset = ["PM2.5"]) # Drop rows w PM2.5 missing

df\_2021["Day\_of\_week"] = df\_2021["Timestamp"].dt.dayofweek # Monday = 0, Sunday = 6

# Get month and day of the week

df\_2021["Month"] = df\_2021["Timestamp"].dt.month

conditions = [df\_2021["Day\_of\_week"] < 5, df\_2021["Day\_of\_week"] >= 5]

choices = ["Weekday", "Weekend"]

# Assign weekday or weekend

df\_2021["Type"] = np.select(conditions, choices, default = "Weekday")

# Group by month and type, then calc the avg PM2.5

monthly\_avg\_pm25 = df\_2021.groupby(["Month", "Type"])["PM2.5"].mean().unstack() # turn multi indexed series to dataframe, for easy plotting

# Plot

plt.figure(figsize=(10, 5))

plt.plot(monthly\_avg\_pm25.index, monthly\_avg\_pm25["Weekday"], marker="o", linestyle="-", label="Weekdays", color="blue")

plt.plot(monthly\_avg\_pm25.index, monthly\_avg\_pm25["Weekend"], marker="x", linestyle="---", label="Weekends", color="red")

plt.xlabel("Month")

plt.ylabel("Average PM2.5 Levels")

plt.title("Monthly Average PM2.5 Levels: Weekdays vs. Weekends (2021)")

plt.xticks(range(1, 13), ["Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"])

plt.legend()

plt.grid(True)

plt.show()

34/94:

import pandas as pd

import matplotlib.pyplot as plt

# Load the dataset

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

df["Timestamp"] = pd.to\_datetime(df["Timestamp"], format = "%Y-%m-%d") # Convert Timestamp to datetime format

# Get rows for year 2021, given station, make a new col for days

df\_2021 = df[(df["Timestamp"].dt.year == 2021) & (df["station"] == "Lal Bahadur Shastri Nagar, Kalaburagi - KSPCB")]

df\_2021 = df\_2021.dropna(subset = ["PM2.5"]) # Drop rows w PM2.5 missing

df\_2021["Day\_of\_week"] = df\_2021["Timestamp"].dt.dayofweek # Monday = 0, Sunday = 6

# Get month and day of the week

df\_2021["Month"] = df\_2021["Timestamp"].dt.month

conditions = [df\_2021["Day\_of\_week"] < 5, df\_2021["Day\_of\_week"] >= 5]

choices = ["Weekday", "Weekend"]

# Assign weekday or weekend

df\_2021["Type"] = np.select(conditions, choices, default = "Weekday")

# Group by month and type, then calc the avg PM2.5

monthly\_avg\_pm25 = df\_2021.groupby(["Month", "Type"])["PM2.5"].mean().unstack() # turn multi indexed series to dataframe, for easy plotting

# Plot

plt.figure(figsize=(10, 5))

plt.plot(monthly\_avg\_pm25.index, monthly\_avg\_pm25["Weekday"], marker="o", linestyle="-", label="Weekdays", color="blue")

plt.plot(monthly\_avg\_pm25.index, monthly\_avg\_pm25["Weekend"], marker="x", linestyle="--", label="Weekends", color="red")

plt.xlabel("Month")

plt.ylabel("Average PM2.5 Levels")

plt.title("Monthly Average PM2.5 Levels: Weekdays vs. Weekends (2021)")

plt.xticks(range(1, 13), ["Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"])

plt.legend()

plt.grid(True)

plt.show()

34/95:

import pandas as pd

import matplotlib.pyplot as plt

# Load the dataset

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

df["Timestamp"] = pd.to\_datetime(df["Timestamp"], format = "%Y-%m-%d") # Convert Timestamp to datetime format

# Get rows for year 2021, given station, make a new col for days

df\_2021 = df[(df["Timestamp"].dt.year == 2021) & (df["station"] == "Lal Bahadur Shastri Nagar, Kalaburagi - KSPCB")]

df\_2021 = df\_2021.dropna(subset = ["PM2.5"]) # Drop rows w PM2.5 missing

df\_2021["Day\_of\_week"] = df\_2021["Timestamp"].dt.dayofweek # Monday = 0, Sunday = 6

# Get month and day of the week

df\_2021["Month"] = df\_2021["Timestamp"].dt.month

conditions = [df\_2021["Day\_of\_week"] < 5, df\_2021["Day\_of\_week"] >= 5]

choices = ["Weekday", "Weekend"]

# Assign weekday or weekend

df\_2021["Type"] = np.select(conditions, choices, default = "Weekday")

# Group by month and type, then calc the avg PM2.5

monthly\_avg\_pm25 = df\_2021.groupby(["Month", "Type"])["PM2.5"].mean().unstack() # turn multi indexed series to dataframe, for easy plotting

# Plot

plt.figure(figsize=(10, 5))

plt.plot(monthly\_avg\_pm25.index, monthly\_avg\_pm25["Weekday"], marker="o", linestyle="-", label="Weekdays", color="blue")

plt.plot(monthly\_avg\_pm25.index, monthly\_avg\_pm25["Weekend"], marker="x", linestyle="--", label="Weekends", color="red")

plt.xlabel("Month")

plt.ylabel("Average PM2.5 Levels")

plt.title("Monthly Average PM2.5 Levels (2021)\nStation: Lal Bahadur Shastri Nagar, Kalaburagi - KSPCB")

plt.xticks(range(1, 13), ["Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"])

plt.legend()

plt.grid(True)

plt.show()

34/96:

import pandas as pd

import matplotlib.pyplot as plt

# Load the dataset

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

df["Timestamp"] = pd.to\_datetime(df["Timestamp"], format = "%Y-%m-%d") # Convert Timestamp to datetime format

# Get rows for year 2021, given station, make a new col for days

df\_2021 = df[(df["Timestamp"].dt.year == 2021) & (df["station"] == "Lal Bahadur Shastri Nagar, Kalaburagi - KSPCB")]

df\_2021 = df\_2021.dropna(subset = ["PM2.5"]) # Drop rows w PM2.5 missing

# Get month and day of the week

df\_2021["Month"] = df\_2021["Timestamp"].dt.month

df\_2021["Day\_of\_week"] = df\_2021["Timestamp"].dt.dayofweek # Monday = 0, Sunday = 6

conditions = [df\_2021["Day\_of\_week"] < 5, df\_2021["Day\_of\_week"] >= 5]

choices = ["Weekday", "Weekend"]

# Assign weekday or weekend

df\_2021["Type"] = np.select(conditions, choices, default = "Weekday")

# Group by month and type, then calc the avg PM2.5

monthly\_avg\_pm25 = df\_2021.groupby(["Month", "Type"])["PM2.5"].mean().unstack() # turn multi indexed series to dataframe, for easy plotting

# Plot

plt.figure(figsize=(10, 5))

plt.plot(monthly\_avg\_pm25.index, monthly\_avg\_pm25["Weekday"], marker="o", linestyle="-", label="Weekdays", color="blue")

plt.plot(monthly\_avg\_pm25.index, monthly\_avg\_pm25["Weekend"], marker="x", linestyle="--", label="Weekends", color="red")

plt.xlabel("Month")

plt.ylabel("Average PM2.5 Levels")

plt.title("Monthly Average PM2.5 Levels (2021)\nStation: Lal Bahadur Shastri Nagar, Kalaburagi - KSPCB")

plt.xticks(range(1, 13), ["Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"])

plt.legend()

plt.grid(True)

plt.show()

34/97:

import pandas as pd

import matplotlib.pyplot as plt

# Load the dataset

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

df["Timestamp"] = pd.to\_datetime(df["Timestamp"], format = "%Y-%m-%d") # Convert Timestamp to datetime format

# Get rows for year 2021, given station, make a new col for days

df\_2021 = df[(df["Timestamp"].dt.year == 2021) & (df["station"] == "Lal Bahadur Shastri Nagar, Kalaburagi - KSPCB")]

df\_2021 = df\_2021.dropna(subset = ["PM2.5"]) # Drop rows w PM2.5 missing

# Get month and day of the week

df\_2021["Month"] = df\_2021["Timestamp"].dt.month

df\_2021["Day\_of\_week"] = df\_2021["Timestamp"].dt.dayofweek # Monday = 0, Sunday = 6

conditions = [df\_2021["Day\_of\_week"] < 5, df\_2021["Day\_of\_week"] >= 5]

choices = ["Weekday", "Weekend"]

# Assign weekday or weekend

df\_2021["Type"] = np.select(conditions, choices, default = "Weekday")

# Group by month and type, then calc the avg PM2.5

monthly\_avg\_pm25 = df\_2021.groupby(["Month", "Type"])["PM2.5"].mean().unstack() # turn multi indexed series to dataframe, for easy plotting

# Plot

plt.figure(figsize=(10, 5))

plt.plot(monthly\_avg\_pm25.index, monthly\_avg\_pm25["Weekday"], marker="o", linestyle="-", label="Weekdays", color="blue")

plt.plot(monthly\_avg\_pm25.index, monthly\_avg\_pm25["Weekend"], marker="s", linestyle="--", label="Weekends", color="red")

plt.xlabel("Month")

plt.ylabel("Average PM2.5 Levels")

plt.title("Monthly Average PM2.5 Levels (2021)\nStation: Lal Bahadur Shastri Nagar, Kalaburagi - KSPCB")

plt.xticks(range(1, 13), ["Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"])

plt.legend()

plt.show()

34/98:

import pandas as pd

import matplotlib.pyplot as plt

# Load the dataset

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

df["Timestamp"] = pd.to\_datetime(df["Timestamp"], format = "%Y-%m-%d") # Convert Timestamp to datetime format

# Get rows for year 2021, given station, make a new col for days

df\_2021 = df[(df["Timestamp"].dt.year == 2021) & (df["station"] == "Lal Bahadur Shastri Nagar, Kalaburagi - KSPCB")]

# df\_2021 = df\_2021.dropna(subset = ["PM2.5"]) # Drop rows w PM2.5 missing

# Get month and day of the week

df\_2021["Month"] = df\_2021["Timestamp"].dt.month

df\_2021["Day\_of\_week"] = df\_2021["Timestamp"].dt.dayofweek # Monday = 0, Sunday = 6

conditions = [df\_2021["Day\_of\_week"] < 5, df\_2021["Day\_of\_week"] >= 5]

choices = ["Weekday", "Weekend"]

# Assign weekday or weekend

df\_2021["Type"] = np.select(conditions, choices, default = "Weekday")

# Group by month and type, then calc the avg PM2.5

monthly\_avg\_pm25 = df\_2021.groupby(["Month", "Type"])["PM2.5"].mean().unstack() # turn multi indexed series to dataframe, for easy plotting

# Plot

plt.figure(figsize=(10, 5))

plt.plot(monthly\_avg\_pm25.index, monthly\_avg\_pm25["Weekday"], marker="o", linestyle="-", label="Weekdays", color="blue")

plt.plot(monthly\_avg\_pm25.index, monthly\_avg\_pm25["Weekend"], marker="s", linestyle="--", label="Weekends", color="red")

plt.xlabel("Month")

plt.ylabel("Average PM2.5 Levels")

plt.title("Monthly Average PM2.5 Levels (2021)\nStation: Lal Bahadur Shastri Nagar, Kalaburagi - KSPCB")

plt.xticks(range(1, 13), ["Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"])

plt.legend()

plt.show()

34/99:

import pandas as pd

import matplotlib.pyplot as plt

# Load the dataset

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

df["Timestamp"] = pd.to\_datetime(df["Timestamp"], format = "%Y-%m-%d") # Convert Timestamp to datetime format

# Get rows for year 2021, given station, make a new col for days

df\_2021 = df[(df["Timestamp"].dt.year == 2021) & (df["station"] == "Lal Bahadur Shastri Nagar, Kalaburagi - KSPCB")]

df\_2021 = df\_2021.dropna(subset = ["PM2.5"]) # Drop rows w PM2.5 missing

# Get month and day of the week

df\_2021["Month"] = df\_2021["Timestamp"].dt.month

df\_2021["Day\_of\_week"] = df\_2021["Timestamp"].dt.dayofweek # Monday = 0, Sunday = 6

conditions = [df\_2021["Day\_of\_week"] < 5, df\_2021["Day\_of\_week"] >= 5]

choices = ["Weekday", "Weekend"]

# Assign weekday or weekend

df\_2021["Type"] = np.select(conditions, choices, default = "Weekday")

# Group by month and type, then calc the avg PM2.5

monthly\_avg\_pm25 = df\_2021.groupby(["Month", "Type"])["PM2.5"].mean().unstack() # turn multi indexed series to dataframe, for easy plotting

# Plot

plt.figure(figsize=(10, 5))

plt.plot(monthly\_avg\_pm25.index, monthly\_avg\_pm25["Weekday"], marker="o", linestyle="-", label="Weekdays", color="blue")

plt.plot(monthly\_avg\_pm25.index, monthly\_avg\_pm25["Weekend"], marker="s", linestyle="--", label="Weekends", color="red")

plt.xlabel("Month")

plt.ylabel("Average PM2.5 Levels")

plt.title("Monthly Average PM2.5 Levels (2021)\nStation: Lal Bahadur Shastri Nagar, Kalaburagi - KSPCB")

plt.xticks(range(1, 13), ["Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"])

plt.legend()

plt.show()

34/100:

import pandas as pd

import matplotlib.pyplot as plt

# Load the dataset

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

df["Timestamp"] = pd.to\_datetime(df["Timestamp"], format = "%Y-%m-%d") # Convert Timestamp to datetime format

# Get rows for year 2021, given station, make a new col for days

df\_2021 = df[(df["Timestamp"].dt.year == 2021) & (df["station"] == "Lal Bahadur Shastri Nagar, Kalaburagi - KSPCB")]

# df\_2021 = df\_2021.dropna(subset = ["PM2.5"]) # Drop rows w PM2.5 missing

# Get month and day of the week

df\_2021["Month"] = df\_2021["Timestamp"].dt.month

df\_2021["Day\_of\_week"] = df\_2021["Timestamp"].dt.dayofweek # Monday = 0, Sunday = 6

conditions = [df\_2021["Day\_of\_week"] < 5, df\_2021["Day\_of\_week"] >= 5]

choices = ["Weekday", "Weekend"]

# Assign weekday or weekend

df\_2021["Type"] = np.select(conditions, choices, default = "Weekday")

# Group by month and type, then calc the avg PM2.5

monthly\_avg\_pm25 = df\_2021.groupby(["Month", "Type"])["PM2.5"].mean().unstack() # turn multi indexed series to dataframe, for easy plotting

# Plot

plt.figure(figsize=(10, 5))

plt.plot(monthly\_avg\_pm25.index, monthly\_avg\_pm25["Weekday"], marker="o", linestyle="-", label="Weekdays", color="blue")

plt.plot(monthly\_avg\_pm25.index, monthly\_avg\_pm25["Weekend"], marker="s", linestyle="--", label="Weekends", color="red")

plt.xlabel("Month")

plt.ylabel("Average PM2.5 Levels")

plt.title("Monthly Average PM2.5 Levels (2021)\nStation: Lal Bahadur Shastri Nagar, Kalaburagi - KSPCB")

plt.xticks(range(1, 13), ["Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"])

plt.legend()

plt.show()

34/101:

import pandas as pd

import matplotlib.pyplot as plt

# Load the dataset

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

df["Timestamp"] = pd.to\_datetime(df["Timestamp"], format = "%Y-%m-%d") # Convert Timestamp to datetime format

# Get rows for year 2021, given station, make a new col for days

df\_2021 = df[(df["Timestamp"].dt.year == 2021) & (df["station"] == "Lal Bahadur Shastri Nagar, Kalaburagi - KSPCB")]

df\_2021 = df\_2021.dropna(subset = ["PM2.5"]) # Drop rows w PM2.5 missing

# Get month and day of the week

df\_2021["Month"] = df\_2021["Timestamp"].dt.month

df\_2021["Day\_of\_week"] = df\_2021["Timestamp"].dt.dayofweek # Monday = 0, Sunday = 6

conditions = [df\_2021["Day\_of\_week"] < 5, df\_2021["Day\_of\_week"] >= 5]

choices = ["Weekday", "Weekend"]

# Assign weekday or weekend

df\_2021["Type"] = np.select(conditions, choices, default = "Weekday")

# Group by month and type, then calc the avg PM2.5

monthly\_avg\_pm25 = df\_2021.groupby(["Month", "Type"])["PM2.5"].mean().unstack() # turn multi indexed series to dataframe, for easy plotting

# Plot

plt.figure(figsize=(10, 5))

plt.plot(monthly\_avg\_pm25.index, monthly\_avg\_pm25["Weekday"], marker="o", linestyle="-", label="Weekdays", color="blue")

plt.plot(monthly\_avg\_pm25.index, monthly\_avg\_pm25["Weekend"], marker="s", linestyle="--", label="Weekends", color="red")

plt.xlabel("Month")

plt.ylabel("Average PM2.5 Levels")

plt.title("Monthly Average PM2.5 Levels (2021)\nStation: Lal Bahadur Shastri Nagar, Kalaburagi - KSPCB")

plt.xticks(range(1, 13), ["Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"])

plt.legend()

plt.show()

34/102:

import pandas as pd

import numpy as np

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

df["Timestamp"] = pd.to\_datetime(df["Timestamp"], format = "%Y-%m-%d") # Convert Timestamp to datetime format

# Get rows for year 2018

df\_2018 = df[df["Timestamp"].dt.year == 2018]

# Mask for the given stations

df\_stations = df\_2018[df\_2018["station"] == "Lal Bahadur Shastri Nagar, Kalaburagi - KSPCB"]

df\_stations = df\_stations.dropna(subset = ["PM2.5"]) # Drop rows w PM2.5 missing

# Define seasons

get\_season = [

(df\_stations["Timestamp"].dt.month.isin([12, 1, 2])), # winter

(df\_stations["Timestamp"].dt.month.isin([3, 4, 5])), # Summer

(df\_stations["Timestamp"].dt.month.isin([6, 7, 8, 9])) # monsoon

]

# Corresponding seasons for each condition

seasons = ["Winter", "Summer", "Monsoon"]

# Assign a season to each row in df\_stations w np.select()

df\_stations["Season"] = np.select(get\_season, seasons, default="Other")

# Calculate average PM2.5 for each season

avg\_pm25\_per\_season = df\_stations.groupby("Season")["PM2.5"].mean()

# Get season w the highest pollution

most\_polluted\_season = avg\_pm25\_per\_season.idxmax()

highest\_avg\_pm25 = avg\_pm25\_per\_season.max()

print(f"The average PM2.5 concentration for each season for stations Lal Bahadur Shastri Nagar, Kalaburagi - KSPCB is:\n{avg\_pm25\_per\_season}")

print(f"\nThe most polluted season in 2018 was {most\_polluted\_season} with an average PM2.5 concentration of {highest\_avg\_pm25:.3f}")

34/103:

import pandas as pd

import numpy as np

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

# Get rows for year 2022

df["Timestamp"] = pd.to\_datetime(df["Timestamp"], format="%Y-%m-%d") # Convert Timestamp col to datetime format

# Get data for 2022

df\_2022 = df[df["Timestamp"].dt.year == 2022].copy()

# Get Month and define seasons

df\_2022["Month"] = df\_2022["Timestamp"].dt.month

df\_2022["Season"] = np.select(

[df\_2022["Month"].between(3, 5), df\_2022["Month"].between(6, 9)],

["Summer", "Monsoon"],

default="Other"

)

# Keep only summer and monsoon data

df\_2022 = df\_2022[df\_2022["Season"].isin(["Summer", "Monsoon"])]

# Group by State and Season, then calculate average PM2.5

seasonal\_avg\_pm25 = df\_2022.groupby(["State", "Season"])["PM2.5"].mean().unstack()

# Compute percentage change

seasonal\_avg\_pm25["Percentage Change"] = ((seasonal\_avg\_pm25["Monsoon"] - seasonal\_avg\_pm25["Summer"]) / seasonal\_avg\_pm25["Summer"]) \* 100

# Find the state with the highest absolute percentage change

most\_changed\_state = seasonal\_avg\_pm25["Percentage Change"].abs().idxmax()

max\_change\_value = seasonal\_avg\_pm25.loc[most\_changed\_state, "Percentage Change"]

# Display results

print(seasonal\_avg\_pm25)

print(f"State with the most difference: {most\_changed\_state} ({max\_change\_value:.2f}%)")

34/104:

import pandas as pd

import numpy as np

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

# Get rows for year 2022

df["Timestamp"] = pd.to\_datetime(df["Timestamp"], format="%Y-%m-%d") # Convert Timestamp col to datetime format

# Get data for 2022

df\_2022 = df[df["Timestamp"].dt.year == 2022].copy()

# Get Month and define seasons

df\_2022["Month"] = df\_2022["Timestamp"].dt.month

df\_2022["Season"] = np.select(

[df\_2022["Month"].between(3, 5), df\_2022["Month"].between(6, 9)],

["Summer", "Monsoon"],

default="Other"

)

# Keep only summer and monsoon data

df\_2022 = df\_2022[df\_2022["Season"].isin(["Summer", "Monsoon"])]

# Group by State and Season, then calculate average PM2.5

seasonal\_avg\_pm25 = df\_2022.groupby(["state", "Season"])["PM2.5"].mean().unstack()

# Compute percentage change

seasonal\_avg\_pm25["Percentage Change"] = ((seasonal\_avg\_pm25["Monsoon"] - seasonal\_avg\_pm25["Summer"]) / seasonal\_avg\_pm25["Summer"]) \* 100

# Find the state with the highest absolute percentage change

most\_changed\_state = seasonal\_avg\_pm25["Percentage Change"].abs().idxmax()

max\_change\_value = seasonal\_avg\_pm25.loc[most\_changed\_state, "Percentage Change"]

# Display results

print(seasonal\_avg\_pm25)

print(f"State with the most difference: {most\_changed\_state} ({max\_change\_value:.2f}%)")

34/105:

df = pd.read\_csv(r""C:\Users\inees\Downloads\State\_data.csv")

df.head()

34/106:

df = pd.read\_csv(r"C:\Users\inees\Downloads\State\_data.csv")

df.head()

34/107:

df = pd.read\_csv(r"C:\Users\inees\Downloads\NCAP\_Funding.csv")

df.head()

34/108:

import pandas as pd

import numpy as np

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

# Get rows for year 2022

df["Timestamp"] = pd.to\_datetime(df["Timestamp"], format="%Y-%m-%d") # Convert Timestamp col to datetime format

# Get data for 2022

df\_2022 = df[df["Timestamp"].dt.year == 2022].copy()

# Get month and seasons

df\_2022["Month"] = df\_2022["Timestamp"].dt.month

summer\_months = [3, 4, 5]

monsoon\_months = [6, 7, 8, 9]

df\_summer = df\_2022[df\_2022['Month'].isin(summer\_months)]

df\_monsoon = df\_2022[df\_2022['Month'].isin(monsoon\_months)]

# Calculate the avg PM2.5 levels for summer, monsoon

summer\_avg = df\_summer['PM2.5'].mean()

monsoon\_avg = df\_monsoon['PM2.5'].mean()

# Calculate the percentage change of pm2.5 levels of monsoon wrt summer

percentage\_change = ((monsoon\_avg - summer\_avg) / summer\_avg) \* 100

print(f"Percentage Change from Summer to Monsoon: {percentage\_change}%")

# state with the highest percentage change

max\_change\_state = percentage\_change.idxmax()

print(f"State with the highest PM2.5 change: {max\_change\_state}")

34/109:

import pandas as pd

import numpy as np

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

# Get rows for year 2022

df["Timestamp"] = pd.to\_datetime(df["Timestamp"], format="%Y-%m-%d") # Convert Timestamp col to datetime format

# Get data for 2022

df\_2022 = df[df["Timestamp"].dt.year == 2022].copy()

# Get month and seasons

df\_2022["Month"] = df\_2022["Timestamp"].dt.month

summer\_months = [3, 4, 5]

monsoon\_months = [6, 7, 8, 9]

df\_summer = df\_2022[df\_2022['Month'].isin(summer\_months)]

df\_monsoon = df\_2022[df\_2022['Month'].isin(monsoon\_months)]

# Calculate the avg PM2.5 levels for summer, monsoon

summer\_avg = df\_summer.groupby('State')['PM2.5'].mean()

monsoon\_avg = df\_monsoon.groupby('State')['PM2.5'].mean()

# Calculate the percentage change of pm2.5 levels of monsoon wrt summer for each state

percentage\_change = ((monsoon\_avg - summer\_avg) / summer\_avg) \* 100

print(f"Percentage Change from Summer to Monsoon: {percentage\_change}%")

# state with the highest percentage change

max\_change\_state = percentage\_change.idxmax()

print(f"State with the highest PM2.5 change: {max\_change\_state}")

34/110:

import pandas as pd

import numpy as np

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

# Get rows for year 2022

df["Timestamp"] = pd.to\_datetime(df["Timestamp"], format="%Y-%m-%d") # Convert Timestamp col to datetime format

# Get data for 2022

df\_2022 = df[df["Timestamp"].dt.year == 2022].copy()

# Get month and seasons

df\_2022["Month"] = df\_2022["Timestamp"].dt.month

summer\_months = [3, 4, 5]

monsoon\_months = [6, 7, 8, 9]

df\_summer = df\_2022[df\_2022['Month'].isin(summer\_months)]

df\_monsoon = df\_2022[df\_2022['Month'].isin(monsoon\_months)]

# Calculate the avg PM2.5 levels for summer, monsoon

summer\_avg = df\_summer.groupby('state')['PM2.5'].mean()

monsoon\_avg = df\_monsoon.groupby('state')['PM2.5'].mean()

# Calculate the percentage change of pm2.5 levels of monsoon wrt summer for each state

percentage\_change = ((monsoon\_avg - summer\_avg) / summer\_avg) \* 100

print(f"Percentage Change from Summer to Monsoon: {percentage\_change}%")

# state with the highest percentage change

max\_change\_state = percentage\_change.idxmax()

print(f"State with the highest PM2.5 change: {max\_change\_state}")

34/111:

import pandas as pd

import numpy as np

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

# Get rows for year 2022

df["Timestamp"] = pd.to\_datetime(df["Timestamp"], format="%Y-%m-%d") # Convert Timestamp col to datetime format

# Get data for 2022

df\_2022 = df[df["Timestamp"].dt.year == 2022].copy()

# Get month and seasons

df\_2022["Month"] = df\_2022["Timestamp"].dt.month

summer\_months = [3, 4, 5]

monsoon\_months = [6, 7, 8, 9]

df\_summer = df\_2022[df\_2022['Month'].isin(summer\_months)]

df\_monsoon = df\_2022[df\_2022['Month'].isin(monsoon\_months)]

# Calculate the avg PM2.5 levels for summer, monsoon

summer\_avg = df\_summer.groupby('state')['PM2.5'].mean()

monsoon\_avg = df\_monsoon.groupby('state')['PM2.5'].mean()

# Calculate the percentage change of pm2.5 levels of monsoon wrt summer for each state

percentage\_change = ((monsoon\_avg - summer\_avg) / summer\_avg) \* 100

print(f"Percentage Change from Summer to Monsoon: \n{percentage\_change}%")

# state with the highest percentage change

max\_change\_state = percentage\_change.idxmax()

print(f"State with the highest PM2.5 change: {max\_change\_state}")

34/112:

import pandas as pd

import numpy as np

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

# Get rows for year 2022

df["Timestamp"] = pd.to\_datetime(df["Timestamp"], format="%Y-%m-%d") # Convert Timestamp col to datetime format

# Get data for 2022

df\_2022 = df[df["Timestamp"].dt.year == 2022].copy()

# Get month and seasons

df\_2022["Month"] = df\_2022["Timestamp"].dt.month

summer\_months = [3, 4, 5]

monsoon\_months = [6, 7, 8, 9]

df\_summer = df\_2022[df\_2022['Month'].isin(summer\_months)]

df\_monsoon = df\_2022[df\_2022['Month'].isin(monsoon\_months)]

# Calculate the avg PM2.5 levels for summer, monsoon

summer\_avg = df\_summer.groupby('state')['PM2.5'].mean()

monsoon\_avg = df\_monsoon.groupby('state')['PM2.5'].mean()

# Calculate the percentage change of pm2.5 levels of monsoon wrt summer for each state

percentage\_change = ((monsoon\_avg - summer\_avg) / summer\_avg) \* 100

print(f"Percentage Change from Summer to Monsoon: \n{percentage\_change}")

# state with the highest percentage change

max\_change\_state = percentage\_change.idxmax()

print(f"State with the highest PM2.5 change: {max\_change\_state}")

34/113:

import pandas as pd

import numpy as np

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

# Get rows for year 2022

df["Timestamp"] = pd.to\_datetime(df["Timestamp"], format="%Y-%m-%d") # Convert Timestamp col to datetime format

# Get data for 2022

df\_2022 = df[df["Timestamp"].dt.year == 2022].copy()

# Get month and seasons

df\_2022["Month"] = df\_2022["Timestamp"].dt.month

summer\_months = [3, 4, 5]

monsoon\_months = [6, 7, 8, 9]

df\_summer = df\_2022[df\_2022['Month'].isin(summer\_months)]

df\_monsoon = df\_2022[df\_2022['Month'].isin(monsoon\_months)]

# Calculate the avg PM2.5 levels for summer, monsoon

summer\_avg = df\_summer.groupby('state')['PM2.5'].mean()

monsoon\_avg = df\_monsoon.groupby('state')['PM2.5'].mean()

# Calculate the percentage change of pm2.5 levels of monsoon wrt summer for each state

percentage\_change = ((monsoon\_avg - summer\_avg) / summer\_avg) \* 100

print(f"Percentage Change from Summer to Monsoon: \n{percentage\_change}")

# state with the highest percentage change

max\_change\_state = percentage\_change.idxmax().abs()

print(f"State with the highest PM2.5 change: {max\_change\_state}")

34/114:

import pandas as pd

import numpy as np

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

# Get rows for year 2022

df["Timestamp"] = pd.to\_datetime(df["Timestamp"], format="%Y-%m-%d") # Convert Timestamp col to datetime format

# Get data for 2022

df\_2022 = df[df["Timestamp"].dt.year == 2022].copy()

# Get month and seasons

df\_2022["Month"] = df\_2022["Timestamp"].dt.month

summer\_months = [3, 4, 5]

monsoon\_months = [6, 7, 8, 9]

df\_summer = df\_2022[df\_2022['Month'].isin(summer\_months)]

df\_monsoon = df\_2022[df\_2022['Month'].isin(monsoon\_months)]

# Calculate the avg PM2.5 levels for summer, monsoon

summer\_avg = df\_summer.groupby('state')['PM2.5'].mean()

monsoon\_avg = df\_monsoon.groupby('state')['PM2.5'].mean()

# Calculate the percentage change of pm2.5 levels of monsoon wrt summer for each state

percentage\_change = ((monsoon\_avg - summer\_avg) / summer\_avg) \* 100

print(f"Percentage Change from Summer to Monsoon: \n{percentage\_change}")

# state with the highest percentage change

max\_change\_state = percentage\_change.abs().idxmax()

print(f"State with the highest PM2.5 change: {max\_change\_state}")

34/115:

import pandas as pd

import numpy as np

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

# Get rows for year 2022

df["Timestamp"] = pd.to\_datetime(df["Timestamp"], format="%Y-%m-%d") # Convert Timestamp col to datetime format

# Get data for 2022

df\_2022 = df[df["Timestamp"].dt.year == 2022].copy()

# Get month and seasons

df\_2022["Month"] = df\_2022["Timestamp"].dt.month

summer\_months = [3, 4, 5]

monsoon\_months = [6, 7, 8, 9]

df\_summer = df\_2022[df\_2022['Month'].isin(summer\_months)]

df\_monsoon = df\_2022[df\_2022['Month'].isin(monsoon\_months)]

# Calculate the avg PM2.5 levels for summer, monsoon

summer\_avg = df\_summer.groupby('state')['PM2.5'].mean()

monsoon\_avg = df\_monsoon.groupby('state')['PM2.5'].mean()

# Calculate the percentage change of pm2.5 levels of monsoon wrt summer for each state

percentage\_change = ((monsoon\_avg - summer\_avg) / summer\_avg) \* 100

print(f"Percentage Change from Summer to Monsoon: \n{percentage\_change}")

# state with the highest percentage change

max\_change\_state = percentage\_change.abs().idxmax()

max\_change\_value = percentage\_change.abs().max()

print(f"State with the highest PM2.5 change: {max\_change\_state} with a percentage decrease of {max\_change\_value} in the monsoon.")

34/116:

import pandas as pd

import numpy as np

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

# Get rows for year 2022

df["Timestamp"] = pd.to\_datetime(df["Timestamp"], format="%Y-%m-%d") # Convert Timestamp col to datetime format

# Get data for 2022

df\_2022 = df[df["Timestamp"].dt.year == 2022].copy()

# Get month and seasons

df\_2022["Month"] = df\_2022["Timestamp"].dt.month

summer\_months = [3, 4, 5]

monsoon\_months = [6, 7, 8, 9]

df\_summer = df\_2022[df\_2022['Month'].isin(summer\_months)]

df\_monsoon = df\_2022[df\_2022['Month'].isin(monsoon\_months)]

# Calculate the avg PM2.5 levels for summer, monsoon

summer\_avg = df\_summer.groupby('state')['PM2.5'].mean()

monsoon\_avg = df\_monsoon.groupby('state')['PM2.5'].mean()

# Calculate the percentage change of pm2.5 levels of monsoon wrt summer for each state

percentage\_change = ((monsoon\_avg - summer\_avg) / summer\_avg) \* 100

print(f"Percentage Change from Summer to Monsoon: \n{percentage\_change}")

# state with the highest percentage change

max\_change\_state = percentage\_change.abs().idxmax()

max\_change\_value = percentage\_change.abs().max()

print(f"State with the highest PM2.5 change: {max\_change\_state} with a percentage decrease of {max\_change\_value: .3f}% in the monsoon.")

34/117:

import pandas as pd

import numpy as np

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

# Get rows for year 2022

df["Timestamp"] = pd.to\_datetime(df["Timestamp"], format="%Y-%m-%d") # Convert Timestamp col to datetime format

# Get data for 2022

df\_2022 = df[df["Timestamp"].dt.year == 2022].copy()

# Get month and seasons

df\_2022["Month"] = df\_2022["Timestamp"].dt.month

summer\_months = [3, 4, 5]

monsoon\_months = [6, 7, 8, 9]

df\_summer = df\_2022[df\_2022['Month'].isin(summer\_months)]

df\_monsoon = df\_2022[df\_2022['Month'].isin(monsoon\_months)]

# Calculate the avg PM2.5 levels for summer, monsoon

summer\_avg = df\_summer.groupby('state')['PM2.5'].mean()

monsoon\_avg = df\_monsoon.groupby('state')['PM2.5'].mean()

# Calculate the percentage change of pm2.5 levels of monsoon wrt summer for each state

percentage\_change = ((monsoon\_avg - summer\_avg) / summer\_avg) \* 100

print(f"Percentage Change from Summer to Monsoon: \n{percentage\_change}")

# state with the highest percentage change

max\_change\_state = percentage\_change.abs().idxmax()

max\_change\_value = percentage\_change.abs().max()

print(f"State with the highest PM2.5 change: {max\_change\_state} with a percentage decrease of{max\_change\_value: .3f}% in the monsoon.")

