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

```
df = pd.read_csv('/content/sample_data/Senior Citizen.csv')
```

```
def clean_cols(name):
    name = name.lower()
    name = name.split('(')[0].strip()
    name = name.replace(' ', '_').replace('&', 'and')
    return name
```

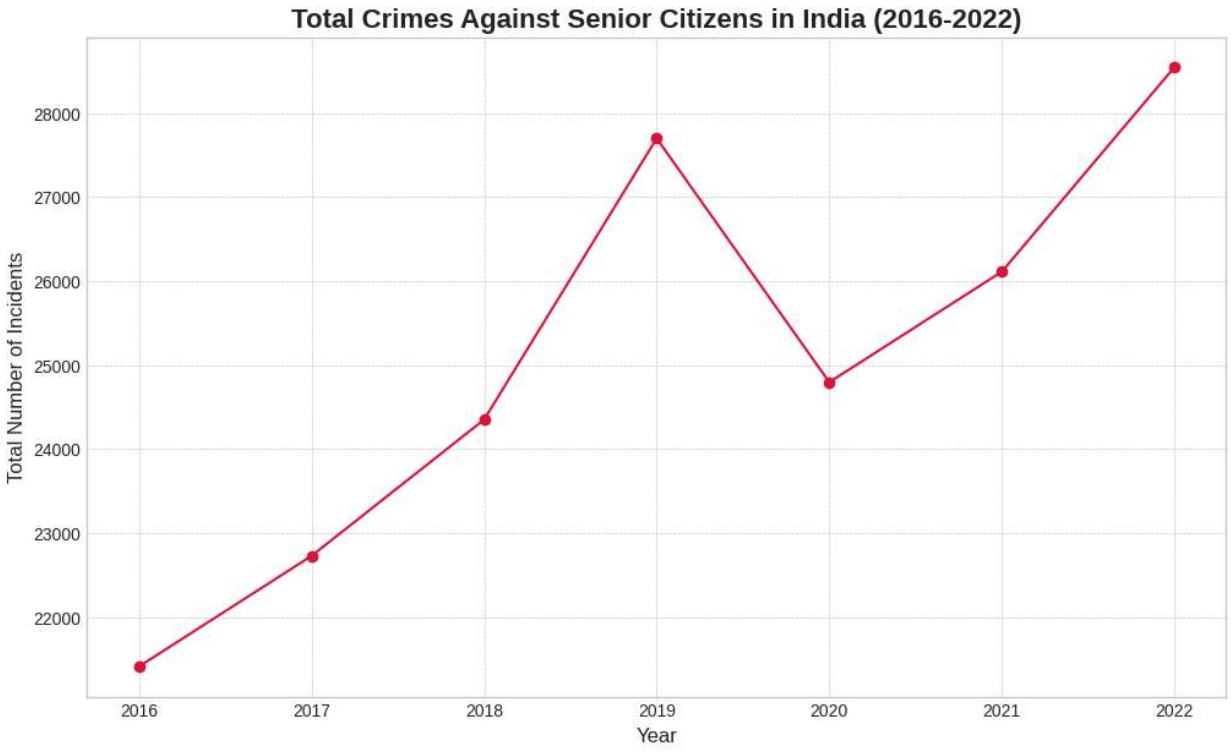
```
df.columns = [clean_cols(col) for col in df.columns]
df.rename(columns={'crime_head': 'crime_head', 'number_of_incidences_or_cases
```

```
df['year'] = df['year'].str.extract(r'(\d{4})').astype(int)
```

```
numeric_cols = ['number_of_incidences', 'number_of_senior_citizen_victims',
for col in numeric_cols:
    df[col] = pd.to_numeric(df[col], errors='coerce').fillna(0)
```

```
df_total = df[df['crime_head'] == 'Total Crimes against Senior Citizen']
yearly_crime = df_total.groupby('year')['number_of_incidences'].sum()
```

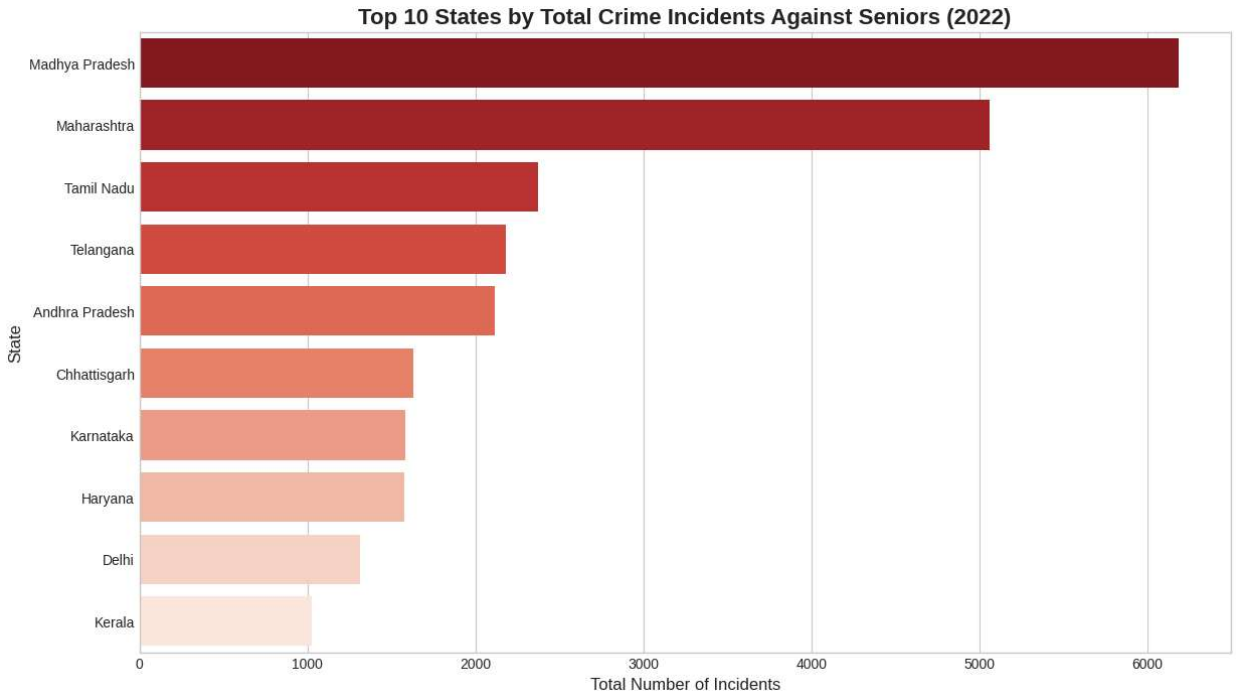
```
plt.style.use('seaborn-v0_8-whitegrid')
fig, ax = plt.subplots(figsize=(12, 7))
yearly_crime.plot(kind='line', marker='o', color='crimson', ax=ax)
ax.set_title('Total Crimes Against Senior Citizens in India (2016-2022)', f
ax.set_xlabel('Year', fontsize=12)
ax.set_ylabel('Total Number of Incidents', fontsize=12)
ax.grid(True, which='both', linestyle='--', linewidth=0.5)
plt.show()
```




```
df_2022_total = df_total[df_total['year'] == 2022]    #States wit Highest N
state_wise_total_2022 = df_2022_total.groupby('state')['number_of_incidence
```

```
fig, ax = plt.subplots(figsize=(14, 8))
sns.barplot(x=state_wise_total_2022.values, y=state_wise_total_2022.index,
ax.set_title('Top 10 States by Total Crime Incidents Against Seniors (2022)
ax.set_xlabel('Total Number of Incidents', fontsize=12)
ax.set_ylabel('State', fontsize=12)
plt.show()
```

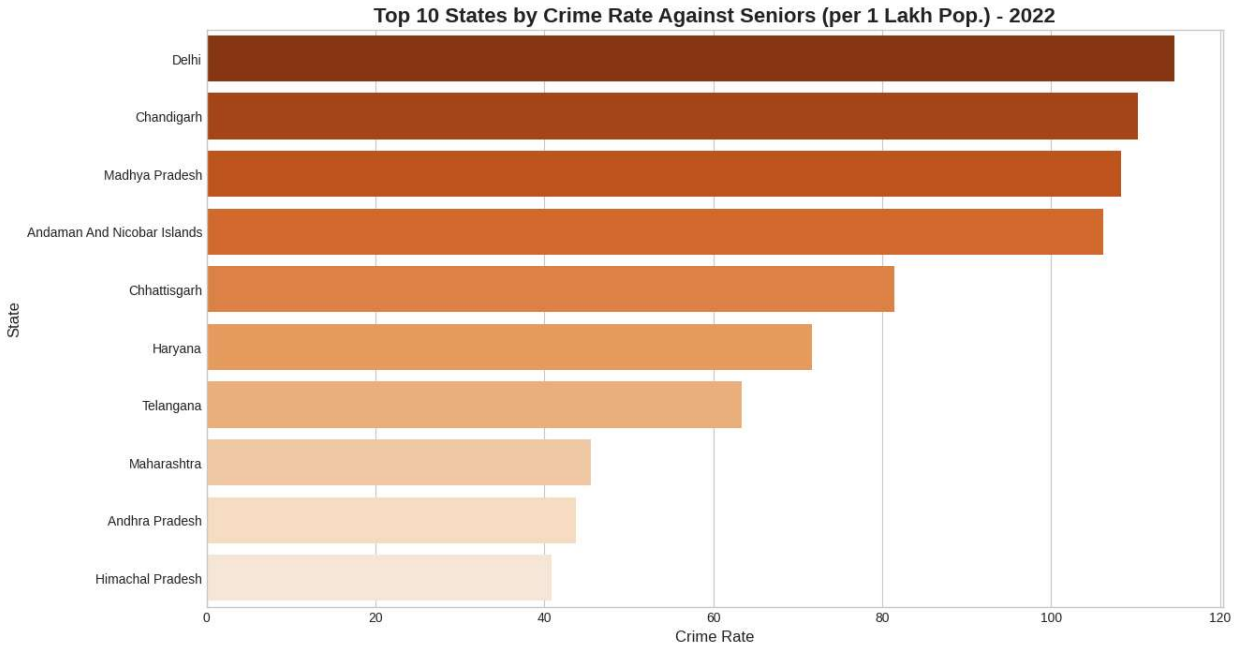
```
/tmp/ipython-input-1939206804.py:2: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y`

  sns.barplot(x=state_wise_total_2022.values, y=state_wise_total_2022.index, palette='Reds_r', ax=ax)
```




```
state_wise_rate_2022 = df_2022_total.set_index('state')['crime_rate_per_lak
```

