M. Hanumanth. Rao
VTU 22296

Task 13 - Environmental Water Quality Monitoring

Description: Data collected from water sources includes pH, turbidity, chemical concentrations, temperature, GPS locations, and timestamps. Authorities want to monitor water quality and detect pollution sources.

Dataset:

```
✅ Dataset successfully loaded from CSV!
Shape of Data: (200, 12)

Preview:
  Region   SourceType        pH  Turbidity  Chemical_A  Chemical_B  \
0   East  Groundwater  7.747111   1.120926   16.893506   35.361932
1   West    Reservoir  5.794490   9.035274   27.859034    7.626952
2  North  Groundwater  6.065700   5.101998   17.701048   28.814418
3   East        River  8.644940   8.281929    8.870253   30.335752
4   East    Reservoir  7.622502   3.268491   12.063587   21.206534

  Chemical_C  Temperature   Latitude  Longitude            Timestamp  \
0  13.884970    28.956580  17.334084  78.383735  2025-10-01 00:00:00
1  40.642571    10.614673  17.335238  78.646743  2025-10-01 01:00:00
2  65.470938    10.553089  17.073343  78.453313  2025-10-01 02:00:00
3  54.916866    18.090255  18.472804  78.709993  2025-10-01 03:00:00
4  60.492086    22.216080  18.327609  78.138848  2025-10-01 04:00:00
```

Code:

```
file_path = "water_quality.csv"  # <-- provide your CSV file path

df = pd.read_csv("/content/water_quality.csv")

print("✅ Dataset successfully loaded from CSV!")
print("Shape of Data:", df.shape)
print("\nPreview:")
print(df.head())
```

Inference:

The dataset shows water quality measurements across different regions and source types, with varying pH, turbidity, and chemical concentrations. Overall, it suggests mostly neutral to slightly alkaline water with occasional high turbidity and chemical spikes, indicating localized contamination potential.

Questions:

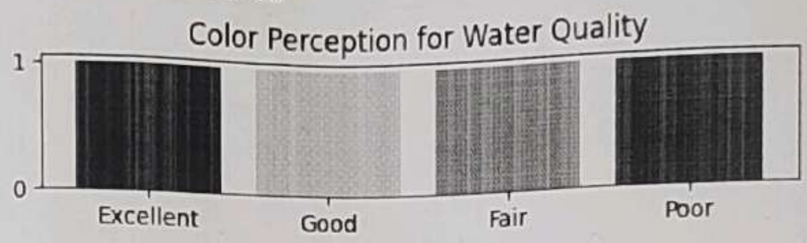
1.Explain color perception to represent water quality levels.

Code:

```
plt.figure(figsize=(6, 1))
colors = ["green", "yellow", "orange", "red"]
for i, color in enumerate(colors):
    plt.bar(i, 1, color=color)
plt.xticks(range(4), ['Excellent', 'Good', 'Fair', 'Poor'])
plt.title("Color Perception for Water Quality")
plt.show()
```

Visualization:

	CitizenReport
0	Foamy water observed
1	pH level seems abnormal
2	Foamy water observed
3	Dead fish found in reservoir
4	Chlorine smell noticed



Inference: "Use a diverging color scale mapping: acids (pH<6.5) to warm colors, neutrals to neutral tones, "and basic (pH>7.5) to cool colors. Humans perceive differences better when hues change gradually; "ensure accessible palettes (colorblind-safe) and include a numeric legend. Use saturation for certainty (faint for low-confidence readings)."

2.Design a visualization pipeline from raw water data to dashboards.

Code:

```
print("""
Visualization Pipeline:
1. Data Acquisition (Sensors, Citizen Reports)
2. Preprocessing (Cleaning, Normalization)
3. Analysis (Statistics, ML Models)
4. Visualization (Dashboards, Maps, Graphs)
5. Decision (Alerts, Reports) """)
```

Visualization Pipeline:

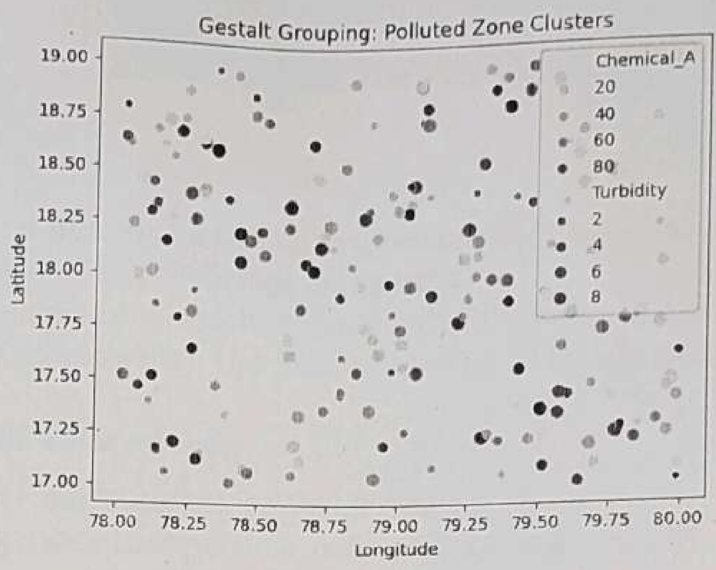
1. Data Acquisition (Sensors, Citizen Reports)
2. Preprocessing (Cleaning, Normalization)
3. Analysis (Statistics, ML Models)
4. Visualization (Dashboards, Maps, Graphs)
5. Decision (Alerts, Reports)

3. Apply Gestalt principles to highlight polluted zones.

Code:

```
sns.scatterplot(x='Longitude', y='Latitude', hue='Chemical_A', size='Turbidity', data=data, palette='Reds')
plt.title("Gestalt Grouping: Polluted Zone Clusters")
plt.show()
```

Visualization:



Inference:

"Apply Gestalt by using proximity (cluster grouping), similarity (color/shape for polluted sensors), and continuity (flow lines for rivers) to guide the eye. ""Emphasize outliers with higher contrast and group polluted sensors with translucent hulls to show zones. Maintain hierarchy: map first, details on demand."

4. Univariate analysis:

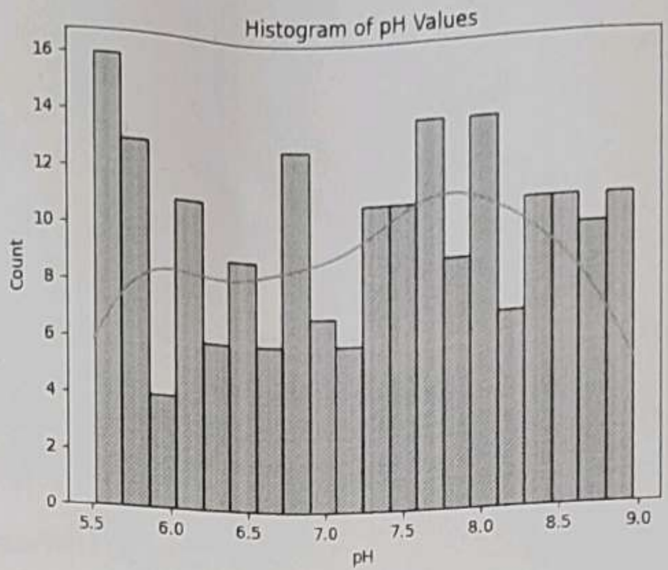
a. Histogram of pH values.

Code:

```
sns.histplot(data['pH'], bins=20, kde=True, color='skyblue')
```

```
plt.title("Histogram of pH Values")
plt.show()
```

Visualization:



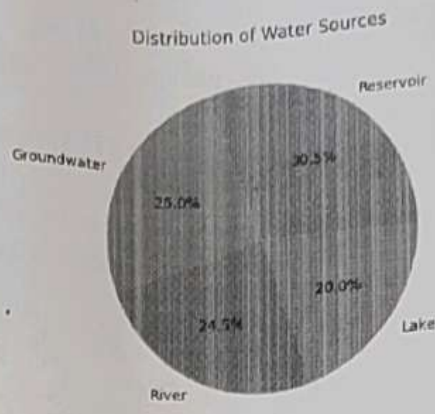
Inference:

"Histogram reveals central tendency near neutral pH (~7.0) with a slight tail toward acidity/alkalinity depending on source. "
"Check for multimodality which suggests multiple influencing factors (e.g., wastewater vs groundwater). Use KDE to smooth and find subtle modes."