34/118:

import pandas as pd

import numpy as np

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

# Get rows for year 2022

df["Timestamp"] = pd.to\_datetime(df["Timestamp"], format="%Y-%m-%d") # Convert Timestamp col to datetime format

# Get data for 2022

df\_2022 = df[df["Timestamp"].dt.year == 2022].copy()

# Get month and seasons

df\_2022["Month"] = df\_2022["Timestamp"].dt.month

summer\_months = [3, 4, 5]

monsoon\_months = [6, 7, 8, 9]

df\_summer = df\_2022[df\_2022['Month'].isin(summer\_months)]

df\_monsoon = df\_2022[df\_2022['Month'].isin(monsoon\_months)]

# Calculate the avg PM2.5 levels for summer, monsoon

summer\_avg = df\_summer.groupby('state')['PM2.5'].mean()

monsoon\_avg = df\_monsoon.groupby('state')['PM2.5'].mean()

# Calculate the percentage change of pm2.5 levels of monsoon wrt summer for each state

percentage\_change = ((monsoon\_avg - summer\_avg) / summer\_avg) \* 100

print(f"Percentage Change from Summer to Monsoon: \n{percentage\_change}")

# state with the highest percentage change

max\_change\_state = percentage\_change.abs().idxmax()

max\_change\_value = percentage\_change.abs().max()

print(f"\nState with the highest PM2.5 change: {max\_change\_state} with a percentage decrease of{max\_change\_value: .3f}% in the monsoon.")

34/119:

import pandas as pd

import matplotlib.pyplot as plt

# Assuming the data is already loaded into the 'df' DataFrame

# Filter for Delhi data

df\_delhi = df[df['City'] == 'Delhi']

# Extract year and month

df\_delhi['Year'] = pd.to\_datetime(df\_delhi['Timestamp'], format='%d-%m-%Y').dt.year

df\_delhi['Month'] = pd.to\_datetime(df\_delhi['Timestamp'], format='%d-%m-%Y').dt.month

# Define seasons

def get\_season(month):

if month in [12, 1, 2]:

return 'Winter'

elif month in [3, 4, 5]:

return 'Summer'

elif month in [6, 7, 8, 9]:

return 'Monsoon'

else:

return 'Other'

df\_delhi['Season'] = df\_delhi['Month'].apply(get\_season)

# Filter data for the years 2017 to 2023

df\_delhi = df\_delhi[(df\_delhi['Year'] >= 2017) & (df\_delhi['Year'] <= 2023)]

# Calculate the average PM2.5 levels for each season across the years

seasonal\_avg\_pm25 = df\_delhi.groupby(['Year', 'Season'])['PM2.5'].mean().unstack()

# Plot the seasonal variation

plt.figure(figsize=(10, 6))

seasonal\_avg\_pm25.plot(kind='line', marker='o', figsize=(12, 6))

plt.title("Average PM2.5 Levels in Delhi (2017-2023) by Season")

plt.xlabel("Year")

plt.ylabel("Average PM2.5 (µg/m³)")

plt.legend(title="Season", loc='upper left')

plt.grid(True)

plt.xticks(rotation=45)

plt.tight\_layout()

plt.show()

34/120:

import pandas as pd

import matplotlib.pyplot as plt

# Load the CPCB Air Pollution Data (Data.csv)

df = pd.read\_csv('CPCB\_Air\_Pollution\_Data.csv')

# Ensure 'Timestamp' is in datetime format

df['Timestamp'] = pd.to\_datetime(df['Timestamp'], format='%d-%m-%Y')

# Filter for Delhi data

df\_delhi = df[df['City'] == 'Delhi']

# Extract year and month

df\_delhi['Year'] = pd.to\_datetime(df\_delhi['Timestamp'], format='%d-%m-%Y').dt.year

df\_delhi['Month'] = pd.to\_datetime(df\_delhi['Timestamp'], format='%d-%m-%Y').dt.month

# Define seasons

def get\_season(month):

if month in [12, 1, 2]:

return 'Winter'

elif month in [3, 4, 5]:

return 'Summer'

elif month in [6, 7, 8, 9]:

return 'Monsoon'

else:

return 'Other'

df\_delhi['Season'] = df\_delhi['Month'].apply(get\_season)

# Filter data for the years 2017 to 2023

df\_delhi = df\_delhi[(df\_delhi['Year'] >= 2017) & (df\_delhi['Year'] <= 2023)]

# Calculate the average PM2.5 levels for each season across the years

seasonal\_avg\_pm25 = df\_delhi.groupby(['Year', 'Season'])['PM2.5'].mean().unstack()

# Plot the seasonal variation

plt.figure(figsize=(10, 6))

seasonal\_avg\_pm25.plot(kind='line', marker='o', figsize=(12, 6))

plt.title("Average PM2.5 Levels in Delhi (2017-2023) by Season")

plt.xlabel("Year")

plt.ylabel("Average PM2.5 (µg/m³)")

plt.legend(title="Season", loc='upper left')

plt.grid(True)

plt.xticks(rotation=45)

plt.tight\_layout()

plt.show()

34/121:

import pandas as pd

import matplotlib.pyplot as plt

# Load the CPCB Air Pollution Data (Data.csv)

df = pd.read\_csv(r"C:\Users\inees\Downloads\NCAP\_Funding.csv")

# Ensure 'Timestamp' is in datetime format

df['Timestamp'] = pd.to\_datetime(df['Timestamp'], format='%d-%m-%Y')

# Filter for Delhi data

df\_delhi = df[df['City'] == 'Delhi']

# Extract year and month

df\_delhi['Year'] = pd.to\_datetime(df\_delhi['Timestamp'], format='%d-%m-%Y').dt.year

df\_delhi['Month'] = pd.to\_datetime(df\_delhi['Timestamp'], format='%d-%m-%Y').dt.month

# Define seasons

def get\_season(month):

if month in [12, 1, 2]:

return 'Winter'

elif month in [3, 4, 5]:

return 'Summer'

elif month in [6, 7, 8, 9]:

return 'Monsoon'

else:

return 'Other'

df\_delhi['Season'] = df\_delhi['Month'].apply(get\_season)

# Filter data for the years 2017 to 2023

df\_delhi = df\_delhi[(df\_delhi['Year'] >= 2017) & (df\_delhi['Year'] <= 2023)]

# Calculate the average PM2.5 levels for each season across the years

seasonal\_avg\_pm25 = df\_delhi.groupby(['Year', 'Season'])['PM2.5'].mean().unstack()

# Plot the seasonal variation

plt.figure(figsize=(10, 6))

seasonal\_avg\_pm25.plot(kind='line', marker='o', figsize=(12, 6))

plt.title("Average PM2.5 Levels in Delhi (2017-2023) by Season")

plt.xlabel("Year")

plt.ylabel("Average PM2.5 (µg/m³)")

plt.legend(title="Season", loc='upper left')

plt.grid(True)

plt.xticks(rotation=45)

plt.tight\_layout()

plt.show()

34/122:

import pandas as pd

import matplotlib.pyplot as plt

# Load the CPCB Air Pollution Data (Data.csv)

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

# Ensure 'Timestamp' is in datetime format

df['Timestamp'] = pd.to\_datetime(df['Timestamp'], format='%d-%m-%Y')

# Filter for Delhi data

df\_delhi = df[df['city'] == 'Delhi']

# Extract year and month

df\_delhi['Year'] = pd.to\_datetime(df\_delhi['Timestamp'], format='%d-%m-%Y').dt.year

df\_delhi['Month'] = pd.to\_datetime(df\_delhi['Timestamp'], format='%d-%m-%Y').dt.month

# Define seasons

def get\_season(month):

if month in [12, 1, 2]:

return 'Winter'

elif month in [3, 4, 5]:

return 'Summer'

elif month in [6, 7, 8, 9]:

return 'Monsoon'

else:

return 'Other'

df\_delhi['Season'] = df\_delhi['Month'].apply(get\_season)

# Filter data for the years 2017 to 2023

df\_delhi = df\_delhi[(df\_delhi['Year'] >= 2017) & (df\_delhi['Year'] <= 2023)]

# Calculate the average PM2.5 levels for each season across the years

seasonal\_avg\_pm25 = df\_delhi.groupby(['Year', 'Season'])['PM2.5'].mean().unstack()

# Plot the seasonal variation

plt.figure(figsize=(10, 6))

seasonal\_avg\_pm25.plot(kind='line', marker='o', figsize=(12, 6))

plt.title("Average PM2.5 Levels in Delhi (2017-2023) by Season")

plt.xlabel("Year")

plt.ylabel("Average PM2.5 (µg/m³)")

plt.legend(title="Season", loc='upper left')

plt.grid(True)

plt.xticks(rotation=45)

plt.tight\_layout()

plt.show()

34/123:

import pandas as pd

import matplotlib.pyplot as plt

# Load the CPCB Air Pollution Data (Data.csv)

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

# Ensure 'Timestamp' is in datetime format

df['Timestamp'] = pd.to\_datetime(df['Timestamp'], format='%Y-%m-%d')

# Filter for Delhi data

df\_delhi = df[df['city'] == 'Delhi']

# Extract year and month

df\_delhi['Year'] = pd.to\_datetime(df\_delhi['Timestamp'], format='%d-%m-%Y').dt.year

df\_delhi['Month'] = pd.to\_datetime(df\_delhi['Timestamp'], format='%d-%m-%Y').dt.month

# Define seasons

def get\_season(month):

if month in [12, 1, 2]:

return 'Winter'

elif month in [3, 4, 5]:

return 'Summer'

elif month in [6, 7, 8, 9]:

return 'Monsoon'

else:

return 'Other'

df\_delhi['Season'] = df\_delhi['Month'].apply(get\_season)

# Filter data for the years 2017 to 2023

df\_delhi = df\_delhi[(df\_delhi['Year'] >= 2017) & (df\_delhi['Year'] <= 2023)]

# Calculate the average PM2.5 levels for each season across the years

seasonal\_avg\_pm25 = df\_delhi.groupby(['Year', 'Season'])['PM2.5'].mean().unstack()

# Plot the seasonal variation

plt.figure(figsize=(10, 6))

seasonal\_avg\_pm25.plot(kind='line', marker='o', figsize=(12, 6))

plt.title("Average PM2.5 Levels in Delhi (2017-2023) by Season")

plt.xlabel("Year")

plt.ylabel("Average PM2.5 (µg/m³)")

plt.legend(title="Season", loc='upper left')

plt.grid(True)

plt.xticks(rotation=45)

plt.tight\_layout()

plt.show()

34/124:

import pandas as pd

import matplotlib.pyplot as plt

# Load the CPCB Air Pollution Data (Data.csv)

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

# Ensure 'Timestamp' is in datetime format

df['Timestamp'] = pd.to\_datetime(df['Timestamp'], format='%Y-%m-%d')

# Filter for Delhi data

df\_delhi = df[df['city'] == 'Delhi'].copy()

# Extract year and month

df\_delhi['Year'] = pd.to\_datetime(df\_delhi['Timestamp'], format='%d-%m-%Y').dt.year

df\_delhi['Month'] = pd.to\_datetime(df\_delhi['Timestamp'], format='%d-%m-%Y').dt.month

# Define seasons

def get\_season(month):

if month in [12, 1, 2]:

return 'Winter'

elif month in [3, 4, 5]:

return 'Summer'

elif month in [6, 7, 8, 9]:

return 'Monsoon'

else:

return 'Other'

df\_delhi['Season'] = df\_delhi['Month'].apply(get\_season)

# Filter data for the years 2017 to 2023

df\_delhi = df\_delhi[(df\_delhi['Year'] >= 2017) & (df\_delhi['Year'] <= 2023)]

# Calculate the average PM2.5 levels for each season across the years

seasonal\_avg\_pm25 = df\_delhi.groupby(['Year', 'Season'])['PM2.5'].mean().unstack()

# Plot the seasonal variation

plt.figure(figsize=(10, 6))

seasonal\_avg\_pm25.plot(kind='line', marker='o', figsize=(12, 6))

plt.title("Average PM2.5 Levels in Delhi (2017-2023) by Season")

plt.xlabel("Year")

plt.ylabel("Average PM2.5 (µg/m³)")

plt.legend(title="Season", loc='upper left')

plt.grid(True)

plt.xticks(rotation=45)

plt.tight\_layout()

plt.show()

34/125:

import pandas as pd

import matplotlib.pyplot as plt

# Load the CPCB Air Pollution Data (Data.csv)

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

# Ensure 'Timestamp' is in datetime format

df['Timestamp'] = pd.to\_datetime(df['Timestamp'], format='%Y-%m-%d')

# Filter data for Delhi and Mumbai

df\_delhi\_mumbai = df[df['City'].isin(['Delhi', 'Mumbai'])].copy()

# Extract year and month from the Timestamp

df\_delhi\_mumbai['Year'] = df\_delhi\_mumbai['Timestamp'].dt.year

df\_delhi\_mumbai['Month'] = df\_delhi\_mumbai['Timestamp'].dt.month

# Filter for years between 2017 and 2023

df\_delhi\_mumbai = df\_delhi\_mumbai[(df\_delhi\_mumbai['Year'] >= 2017) & (df\_delhi\_mumbai['Year'] <= 2023)]

# Calculate monthly average PM2.5 levels for both Delhi and Mumbai

city\_avg\_pm25 = df\_delhi\_mumbai.groupby(['Year', 'Month', 'City'])['PM2.5'].mean().unstack()

# Plot the time-series comparison

plt.figure(figsize=(12, 6))

city\_avg\_pm25['Delhi'].plot(label='Delhi', marker='o', linestyle='-', color='b')

city\_avg\_pm25['Mumbai'].plot(label='Mumbai', marker='o', linestyle='-', color='r')

# Add labels and title

plt.title("PM2.5 Levels in Delhi vs Mumbai (2017-2023)")

plt.xlabel("Year")

plt.ylabel("Average PM2.5 (µg/m³)")

plt.legend()

plt.grid(True)

plt.xticks(rotation=45)

plt.tight\_layout()

# Show the plot

plt.show()

34/126:

import pandas as pd

import matplotlib.pyplot as plt

# Load the CPCB Air Pollution Data (Data.csv)

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

# Ensure 'Timestamp' is in datetime format

df['Timestamp'] = pd.to\_datetime(df['Timestamp'], format='%Y-%m-%d')

# Filter for Delhi data

df\_delhi = df[df['city'] == 'Delhi'].copy()

# Extract year and month

df\_delhi['Year'] = pd.to\_datetime(df\_delhi['Timestamp'], format='%Y-%m-%d').dt.year

df\_delhi['Month'] = pd.to\_datetime(df\_delhi['Timestamp'], format='%Y-%m-%d').dt.month

# Define seasons

def get\_season(month):

if month in [12, 1, 2]:

return 'Winter'

elif month in [3, 4, 5]:

return 'Summer'

elif month in [6, 7, 8, 9]:

return 'Monsoon'

else:

return 'Other'

df\_delhi['Season'] = df\_delhi['Month'].apply(get\_season)

# Filter data for the years 2017 to 2023

df\_delhi = df\_delhi[(df\_delhi['Year'] >= 2017) & (df\_delhi['Year'] <= 2023)]

# Calculate the average PM2.5 levels for each season across the years

seasonal\_avg\_pm25 = df\_delhi.groupby(['Year', 'Season'])['PM2.5'].mean().unstack()

# Plot the seasonal variation

plt.figure(figsize=(10, 6))

seasonal\_avg\_pm25.plot(kind='line', marker='o', figsize=(12, 6))

plt.title("Average PM2.5 Levels in Delhi (2017-2023) by Season")

plt.xlabel("Year")

plt.ylabel("Average PM2.5 (µg/m³)")

plt.legend(title="Season", loc='upper left')

plt.grid(True)

plt.xticks(rotation=45)

plt.tight\_layout()

plt.show()

34/127:

import pandas as pd

import matplotlib.pyplot as plt

# Load the CPCB Air Pollution Data (Data.csv)

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

# Ensure 'Timestamp' is in datetime format

df['Timestamp'] = pd.to\_datetime(df['Timestamp'], format='%Y-%m-%d')

# Filter data for Delhi and Mumbai

df\_delhi\_mumbai = df[df['City'].isin(['Delhi', 'Mumbai'])].copy()

# Extract year and month from the Timestamp

df\_delhi\_mumbai['Year'] = df\_delhi\_mumbai['Timestamp'].dt.year

df\_delhi\_mumbai['Month'] = df\_delhi\_mumbai['Timestamp'].dt.month

# Filter for years between 2017 and 2023

df\_delhi\_mumbai = df\_delhi\_mumbai[(df\_delhi\_mumbai['Year'] >= 2017) & (df\_delhi\_mumbai['Year'] <= 2023)]

# Calculate monthly average PM2.5 levels for both Delhi and Mumbai

city\_avg\_pm25 = df\_delhi\_mumbai.groupby(['Year', 'Month', 'City'])['PM2.5'].mean().unstack()

# Plot the time-series comparison

plt.figure(figsize=(12, 6))

city\_avg\_pm25['Delhi'].plot(label='Delhi', marker='o', linestyle='-', color='b')

city\_avg\_pm25['Mumbai'].plot(label='Mumbai', marker='o', linestyle='-', color='r')

# Add labels and title

plt.title("PM2.5 Levels in Delhi vs Mumbai (2017-2023)")

plt.xlabel("Year")

plt.ylabel("Average PM2.5 (µg/m³)")

plt.legend()

plt.grid(True)

plt.xticks(rotation=45)

plt.tight\_layout()

# Show the plot

plt.show()

34/128:

import pandas as pd

import matplotlib.pyplot as plt

# Load the CPCB Air Pollution Data (Data.csv)

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

# Ensure 'Timestamp' is in datetime format

df['Timestamp'] = pd.to\_datetime(df['Timestamp'], format='%Y-%m-%d')

# Filter data for Delhi and Mumbai

df\_delhi\_mumbai = df[df['city'].isin(['Delhi', 'Mumbai'])].copy()

# Extract year and month from the Timestamp

df\_delhi\_mumbai['Year'] = df\_delhi\_mumbai['Timestamp'].dt.year

df\_delhi\_mumbai['Month'] = df\_delhi\_mumbai['Timestamp'].dt.month

# Filter for years between 2017 and 2023

df\_delhi\_mumbai = df\_delhi\_mumbai[(df\_delhi\_mumbai['Year'] >= 2017) & (df\_delhi\_mumbai['Year'] <= 2023)]

# Calculate monthly average PM2.5 levels for both Delhi and Mumbai

city\_avg\_pm25 = df\_delhi\_mumbai.groupby(['Year', 'Month', 'City'])['PM2.5'].mean().unstack()

# Plot the time-series comparison

plt.figure(figsize=(12, 6))

city\_avg\_pm25['Delhi'].plot(label='Delhi', marker='o', linestyle='-', color='b')

city\_avg\_pm25['Mumbai'].plot(label='Mumbai', marker='o', linestyle='-', color='r')

# Add labels and title

plt.title("PM2.5 Levels in Delhi vs Mumbai (2017-2023)")

plt.xlabel("Year")

plt.ylabel("Average PM2.5 (µg/m³)")

plt.legend()

plt.grid(True)

plt.xticks(rotation=45)

plt.tight\_layout()

# Show the plot

plt.show()

34/129:

import pandas as pd

import matplotlib.pyplot as plt

# Load the CPCB Air Pollution Data (Data.csv)

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

# Ensure 'Timestamp' is in datetime format

df['Timestamp'] = pd.to\_datetime(df['Timestamp'], format='%Y-%m-%d')

# Filter data for Delhi and Mumbai

df\_delhi\_mumbai = df[df['city'].isin(['Delhi', 'Mumbai'])].copy()

# Extract year and month from the Timestamp

df\_delhi\_mumbai['Year'] = df\_delhi\_mumbai['Timestamp'].dt.year

df\_delhi\_mumbai['Month'] = df\_delhi\_mumbai['Timestamp'].dt.month

# Filter for years between 2017 and 2023

df\_delhi\_mumbai = df\_delhi\_mumbai[(df\_delhi\_mumbai['Year'] >= 2017) & (df\_delhi\_mumbai['Year'] <= 2023)]

# Calculate monthly average PM2.5 levels for both Delhi and Mumbai

city\_avg\_pm25 = df\_delhi\_mumbai.groupby(['Year', 'Month', 'city'])['PM2.5'].mean().unstack()

# Plot the time-series comparison

plt.figure(figsize=(12, 6))

city\_avg\_pm25['Delhi'].plot(label='Delhi', marker='o', linestyle='-', color='b')

city\_avg\_pm25['Mumbai'].plot(label='Mumbai', marker='o', linestyle='-', color='r')

# Add labels and title

plt.title("PM2.5 Levels in Delhi vs Mumbai (2017-2023)")

plt.xlabel("Year")

plt.ylabel("Average PM2.5 (µg/m³)")

plt.legend()

plt.grid(True)

plt.xticks(rotation=45)

plt.tight\_layout()

# Show the plot

plt.show()

34/130:

import pandas as pd

import matplotlib.pyplot as plt

# Load the CPCB Air Pollution Data (Data.csv)

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

# Ensure 'Timestamp' is in datetime format

df['Timestamp'] = pd.to\_datetime(df['Timestamp'], format='%Y-%m-%d')

# Filter data for Delhi and Mumbai

df\_delhi\_mumbai = df[df['city'].isin(['Delhi', 'Mumbai'])].copy()

# Extract year and month from the Timestamp

df\_delhi\_mumbai['Year'] = df\_delhi\_mumbai['Timestamp'].dt.year

df\_delhi\_mumbai['Month'] = df\_delhi\_mumbai['Timestamp'].dt.month

# Filter for years between 2017 and 2023

df\_delhi\_mumbai = df\_delhi\_mumbai[(df\_delhi\_mumbai['Year'] >= 2017) & (df\_delhi\_mumbai['Year'] <= 2023)]

# Calculate monthly average PM2.5 levels for both Delhi and Mumbai

city\_avg\_pm25 = df\_delhi\_mumbai.groupby(['Year', 'Month', 'city'])['PM2.5'].mean().unstack()

# Plot the time-series comparison

plt.figure(figsize=(12, 6))

city\_avg\_pm25['Delhi'].plot(label='Delhi', marker='.', linestyle='-', color='b')

city\_avg\_pm25['Mumbai'].plot(label='Mumbai', marker='o', linestyle='-', color='r')

# Add labels and title

plt.title("PM2.5 Levels in Delhi vs Mumbai (2017-2023)")

plt.xlabel("Year")

plt.ylabel("Average PM2.5 (µg/m³)")

plt.legend()

plt.grid(True)

plt.xticks(rotation=45)

plt.tight\_layout()

# Show the plot

plt.show()

34/131:

import pandas as pd

import matplotlib.pyplot as plt

# Load the CPCB Air Pollution Data (Data.csv)

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

# Ensure 'Timestamp' is in datetime format

df['Timestamp'] = pd.to\_datetime(df['Timestamp'], format='%Y-%m-%d')

# Filter data for Delhi and Mumbai

df\_delhi\_mumbai = df[df['city'].isin(['Delhi', 'Mumbai'])].copy()

# Extract year and month from the Timestamp

df\_delhi\_mumbai['Year'] = df\_delhi\_mumbai['Timestamp'].dt.year

df\_delhi\_mumbai['Month'] = df\_delhi\_mumbai['Timestamp'].dt.month

# Filter for years between 2017 and 2023

df\_delhi\_mumbai = df\_delhi\_mumbai[(df\_delhi\_mumbai['Year'] >= 2017) & (df\_delhi\_mumbai['Year'] <= 2023)]

# Calculate monthly average PM2.5 levels for both Delhi and Mumbai

city\_avg\_pm25 = df\_delhi\_mumbai.groupby(['Year', 'Month', 'city'])['PM2.5'].mean().unstack()

# Plot the time-series comparison

plt.figure(figsize=(12, 6))

city\_avg\_pm25['Delhi'].plot(label='Delhi', marker='s', linestyle='-', color='b')

city\_avg\_pm25['Mumbai'].plot(label='Mumbai', marker='o', linestyle='-', color='r')

# Add labels and title

plt.title("PM2.5 Levels in Delhi vs Mumbai (2017-2023)")

plt.xlabel("Year")

plt.ylabel("Average PM2.5 (µg/m³)")

plt.legend()

plt.grid(True)

plt.xticks(rotation=45)

plt.tight\_layout()

# Show the plot

plt.show()

34/132:

import pandas as pd

# Load the datasets

df\_pollution = pd.read\_csv('Data.csv') # Air quality monitoring data

df\_population = pd.read\_csv('State\_data.csv') # State population data

# Step 1: Count the number of monitoring stations per state

stations\_per\_state = df\_pollution.groupby('State')['Station'].nunique().reset\_index()

stations\_per\_state.columns = ['State', 'Num\_Stations']

# Step 2: Merge with state population data

stations\_population = pd.merge(stations\_per\_state, df\_population[['State', 'Population']], on='State', how='left')

# Step 3: Calculate the number of stations per million people

stations\_population['Stations\_per\_Million'] = (stations\_population['Num\_Stations'] / stations\_population['Population']) \* 1\_000\_000

# Step 4: Find the state with the highest ratio

max\_stations\_state = stations\_population.loc[stations\_population['Stations\_per\_Million'].idxmax()]

print(f"The state with the highest number of monitoring stations relative to its population is {max\_stations\_state['State']}, with {max\_stations\_state['Stations\_per\_Million']:.2f} stations per

34/133:

import pandas as pd

# Load the datasets

df\_pollution = pd.read\_csv('Data.csv') # Air quality monitoring data

df\_population = pd.read\_csv('State\_data.csv') # State population data

# Step 1: Count the number of monitoring stations per state

stations\_per\_state = df\_pollution.groupby('State')['Station'].nunique().reset\_index()

stations\_per\_state.columns = ['State', 'Num\_Stations']

# Step 2: Merge with state population data

stations\_population = pd.merge(stations\_per\_state, df\_population[['State', 'Population']], on='State', how='left')

# Step 3: Calculate the number of stations per million people

stations\_population['Stations\_per\_Million'] = (stations\_population['Num\_Stations'] / stations\_population['Population']) \* 1\_000\_000

# Step 4: Find the state with the highest ratio

max\_stations\_state = stations\_population.loc[stations\_population['Stations\_per\_Million'].idxmax()]

print(f"The state with the highest number of monitoring stations relative to its population is {max\_stations\_state['State']}, with {max\_stations\_state['Stations\_per\_Million']:.2f} stations per million people.")

34/134:

import pandas as pd

# Load the datasets

df\_pollution = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv") # Air quality monitoring data

df\_population = pd.read\_csv(r"C:\Users\inees\Downloads\State\_data.csv") # State population data

# Step 1: Count the number of monitoring stations per state

stations\_per\_state = df\_pollution.groupby('State')['Station'].nunique().reset\_index()

stations\_per\_state.columns = ['State', 'Num\_Stations']

# Step 2: Merge with state population data

stations\_population = pd.merge(stations\_per\_state, df\_population[['State', 'Population']], on='State', how='left')

# Step 3: Calculate the number of stations per million people

stations\_population['Stations\_per\_Million'] = (stations\_population['Num\_Stations'] / stations\_population['Population']) \* 1\_000\_000

# Step 4: Find the state with the highest ratio

max\_stations\_state = stations\_population.loc[stations\_population['Stations\_per\_Million'].idxmax()]

print(f"The state with the highest number of monitoring stations relative to its population is {max\_stations\_state['State']}, with {max\_stations\_state['Stations\_per\_Million']:.2f} stations per million people.")

34/135:

import pandas as pd

# Load the datasets

df\_pollution = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv") # Air quality monitoring data

df\_population = pd.read\_csv(r"C:\Users\inees\Downloads\State\_data.csv") # State population data

# Step 1: Count the number of monitoring stations per state

stations\_per\_state = df\_pollution.groupby('state')['station'].nunique().reset\_index()

stations\_per\_state.columns = ['state', 'Num\_Stations']

# Step 2: Merge with state population data

stations\_population = pd.merge(stations\_per\_state, df\_population[['State', 'Population']], on='State', how='left')

# Step 3: Calculate the number of stations per million people

stations\_population['Stations\_per\_Million'] = (stations\_population['Num\_Stations'] / stations\_population['Population']) \* 1\_000\_000

# Step 4: Find the state with the highest ratio

max\_stations\_state = stations\_population.loc[stations\_population['Stations\_per\_Million'].idxmax()]

print(f"The state with the highest number of monitoring stations relative to its population is {max\_stations\_state['State']}, with {max\_stations\_state['Stations\_per\_Million']:.2f} stations per million people.")

34/136:

import pandas as pd

# Load the datasets

df\_pollution = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv") # Air quality monitoring data

df\_population = pd.read\_csv(r"C:\Users\inees\Downloads\State\_data.csv") # State population data

# Step 1: Count the number of monitoring stations per state

stations\_per\_state = df\_pollution.groupby('state')['station'].nunique().reset\_index()

stations\_per\_state.columns = ['state', 'Num\_Stations']

# Step 2: Merge with state population data

stations\_population = pd.merge(stations\_per\_state, df\_population[['State', 'Population']], on='State', how='left')

# Step 3: Calculate the number of stations per million people

stations\_population['Stations\_per\_Million'] = (stations\_population['Num\_Stations'] / stations\_population['population']) \* 1\_000\_000

# Step 4: Find the state with the highest ratio

max\_stations\_state = stations\_population.loc[stations\_population['Stations\_per\_Million'].idxmax()]

print(f"The state with the highest number of monitoring stations relative to its population is {max\_stations\_state['State']}, with {max\_stations\_state['Stations\_per\_Million']:.2f} stations per million people.")

34/137:

import pandas as pd

# Load the datasets

df\_pollution = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv") # Air quality monitoring data

df\_pollution["State"] = df\_pollution["state"]

df\_population = pd.read\_csv(r"C:\Users\inees\Downloads\State\_data.csv") # State population data

# Step 1: Count the number of monitoring stations per state

stations\_per\_state = df\_pollution.groupby('state')['station'].nunique().reset\_index()

stations\_per\_state.columns = ['state', 'Num\_Stations']

# Step 2: Merge with state population data

stations\_population = pd.merge(stations\_per\_state, df\_population[['State', 'Population']], on='State', how='left')

# Step 3: Calculate the number of stations per million people

stations\_population['Stations\_per\_Million'] = (stations\_population['Num\_Stations'] / stations\_population['population']) \* 1\_000\_000

# Step 4: Find the state with the highest ratio

max\_stations\_state = stations\_population.loc[stations\_population['Stations\_per\_Million'].idxmax()]

print(f"The state with the highest number of monitoring stations relative to its population is {max\_stations\_state['State']}, with {max\_stations\_state['Stations\_per\_Million']:.2f} stations per million people.")

34/138:

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

df.rename(columns={'state': 'State'}, inplace=True)

df.head()

34/139:

import pandas as pd

# Load the datasets

df\_pollution = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv") # Air quality monitoring data

df\_pollution.rename(columns={'state': 'State'}, inplace=True)

df\_population = pd.read\_csv(r"C:\Users\inees\Downloads\State\_data.csv") # State population data

# Step 1: Count the number of monitoring stations per state

stations\_per\_state = df\_pollution.groupby('state')['station'].nunique().reset\_index()

stations\_per\_state.columns = ['state', 'Num\_Stations']

# Step 2: Merge with state population data

stations\_population = pd.merge(stations\_per\_state, df\_population[['State', 'Population']], on='State', how='left')

# Step 3: Calculate the number of stations per million people

stations\_population['Stations\_per\_Million'] = (stations\_population['Num\_Stations'] / stations\_population['population']) \* 1\_000\_000

# Step 4: Find the state with the highest ratio

max\_stations\_state = stations\_population.loc[stations\_population['Stations\_per\_Million'].idxmax()]

print(f"The state with the highest number of monitoring stations relative to its population is {max\_stations\_state['State']}, with {max\_stations\_state['Stations\_per\_Million']:.2f} stations per million people.")

34/140:

import pandas as pd

# Load the datasets

df\_pollution = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv") # Air quality monitoring data

df\_pollution.rename(columns={'state': 'State'}, inplace=True)

df\_population = pd.read\_csv(r"C:\Users\inees\Downloads\State\_data.csv") # State population data

# Step 1: Count the number of monitoring stations per state

stations\_per\_state = df\_pollution.groupby('State')['station'].nunique().reset\_index()

stations\_per\_state.columns = ['State', 'Num\_Stations']

# Step 2: Merge with state population data

stations\_population = pd.merge(stations\_per\_state, df\_population[['State', 'Population']], on='State', how='left')

# Step 3: Calculate the number of stations per million people

stations\_population['Stations\_per\_Million'] = (stations\_population['Num\_Stations'] / stations\_population['population']) \* 1\_000\_000

# Step 4: Find the state with the highest ratio

max\_stations\_state = stations\_population.loc[stations\_population['Stations\_per\_Million'].idxmax()]

print(f"The state with the highest number of monitoring stations relative to its population is {max\_stations\_state['State']}, with {max\_stations\_state['Stations\_per\_Million']:.2f} stations per million people.")

34/141:

import pandas as pd

# Load the datasets

df\_pollution = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv") # Air quality monitoring data

df\_pollution.rename(columns={'state': 'State'}, inplace=True)

df\_population = pd.read\_csv(r"C:\Users\inees\Downloads\State\_data.csv") # State population data

# Step 1: Count the number of monitoring stations per state

stations\_per\_state = df\_pollution.groupby('State')['station'].nunique().reset\_index()

stations\_per\_state.columns = ['State', 'Num\_Stations']

# Step 2: Merge with state population data

stations\_population = pd.merge(stations\_per\_state, df\_population[['State', 'Population']], on='State', how='left')

# Step 3: Calculate the number of stations per million people

stations\_population['Stations\_per\_Million'] = (stations\_population['Num\_Stations'] / stations\_population['Population']) \* 1\_000\_000

# Step 4: Find the state with the highest ratio

max\_stations\_state = stations\_population.loc[stations\_population['Stations\_per\_Million'].idxmax()]

print(f"The state with the highest number of monitoring stations relative to its population is {max\_stations\_state['State']}, with {max\_stations\_state['Stations\_per\_Million']:.2f} stations per million people.")