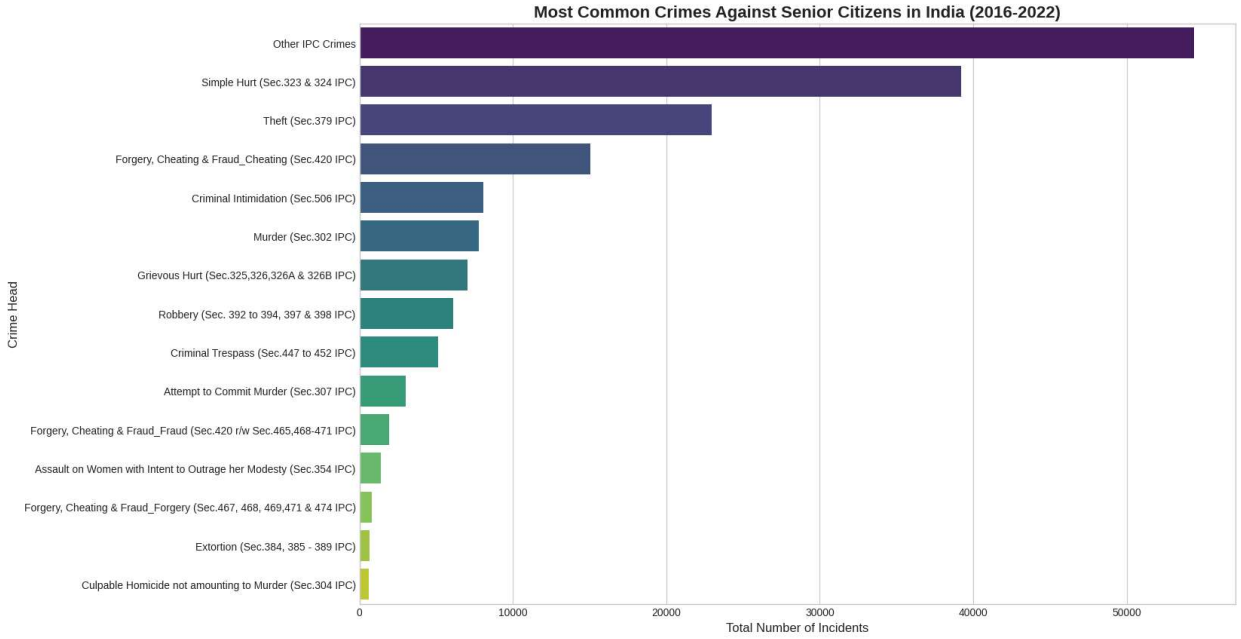
```
fig, ax = plt.subplots(figsize=(14, 8))
sns.barplot(x=state_wise_rate_2022.values, y=state_wise_rate_2022.index, pa
ax.set_title('Top 10 States by Crime Rate Against Seniors (per 1 Lakh Pop.)
ax.set_xlabel('Crime Rate', fontsize=12)
ax.set_ylabel('State', fontsize=12)
plt.show()
```

```
/tmp/ipython-input-2598200985.py:2: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y`

  sns.barplot(x=state_wise_rate_2022.values, y=state_wise_rate_2022.index, palette='Oranges_r', ax=ax)
```




```
df_crimes_only = df[~df['crime_head'].str.contains('Total')]
crime_type_counts = df_crimes_only.groupby('crime_head')['number_of_inciden

# Plotting
fig, ax = plt.subplots(figsize=(15, 10))
sns.barplot(x=crime_type_counts.values, y=crime_type_counts.index, palette=
ax.set_title('Most Common Crimes Against Senior Citizens in India (2016-202
ax.set_xlabel('Total Number of Incidents', fontsize=12)
ax.set_ylabel('Crime Head', fontsize=12)
plt.show()
```

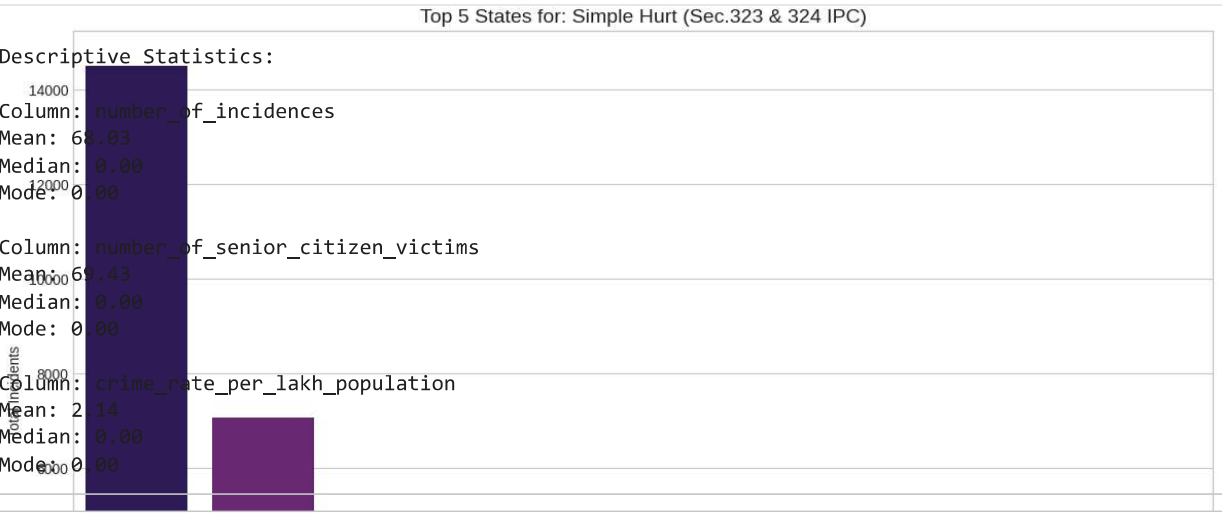
```
/tmp/ipython-input-1473693672.py:6: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y`

  sns.barplot(x=crime_type_counts.values, y=crime_type_counts.index, palette='viridis', ax=ax)
```




```
top_crimes = ['Simple Hurt (Sec.323 & 324 IPC)', 'Theft (Sec.379 IPC)', 'Fo

fig, axes = plt.subplots(3, 1, figsize=(12, 24), sharex=True)
fig.suptitle('State-wise Hotspots for Top 3 Crimes (2016-2022)', fontsize=2

for i, crime in enumerate(top_crimes):
    crime_data = df[df['crime_head'] == crime].groupby('state')['number_of_
    sns.barplot(x=crime_data.index, y=crime_data.values, ax=axes[i], palett
    axes[i].set_title(f'Top 5 States for: {crime}', fontsize=14)
    axes[i].set_ylabel('Total Incidents')
    axes[i].set_xlabel('')

plt.tight_layout(rect=[0, 0, 1, 0.96])
plt.show()
```

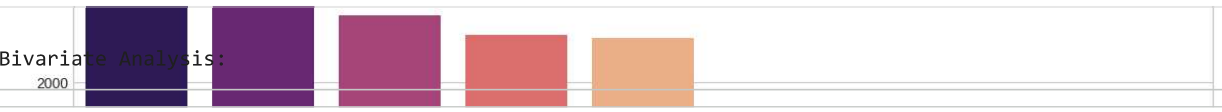
```
/tmp/ipython-input-2793861492.py:8: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x`

  sns.barplot(x=crime_data.index, y=crime_data.values, ax=axes[i], palette='magma')
/tmp/ipython-input-2793861492.py:8: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x`
```

```
# Descriptive Statistics (Mean, Median, Mode)
print("\nDescriptive Statistics:")
for col in numeric_cols:
    print(f"\nColumn: {col}")
    print(f"Mean: {df[col].mean():.2f}")
    print(f"Median: {df[col].median():.2f}")
    print(f"Mode: {df[col].mode()[0]:.2f}")
```



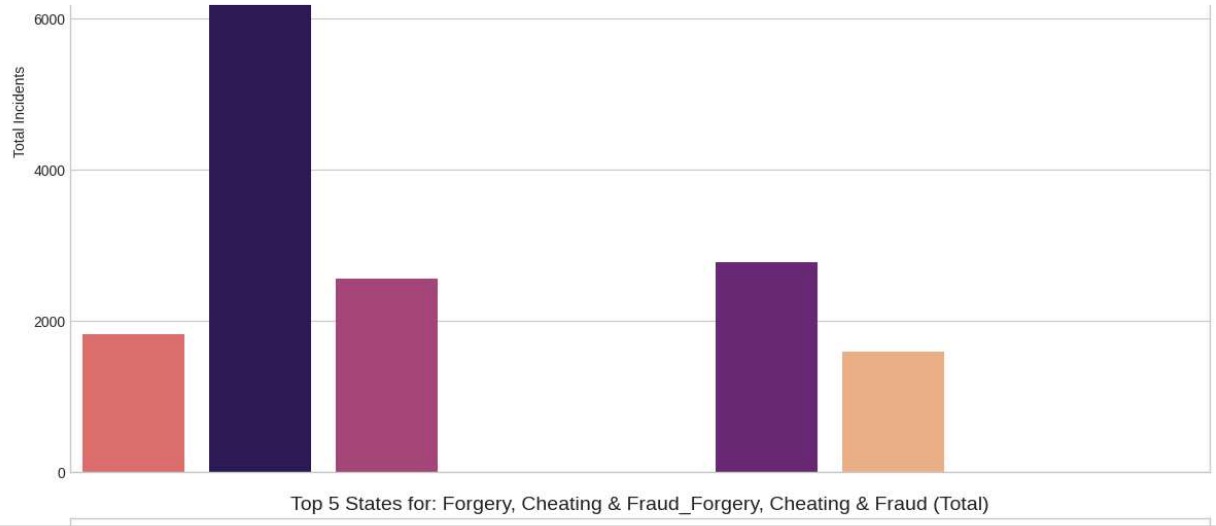

```
Descriptive Statistics:

Column: number_of_incidences
Mean: 68.03
Median: 0.00
Mode: 0.00

Column: number_of_senior_citizen_victims
Mean: 69.43
Median: 0.00
Mode: 0.00

Column: crime_rate_per_lakh_population
Mean: 2.14
Median: 0.00
Mode: 0.00
```

```
print("\nBivariate Analysis:")
```