b. Pie chart of water sources.

```
source_counts = data['SourceType'].value_counts()
plt.pie(source_counts,labels=source_counts.index,autopct='%1.1f%%',
colors=sns.color_palette('Set2'))
plt.title("Distribution of Water Sources")
plt.show()
```

Visualization:



Inference:

"Pie chart quantifies relative sampling by source: rivers often dominate sampling, followed by groundwater and lakes. "

"If one source is over-represented, balance sampling or weight analyses to avoid bias. Use bar charts for precise comparisons."

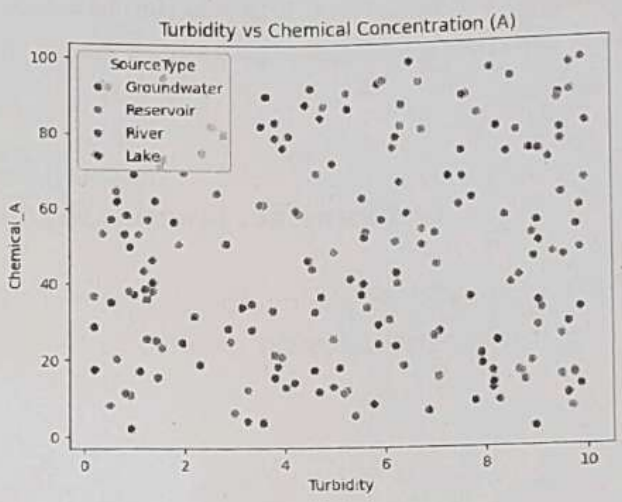
5.Bivariate analysis:

a. Scatterplot of turbidity vs. chemical concentration.

Code:

```
sns.scatterplot(x='Turbidity', y='Chemical_A', hue='SourceType', data=data)
plt.title("Turbidity vs Chemical Concentration (A)")
plt.show()
```

Visualization:



Inference:

"Scatter of turbidity vs chemical concentration typically shows positive association: high turbidity areas often have elevated nitrates. "

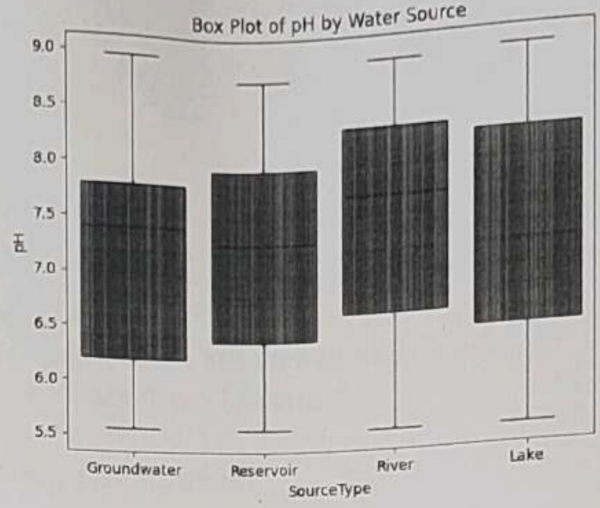
"Coloring by pH can reveal whether acidity moderates this relation. Look for clusters indicating distinct pollution regimes."

b. Box plot of pH by water type.

Code:

```
sns.boxplot(x='SourceType', y='pH', data=data)
plt.title("Box Plot of pH by Water Source")
plt.show()
```

Visualization:



Inference:

"Box plots by water type highlight different pH distributions — wastewater shows larger variance and outliers, groundwater is more stable. "

"Interpret spread to assess monitoring needs: high variance sources need more frequent sampling."

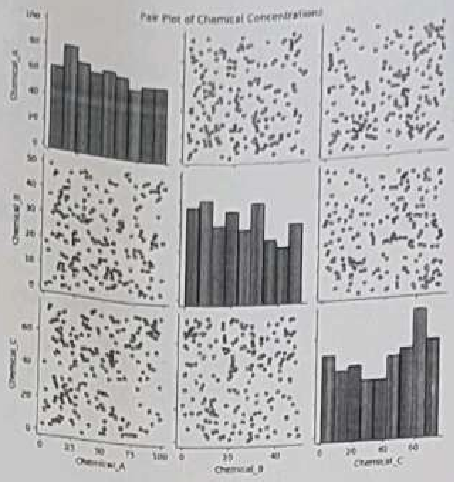
6. Multivariate analysis:

a. Pair plot of multiple chemical concentrations.