34/142:

import pandas as pd

import matplotlib.pyplot as plt

# Load datasets

df = pd.read\_csv('Data.csv', parse\_dates=['Timestamp'])

df.rename(columns={'state': 'State'}, inplace=True)

state\_data = pd.read\_csv('State\_data.csv')

# Extract year and filter for 2023

df['Year'] = df['Timestamp'].dt.year

df\_2023 = df[df['Year'] == 2023].copy()

# Group by state and calculate average PM2.5 levels

state\_pm25 = df\_2023.groupby('State')['PM2.5'].mean()

# Merge with population data

state\_pm25 = state\_pm25.reset\_index()

state\_data = state\_data[['State', 'Population']]

merged = pd.merge(state\_pm25, state\_data, on='State', how='left')

# Calculate per capita PM2.5 exposure

merged['Per\_Capita\_PM2.5'] = merged['PM2.5'] / merged['Population']

# Get top 5 most polluted states per capita

top5\_states = merged.nlargest(5, 'Per\_Capita\_PM2.5')

# Plot bar chart

plt.figure(figsize=(10, 6))

plt.bar(top5\_states['State'], top5\_states['Per\_Capita\_PM2.5'], color='red')

plt.xlabel('State')

plt.ylabel('Per Capita PM2.5 Exposure')

plt.title('Top 5 Most Polluted States (Per Capita PM2.5 Exposure) in 2023')

plt.xticks(rotation=45)

plt.show()

34/143:

df = pd.read\_csv(r"C:\Users\inees\Downloads\NCAP\_Funding.csv")

df.head()

34/144:

import pandas as pd

import matplotlib.pyplot as plt

# Load datasets

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv", parse\_dates=['Timestamp'])

df.rename(columns={'state': 'State'}, inplace=True)

state\_data = pd.read\_csv(r"C:\Users\inees\Downloads\State\_data.csv")

# Extract year and filter for 2023

df['Year'] = df['Timestamp'].dt.year

df\_2023 = df[df['Year'] == 2023].copy()

# Group by state and calculate average PM2.5 levels

state\_pm25 = df\_2023.groupby('State')['PM2.5'].mean()

# Merge with population data

state\_pm25 = state\_pm25.reset\_index()

state\_data = state\_data[['State', 'Population']]

merged = pd.merge(state\_pm25, state\_data, on='State', how='left')

# Calculate per capita PM2.5 exposure

merged['Per\_Capita\_PM2.5'] = merged['PM2.5'] / merged['Population']

# Get top 5 most polluted states per capita

top5\_states = merged.nlargest(5, 'Per\_Capita\_PM2.5')

# Plot bar chart

plt.figure(figsize=(10, 6))

plt.bar(top5\_states['State'], top5\_states['Per\_Capita\_PM2.5'], color='red')

plt.xlabel('State')

plt.ylabel('Per Capita PM2.5 Exposure')

plt.title('Top 5 Most Polluted States (Per Capita PM2.5 Exposure) in 2023')

plt.xticks(rotation=45)

plt.show()

34/145:

import pandas as pd

import matplotlib.pyplot as plt

# Load datasets

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv", parse\_dates=['Timestamp'])

df.rename(columns={'state': 'State'}, inplace=True)

state\_data = pd.read\_csv(r"C:\Users\inees\Downloads\State\_data.csv")

# Extract year and filter for 2023

df['Year'] = df['Timestamp'].dt.year

df\_2023 = df[df['Year'] == 2023].copy()

# Group by state and calculate average PM2.5 levels

state\_pm25 = df\_2023.groupby('State')['PM2.5'].mean()

# Merge with population data

state\_pm25 = state\_pm25.reset\_index()

state\_data = state\_data[['State', 'Population']]

merged = pd.merge(state\_pm25, state\_data, on='State', how='left')

# Calculate per capita PM2.5 exposure

merged['Per\_Capita\_PM2.5'] = merged['PM2.5'] / merged['Population']

# Get top 5 most polluted states per capita

top5\_states = merged.nlargest(5, 'Per\_Capita\_PM2.5')

# Plot bar chart

plt.figure(figsize=(10, 6))

plt.bar(top5\_states['State'], top5\_states['Per\_Capita\_PM2.5'])

plt.xlabel('State')

plt.ylabel('Per Capita PM2.5 Exposure')

plt.title('Top 5 Most Polluted States (Per Capita PM2.5 Exposure) in 2023')

plt.xticks(rotation=45)

plt.show()

34/146:

import pandas as pd

import matplotlib.pyplot as plt

# File paths

state\_data\_path = r"C:\Users\inees\Downloads\State\_data.csv"

data\_path = r"C:\Users\inees\Downloads\Data.csv"

# Load datasets

df = pd.read\_csv(data\_path, parse\_dates=['Timestamp'])

df.rename(columns={'state': 'State'}, inplace=True)

state\_data = pd.read\_csv(state\_data\_path)

# Extract year and filter for the latest available year (e.g., 2023)

df['Year'] = df['Timestamp'].dt.year

latest\_year = df['Year'].max()

df\_latest = df[df['Year'] == latest\_year]

# Group by state and calculate average PM2.5 levels

state\_pm25 = df\_latest.groupby('State')['PM2.5'].mean().reset\_index()

# Calculate population density (Population / Area)

state\_data['Population Density'] = state\_data['Population'] / state\_data['Area (km²)']

# Merge the data

merged = pd.merge(state\_pm25, state\_data[['State', 'Population Density']], on='State', how='left')

# Plot scatter plot

plt.figure(figsize=(10, 6))

plt.scatter(merged['Population Density'], merged['PM2.5'], color='blue', alpha=0.7)

plt.xlabel('Population Density (people per km²)')

plt.ylabel('Average PM2.5 Concentration')

plt.title(f'Population Density vs. PM2.5 Concentration ({latest\_year})')

plt.grid(True)

# Show plot

plt.show()

34/147:

import pandas as pd

import matplotlib.pyplot as plt

# File paths

state\_data\_path = r"C:\Users\inees\Downloads\State\_data.csv"

data\_path = r"C:\Users\inees\Downloads\Data.csv"

# Load datasets

df = pd.read\_csv(data\_path, parse\_dates=['Timestamp'])

df.rename(columns={'state': 'State'}, inplace=True)

state\_data = pd.read\_csv(state\_data\_path)

# Extract year and filter for the latest available year (e.g., 2023)

df['Year'] = df['Timestamp'].dt.year

latest\_year = df['Year'].max()

df\_latest = df[df['Year'] == latest\_year]

# Group by state and calculate average PM2.5 levels

state\_pm25 = df\_latest.groupby('State')['PM2.5'].mean().reset\_index()

# Calculate population density (Population / Area)

state\_data['Population Density'] = state\_data['Population'] / state\_data['Area']

# Merge the data

merged = pd.merge(state\_pm25, state\_data[['State', 'Population Density']], on='State', how='left')

# Plot scatter plot

plt.figure(figsize=(10, 6))

plt.scatter(merged['Population Density'], merged['PM2.5'], color='blue', alpha=0.7)

plt.xlabel('Population Density (people per km²)')

plt.ylabel('Average PM2.5 Concentration')

plt.title(f'Population Density vs. PM2.5 Concentration ({latest\_year})')

plt.grid(True)

# Show plot

plt.show()

34/148:

import pandas as pd

import matplotlib.pyplot as plt

# File paths

state\_data\_path = r"C:\Users\inees\Downloads\State\_data.csv"

data\_path = r"C:\Users\inees\Downloads\Data.csv"

# Load datasets

df = pd.read\_csv(data\_path, parse\_dates=['Timestamp'])

df.rename(columns={'state': 'State'}, inplace=True)

state\_data = pd.read\_csv(state\_data\_path)

# Extract year and filter for the latest available year (e.g., 2023)

df['Year'] = df['Timestamp'].dt.year

latest\_year = df['Year'].max()

df\_latest = df[df['Year'] == latest\_year]

# Group by state and calculate average PM2.5 levels

state\_pm25 = df\_latest.groupby('State')['PM2.5'].mean().reset\_index()

# Calculate population density (Population / Area)

state\_data['Population Density'] = state\_data['Population'] / state\_data['Area (km2)']

# Merge the data

merged = pd.merge(state\_pm25, state\_data[['State', 'Population Density']], on='State', how='left')

# Plot scatter plot

plt.figure(figsize=(10, 6))

plt.scatter(merged['Population Density'], merged['PM2.5'], color='blue', alpha=0.7)

plt.xlabel('Population Density (people per km²)')

plt.ylabel('Average PM2.5 Concentration')

plt.title(f'Population Density vs. PM2.5 Concentration ({latest\_year})')

plt.grid(True)

# Show plot

plt.show()

34/149:

import pandas as pd

import matplotlib.pyplot as plt

# Load datasets

pollution\_data\_path = r"C:\Users\inees\Downloads\Data.csv"

state\_data\_path = r"C:\Users\inees\Downloads\State\_data.csv"

df = pd.read\_csv(pollution\_data\_path, parse\_dates=['Timestamp'])

df.rename(columns={'state': 'State'}, inplace=True)

state\_data = pd.read\_csv(state\_data\_path)

# Group by state and calculate average PM2.5

pollution = df.groupby('State')['PM2.5'].mean().reset\_index()

# Merge pollution data with state area

merged = pd.merge(pollution, state\_data[['State', 'Area (km²)']], on='State', how='inner')

# Calculate PM2.5 concentration per square kilometer

merged['PM2.5 per km²'] = merged['PM2.5'] / merged['Area (km²)']

# Sort states by PM2.5 per km² in descending order

merged = merged.sort\_values(by='PM2.5 per km²', ascending=False)

# Plot

plt.figure(figsize=(12, 6))

plt.bar(merged['State'], merged['PM2.5 per km²'], color='red', alpha=0.7)

plt.xticks(rotation=90) # Rotate state names for better readability

plt.xlabel('State')

plt.ylabel('PM2.5 Concentration per km²')

plt.title('PM2.5 Concentration per Square Kilometer by State')

plt.grid(axis='y', linestyle='--', alpha=0.7)

# Show plot

plt.show()

# Find the state with the highest PM2.5 per km²

most\_polluted\_state = merged.iloc[0]['State']

highest\_pm25\_value = merged.iloc[0]['PM2.5 per km²']

print(f"State with the highest PM2.5 concentration per km²: {most\_polluted\_state} ({highest\_pm25\_value:.2f})")

34/150:

import pandas as pd

import matplotlib.pyplot as plt

# Load datasets

pollution\_data\_path = r"C:\Users\inees\Downloads\Data.csv"

state\_data\_path = r"C:\Users\inees\Downloads\State\_data.csv"

df = pd.read\_csv(pollution\_data\_path, parse\_dates=['Timestamp'])

df.rename(columns={'state': 'State'}, inplace=True)

state\_data = pd.read\_csv(state\_data\_path)

# Group by state and calculate average PM2.5

pollution = df.groupby('State')['PM2.5'].mean().reset\_index()

# Merge pollution data with state area

merged = pd.merge(pollution, state\_data[['State', 'Area (km2)']], on='State', how='inner')

# Calculate PM2.5 concentration per square kilometer

merged['PM2.5 per km2'] = merged['PM2.5'] / merged['Area (km2)']

# Sort states by PM2.5 per km² in descending order

merged = merged.sort\_values(by='PM2.5 per km2', ascending=False)

# Plot

plt.figure(figsize=(12, 6))

plt.bar(merged['State'], merged['PM2.5 per km2'], color='red', alpha=0.7)

plt.xticks(rotation=90) # Rotate state names for better readability

plt.xlabel('State')

plt.ylabel('PM2.5 Concentration per km2')

plt.title('PM2.5 Concentration per Square Kilometer by State')

plt.grid(axis='y', linestyle='--', alpha=0.7)

# Show plot

plt.show()

# Find the state with the highest PM2.5 per km²

most\_polluted\_state = merged.iloc[0]['State']

highest\_pm25\_value = merged.iloc[0]['PM2.5 per km2']

print(f"State with the highest PM2.5 concentration per km2: {most\_polluted\_state} ({highest\_pm25\_value:.2f})")

34/151:

import pandas as pd

import matplotlib.pyplot as plt

# Load datasets

pollution\_data\_path = r"C:\Users\inees\Downloads\Data.csv"

state\_data\_path = r"C:\Users\inees\Downloads\State\_data.csv"

df = pd.read\_csv(pollution\_data\_path, parse\_dates=['Timestamp'])

df.rename(columns={'state': 'State'}, inplace=True)

state\_data = pd.read\_csv(state\_data\_path)

# Group by state and calculate average PM2.5

pollution = df.groupby('State')['PM2.5'].mean().reset\_index()

# Merge pollution data with state area

merged = pd.merge(pollution, state\_data[['State', 'Area (km2)']], on='State', how='inner')

# Calculate PM2.5 concentration per square kilometer

merged['PM2.5 per km2'] = merged['PM2.5'] / merged['Area (km2)']

# Sort states by PM2.5 per km² in descending order

merged = merged.sort\_values(by='PM2.5 per km2', ascending=False)

# Plot

plt.figure(figsize=(12, 6))

plt.bar(merged['State'], merged['PM2.5 per km2'], alpha=0.7)

plt.xticks(rotation=90) # Rotate state names for better readability

plt.xlabel('State')

plt.ylabel('PM2.5 Concentration per km2')

plt.title('PM2.5 Concentration per Square Kilometer by State')

plt.grid(axis='y', linestyle='--', alpha=0.7)

# Show plot

plt.show()

# Find the state with the highest PM2.5 per km²

most\_polluted\_state = merged.iloc[0]['State']

highest\_pm25\_value = merged.iloc[0]['PM2.5 per km2']

print(f"State with the highest PM2.5 concentration per km2: {most\_polluted\_state} ({highest\_pm25\_value:.2f})")

34/152:

import pandas as pd

import matplotlib.pyplot as plt

# Load datasets

pollution\_data\_path = r"C:\Users\inees\Downloads\Data.csv"

state\_data\_path = r"C:\Users\inees\Downloads\State\_data.csv"

df = pd.read\_csv(pollution\_data\_path, parse\_dates=['Timestamp'])

df.rename(columns={'state': 'State'}, inplace=True)

state\_data = pd.read\_csv(state\_data\_path)

# Group by state and calculate average PM2.5

pollution = df.groupby('State')['PM2.5'].mean().reset\_index()

# Merge pollution data with state area

merged = pd.merge(pollution, state\_data[['State', 'Area (km2)']], on='State', how='inner')

# Calculate PM2.5 concentration per square kilometer

merged['PM2.5 per km2'] = merged['PM2.5'] / merged['Area (km2)']

# Sort states by PM2.5 per km² in descending order

merged = merged.sort\_values(by='PM2.5 per km2', ascending=False)

# Plot

plt.figure(figsize=(12, 6))

plt.bar(merged['State'], merged['PM2.5 per km2'])

plt.xticks(rotation=90) # Rotate state names for better readability

plt.xlabel('State')

plt.ylabel('PM2.5 Concentration per km2')

plt.title('PM2.5 Concentration per Square Kilometer by State')

plt.grid(axis='y', linestyle='--', alpha=0.7)

# Show plot

plt.show()

# Find the state with the highest PM2.5 per km²

most\_polluted\_state = merged.iloc[0]['State']

highest\_pm25\_value = merged.iloc[0]['PM2.5 per km2']

print(f"State with the highest PM2.5 concentration per km2: {most\_polluted\_state} ({highest\_pm25\_value:.2f})")

34/153:

import pandas as pd

import matplotlib.pyplot as plt

# Load datasets

data\_path = r"C:\Users\inees\Downloads\Data.csv"

state\_data\_path = r"C:\Users\inees\Downloads\State\_data.csv"

Data = pd.read\_csv(data\_path)

State\_data = pd.read\_csv(state\_data\_path)

# Standardizing column names for merging

Data.rename(columns={'state': 'State'}, inplace=True)

# Remove NaN values in PM2.5 column

filteredData = Data.dropna(subset=['PM2.5'])

# Count the number of unique monitoring stations in each state

station\_counts = Data.groupby('State')['Station'].nunique().reset\_index()

station\_counts.rename(columns={'Station': 'Number of Stations'}, inplace=True)

# Merge with State\_data to get the area of each state

merged\_data = station\_counts.merge(State\_data, on="State")

# Calculate monitoring station density (stations per km²)

merged\_data['Stations per km'] = merged\_data['Number of Stations'] / merged\_data['Area (km²)']

# Plot the bar chart

plt.figure(figsize=(12, 6))

plt.bar(merged\_data['State'], merged\_data['Stations per km'], color='skyblue')

plt.xticks(rotation=90)

plt.xlabel("State")

plt.ylabel("Stations per km")

plt.title("Density of Monitoring Stations in Each State")

plt.show()

# Identify the state with the highest density

max\_density\_state = merged\_data.loc[merged\_data['Stations per km'].idxmax(), 'State']

print(f"State with the highest monitoring station density: {max\_density\_state}")

34/154:

import pandas as pd

import matplotlib.pyplot as plt

# Load datasets

data\_path = r"C:\Users\inees\Downloads\Data.csv"

state\_data\_path = r"C:\Users\inees\Downloads\State\_data.csv"

Data = pd.read\_csv(data\_path)

State\_data = pd.read\_csv(state\_data\_path)

# Standardizing column names for merging

Data.rename(columns={'state': 'State'}, inplace=True)

# Remove NaN values in PM2.5 column

filteredData = Data.dropna(subset=['PM2.5'])

# Count the number of unique monitoring stations in each state

station\_counts = Data.groupby('State')['station'].nunique().reset\_index()

station\_counts.rename(columns={'Station': 'Number of Stations'}, inplace=True)

# Merge with State\_data to get the area of each state

merged\_data = station\_counts.merge(State\_data, on="State")

# Calculate monitoring station density (stations per km²)

merged\_data['Stations per km'] = merged\_data['Number of Stations'] / merged\_data['Area (km²)']

# Plot the bar chart

plt.figure(figsize=(12, 6))

plt.bar(merged\_data['State'], merged\_data['Stations per km'])

plt.xticks(rotation=90)

plt.xlabel("State")

plt.ylabel("Stations per km")

plt.title("Density of Monitoring Stations in Each State")

plt.show()

# Identify the state with the highest density

max\_density\_state = merged\_data.loc[merged\_data['Stations per km'].idxmax(), 'State']

print(f"State with the highest monitoring station density: {max\_density\_state}")

34/155:

import pandas as pd

import matplotlib.pyplot as plt

# Load datasets

data\_path = r"C:\Users\inees\Downloads\Data.csv"

state\_data\_path = r"C:\Users\inees\Downloads\State\_data.csv"

Data = pd.read\_csv(data\_path)

State\_data = pd.read\_csv(state\_data\_path)

# Standardizing column names for merging

Data.rename(columns={'state': 'State'}, inplace=True)

# Remove NaN values in PM2.5 column

filteredData = Data.dropna(subset=['PM2.5'])

# Count the number of unique monitoring stations in each state

station\_counts = Data.groupby('State')['station'].nunique().reset\_index()

station\_counts.rename(columns={'station': 'Number of Stations'}, inplace=True)

# Merge with State\_data to get the area of each state

merged\_data = station\_counts.merge(State\_data, on="State")

# Calculate monitoring station density (stations per km²)

merged\_data['Stations per km'] = merged\_data['Number of Stations'] / merged\_data['Area (km²)']

# Plot the bar chart

plt.figure(figsize=(12, 6))

plt.bar(merged\_data['State'], merged\_data['Stations per km'])

plt.xticks(rotation=90)

plt.xlabel("State")

plt.ylabel("Stations per km")

plt.title("Density of Monitoring Stations in Each State")

plt.show()

# Identify the state with the highest density

max\_density\_state = merged\_data.loc[merged\_data['Stations per km'].idxmax(), 'State']

print(f"State with the highest monitoring station density: {max\_density\_state}")

34/156:

import pandas as pd

import matplotlib.pyplot as plt

# Load datasets

data\_path = r"C:\Users\inees\Downloads\Data.csv"

state\_data\_path = r"C:\Users\inees\Downloads\State\_data.csv"

Data = pd.read\_csv(data\_path)

State\_data = pd.read\_csv(state\_data\_path)

# Standardizing column names for merging

Data.rename(columns={'state': 'State'}, inplace=True)

# Remove NaN values in PM2.5 column

filteredData = Data.dropna(subset=['PM2.5'])

# Count the number of unique monitoring stations in each state

station\_counts = Data.groupby('State')['station'].nunique().reset\_index()

station\_counts.rename(columns={'station': 'Number of Stations'}, inplace=True)

# Merge with State\_data to get the area of each state

merged\_data = station\_counts.merge(State\_data, on="State")

# Calculate monitoring station density (stations per km2)

merged\_data['Stations per km2'] = merged\_data['Number of Stations'] / merged\_data['Area (km2)']

# Plot the bar chart

plt.figure(figsize=(12, 6))

plt.bar(merged\_data['State'], merged\_data['Stations per km2'])

plt.xticks(rotation=90)

plt.xlabel("State")

plt.ylabel("Stations per km2")

plt.title("Density of Monitoring Stations in Each State")

plt.show()

# Identify the state with the highest density

max\_density\_state = merged\_data.loc[merged\_data['Stations per km2'].idxmax(), 'State']

print(f"State with the highest monitoring station density: {max\_density\_state}")

34/157:

import pandas as pd

# Filter data for 2021

year\_mask = Data['Timestamp'].str[:4] == '2021'

twenty21 = Data.loc[year\_mask]

# Calculate the mean PM2.5 for each state in 2021

mean\_pm25 = twenty21.groupby(['State'])['PM2.5'].mean().reset\_index()

# Filter the data for Maharashtra and Madhya Pradesh

ans = mean\_pm25[mean\_pm25['State'].isin(['Maharashtra', 'Madhya Pradesh'])]

print("Average PM2.5 for Maharashtra and Madhya Pradesh in 2021:")

print(ans, '\n')

# Calculate population density for Maharashtra and Madhya Pradesh

mhmp = State\_data[State\_data['State'].isin(['Maharashtra', 'Madhya Pradesh'])]

PopulationDensity = mhmp['Population'] / mhmp['Area (km2)']

PopulationDensity.name = 'Population Density'

# Join the population density with the PM2.5 levels

ans = ans.set\_index('State').join(PopulationDensity).reset\_index()

print("Population Density and Average PM2.5 levels:")

print(ans)

34/158:

import pandas as pd

import matplotlib.pyplot as plt

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

df['Timestamp'] = pd.to\_datetime(df['Timestamp'], format='%Y-%m-%d')

# Get Delhi data

df\_delhi = df[df['city'] == 'Delhi'].copy()

# Get year and month

df\_delhi['Year'] = pd.to\_datetime(df\_delhi['Timestamp'], format='%Y-%m-%d').dt.year

df\_delhi['Month'] = pd.to\_datetime(df\_delhi['Timestamp'], format='%Y-%m-%d').dt.month

# Define seasons

def get\_season(month):

if month in [12, 1, 2]:

return 'Winter'

elif month in [3, 4, 5]:

return 'Summer'

elif month in [6, 7, 8, 9]:

return 'Monsoon'

else:

return 'Other'

df\_delhi['Season'] = df\_delhi['Month'].apply(get\_season)

# Filter data for the years 2017 to 2023, summer winter monsoon seasons

df\_delhi = df\_delhi[(df\_delhi['Year'] >= 2017) &

(df\_delhi['Year'] <= 2023) &

(df\_delhi['Season'].isin(['Winter', 'Summer', 'Monsoon']))]

# Calculate the avg PM2.5 levels for each season from 2017 - 2023

seasonal\_avg\_pm25 = df\_delhi.groupby(['Year', 'Season'])['PM2.5'].mean().unstack()

# Plot

plt.figure(figsize=(10, 6))

seasonal\_avg\_pm25.plot(kind='line', marker='o', figsize=(12, 6))

plt.title("Average PM2.5 Levels in Delhi (2017-2023) by Season")

plt.xlabel("Year")

plt.ylabel("Average PM2.5")

plt.legend(title="Season", loc='upper left')

plt.grid(True)

plt.tight\_layout()

plt.show()

34/159: From the graph, we can see that winter months have the highest pollution levels due to atmospheric inversion and low wind speeds. Monsoon months show a significant decrease in PM2.5 levels due to rain, while summer has moderate pollution levels due to better air circulation but can still experience pollution spikes.

34/160:

import pandas as pd

import matplotlib.pyplot as plt

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

df['Timestamp'] = pd.to\_datetime(df['Timestamp'], format='%Y-%m-%d')

# Get data for Delhi and Mumbai

df\_delhi\_mumbai = df[df['city'].isin(['Delhi', 'Mumbai'])].copy()

# Get year and month from the Timestamp

df\_delhi\_mumbai['Year'] = df\_delhi\_mumbai['Timestamp'].dt.year

df\_delhi\_mumbai['Month'] = df\_delhi\_mumbai['Timestamp'].dt.month

# Get data for years between 2017 and 2023

df\_delhi\_mumbai = df\_delhi\_mumbai[(df\_delhi\_mumbai['Year'] >= 2017) &

(df\_delhi\_mumbai['Year'] <= 2023)]

# monthly avg PM2.5 levels for Delhi and Mumbai

city\_avg\_pm25 = df\_delhi\_mumbai.groupby(['Year', 'Month', 'city'])['PM2.5'].mean().unstack()

# Plot

plt.figure(figsize=(12, 6))

city\_avg\_pm25['Delhi'].plot(label='Delhi', marker='s', linestyle='-', color='sb')

city\_avg\_pm25['Mumbai'].plot(label='Mumbai', marker='o', linestyle='-', color='r')

plt.title("PM2.5 Levels in Delhi and Mumbai (2017-2023)")

plt.xlabel("Year")

plt.ylabel("Average PM2.5")

plt.legend()

plt.tight\_layout()

# Show the plot

plt.show()

34/161:

import pandas as pd

import matplotlib.pyplot as plt

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

df['Timestamp'] = pd.to\_datetime(df['Timestamp'], format='%Y-%m-%d')

# Get data for Delhi and Mumbai

df\_delhi\_mumbai = df[df['city'].isin(['Delhi', 'Mumbai'])].copy()

# Get year and month from the Timestamp

df\_delhi\_mumbai['Year'] = df\_delhi\_mumbai['Timestamp'].dt.year

df\_delhi\_mumbai['Month'] = df\_delhi\_mumbai['Timestamp'].dt.month

# Get data for years between 2017 and 2023

df\_delhi\_mumbai = df\_delhi\_mumbai[(df\_delhi\_mumbai['Year'] >= 2017) &

(df\_delhi\_mumbai['Year'] <= 2023)]

# monthly avg PM2.5 levels for Delhi and Mumbai

city\_avg\_pm25 = df\_delhi\_mumbai.groupby(['Year', 'Month', 'city'])['PM2.5'].mean().unstack()

# Plot

plt.figure(figsize=(12, 6))

city\_avg\_pm25['Delhi'].plot(label='Delhi', marker='s', linestyle='-', color='black')

city\_avg\_pm25['Mumbai'].plot(label='Mumbai', marker='o', linestyle='-', color='skyblue')

plt.title("PM2.5 Levels in Delhi and Mumbai (2017-2023)")

plt.xlabel("Year")

plt.ylabel("Average PM2.5")

plt.legend()

plt.tight\_layout()

# Show the plot

plt.show()

34/162:

import pandas as pd

import matplotlib.pyplot as plt

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

df['Timestamp'] = pd.to\_datetime(df['Timestamp'], format='%Y-%m-%d')

# Get data for Delhi and Mumbai

df\_delhi\_mumbai = df[df['city'].isin(['Delhi', 'Mumbai'])].copy()

# Get year and month from the Timestamp

df\_delhi\_mumbai['Year'] = df\_delhi\_mumbai['Timestamp'].dt.year

df\_delhi\_mumbai['Month'] = df\_delhi\_mumbai['Timestamp'].dt.month

# Get data for years between 2017 and 2023

df\_delhi\_mumbai = df\_delhi\_mumbai[(df\_delhi\_mumbai['Year'] >= 2017) &

(df\_delhi\_mumbai['Year'] <= 2023)]

# monthly avg PM2.5 levels for Delhi and Mumbai

city\_avg\_pm25 = df\_delhi\_mumbai.groupby(['Year', 'Month', 'city'])['PM2.5'].mean().unstack()

# Plot

plt.figure(figsize=(12, 6))

city\_avg\_pm25['Delhi'].plot(label='Delhi', marker='o', linestyle='-', color='skyblue')

city\_avg\_pm25['Mumbai'].plot(label='Mumbai', marker='o', linestyle='-', color='black')

plt.title("PM2.5 Levels in Delhi and Mumbai (2017-2023)")

plt.xlabel("Year")

plt.ylabel("Average PM2.5")

plt.legend()

plt.tight\_layout()

# Show the plot

plt.show()

34/163:

import pandas as pd

import matplotlib.pyplot as plt

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

df['Timestamp'] = pd.to\_datetime(df['Timestamp'], format='%Y-%m-%d')

# Get data for Delhi and Mumbai

df\_delhi\_mumbai = df[df['city'].isin(['Delhi', 'Mumbai'])].copy()

# Get year and month from the Timestamp

df\_delhi\_mumbai['Year'] = df\_delhi\_mumbai['Timestamp'].dt.year

df\_delhi\_mumbai['Month'] = df\_delhi\_mumbai['Timestamp'].dt.month

# Get data for years between 2017 and 2023

df\_delhi\_mumbai = df\_delhi\_mumbai[(df\_delhi\_mumbai['Year'] >= 2017) &

(df\_delhi\_mumbai['Year'] <= 2023)]

# monthly avg PM2.5 levels for Delhi and Mumbai

city\_avg\_pm25 = df\_delhi\_mumbai.groupby(['Year', 'Month', 'city'])['PM2.5'].mean().unstack()

print(city\_avg\_pm25)

# Plot

plt.figure(figsize=(12, 6))

city\_avg\_pm25['Delhi'].plot(label='Delhi', marker='o', linestyle='-', color='skyblue')

city\_avg\_pm25['Mumbai'].plot(label='Mumbai', marker='o', linestyle='-', color='black')

plt.title("PM2.5 Levels in Delhi and Mumbai (2017-2023)")

plt.xlabel("Year")

plt.ylabel("Average PM2.5")

plt.legend()

plt.tight\_layout()

# Show the plot

plt.show()

34/164:

import pandas as pd

import matplotlib.pyplot as plt

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

df['Timestamp'] = pd.to\_datetime(df['Timestamp'], format='%Y-%m-%d')

# Get data for Delhi and Mumbai

df\_delhi\_mumbai = df[df['city'].isin(['Delhi', 'Mumbai'])].copy()

# Get year and month from the Timestamp

df\_delhi\_mumbai['Year'] = df\_delhi\_mumbai['Timestamp'].dt.year

df\_delhi\_mumbai['Month'] = df\_delhi\_mumbai['Timestamp'].dt.month

# Get data for years between 2017 and 2023

df\_delhi\_mumbai = df\_delhi\_mumbai[(df\_delhi\_mumbai['Year'] >= 2017) &

(df\_delhi\_mumbai['Year'] <= 2023)]

# monthly avg PM2.5 levels for Delhi and Mumbai

city\_avg\_pm25 = df\_delhi\_mumbai.groupby(['Year', 'Month', 'city'])['PM2.5'].mean().unstack()

# Plot

plt.figure(figsize=(12, 6))

city\_avg\_pm25['Delhi'].plot(label='Delhi', marker='o', linestyle='-', color='skyblue')

city\_avg\_pm25['Mumbai'].plot(label='Mumbai', marker='o', linestyle='-', color='black')

plt.title("PM2.5 Levels in Delhi and Mumbai (2017-2023)")

plt.xlabel("Year")

plt.ylabel("Average PM2.5")

plt.legend()

plt.tight\_layout()

# Show the plot

plt.show()

34/165:

import pandas as pd

import matplotlib.pyplot as plt

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

df['Timestamp'] = pd.to\_datetime(df['Timestamp'], format='%Y-%m-%d')

# Get data for Delhi and Mumbai

df\_delhi\_mumbai = df[df['city'].isin(['Delhi', 'Mumbai'])].copy()

# Get year and month from the Timestamp

df\_delhi\_mumbai['Year'] = df\_delhi\_mumbai['Timestamp'].dt.year

df\_delhi\_mumbai['Month'] = df\_delhi\_mumbai['Timestamp'].dt.month

# Get data for years between 2017 and 2023

df\_delhi\_mumbai = df\_delhi\_mumbai[(df\_delhi\_mumbai['Year'] >= 2017) &

(df\_delhi\_mumbai['Year'] <= 2023)]

# monthly avg PM2.5 levels for Delhi and Mumbai

city\_avg\_pm25 = df\_delhi\_mumbai.groupby(['Year', 'Month', 'city'])['PM2.5'].mean().unstack()

# Plot

plt.figure(figsize=(12, 6))

city\_avg\_pm25['Delhi'].plot(label='Delhi', marker='o', linestyle='-', color='skyblue')

city\_avg\_pm25['Mumbai'].plot(label='Mumbai', marker='o', linestyle='-', color='black')

plt.title("PM2.5 Levels in Delhi and Mumbai (2017-2023)")

plt.xlabel("Year")

plt.ylabel("Average PM2.5")

plt.legend()

plt.tight\_layout()

plt.show()

# Calculate fluctuations (standard deviation) for Delhi and Mumbai

delhi\_fluctuation = city\_avg\_pm25['Delhi'].std()

mumbai\_fluctuation = city\_avg\_pm25['Mumbai'].std()

print(f"Delhi Fluctuation (Standard Deviation): {delhi\_fluctuation}")

print(f"Mumbai Fluctuation (Standard Deviation): {mumbai\_fluctuation}")

if delhi\_fluctuation > mumbai\_fluctuation:

print("Delhi had more fluctuations in PM2.5 levels.")

else:

print("Mumbai had more fluctuations in PM2.5 levels.")

34/166:

import pandas as pd

import matplotlib.pyplot as plt

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

df['Timestamp'] = pd.to\_datetime(df['Timestamp'], format='%Y-%m-%d')

# Get data for Delhi and Mumbai

df\_delhi\_mumbai = df[df['city'].isin(['Delhi', 'Mumbai'])].copy()

# Get year and month from the Timestamp

df\_delhi\_mumbai['Year'] = df\_delhi\_mumbai['Timestamp'].dt.year

df\_delhi\_mumbai['Month'] = df\_delhi\_mumbai['Timestamp'].dt.month

# Get data for years between 2017 and 2023

df\_delhi\_mumbai = df\_delhi\_mumbai[(df\_delhi\_mumbai['Year'] >= 2017) &

(df\_delhi\_mumbai['Year'] <= 2023)]

# monthly avg PM2.5 levels for Delhi and Mumbai

city\_avg\_pm25 = df\_delhi\_mumbai.groupby(['Year', 'Month', 'city'])['PM2.5'].mean().unstack()

# Plot

plt.figure(figsize=(12, 6))

city\_avg\_pm25['Delhi'].plot(label='Delhi', marker='o', linestyle='-', color='skyblue')

city\_avg\_pm25['Mumbai'].plot(label='Mumbai', marker='o', linestyle='-', color='black')

plt.title("PM2.5 Levels in Delhi and Mumbai (2017-2023)")

plt.xlabel("Year")

plt.ylabel("Average PM2.5")

plt.legend()

plt.tight\_layout()

plt.show()

# Calculate fluctuations (standard deviation) for Delhi and Mumbai

delhi\_fluctuation = city\_avg\_pm25['Delhi'].std()

mumbai\_fluctuation = city\_avg\_pm25['Mumbai'].std()

print(f"Delhi Fluctuation (Standard Deviation): {delhi\_fluctuation:.3f}")

print(f"Mumbai Fluctuation (Standard Deviation): {mumbai\_fluctuation:.3f}")

if delhi\_fluctuation > mumbai\_fluctuation:

print("Delhi had more fluctuations in PM2.5 levels.")

else:

print("Mumbai had more fluctuations in PM2.5 levels.")

34/167:

import pandas as pd

# Load the datasets

df\_pollution = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

df\_pollution.rename(columns={'state': 'State'}, inplace=True)

df\_population = pd.read\_csv(r"C:\Users\inees\Downloads\State\_data.csv")

# No of monitoring stations per state

stations\_per\_state = df\_pollution.groupby('State')['station'].count().reset\_index

print(stations\_per\_state)

stations\_per\_state.columns = ['State', 'Num\_Stations']

# Step 2: Merge with state population data

stations\_population = pd.merge(stations\_per\_state, df\_population[['State', 'Population']], on='State', how='left')

# Step 3: Calculate the number of stations per million people

stations\_population['Stations\_per\_Million'] = (stations\_population['Num\_Stations'] / stations\_population['Population']) \* 1\_000\_000

# Step 4: Find the state with the highest ratio

max\_stations\_state = stations\_population.loc[stations\_population['Stations\_per\_Million'].idxmax()]

print(f"The state with the highest number of monitoring stations relative to its population is {max\_stations\_state['State']}, with {max\_stations\_state['Stations\_per\_Million']:.2f} stations per million people.")

34/168:

import pandas as pd

# Load the datasets

df\_pollution = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

df\_pollution.rename(columns={'state': 'State'}, inplace=True)

df\_population = pd.read\_csv(r"C:\Users\inees\Downloads\State\_data.csv")

# No of monitoring stations per state

stations\_per\_state = df\_pollution.groupby('State')['station'].count().reset\_index

stations\_per\_state.columns = ['State', 'Num\_Stations']

# Merge with state population data

stations\_population = pd.merge(stations\_per\_state, df\_population[['State', 'Population']], on='State', how='left')

# No of stations per million people

stations\_population['Stations\_per\_Million'] = (stations\_population['Num\_Stations'] / stations\_population['Population']) \* 1000000

# State with the highest ratio

max\_stations\_state = stations\_population.loc[stations\_population['Stations\_per\_Million'].idxmax()]

print(f"The state with the highest number of monitoring stations relative to its population is {max\_stations\_state['State']}, with {max\_stations\_state['Stations\_per\_Million']:.2f} stations per million people.")

