Dhruvik mori [ProdigyInfotech]

To analyze traffic accident data and identify patterns related to **road conditions**, **weather**, and **time of day**, we follow a structured data analysis pipeline. Here's a complete walkthrough including data preparation, analysis, and visualizations. If you provide a real dataset (CSV or Excel), I can apply this to your data directly.

Objective

Identify accident hotspots and key contributing factors—like poor weather or nighttime conditions—to improve safety and aid decision-makers.

---

1. Sample Dataset Structure

Assume a dataset with columns:

- Date, Time, Location, Latitude, Longitude
- Weather_Condition (e.g., Clear, Rain, Fog)
- Road_Condition (e.g., Dry, Wet, Snow)
- Accident_Severity (e.g., Minor, Major, Fatal)

---

2. Load and Preprocess Data import

pandas as pd

Load dataset df =

pd.read_csv("traffic_accidents.csv")

Convert date and time df['Date'] = pd.to_datetime(df['Date'])

df['Hour'] = pd.to_datetime(df['Time'], format='%H:%M').dt.hour

Categorize time of day

```
def time_period(hour):
if 5 <= hour < 12:
return 'Morning'     elif
12 <= hour < 17:
return 'Afternoon'     elif
17 <= hour < 21:
return 'Evening'     else:
       return 'Night'

df['Time_of_Day'] = df['Hour'].apply(time_period)
```

---

3. Analyze Patterns

A. Accident Counts by Time of Day

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

sns.countplot(data=df, x='Time_of_Day', order=['Morning', 'Afternoon', 'Evening', 'Night'],
palette='coolwarm') plt.title('Accidents by Time of Day') plt.ylabel('Number of Accidents')
plt.show()
```

B. Accidents by Weather and Road Conditions plt.figure(figsize=(12,

5))

```
sns.countplot(data=df, x='Weather_Condition',
order=df['Weather_Condition'].value_counts().index, palette='Set2') plt.title('Accidents by
Weather Condition') plt.xticks(rotation=45) plt.show()

plt.figure(figsize=(12, 5))
```

```
sns.countplot(data=df, x='Road_Condition', order=df['Road_Condition'].value_counts().index,
palette='Set3') plt.title('Accidents by Road Condition') plt.xticks(rotation=45) plt.show()
```

---

4. Visualize Accident Hotspots (Geospatial)

If you have latitude/longitude, use folium to map accidents.

```
import folium from folium.plugins import HeatMap

# Filter relevant columns and drop nulls map_df
= df[['Latitude', 'Longitude']].dropna()

# Create map centered around median location
m = folium.Map(location=[map_df['Latitude'].median(), map_df['Longitude'].median()],
zoom_start=11)

# Add heatmap layer
HeatMap(data=map_df[['Latitude', 'Longitude']].values, radius=10).add_to(m)

m.save('accident_hotspots_map.html')
```

This creates an interactive heatmap (accident_hotspots_map.html) that can be opened in a browser.

---

5. Correlation of Severity with Conditions

```
severity_ct =
pd.crosstab(index=df['Accident_Severity'], columns=df['Weather_Condition'])
severity_ct.plot(kind='bar', stacked=True, figsize=(10,6), colormap='tab10')
plt.title('Severity vs. Weather Condition') plt.xlabel('Accident Severity') plt.ylabel('Number
of Accidents') plt.legend(title='Weather') plt.show()
```

---

6. Summary & Insights

Key Takeaways:

- **Time of Day:** Accidents peak during **Evening** and **Night**, possibly due to poor visibility and fatigue.
- **Weather Impact:** Rain and fog correlate with more severe accidents.
- **Road Conditions:** Wet or icy roads significantly increase accident risk.
- **Hotspots:** Geospatial visualization reveals high-density accident zones—ideal for installing safety signage or speed controls.