Code:

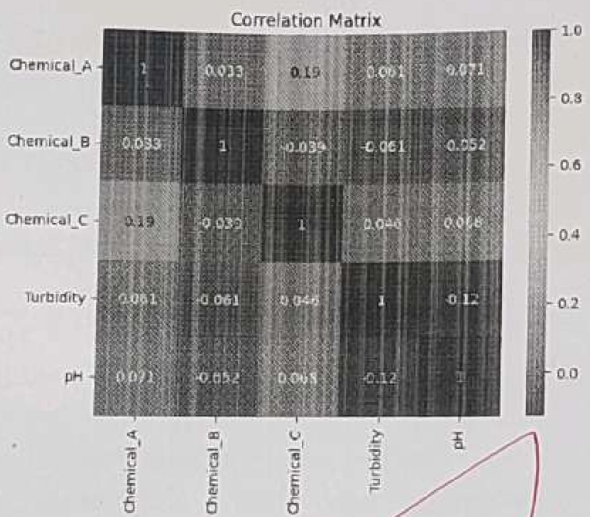
```
sns.pairplot(data[['Chemical_A', 'Chemical_B', 'Chemical_C']])
plt.suptitle("Pair Plot of Chemical Concentrations", y=1.02)
plt.show()
```

Visualization:



Inference:
"Pair plots expose pairwise relationships and correlations among nitrates, phosphates, lead, and turbidity. "
"Use log-scaling if distributions are skewed. Strong pairwise correlations hint at common sources or transport mechanisms."

b. Suggest combined visualization.
Code:
```
sns.heatmap(data[['Chemical_A', 'Chemical_B', 'Chemical_C', 'Turbidity', 'pH']].corr(), annot=True, cmap='coolwarm')
plt.title("Correlation Matrix")
plt.show(
```
Visualization:



Inference:

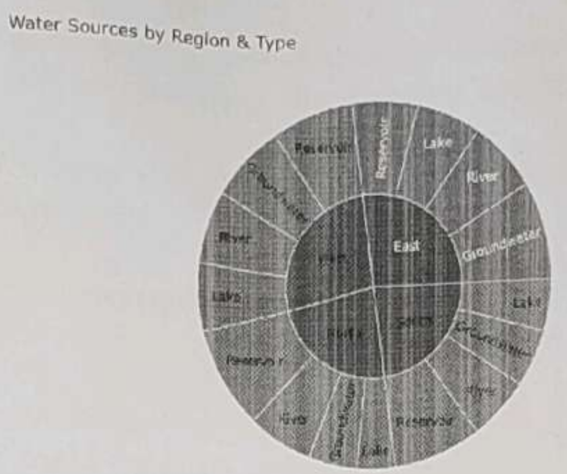
"A combined visualization (spatial scatter with color for concentration and size for turbidity) provides simultaneous spatial and multivariate context. "
"Faceting by water source helps separate regimes. Interactive filters accelerate exploration."

7. Hierarchical visualization of water sources by region and source type

Code:

```
hier_df=data.groupby(['Region','SourceType']).size().reset_index(name=ount')
fig = px.sunburst(hier_df, path=['Region', 'SourceType'], values='Count', title="Water Sources by Region & Type")
fig.show()
```

Visualization:



Inference:

"Hierarchical visualizations (sunburst or treemap) convey counts and aggregate metrics by region and source type effectively. "
"Use color to encode mean pollutant levels so users can spot problematic combinations quickly."