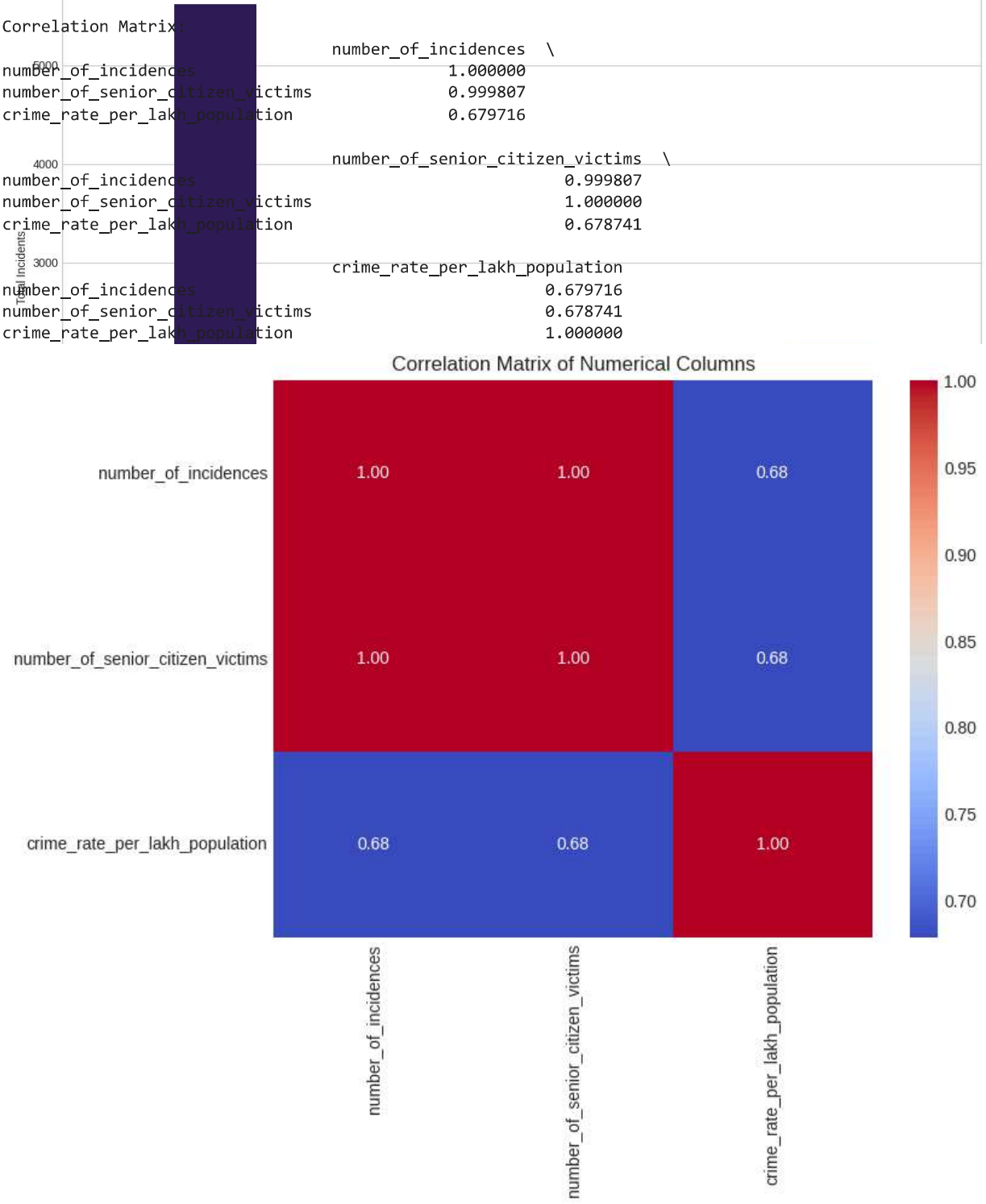



```
Bivariate Analysis:
```

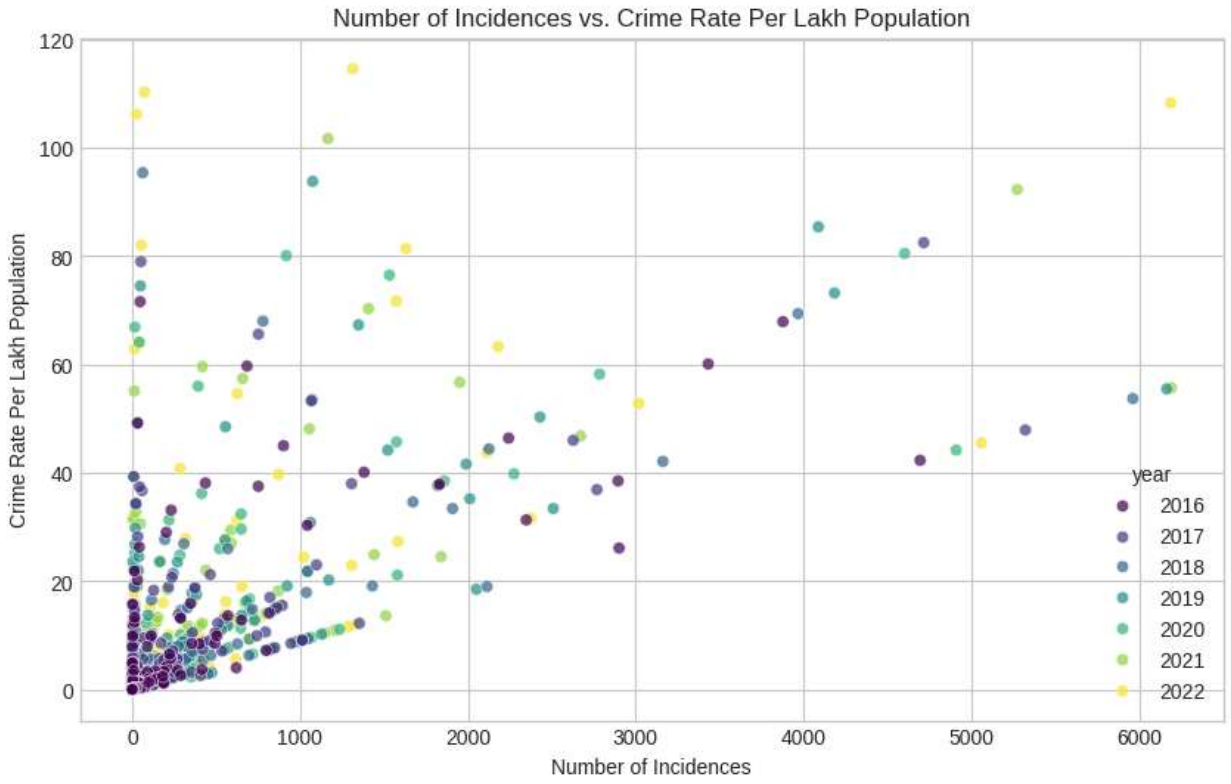
```
correlation_matrix = df[numeric_cols].corr()
print("\nCorrelation Matrix:")
print(correlation_matrix)

plt.figure(figsize=(8, 6))
sns.heatmap(correlation_matrix, annot=True, cmap="coolwarm", fmt=".2f")
plt.title("Correlation Matrix of Numerical Columns")
plt.show()
```



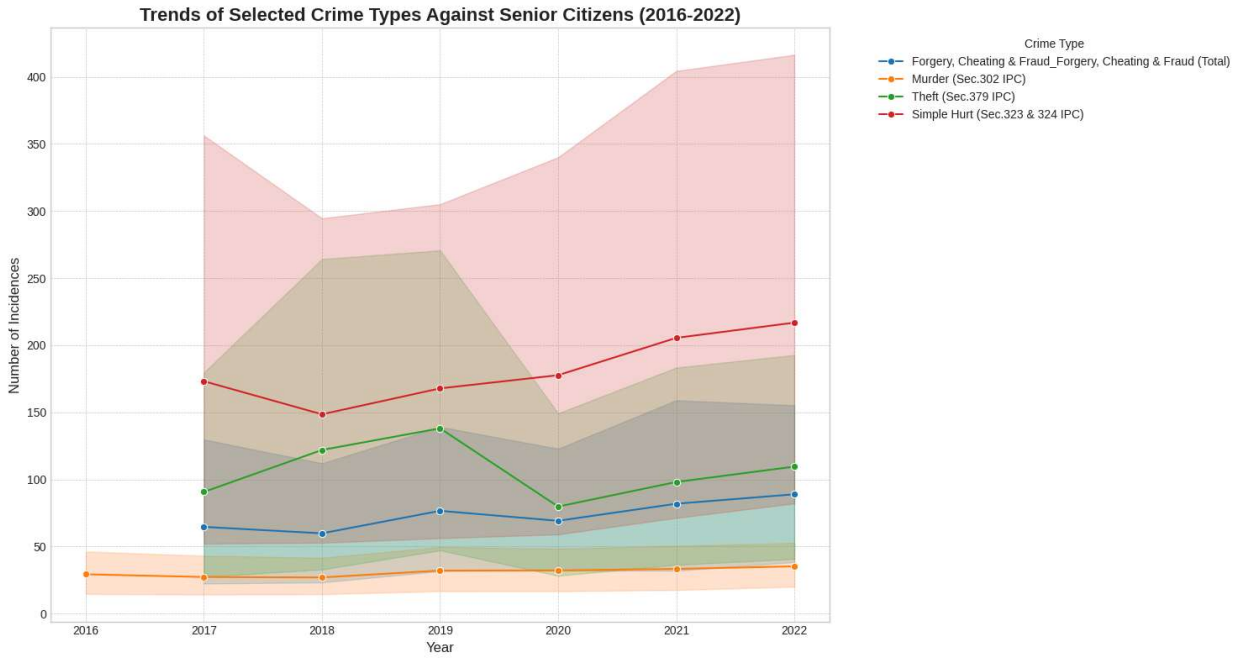
```
Correlation Matrix:

                                  number_of_incidences  \
number_of_incidences                          1.000000
number_of_senior_citizen_victims              0.999807
crime_rate_per_lakh_population                0.679716

                                  number_of_senior_citizen_victims  \
number_of_incidences                                      0.999807
number_of_senior_citizen_victims                          1.000000
crime_rate_per_lakh_population                            0.678741

                                  crime_rate_per_lakh_population
number_of_incidences                                    0.679716
number_of_senior_citizen_victims                        0.678741
crime_rate_per_lakh_population                          1.000000
```