34/169:

import pandas as pd

import matplotlib.pyplot as plt

# Load datasets

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv", parse\_dates=['Timestamp'])

df.rename(columns={'state': 'State'}, inplace=True)

state\_data = pd.read\_csv(r"C:\Users\inees\Downloads\State\_data.csv")

# Extract year and filter for 2023

df['Year'] = df['Timestamp'].dt.year

df\_2023 = df[df['Year'] == 2023].copy()

# Group by state and calculate average PM2.5 levels

state\_pm25 = df\_2023.groupby('State')['PM2.5'].mean()

# Merge with population data

state\_pm25 = state\_pm25.reset\_index()

state\_data = state\_data[['State', 'Population']]

merged = pd.merge(state\_pm25, state\_data, on='State', how='left')

# Calculate per capita PM2.5 exposure

merged['Per\_Capita\_PM2.5'] = merged['PM2.5'] / merged['Population']

# Get top 5 most polluted states per capita

top5\_states = merged.nlargest(5, 'Per\_Capita\_PM2.5')

# Plot bar chart

plt.figure(figsize=(10, 6))

plt.bar(top5\_states['State'], top5\_states['Per\_Capita\_PM2.5'], color = 'skyblue')

plt.xlabel('State')

plt.ylabel('Per Capita PM2.5 Exposure')

plt.title('Top 5 Most Polluted States (Per Capita PM2.5 Exposure) in 2023')

plt.xticks(rotation=45)

plt.show()

34/170:

import pandas as pd

import matplotlib.pyplot as plt

# Load datasets

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv", parse\_dates=['Timestamp'])

df.rename(columns={'state': 'State'}, inplace=True)

state\_data = pd.read\_csv(r"C:\Users\inees\Downloads\State\_data.csv")

# Extract year and filter for 2023

df['Year'] = df['Timestamp'].dt.year

df\_2023 = df[df['Year'] == 2023].copy()

# Group by state and calculate average PM2.5 levels

state\_pm25 = df\_2023.groupby('State')['PM2.5'].mean()

# Merge with population data

state\_pm25 = state\_pm25.reset\_index()

state\_data = state\_data[['State', 'Population']]

merged = pd.merge(state\_pm25, state\_data, on='State', how='left')

# Calculate per capita PM2.5 exposure

merged['Per\_Capita\_PM2.5'] = merged['PM2.5'] / merged['Population']

# Get top 5 most polluted states per capita

top5\_states = merged.nlargest(5, 'Per\_Capita\_PM2.5')

# Plot bar chart

plt.figure(figsize=(10, 6))

plt.bar(top5\_states['State'], top5\_states['Per\_Capita\_PM2.5'], color = 'skyblue')

plt.xlabel('State')

plt.ylabel('Per Capita PM2.5 Exposure')

plt.title('Top 5 Most Polluted States (Per Capita PM2.5 Exposure) in 2023')

plt.show()

34/171:

import pandas as pd

# Load the datasets

df\_pollution = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

df\_pollution.rename(columns={'state': 'State'}, inplace=True)

df\_population = pd.read\_csv(r"C:\Users\inees\Downloads\State\_data.csv")

# No of monitoring stations per state

stations\_per\_state = df\_pollution.groupby('State')['station'].count().reset\_index()

stations\_per\_state.columns = ['State', 'Num\_Stations']

# Merge with state population data

stations\_population = pd.merge(stations\_per\_state, df\_population[['State', 'Population']], on='State', how='left')

# No of stations per million people

stations\_population['Stations\_per\_Million'] = (stations\_population['Num\_Stations'] / stations\_population['Population']) \* 1000000

# State with the highest ratio

max\_stations\_state = stations\_population.loc[stations\_population['Stations\_per\_Million'].idxmax()]

print(f"The state with the highest number of monitoring stations relative to its population is {max\_stations\_state['State']}, with {max\_stations\_state['Stations\_per\_Million']:.2f} stations per million people.")

34/172:

import pandas as pd

import matplotlib.pyplot as plt

# Load datasets

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv", parse\_dates=['Timestamp'])

df.rename(columns={'state': 'State'}, inplace=True)

state\_data = pd.read\_csv(r"C:\Users\inees\Downloads\State\_data.csv")

# Get data for 2023

df\_2023 = df[df['Timestamp'].dt.year == 2023].copy()

# Group by state and calc avg PM2.5 levels

state\_pm25 = df\_2023.groupby('State')['PM2.5'].mean()

# Merge with population data

state\_pm25 = state\_pm25.reset\_index()

state\_data = state\_data[['State', 'Population']]

merged = pd.merge(state\_pm25, state\_data, on='State', how='left')

# Calc per capita PM2.5 exposure

merged['Per\_Capita\_PM2.5'] = merged['PM2.5'] / merged['Population']

# Top 5 most polluted states per capita

top5\_states = merged.nlargest(5, 'Per\_Capita\_PM2.5')

# Plot

plt.figure(figsize=(10, 6))

plt.bar(top5\_states['State'], top5\_states['Per\_Capita\_PM2.5'], color = 'skyblue')

plt.xlabel('State')

plt.ylabel('Per Capita PM2.5 Exposure')

plt.title('Top 5 Most Polluted States in terms of per capita PM2.5 exposure in 2023')

plt.show()

34/173:

import pandas as pd

import matplotlib.pyplot as plt

# Load datasets

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv", parse\_dates=['Timestamp'])

df.rename(columns={'state': 'State'}, inplace=True)

state\_data = pd.read\_csv(r"C:\Users\inees\Downloads\State\_data.csv")

# Get data for 2023

df\_2023 = df[df['Timestamp'].dt.year == 2023].copy()

# Group by state and calc avg PM2.5 levels

state\_pm25 = df\_2023.groupby('State')['PM2.5'].mean()

# Merge with population data

state\_pm25 = state\_pm25.reset\_index()

state\_data = state\_data[['State', 'Population']]

merged = pd.merge(state\_pm25, state\_data, on='State', how='left')

# Calc per capita PM2.5 exposure

merged['Per\_Capita\_PM2.5'] = merged['PM2.5'] / merged['Population']

# Top 5 most polluted states per capita

top5\_states = merged.nlargest(5, 'Per\_Capita\_PM2.5')

# Plot

plt.figure(figsize=(10, 6))

plt.bar(top5\_states['State'], top5\_states['Per\_Capita\_PM2.5'], color = 'skyblue')

plt.xlabel('State')

plt.ylabel('Per Capita PM2.5 Exposure')

plt.title('Top 5 Most Polluted States \nin terms of per capita PM2.5 exposure in 2023')

plt.show()

34/174:

import pandas as pd

import matplotlib.pyplot as plt

# Load the datasets

df\_population = pd.read\_csv(r"C:\Users\inees\Downloads\State\_data.csv")

df\_pollution = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

# Calculate population density: Population / Area

df\_population['Population Density'] = df\_population['Population'] / df\_population['Area (km²)']

# Calculate average PM2.5 levels for each state

df\_pollution['State'] = df\_pollution['state'] # Make sure the 'State' column is properly named

average\_pm25 = df\_pollution.groupby('State')['PM2.5'].mean().reset\_index()

# Merge population density data with the average PM2.5 data

merged\_data = pd.merge(df\_population[['State', 'Population Density']], average\_pm25, on='State', how='inner')

# Plotting the relationship

plt.figure(figsize=(10, 6))

plt.scatter(merged\_data['Population Density'], merged\_data['PM2.5'], color='b', marker='o')

# Adding labels and title

plt.title('Population Density vs Average PM2.5 Concentration for Each State', fontsize=14)

plt.xlabel('Population Density (people per km²)', fontsize=12)

plt.ylabel('Average PM2.5 Concentration (µg/m³)', fontsize=12)

plt.show()

34/175:

import pandas as pd

import matplotlib.pyplot as plt

# Load the datasets

df\_population = pd.read\_csv(r"C:\Users\inees\Downloads\State\_data.csv")

df\_pollution['State'] = df\_pollution['state']

df\_pollution = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

# Calc population density

df\_population['Population Density'] = df\_population['Population'] / df\_population['Area (km2)']

# Calc avg PM2.5 levels for each state

average\_pm25 = df\_pollution.groupby('State')['PM2.5'].mean().reset\_index()

# Merge

merged = pd.merge(df\_population[['State', 'Population Density']], average\_pm25, on='State', how='inner')

# Plotting the relationship

plt.figure(figsize=(10, 6))

plt.scatter(merged['Population Density'], merged['PM2.5'], color='b', marker='o')

plt.title('Population Density vs Average PM2.5 Concentration for Each State', fontsize=14)

plt.xlabel('Population Density (people per km2)', fontsize=12)

plt.ylabel('Average PM2.5 Concentration', fontsize=12)

plt.show()

34/176:

import pandas as pd

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

# Get rows for year 2023

df["Timestamp"] = pd.to\_datetime(df["Timestamp"], format="%Y-%m-%d")

df\_2023 = df[df["Timestamp"].dt.year == 2023]

# Drop rows w PM2.5 missing

df\_2023 = df\_2023.dropna(subset = ["PM2.5"])

# hazardous PM2.5 levels

hazardous\_data = df\_2023[df\_2023["PM2.5"] > 300]

hazardous\_days\_per\_state = hazardous\_data.groupby("state")["Timestamp"].count()

# Find the state with the highest number of hazardous days

most\_hazardous\_state = hazardous\_days\_per\_state.idxmax()

most\_hazardous\_days = hazardous\_days\_per\_state.max()

print(f"The state with the most days having hazardous PM2.5 levels in 2023 is {most\_hazardous\_state} with {most\_hazardous\_days} days.")

34/177:

import pandas as pd

import matplotlib.pyplot as plt

# Load the datasets

df\_population = pd.read\_csv(r"C:\Users\inees\Downloads\State\_data.csv")

df\_pollution.rename(columns={'state': 'State'}, inplace=True)

df\_pollution = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

# Calc population density

df\_population['Population Density'] = df\_population['Population'] / df\_population['Area (km2)']

# Calc avg PM2.5 levels for each state

average\_pm25 = df\_pollution.groupby('State')['PM2.5'].mean().reset\_index()

# Merge

merged = pd.merge(df\_population[['State', 'Population Density']], average\_pm25, on='State', how='inner')

# Plotting the relationship

plt.figure(figsize=(10, 6))

plt.scatter(merged['Population Density'], merged['PM2.5'], color='b', marker='o')

plt.title('Population Density vs Average PM2.5 Concentration for Each State', fontsize=14)

plt.xlabel('Population Density (people per km2)', fontsize=12)

plt.ylabel('Average PM2.5 Concentration', fontsize=12)

plt.show()

34/178:

import pandas as pd

import matplotlib.pyplot as plt

# Load the datasets

df\_population = pd.read\_csv(r"C:\Users\inees\Downloads\State\_data.csv")

df\_pollution = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

df\_pollution.rename(columns={'state': 'State'}, inplace=True)

# Calc population density

df\_population['Population Density'] = df\_population['Population'] / df\_population['Area (km2)']

# Calc avg PM2.5 levels for each state

average\_pm25 = df\_pollution.groupby('State')['PM2.5'].mean().reset\_index()

# Merge

merged = pd.merge(df\_population[['State', 'Population Density']], average\_pm25, on='State', how='inner')

# Plotting the relationship

plt.figure(figsize=(10, 6))

plt.scatter(merged['Population Density'], merged['PM2.5'], color='b', marker='o')

plt.title('Population Density vs Average PM2.5 Concentration for Each State', fontsize=14)

plt.xlabel('Population Density (people per km2)', fontsize=12)

plt.ylabel('Average PM2.5 Concentration', fontsize=12)

plt.show()

34/179:

import pandas as pd

import matplotlib.pyplot as plt

df\_population = pd.read\_csv(r"C:\Users\inees\Downloads\State\_data.csv")

df\_pollution = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

df\_pollution.rename(columns={'state': 'State'}, inplace=True)

# Calc population density

df\_population['Population Density'] = df\_population['Population'] / df\_population['Area (km2)']

# Calc avg PM2.5 levels for each state

average\_pm25 = df\_pollution.groupby('State')['PM2.5'].mean().reset\_index()

# Merge

merged = pd.merge(df\_population[['State', 'Population Density']], average\_pm25, on='State', how='inner')

# Plotting the relationship

plt.figure(figsize=(10, 6))

plt.scatter(merged['Population Density'], merged['PM2.5'], color='b', marker='o')

plt.title('Population Density vs Average PM2.5 Concentration for Each State', fontsize=14)

plt.xlabel('Population Density (people per km2)', fontsize=12)

plt.ylabel('Average PM2.5 Concentration', fontsize=12)

plt.show()

34/180:

import pandas as pd

import matplotlib.pyplot as plt

# Load datasets

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv", parse\_dates=['Timestamp'])

df.rename(columns={'state': 'State'}, inplace=True)

state\_data = pd.read\_csv(r"C:\Users\inees\Downloads\State\_data.csv")

# Get data for 2023

df\_2023 = df[df['Timestamp'].dt.year == 2023].copy()

# Group by state and calc avg PM2.5 levels

state\_pm25 = df\_2023.groupby('State')['PM2.5'].mean()

# Merge with population data

state\_pm25 = state\_pm25.reset\_index()

state\_data = state\_data[['State', 'Population']]

merged = pd.merge(state\_pm25, state\_data, on='State', how='left')

# Calc per capita PM2.5 exposure

merged['Per\_Capita\_PM2.5'] = merged['PM2.5'] / merged['Population']

# Top 5 most polluted states per capita

top5\_states = merged.nlargest(5, 'Per\_Capita\_PM2.5')

# Plot

plt.figure(figsize=(10, 6))

plt.bar(top5\_states['State'], top5\_states['Per\_Capita\_PM2.5'], color = 'skyblue')

plt.xlabel('State')

plt.ylabel('Per Capita PM2.5 Exposure')

plt.title('Top 5 Most Polluted States \nin terms of per capita PM2.5 exposure in 2023')

plt.grid(True)

plt.show()

34/181:

import pandas as pd

import matplotlib.pyplot as plt

# Load datasets

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv", parse\_dates=['Timestamp'])

df.rename(columns={'state': 'State'}, inplace=True)

state\_data = pd.read\_csv(r"C:\Users\inees\Downloads\State\_data.csv")

# Get data for 2023

df\_2023 = df[df['Timestamp'].dt.year == 2023].copy()

# Group by state and calc avg PM2.5 levels

state\_pm25 = df\_2023.groupby('State')['PM2.5'].mean()

# Merge with population data

state\_pm25 = state\_pm25.reset\_index()

state\_data = state\_data[['State', 'Population']]

merged = pd.merge(state\_pm25, state\_data, on='State', how='left')

# Calc per capita PM2.5 exposure

merged['Per\_Capita\_PM2.5'] = merged['PM2.5'] / merged['Population']

# Top 5 most polluted states per capita

top5\_states = merged.nlargest(5, 'Per\_Capita\_PM2.5')

# Plot

plt.figure(figsize=(10, 6))

plt.bar(top5\_states['State'], top5\_states['Per\_Capita\_PM2.5'], color = 'skyblue')

plt.xlabel('State')

plt.ylabel('Per Capita PM2.5 Exposure')

plt.title('Top 5 Most Polluted States \nin terms of per capita PM2.5 exposure in 2023')

plt.show()

34/182:

import pandas as pd

import matplotlib.pyplot as plt

# Load the datasets

df\_population = pd.read\_csv(r"C:\Users\inees\Downloads\State\_data.csv")

df\_pollution = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

df\_pollution.rename(columns={'state': 'State'}, inplace=True)

# Average PM2.5 for each state

pollution = df\_pollution.groupby('State')['PM2.5'].mean().reset\_index()

# Merge

merged = pd.merge(pollution, df\_population[['State', 'Area (km²)']], on='State', how='inner')

# PM2.5 concentration per km2

merged['PM2.5 per km²'] = merged['PM2.5'] / merged['Area (km²)']

# Plot

plt.figure(figsize=(15, 6))

plt.bar(merged['State'], merged['PM2.5 per km²'])

plt.xticks(rotation=90)

plt.xlabel('State')

plt.ylabel('PM2.5 Concentration per km²')

plt.title('Pollution levels per state relative to their area')

plt.grid(True)

plt.show()

# state with the highest PM2.5 per square kilometer

most\_polluted\_state = merged.loc[merged['PM2.5 per km²'].idxmax(), 'State']

highest\_pm25\_value = merged['PM2.5 per km²'].max()

print(f"State with the highest PM2.5 concentration per km²: {most\_polluted\_state} ({highest\_pm25\_value:.2f})")

34/183:

import pandas as pd

import matplotlib.pyplot as plt

# Load the datasets

df\_population = pd.read\_csv(r"C:\Users\inees\Downloads\State\_data.csv")

df\_pollution = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

df\_pollution.rename(columns={'state': 'State'}, inplace=True)

# Average PM2.5 for each state

pollution = df\_pollution.groupby('State')['PM2.5'].mean().reset\_index()

# Merge

merged = pd.merge(pollution, df\_population[['State', 'Area (km2)']], on='State', how='inner')

# PM2.5 concentration per km2

merged['PM2.5 per km2'] = merged['PM2.5'] / merged['Area (km2)']

# Plot

plt.figure(figsize=(15, 6))

plt.bar(merged['State'], merged['PM2.5 per km2'])

plt.xticks(rotation=90)

plt.xlabel('State')

plt.ylabel('PM2.5 Concentration per km2')

plt.title('Pollution levels per state relative to their area')

plt.grid(True)

plt.show()

# state with the highest PM2.5 per square kilometer

most\_polluted\_state = merged.loc[merged['PM2.5 per km²'].idxmax(), 'State']

highest\_pm25\_value = merged['PM2.5 per km²'].max()

print(f"State with the highest PM2.5 concentration per km²: {most\_polluted\_state} ({highest\_pm25\_value:.2f})")

34/184:

import pandas as pd

import matplotlib.pyplot as plt

# Load the datasets

df\_population = pd.read\_csv(r"C:\Users\inees\Downloads\State\_data.csv")

df\_pollution = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

df\_pollution.rename(columns={'state': 'State'}, inplace=True)

# Average PM2.5 for each state

pollution = df\_pollution.groupby('State')['PM2.5'].mean().reset\_index()

# Merge

merged = pd.merge(pollution, df\_population[['State', 'Area (km2)']], on='State', how='inner')

# PM2.5 concentration per km2

merged['PM2.5 per km2'] = merged['PM2.5'] / merged['Area (km2)']

# Plot

plt.figure(figsize=(15, 6))

plt.bar(merged['State'], merged['PM2.5 per km2'])

plt.xticks(rotation=90)

plt.xlabel('State')

plt.ylabel('PM2.5 Concentration per km2')

plt.title('Pollution levels per state relative to their area')

plt.grid(True)

plt.show()

# state with the highest PM2.5 per square kilometer

most\_polluted\_state = merged.loc[merged['PM2.5 per km²'].idxmax(), 'State']

highest\_pm25\_value = merged['PM2.5 per km2'].max()

print(f"State with the highest PM2.5 concentration per km2: {most\_polluted\_state} ({highest\_pm25\_value:.2f})")

34/185:

import pandas as pd

import matplotlib.pyplot as plt

# Load the datasets

df\_population = pd.read\_csv(r"C:\Users\inees\Downloads\State\_data.csv")

df\_pollution = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

df\_pollution.rename(columns={'state': 'State'}, inplace=True)

# Average PM2.5 for each state

pollution = df\_pollution.groupby('State')['PM2.5'].mean().reset\_index()

# Merge

merged = pd.merge(pollution, df\_population[['State', 'Area (km2)']], on='State', how='inner')

# PM2.5 concentration per km2

merged['PM2.5 per km2'] = merged['PM2.5'] / merged['Area (km2)']

# Plot

plt.figure(figsize=(15, 6))

plt.bar(merged['State'], merged['PM2.5 per km2'])

plt.xticks(rotation=90)

plt.xlabel('State')

plt.ylabel('PM2.5 Concentration per km2')

plt.title('Pollution levels per state relative to their area')

plt.grid(True)

plt.show()

# state with the highest PM2.5 per square kilometer

most\_polluted\_state = merged.loc[merged['PM2.5 per km2'].idxmax(), 'State']

highest\_pm25\_value = merged['PM2.5 per km2'].max()

print(f"State with the highest PM2.5 concentration per km2: {most\_polluted\_state} ({highest\_pm25\_value:.2f})")

34/186:

import pandas as pd

import matplotlib.pyplot as plt

# Load the datasets

df\_population = pd.read\_csv(r"C:\Users\inees\Downloads\State\_data.csv")

df\_pollution = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

df\_pollution.rename(columns={'state': 'State'}, inplace=True)

# Average PM2.5 for each state

pollution = df\_pollution.groupby('State')['PM2.5'].mean().reset\_index()

# Merge

merged = pd.merge(pollution, df\_population[['State', 'Area (km2)']], on='State', how='inner')

# PM2.5 concentration per km2

merged['PM2.5 per km2'] = merged['PM2.5'] / merged['Area (km2)']

# Plot

plt.figure(figsize=(15, 6))

plt.bar(merged['State'], merged['PM2.5 per km2'])

plt.xticks(rotation=90)

plt.xlabel('State')

plt.ylabel('PM2.5 Concentration per km2')

plt.title('Pollution levels per state relative to their area')

plt.grid(True)

plt.show()

# state with the highest PM2.5 per square kilometer

most\_polluted\_state = merged.loc[merged['PM2.5 per km2'].idxmax(), 'State']

highest\_pm25\_value = merged['PM2.5 per km2'].max()

print(f"State with the highest PM2.5 concentration per km2: {most\_polluted\_state} ({highest\_pm25\_value:.3f})")

34/187:

import pandas as pd

import matplotlib.pyplot as plt

# Load the datasets

df\_population = pd.read\_csv(r"C:\Users\inees\Downloads\State\_data.csv")

df\_pollution = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

df\_pollution.rename(columns={'state': 'State'}, inplace=True)

# Average PM2.5 for each state

pollution = df\_pollution.groupby('State')['PM2.5'].mean().reset\_index()

# Merge

merged = pd.merge(pollution, df\_population[['State', 'Area (km2)']], on='State', how='inner')

# PM2.5 concentration per km2

merged['PM2.5 per km2'] = merged['PM2.5'] / merged['Area (km2)']

# Sort

merged\_sorted = merged.sort\_values(by='PM2.5 per km2', ascending=False)

# Plot

plt.figure(figsize=(15, 6))

plt.bar(merged\_sorted['State'], merged\_sorted['PM2.5 per km2'])

plt.xticks(rotation=90)

plt.xlabel('State')

plt.ylabel('PM2.5 Concentration per km2')

plt.title('Pollution levels per state relative to their area')

plt.grid(True)

plt.show()

# State with the highest PM2.5 per km2

most\_polluted\_state = merged\_sorted.iloc[0]['State']

highest\_pm25\_value = merged\_sorted.iloc[0]['PM2.5 per km2']

print(f"State with the highest PM2.5 concentration per km2: {most\_polluted\_state} ({highest\_pm25\_value:.3f})")

34/188:

import pandas as pd

import matplotlib.pyplot as plt

Data = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

State\_data = pd.read\_csv(r"C:\Users\inees\Downloads\State\_data.csv")

Data.rename(columns={'state': 'State'}, inplace=True)

# number of monitoring stations in each state

station\_counts = Data.groupby('State')['station'].count().reset\_index()

station\_counts.rename(columns={'station': 'Number of Stations'}, inplace=True)

# Merge to get area of each state

merged\_data = station\_counts.merge(State\_data, on="State")

# Calculate monitoring stations per km2

merged\_data['Stations per km2'] = merged\_data['Number of Stations'] / merged\_data['Area (km2)']

# Plot

plt.figure(figsize=(12, 6))

plt.bar(merged\_data['State'], merged\_data['Stations per km2'])

plt.xticks(rotation=90)

plt.xlabel("State")

plt.ylabel("Stations per km2")

plt.title("Density of Monitoring Stations in Each State")

plt.show()

# state with the highest stations density

max\_density\_state = merged\_data.loc[merged\_data['Stations per km2'].idxmax(), 'State']

print(f"State with the highest monitoring station density: {max\_density\_state}")

34/189:

import pandas as pd

import matplotlib.pyplot as plt

Data = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

State\_data = pd.read\_csv(r"C:\Users\inees\Downloads\State\_data.csv")

Data.rename(columns={'state': 'State'}, inplace=True)

# number of monitoring stations in each state

station\_counts = Data.groupby('State')['station'].count().reset\_index()

station\_counts.rename(columns={'station': 'Number of Stations'}, inplace=True)

# Merge to get area of each state

merged\_data = station\_counts.merge(State\_data, on="State")

# Calculate monitoring stations per km2

merged\_data['Stations per km2'] = merged\_data['Number of Stations'] / merged\_data['Area (km2)']

# Plot

plt.figure(figsize=(12, 6))

plt.bar(merged\_data['State'], merged\_data['Stations per km2'], color = 'skyblue')

plt.xticks(rotation=90)

plt.xlabel("State")

plt.ylabel("Stations per km2")

plt.title("Density of Monitoring Stations in Each State")

plt.show()

# state with the highest stations density

max\_density\_state = merged\_data.loc[merged\_data['Stations per km2'].idxmax(), 'State']

print(f"State with the highest monitoring station density: {max\_density\_state}")

34/190:

import pandas as pd

df\_pollution = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

df\_population = pd.read\_csv(r"C:\Users\inees\Downloads\State\_data.csv")

df\_pollution.rename(columns={'state': 'State'}, inplace=True)

df\_pollution['Timestamp'] = pd.to\_datetime(df\_pollution['Timestamp'], format='%Y-%m-%d')

# Get the data for 2021

df\_2021 = df\_pollution[df\_pollution['Timestamp'].dt.year == 2021].copy()

# Average PM2.5 for Maharashtra and Madhya Pradesh in 2021

avg\_pm25\_2021 = df\_2021.groupby('State')['PM2.5'].mean().reset\_index()

# Get data for Maharashtra and Madhya Pradesh

avg\_pm25\_2021 = avg\_pm25\_2021[avg\_pm25\_2021['State'].isin(['Maharashtra', 'Madhya Pradesh'])]

print("Average PM2.5 for Maharashtra and Madhya Pradesh in 2021:")

print(avg\_pm25\_2021, '\n')

# Population density for Maharashtra and Madhya Pradesh

df\_mh\_mp = df\_population[df\_population['State'].isin(['Maharashtra', 'Madhya Pradesh'])]

df\_mh\_mp['Population Density'] = df\_mh\_mp['Population'] / df\_mh\_mp['Area (km2)']

# Join population density with PM2.5 levels

merged = pd.merge(avg\_pm25\_2021, df\_mh\_mp[['State', 'Population Density']], on='State')

# Display the results

print("Population Density and Average PM2.5 levels:")

print(merged)

34/191:

import pandas as pd

df\_pollution = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

df\_population = pd.read\_csv(r"C:\Users\inees\Downloads\State\_data.csv")

df\_pollution.rename(columns={'state': 'State'}, inplace=True)

df\_pollution['Timestamp'] = pd.to\_datetime(df\_pollution['Timestamp'], format='%Y-%m-%d')

# Get the data for 2021

df\_2021 = df\_pollution[df\_pollution['Timestamp'].dt.year == 2021].copy()

# Average PM2.5 for Maharashtra and Madhya Pradesh in 2021

avg\_pm25\_2021 = df\_2021.groupby('State')['PM2.5'].mean().reset\_index()

# Get data for Maharashtra and Madhya Pradesh

avg\_pm25\_2021 = avg\_pm25\_2021[avg\_pm25\_2021['State'].isin(['Maharashtra', 'Madhya Pradesh'])]

print("Average PM2.5 for Maharashtra and Madhya Pradesh in 2021:")

print(avg\_pm25\_2021, '\n')

# Population density for Maharashtra and Madhya Pradesh

df\_mh\_mp = df\_population[df\_population['State'].isin(['Maharashtra', 'Madhya Pradesh'])]

df\_mh\_mp['Population Density'] = df\_mh\_mp['Population'] / df\_mh\_mp['Area (km2)'].copy()

# Join population density with PM2.5 levels

merged = pd.merge(avg\_pm25\_2021, df\_mh\_mp[['State', 'Population Density']], on='State')

# Display the results

print("Population Density and Average PM2.5 levels:")

print(merged)

34/192:

import pandas as pd

df\_pollution = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

df\_population = pd.read\_csv(r"C:\Users\inees\Downloads\State\_data.csv")

df\_pollution.rename(columns={'state': 'State'}, inplace=True)

df\_pollution['Timestamp'] = pd.to\_datetime(df\_pollution['Timestamp'], format='%Y-%m-%d')

# Get the data for 2021

df\_2021 = df\_pollution[df\_pollution['Timestamp'].dt.year == 2021].copy()

# Average PM2.5 for Maharashtra and Madhya Pradesh in 2021

avg\_pm25\_2021 = df\_2021.groupby('State')['PM2.5'].mean().reset\_index()

# Get data for Maharashtra and Madhya Pradesh

avg\_pm25\_2021 = avg\_pm25\_2021[avg\_pm25\_2021['State'].isin(['Maharashtra', 'Madhya Pradesh'])]

print("Average PM2.5 for Maharashtra and Madhya Pradesh in 2021:")

print(avg\_pm25\_2021, '\n')

# Population density for Maharashtra and Madhya Pradesh

df\_mh\_mp = df\_population[df\_population['State'].isin(['Maharashtra', 'Madhya Pradesh'])]

df\_mh\_mp['Population Density'] = df\_mh\_mp['Population'] / df\_mh\_mp['Area (km2)']

# Join population density with PM2.5 levels

merged = pd.merge(avg\_pm25\_2021, df\_mh\_mp[['State', 'Population Density']], on='State')

# Display the results

print("Population Density and Average PM2.5 levels:")

print(merged)

34/193:

import pandas as pd

df\_pollution = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

df\_population = pd.read\_csv(r"C:\Users\inees\Downloads\State\_data.csv")

df\_pollution.rename(columns={'state': 'State'}, inplace=True)

df\_pollution['Timestamp'] = pd.to\_datetime(df\_pollution['Timestamp'], format='%Y-%m-%d')

# Get the data for 2021

df\_2021 = df\_pollution[df\_pollution['Timestamp'].dt.year == 2021].copy()

# Average PM2.5 for Maharashtra and Madhya Pradesh in 2021

avg\_pm25\_2021 = df\_2021.groupby('State')['PM2.5'].mean().reset\_index()

# Get data for Maharashtra and Madhya Pradesh

avg\_pm25\_2021 = avg\_pm25\_2021[avg\_pm25\_2021['State'].isin(['Maharashtra', 'Madhya Pradesh'])]

print("Average PM2.5 for Maharashtra and Madhya Pradesh in 2021:")

print(avg\_pm25\_2021, '\n')

# Population density for Maharashtra and Madhya Pradesh

df\_mh\_mp = df\_population[df\_population['State'].isin(['Maharashtra', 'Madhya Pradesh'])].copy()

df\_mh\_mp['Population Density'] = df\_mh\_mp['Population'] / df\_mh\_mp['Area (km2)']

# Join population density with PM2.5 levels

merged = pd.merge(avg\_pm25\_2021, df\_mh\_mp[['State', 'Population Density']], on='State')

# Display the results

print("Population Density and Average PM2.5 levels:")

print(merged)

34/194:

import pandas as pd

df\_pollution = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

df\_population = pd.read\_csv(r"C:\Users\inees\Downloads\State\_data.csv")

df\_pollution.rename(columns={'state': 'State'}, inplace=True)

df\_pollution['Timestamp'] = pd.to\_datetime(df\_pollution['Timestamp'], format='%Y-%m-%d')

# Get the data for 2021

df\_2021 = df\_pollution[df\_pollution['Timestamp'].dt.year == 2021].copy()

# Average PM2.5 for Maharashtra and Madhya Pradesh in 2021

avg\_pm25\_2021 = df\_2021.groupby('State')['PM2.5'].mean().reset\_index()

# Get data for Maharashtra and Madhya Pradesh

avg\_pm25\_2021 = avg\_pm25\_2021[avg\_pm25\_2021['State'].isin(['Maharashtra', 'Madhya Pradesh'])]

# Population density for Maharashtra and Madhya Pradesh

df\_mh\_mp = df\_population[df\_population['State'].isin(['Maharashtra', 'Madhya Pradesh'])].copy()

df\_mh\_mp['Population Density'] = df\_mh\_mp['Population'] / df\_mh\_mp['Area (km2)']

# Join population density with PM2.5 levels

merged = pd.merge(avg\_pm25\_2021, df\_mh\_mp[['State', 'Population Density']], on='State')

# Display the results

print("Population Density and Average PM2.5 levels:")

print(merged)

34/195:

import pandas as pd

df\_pollution = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

df\_population = pd.read\_csv(r"C:\Users\inees\Downloads\State\_data.csv")

df\_pollution.rename(columns={'state': 'State'}, inplace=True)

df\_pollution['Timestamp'] = pd.to\_datetime(df\_pollution['Timestamp'], format='%Y-%m-%d')

# Get the data for 2021

df\_2021 = df\_pollution[df\_pollution['Timestamp'].dt.year == 2021].copy()

# Average PM2.5 for Maharashtra and Madhya Pradesh in 2021

avg\_pm25\_2021 = df\_2021.groupby('State')['PM2.5'].mean().reset\_index()

# Get data for Maharashtra and Madhya Pradesh

avg\_pm25\_2021 = avg\_pm25\_2021[avg\_pm25\_2021['State'].isin(['Maharashtra', 'Madhya Pradesh'])]

# Population density for Maharashtra and Madhya Pradesh

df\_mh\_mp = df\_population[df\_population['State'].isin(['Maharashtra', 'Madhya Pradesh'])].copy()

df\_mh\_mp['Population Density'] = df\_mh\_mp['Population'] / df\_mh\_mp['Area (km2)']

# Join population density with PM2.5 levels

merged = pd.merge(avg\_pm25\_2021, df\_mh\_mp[['State', 'Population Density']], on='State')

# Display the results

print("Population Density and Average PM2.5 levels:")

print(merged)

34/196:

import pandas as pd

df\_pollution = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

df\_population = pd.read\_csv(r"C:\Users\inees\Downloads\State\_data.csv")

df\_pollution.rename(columns={'state': 'State'}, inplace=True)

df\_pollution['Timestamp'] = pd.to\_datetime(df\_pollution['Timestamp'], format='%Y-%m-%d')

# Get the data for 2021

df\_2021 = df\_pollution[df\_pollution['Timestamp'].dt.year == 2021].copy()

# Average PM2.5 for Maharashtra and Madhya Pradesh in 2021

avg\_pm25\_2021 = df\_2021.groupby('State')['PM2.5'].mean().reset\_index()

# Get data for Maharashtra and Madhya Pradesh

avg\_pm25\_2021 = avg\_pm25\_2021[avg\_pm25\_2021['State'].isin(['Maharashtra', 'Madhya Pradesh'])]

# Population density for Maharashtra and Madhya Pradesh

df\_mh\_mp = df\_population[df\_population['State'].isin(['Maharashtra', 'Madhya Pradesh'])].copy()

df\_mh\_mp['Population Density'] = df\_mh\_mp['Population'] / df\_mh\_mp['Area (km2)']

# Join population density with PM2.5 levels

merged = pd.merge(avg\_pm25\_2021, df\_mh\_mp[['State', 'Population Density']], on='State')

# Display the results

print("Population Density and Average PM2.5 levels:\n")

print(merged)

34/197:

import pandas as pd

df\_pollution = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

df\_population = pd.read\_csv(r"C:\Users\inees\Downloads\State\_data.csv")

df\_ncap = pd.read\_csv(r"C:\Users\inees\Downloads\NCAP\_Funding.csv")

df\_pollution.rename(columns={'state': 'State'}, inplace=True)

df\_pollution['Timestamp'] = pd.to\_datetime(df\_pollution['Timestamp'], format='%Y-%m-%d')

# Get the data for 2021

df\_2021 = df\_pollution[df\_pollution['Timestamp'].dt.year == 2021].copy()

# Average PM2.5 for each state in 2021

avg\_pm25\_2021 = df\_2021.groupby('State')['PM2.5'].mean().reset\_index()

# Get the list of states that received NCAP funding

ncap\_states = df\_ncap['State'].tolist()