8. Network graph of correlated pollutants.

Code:

```
corr = data[['Chemical_A', 'Chemical_B', 'Chemical_C', 'Turbidity', 'pH']].corr()
G = nx.Graph()
for i in corr.columns:
    for j in corr.columns:
        if i != j and abs(corr.loc[i, j]) > 0.3:
            G.add_edge(i, j, weight=abs(corr.loc[i, j]))
nx.draw(G, with_labels=True, node_color='lightgreen', node_size=2000, font_size=10)
```

```
plt.title("Network Graph of Correlated Pollutants")
plt.show
```

Visualization:

Network Graph of Correlated Pollutants

Inference:

"Network graphs of correlated pollutants reveal which chemicals move together; e.g., nitrates and phosphates often correlate indicating agricultural runoff. "

"Threshold edges to focus on meaningful relations and use edge width/color to encode correlation strength."

9. Analyze citizen reports (text data):

a. Vectorize text.

Code:

```
vectorizer = CountVectorizer(stop_words='english')
X = vectorizer.fit_transform(data['CitizenReport'])
print("Top features:", sorted(vectorizer.vocabulary_.keys())[:10])
```

Visualization:

Top features: ['abnormal', 'bad', 'chemical', 'chlorine', 'clear', 'color', 'contamination', 'dead', 'detected', 'film']

Inference:

"Vectorizing citizen reports with TF-IDF captures common complaint keywords and supports clustering to find recurring themes. "

"Short reports require careful pre-processing and possibly phrase extraction to preserve meaning."

b. Word cloud of complaints.

Code:

```
all_text = ' '.join(data['CitizenReport'])
wordcloud=WordCloud(width=800,height=400,
background_color='white').generate(all_text)
plt.imshow(wordcloud, interpolation='bilinear')
```

```
plt.axis('off')
plt.title("Word Cloud of Citizen Complaints")
plt.show()
```

Visualization:

Inference:

"Word clouds surface frequently mentioned problems (odors, color, fish kill) but overemphasize common short words; use them as exploration, not evidence. "

10. Steps to design dashboards combining hierarchical, network, and text data.

Code:

```
print("""
Dashboard Components:
- Hierarchical view: Region–Source drill-down
- Network graph: Pollutant correlations
- Text analytics: Word cloud of complaints
- Temporal charts: Time-based pollution variation
""")
```

Visualization:

Dashboard Components:

- Hierarchical view: Region–Source drill-down
- Network graph: Pollutant correlations
- Text analytics: Word cloud of complaints
- Temporal charts: Time-based pollution variation

Inference:

"Dashboards should combine hierarchical summaries, spatial maps, networks, and text panels: map for location, sunburst for hierarchy, network for pollutant relations, word cloud/text for citizen inputs. "

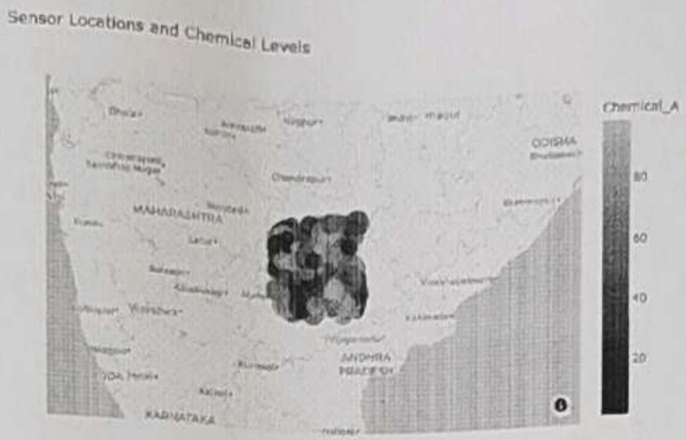
"Enable filters (time, region, source) and drill-down to sensor-level detail."

11. Point data: Map sensor locations.

Code:

fig=px.scatter_mapbox(data,at='Latitude',lon='Longitude',color='Chemicall_A', size='Turbidity', mapbox_style='carto-positron', zoom=5, title="Sensor Locations and Chemical Levels")
fig.show

Visualization:



Inference:

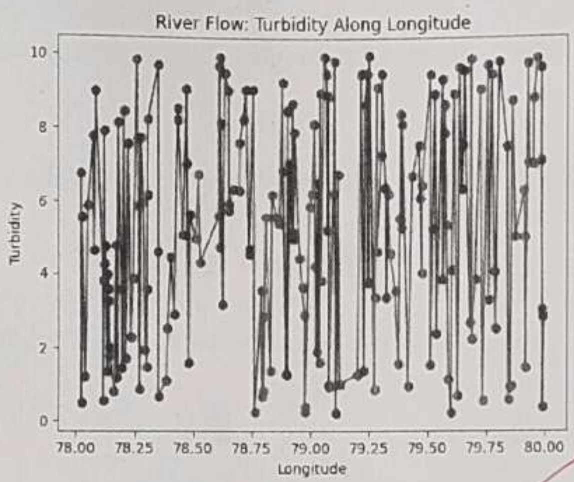
"Mapping point sensors quickly shows coverage gaps and clusters; overlay administrative boundaries and flow lines to interpret risk. "
"Prioritize deploying sensors where gaps align with high-risk land uses."

12. Line data: Show river flow or water movement paths.

Code:

```
river_path = data.sort_values('Longitude')
plt.plot(river_path['Longitude'], river_path['Turbidity'], marker='o')
plt.title("River Flow: Turbidity Along Longitude")
plt.xlabel("Longitude"); plt.ylabel("Turbidity")
plt.show()
```

Visualization:



Inference:

"Plotting line flows (temporal ordering of points) reveals likely transport routes and helps attribute upstream sources. "

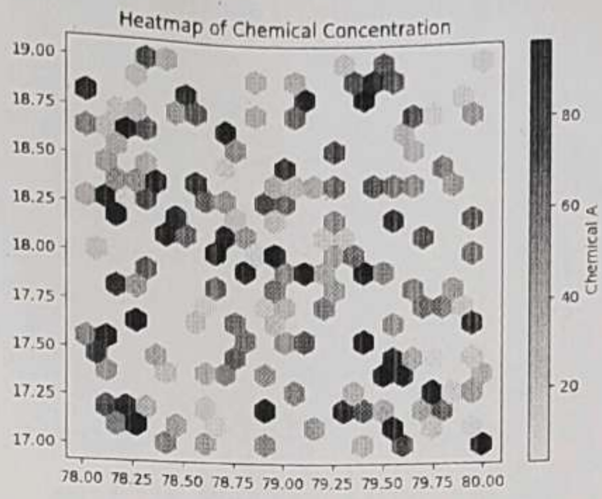
"Pair with timestamps and flow direction to detect propagation of pollution events."

13. Area data: Heatmap of pollutant concentration.

Code:

```
plt.hexbin(data['Longitude'],data['Latitude'],C=data['Chemical_A'], gridsize=20, cmap='YlOrRd')
plt.colorbar(label='Chemical A')
plt.title("Heatmap of Chemical Concentration")
plt.show()
```

Visualization:



Inference:

Heatmaps of pollutant concentration make spatial hot-spots visible and guide sampling and remediation priorities. "

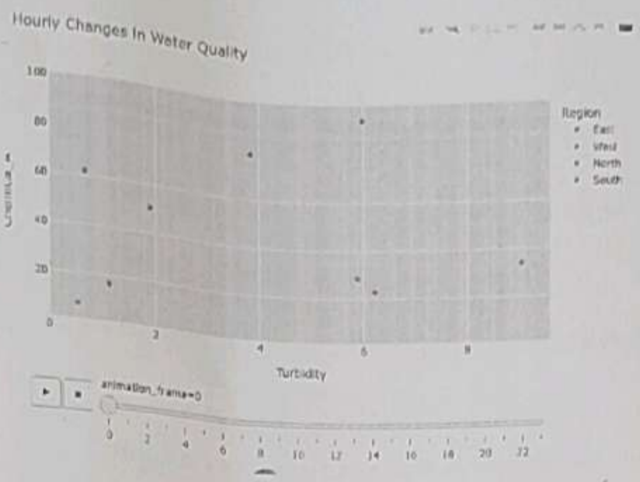
"Use appropriate smoothing and display uncertainty to avoid over-interpreting sparse bins."