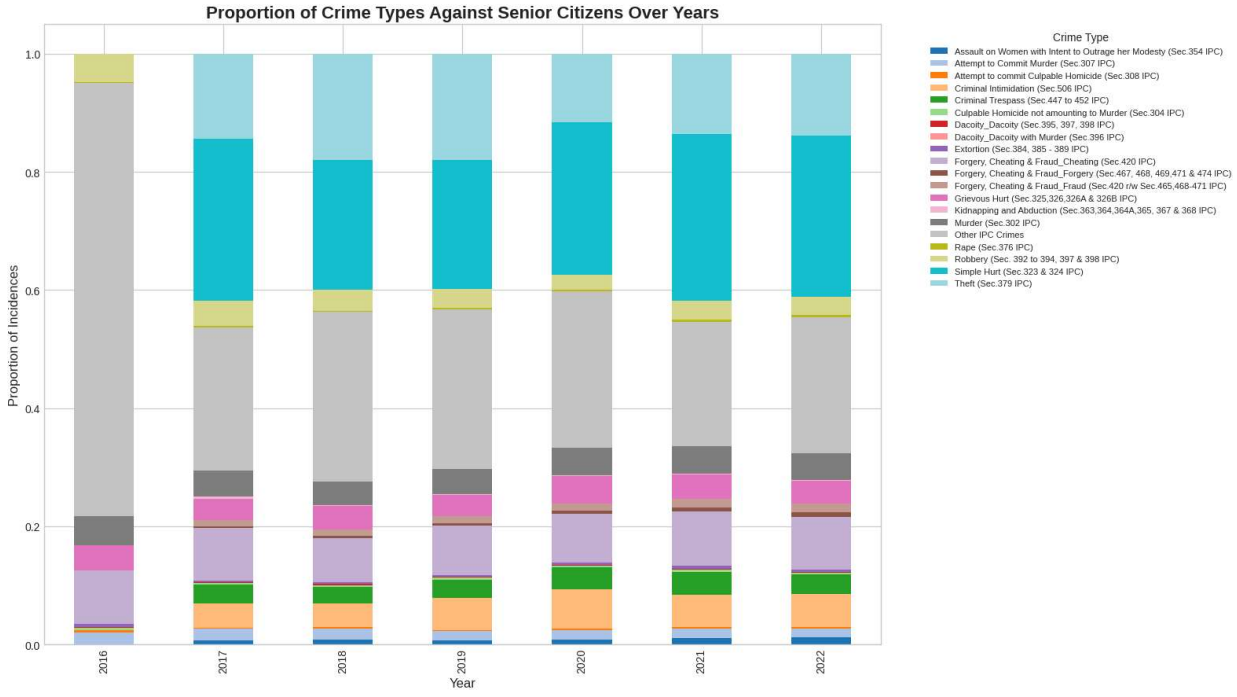

```
# Scatter plot: Number of Incidences vs. Crime Rate Per Lakh Population
plt.figure(figsize=(10, 6))
sns.scatterplot(data=df, x='number_of_incidences', y='crime_rate_per_lakh_p
plt.title('Number of Incidences vs. Crime Rate Per Lakh Population')
plt.xlabel('Number of Incidences')
plt.ylabel('Crime Rate Per Lakh Population')
plt.show()
```




```
# 1. Trend of specific crime types over years
selected_crime_types = [
    "Simple Hurt (Sec.323 & 324 IPC)",
    "Theft (Sec.379 IPC)",
    "Forgery, Cheating & Fraud_Forgery, Cheating & Fraud (Total)",
    "Murder (Sec.302 IPC)",
    "Robbery (Sec.392-394 IPC)"
]

df_selected_crimes = df[df["crime_head"].isin(selected_crime_types)]

plt.figure(figsize=(15, 8))
sns.lineplot(data=df_selected_crimes, x="year", y="number_of_incidences", h
plt.title("Trends of Selected Crime Types Against Senior Citizens (2016-202
plt.xlabel("Year", fontsize=12)
plt.ylabel("Number of Incidences", fontsize=12)
plt.legend(title="Crime Type", bbox_to_anchor=(1.05, 1), loc="upper left")
plt.grid(True, which="both", linestyle="--", linewidth=0.5)
plt.tight_layout()
plt.show()
```

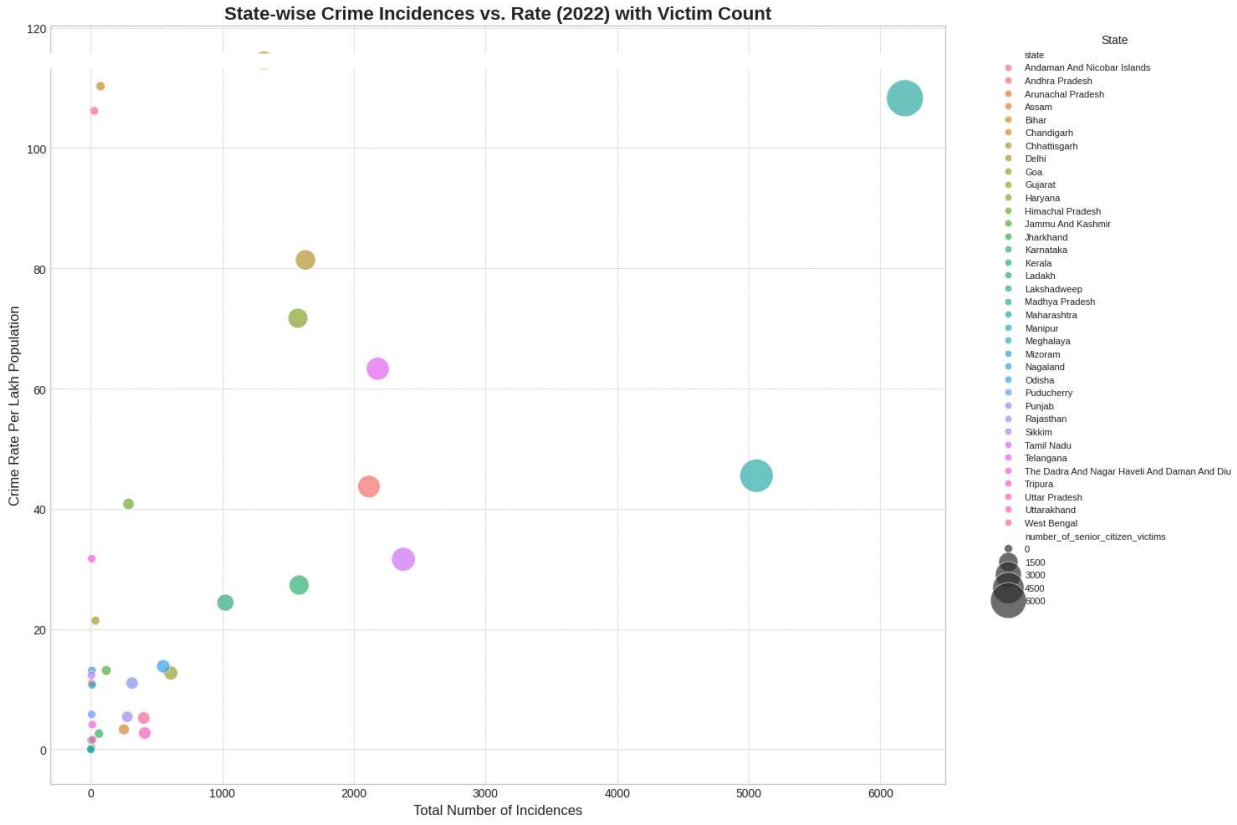



```
# 4. Crime Type Distribution by Year (Stacked Bar Chart)
crime_type_yearly = df_crimes_only.groupby(["year", "crime_head"])["number_
crime_type_yearly_percentage = crime_type_yearly.apply(lambda x: x / x.sum(

plt.figure(figsize=(16, 9))
crime_type_yearly_percentage.plot(kind="bar", stacked=True, colormap="tab20
plt.title("Proportion of Crime Types Against Senior Citizens Over Years", f
plt.xlabel("Year", fontsize=12)
plt.ylabel("Proportion of Incidences", fontsize=12)
plt.legend(title="Crime Type", bbox_to_anchor=(1.05, 1), loc="upper left",
plt.tight_layout()
plt.show()
```




```
plt.figure(figsize=(15, 10))
sns.scatterplot(
```