# Split into states with and without NCAP funding

ncap\_funded = avg\_pm25\_2021[avg\_pm25\_2021['State'].isin(ncap\_states)]

non\_ncap\_funded = avg\_pm25\_2021[~avg\_pm25\_2021['State'].isin(ncap\_states)]

# Calculate the average PM2.5 levels for each group

avg\_pm25\_ncap = ncap\_funded['PM2.5'].mean()

avg\_pm25\_non\_ncap = non\_ncap\_funded['PM2.5'].mean()

print(f"Average PM2.5 for states with NCAP funding in 2021: {avg\_pm25\_ncap:.2f}")

print(f"Average PM2.5 for states without NCAP funding in 2021: {avg\_pm25\_non\_ncap:.2f}")

34/198:

import pandas as pd

import matplotlib.pyplot as plt

# Load datasets

df\_pollution = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

df\_funding = pd.read\_csv(r"C:\Users\inees\Downloads\NCAP\_Funding.csv")

# Rename columns for consistency

df\_pollution.rename(columns={'state': 'State'}, inplace=True)

# Convert timestamps to datetime

df\_pollution['Timestamp'] = pd.to\_datetime(df\_pollution['Timestamp'])

# Filter for Assam

df\_assam = df\_pollution[df\_pollution['State'] == 'Assam'].copy()

# Extract year and month

df\_assam['YearMonth'] = df\_assam['Timestamp'].dt.to\_period('M')

# Monthly average PM2.5 levels

monthly\_pm25 = df\_assam.groupby('YearMonth')['PM2.5'].mean().reset\_index()

monthly\_pm25['YearMonth'] = monthly\_pm25['YearMonth'].astype(str)

# Filter NCAP funding data for Assam

funding\_assam = df\_funding[df\_funding['State'] == 'Assam']

# Merge PM2.5 levels with funding data

merged = pd.merge(monthly\_pm25, funding\_assam, on='YearMonth', how='left')

# Plot time-series graph

fig, ax1 = plt.subplots(figsize=(12, 6))

# PM2.5 levels (primary Y-axis)

ax1.plot(merged['YearMonth'], merged['PM2.5'], marker='o', linestyle='-', color='red', label='PM2.5 Level')

ax1.set\_xlabel('Year-Month')

ax1.set\_ylabel('PM2.5 Level', color='red')

ax1.tick\_params(axis='y', labelcolor='red')

# Funding data (secondary Y-axis)

ax2 = ax1.twinx()

ax2.plot(merged['YearMonth'], merged['Funding'], marker='s', linestyle='--', color='blue', label='NCAP Funding')

ax2.set\_ylabel('NCAP Funding (in crores)', color='blue')

ax2.tick\_params(axis='y', labelcolor='blue')

# Labels & Title

fig.suptitle('PM2.5 Levels vs NCAP Funding in Assam (Time Series)')

ax1.set\_xticklabels(merged['YearMonth'], rotation=45)

ax1.grid()

# Show the plot

plt.show()

34/199:

import pandas as pd

import matplotlib.pyplot as plt

# Load datasets

df\_pollution = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

df\_funding = pd.read\_csv(r"C:\Users\inees\Downloads\NCAP\_Funding.csv")

# Rename column for consistency

df\_pollution.rename(columns={'state': 'State'}, inplace=True)

# Convert timestamp to datetime

df\_pollution['Timestamp'] = pd.to\_datetime(df\_pollution['Timestamp'])

# Filter for Assam

df\_assam = df\_pollution[df\_pollution['State'] == 'Assam'].copy()

# Create Year-Month column

df\_assam['YearMonth'] = df\_assam['Timestamp'].dt.to\_period('M').astype(str)

# Calculate monthly average PM2.5 levels

monthly\_pm25 = df\_assam.groupby('YearMonth')['PM2.5'].mean().reset\_index()

# Process NCAP funding dataset

df\_funding\_assam = df\_funding[df\_funding['State'] == 'Assam']

# Aggregate total funding for Assam per year

funding\_per\_year = df\_funding\_assam.groupby('State')[['Amount released during FY 2019-20',

'Amount released during FY 2020-21',

'Amount released during FY 2021-22']].sum().reset\_index()

# Convert to long format for merging

funding\_per\_year = funding\_per\_year.melt(id\_vars=['State'], var\_name='Year', value\_name='Funding')

# Extract only the year from 'Year' column

funding\_per\_year['Year'] = funding\_per\_year['Year'].str.extract(r'(\d{4})')

# Convert to proper YearMonth format

funding\_per\_year['YearMonth'] = funding\_per\_year['Year'] + '-01'

# Merge PM2.5 levels with NCAP funding

merged = pd.merge(monthly\_pm25, funding\_per\_year[['YearMonth', 'Funding']], on='YearMonth', how='left')

# Plot time-series graph

fig, ax1 = plt.subplots(figsize=(12, 6))

# PM2.5 levels (Primary Y-axis)

ax1.plot(merged['YearMonth'], merged['PM2.5'], marker='o', linestyle='-', color='red', label='PM2.5 Level')

ax1.set\_xlabel('Year-Month')

ax1.set\_ylabel('PM2.5 Level', color='red')

ax1.tick\_params(axis='y', labelcolor='red')

# Funding Data (Secondary Y-axis)

ax2 = ax1.twinx()

ax2.plot(merged['YearMonth'], merged['Funding'], marker='s', linestyle='--', color='blue', label='NCAP Funding')

ax2.set\_ylabel('NCAP Funding (in crores)', color='blue')

ax2.tick\_params(axis='y', labelcolor='blue')

# Labels & Title

fig.suptitle('PM2.5 Levels vs NCAP Funding in Assam (Time Series)')

ax1.set\_xticklabels(merged['YearMonth'], rotation=45)

ax1.grid()

# Show the plot

plt.show()

34/200:

import pandas as pd

import matplotlib.pyplot as plt

# Load datasets

df\_pollution = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

df\_funding = pd.read\_csv(r"C:\Users\inees\Downloads\NCAP\_Funding.csv")

# Rename column for consistency

df\_pollution.rename(columns={'state': 'State'}, inplace=True)

# Convert timestamp to datetime

df\_pollution['Timestamp'] = pd.to\_datetime(df\_pollution['Timestamp'])

# Filter for Assam

df\_assam = df\_pollution[df\_pollution['State'] == 'Assam'].copy()

# Create Year-Month column

df\_assam['YearMonth'] = df\_assam['Timestamp'].dt.to\_period('M').astype(str)

# Calculate monthly average PM2.5 levels

monthly\_pm25 = df\_assam.groupby('YearMonth')['PM2.5'].mean().reset\_index()

# Process NCAP funding dataset

df\_funding\_assam = df\_funding[df\_funding['State'] == 'Assam']

# Aggregate total funding for Assam per year

funding\_per\_year = df\_funding\_assam.groupby('State')[['Amount released during FY 2019-20',

'Amount released during FY 2020-21',

'Amount released during FY 2021-22']].sum().reset\_index()

# Convert to long format for merging

funding\_per\_year = funding\_per\_year.melt(id\_vars=['State'], var\_name='Year', value\_name='Funding')

# Extract only the year from 'Year' column

funding\_per\_year['Year'] = funding\_per\_year['Year'].str.extract(r'(\d{4})')

# Convert to proper YearMonth format

funding\_per\_year['YearMonth'] = funding\_per\_year['Year'] + '-01'

# Convert 'YearMonth' to datetime format

merged['YearMonth'] = pd.to\_datetime(merged['YearMonth'])

# Ensure 'Funding' is numeric and replace NaN with 0

merged['Funding'] = pd.to\_numeric(merged['Funding'], errors='coerce').fillna(0)

# Sort values by time

merged = merged.sort\_values('YearMonth')

# Plot time-series graph

fig, ax1 = plt.subplots(figsize=(12, 6))

# PM2.5 levels (Primary Y-axis)

ax1.plot(merged['YearMonth'], merged['PM2.5'], marker='o', linestyle='-', color='red', label='PM2.5 Level')

ax1.set\_xlabel('Year-Month')

ax1.set\_ylabel('PM2.5 Level', color='red')

ax1.tick\_params(axis='y', labelcolor='red')

# Funding Data (Secondary Y-axis)

ax2 = ax1.twinx()

ax2.plot(merged['YearMonth'], merged['Funding'], marker='s', linestyle='--', color='blue', label='NCAP Funding')

ax2.set\_ylabel('NCAP Funding (in crores)', color='blue')

ax2.tick\_params(axis='y', labelcolor='blue')

# Labels & Title

fig.suptitle('PM2.5 Levels vs NCAP Funding in Assam (Time Series)')

ax1.grid()

plt.xticks(rotation=45)

# Show the plot

plt.show()

34/201:

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

# Sample dataset (Replace this with your actual data)

data = {

'State': ['Assam', 'Tamil Nadu', 'Maharashtra', 'Uttar Pradesh', 'Delhi'],

'Area (sq km)': [78438, 130058, 307713, 243286, 1484],

'Total Funding (in crores)': [50, 75, 120, 90, 60]

}

df = pd.DataFrame(data)

# Set style

sns.set(style="whitegrid")

# Create the scatter plot

plt.figure(figsize=(10, 6))

scatter = sns.scatterplot(

data=df,

x='Area (sq km)',

y='Total Funding (in crores)',

hue='State',

palette='Set1', # Different colors for each state

s=100 # Marker size

)

# Adjusting the legend to be outside the plot

plt.legend(title="State", bbox\_to\_anchor=(1.05, 1), loc='upper left')

# Labels and Title

plt.xlabel("State Area (sq km)")

plt.ylabel("Total NCAP Funding (in crores)")

plt.title("Scatter Plot: Area vs NCAP Funding Received by State")

# Show the plot

plt.show()

34/202:

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

# Sample dataset (Replace this with your actual data)

data = {

'State': ['Assam', 'Tamil Nadu', 'Maharashtra', 'Uttar Pradesh', 'Delhi'],

'Area (sq km)': [78438, 130058, 307713, 243286, 1484],

'Total Funding (in crores)': [50, 75, 120, 90, 60]

}

df = pd.DataFrame(data)

# Set style

sns.set(style="whitegrid")

# Create the scatter plot

plt.figure(figsize=(10, 6))

scatter = sns.scatterplot(

data=df,

x='Area (sq km)',

y='Total Funding (in crores)',

hue='State',

palette='Set1', # Different colors for each state

s=100 # Marker size

)

# Adjusting the legend to be outside the plot

plt.legend(title="State", bbox\_to\_anchor=(1.05, 1), loc='upper left')

# Labels and Title

plt.xlabel("State Area (sq km)")

plt.ylabel("Total NCAP Funding (in crores)")

plt.title("Scatter Plot: Area vs NCAP Funding Received by State")

# Show the plot

plt.show()

34/203:

df = pd.read\_csv(r"C:\Users\inees\Downloads\NCAP\_Funding.csv")

df

34/204:

df = pd.read\_csv(r"C:\Users\inees\Downloads\NCAP\_Funding.csv")

df.head()

34/205:

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

df\_pollution = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

df\_population = pd.read\_csv(r"C:\Users\inees\Downloads\State\_data.csv")

df\_ncap = pd.read\_csv(r"C:\Users\inees\Downloads\NCAP\_Funding.csv")

df\_pollution.rename(columns={'state': 'State'}, inplace=True)

df\_pollution['Timestamp'] = pd.to\_datetime(df\_pollution['Timestamp'], format='%Y-%m-%d')

# # Set style

# sns.set(style="whitegrid")

# Create the scatter plot

plt.figure(figsize=(10, 6))

scatter = sns.scatterplot(

data=df,

x='Area (sq km)',

y='Total Funding (in crores)',

hue='State',

palette='Set1', # Different colors for each state

s=100 # Marker size

)

# Adjusting the legend to be outside the plot

plt.legend(title="State", bbox\_to\_anchor=(1.05, 1), loc='upper left')

# Labels and Title

plt.xlabel("State Area (sq km)")

plt.ylabel("Total NCAP Funding (in crores)")

plt.title("Scatter Plot: Area vs NCAP Funding Received by State")

# Show the plot

plt.show()

34/206:

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

# Load datasets

df\_population = pd.read\_csv(r"C:\Users\inees\Downloads\State\_data.csv")

df\_ncap = pd.read\_csv(r"C:\Users\inees\Downloads\NCAP\_Funding.csv")

# Rename columns for consistency

df\_population.rename(columns={'Area (km2)': 'Area\_sq\_km'}, inplace=True)

df\_ncap.rename(columns={'Total Fund Released': 'Total\_Funding\_Crores'}, inplace=True)

# Group NCAP data by state to get total funding per state

df\_ncap\_statewise = df\_ncap.groupby('State', as\_index=False)['Total\_Funding\_Crores'].sum()

# Merge datasets based on State

df\_merged = pd.merge(df\_population, df\_ncap\_statewise, on="State", how="inner")

# Set style

sns.set(style="whitegrid")

# Create scatter plot

plt.figure(figsize=(12, 7))

scatter = sns.scatterplot(

data=df\_merged,

x='Area\_sq\_km',

y='Total\_Funding\_Crores',

hue='State',

palette='Set1',

s=100

)

# Adjust legend position

plt.legend(title="State", bbox\_to\_anchor=(1.05, 1), loc='upper left')

# Labels and title

plt.xlabel("State Area (sq km)")

plt.ylabel("Total NCAP Funding (in crores)")

plt.title("State-wise Area vs NCAP Funding Received")

# Show plot

plt.show()

34/207:

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

# Load datasets

df\_population = pd.read\_csv(r"C:\Users\inees\Downloads\State\_data.csv")

df\_ncap = pd.read\_csv(r"C:\Users\inees\Downloads\NCAP\_Funding.csv")

# Rename columns for consistency

df\_population.rename(columns={'Area (km2)': 'Area\_sq\_km'}, inplace=True)

df\_ncap.rename(columns={'Total Fund Released': 'Total\_Funding\_Crores'}, inplace=True)

# Group NCAP data by state to get total funding per state

df\_ncap\_statewise = df\_ncap.groupby('State', as\_index=False)['Total\_Funding\_Crores'].sum()

# Merge datasets based on State

df\_merged = pd.merge(df\_population, df\_ncap\_statewise, on="State", how="inner")

# Set style

sns.set(style="whitegrid")

# Create scatter plot

plt.figure(figsize=(12, 7))

scatter = sns.scatterplot(

data=df\_merged,

x='Area\_sq\_km',

y='Total\_Funding\_Crores',

hue='State',

palette='Set1',

s=100

)

# Adjust legend position

plt.legend(title="State", bbox\_to\_anchor=(1.05, 1), loc='upper left')

# Labels and title

plt.xlabel("State Area (sq km)")

plt.ylabel("Total NCAP Funding (in crores)")

plt.title("State-wise Area vs NCAP Funding Received")

# Show plot

plt.show()

34/208:

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

# Load datasets

df\_population = pd.read\_csv(r"C:\Users\inees\Downloads\State\_data.csv")

df\_ncap = pd.read\_csv(r"C:\Users\inees\Downloads\NCAP\_Funding.csv")

# Rename columns for consistency

df\_population.rename(columns={'Area (km²)': 'Area\_sq\_km'}, inplace=True)

df\_ncap.rename(columns={'Total Fund Released': 'Total\_Funding\_Crores'}, inplace=True)

# Group NCAP data by state to get total funding per state

df\_ncap\_statewise = df\_ncap.groupby('State', as\_index=False)['Total\_Funding\_Crores'].sum()

# Merge datasets based on State

df\_merged = pd.merge(df\_population, df\_ncap\_statewise, on="State", how="inner")

# Set style

sns.set(style="whitegrid")

# Create scatter plot

plt.figure(figsize=(14, 8)) # Increased figure size for better clarity

scatter = sns.scatterplot(

data=df\_merged,

x='Area\_sq\_km',

y='Total\_Funding\_Crores',

hue='State',

palette='Set1',

s=100

)

# Adjust legend position

plt.legend(title="State", bbox\_to\_anchor=(1.05, 1), loc='upper left')

# Labels and title

plt.xlabel("State Area (sq km)")

plt.ylabel("Total NCAP Funding (in crores)")

plt.title("State-wise Area vs NCAP Funding Received")

# Show plot

plt.show()

34/209:

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

# Load datasets

df\_population = pd.read\_csv(r"C:\Users\inees\Downloads\State\_data.csv")

df\_ncap = pd.read\_csv(r"C:\Users\inees\Downloads\NCAP\_Funding.csv")

# Rename columns for consistency

df\_population.rename(columns={'Area (km²)': 'Area\_sq\_km'}, inplace=True)

df\_ncap.rename(columns={'Total Fund Released': 'Total\_Funding\_Crores'}, inplace=True)

# Group NCAP data by state to get total funding per state

df\_ncap\_statewise = df\_ncap.groupby('State', as\_index=False)['Total\_Funding\_Crores'].sum()

# Merge datasets based on State

df\_merged = pd.merge(df\_population, df\_ncap\_statewise, on="State", how="inner")

# Set style

sns.set(style="whitegrid")

# Create scatter plot

plt.figure(figsize=(14, 8)) # Increased figure size for better clarity

scatter = sns.scatterplot(

data=df\_merged,

x='Area\_sq\_km',

y='Total\_Funding\_Crores',

hue='State',

palette='Set1',

s=100

)

# Adjust legend position

plt.legend(title="State", bbox\_to\_anchor=(1.05, 1), loc='upper left')

# Labels and title

plt.xlabel("State Area (sq km)")

plt.ylabel("Total NCAP Funding (in crores)")

plt.title("State-wise Area vs NCAP Funding Received")

# Show plot

plt.show()

34/210:

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

# Load datasets

df\_population = pd.read\_csv(r"C:\Users\inees\Downloads\State\_data.csv")

df\_ncap = pd.read\_csv(r"C:\Users\inees\Downloads\NCAP\_Funding.csv")

# Rename columns for consistency

df\_population.rename(columns={'Area (km²)': 'Area\_sq\_km'}, inplace=True)

df\_ncap.rename(columns={'Total Fund Released': 'Total\_Funding\_Crores'}, inplace=True)

# Group NCAP data by state to get total funding per state

df\_ncap\_statewise = df\_ncap.groupby('State', as\_index=False)['Total\_Funding\_Crores'].sum()

# Merge datasets based on State

df\_merged = pd.merge(df\_population, df\_ncap\_statewise, on="State", how="inner")

# Set style

sns.set(style="whitegrid")

# Create scatter plot

plt.figure(figsize=(14, 8)) # Increased figure size for better clarity

scatter = sns.scatterplot(

data=df\_merged,

x='Area\_sq\_km',

y='Total\_Funding\_Crores',

hue='State',

palette='Set1',

s=100

)

# Adjust legend position

plt.legend(title="State", bbox\_to\_anchor=(1.05, 1), loc='upper left')

# Labels and title

plt.xlabel("State Area (sq km)")

plt.ylabel("Total NCAP Funding (in crores)")

plt.title("State-wise Area vs NCAP Funding Received")

# Show plot

plt.show()

34/211:

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

# Load datasets

df\_population = pd.read\_csv(r"C:\Users\inees\Downloads\State\_data.csv")

df\_ncap = pd.read\_csv(r"C:\Users\inees\Downloads\NCAP\_Funding.csv")

# Rename columns for consistency

df\_population.rename(columns={'Area (km2)': 'Area\_sq\_km'}, inplace=True)

df\_ncap.rename(columns={'Total Fund Released': 'Total\_Funding\_Crores'}, inplace=True)

# Group NCAP data by state to get total funding per state

df\_ncap\_statewise = df\_ncap.groupby('State', as\_index=False)['Total\_Funding\_Crores'].sum()

# Merge datasets based on State

df\_merged = pd.merge(df\_population, df\_ncap\_statewise, on="State", how="inner")

# Set style

sns.set(style="whitegrid")

# Create scatter plot

plt.figure(figsize=(14, 8)) # Increased figure size for better clarity

scatter = sns.scatterplot(

data=df\_merged,

x='Area\_sq\_km',

y='Total\_Funding\_Crores',

hue='State',

palette='Set1',

s=100

)

# Adjust legend position

plt.legend(title="State", bbox\_to\_anchor=(1.05, 1), loc='upper left')

# Labels and title

plt.xlabel("State Area (sq km)")

plt.ylabel("Total NCAP Funding (in crores)")

plt.title("State-wise Area vs NCAP Funding Received")

# Show plot

plt.show()

34/212:

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

# Load datasets

df\_population = pd.read\_csv(r"C:\Users\inees\Downloads\State\_data.csv")

df\_pollution.rename(columns={'state': 'State'}, inplace=True)

df\_ncap = pd.read\_csv(r"C:\Users\inees\Downloads\NCAP\_Funding.csv")

# Rename columns for consistency

df\_population.rename(columns={'Area (km²)': 'Area\_sq\_km'}, inplace=True)

df\_ncap.rename(columns={'Total Fund Released': 'Total\_Funding\_Crores'}, inplace=True)

# Group NCAP data by state to get total funding per state

df\_ncap\_statewise = df\_ncap.groupby('State', as\_index=False)['Total\_Funding\_Crores'].sum()

# Merge datasets based on State

df\_merged = pd.merge(df\_population, df\_ncap\_statewise, on="State", how="inner")

# Set style

sns.set(style="whitegrid")

# Create scatter plot

plt.figure(figsize=(14, 8)) # Increased figure size for better clarity

scatter = sns.scatterplot(

data=df\_merged,

x='Area\_sq\_km',

y='Total\_Funding\_Crores',

hue='State',

palette='Set1',

s=100

)

# Adjust legend position

plt.legend(title="State", bbox\_to\_anchor=(1.05, 1), loc='upper left')

# Labels and title

plt.xlabel("State Area (sq km)")

plt.ylabel("Total NCAP Funding (in crores)")

plt.title("State-wise Area vs NCAP Funding Received")

# Show plot

plt.show()

34/213:

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

# Load datasets

df\_population = pd.read\_csv(r"C:\Users\inees\Downloads\State\_data.csv")

df\_ncap = pd.read\_csv(r"C:\Users\inees\Downloads\NCAP\_Funding.csv")

# Rename columns for consistency

df\_population.rename(columns={'Area (km2)': 'Area\_sq\_km'}, inplace=True)

df\_ncap.rename(columns={'Total Fund Released': 'Total\_Funding\_Crores'}, inplace=True)

# Group NCAP data by state to get total funding per state

df\_ncap\_statewise = df\_ncap.groupby('State', as\_index=False)['Total\_Funding\_Crores'].sum()

# Merge datasets based on State

df\_merged = pd.merge(df\_population, df\_ncap\_statewise, on="State", how="inner")

# Set style

sns.set(style="whitegrid")

# Create scatter plot

plt.figure(figsize=(14, 8)) # Increased figure size for better clarity

scatter = sns.scatterplot(

data=df\_merged,

x='Area\_sq\_km',

y='Total\_Funding\_Crores',

hue='State',

palette='Set1',

s=100

)

# Adjust legend position

plt.legend(title="State", bbox\_to\_anchor=(1.05, 1), loc='upper left')

# Labels and title

plt.xlabel("State Area (sq km)")

plt.ylabel("Total NCAP Funding (in crores)")

plt.title("State-wise Area vs NCAP Funding Received")

# Show plot

plt.show()

34/214:

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

# Load datasets

df\_population = pd.read\_csv(r"C:\Users\inees\Downloads\State\_data.csv")

df\_ncap = pd.read\_csv(r"C:\Users\inees\Downloads\NCAP\_Funding.csv")

# Rename columns for consistency

df\_population.rename(columns={'Area (km2)': 'Area\_sq\_km'}, inplace=True)

df\_ncap.rename(columns={'Total fund released': 'Total\_Funding\_Crores'}, inplace=True)

# Group NCAP data by state to get total funding per state

df\_ncap\_statewise = df\_ncap.groupby('State', as\_index=False)['Total\_Funding\_Crores'].sum()

# Merge datasets based on State

df\_merged = pd.merge(df\_population, df\_ncap\_statewise, on="State", how="inner")

# Set style

sns.set(style="whitegrid")

# Create scatter plot

plt.figure(figsize=(14, 8)) # Adjusted size for better clarity

scatter = sns.scatterplot(

data=df\_merged,

x='Area\_sq\_km',

y='Total\_Funding\_Crores',

hue='State',

palette='Set1',

s=100

)

# Adjust legend position outside the plot

plt.legend(title="State", bbox\_to\_anchor=(1.05, 1), loc='upper left')

# Labels and title

plt.xlabel("State Area (sq km)")

plt.ylabel("Total NCAP Funding (in crores)")

plt.title("State-wise Area vs NCAP Funding Received")

# Show plot

plt.tight\_layout() # Ensure the legend fits outside the plot

plt.show()

34/215:

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

# Load datasets

df\_population = pd.read\_csv(r"C:\Users\inees\Downloads\State\_data.csv")

df\_ncap = pd.read\_csv(r"C:\Users\inees\Downloads\NCAP\_Funding.csv")

# Rename columns for consistency

df\_population.rename(columns={'Area (km2)': 'Area\_sq\_km'}, inplace=True)

df\_ncap.rename(columns={'Total fund released': 'Total\_Funding\_Crores'}, inplace=True)

# Group NCAP data by state to get total funding per state

df\_ncap\_statewise = df\_ncap.groupby('State', as\_index=False)['Total\_Funding\_Crores'].sum()

# Merge datasets based on State

df\_merged = pd.merge(df\_population, df\_ncap\_statewise, on="State", how="inner")

# Set style

sns.set(style="whitegrid")

# Create scatter plot

plt.figure(figsize=(14, 8)) # Adjusted size for better clarity

scatter = sns.scatterplot(

data=df\_merged,

x='Area\_sq\_km',

y='Total\_Funding\_Crores',

hue='State',

palette=sns.color\_palette("tab20", len(df\_merged['State'].unique())),

s=100

)

# Adjust legend position outside the plot

plt.legend(title="State", bbox\_to\_anchor=(1.05, 1), loc='upper left')

# Labels and title

plt.xlabel("State Area (sq km)")

plt.ylabel("Total NCAP Funding (in crores)")

plt.title("State-wise Area vs NCAP Funding Received")

# Show plot

plt.tight\_layout() # Ensure the legend fits outside the plot

plt.show()

35/1:

#Grouping the pollution data by state and finding the mean PM2.5 levels of each state, resetting index so that i can still access the 'state' column

Data5\_1 = (Data.groupby(['state'])['PM2.5'].mean()).reset\_index()

#Getting only the states in State\_data whose PM2.5 levels is available (precautionary step) and then sorting by state name

states = (State\_data[State\_data['State'].isin(Data5\_1['state'])]).sort\_values(by='State')

#Finding the PM2.5 level/area

PM25byArea = Data5\_1['PM2.5']/states['Area (km2)']

#Plotting the graph

plt.bar(states['State'], PM25byArea)

plt.ylabel("PM2.5 per sq. km")

plt.xlabel("State")

plt.title("Bar Plot of State vs PM2.5 per sq.km")

plt.figure(figsize=(10, 6))

plt.tight\_layout()

plt.rotation(90)

plt.show()

34/216:

import pandas as pd

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

df['Timestamp'] = pd.to\_datetime(df['Timestamp'], format='%Y-%m-%d')

# Get data for 2023

df\_2023 = df[df['Timestamp'].dt.year == 2023].copy()

# Group by City and calculate avg PM2.5

city\_avg\_pm25 = df\_2023.groupby('City')['PM2.5'].mean()

# Most polluted city

most\_polluted\_city = city\_avg\_pm25.idxmax()

highest\_avg\_pm25 = city\_avg\_pm25.max()

# Get data for the most polluted city

df\_most\_polluted = df\_2023[df\_2023['City'] == most\_polluted\_city]

# Count the number of days with PM2.5 > 300

hazardous\_days = (df\_most\_polluted['PM2.5'] > 300).sum()

# Total number of days recorded for this city

total\_days = df\_most\_polluted['Timestamp'].count()

# Calculate percentage of hazardous days

percentage\_hazardous\_days = (hazardous\_days / total\_days) \* 100

print(f"Most Polluted City in 2023: {most\_polluted\_city}")

print(f"Highest Average PM2.5: {highest\_avg\_pm25:.3f} µg/m³")

print(f"Total Days Recorded for{most\_polluted\_city}: {total\_days}")

print(f"No. of hazardous Days: {hazardous\_days}")

print(f"Percentage of Hazardous Days: {percentage\_hazardous\_days:.3f}%")

34/217:

import pandas as pd

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

df['Timestamp'] = pd.to\_datetime(df['Timestamp'], format='%Y-%m-%d')

# Get data for 2023

df\_2023 = df[df['Timestamp'].dt.year == 2023].copy()

# Group by City and calculate avg PM2.5

city\_avg\_pm25 = df\_2023.groupby('city')['PM2.5'].mean()

# Most polluted city

most\_polluted\_city = city\_avg\_pm25.idxmax()

highest\_avg\_pm25 = city\_avg\_pm25.max()

# Get data for the most polluted city

df\_most\_polluted = df\_2023[df\_2023['city'] == most\_polluted\_city]

# Count the number of days with PM2.5 > 300

hazardous\_days = (df\_most\_polluted['PM2.5'] > 300).sum()

# Total number of days recorded for this city

total\_days = df\_most\_polluted['Timestamp'].count()

# Calculate percentage of hazardous days

percentage\_hazardous\_days = (hazardous\_days / total\_days) \* 100

print(f"Most Polluted City in 2023: {most\_polluted\_city}")

print(f"Highest Average PM2.5: {highest\_avg\_pm25:.3f} µg/m³")

print(f"Total Days Recorded for{most\_polluted\_city}: {total\_days}")

print(f"No. of hazardous Days: {hazardous\_days}")

print(f"Percentage of Hazardous Days: {percentage\_hazardous\_days:.3f}%")

34/218:

Data7\_1=df[["Timestamp","city","PM2.5"]].copy()

Data7\_1["Timestamp"]=pd.to\_datetime(Data7\_1["Timestamp"])

Data7\_1=Data7\_1.set\_index("Timestamp").loc['2023-01-01':'2023-12-31']

Most\_Polluted=Data7\_1.groupby(["city"]).mean()

index1=Most\_Polluted.loc[(Most\_Polluted["PM2.5"]==Most\_Polluted["PM2.5"].max())]

City=index1.index[0]

Data7\_1=Data7\_1.loc[(Data7\_1["city"]==City)]

Percentage=len(Data7\_1[Data7\_1["PM2.5"]>300])/len(Data7\_1[Data7\_1["PM2.5"]>0])\*100

Percentage

34/219:

import pandas as pd

# Load dataset

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

# Convert Timestamp to datetime format

df["Timestamp"] = pd.to\_datetime(df["Timestamp"])

# Select necessary columns

Data7\_1 = df[["Timestamp", "City", "PM2.5"]].copy()

# Filter data for 2023

Data7\_1 = Data7\_1.set\_index("Timestamp").loc["2023-01-01":"2023-12-31"]

# Compute the most polluted city (highest average PM2.5)

Most\_Polluted = Data7\_1.groupby("City")["PM2.5"].mean()

City = Most\_Polluted.idxmax() # City with highest PM2.5

# Filter data for the most polluted city

Data7\_1 = Data7\_1[Data7\_1["City"] == City]

# Calculate percentage of days with PM2.5 > 300 µg/m³

hazardous\_days = (Data7\_1["PM2.5"] > 300).sum()

total\_days = Data7\_1["PM2.5"].count() # Count only non-null days

Percentage = (hazardous\_days / total\_days) \* 100

# Print results

print(f"Most Polluted City in 2023: {City}")

print(f"Percentage of Hazardous Days (PM2.5 > 300 µg/m³): {Percentage:.2f}%")

34/220:

import pandas as pd

# Load dataset

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

# Convert Timestamp to datetime format

df["Timestamp"] = pd.to\_datetime(df["Timestamp"])

# Select necessary columns

Data7\_1 = df[["Timestamp", "city", "PM2.5"]].copy()

# Filter data for 2023

Data7\_1 = Data7\_1.set\_index("Timestamp").loc["2023-01-01":"2023-12-31"]

# Compute the most polluted city (highest average PM2.5)

Most\_Polluted = Data7\_1.groupby("city")["PM2.5"].mean()

City = Most\_Polluted.idxmax() # City with highest PM2.5

# Filter data for the most polluted city

Data7\_1 = Data7\_1[Data7\_1["city"] == City]

# Calculate percentage of days with PM2.5 > 300 µg/m³

hazardous\_days = (Data7\_1["PM2.5"] > 300).sum()

total\_days = Data7\_1["PM2.5"].count() # Count only non-null days

Percentage = (hazardous\_days / total\_days) \* 100

# Print results

print(f"Most Polluted City in 2023: {City}")

print(f"Percentage of Hazardous Days (PM2.5 > 300 µg/m³): {Percentage:.2f}%")

34/221:

import pandas as pd

# Load dataset

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

# Convert Timestamp to datetime format

df["Timestamp"] = pd.to\_datetime(df["Timestamp"])

# Select necessary columns

Data7\_1 = df[["Timestamp", "city", "PM2.5"]].copy()

# Filter data for 2023

Data7\_1 = Data7\_1.set\_index("Timestamp").loc["2023-01-01":"2023-12-31"]

# Compute the most polluted city (highest average PM2.5)

Most\_Polluted = Data7\_1.groupby("city")["PM2.5"].mean()

City = Most\_Polluted.idxmax() # City with highest PM2.5

# Filter data for the most polluted city

Data7\_1 = Data7\_1[Data7\_1["city"] == City]

# Calculate percentage of days with PM2.5 > 300 µg/m³

hazardous\_days = (Data7\_1["PM2.5"] > 300).sum()

total\_days = Data7\_1["PM2.5"].count() # Count only non-null days

print(total\_days)

Percentage = (hazardous\_days / total\_days) \* 100

# Print results

print(f"Most Polluted City in 2023: {City}")

print(f"Percentage of Hazardous Days (PM2.5 > 300 µg/m³): {Percentage:.2f}%")

34/222:

import pandas as pd

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

df['Timestamp'] = pd.to\_datetime(df['Timestamp'], format='%Y-%m-%d')

# Get data for 2023

df\_2023 = df[df['Timestamp'].dt.year == 2023].copy()

# Group by City and calculate avg PM2.5

city\_avg\_pm25 = df\_2023.groupby('city')['PM2.5'].mean()

# Most polluted city

most\_polluted\_city = city\_avg\_pm25.idxmax()

highest\_avg\_pm25 = city\_avg\_pm25.max()

# Get data for the most polluted city

df\_most\_polluted = df\_2023[df\_2023['city'] == most\_polluted\_city]

# Count the number of days with PM2.5 > 300

hazardous\_days = (df\_most\_polluted['PM2.5'] > 300).sum()

# Total number of days recorded for this city

total\_days = df\_most\_polluted['Timestamp'].nunique()

# Calculate percentage of hazardous days

percentage\_hazardous\_days = (hazardous\_days / total\_days) \* 100

print(f"Most Polluted City in 2023: {most\_polluted\_city}")

print(f"Highest Average PM2.5: {highest\_avg\_pm25:.3f} µg/m³")

print(f"Total Days Recorded for {most\_polluted\_city}: {total\_days}")

print(f"No. of hazardous Days: {hazardous\_days}")

print(f"Percentage of Hazardous Days: {percentage\_hazardous\_days:.3f}%")

34/223:

import pandas as pd

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

df['Timestamp'] = pd.to\_datetime(df['Timestamp'], format='%Y-%m-%d')

# Get data for 2023

df\_2023 = df[df['Timestamp'].dt.year == 2023].copy()

# Group by City and calculate avg PM2.5

city\_avg\_pm25 = df\_2023.groupby('city')['PM2.5'].mean()

# Most polluted city

most\_polluted\_city = city\_avg\_pm25.idxmax()

highest\_avg\_pm25 = city\_avg\_pm25.max()

# Get data for the most polluted city

df\_most\_polluted = df\_2023[df\_2023['city'] == most\_polluted\_city]

# Count the number of days with PM2.5 > 300

hazardous\_days = (df\_most\_polluted['PM2.5'] > 300).sum()

# Total number of days recorded for this city

total\_days = df\_most\_polluted['Timestamp'].nunique()