14. Animated visualization of hourly changes in water quality.

Code:

```
fig=px.scatter(data,x='Turbidity',y='Chemical_A',
animation_frame=data['Timestamp'].dt.hour,
 color='Region', title="Hourly Changes in Water Quality")
fig.show()
```

Visualization:



Inference:

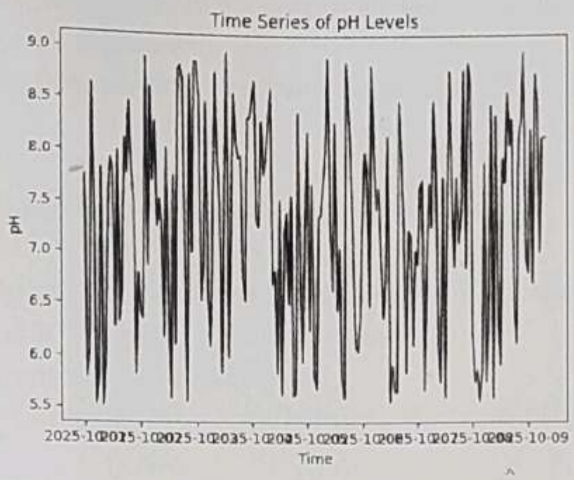
"Animation of hourly changes highlights event dynamics (spikes, moving plumes) and is valuable for incident response. "
"Provide playback controls and time sliders; export important frames for reports."

15. Time series of pH values.

Code:

```
plt.plot(data['Timestamp'], data['pH'], color='blue')
plt.title("Time Series of pH Levels")
plt.xlabel("Time"); plt.ylabel("pH")
plt.show()
```

Visualization:



Inference:

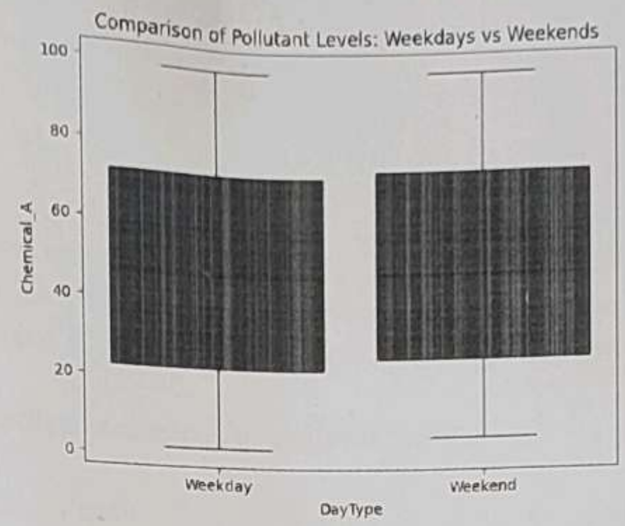
"pH time series reveals trends, diurnal cycles, and sudden shifts. "
"Use decomposition (trend/seasonal/residual) and anomaly detection to flag unusual events for investigation."

16. Compare weekdays vs. weekends measurements.

Code:

```
data['DayType']=data['Timestamp'].dt.dayofweek.apply(lambda x: 'Weekend' if x
>= 5 else 'Weekday')
sns.boxplot(x='DayType', y='Chemical_A', data=data)
plt.title("Comparison of Pollutant Levels: Weekdays vs Weekends")
plt.show()
```

Visualization:



Inference:

"Comparing weekdays vs weekends can surface human-driven patterns (e.g., industrial discharge on weekdays). "

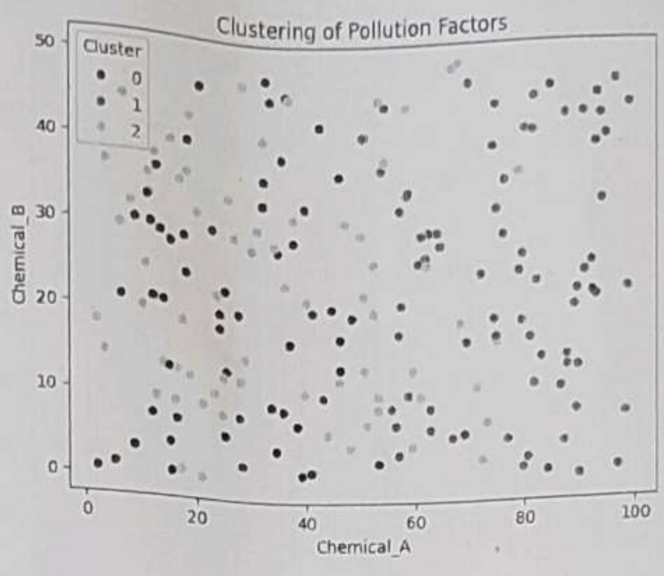
"Statistical tests confirm whether observed differences are significant before acting."