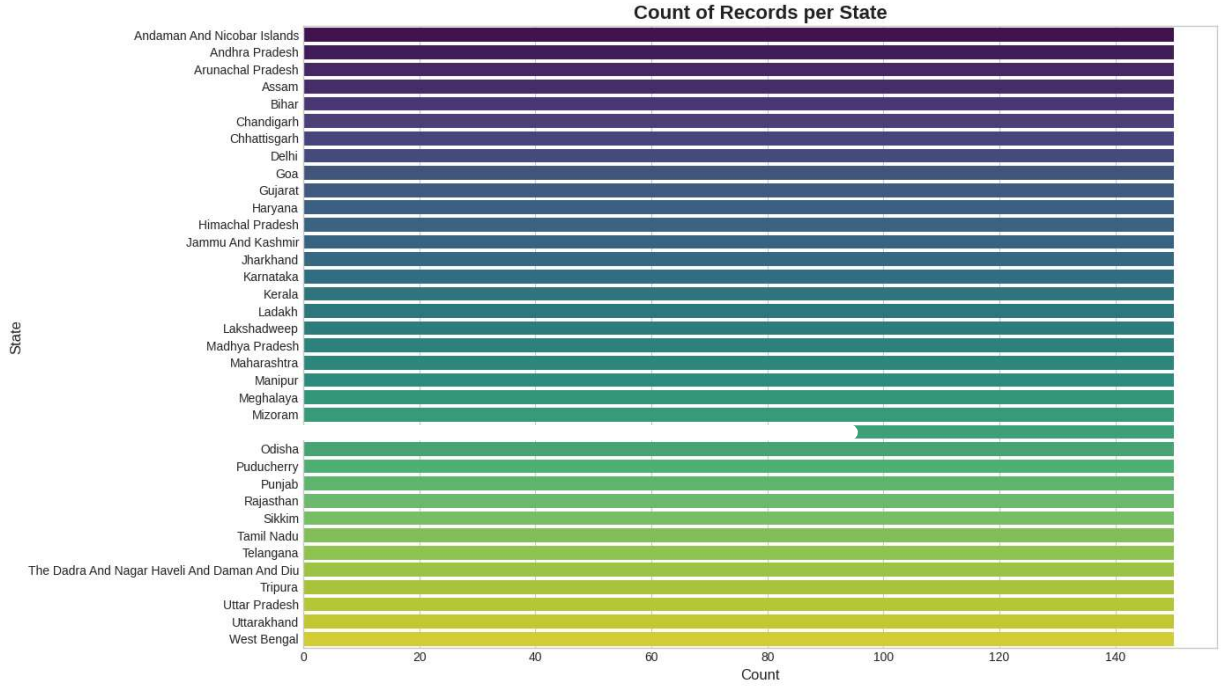
```
    data=df_2022_total,
    x="number_of_incidences",
    y="crime_rate_per_lakh_population",
    size="number_of_senior_citizen_victims",
    hue="state",
    sizes=(50, 1000),
    alpha=0.7
)
plt.title("State-wise Crime Incidences vs. Rate (2022) with Victim Count",
plt.xlabel("Total Number of Incidences", fontsize=12)
plt.ylabel("Crime Rate Per Lakh Population", fontsize=12)
plt.legend(title="State", bbox_to_anchor=(1.05, 1), loc="upper left", fonts
plt.grid(True, which="both", linestyle="--", linewidth=0.5)
plt.tight_layout()
plt.show()
```



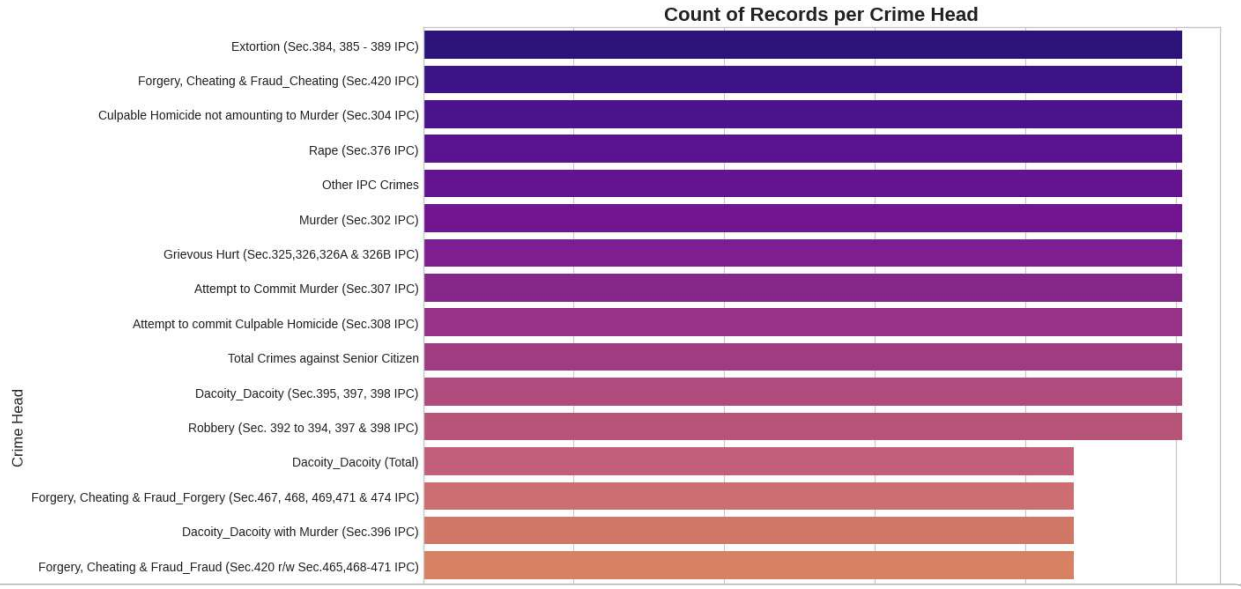

```
# Count plots for categorical columns (e.g., 'state' and 'crime_head' if we
# Since 'state' and 'crime_head' are already used in other plots, let's foc

plt.figure(figsize=(14, 8))
sns.countplot(data=df, y='state', order=df['state'].value_counts().index, p
plt.title('Count of Records per State', fontsize=16, weight='bold')
plt.xlabel('Count', fontsize=12)
plt.ylabel('State', fontsize=12)
plt.tight_layout()
plt.show()

plt.figure(figsize=(14, 10))
sns.countplot(data=df, y='crime_head', order=df['crime_head'].value_counts(
plt.title('Count of Records per Crime Head', fontsize=16, weight='bold')
plt.xlabel('Count', fontsize=12)
```

```
plt.ylabel('Crime Head', fontsize=12)
plt.tight_layout()
plt.show()
```

```
/tmp/ipython-input-4045360908.py:5: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y`

  sns.countplot(data=df, y='state', order=df['state'].value_counts().index, palette='viridis')
```

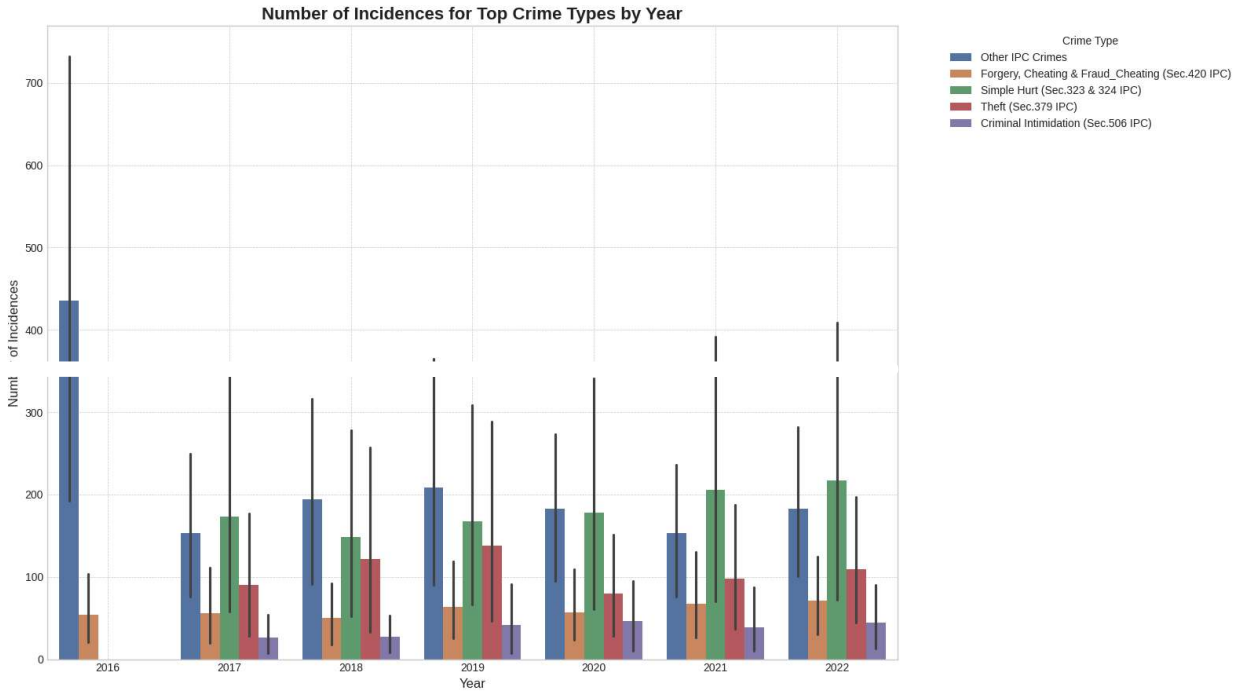



```
/tmp/ipython-input-4045360908.py:13: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y`

  sns.countplot(data=df, y='crime_head', order=df['crime_head'].value_counts().index, palette='plasma'
```