# Calculate percentage of hazardous days

percentage\_hazardous\_days = (hazardous\_days / total\_days) \* 100

print(f"Most Polluted City in 2023: {most\_polluted\_city}")

print(f"Highest Average PM2.5: {highest\_avg\_pm25:.3f} µg/m³")

print(f"Total Days Recorded for {most\_polluted\_city}: {total\_days}")

print(f"No. of hazardous Days: {hazardous\_days}")

print(f"Percentage of Hazardous Days: {percentage\_hazardous\_days:.3f}%")

34/224:

import pandas as pd

# Load dataset

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

# Convert Timestamp to datetime format

df["Timestamp"] = pd.to\_datetime(df["Timestamp"])

# Select necessary columns

Data7\_1 = df[["Timestamp", "city", "PM2.5"]].copy()

# Filter data for 2023

Data7\_1 = Data7\_1.set\_index("Timestamp").loc["2023-01-01":"2023-12-31"]

# Compute the most polluted city (highest average PM2.5)

Most\_Polluted = Data7\_1.groupby("city")["PM2.5"].mean()

City = Most\_Polluted.idxmax() # City with highest PM2.5

# Filter data for the most polluted city

Data7\_1 = Data7\_1[Data7\_1["city"] == City]

# Calculate percentage of days with PM2.5 > 300 µg/m³

hazardous\_days = (Data7\_1["PM2.5"] > 300).sum()

total\_days = Data7\_1["PM2.5"].count() # Count only non-null days

print(total\_days)

Percentage = (hazardous\_days / total\_days) \* 100

# Print results

print(f"Most Polluted City in 2023: {City}")

print(f"Percentage of Hazardous Days (PM2.5 > 300 µg/m³): {Percentage:.2f}%")

3/13

34/225:

import pandas as pd

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

df['Timestamp'] = pd.to\_datetime(df['Timestamp'], format='%Y-%m-%d')

# Get data for 2023

df\_2023 = df[df['Timestamp'].dt.year == 2023].copy()

# Group by City and calculate avg PM2.5

city\_avg\_pm25 = df\_2023.groupby('city')['PM2.5'].mean()

# Most polluted city

most\_polluted\_city = city\_avg\_pm25.idxmax()

highest\_avg\_pm25 = city\_avg\_pm25.max()

# Get data for the most polluted city

df\_most\_polluted = df\_2023[df\_2023['city'] == most\_polluted\_city]

print(df\_most\_polluted)

# Count the number of days with PM2.5 > 300

hazardous\_days = (df\_most\_polluted['PM2.5'] > 300).sum()

# Total number of days recorded for this city

total\_days = df\_most\_polluted['Timestamp'].nunique()

# Calculate percentage of hazardous days

percentage\_hazardous\_days = (hazardous\_days / total\_days) \* 100

print(f"Most Polluted City in 2023: {most\_polluted\_city}")

print(f"Highest Average PM2.5: {highest\_avg\_pm25:.3f} µg/m³")

print(f"Total Days Recorded for {most\_polluted\_city}: {total\_days}")

print(f"No. of hazardous Days: {hazardous\_days}")

print(f"Percentage of Hazardous Days: {percentage\_hazardous\_days:.3f}%")

34/226:

import pandas as pd

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

df['Timestamp'] = pd.to\_datetime(df['Timestamp'], format='%Y-%m-%d')

# Get data for 2023

df\_2023 = df[df['Timestamp'].dt.year == 2023].copy()

# Group by City and calculate avg PM2.5

city\_avg\_pm25 = df\_2023.groupby('city')['PM2.5'].mean()

# Most polluted city

most\_polluted\_city = city\_avg\_pm25.idxmax()

highest\_avg\_pm25 = city\_avg\_pm25.max()

# Get data for the most polluted city

df\_most\_polluted = df\_2023[df\_2023['city'] == most\_polluted\_city]

df\_most\_polluted = df\_most\_polluted.dropna(subset = ["PM2.5"], inplace = True)

# Count the number of days with PM2.5 > 300

hazardous\_days = (df\_most\_polluted['PM2.5'] > 300).sum()

# Total number of days recorded for this city

total\_days = df\_most\_polluted['Timestamp'].nunique()

# Calculate percentage of hazardous days

percentage\_hazardous\_days = (hazardous\_days / total\_days) \* 100

print(f"Most Polluted City in 2023: {most\_polluted\_city}")

print(f"Highest Average PM2.5: {highest\_avg\_pm25:.3f} µg/m³")

print(f"Total Days Recorded for {most\_polluted\_city}: {total\_days}")

print(f"No. of hazardous Days: {hazardous\_days}")

print(f"Percentage of Hazardous Days: {percentage\_hazardous\_days:.3f}%")

34/227:

import pandas as pd

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

df['Timestamp'] = pd.to\_datetime(df['Timestamp'], format='%Y-%m-%d')

# Get data for 2023

df\_2023 = df[df['Timestamp'].dt.year == 2023].copy()

# Group by City and calculate avg PM2.5

city\_avg\_pm25 = df\_2023.groupby('city')['PM2.5'].mean()

# Most polluted city

most\_polluted\_city = city\_avg\_pm25.idxmax()

highest\_avg\_pm25 = city\_avg\_pm25.max()

# Get data for the most polluted city

df\_most\_polluted = df\_2023[df\_2023['city'] == most\_polluted\_city]

df\_most\_polluted = df\_most\_polluted.dropna(subset = ["PM2.5"]).copy()

# Count the number of days with PM2.5 > 300

hazardous\_days = (df\_most\_polluted['PM2.5'] > 300).sum()

# Total number of days recorded for this city

total\_days = df\_most\_polluted['Timestamp'].nunique()

# Calculate percentage of hazardous days

percentage\_hazardous\_days = (hazardous\_days / total\_days) \* 100

print(f"Most Polluted City in 2023: {most\_polluted\_city}")

print(f"Highest Average PM2.5: {highest\_avg\_pm25:.3f} µg/m³")

print(f"Total Days Recorded for {most\_polluted\_city}: {total\_days}")

print(f"No. of hazardous Days: {hazardous\_days}")

print(f"Percentage of Hazardous Days: {percentage\_hazardous\_days:.3f}%")

34/228:

import pandas as pd

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

df['Timestamp'] = pd.to\_datetime(df['Timestamp'], format='%Y-%m-%d')

# Get data for 2023

df\_2023 = df[df['Timestamp'].dt.year == 2023].copy()

# Group by City and calculate avg PM2.5

city\_avg\_pm25 = df\_2023.groupby('city')['PM2.5'].mean()

# Most polluted city

most\_polluted\_city = city\_avg\_pm25.idxmax()

highest\_avg\_pm25 = city\_avg\_pm25.max()

# Get data for the most polluted city

df\_most\_polluted = df\_2023[df\_2023['city'] == most\_polluted\_city]

df\_most\_polluted = df\_most\_polluted.dropna(subset = ["PM2.5"]).copy()

# Count the number of days with PM2.5 > 300

hazardous\_days = (df\_most\_polluted['PM2.5'] > 300).sum()

# Total number of days recorded for this city

total\_days = df\_most\_polluted['Timestamp'].nunique()

# Calculate percentage of hazardous days

percentage\_hazardous\_days = (hazardous\_days / total\_days) \* 100

print(f"Most Polluted City in 2023: {most\_polluted\_city}")

print(f"Percentage of Hazardous Days: {percentage\_hazardous\_days:.3f}%")

34/229:

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

path1 = r"C:\Users\inees\Downloads\Data.csv" # Data.csv

path2 = r"C:\Users\inees\Downloads\State\_data.csv" # State\_data.csv

path3 = r"C:\Users\inees\Downloads\NCAP\_Funding.csv" # NCAP\_Funding.csv

34/230:

df = pd.read\_csv(path1)

df.rename(columns={'state': 'State'}, inplace=True)

df.head()

34/231:

import pandas as pd

df = pd.read\_csv(path1)

# Avg PM2.5 for each state

df\_dropped = df.dropna(subset = ["PM2.5"]) # Drop rows w PM2.5 missing

avg\_pm = df\_dropped.groupby( by = "state")["PM2.5"].mean()

idxm = avg\_pm.idxmax() # Get the state w highest avg

m = avg\_pm.max() # Get the value of the hight avg PM2.5 Conc

print(f"{idxm} has the highest average PM2.5 concentration (of {m:.3f}) across all stations and across all years.")

34/232:

import pandas as pd

df = pd.read\_csv(path1)

# Get rows for year 2023

df["Timestamp"] = pd.to\_datetime(df["Timestamp"], format="%Y-%m-%d")

df\_2023 = df[df["Timestamp"].dt.year == 2023]

# Drop rows w PM2.5 missing

df\_2023 = df\_2023.dropna(subset = ["PM2.5"])

# hazardous PM2.5 levels

hazardous\_data = df\_2023[df\_2023["PM2.5"] > 300]

hazardous\_days\_per\_state = hazardous\_data.groupby("state")["Timestamp"].count()

# Find the state with the highest number of hazardous days

most\_hazardous\_state = hazardous\_days\_per\_state.idxmax()

most\_hazardous\_days = hazardous\_days\_per\_state.max()

print(f"The state with the most days having hazardous PM2.5 levels in 2023 is {most\_hazardous\_state} with {most\_hazardous\_days} days.")

34/233:

import pandas as pd

df = pd.read\_csv(path1)

# Get rows for year 2023

df["Timestamp"] = pd.to\_datetime(df["Timestamp"], format="%Y-%m-%d") # Convert Timestamp col to datetime format

df\_2023 = df[df["Timestamp"].dt.year == 2023]

# Drop rows w PM2.5 missing

df\_2023 = df\_2023.dropna(subset = ["PM2.5"])

# Calc variation (std dev) of PM2.5 per state

var\_per\_state = df\_2023.groupby("state")["PM2.5"].std()

# Find the state with the highest variability

most\_var\_state = var\_per\_state.idxmax()

highest\_var = var\_per\_state.max()

print(f"The state with the highest variability in PM2.5 levels in 2023 is {most\_var\_state} with a standard deviation of {highest\_var:.3f}.")

34/234:

import pandas as pd

df = pd.read\_csv(path1)

# Get rows for year 2020-21

df["Timestamp"] = pd.to\_datetime(df["Timestamp"], format="%Y-%m-%d") # Convert Timestamp col to datetime format

df\_covid = df[(df["Timestamp"].dt.year >= 2020) & (df["Timestamp"].dt.year <= 2021)]

# Avg PM2.5 for each state

df\_dropped = df\_covid.dropna(subset = ["PM2.5"]) # Drop rows w PM2.5 missing

avg\_pm = df\_dropped.groupby( by = "state")["PM2.5"].mean()

idxm = avg\_pm.idxmin() # Get the state w min avg

m = avg\_pm.min() # Get the value of the min avg PM2.5 Conc

print(f"{idxm} has the lowest average PM2.5 concentration (of {m:.3f}) across all stations and across years 2020-2021.")

34/235:

import pandas as pd

df = pd.read\_csv(path1)

df["Timestamp"] = pd.to\_datetime(df["Timestamp"], format = "%Y-%m-%d") # Convert Timestamp to datetime format

# Get data for Aug 2020

df\_aug\_2020 = df[(df["Timestamp"].dt.year == 2020) & (df["Timestamp"].dt.month == 8)]

df\_aug\_2020 = df\_aug\_2020.dropna(subset =["PM2.5"]) # Drop rows w PM2.5 missing

# Find max of pm2.5

idxm = df\_aug\_2020["PM2.5"].idxmax()

max\_station, max\_state, max\_pm = df.loc[idxm, ["station", "state", "PM2.5"]]

print(f"{max\_station} in the state of {max\_state} recorded the highest PM2.5 value (of {max\_pm}) in August 2020.")

34/236:

import pandas as pd

import numpy as np

df = pd.read\_csv(path1)

df["Timestamp"] = pd.to\_datetime(df["Timestamp"], format = "%Y-%m-%d") # Convert Timestamp to datetime format

# Get rows for year 2018

df\_2018 = df[df["Timestamp"].dt.year == 2018]

# Mask for the given stations

df\_stations = df\_2018[df\_2018["station"] == "Lal Bahadur Shastri Nagar, Kalaburagi - KSPCB"]

df\_stations = df\_stations.dropna(subset = ["PM2.5"]) # Drop rows w PM2.5 missing

# Define seasons

get\_season = [

(df\_stations["Timestamp"].dt.month.isin([12, 1, 2])), # winter

(df\_stations["Timestamp"].dt.month.isin([3, 4, 5])), # Summer

(df\_stations["Timestamp"].dt.month.isin([6, 7, 8, 9])) # monsoon

]

# Corresponding seasons for each condition

seasons = ["Winter", "Summer", "Monsoon"]

# Assign a season to each row in df\_stations w np.select()

df\_stations["Season"] = np.select(get\_season, seasons, default="Other")

# Calculate average PM2.5 for each season

avg\_pm25\_per\_season = df\_stations.groupby("Season")["PM2.5"].mean()

# Get season w the highest pollution

most\_polluted\_season = avg\_pm25\_per\_season.idxmax()

highest\_avg\_pm25 = avg\_pm25\_per\_season.max()

print(f"The average PM2.5 concentration for each season for stations Lal Bahadur Shastri Nagar, Kalaburagi - KSPCB is:\n{avg\_pm25\_per\_season}")

print(f"\nThe most polluted season in 2018 was {most\_polluted\_season} with an average PM2.5 concentration of {highest\_avg\_pm25:.3f}")

34/237:

import pandas as pd

import matplotlib.pyplot as plt

df = pd.read\_csv(path1)

df["Timestamp"] = pd.to\_datetime(df["Timestamp"], format = "%Y-%m-%d") # Convert Timestamp to datetime format

# Get rows for year 2021, given station, make a new col for days

df\_2021 = df[(df["Timestamp"].dt.year == 2021) & (df["station"] == "Lal Bahadur Shastri Nagar, Kalaburagi - KSPCB")]

df\_2021 = df\_2021.dropna(subset = ["PM2.5"]) # Drop rows w PM2.5 missing

# Get month and day of the week

df\_2021["Month"] = df\_2021["Timestamp"].dt.month

df\_2021["Day\_of\_week"] = df\_2021["Timestamp"].dt.dayofweek # Monday = 0, Sunday = 6

conditions = [df\_2021["Day\_of\_week"] < 5, df\_2021["Day\_of\_week"] >= 5]

choices = ["Weekday", "Weekend"]

# Assign weekday or weekend

df\_2021["Type"] = np.select(conditions, choices, default = "Weekday")

# Group by month and type, then calc the avg PM2.5

monthly\_avg\_pm25 = df\_2021.groupby(["Month", "Type"])["PM2.5"].mean().unstack() # turn multi indexed series to dataframe, for easy plotting

# Plot

plt.figure(figsize=(10, 5))

plt.plot(monthly\_avg\_pm25.index, monthly\_avg\_pm25["Weekday"], marker="o", linestyle="-", label="Weekdays", color="blue")

plt.plot(monthly\_avg\_pm25.index, monthly\_avg\_pm25["Weekend"], marker="s", linestyle="--", label="Weekends", color="red")

plt.xlabel("Month")

plt.ylabel("Average PM2.5 Levels")

plt.title("Monthly Average PM2.5 Levels (2021)\nStation: Lal Bahadur Shastri Nagar, Kalaburagi - KSPCB")

plt.xticks(range(1, 13), ["Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"])

plt.legend()

plt.show()

34/238:

import pandas as pd

import matplotlib.pyplot as plt

df = pd.read\_csv(path1)

df["Timestamp"] = pd.to\_datetime(df["Timestamp"], format = "%Y-%m-%d") # Convert Timestamp to datetime format

# Get rows for year 2021, given station, make a new col for days

df\_2021 = df[(df["Timestamp"].dt.year == 2021) & (df["station"] == "Lal Bahadur Shastri Nagar, Kalaburagi - KSPCB")]

df\_2021 = df\_2021.dropna(subset = ["PM2.5"]) # Drop rows w PM2.5 missing

# Get month and day of the week

df\_2021["Month"] = df\_2021["Timestamp"].dt.month

df\_2021["Day\_of\_week"] = df\_2021["Timestamp"].dt.dayofweek # Monday = 0, Sunday = 6

conditions = [df\_2021["Day\_of\_week"] < 5, df\_2021["Day\_of\_week"] >= 5]

choices = ["Weekday", "Weekend"]

# Assign weekday or weekend

df\_2021["Type"] = np.select(conditions, choices, default = "Weekday")

# Group by month and type, then calc the avg PM2.5

monthly\_avg\_pm25 = df\_2021.groupby(["Month", "Type"])["PM2.5"].mean().unstack() # turn multi indexed series to dataframe, for easy plotting

# Plot

plt.figure(figsize=(10, 5))

plt.plot(monthly\_avg\_pm25.index, monthly\_avg\_pm25["Weekday"], marker="o", linestyle="-", label="Weekdays", color="blue")

plt.plot(monthly\_avg\_pm25.index, monthly\_avg\_pm25["Weekend"], marker="s", linestyle="--", label="Weekends", color="red")

plt.xlabel("Month")

plt.ylabel("Average PM2.5 Levels")

plt.title("Monthly Average PM2.5 Levels (2021)\nStation: Lal Bahadur Shastri Nagar, Kalaburagi - KSPCB")

plt.xticks(range(1, 13), ["Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"])

plt.legend()

plt.show()

34/239:

import pandas as pd

import numpy as np

df = pd.read\_csv(path1)

# Get rows for year 2022

df["Timestamp"] = pd.to\_datetime(df["Timestamp"], format="%Y-%m-%d") # Convert Timestamp col to datetime format

# Get data for 2022

df\_2022 = df[df["Timestamp"].dt.year == 2022].copy()

# Get month and seasons

df\_2022["Month"] = df\_2022["Timestamp"].dt.month

summer\_months = [3, 4, 5]

monsoon\_months = [6, 7, 8, 9]

df\_summer = df\_2022[df\_2022['Month'].isin(summer\_months)]

df\_monsoon = df\_2022[df\_2022['Month'].isin(monsoon\_months)]

# Calculate the avg PM2.5 levels for summer, monsoon

summer\_avg = df\_summer.groupby('state')['PM2.5'].mean()

monsoon\_avg = df\_monsoon.groupby('state')['PM2.5'].mean()

# Calculate the percentage change of pm2.5 levels of monsoon wrt summer for each state

percentage\_change = ((monsoon\_avg - summer\_avg) / summer\_avg) \* 100

print(f"Percentage Change from Summer to Monsoon: \n{percentage\_change}")

# state with the highest percentage change

max\_change\_state = percentage\_change.abs().idxmax()

max\_change\_value = percentage\_change.abs().max()

print(f"\nState with the highest PM2.5 change: {max\_change\_state} with a percentage decrease of{max\_change\_value: .3f}% in the monsoon.")

34/240:

import pandas as pd

import matplotlib.pyplot as plt

df = pd.read\_csv(path1)

df['Timestamp'] = pd.to\_datetime(df['Timestamp'], format='%Y-%m-%d')

# Get Delhi data

df\_delhi = df[df['city'] == 'Delhi'].copy()

# Get year and month

df\_delhi['Year'] = pd.to\_datetime(df\_delhi['Timestamp'], format='%Y-%m-%d').dt.year

df\_delhi['Month'] = pd.to\_datetime(df\_delhi['Timestamp'], format='%Y-%m-%d').dt.month

# Define seasons

def get\_season(month):

if month in [12, 1, 2]:

return 'Winter'

elif month in [3, 4, 5]:

return 'Summer'

elif month in [6, 7, 8, 9]:

return 'Monsoon'

else:

return 'Other'

df\_delhi['Season'] = df\_delhi['Month'].apply(get\_season)

# Filter data for the years 2017 to 2023, summer winter monsoon seasons

df\_delhi = df\_delhi[(df\_delhi['Year'] >= 2017) &

(df\_delhi['Year'] <= 2023) &

(df\_delhi['Season'].isin(['Winter', 'Summer', 'Monsoon']))]

# Calculate the avg PM2.5 levels for each season from 2017 - 2023

seasonal\_avg\_pm25 = df\_delhi.groupby(['Year', 'Season'])['PM2.5'].mean().unstack()

# Plot

plt.figure(figsize=(10, 6))

seasonal\_avg\_pm25.plot(kind='line', marker='o', figsize=(12, 6))

plt.title("Average PM2.5 Levels in Delhi (2017-2023) by Season")

plt.xlabel("Year")

plt.ylabel("Average PM2.5")

plt.legend(title="Season", loc='upper left')

plt.grid(True)

plt.tight\_layout()

plt.show()

34/241:

import pandas as pd

import matplotlib.pyplot as plt

df = pd.read\_csv(path1)

df['Timestamp'] = pd.to\_datetime(df['Timestamp'], format='%Y-%m-%d')

# Get data for Delhi and Mumbai

df\_delhi\_mumbai = df[df['city'].isin(['Delhi', 'Mumbai'])].copy()

# Get year and month from the Timestamp

df\_delhi\_mumbai['Year'] = df\_delhi\_mumbai['Timestamp'].dt.year

df\_delhi\_mumbai['Month'] = df\_delhi\_mumbai['Timestamp'].dt.month

# Get data for years between 2017 and 2023

df\_delhi\_mumbai = df\_delhi\_mumbai[(df\_delhi\_mumbai['Year'] >= 2017) &

(df\_delhi\_mumbai['Year'] <= 2023)]

# monthly avg PM2.5 levels for Delhi and Mumbai

city\_avg\_pm25 = df\_delhi\_mumbai.groupby(['Year', 'Month', 'city'])['PM2.5'].mean().unstack()

# Plot

plt.figure(figsize=(12, 6))

city\_avg\_pm25['Delhi'].plot(label='Delhi', marker='o', linestyle='-', color='skyblue')

city\_avg\_pm25['Mumbai'].plot(label='Mumbai', marker='o', linestyle='-', color='black')

plt.title("PM2.5 Levels in Delhi and Mumbai (2017-2023)")

plt.xlabel("Year")

plt.ylabel("Average PM2.5")

plt.legend()

plt.tight\_layout()

plt.show()

# Calculate fluctuations (standard deviation) for Delhi and Mumbai

delhi\_fluctuation = city\_avg\_pm25['Delhi'].std()

mumbai\_fluctuation = city\_avg\_pm25['Mumbai'].std()

print(f"Delhi Fluctuation (Standard Deviation): {delhi\_fluctuation:.3f}")

print(f"Mumbai Fluctuation (Standard Deviation): {mumbai\_fluctuation:.3f}")

if delhi\_fluctuation > mumbai\_fluctuation:

print("Delhi had more fluctuations in PM2.5 levels.")

else:

print("Mumbai had more fluctuations in PM2.5 levels.")

34/242:

import pandas as pd

df\_pollution = pd.read\_csv(path1)

df\_pollution.rename(columns={'state': 'State'}, inplace=True)

df\_population = pd.read\_csv(path2)

# No of monitoring stations per state

stations\_per\_state = df\_pollution.groupby('State')['station'].count().reset\_index()

stations\_per\_state.columns = ['State', 'Num\_Stations']

# Merge with state population data

stations\_population = pd.merge(stations\_per\_state, df\_population[['State', 'Population']], on='State', how='left')

# No of stations per million people

stations\_population['Stations\_per\_Million'] = (stations\_population['Num\_Stations'] / stations\_population['Population']) \* 1000000

# State with the highest ratio

max\_stations\_state = stations\_population.loc[stations\_population['Stations\_per\_Million'].idxmax()]

print(f"The state with the highest number of monitoring stations relative to its population is {max\_stations\_state['State']}, with {max\_stations\_state['Stations\_per\_Million']:.2f} stations per million people.")

34/243:

import pandas as pd

import matplotlib.pyplot as plt

# Load datasets

df = pd.read\_csv(path1, parse\_dates=['Timestamp'])

df.rename(columns={'state': 'State'}, inplace=True)

state\_data = pd.read\_csv(path2)

# Get data for 2023

df\_2023 = df[df['Timestamp'].dt.year == 2023].copy()

# Group by state and calc avg PM2.5 levels

state\_pm25 = df\_2023.groupby('State')['PM2.5'].mean()

# Merge with population data

state\_pm25 = state\_pm25.reset\_index()

state\_data = state\_data[['State', 'Population']]

merged = pd.merge(state\_pm25, state\_data, on='State', how='left')

# Calc per capita PM2.5 exposure

merged['Per\_Capita\_PM2.5'] = merged['PM2.5'] / merged['Population']

# Top 5 most polluted states per capita

top5\_states = merged.nlargest(5, 'Per\_Capita\_PM2.5')

# Plot

plt.figure(figsize=(10, 6))

plt.bar(top5\_states['State'], top5\_states['Per\_Capita\_PM2.5'], color = 'skyblue')

plt.xlabel('State')

plt.ylabel('Per Capita PM2.5 Exposure')

plt.title('Top 5 Most Polluted States \nin terms of per capita PM2.5 exposure in 2023')

plt.show()

34/244:

import pandas as pd

import matplotlib.pyplot as plt

df\_population = pd.read\_csv(path2)

df\_pollution = pd.read\_csv(path1)

df\_pollution.rename(columns={'state': 'State'}, inplace=True)

# Calc population density

df\_population['Population Density'] = df\_population['Population'] / df\_population['Area (km2)']

# Calc avg PM2.5 levels for each state

average\_pm25 = df\_pollution.groupby('State')['PM2.5'].mean().reset\_index()

# Merge

merged = pd.merge(df\_population[['State', 'Population Density']], average\_pm25, on='State', how='inner')

# Plotting the relationship

plt.figure(figsize=(10, 6))

plt.scatter(merged['Population Density'], merged['PM2.5'], color='b', marker='o')

plt.title('Population Density vs Average PM2.5 Concentration for Each State', fontsize=14)

plt.xlabel('Population Density (people per km2)', fontsize=12)

plt.ylabel('Average PM2.5 Concentration', fontsize=12)

plt.show()

34/245:

import pandas as pd

import matplotlib.pyplot as plt

# Load the datasets

df\_population = pd.read\_csv(path2)

df\_pollution = pd.read\_csv(path1)

df\_pollution.rename(columns={'state': 'State'}, inplace=True)

# Average PM2.5 for each state

pollution = df\_pollution.groupby('State')['PM2.5'].mean().reset\_index()

# Merge

merged = pd.merge(pollution, df\_population[['State', 'Area (km2)']], on='State', how='inner')

# PM2.5 concentration per km2

merged['PM2.5 per km2'] = merged['PM2.5'] / merged['Area (km2)']

# Sort

merged\_sorted = merged.sort\_values(by='PM2.5 per km2', ascending=False)

# Plot

plt.figure(figsize=(15, 6))

plt.bar(merged\_sorted['State'], merged\_sorted['PM2.5 per km2'])

plt.xticks(rotation=90)

plt.xlabel('State')

plt.ylabel('PM2.5 Concentration per km2')

plt.title('Pollution levels per state relative to their area')

plt.grid(True)

plt.show()

# State with the highest PM2.5 per km2

most\_polluted\_state = merged\_sorted.iloc[0]['State']

highest\_pm25\_value = merged\_sorted.iloc[0]['PM2.5 per km2']

print(f"State with the highest PM2.5 concentration per km2: {most\_polluted\_state} ({highest\_pm25\_value:.3f})")

34/246:

import pandas as pd

import matplotlib.pyplot as plt

Data = pd.read\_csv(path1)

State\_data = pd.read\_csv(path2)

Data.rename(columns={'state': 'State'}, inplace=True)

# number of monitoring stations in each state

station\_counts = Data.groupby('State')['station'].count().reset\_index()

station\_counts.rename(columns={'station': 'Number of Stations'}, inplace=True)

# Merge to get area of each state

merged\_data = station\_counts.merge(State\_data, on="State")

# Calculate monitoring stations per km2

merged\_data['Stations per km2'] = merged\_data['Number of Stations'] / merged\_data['Area (km2)']

# Plot

plt.figure(figsize=(12, 6))

plt.bar(merged\_data['State'], merged\_data['Stations per km2'], color = 'skyblue')

plt.xticks(rotation=90)

plt.xlabel("State")

plt.ylabel("Stations per km2")

plt.title("Density of Monitoring Stations in Each State")

plt.show()

# state with the highest stations density

max\_density\_state = merged\_data.loc[merged\_data['Stations per km2'].idxmax(), 'State']

print(f"State with the highest monitoring station density: {max\_density\_state}")

34/247:

import pandas as pd

df\_pollution = pd.read\_csv(path1)

df\_population = pd.read\_csv(path2)

df\_pollution.rename(columns={'state': 'State'}, inplace=True)

df\_pollution['Timestamp'] = pd.to\_datetime(df\_pollution['Timestamp'], format='%Y-%m-%d')

# Get the data for 2021

df\_2021 = df\_pollution[df\_pollution['Timestamp'].dt.year == 2021].copy()

# Average PM2.5 for Maharashtra and Madhya Pradesh in 2021

avg\_pm25\_2021 = df\_2021.groupby('State')['PM2.5'].mean().reset\_index()

# Get data for Maharashtra and Madhya Pradesh

avg\_pm25\_2021 = avg\_pm25\_2021[avg\_pm25\_2021['State'].isin(['Maharashtra', 'Madhya Pradesh'])]

# Population density for Maharashtra and Madhya Pradesh

df\_mh\_mp = df\_population[df\_population['State'].isin(['Maharashtra', 'Madhya Pradesh'])].copy()

df\_mh\_mp['Population Density'] = df\_mh\_mp['Population'] / df\_mh\_mp['Area (km2)']

# Join population density with PM2.5 levels

merged = pd.merge(avg\_pm25\_2021, df\_mh\_mp[['State', 'Population Density']], on='State')

# Display the results

print("Population Density and Average PM2.5 levels:\n")

print(merged)

34/248:

import pandas as pd

df\_pollution = pd.read\_csv(path1)

df\_population = pd.read\_csv(path2)

df\_ncap = pd.read\_csv(path3)

df\_pollution.rename(columns={'state': 'State'}, inplace=True)

df\_pollution['Timestamp'] = pd.to\_datetime(df\_pollution['Timestamp'], format='%Y-%m-%d')

# Get the data for 2021

df\_2021 = df\_pollution[df\_pollution['Timestamp'].dt.year == 2021].copy()

# Average PM2.5 for each state in 2021

avg\_pm25\_2021 = df\_2021.groupby('State')['PM2.5'].mean().reset\_index()

# Get the list of states that received NCAP funding

ncap\_states = df\_ncap['State'].tolist()

# Split into states with and without NCAP funding

ncap\_funded = avg\_pm25\_2021[avg\_pm25\_2021['State'].isin(ncap\_states)]

non\_ncap\_funded = avg\_pm25\_2021[~avg\_pm25\_2021['State'].isin(ncap\_states)]

# Calculate the average PM2.5 levels for each group

avg\_pm25\_ncap = ncap\_funded['PM2.5'].mean()

avg\_pm25\_non\_ncap = non\_ncap\_funded['PM2.5'].mean()

print(f"Average PM2.5 for states with NCAP funding in 2021: {avg\_pm25\_ncap:.2f}")

print(f"Average PM2.5 for states without NCAP funding in 2021: {avg\_pm25\_non\_ncap:.2f}")

34/249:

import pandas as pd

import matplotlib.pyplot as plt

# Load datasets

df\_pollution = pd.read\_csv(path1)

df\_funding = pd.read\_csv(path3)

# Rename column for consistency

df\_pollution.rename(columns={'state': 'State'}, inplace=True)

# Convert timestamp to datetime

df\_pollution['Timestamp'] = pd.to\_datetime(df\_pollution['Timestamp'])

# Filter for Assam

df\_assam = df\_pollution[df\_pollution['State'] == 'Assam'].copy()

# Create Year-Month column

df\_assam['YearMonth'] = df\_assam['Timestamp'].dt.to\_period('M').astype(str)

# Calculate monthly average PM2.5 levels

monthly\_pm25 = df\_assam.groupby('YearMonth')['PM2.5'].mean().reset\_index()

# Process NCAP funding dataset

df\_funding\_assam = df\_funding[df\_funding['State'] == 'Assam']

# Aggregate total funding for Assam per year

funding\_per\_year = df\_funding\_assam.groupby('State')[['Amount released during FY 2019-20',

'Amount released during FY 2020-21',

'Amount released during FY 2021-22']].sum().reset\_index()

# Convert to long format for merging

funding\_per\_year = funding\_per\_year.melt(id\_vars=['State'], var\_name='Year', value\_name='Funding')

# Extract only the year from 'Year' column

funding\_per\_year['Year'] = funding\_per\_year['Year'].str.extract(r'(\d{4})')

# Convert to proper YearMonth format

funding\_per\_year['YearMonth'] = funding\_per\_year['Year'] + '-01'

# Convert 'YearMonth' to datetime format

merged['YearMonth'] = pd.to\_datetime(merged['YearMonth'])

# Ensure 'Funding' is numeric and replace NaN with 0

merged['Funding'] = pd.to\_numeric(merged['Funding'], errors='coerce').fillna(0)

# Sort values by time

merged = merged.sort\_values('YearMonth')

# Plot time-series graph

fig, ax1 = plt.subplots(figsize=(12, 6))

# PM2.5 levels (Primary Y-axis)

ax1.plot(merged['YearMonth'], merged['PM2.5'], marker='o', linestyle='-', color='red', label='PM2.5 Level')

ax1.set\_xlabel('Year-Month')

ax1.set\_ylabel('PM2.5 Level', color='red')

ax1.tick\_params(axis='y', labelcolor='red')

# Funding Data (Secondary Y-axis)

ax2 = ax1.twinx()

ax2.plot(merged['YearMonth'], merged['Funding'], marker='s', linestyle='--', color='blue', label='NCAP Funding')

ax2.set\_ylabel('NCAP Funding (in crores)', color='blue')

ax2.tick\_params(axis='y', labelcolor='blue')

# Labels & Title

fig.suptitle('PM2.5 Levels vs NCAP Funding in Assam (Time Series)')

ax1.grid()

plt.xticks(rotation=45)

# Show the plot

plt.show()

34/250:

import pandas as pd

import matplotlib.pyplot as plt

# Load datasets

df\_pollution = pd.read\_csv(path1)

df\_funding = pd.read\_csv(path3)

# Rename column for consistency

df\_pollution.rename(columns={'state': 'State'}, inplace=True)

# Convert timestamp to datetime

df\_pollution['Timestamp'] = pd.to\_datetime(df\_pollution['Timestamp'])

# Filter for Assam

df\_assam = df\_pollution[df\_pollution['State'] == 'Assam'].copy()

# Create Year-Month column

df\_assam['YearMonth'] = df\_assam['Timestamp'].dt.to\_period('M').astype(str).copy()

# Calculate monthly average PM2.5 levels

monthly\_pm25 = df\_assam.groupby('YearMonth')['PM2.5'].mean().reset\_index()

# Process NCAP funding dataset

df\_funding\_assam = df\_funding[df\_funding['State'] == 'Assam']

# Aggregate total funding for Assam per year

funding\_per\_year = df\_funding\_assam.groupby('State')[['Amount released during FY 2019-20',

'Amount released during FY 2020-21',

'Amount released during FY 2021-22']].sum().reset\_index()

# Convert to long format for merging

funding\_per\_year = funding\_per\_year.melt(id\_vars=['State'], var\_name='Year', value\_name='Funding')

# Extract only the year from 'Year' column

funding\_per\_year['Year'] = funding\_per\_year['Year'].str.extract(r'(\d{4})')

# Convert to proper YearMonth format

funding\_per\_year['YearMonth'] = funding\_per\_year['Year'] + '-01'

# Convert 'YearMonth' to datetime format

merged['YearMonth'] = pd.to\_datetime(merged['YearMonth'])

# Ensure 'Funding' is numeric and replace NaN with 0

merged['Funding'] = pd.to\_numeric(merged['Funding'], errors='coerce').fillna(0)

# Sort values by time

merged = merged.sort\_values('YearMonth')

# Plot time-series graph

fig, ax1 = plt.subplots(figsize=(12, 6))