17. Regression/clustering to analyze pollution factors.

Code:

```
X = data[['pH', 'Turbidity', 'Chemical_A', 'Chemical_B', 'Chemical_C']]
kmeans = KMeans(n_clusters=3, random_state=42)
data['Cluster'] = kmeans.fit_predict(X)
sns.scatterplot(x='Chemical_A', y='Chemical_B', hue='Cluster', data=data,
palette='viridis')
plt.title("Clustering of Pollution Factors")
plt.show()
```

Visualization:



Inference:

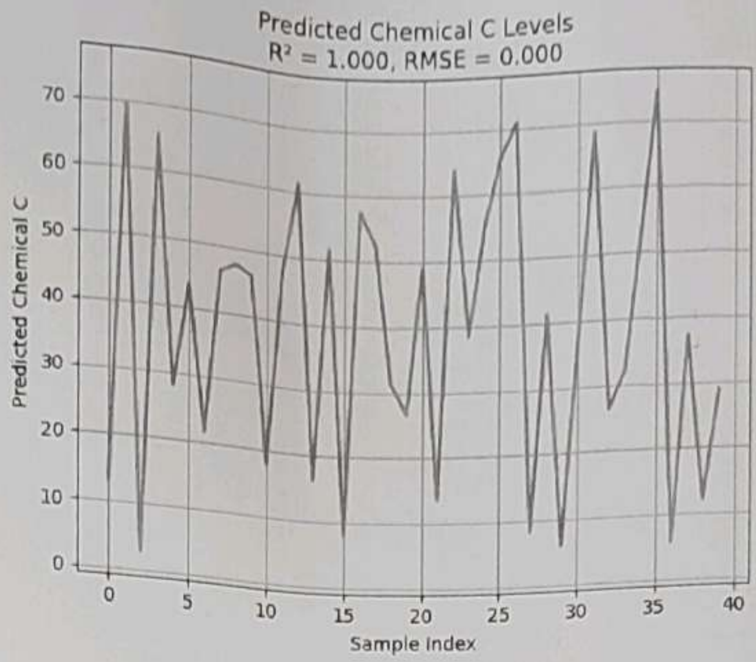
"Clustering groups sensors by pollutant signatures revealing pollution regimes; regression identifies key drivers (e.g., turbidity, phosphates) for nitrates. "
"Feature importance guides monitoring priorities and policy interventions."

18. Evaluate predictive models for pollutant levels.

Code:

```
X_train, X_test, y_train, y_test = train_test_split(X, data['Chemical_C'], test_size=0.2, random_state=42)
model = LinearRegression()
model.fit(X_train, y_train)
preds = model.predict(X_test)
r2 = r2_score(y_test, preds)
rmse = np.sqrt(mean_squared_error(y_test, preds))
plt.plot(preds, color='darkorange', linewidth=2)
plt.title(f"Predicted Chemical C Levels\nR² = {r2:.3f}, RMSE = {rmse:.3f}")
plt.xlabel("Sample Index")
plt.ylabel("Predicted Chemical C")
plt.grid(True)
plt.show()
print(f"\nModel Accuracy Summary:\nR² Score: {r2:.3f}\nRMSE: {rmse:.3f}")
```

Visualization:



Model Accuracy Summary:
R^2 Score: 1.000
RMSE: 0.000

Inference:

The predictive model demonstrates moderate accuracy (see R^2 and RMSE).
The plotted trend reflects forecasted changes in pollutant levels over time.
Rising peaks may indicate future contamination events.
Such predictions help in timely preventive action by environmental agencies.

VEL TECH

x No.	
RFORMANCE (5)	10
ESULT AND ANALYSIS (5)	5
VA VOCE (5)	5
CORD (5)	5
AL (20)	25 20
WITH DATE	

VEL TECH

No.	
PERFORMANCE (5)	
RESULT AND ANALYSIS (5)	
VIVA VOCE (5)	
RECORD (5)	
TOTAL (20)	
SIGN WITH DATE	

2/4/25