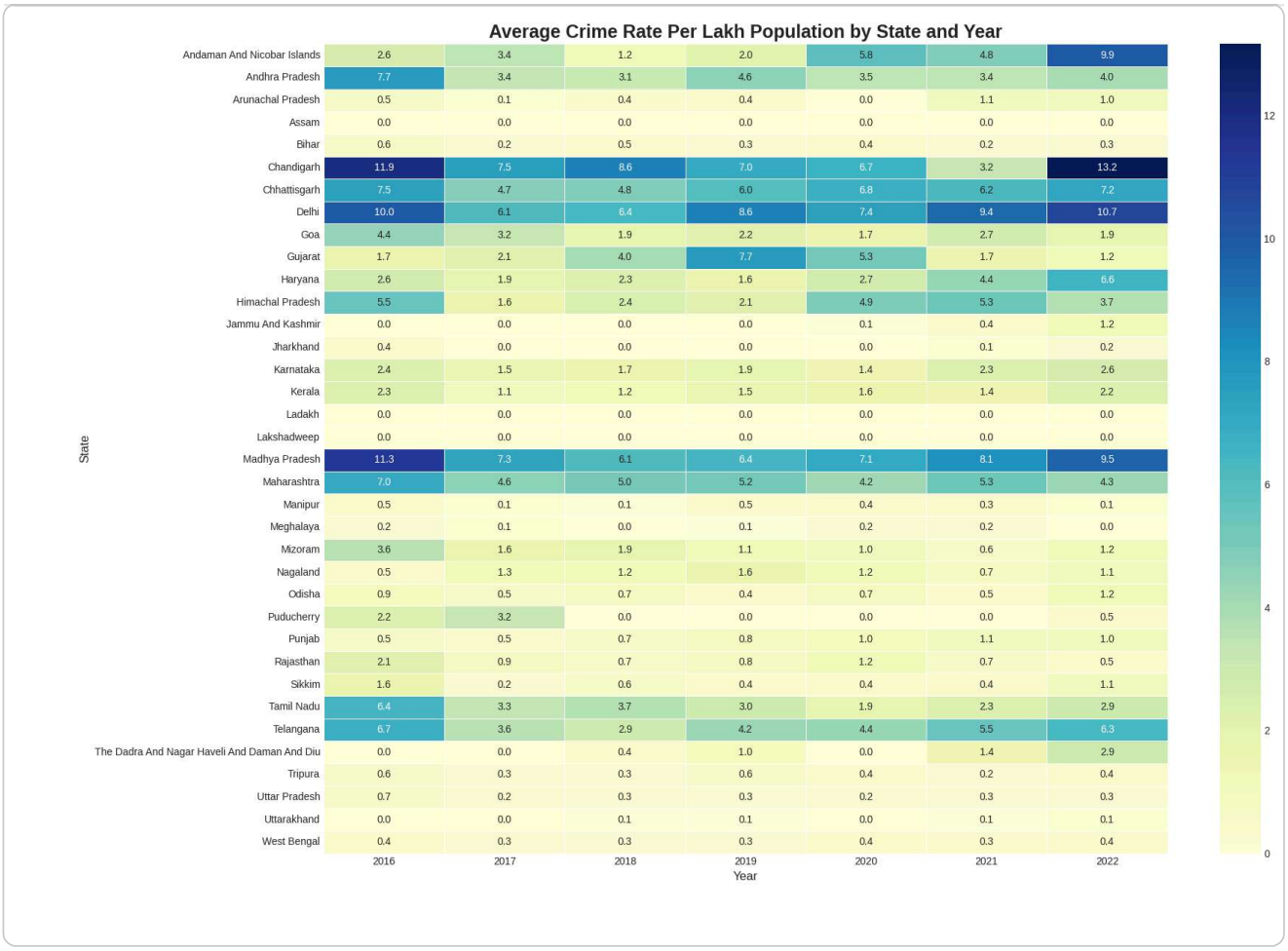



```
# Let's use the top 5 crime types from previous analysis
top_5_crimes = crime_type_counts.head(5).index.tolist()
df_top_5_crimes = df[df['crime_head'].isin(top_5_crimes)]

plt.figure(figsize=(16, 9))
sns.barplot(data=df_top_5_crimes, x='year', y='number_of_incidences', hue='
plt.title('Number of Incidences for Top Crime Types by Year', fontsize=16,
plt.xlabel('Year', fontsize=12)
plt.ylabel('Number of Incidences', fontsize=12)
plt.legend(title='Crime Type', bbox_to_anchor=(1.05, 1), loc='upper left')
```

```
plt.grid(True, which='both', linestyle='--', linewidth=0.5)
plt.tight_layout()
plt.show()
```




```
#Heatmap of Crime Rates by State and Year
state_year_crime_rate = df.pivot_table(values="crime_rate_per_lakh_populati
plt.figure(figsize=(18, 12))
sns.heatmap(state_year_crime_rate, annot=True, cmap="YlGnBu", fmt=".1f", li
plt.title("Average Crime Rate Per Lakh Population by State and Year", fonts
plt.xlabel("Year", fontsize=12)
plt.ylabel("State", fontsize=12)
plt.tight_layout()
plt.show()
```



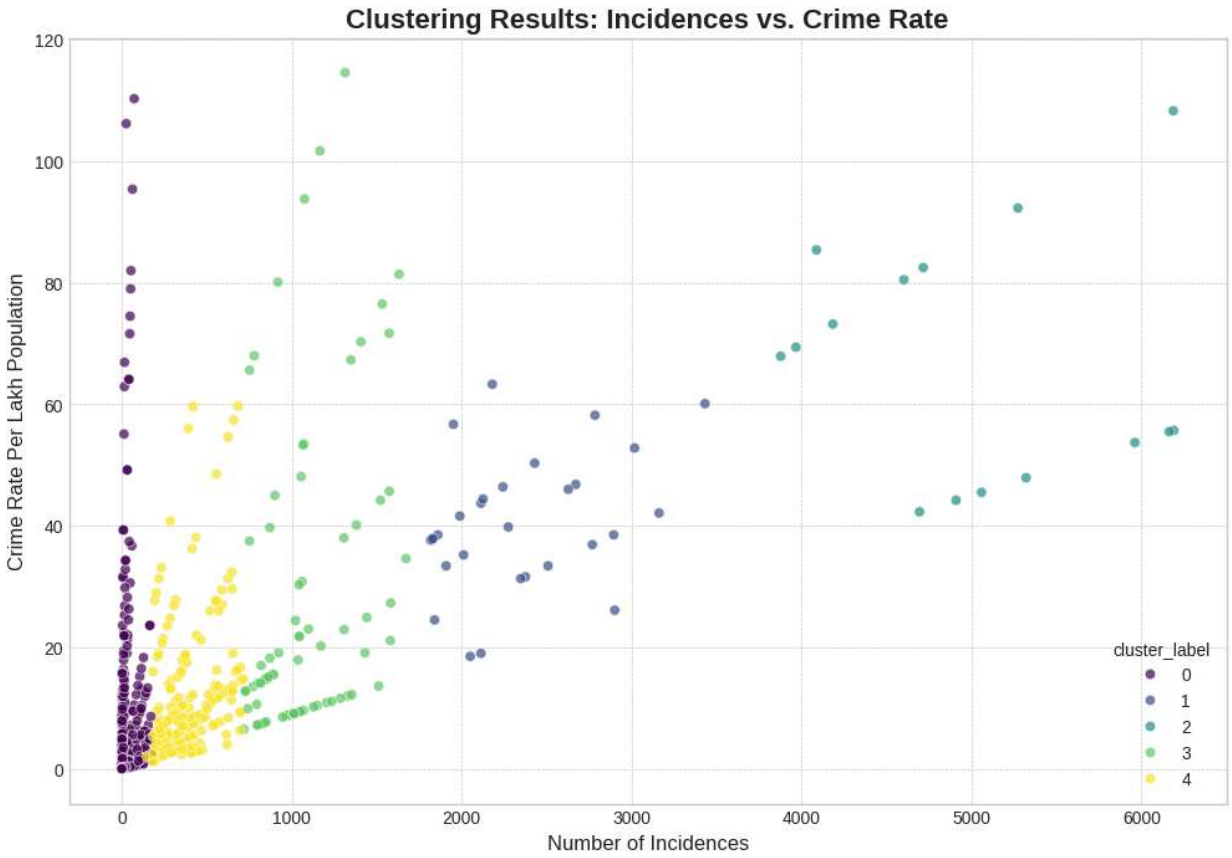
Task

Perform classification and clustering on the data, and present the hidden results.

```
df_clustering = df[['number_of_incidences', 'number_of_senior_citizen_victi
df_classification = df[['number_of_incidences', 'number_of_senior_citizen_v
```

```
from sklearn.cluster import KMeans

kmeans = KMeans(n_clusters=5, random_state=42, n_init=10)
kmeans.fit(df_clustering)
df['cluster_label'] = kmeans.labels_
```

```
plt.figure(figsize=(12, 8))
sns.scatterplot(data=df, x='number_of_incidences', y='crime_rate_per_lakh_p
plt.title('Clustering Results: Incidences vs. Crime Rate', fontsize=16, wei
plt.xlabel('Number of Incidences', fontsize=12)
plt.ylabel('Crime Rate Per Lakh Population', fontsize=12)
plt.grid(True, which='both', linestyle='--', linewidth=0.5)
plt.show()
```




```
from sklearn.model_selection import train_test_split

X = df_classification.drop('crime_head', axis=1)
y = df_classification['crime_head']
```

Reasoning: Split the data into training and testing sets using the defined features and target, and specified parameters.

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, ra
```

```
from sklearn.ensemble import RandomForestClassifier

model = RandomForestClassifier(random_state=42)
model.fit(X_train, y_train)
```

```
RandomForestClassifier
RandomForestClassifier(random_state=42)
```

```
from sklearn.metrics import accuracy_score, classification_report

y_pred = model.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.2f}")

print("\nClassification Report:")
print(classification_report(y_test, y_pred))
```

```
Accuracy: 0.12

Classification Report:
                                                                precision    recall  f1-score   su

  Assault on Women with Intent to Outrage her Modesty (Sec.354 IPC)       0.27      0.07      0.11
                             Attempt to Commit Murder (Sec.307 IPC)       0.19      0.12      0.15
                   Attempt to commit Culpable Homicide (Sec.308 IPC)       0.00      0.00      0.00
                                Criminal Intimidation (Sec.506 IPC)       0.06      0.02      0.03
                             Criminal Trespass (Sec.447 to 452 IPC)       0.12      0.05      0.07
            Culpable Homicide not amounting to Murder (Sec.304 IPC)       0.09      0.02      0.03
                           Dacoity_Dacoity (Sec.395, 397, 398 IPC)       0.00      0.00      0.00
                                       Dacoity_Dacoity (Total)       0.06      0.05      0.05
                          Dacoity_Dacoity with Murder (Sec.396 IPC)       0.07      0.91      0.13
                                 Extortion (Sec.384, 385 - 389 IPC)       0.00      0.00      0.00
                   Forgery, Cheating & Fraud_Cheating (Sec.420 IPC)       0.20      0.14      0.16
Forgery, Cheating & Fraud_Forgery (Sec.467, 468, 469,471 & 474 IPC)       0.00      0.00      0.00
          Forgery, Cheating & Fraud_Forgery, Cheating & Fraud (Total)       0.11      0.05      0.07
  Forgery, Cheating & Fraud_Fraud (Sec.420 r/w Sec.465,468-471 IPC)       0.11      0.05      0.06
                        Grievous Hurt (Sec.325,326,326A & 326B IPC)       0.12      0.08      0.10
     Kidnapping and Abduction (Sec.363,364,364A,365, 367 & 368 IPC)       0.30      0.14      0.19
                                                 Murder (Sec.302 IPC)       0.28      0.24      0.26
                                                     Other IPC Crimes       0.15      0.08      0.10
                                                  Rape (Sec.376 IPC)       0.11      0.16      0.13
                           Robbery (Sec. 392 to 394, 397 & 398 IPC)       0.15      0.06      0.09
                                    Simple Hurt (Sec.323 & 324 IPC)       0.09      0.07      0.08
                                                  Theft (Sec.379 IPC)       0.11      0.05      0.07
                              Total Crimes against Senior Citizen       0.41      0.35      0.38

                                                           accuracy                           0.12
                                                          macro avg       0.13      0.12      0.10
                                                       weighted avg       0.13      0.12      0.10

/usr/local/lib/python3.12/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.12/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.12/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```

Analyze clusters by state

Subtask:

Determine the distribution of states within each cluster.