# PM2.5 levels (Primary Y-axis)

ax1.plot(merged['YearMonth'], merged['PM2.5'], marker='o', linestyle='-', color='red', label='PM2.5 Level')

ax1.set\_xlabel('Year-Month')

ax1.set\_ylabel('PM2.5 Level', color='red')

ax1.tick\_params(axis='y', labelcolor='red')

# Funding Data (Secondary Y-axis)

ax2 = ax1.twinx()

ax2.plot(merged['YearMonth'], merged['Funding'], marker='s', linestyle='--', color='blue', label='NCAP Funding')

ax2.set\_ylabel('NCAP Funding (in crores)', color='blue')

ax2.tick\_params(axis='y', labelcolor='blue')

# Labels & Title

fig.suptitle('PM2.5 Levels vs NCAP Funding in Assam (Time Series)')

ax1.grid()

plt.xticks(rotation=45)

# Show the plot

plt.show()

34/251:

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

# Load datasets

df\_population = pd.read\_csv(path1)

df\_ncap = pd.read\_csv(path3)

# Rename columns for consistency

df\_population.rename(columns={'Area (km2)': 'Area\_sq\_km'}, inplace=True)

df\_ncap.rename(columns={'Total fund released': 'Total\_Funding\_Crores'}, inplace=True)

# Group NCAP data by state to get total funding per state

df\_ncap\_statewise = df\_ncap.groupby('State', as\_index=False)['Total\_Funding\_Crores'].sum()

# Merge datasets based on State

df\_merged = pd.merge(df\_population, df\_ncap\_statewise, on="State", how="inner")

# Set style

sns.set(style="whitegrid")

# Create scatter plot

plt.figure(figsize=(14, 8)) # Adjusted size for better clarity

scatter = sns.scatterplot(

data=df\_merged,

x='Area\_sq\_km',

y='Total\_Funding\_Crores',

hue='State',

palette=sns.color\_palette("tab20", len(df\_merged['State'].unique())),

s=100

)

# Adjust legend position outside the plot

plt.legend(title="State", bbox\_to\_anchor=(1.05, 1), loc='upper left')

# Labels and title

plt.xlabel("State Area (sq km)")

plt.ylabel("Total NCAP Funding (in crores)")

plt.title("State-wise Area vs NCAP Funding Received")

# Show plot

plt.tight\_layout() # Ensure the legend fits outside the plot

plt.show()

34/252:

import pandas as pd

df = pd.read\_csv(path1)

df['Timestamp'] = pd.to\_datetime(df['Timestamp'], format='%Y-%m-%d')

# Get data for 2023

df\_2023 = df[df['Timestamp'].dt.year == 2023].copy()

# Group by City and calculate avg PM2.5

city\_avg\_pm25 = df\_2023.groupby('city')['PM2.5'].mean()

# Most polluted city

most\_polluted\_city = city\_avg\_pm25.idxmax()

highest\_avg\_pm25 = city\_avg\_pm25.max()

# Get data for the most polluted city

df\_most\_polluted = df\_2023[df\_2023['city'] == most\_polluted\_city]

df\_most\_polluted = df\_most\_polluted.dropna(subset = ["PM2.5"]).copy()

# Count the number of days with PM2.5 > 300

hazardous\_days = (df\_most\_polluted['PM2.5'] > 300).sum()

# Total number of days recorded for this city

total\_days = df\_most\_polluted['Timestamp'].nunique()

# Calculate percentage of hazardous days

percentage\_hazardous\_days = (hazardous\_days / total\_days) \* 100

print(f"Most Polluted City in 2023: {most\_polluted\_city}")

print(f"Percentage of Hazardous Days: {percentage\_hazardous\_days:.3f}%")

34/253:

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

df = pd.read\_csv(path1)

df['Timestamp'] = pd.to\_datetime(df['Timestamp'], format='%Y-%m-%d')

# Get data for Delhi

df\_delhi = df[df['city'] == 'Delhi'].copy()

# Get year, month, and day

df\_delhi['Year'] = df\_delhi['Timestamp'].dt.year

df\_delhi['Month'] = df\_delhi['Timestamp'].dt.month

df\_delhi['Day'] = df\_delhi['Timestamp'].dt.day

# Get data for the years 2018-2020

df\_delhi = df\_delhi[df\_delhi['Year'].isin([2018, 2019, 2020])]

# Define festival periods

diwali\_months = [10, 11] # October & November

new\_year\_days = [(1, 1)] # January 1st

# Get Diwali and New Year Data

df\_diwali = df\_delhi[df\_delhi['Month'].isin(diwali\_months)].copy()

df\_new\_year = df\_delhi[(df\_delhi['Month'] == 1) & (df\_delhi['Day'] == 1)].copy()

# Plot PM2.5 levels over time

plt.figure(figsize=(14, 6))

sns.lineplot(data=df\_delhi, x='Timestamp', y='PM2.5', label="PM2.5 Levels", color="blue")

# Mark Diwali spikes with arrows

diwali\_spikes = df\_diwali[df\_diwali['PM2.5'] > df\_diwali['PM2.5'].quantile(0.90)] # Top 10% as spikes

for index, row in diwali\_spikes.iterrows():

plt.annotate('Diwali Spike', xy=(row['Timestamp'], row['PM2.5']),

xytext=(row['Timestamp'], row['PM2.5'] + 50),

arrowprops=dict(facecolor='red', shrink=0.05),

fontsize=9, color='red')

# Mark New Year spikes with arrows

for index, row in df\_new\_year.iterrows():

plt.annotate('New Year Spike', xy=(row['Timestamp'], row['PM2.5']),

xytext=(row['Timestamp'], row['PM2.5'] + 50),

arrowprops=dict(facecolor='green', shrink=0.05),

fontsize=9, color='green')

plt.xlabel("Year")

plt.ylabel("PM2.5 (µg/m³)")

plt.title("PM2.5 Levels in Delhi (2018-2020) - Diwali & New Year Spikes")

plt.legend()

plt.xticks(rotation=45)

plt.show()

34/254:

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

df = pd.read\_csv(path1)

df['Timestamp'] = pd.to\_datetime(df['Timestamp'], format='%Y-%m-%d')

# Get data for Delhi

df\_delhi = df[df['city'] == 'Delhi'].copy()

# Get year, month, and day

df\_delhi['Year'] = df\_delhi['Timestamp'].dt.year

df\_delhi['Month'] = df\_delhi['Timestamp'].dt.month

df\_delhi['Day'] = df\_delhi['Timestamp'].dt.day

# Get data for the years 2018-2020

df\_delhi = df\_delhi[df\_delhi['Year'].isin([2018, 2019, 2020])]

# Define festival periods

diwali\_months = [10, 11] # October & November

new\_year\_days = [(1, 1)] # January 1st

# Get Diwali and New Year Data

df\_diwali = df\_delhi[df\_delhi['Month'].isin(diwali\_months)].copy()

df\_new\_year = df\_delhi[(df\_delhi['Month'] == 1) & (df\_delhi['Day'] == 1)].copy()

# Plot PM2.5 levels over time

plt.figure(figsize=(14, 6))

sns.lineplot(data=df\_delhi, x='Timestamp', y='PM2.5', label="PM2.5 Levels", color="blue")

# Mark Diwali spikes with arrows

diwali\_spikes = df\_diwali[df\_diwali['PM2.5'] > df\_diwali['PM2.5'].quantile(0.90)] # Top 10% as spikes

for index, row in diwali\_spikes.iterrows():

plt.annotate('Diwali Spike', xy=(row['Timestamp'], row['PM2.5']),

xytext=(row['Timestamp'], row['PM2.5'] + 50),

arrowprops=dict(facecolor='red', shrink=0.05),

fontsize=9, color='red')

# Mark New Year spikes with arrows

for index, row in df\_new\_year.iterrows():

plt.annotate('New Year Spike', xy=(row['Timestamp'], row['PM2.5']),

xytext=(row['Timestamp'], row['PM2.5'] + 50),

arrowprops=dict(facecolor='green', shrink=0.05),

fontsize=9, color='green')

plt.xlabel("Year")

plt.ylabel("PM2.5 (µg/m³)")

plt.title("PM2.5 Levels in Delhi (2018-2020) - Diwali & New Year Spikes")

plt.legend()

plt.xticks(rotation=45)

plt.show()

36/1:

import pandas as pd

df = pd.read\_csv(path1)

df['Timestamp'] = pd.to\_datetime(df['Timestamp'], format='%Y-%m-%d')

# Get data for 2023

df\_2023 = df[df['Timestamp'].dt.year == 2023].copy()

# Group by City and calculate avg PM2.5

city\_avg\_pm25 = df\_2023.groupby('city')['PM2.5'].mean()

# Most polluted city

most\_polluted\_city = city\_avg\_pm25.idxmax()

highest\_avg\_pm25 = city\_avg\_pm25.max()

# Get data for the most polluted city

df\_most\_polluted = df\_2023[df\_2023['city'] == most\_polluted\_city]

df\_most\_polluted = df\_most\_polluted.dropna(subset = ["PM2.5"]).copy()

# Count the number of days with PM2.5 > 300

hazardous\_days = (df\_most\_polluted['PM2.5'] > 300).sum()

# Total number of days recorded for this city

total\_days = df\_most\_polluted['Timestamp'].nunique()

# Calculate percentage of hazardous days

percentage\_hazardous\_days = (hazardous\_days / total\_days) \* 100

print(f"Most Polluted City in 2023: {most\_polluted\_city}")

print(f"Percentage of Hazardous Days: {percentage\_hazardous\_days:.3f}%")

36/2:

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

path1 = r"C:\Users\inees\Downloads\Data.csv" # Data.csv

path2 = r"C:\Users\inees\Downloads\State\_data.csv" # State\_data.csv

path3 = r"C:\Users\inees\Downloads\NCAP\_Funding.csv" # NCAP\_Funding.csv

36/3:

import pandas as pd

df = pd.read\_csv(path1)

df['Timestamp'] = pd.to\_datetime(df['Timestamp'], format='%Y-%m-%d')

# Get data for 2023

df\_2023 = df[df['Timestamp'].dt.year == 2023].copy()

# Group by City and calculate avg PM2.5

city\_avg\_pm25 = df\_2023.groupby('city')['PM2.5'].mean()

# Most polluted city

most\_polluted\_city = city\_avg\_pm25.idxmax()

highest\_avg\_pm25 = city\_avg\_pm25.max()

# Get data for the most polluted city

df\_most\_polluted = df\_2023[df\_2023['city'] == most\_polluted\_city]

df\_most\_polluted = df\_most\_polluted.dropna(subset = ["PM2.5"]).copy()

# Count the number of days with PM2.5 > 300

hazardous\_days = (df\_most\_polluted['PM2.5'] > 300).sum()

# Total number of days recorded for this city

total\_days = df\_most\_polluted['Timestamp'].nunique()

# Calculate percentage of hazardous days

percentage\_hazardous\_days = (hazardous\_days / total\_days) \* 100

print(f"Most Polluted City in 2023: {most\_polluted\_city}")

print(f"Percentage of Hazardous Days: {percentage\_hazardous\_days:.3f}%")

36/4:

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

df = pd.read\_csv(path1)

df['Timestamp'] = pd.to\_datetime(df['Timestamp'], format='%Y-%m-%d')

# Get data for Delhi

df\_delhi = df[df['city'] == 'Delhi'].copy()

# Get year, month, and day

df\_delhi['Year'] = df\_delhi['Timestamp'].dt.year

df\_delhi['Month'] = df\_delhi['Timestamp'].dt.month

df\_delhi['Day'] = df\_delhi['Timestamp'].dt.day

# Get data for the years 2018-2020

df\_delhi = df\_delhi[df\_delhi['Year'].isin([2018, 2019, 2020])]

# Define festival periods

diwali\_months = [10, 11] # October & November

new\_year\_days = [(1, 1)] # January 1st

# Get Diwali and New Year Data

df\_diwali = df\_delhi[df\_delhi['Month'].isin(diwali\_months)].copy()

df\_new\_year = df\_delhi[(df\_delhi['Month'] == 1) & (df\_delhi['Day'] == 1)].copy()

# Plot PM2.5 levels over time

plt.figure(figsize=(14, 6))

sns.lineplot(data=df\_delhi, x='Timestamp', y='PM2.5', label="PM2.5 Levels", color="blue")

# Mark Diwali spikes with arrows

diwali\_spikes = df\_diwali[df\_diwali['PM2.5'] > df\_diwali['PM2.5'].quantile(0.90)] # Top 10% as spikes

for index, row in diwali\_spikes.iterrows():

plt.annotate('Diwali Spike', xy=(row['Timestamp'], row['PM2.5']),

xytext=(row['Timestamp'], row['PM2.5'] + 50),

arrowprops=dict(facecolor='red', shrink=0.05),

fontsize=9, color='red')

# Mark New Year spikes with arrows

for index, row in df\_new\_year.iterrows():

plt.annotate('New Year Spike', xy=(row['Timestamp'], row['PM2.5']),

xytext=(row['Timestamp'], row['PM2.5'] + 50),

arrowprops=dict(facecolor='green', shrink=0.05),

fontsize=9, color='green')

plt.xlabel("Year")

plt.ylabel("PM2.5 (µg/m³)")

plt.title("PM2.5 Levels in Delhi (2018-2020) - Diwali & New Year Spikes")

plt.legend()

plt.xticks(rotation=45)

plt.show()

37/1:

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

path1 = r"C:\Users\inees\Downloads\Data.csv" # Data.csv

path2 = r"C:\Users\inees\Downloads\State\_data.csv" # State\_data.csv

path3 = r"C:\Users\inees\Downloads\NCAP\_Funding.csv" # NCAP\_Funding.csv

37/2:

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

path1 = r"C:\Users\inees\Downloads\Data.csv" # Data.csv

path2 = r"C:\Users\inees\Downloads\State\_data.csv" # State\_data.csv

path3 = r"C:\Users\inees\Downloads\NCAP\_Funding.csv" # NCAP\_Funding.csv

37/3:

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

# Load datasets

df\_pollution = pd.read\_csv(path1)

df\_ncap = pd.read\_csv(path3)

# Convert timestamp to datetime format

df\_pollution['Timestamp'] = pd.to\_datetime(df\_pollution['Timestamp'], format='%Y-%m-%d')

# Rename columns for consistency

df\_ncap.rename(columns={'Amount released during FY 2020-21': 'Funding\_2020'}, inplace=True)

# Identify the state with highest NCAP funding in 2020

state\_highest\_funding = df\_ncap.groupby("State")['Funding\_2020'].sum().idxmax()

highest\_funding\_amount = df\_ncap.groupby("State")['Funding\_2020'].sum().max()

print(f"The state with the highest NCAP funding in 2020: {state\_highest\_funding} (₹{highest\_funding\_amount} Crores)")

# Filter pollution data for cities in that state

df\_state\_cities = df\_ncap[df\_ncap['State'] == state\_highest\_funding]['City'].unique()

df\_state\_pollution = df\_pollution[df\_pollution['city'].isin(df\_state\_cities)]

# Extract Yearly PM2.5 averages (2018-2023)

df\_state\_pollution['Year'] = df\_state\_pollution['Timestamp'].dt.year

yearly\_pm25 = df\_state\_pollution.groupby('Year')['PM2.5'].mean()

# Plot PM2.5 trends

plt.figure(figsize=(10, 5))

sns.lineplot(x=yearly\_pm25.index, y=yearly\_pm25.values, marker='o', label=state\_highest\_funding, color='blue')

# Mark NCAP intervention year (2020)

plt.axvline(x=2020, linestyle='--', color='red', label='NCAP Funding (2020)')

# Labels and Title

plt.xlabel("Year")

plt.ylabel("Average PM2.5 (µg/m³)")

plt.title(f"PM2.5 Trends in {state\_highest\_funding} (2018-2023) - Impact of NCAP Funding")

plt.legend()

plt.grid(True)

plt.xticks(range(2018, 2024))

# Show plot

plt.show()

37/4:

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

# Load datasets

df\_pollution = pd.read\_csv(path1)

df\_ncap = pd.read\_csv(path3)

# Convert timestamp to datetime format

df\_pollution['Timestamp'] = pd.to\_datetime(df\_pollution['Timestamp'], format='%Y-%m-%d')

# Rename columns for consistency

df\_ncap.rename(columns={'Amount released during FY 2020-21': 'Funding\_2020'}, inplace=True)

# Identify the state with highest NCAP funding in 2020

state\_highest\_funding = df\_ncap.groupby("State")['Funding\_2020'].sum().idxmax()

highest\_funding\_amount = df\_ncap.groupby("State")['Funding\_2020'].sum().max()

print(f"The state with the highest NCAP funding in 2020: {state\_highest\_funding} (₹{highest\_funding\_amount} Crores)")

# Filter pollution data for cities in that state

df\_state\_cities = df\_ncap[df\_ncap['State'] == state\_highest\_funding]['City'].unique()

df\_state\_pollution = df\_pollution[df\_pollution['city'].isin(df\_state\_cities)]

# Extract Yearly PM2.5 averages (2018-2023)

df\_state\_pollution['Year'] = df\_state\_pollution['Timestamp'].dt.year.copy()

yearly\_pm25 = df\_state\_pollution.groupby('Year')['PM2.5'].mean()

# Plot PM2.5 trends

plt.figure(figsize=(10, 5))

sns.lineplot(x=yearly\_pm25.index, y=yearly\_pm25.values, marker='o', label=state\_highest\_funding, color='blue')

# Mark NCAP intervention year (2020)

plt.axvline(x=2020, linestyle='--', color='red', label='NCAP Funding (2020)')

# Labels and Title

plt.xlabel("Year")

plt.ylabel("Average PM2.5 (µg/m³)")

plt.title(f"PM2.5 Trends in {state\_highest\_funding} (2018-2023) - Impact of NCAP Funding")

plt.legend()

plt.grid(True)

plt.xticks(range(2018, 2024))

# Show plot

plt.show()

37/5:

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

# Load datasets

df\_pollution = pd.read\_csv(path1)

df\_ncap = pd.read\_csv(path3)

# Convert timestamp to datetime format

df\_pollution['Timestamp'] = pd.to\_datetime(df\_pollution['Timestamp'], format='%Y-%m-%d')

# Rename columns for consistency

df\_ncap.rename(columns={'Amount released during FY 2020-21': 'Funding\_2020'}, inplace=True)

# Identify the state with highest NCAP funding in 2020

state\_highest\_funding = df\_ncap.groupby("State")['Funding\_2020'].sum().idxmax()

highest\_funding\_amount = df\_ncap.groupby("State")['Funding\_2020'].sum().max()

print(f"The state with the highest NCAP funding in 2020: {state\_highest\_funding} (₹{highest\_funding\_amount} Crores)")

# Filter pollution data for cities in that state

df\_state\_cities = df\_ncap[df\_ncap['State'] == state\_highest\_funding]['City'].unique()

df\_state\_pollution = df\_pollution[df\_pollution['city'].isin(df\_state\_cities)].copy()

# Extract Yearly PM2.5 averages (2018-2023)

df\_state\_pollution['Year'] = df\_state\_pollution['Timestamp'].dt.year

yearly\_pm25 = df\_state\_pollution.groupby('Year')['PM2.5'].mean()

# Plot PM2.5 trends

plt.figure(figsize=(10, 5))

sns.lineplot(x=yearly\_pm25.index, y=yearly\_pm25.values, marker='o', label=state\_highest\_funding, color='blue')

# Mark NCAP intervention year (2020)

plt.axvline(x=2020, linestyle='--', color='red', label='NCAP Funding (2020)')

# Labels and Title

plt.xlabel("Year")

plt.ylabel("Average PM2.5 (µg/m³)")

plt.title(f"PM2.5 Trends in {state\_highest\_funding} (2018-2023) - Impact of NCAP Funding")

plt.legend()

plt.grid(True)

plt.xticks(range(2018, 2024))

# Show plot

plt.show()

37/6:

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

# Load datasets

df\_pollution = pd.read\_csv(path1)

df\_ncap = pd.read\_csv(path3)

df\_pollution['Timestamp'] = pd.to\_datetime(df\_pollution['Timestamp'], format='%Y-%m-%d')

df\_ncap.rename(columns={'Amount released during FY 2020-21': 'Funding\_2020'}, inplace=True)

# Identify the state with highest NCAP funding in 2020

state\_highest\_funding = df\_ncap.groupby("State")['Funding\_2020'].sum().idxmax()

highest\_funding\_amount = df\_ncap.groupby("State")['Funding\_2020'].sum().max()

print(f"The state with the highest NCAP funding in 2020: {state\_highest\_funding} (₹{highest\_funding\_amount} Crores)")

# pollution data for cities in that state

df\_state\_cities = df\_ncap[df\_ncap['State'] == state\_highest\_funding]['City'].unique()

df\_state\_pollution = df\_pollution[df\_pollution['city'].isin(df\_state\_cities)].copy()

# Yearly PM2.5 averages (2018-2023)

df\_state\_pollution['Year'] = df\_state\_pollution['Timestamp'].dt.year

yearly\_pm25 = df\_state\_pollution.groupby('Year')['PM2.5'].mean()

# Plot

plt.figure(figsize=(10, 5))

sns.lineplot(x=yearly\_pm25.index, y=yearly\_pm25.values, marker='o', label=state\_highest\_funding, color='blue')

# Mark NCAP intervention year (2020)

plt.axvline(x=2020, linestyle='--', color='red', label='NCAP Funding (2020)')

plt.xlabel("Year")

plt.ylabel("Average PM2.5")

plt.title(f"PM2.5 Trends in {state\_highest\_funding} (2018-2023) - Impact of NCAP Funding")

plt.legend()

plt.grid(True)

plt.xticks(range(2018, 2024))

plt.show()

37/7:

import pandas as pd

import geopandas as gpd

import matplotlib.pyplot as plt

# Load air pollution dataset

df = pd.read\_csv(path1)

# Load India map shapefile

india\_map = gpd.read\_file(gpd.datasets.get\_path('naturalearth\_lowres'))

india\_map = india\_map[india\_map.name == "India"] # Filter for India

# Convert DataFrame to GeoDataFrame

gdf = gpd.GeoDataFrame(df, geometry=gpd.points\_from\_xy(df['Longitude'], df['Latitude']))

# Plot the map

fig, ax = plt.subplots(figsize=(10, 10))

india\_map.plot(ax=ax, color='lightgrey', edgecolor='black') # Plot India map

gdf.plot(ax=ax, color='red', markersize=10, alpha=0.6) # Plot sensor locations

# Labels and title

plt.xlabel("Longitude")

plt.ylabel("Latitude")

plt.title("Air Quality Monitoring Stations Across India")

plt.show()

37/8:

import pandas as pd

import geopandas as gpd

import matplotlib.pyplot as plt

# Load air pollution dataset

df = pd.read\_csv(path1)

# Load India map shapefile

india\_map = gpd.read\_file(gpd.datasets.get\_path('naturalearth\_lowres'))

india\_map = india\_map[india\_map.name == "India"] # Filter for India

# Convert DataFrame to GeoDataFrame

gdf = gpd.GeoDataFrame(df, geometry=gpd.points\_from\_xy(df['Longitude'], df['Latitude']))

# Plot the map

fig, ax = plt.subplots(figsize=(10, 10))

india\_map.plot(ax=ax, color='lightgrey', edgecolor='black') # Plot India map

gdf.plot(ax=ax, color='red', markersize=10, alpha=0.6) # Plot sensor locations

# Labels and title

plt.xlabel("Longitude")

plt.ylabel("Latitude")

plt.title("Air Quality Monitoring Stations Across India")

plt.show()

37/9:

import pandas as pd

import geopandas as gpd

import matplotlib.pyplot as plt

# Load air pollution dataset

df = pd.read\_csv(path1)

# Load India map shapefile

url = 'https://www.naturalearthdata.com/http//www.naturalearthdata.com/download/110m/cultural/ne\_110m\_admin\_0\_countries.zip'

india\_map = gpd.read\_file(url)

india\_map = india\_map[india\_map.name == "India"] # Filter for India

# Convert DataFrame to GeoDataFrame

gdf = gpd.GeoDataFrame(df, geometry=gpd.points\_from\_xy(df['Longitude'], df['Latitude']))

# Plot the map

fig, ax = plt.subplots(figsize=(10, 10))

india\_map.plot(ax=ax, color='lightgrey', edgecolor='black') # Plot India map

gdf.plot(ax=ax, color='red', markersize=10, alpha=0.6) # Plot sensor locations

# Labels and title

plt.xlabel("Longitude")

plt.ylabel("Latitude")

plt.title("Air Quality Monitoring Stations Across India")

plt.show()

37/10:

import matplotlib.pyplot as plt

import cartopy.crs as ccrs

import cartopy.feature as cfeature

# Grouping data by station and calculating mean of latitude and longitude

x = Data[["latitude", "longitude", "station"]].groupby(["station"]).mean()

print(x)

# Create a map with a specific projection (for India, PlateCarree works well)

fig, ax = plt.subplots(figsize=(10, 10), subplot\_kw={'projection': ccrs.PlateCarree()})

ax.coastlines()

ax.add\_feature(cfeature.BORDERS, linestyle='-')

ax.add\_feature(cfeature.LAND, facecolor='white')

ax.add\_feature(cfeature.OCEAN, facecolor='lightblue')

# Set extent for the map (India's boundaries)

ax.set\_extent([68, 98, 6, 37], crs=ccrs.PlateCarree()) # [min\_lon, max\_lon, min\_lat, max\_lat]

# Plot sensor locations

for i in range(len(x)):

ax.plot(x.iloc[i, 1], x.iloc[i, 0], marker='o', color='red', markersize=8, transform=ccrs.PlateCarree())

ax.text(x.iloc[i, 1] + 0.2, x.iloc[i, 0] + 0.2, x.index[i], color='black', fontsize=12, transform=ccrs.PlateCarree()) # Label city names

ax.set\_title('Sensor Locations in India')

plt.show()

37/11:

import matplotlib.pyplot as plt

import cartopy.crs as ccrs

import cartopy.feature as cfeature

# Grouping data by station and calculating mean of latitude and longitude

x = Data[["latitude", "longitude", "station"]].groupby(["station"]).mean()

print(x)

# Create a map with a specific projection (for India, PlateCarree works well)

fig, ax = plt.subplots(figsize=(10, 10), subplot\_kw={'projection': ccrs.PlateCarree()})

ax.coastlines()

ax.add\_feature(cfeature.BORDERS, linestyle='-')

ax.add\_feature(cfeature.LAND, facecolor='white')

ax.add\_feature(cfeature.OCEAN, facecolor='lightblue')

# Set extent for the map (India's boundaries)

ax.set\_extent([68, 98, 6, 37], crs=ccrs.PlateCarree()) # [min\_lon, max\_lon, min\_lat, max\_lat]

# Plot sensor locations

for i in range(len(x)):

ax.plot(x.iloc[i, 1], x.iloc[i, 0], marker='o', color='red', markersize=8, transform=ccrs.PlateCarree())

ax.text(x.iloc[i, 1] + 0.2, x.iloc[i, 0] + 0.2, x.index[i], color='black', fontsize=12, transform=ccrs.PlateCarree()) # Label city names

ax.set\_title('Sensor Locations in India')

plt.show()

37/12:

import matplotlib.pyplot as plt

import cartopy.crs as ccrs

import cartopy.feature as cfeature

df = pd.read\_csv(path1)

# Grouping data by station and calculating mean of latitude and longitude

x = df[["latitude", "longitude", "station"]].groupby(["station"]).mean()

print(x)

# Create a map with a specific projection (for India, PlateCarree works well)

fig, ax = plt.subplots(figsize=(10, 10), subplot\_kw={'projection': ccrs.PlateCarree()})

ax.coastlines()

ax.add\_feature(cfeature.BORDERS, linestyle='-')

ax.add\_feature(cfeature.LAND, facecolor='white')

ax.add\_feature(cfeature.OCEAN, facecolor='lightblue')

# Set extent for the map (India's boundaries)

ax.set\_extent([68, 98, 6, 37], crs=ccrs.PlateCarree()) # [min\_lon, max\_lon, min\_lat, max\_lat]

# Plot sensor locations

for i in range(len(x)):

ax.plot(x.iloc[i, 1], x.iloc[i, 0], marker='o', color='red', markersize=8, transform=ccrs.PlateCarree())

ax.text(x.iloc[i, 1] + 0.2, x.iloc[i, 0] + 0.2, x.index[i], color='black', fontsize=12, transform=ccrs.PlateCarree()) # Label city names

ax.set\_title('Sensor Locations in India')

plt.show()

37/13:

import matplotlib.pyplot as plt

import cartopy.crs as ccrs

import cartopy.feature as cfeature

df = pd.read\_csv(path1)

# Grouping data by station and calculating mean of latitude and longitude

x = df[["latitude", "longitude", "station"]].groupby(["station"]).mean()

print(x)

# Create a map with a specific projection (for India, PlateCarree works well)

fig, ax = plt.subplots(figsize=(10, 10), subplot\_kw={'projection': ccrs.PlateCarree()})

ax.coastlines()

ax.add\_feature(cfeature.BORDERS, linestyle='-')

ax.add\_feature(cfeature.LAND, facecolor='white')

ax.add\_feature(cfeature.OCEAN, facecolor='lightblue')

# Set extent for the map (India's boundaries)

ax.set\_extent([68, 98, 6, 37], crs=ccrs.PlateCarree()) # [min\_lon, max\_lon, min\_lat, max\_lat]

# Plot sensor locations

for i in range(len(x)):

ax.plot(x.iloc[i, 1], x.iloc[i, 0], marker='o', color='red', markersize=8, transform=ccrs.PlateCarree())

ax.set\_title('Sensor Locations in India')

plt.show()

37/14:

import matplotlib.pyplot as plt

import cartopy.crs as ccrs

import cartopy.feature as cfeature

df = pd.read\_csv(path1)

# Grouping data by station and calculating mean of latitude and longitude

x = df[["latitude", "longitude", "station"]].groupby(["station"]).mean()

print(x)

# Create a map with a specific projection (for India, PlateCarree works well)

fig, ax = plt.subplots(figsize=(15, 15), subplot\_kw={'projection': ccrs.PlateCarree()})

ax.coastlines()

ax.add\_feature(cfeature.BORDERS, linestyle='-')

ax.add\_feature(cfeature.LAND, facecolor='white')

ax.add\_feature(cfeature.OCEAN, facecolor='lightblue')

# Set extent for the map (India's boundaries)

ax.set\_extent([68, 98, 6, 37], crs=ccrs.PlateCarree()) # [min\_lon, max\_lon, min\_lat, max\_lat]

# Plot sensor locations

for i in range(len(x)):

ax.plot(x.iloc[i, 1], x.iloc[i, 0], marker='.', color='red', markersize=5, transform=ccrs.PlateCarree())

ax.set\_title('Sensor Locations in India')

plt.show()

37/15:

import matplotlib.pyplot as plt

import cartopy.crs as ccrs

import cartopy.feature as cfeature

df = pd.read\_csv(path1)

# Grouping data by station and calculating mean of latitude and longitude

x = df[["latitude", "longitude", "station"]].groupby(["station"]).mean()

print(x)

# Create a map with a specific projection (for India, PlateCarree works well)

fig, ax = plt.subplots(figsize=(15, 15), subplot\_kw={'projection': ccrs.PlateCarree()})

ax.coastlines()

ax.add\_feature(cfeature.BORDERS, linestyle='-')

ax.add\_feature(cfeature.LAND, facecolor='white')

ax.add\_feature(cfeature.OCEAN, facecolor='lightblue')

# Set extent for the map (India's boundaries)

ax.set\_extent([68, 98, 6, 37], crs=ccrs.PlateCarree()) # [min\_lon, max\_lon, min\_lat, max\_lat]

# Plot sensor locations

for i in range(len(x)):

ax.plot(x.iloc[i, 1], x.iloc[i, 0], marker='o', color='red', markersize=5, transform=ccrs.PlateCarree())

ax.set\_title('Sensor Locations in India')

plt.show()

37/16:

import pandas as pd

import geopandas as gpd

import matplotlib.pyplot as plt

# Load air pollution dataset

df = pd.read\_csv(r"C:\Users\inees\Downloads\Data.csv")

# Load India map shapefile

india\_map = gpd.read\_file(gpd.datasets.get\_path('naturalearth\_lowres'))

india\_map = india\_map[india\_map.name == "India"] # Filter for India

# Convert DataFrame to GeoDataFrame

gdf = gpd.GeoDataFrame(df, geometry=gpd.points\_from\_xy(df['Longitude'], df['Latitude']))

# Plot the map

fig, ax = plt.subplots(figsize=(10, 10))

india\_map.plot(ax=ax, color='lightgrey', edgecolor='black') # Plot India map

gdf.plot(ax=ax, color='red', markersize=10, alpha=0.6) # Plot sensor locations

# Labels and title

plt.xlabel("Longitude")

plt.ylabel("Latitude")

plt.title("Air Quality Monitoring Stations Across India")

plt.show()

37/17:

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

# Load datasets

df\_pollution = pd.read\_csv(path1)

df\_ncap = pd.read\_csv(path3)

df\_pollution['Timestamp'] = pd.to\_datetime(df\_pollution['Timestamp'], format='%Y-%m-%d')

df\_ncap.rename(columns={'Amount released during FY 2020-21': 'Funding\_2020'}, inplace=True)

# Identify the state with highest NCAP funding in 2020

state\_highest\_funding = df\_ncap.groupby("State")['Funding\_2020'].sum().idxmax()

highest\_funding\_amount = df\_ncap.groupby("State")['Funding\_2020'].sum().max()

print(f"The state with the highest NCAP funding in 2020: {state\_highest\_funding}")

# pollution data for cities in that state

df\_state\_cities = df\_ncap[df\_ncap['State'] == state\_highest\_funding]['City'].unique()

df\_state\_pollution = df\_pollution[df\_pollution['city'].isin(df\_state\_cities)].copy()

# Yearly PM2.5 averages (2018-2023)

df\_state\_pollution['Year'] = df\_state\_pollution['Timestamp'].dt.year

yearly\_pm25 = df\_state\_pollution.groupby('Year')['PM2.5'].mean()

# Plot

plt.figure(figsize=(10, 5))

sns.lineplot(x=yearly\_pm25.index, y=yearly\_pm25.values, marker='o', label=state\_highest\_funding, color='blue')

# Mark NCAP intervention year (2020)

plt.axvline(x=2020, linestyle='--', color='red', label='NCAP Funding (2020)')

plt.xlabel("Year")

plt.ylabel("Average PM2.5")

plt.title(f"PM2.5 Trends in {state\_highest\_funding} (2018-2023) - Impact of NCAP Funding")

plt.legend()

plt.grid(True)

plt.xticks(range(2018, 2024))

plt.show()

37/18:

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

# Load datasets

df\_population = pd.read\_csv(path1)

df\_ncap = pd.read\_csv(path3)

# Rename columns for consistency

df\_population.rename(columns={'Area (km2)': 'Area\_sq\_km'}, inplace=True)

df\_ncap.rename(columns={'Total fund released': 'Total\_Funding\_Crores'}, inplace=True)

# Group NCAP data by state to get total funding per state

df\_ncap\_statewise = df\_ncap.groupby('State', as\_index=False)['Total\_Funding\_Crores'].sum()

# Merge datasets based on State

df\_merged = pd.merge(df\_population, df\_ncap\_statewise, on="State", how="inner")

# Set style

sns.set(style="whitegrid")

# Create scatter plot

plt.figure(figsize=(14, 8)) # Adjusted size for better clarity

scatter = sns.scatterplot(

data=df\_merged,

x='Area\_sq\_km',

y='Total\_Funding\_Crores',

hue='State',

palette=sns.color\_palette("tab20", len(df\_merged['State'].unique())),

s=100

)

# Adjust legend position outside the plot

plt.legend(title="State", bbox\_to\_anchor=(1.05, 1), loc='upper left')

# Labels and title

plt.xlabel("State Area (sq km)")

plt.ylabel("Total NCAP Funding (in crores)")

plt.title("State-wise Area vs NCAP Funding Received")