Reasoning: Group the data by cluster and state and count the occurrences to understand the state distribution within each cluster.

```
cluster_state_distribution = df.groupby(['cluster_label', 'state']).size().
```

Visualize top states per cluster

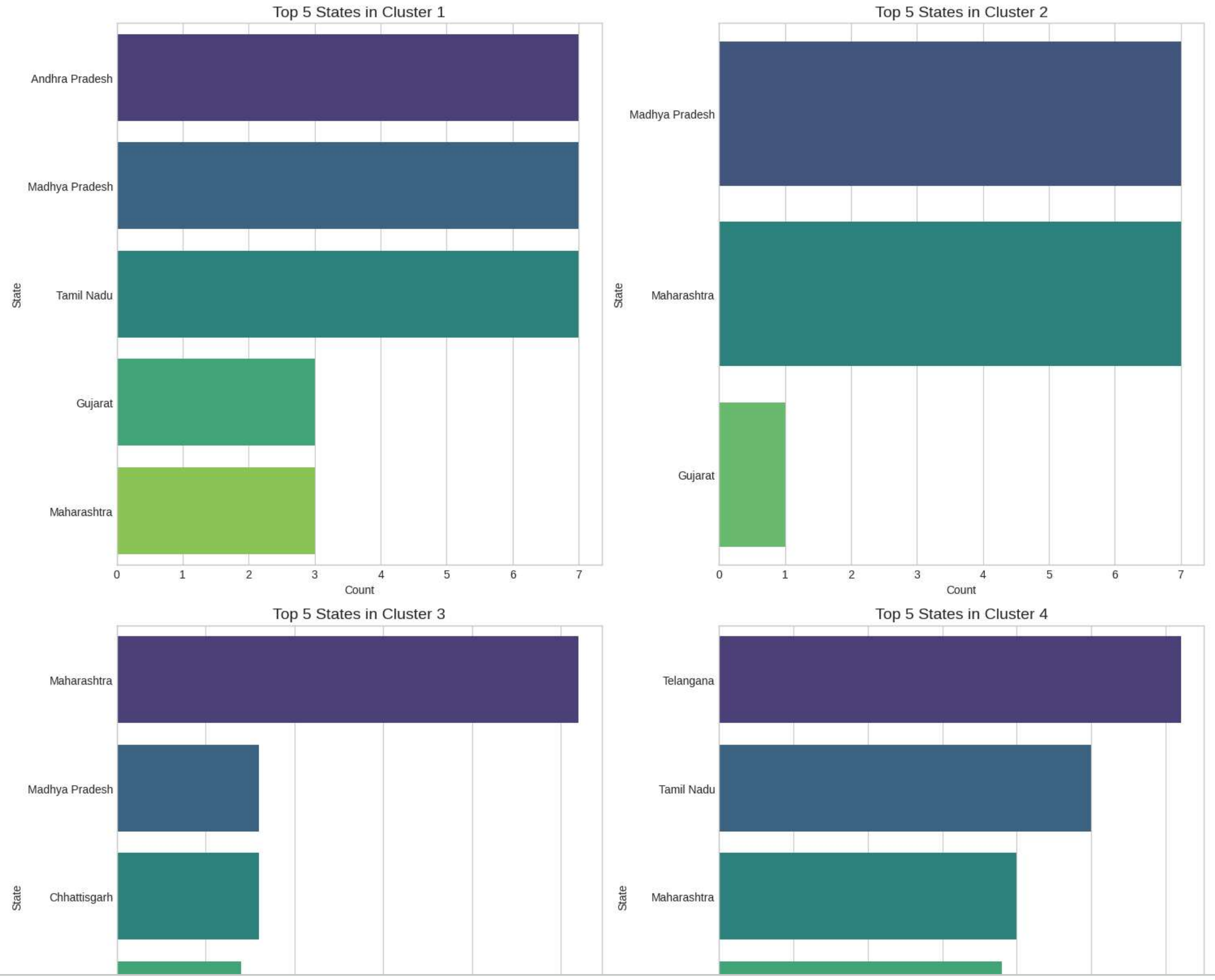
Subtask:

Create visualizations to show the top states within specific clusters.

```
plt.figure(figsize=(15, 10))
fig, axes = plt.subplots(2, 2, figsize=(15, 15))
axes = axes.flatten()

for i, cluster in enumerate([1, 2, 3, 4]):
    cluster_data = cluster_state_distribution[cluster_state_distribution['c
```

```
    top_states = cluster_data.sort_values(by='state_count', ascending=False

    sns.barplot(x='state_count', y='state', data=top_states, palette='virid
    axes[i].set_title(f'Top 5 States in Cluster {cluster}', fontsize=14)
    axes[i].set_xlabel('Count', fontsize=10)
    axes[i].set_ylabel('State', fontsize=10)

plt.tight_layout()
plt.show()
```

```
/tmp/ipython-input-466582195.py:9: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y`

  sns.barplot(x='state_count', y='state', data=top_states, palette='viridis', ax=axes[i])
/tmp/ipython-input-466582195.py:9: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y`

  sns.barplot(x='state_count', y='state', data=top_states, palette='viridis', ax=axes[i])
/tmp/ipython-input-466582195.py:9: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y`

  sns.barplot(x='state_count', y='state', data=top_states, palette='viridis', ax=axes[i])
/tmp/ipython-input-466582195.py:9: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y`

  sns.barplot(x='state_count', y='state', data=top_states, palette='viridis', ax=axes[i])
<Figure size 1500x1000 with 0 Axes>
```