# Show plot

plt.tight\_layout() # Ensure the legend fits outside the plot

plt.show()

37/19:

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

# Load datasets

df\_population = pd.read\_csv(path2)

df\_ncap = pd.read\_csv(path3)

# Rename columns for consistency

df\_population.rename(columns={'Area (km2)': 'Area\_sq\_km'}, inplace=True)

df\_ncap.rename(columns={'Total fund released': 'Total\_Funding\_Crores'}, inplace=True)

# Group NCAP data by state to get total funding per state

df\_ncap\_statewise = df\_ncap.groupby('State', as\_index=False)['Total\_Funding\_Crores'].sum()

# Merge datasets based on State

df\_merged = pd.merge(df\_population, df\_ncap\_statewise, on="State", how="inner")

# Set style

sns.set(style="whitegrid")

# Create scatter plot

plt.figure(figsize=(14, 8)) # Adjusted size for better clarity

scatter = sns.scatterplot(

data=df\_merged,

x='Area\_sq\_km',

y='Total\_Funding\_Crores',

hue='State',

palette=sns.color\_palette("tab20", len(df\_merged['State'].unique())),

s=100

)

# Adjust legend position outside the plot

plt.legend(title="State", bbox\_to\_anchor=(1.05, 1), loc='upper left')

# Labels and title

plt.xlabel("State Area (sq km)")

plt.ylabel("Total NCAP Funding (in crores)")

plt.title("State-wise Area vs NCAP Funding Received")

# Show plot

plt.tight\_layout() # Ensure the legend fits outside the plot

plt.show()

37/20:

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

df = pd.read\_csv(path1)

df['Timestamp'] = pd.to\_datetime(df['Timestamp'], format='%Y-%m-%d')

# Get data for Delhi

df\_delhi = df[df['city'] == 'Delhi'].copy()

# Get year, month, and day

df\_delhi['Year'] = df\_delhi['Timestamp'].dt.year

df\_delhi['Month'] = df\_delhi['Timestamp'].dt.month

df\_delhi['Day'] = df\_delhi['Timestamp'].dt.day

# Get data for the years 2018-2020

df\_delhi = df\_delhi[df\_delhi['Year'].isin([2018, 2019, 2020])]

# Define festival periods

diwali\_months = [10, 11] # October & November

new\_year\_days = [(1, 1)] # January 1st

# Get Diwali and New Year Data

df\_diwali = df\_delhi[df\_delhi['Month'].isin(diwali\_months)].copy()

df\_new\_year = df\_delhi[(df\_delhi['Month'] == 1) & (df\_delhi['Day'] == 1)].copy()

# Plot PM2.5 levels over time

plt.figure(figsize=(14, 6))

sns.lineplot(data=df\_delhi, x='Timestamp', y='PM2.5', label="PM2.5 Levels", color="blue")

# Mark Diwali spikes with arrows

diwali\_spikes = df\_diwali[df\_diwali['PM2.5'] > df\_diwali['PM2.5'].quantile(0.90)] # Top 10% as spikes

for index, row in diwali\_spikes.iterrows():

plt.annotate('Diwali Spike', xy=(row['Timestamp'], row['PM2.5']),

xytext=(row['Timestamp'], row['PM2.5'] + 50),

arrowprops=dict(facecolor='red', shrink=0.05),

fontsize=9, color='red')

# Mark New Year spikes with arrows

for index, row in df\_new\_year.iterrows():

plt.annotate('New Year Spike', xy=(row['Timestamp'], row['PM2.5']),

xytext=(row['Timestamp'], row['PM2.5'] + 50),

arrowprops=dict(facecolor='green', shrink=0.05),

fontsize=9, color='green')

plt.xlabel("Year")

plt.ylabel("PM2.5 (µg/m³)")

plt.title("PM2.5 Levels in Delhi (2018-2020) - Diwali & New Year Spikes")

plt.legend()

plt.xticks(rotation=45)

plt.show()

37/21:

from matplotlib.ticker import MaxNLocator

df\_funding = pd.read\_csv(path3)

df\_funding\_assam = df\_funding[df\_funding["State"] == 'Assam']

df\_PM\_assam = df[df["state"] == "Assam"]

#df\_funding\_assam

df\_PM\_assam["year"] = df\_PM\_assam["Timestamp"].dt.year

df\_PM\_assam = df\_PM\_assam.groupby('year')["PM2.5"].mean()

#df\_funding\_assam\_cleaned = df\_funding\_assam.dropna()

funding\_cols = [

"Amount released during FY 2019-20",

"Amount released during FY 2020-21",

"Amount released during FY 2021-22",

]

final\_funding\_cols = []

#dropping the columns which only have NaN values:

for col in funding\_cols:

if df\_funding\_assam[col].count() > 0:

final\_funding\_cols.append(col)

funding\_per\_year = {}

for i, column in enumerate(final\_funding\_cols, start=2019):

funding\_per\_year[i] = df\_funding\_assam[column].sum()

df\_funding\_assam\_grouped = pd.DataFrame(list(funding\_per\_year.items()), columns=["year", "NCAP Funding"])

df\_combined = pd.merge(df\_PM\_assam, df\_funding\_assam\_grouped, on="year", how="inner")

df\_combined

#avoid scientific notation

plt.ticklabel\_format(useOffset=False)

#keep only integer ticks

plt.gca().xaxis.set\_major\_locator(MaxNLocator(integer=True))

plt.plot(df\_combined["year"].tolist(), df\_combined["PM2.5"].tolist(), label = "PM 2.5 levels")

plt.plot(df\_combined["year"].tolist(), df\_combined["NCAP Funding"].tolist(), label = "NCAP Funding")

plt.legend()

plt.xlabel("Years")

plt.grid(True)

plt.show()

37/22:

from matplotlib.ticker import MaxNLocator

df\_funding = pd.read\_csv(path3)

df\_funding\_assam = df\_funding[df\_funding["State"] == 'Assam']

df\_PM\_assam = df[df["state"] == "Assam"].copy()

#df\_funding\_assam

df\_PM\_assam["year"] = df\_PM\_assam["Timestamp"].dt.year

df\_PM\_assam = df\_PM\_assam.groupby('year')["PM2.5"].mean()

#df\_funding\_assam\_cleaned = df\_funding\_assam.dropna()

funding\_cols = [

"Amount released during FY 2019-20",

"Amount released during FY 2020-21",

"Amount released during FY 2021-22",

]

final\_funding\_cols = []

#dropping the columns which only have NaN values:

for col in funding\_cols:

if df\_funding\_assam[col].count() > 0:

final\_funding\_cols.append(col)

funding\_per\_year = {}

for i, column in enumerate(final\_funding\_cols, start=2019):

funding\_per\_year[i] = df\_funding\_assam[column].sum()

df\_funding\_assam\_grouped = pd.DataFrame(list(funding\_per\_year.items()), columns=["year", "NCAP Funding"])

df\_combined = pd.merge(df\_PM\_assam, df\_funding\_assam\_grouped, on="year", how="inner")

df\_combined

#avoid scientific notation

plt.ticklabel\_format(useOffset=False)

#keep only integer ticks

plt.gca().xaxis.set\_major\_locator(MaxNLocator(integer=True))

plt.plot(df\_combined["year"].tolist(), df\_combined["PM2.5"].tolist(), label = "PM 2.5 levels")

plt.plot(df\_combined["year"].tolist(), df\_combined["NCAP Funding"].tolist(), label = "NCAP Funding")

plt.legend()

plt.xlabel("Years")

plt.grid(True)

plt.show()

37/23:

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

# Assuming the necessary data is already loaded in the following dataframes:

# df (contains PM2.5 data) and df\_stations (contains station data with 'State' and 'Station ID')

# 1. Calculate the station density for each state (stations per unit area)

df\_stations = pd.read\_csv(r"C:\path\to\station\_data.csv") # Station data with 'State' and 'Area'

df\_stations['Station\_Density'] = df\_stations.groupby('State')['Station ID'].transform('count') / df\_stations.groupby('State')['Area'].transform('mean')

# 2. Calculate the variance (as a proxy for accuracy) of PM2.5 levels across stations in each state

df\_pm25 = pd.read\_csv(r"C:\path\to\pm25\_data.csv") # Data with 'State', 'Station ID', and 'PM2.5'

pm25\_variance = df\_pm25.groupby('State')['PM2.5'].var()

# Merge the station density and PM2.5 variance data

state\_data = pd.DataFrame({

'Station\_Density': df\_stations.groupby('State')['Station\_Density'].first(),

'PM2.5\_Variance': pm25\_variance

}).dropna() # Drop rows with missing values

# 3. Plot the relationship between station density and PM2.5 variance

plt.figure(figsize=(10, 6))

sns.scatterplot(data=state\_data, x='Station\_Density', y='PM2.5\_Variance', palette='viridis')

plt.title('Station Density vs. PM2.5 Variance (Accuracy of Measurements)')

plt.xlabel('Station Density (Stations per unit area)')

plt.ylabel('PM2.5 Variance (Across stations in state)')

plt.grid(True)

plt.show()

# Optionally: Calculate correlation

correlation = state\_data['Station\_Density'].corr(state\_data['PM2.5\_Variance'])

print(f"Correlation between station density and PM2.5 variance: {correlation:.2f}")

37/24:

import pandas as pd

# Define the paths to the datasets

path1 = r"C:\path\to\Data.csv" # CPCB Air Pollution Data

path2 = r"C:\path\to\NCAP\_Funding.csv" # NCAP Funding for Indian Cities

path3 = r"C:\path\to\State\_data.csv" # State-wise Population and Area

# Load the datasets

df\_pollution = pd.read\_csv(path1)

df\_funding = pd.read\_csv(path2)

df\_state\_data = pd.read\_csv(path3)

# Inspect the first few rows of each dataset

print("Pollution Data:")

print(df\_pollution.head())

print("\nNCAP Funding Data:")

print(df\_funding.head())

print("\nState-wise Data:")

print(df\_state\_data.head())

# Clean up: Handle any missing or incorrect values if necessary

# You can check for missing values

print("\nMissing values in Pollution Data:")

print(df\_pollution.isnull().sum())

print("\nMissing values in NCAP Funding Data:")

print(df\_funding.isnull().sum())

print("\nMissing values in State Data:")

print(df\_state\_data.isnull().sum())

# Merge the Pollution and NCAP Funding Data based on State and City

# Merge by 'State' and 'City' (for matching cities within states)

merged\_data = pd.merge(df\_pollution, df\_funding, on=["State", "City"], how="left")

# Merge the merged data with State-wise Population and Area Data based on 'State'

merged\_data = pd.merge(merged\_data, df\_state\_data, on="State", how="left")

# Inspect the merged data

print("\nMerged Data:")

print(merged\_data.head())

# Save the merged data for future analysis (if required)

merged\_data.to\_csv(r"C:\path\to\merged\_data.csv", index=False)

37/25:

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

path1 = r"C:\Users\inees\Downloads\Data.csv" # Data.csv

path2 = r"C:\Users\inees\Downloads\State\_data.csv" # State\_data.csv

path3 = r"C:\Users\inees\Downloads\NCAP\_Funding.csv" # NCAP\_Funding.csv

37/26:

import pandas as pd

# Load the datasets

df\_pollution = pd.read\_csv(path1)

df\_funding = pd.read\_csv(path2)

df\_state\_data = pd.read\_csv(path3)

# Inspect the first few rows of each dataset

print("Pollution Data:")

print(df\_pollution.head())

print("\nNCAP Funding Data:")

print(df\_funding.head())

print("\nState-wise Data:")

print(df\_state\_data.head())

# Clean up: Handle any missing or incorrect values if necessary

# You can check for missing values

print("\nMissing values in Pollution Data:")

print(df\_pollution.isnull().sum())

print("\nMissing values in NCAP Funding Data:")

print(df\_funding.isnull().sum())

print("\nMissing values in State Data:")

print(df\_state\_data.isnull().sum())

# Merge the Pollution and NCAP Funding Data based on State and City

# Merge by 'State' and 'City' (for matching cities within states)

merged\_data = pd.merge(df\_pollution, df\_funding, on=["State", "City"], how="left")

# Merge the merged data with State-wise Population and Area Data based on 'State'

merged\_data = pd.merge(merged\_data, df\_state\_data, on="State", how="left")

# Inspect the merged data

print("\nMerged Data:")

print(merged\_data.head())

# Save the merged data for future analysis (if required)

merged\_data.to\_csv(r"C:\path\to\merged\_data.csv", index=False)

37/27:

import pandas as pd

# Load the datasets

df\_pollution = pd.read\_csv(path1)

df\_pollution['Timestamp'] = pd.to\_datetime(df\_pollution['Timestamp'], format='%Y-%m-%d')

df\_pollution.rename(columns={'state': 'State'}, inplace=True)

df\_funding = pd.read\_csv(path2)

df\_state\_data = pd.read\_csv(path3)

# Inspect the first few rows of each dataset

print("Pollution Data:")

print(df\_pollution.head())

print("\nNCAP Funding Data:")

print(df\_funding.head())

print("\nState-wise Data:")

print(df\_state\_data.head())

# Clean up: Handle any missing or incorrect values if necessary

# You can check for missing values

print("\nMissing values in Pollution Data:")

print(df\_pollution.isnull().sum())

print("\nMissing values in NCAP Funding Data:")

print(df\_funding.isnull().sum())

print("\nMissing values in State Data:")

print(df\_state\_data.isnull().sum())

# Merge the Pollution and NCAP Funding Data based on State and City

# Merge by 'State' and 'City' (for matching cities within states)

merged\_data = pd.merge(df\_pollution, df\_funding, on=["State", "City"], how="left")

# Merge the merged data with State-wise Population and Area Data based on 'State'

merged\_data = pd.merge(merged\_data, df\_state\_data, on="State", how="left")

# Inspect the merged data

print("\nMerged Data:")

print(merged\_data.head())

# Save the merged data for future analysis (if required)

merged\_data.to\_csv(r"C:\path\to\merged\_data.csv", index=False)

37/28:

import pandas as pd

# Load the datasets

df\_pollution = pd.read\_csv(path1)

df\_pollution['Timestamp'] = pd.to\_datetime(df\_pollution['Timestamp'], format='%Y-%m-%d')

df\_pollution.rename(columns={'state': 'State'}, inplace=True)

df\_funding = pd.read\_csv(path2)

df\_state\_data = pd.read\_csv(path3)

# Inspect the first few rows of each dataset

print("Pollution Data:")

print(df\_pollution.head())

print("\nNCAP Funding Data:")

print(df\_funding.head())

print("\nState-wise Data:")

print(df\_state\_data.head())

# Clean up: Handle any missing or incorrect values if necessary

# You can check for missing values

print("\nMissing values in Pollution Data:")

print(df\_pollution.isnull().sum())

print("\nMissing values in NCAP Funding Data:")

print(df\_funding.isnull().sum())

print("\nMissing values in State Data:")

print(df\_state\_data.isnull().sum())

# Merge the Pollution and NCAP Funding Data based on State and City

# Merge by 'State' and 'City' (for matching cities within states)

merged\_data = pd.merge(df\_pollution, df\_funding, on=["State", "city"], how="left")

# Merge the merged data with State-wise Population and Area Data based on 'State'

merged\_data = pd.merge(merged\_data, df\_state\_data, on="State", how="left")

# Inspect the merged data

print("\nMerged Data:")

print(merged\_data.head())

# Save the merged data for future analysis (if required)

merged\_data.to\_csv(r"C:\path\to\merged\_data.csv", index=False)

37/29:

import pandas as pd

# Load the datasets

df\_pollution = pd.read\_csv(path1)

df\_pollution['Timestamp'] = pd.to\_datetime(df\_pollution['Timestamp'], format='%Y-%m-%d')

df\_pollution.rename(columns={'state': 'State'}, inplace=True)

df\_pollution.rename(columns={'city': 'City'}, inplace=True)

df\_funding = pd.read\_csv(path2)

df\_state\_data = pd.read\_csv(path3)

# Merge the Pollution and NCAP Funding Data based on State and City

# Merge by 'State' and 'City' (for matching cities within states)

merged\_data = pd.merge(df\_pollution, df\_funding, on=["State", "City"], how="left")

# Merge the merged data with State-wise Population and Area Data based on 'State'

merged\_data = pd.merge(merged\_data, df\_state\_data, on="State", how="left")

# Inspect the merged data

print("\nMerged Data:")

print(merged\_data.head())

# Save the merged data for future analysis (if required)

merged\_data.to\_csv(r"C:\path\to\merged\_data.csv", index=False)

37/30:

import pandas as pd

# Load the datasets

df\_pollution = pd.read\_csv(path1)

df\_pollution['Timestamp'] = pd.to\_datetime(df\_pollution['Timestamp'], format='%Y-%m-%d')

df\_pollution.rename(columns={'state': 'State'}, inplace=True)

df\_pollution.rename(columns={'city': 'City'}, inplace=True)

df\_funding = pd.read\_csv(path2)

df\_state\_data = pd.read\_csv(path3)

# Merge the Pollution and NCAP Funding Data based on State and City

# Merge by 'State' and 'City' (for matching cities within states)

merged\_data = pd.merge(df\_pollution, df\_funding, on="City", how="left")

# Merge the merged data with State-wise Population and Area Data based on 'State'

merged\_data = pd.merge(merged\_data, df\_state\_data, on="State", how="left")

# Inspect the merged data

print("\nMerged Data:")

print(merged\_data.head())

# Save the merged data for future analysis (if required)

merged\_data.to\_csv(r"C:\path\to\merged\_data.csv", index=False)

37/31:

import pandas as pd

# Load the datasets

df\_pollution = pd.read\_csv(path1)

df\_pollution['Timestamp'] = pd.to\_datetime(df\_pollution['Timestamp'], format='%Y-%m-%d')

df\_pollution.rename(columns={'state': 'State'}, inplace=True)

df\_pollution.rename(columns={'city': 'City'}, inplace=True)

df\_funding = pd.read\_csv(path3)

df\_state\_data = pd.read\_csv(path2)

# Merge the Pollution and NCAP Funding Data based on State and City

# Merge by 'State' and 'City' (for matching cities within states)

merged\_data = pd.merge(df\_pollution, df\_funding, on=["State", "City"], how="left")

# Merge the merged data with State-wise Population and Area Data based on 'State'

merged\_data = pd.merge(merged\_data, df\_state\_data, on="State", how="left")

# Inspect the merged data

print("\nMerged Data:")

print(merged\_data.head())

# Save the merged data for future analysis (if required)

merged\_data.to\_csv(r"C:\path\to\merged\_data.csv", index=False)

37/32:

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

# 1. Load the datasets

df\_stations = pd.read\_csv(r"C:\path\to\Data.csv") # Air pollution data with 'State', 'Station', 'PM2.5'

df\_funding = pd.read\_csv(r"C:\path\to\NCAP\_Funding.csv") # Funding data with 'State', 'City', etc.

df\_state = pd.read\_csv(r"C:\path\to\State\_data.csv") # Population and area data with 'State', 'Population', 'Area (km²)'

# 2. Calculate station density for each state (stations per unit area)

# Extract unique stations per state from the air pollution data

station\_counts = df\_stations.groupby('State')['Station'].nunique()

state\_area = df\_state.set\_index('State')['Area (km²)'] # Set state as index for easy lookup

station\_density = station\_counts / state\_area

# 3. Calculate the variance of PM2.5 levels across stations in each state

# Drop rows with missing PM2.5 data

df\_stations = df\_stations.dropna(subset=['PM2.5'])

pm25\_variance = df\_stations.groupby('State')['PM2.5'].var()

# 4. Merge the station density and PM2.5 variance data

state\_data = pd.DataFrame({

'Station\_Density': station\_density,

'PM2.5\_Variance': pm25\_variance

}).dropna() # Drop rows with missing values

# 5. Merge NCAP funding data to include funding for each state

funding\_data = df\_funding.groupby('State').agg({

'Total Fund Released': 'sum'

}).reset\_index()

# Merge funding data with station density and PM2.5 variance

state\_data = pd.merge(state\_data, funding\_data, on='State', how='left')

# 6. Plot the relationship between station density and PM2.5 variance

plt.figure(figsize=(10, 6))

sns.scatterplot(data=state\_data, x='Station\_Density', y='PM2.5\_Variance', palette='viridis')

plt.title('Station Density vs. PM2.5 Variance (Accuracy of Measurements)')

plt.xlabel('Station Density (Stations per unit area)')

plt.ylabel('PM2.5 Variance (Across stations in state)')

plt.grid(True)

plt.show()

# 7. Optionally: Calculate correlation

correlation = state\_data['Station\_Density'].corr(state\_data['PM2.5\_Variance'])

print(f"Correlation between station density and PM2.5 variance: {correlation:.2f}")

# 8. Plot the relationship between funding and PM2.5 variance

plt.figure(figsize=(10, 6))

sns.scatterplot(data=state\_data, x='Total Fund Released', y='PM2.5\_Variance', palette='viridis')

plt.title('Total Funding vs. PM2.5 Variance')

plt.xlabel('Total Fund Released (in crores)')

plt.ylabel('PM2.5 Variance (Across stations in state)')

plt.grid(True)

plt.show()

# 9. Optionally: Calculate correlation between funding and PM2.5 variance

funding\_correlation = state\_data['Total Fund Released'].corr(state\_data['PM2.5\_Variance'])

print(f"Correlation between funding and PM2.5 variance: {funding\_correlation:.2f}")

37/33:

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

df\_stations = pd.read\_csv(path1)

df\_funding = pd.read\_csv(path3)

df\_state = pd.read\_csv(path2)

df\_stations.rename(columns={'state': 'State'}, inplace=True)

df\_stations['Timestamp'] = pd.to\_datetime(df\_pollution['Timestamp'], format='%Y-%m-%d')

# 2. Calculate station density for each state (stations per unit area)

station\_counts = df\_stations.groupby('State')['Station'].nunique()

state\_area = df\_state.set\_index('State')['Area (km2)']

station\_density = station\_counts / state\_area

# 3. Calculate the variance of PM2.5 levels across stations in each state

# Drop rows with missing PM2.5 data

df\_stations = df\_stations.dropna(subset=['PM2.5'])

pm25\_variance = df\_stations.groupby('State')['PM2.5'].var()

# 4. Merge the station density and PM2.5 variance data

state\_data = pd.DataFrame({

'Station\_Density': station\_density,

'PM2.5\_Variance': pm25\_variance

}).dropna() # Drop rows with missing values

# 5. Merge NCAP funding data to include funding for each state

funding\_data = df\_funding.groupby('State').agg({

'Total Fund Released': 'sum'

}).reset\_index()

# Merge funding data with station density and PM2.5 variance

state\_data = pd.merge(state\_data, funding\_data, on='State', how='left')

# 6. Plot the relationship between station density and PM2.5 variance

plt.figure(figsize=(10, 6))

sns.scatterplot(data=state\_data, x='Station\_Density', y='PM2.5\_Variance', palette='viridis')

plt.title('Station Density vs. PM2.5 Variance (Accuracy of Measurements)')

plt.xlabel('Station Density (Stations per unit area)')

plt.ylabel('PM2.5 Variance (Across stations in state)')

plt.grid(True)

plt.show()

# 7. Optionally: Calculate correlation

correlation = state\_data['Station\_Density'].corr(state\_data['PM2.5\_Variance'])

print(f"Correlation between station density and PM2.5 variance: {correlation:.2f}")

# 8. Plot the relationship between funding and PM2.5 variance

plt.figure(figsize=(10, 6))

sns.scatterplot(data=state\_data, x='Total Fund Released', y='PM2.5\_Variance', palette='viridis')

plt.title('Total Funding vs. PM2.5 Variance')

plt.xlabel('Total Fund Released (in crores)')

plt.ylabel('PM2.5 Variance (Across stations in state)')

plt.grid(True)

plt.show()

# 9. Optionally: Calculate correlation between funding and PM2.5 variance

funding\_correlation = state\_data['Total Fund Released'].corr(state\_data['PM2.5\_Variance'])

print(f"Correlation between funding and PM2.5 variance: {funding\_correlation:.2f}")

37/34:

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

df\_stations = pd.read\_csv(path1)

df\_funding = pd.read\_csv(path3)

df\_state = pd.read\_csv(path2)

df\_stations.rename(columns={'state': 'State'}, inplace=True)

df\_stations['Timestamp'] = pd.to\_datetime(df\_pollution['Timestamp'], format='%Y-%m-%d')

# 2. Calculate station density for each state (stations per unit area)

station\_counts = df\_stations.groupby('State')['station'].nunique()

state\_area = df\_state.set\_index('State')['Area (km2)']

station\_density = station\_counts / state\_area

# 3. Calculate the variance of PM2.5 levels across stations in each state

# Drop rows with missing PM2.5 data

df\_stations = df\_stations.dropna(subset=['PM2.5'])

pm25\_variance = df\_stations.groupby('State')['PM2.5'].var()

# 4. Merge the station density and PM2.5 variance data

state\_data = pd.DataFrame({

'Station\_Density': station\_density,

'PM2.5\_Variance': pm25\_variance

}).dropna() # Drop rows with missing values

# 5. Merge NCAP funding data to include funding for each state

funding\_data = df\_funding.groupby('State').agg({

'Total Fund Released': 'sum'

}).reset\_index()

# Merge funding data with station density and PM2.5 variance

state\_data = pd.merge(state\_data, funding\_data, on='State', how='left')

# 6. Plot the relationship between station density and PM2.5 variance

plt.figure(figsize=(10, 6))

sns.scatterplot(data=state\_data, x='Station\_Density', y='PM2.5\_Variance', palette='viridis')

plt.title('Station Density vs. PM2.5 Variance (Accuracy of Measurements)')

plt.xlabel('Station Density (Stations per unit area)')

plt.ylabel('PM2.5 Variance (Across stations in state)')

plt.grid(True)

plt.show()

# 7. Optionally: Calculate correlation

correlation = state\_data['Station\_Density'].corr(state\_data['PM2.5\_Variance'])

print(f"Correlation between station density and PM2.5 variance: {correlation:.2f}")

# 8. Plot the relationship between funding and PM2.5 variance

plt.figure(figsize=(10, 6))

sns.scatterplot(data=state\_data, x='Total Fund Released', y='PM2.5\_Variance', palette='viridis')

plt.title('Total Funding vs. PM2.5 Variance')

plt.xlabel('Total Fund Released (in crores)')

plt.ylabel('PM2.5 Variance (Across stations in state)')

plt.grid(True)

plt.show()

# 9. Optionally: Calculate correlation between funding and PM2.5 variance

funding\_correlation = state\_data['Total Fund Released'].corr(state\_data['PM2.5\_Variance'])

print(f"Correlation between funding and PM2.5 variance: {funding\_correlation:.2f}")

37/35:

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

df\_stations = pd.read\_csv(path1)

df\_funding = pd.read\_csv(path3)

df\_state = pd.read\_csv(path2)

df\_stations.rename(columns={'state': 'State'}, inplace=True)

df\_stations['Timestamp'] = pd.to\_datetime(df\_pollution['Timestamp'], format='%Y-%m-%d')

# 2. Calculate station density for each state (stations per unit area)

station\_counts = df\_stations.groupby('State')['station'].nunique()

state\_area = df\_state.set\_index('State')['Area (km2)']

station\_density = station\_counts / state\_area

# 3. Calculate the variance of PM2.5 levels across stations in each state

# Drop rows with missing PM2.5 data

df\_stations = df\_stations.dropna(subset=['PM2.5'])

pm25\_variance = df\_stations.groupby('State')['PM2.5'].var()

# 4. Merge the station density and PM2.5 variance data

state\_data = pd.DataFrame({

'Station\_Density': station\_density,

'PM2.5\_Variance': pm25\_variance

}).dropna() # Drop rows with missing values

# 6. Plot the relationship between station density and PM2.5 variance

plt.figure(figsize=(10, 6))

sns.scatterplot(data=state\_data, x='Station\_Density', y='PM2.5\_Variance', palette='viridis')

plt.title('Station Density vs. PM2.5 Variance (Accuracy of Measurements)')

plt.xlabel('Station Density (Stations per unit area)')

plt.ylabel('PM2.5 Variance (Across stations in state)')

plt.grid(True)

plt.show()

37/36:

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

df\_stations = pd.read\_csv(path1)

df\_funding = pd.read\_csv(path3)

df\_state = pd.read\_csv(path2)

df\_stations.rename(columns={'state': 'State'}, inplace=True)

df\_stations['Timestamp'] = pd.to\_datetime(df\_pollution['Timestamp'], format='%Y-%m-%d')

# 2. Calculate station density for each state (stations per unit area)

station\_counts = df\_stations.groupby('State')['station'].nunique()

state\_area = df\_state.set\_index('State')['Area (km2)']

station\_density = station\_counts / state\_area

# 3. Calculate the variance of PM2.5 levels across stations in each state

# Drop rows with missing PM2.5 data

df\_stations = df\_stations.dropna(subset=['PM2.5'])

pm25\_variance = df\_stations.groupby('State')['PM2.5'].var()

# 4. Merge the station density and PM2.5 variance data

state\_data = pd.DataFrame({

'Station\_Density': station\_density,

'PM2.5\_Variance': pm25\_variance

}).dropna() # Drop rows with missing values

# 6. Plot the relationship between station density and PM2.5 variance

plt.figure(figsize=(10, 6))

sns.scatterplot(data=state\_data, x='Station\_Density', y='PM2.5\_Variance', hue='State')

plt.title('Station Density vs. PM2.5 Variance (Accuracy of Measurements)')

plt.xlabel('Station Density (Stations per unit area)')

plt.ylabel('PM2.5 Variance (Across stations in state)')

plt.grid(True)

plt.show()

37/37:

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

df = pd.read\_csv(path1)

df['Timestamp'] = pd.to\_datetime(df['Timestamp'], format='%Y-%m-%d')

# Get data for Delhi

df\_delhi = df[df['city'] == 'Delhi'].copy()

# Get year, month, and day

df\_delhi['Year'] = df\_delhi['Timestamp'].dt.year

df\_delhi['Month'] = df\_delhi['Timestamp'].dt.month

df\_delhi['Day'] = df\_delhi['Timestamp'].dt.day

# Get data for the years 2018-2020

df\_delhi = df\_delhi[df\_delhi['Year'].isin([2018, 2019, 2020])]

# Define festival periods

diwali\_months = [10, 11] # October & November

new\_year\_days = [(1, 1)] # January 1st

# Get Diwali and New Year Data

df\_diwali = df\_delhi[df\_delhi['Month'].isin(diwali\_months)].copy()

df\_new\_year = df\_delhi[(df\_delhi['Month'] == 1) & (df\_delhi['Day'] == 1)].copy()

# Plot PM2.5 levels over time

plt.figure(figsize=(14, 6))

sns.lineplot(data=df\_delhi, x='Timestamp', y='PM2.5', label="PM2.5 Levels", color="blue")

# Mark Diwali spikes with arrows

diwali\_spikes = df\_diwali[df\_diwali['PM2.5'] > df\_diwali['PM2.5'].quantile(0.90)] # Top 10% as spikes

for index, row in diwali\_spikes.iterrows():

plt.annotate('Diwali Spike', xy=(row['Timestamp'], row['PM2.5']),

xytext=(row['Timestamp'], row['PM2.5'] + 50),

arrowprops=dict(facecolor='red', shrink=0.05),

fontsize=9, color='red')

# Mark New Year spikes with arrows

for index, row in df\_new\_year.iterrows():

plt.annotate('New Year Spike', xy=(row['Timestamp'], row['PM2.5']),

xytext=(row['Timestamp'], row['PM2.5'] + 50),

arrowprops=dict(facecolor='green', shrink=0.05),

fontsize=9, color='green')

plt.xlabel("Year")

plt.ylabel("PM2.5 (µg/m³)")

plt.title("PM2.5 Levels in Delhi (2018-2020) - Diwali & New Year Spikes")

plt.legend()

plt.xticks(rotation=45)

plt.show()

37/38:

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

df = pd.read\_csv(path1)

df['Timestamp'] = pd.to\_datetime(df['Timestamp'], format='%Y-%m-%d')

# Get data for Delhi

df\_delhi = df[df['city'] == 'Delhi'].copy()

# Get year, month, and day

df\_delhi['Year'] = df\_delhi['Timestamp'].dt.year

df\_delhi['Month'] = df\_delhi['Timestamp'].dt.month

df\_delhi['Day'] = df\_delhi['Timestamp'].dt.day

# Get data for the years 2018-2020

df\_delhi = df\_delhi[df\_delhi['Year'].isin([2018, 2019, 2020])]

# Define festival periods

diwali\_months = [10, 11] # October & November

new\_year\_days = [(1, 1)] # January 1st

# Get Diwali and New Year Data

df\_diwali = df\_delhi[df\_delhi['Month'].isin(diwali\_months)].copy()

df\_new\_year = df\_delhi[(df\_delhi['Month'] == 1) & (df\_delhi['Day'] == 1)].copy()

# Plot PM2.5 levels over time

plt.figure(figsize=(14, 6))

sns.lineplot(data=df\_delhi, x='Timestamp', y='PM2.5', label="PM2.5 Levels", color="blue")

# Mark Diwali spikes with arrows

diwali\_spikes = df\_diwali[df\_diwali['PM2.5'] > df\_diwali['PM2.5'].quantile(0.90)] # Top 10% as spikes

for index, row in diwali\_spikes.iterrows():

plt.annotate('Diwali Spike', xy=(row['Timestamp'], row['PM2.5']),

xytext=(row['Timestamp'], row['PM2.5'] + 50),

arrowprops=dict(facecolor='red', shrink=0.05),

fontsize=9, color='red')

# Mark New Year spikes with arrows

for index, row in df\_new\_year.iterrows():

plt.annotate('New Year Spike', xy=(row['Timestamp'], row['PM2.5']),

xytext=(row['Timestamp'], row['PM2.5'] + 50),

arrowprops=dict(facecolor='green', shrink=0.05),

fontsize=9, color='green')

plt.xlabel("Year")

plt.ylabel("PM2.5 (µg/m³)")

plt.title("PM2.5 Levels in Delhi (2018-2020) - Diwali & New Year Spikes")

plt.legend()

plt.xticks(rotation=45)

plt.show()

37/39:

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

# Load datasets

df\_pollution = pd.read\_csv(path1)

df\_ncap = pd.read\_csv(path3)

df\_pollution['Timestamp'] = pd.to\_datetime(df\_pollution['Timestamp'], format='%Y-%m-%d')

df\_ncap.rename(columns={'Amount released during FY 2020-21': 'Funding\_2020'}, inplace=True)

# Identify the state with highest NCAP funding in 2020

state\_highest\_funding = df\_ncap.groupby("State")['Funding\_2020'].sum().idxmax()

highest\_funding\_amount = df\_ncap.groupby("State")['Funding\_2020'].sum().max()

print(f"The state with the highest NCAP funding in 2020: {state\_highest\_funding}")

# pollution data for cities in that state

df\_state\_cities = df\_ncap[df\_ncap['State'] == state\_highest\_funding]['City'].unique()

df\_state\_pollution = df\_pollution[df\_pollution['city'].isin(df\_state\_cities)].copy()

# Yearly PM2.5 averages (2018-2023)

df\_state\_pollution['Year'] = df\_state\_pollution['Timestamp'].dt.year

yearly\_pm25 = df\_state\_pollution.groupby('Year')['PM2.5'].mean()

# Plot

plt.figure(figsize=(10, 5))

sns.lineplot(x=yearly\_pm25.index, y=yearly\_pm25.values, marker='o', label=state\_highest\_funding, color='blue')

# Mark NCAP intervention year (2020)

plt.axvline(x=2020, linestyle='--', color='red', label='NCAP Funding (2020)')

plt.xlabel("Year")

plt.ylabel("Average PM2.5")

plt.title(f"PM2.5 Trends in {state\_highest\_funding} (2018-2023) - Impact of NCAP Funding")

plt.legend()

plt.grid(True)

plt.xticks(range(2018, 2024))

plt.show()