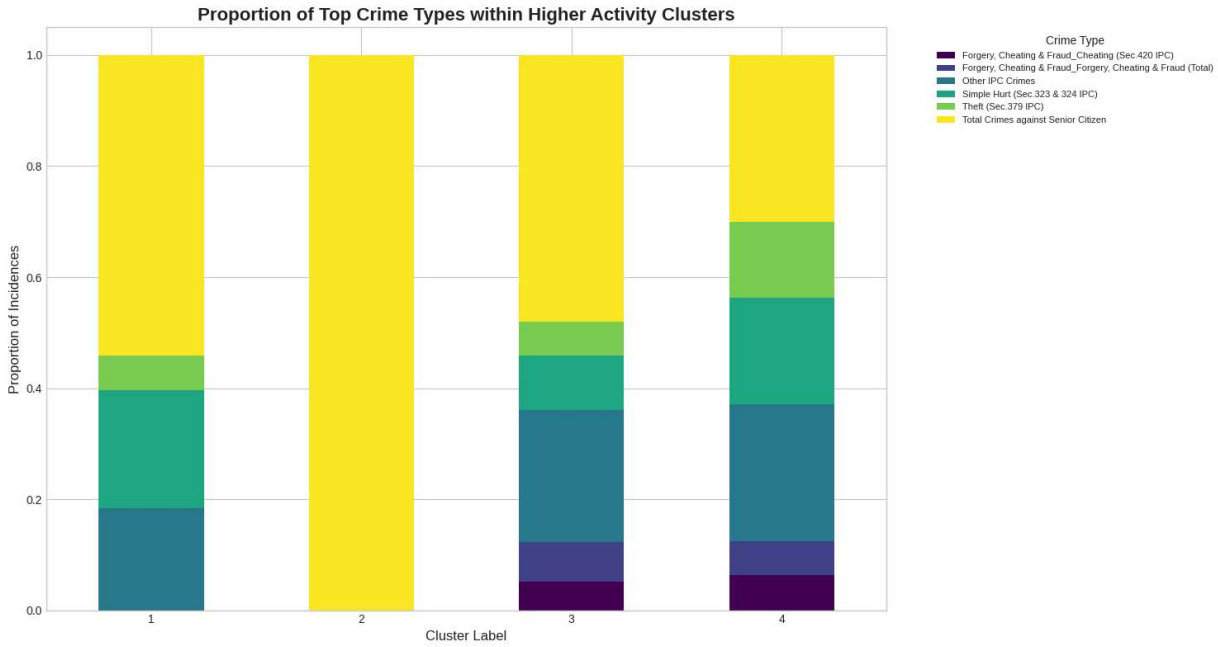


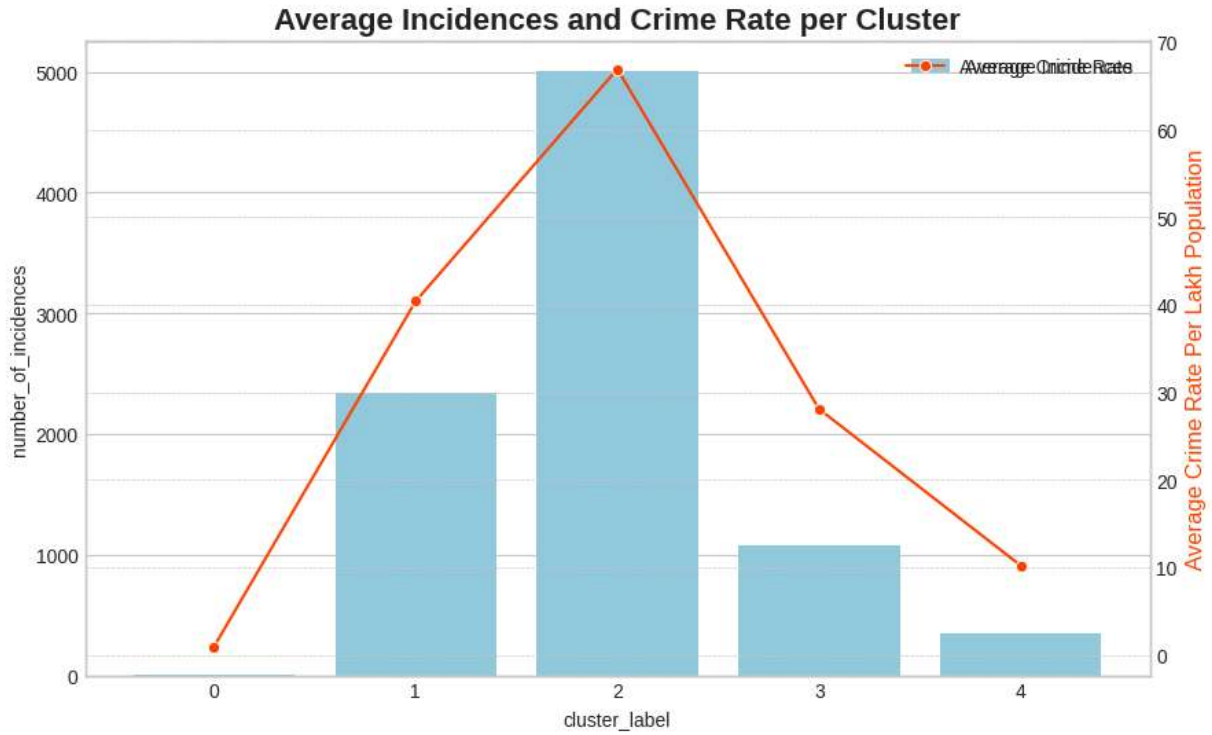

```
# Let's visualize the distribution of crime types within each cluster using
# We'll focus on the clusters with higher activity (1, 2, 3, 4) and the top

# First, let's find the top crime types within each of the higher activity 
```

```
top_crimes_per_cluster = {}
for cluster in [1, 2, 3, 4]:
    df_cluster = df[df['cluster_label'] == cluster]
    crime_counts = df_cluster['crime_head'].value_counts().head(5)
    top_crimes_per_cluster[cluster] = crime_counts.index.tolist()

# Now, let's prepare the data for the stacked bar chart
crime_cluster_distribution = df[df['cluster_label'].isin([1, 2, 3, 4])]
crime_cluster_distribution = crime_cluster_distribution[
    crime_cluster_distribution['crime_head'].isin([item for sublist in top_
]

crime_cluster_pivot = crime_cluster_distribution.pivot_table(
    index='cluster_label',
    columns='crime_head',
    values='number_of_incidences',
    aggfunc='sum',
    fill_value=0
)

# Normalize the data to show proportions within each cluster
crime_cluster_pivot_normalized = crime_cluster_pivot.apply(lambda x: x / x.

plt.figure(figsize=(15, 8))
crime_cluster_pivot_normalized.plot(kind='bar', stacked=True, colormap='vir
plt.title('Proportion of Top Crime Types within Higher Activity Clusters',
plt.xlabel('Cluster Label', fontsize=12)
plt.ylabel('Proportion of Incidences', fontsize=12)
plt.xticks(rotation=0)
plt.legend(title='Crime Type', bbox_to_anchor=(1.05, 1), loc='upper left',
plt.tight_layout()
plt.show()

# 2. Visualize the average crime rate and number of incidences per cluster
cluster_summary = df.groupby('cluster_label')[['number_of_incidences', 'cri

plt.figure(figsize=(10, 6))
sns.barplot(data=cluster_summary, x='cluster_label', y='number_of_incidence
ax2 = plt.twinx()
sns.lineplot(data=cluster_summary, x='cluster_label', y='crime_rate_per_lak

plt.title('Average Incidences and Crime Rate per Cluster', fontsize=16, wei
plt.xlabel('Cluster Label', fontsize=12)
plt.ylabel('Average Number of Incidences', fontsize=12, color='skyblue')
ax2.set_ylabel('Average Crime Rate Per Lakh Population', fontsize=12, color
plt.grid(True, which='both', linestyle='--', linewidth=0.5)
plt.show()

# 3. Visualize the distribution of the most frequent crime type across the

# Find the most frequent crime type in Cluster 4
most_frequent_crime_cluster_4 = df[df['cluster_label'] == 4]['crime_head'].

# Get the top 5 states in Cluster 4
top_states_cluster_4 = cluster_state_distribution[cluster_state_distributio
```

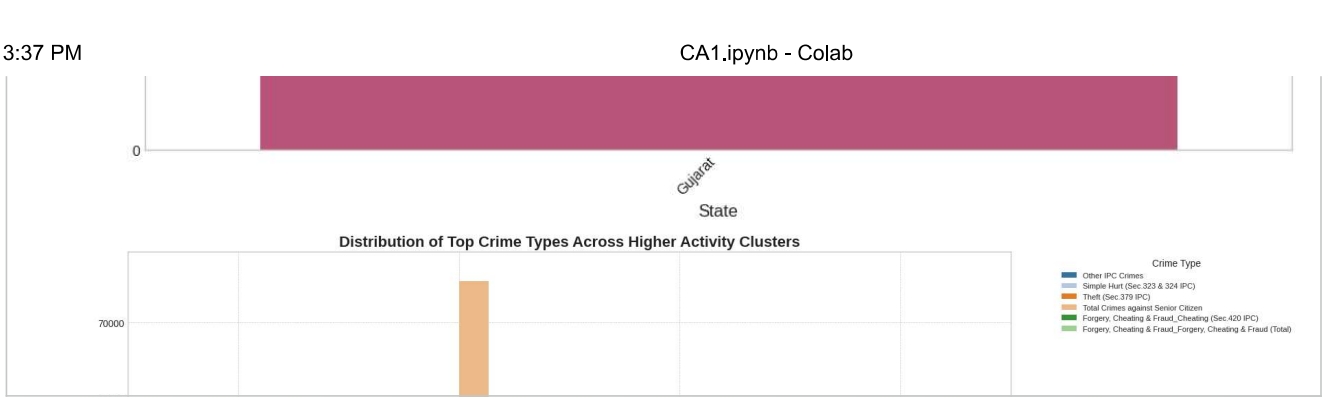
```
# Filter data for the most frequent crime type in Cluster 4 and the top sta
df_most_frequent_crime_cluster_4 = df[
    (df['crime_head'] == most_frequent_crime_cluster_4) &
    (df['state'].isin(top_states_cluster_4)) &
    (df['cluster_label'] == 4) # Ensure we are only considering data points
]

# Group by state and sum the number of incidences
crime_in_top_states_cluster_4 = df_most_frequent_crime_cluster_4.groupby('s

plt.figure(figsize=(12, 7))
sns.barplot(x=crime_in_top_states_cluster_4.index, y=crime_in_top_states_cl
plt.title(f'Total Incidences of "{most_frequent_crime_cluster_4}" in Top St
plt.xlabel('State', fontsize=12)
plt.ylabel('Total Number of Incidences', fontsize=12)
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()

# Additional visualizations as requested by the user

# Visualize crime types distribution within clusters using a different appr
# Using a grouped bar chart to show the top crime types within each cluster
top_crimes_long_format = crime_cluster_distribution.groupby(['cluster_label

# Filter to include only the top crimes identified earlier
top_crimes_long_format = top_crimes_long_format[
    top_crimes_long_format['crime_head'].isin([item for sublist in top_crim
]

plt.figure(figsize=(18, 10))
sns.barplot(data=top_crimes_long_format, x='cluster_label', y='number_of_in
plt.title('Distribution of Top Crime Types Across Higher Activity Clusters'
plt.xlabel('Cluster Label', fontsize=12)
plt.ylabel('Total Number of Incidences', fontsize=12)
plt.legend(title='Crime Type', bbox_to_anchor=(1.05, 1), loc='upper left',
plt.grid(True, which='both', linestyle='--', linewidth=0.5)
plt.tight_layout()
plt.show()


# Visualize the relationship between numerical features and cluster labels
# Using 'viridis' color palette as requested (user mentioned 'iris', but 'v
plt.figure(figsize=(12, 8))
sns.stripplot(data=df, x='cluster_label', y='number_of_incidences', hue='cl
plt.title('Cluster Distribution by Number of Incidences with Jitter', fonts
plt.xlabel('Cluster Label', fontsize=12)
plt.ylabel('Number of Incidences', fontsize=12)
plt.grid(True, which='both', linestyle='--', linewidth=0.5)
plt.show()

plt.figure(figsize=(12, 8))
sns.stripplot(data=df, x='cluster_label', y='crime_rate_per_lakh_population
plt.title('Cluster Distribution by Crime Rate Per Lakh Population with Jitt
```

```
plt.xlabel('Cluster Label', fontsize=12)
plt.ylabel('Crime Rate Per Lakh Population', fontsize=12)
plt.grid(True, which='both', linestyle='--', linewidth=0.5)
plt.show()


# A chord chart is typically used for visualizing relationships between ent
# It's not directly applicable to visualize the distribution of crime types
# Instead, let's create a different unique visualization focusing on the to

# Get the top 10 states by total incidences
top_10_states_overall = df.groupby('state')['number_of_incidences'].sum().s

# Filter the data for top states and top 10 crime types (based on overall c
top_10_crimes_overall = df_crimes_only.groupby('crime_head')['number_of_inc

df_top_states_crimes = df[
    (df['state'].isin(top_10_states_overall)) &
    (df['crime_head'].isin(top_10_crimes_overall))
]

# Create a pivot table for heatmap visualization
state_crime_pivot = df_top_states_crimes.pivot_table(
    index='state',
    columns='crime_head',
    values='number_of_incidences',
    aggfunc='sum',
    fill_value=0
)

# Convert the pivot table to integers for annotation
state_crime_pivot_int = state_crime_pivot.astype(int)

# Print the pivot table and its data types for debugging
print("State Crime Pivot Table:")
print(state_crime_pivot)
print("\nData types of State Crime Pivot Table:")
print(state_crime_pivot.dtypes)
print("\nState Crime Pivot Table (Integer):")
print(state_crime_pivot_int)
print("\nData types of State Crime Pivot Table (Integer):")
print(state_crime_pivot_int.dtypes)


# Visualize using a heatmap with 'brewer' palette (e.g., 'Blues', 'Greens',
plt.figure(figsize=(16, 10))
sns.heatmap(state_crime_pivot, annot=state_crime_pivot_int, cmap='Blues', f
plt.title('Total Incidences of Top Crime Types in Top States (2016-2022)',
plt.xlabel('Crime Type', fontsize=12)
plt.ylabel('State', fontsize=12)
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```






```
/tmp/ipython-input-255537304.py:72: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x`

  sns.barplot(x=crime_in_top_states_cluster_4.index, y=crime_in_top_states_cluster_4.values, palette='
```


0
Gujarat
State

Distribution of Top Crime Types Across Higher Activity Clusters
70000
Crime Type
Other IPC Crimes
Simple Hurt (Sec.323 & 324 IPC)
Theft (Sec.379 IPC)
Total Crimes against Senior Citizen
Forgery, Cheating & Fraud_Cheating (Sec.420 IPC)
Forgery, Cheating & Fraud_Forgery, Cheating & Fraud (Total)