Report

Data Loading and preparation

```
#%pip install pandas
import pandas as pd
```

```
df=pd.read_csv("Districtwise_Crime_of_India_2001_to_2014 - Sheet1.csv")
df.drop(columns=['Unnamed: 0'], inplace=True)
df.head()
```

	STATE/UT	DISTRICT	YEAR	MURDER	ATTEMPT TO MURDER	CULPABLE HOMICIDE NOT AMOUNTING TO MURDER	RAPE	CUSTODIAL RAPE	OTHER RAPE	KIDNAPPING & ABDUCTION	...	ARSON	HURT/GREVIOUS HURT	DOWRY DEATHS	ASSAUL O WOME WIT INTEN T OUTRAG HE MODEST
0	ANDHRA PRADESH	ADILABAD	2001	101	60	17	50	0	50	46	...	30	1131	16	14
1	ANDHRA PRADESH	ANANTAPUR	2001	151	125	1	23	0	23	53	...	69	1543	7	11
2	ANDHRA PRADESH	CHITTOOR	2001	101	57	2	27	0	27	59	...	38	2088	14	11
3	ANDHRA PRADESH	CUDDAPAH	2001	80	53	1	20	0	20	25	...	23	795	17	12
4	ANDHRA PRADESH	EAST GODAVARI	2001	82	67	1	23	0	23	49	...	41	1244	12	10

5 rows × 33 columns

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10678 entries, 0 to 10677
Data columns (total 33 columns):
 #   Column                                              Non-Null Count  Dtype
---  ------                                              --------------  -----
 0   STATE/UT                                            10678 non-null  object
 1   DISTRICT                                            10678 non-null  object
 2   YEAR                                                10678 non-null  int64
 3   MURDER                                              10678 non-null  int64
 4   ATTEMPT TO MURDER                                   10678 non-null  int64
 5   CULPABLE HOMICIDE NOT AMOUNTING TO MURDER           10678 non-null  int64
 6   RAPE                                                10678 non-null  int64
 7   CUSTODIAL RAPE                                      10678 non-null  int64
 8   OTHER RAPE                                          10678 non-null  int64
 9   KIDNAPPING & ABDUCTION                              10678 non-null  int64
 10  KIDNAPPING AND ABDUCTION OF WOMEN AND GIRLS         10678 non-null  int64
 11  KIDNAPPING AND ABDUCTION OF OTHERS                  10678 non-null  int64
 12  DACOITY                                             10678 non-null  int64
 13  PREPARATION AND ASSEMBLY FOR DACOITY                10678 non-null  int64
 14  ROBBERY                                             10678 non-null  int64
 15  BURGLARY                                            10678 non-null  int64
 16  THEFT                                               10678 non-null  int64
 17  AUTO THEFT                                          10678 non-null  int64
 18  OTHER THEFT                                         10678 non-null  int64
 19  RIOTS                                               10678 non-null  int64
 20  CRIMINAL BREACH OF TRUST                            10678 non-null  int64
 21  CHEATING                                            10678 non-null  int64
 22  COUNTERFIETING                                      10678 non-null  int64
 23  ARSON                                               10678 non-null  int64
 24  HURT/GREVIOUS HURT                                  10678 non-null  int64
 25  DOWRY DEATHS                                        10678 non-null  int64
 26  ASSAULT ON WOMEN WITH INTENT TO OUTRAGE HER MODESTY  10678 non-null  int64
 27  INSULT TO MODESTY OF WOMEN                          10678 non-null  int64
 28  CRUELTY BY HUSBAND OR HIS RELATIVES                 10678 non-null  int64
 29  IMPORTATION OF GIRLS FROM FOREIGN COUNTRIES          10678 non-null  int64
 30  CAUSING DEATH BY NEGLIGENCE                          10678 non-null  int64
 31  OTHER IPC CRIMES                                     10678 non-null  int64
 32  TOTAL IPC CRIMES                                     10678 non-null  int64
dtypes: int64(31), object(2)
memory usage: 2.7+ MB
```

```
df['STATE/UT'].unique()
```

```
array(['ANDHRA PRADESH', 'ARUNACHAL PRADESH', 'ASSAM', 'BIHAR',
       'CHHATTISGARH', 'GOA', 'GUJARAT', 'HARYANA', 'HIMACHAL PRADESH',
       'JAMMU & KASHMIR', 'JHARKHAND', 'KARNATAKA', 'KERALA',
       'MADHYA PRADESH', 'MAHARASHTRA', 'MANIPUR', 'MEGHALAYA', 'MIZORAM',
       'NAGALAND', 'ODISHA', 'PUNJAB', 'RAJASTHAN', 'SIKKIM',
       'TAMIL NADU', 'TRIPURA', 'UTTAR PRADESH', 'UTTARAKHAND',
       'WEST BENGAL', 'A & N ISLANDS', 'CHANDIGARH', 'D & N HAVELI',
       'DAMAN & DIU', 'DELHI UT', 'LAKSHADWEEP', 'PUDUCHERRY',
       'TELANGANA', 'A&N ISLANDS', 'D&N HAVELI'], dtype=object)
```

correcting state name error.

```
df['STATE/UT'] = df['STATE/UT'].replace('D & N HAVELI', 'D&N HAVELI')
```

```
df['STATE/UT'].unique()
```

```
array(['ANDHRA PRADESH', 'ARUNACHAL PRADESH', 'ASSAM', 'BIHAR',
       'CHHATTISGARH', 'GOA', 'GUJARAT', 'HARYANA', 'HIMACHAL PRADESH',
       'JAMMU & KASHMIR', 'JHARKHAND', 'KARNATAKA', 'KERALA',
       'MADHYA PRADESH', 'MAHARASHTRA', 'MANIPUR', 'MEGHALAYA', 'MIZORAM',
       'NAGALAND', 'ODISHA', 'PUNJAB', 'RAJASTHAN', 'SIKKIM',
       'TAMIL NADU', 'TRIPURA', 'UTTAR PRADESH', 'UTTARAKHAND',
       'WEST BENGAL', 'A & N ISLANDS', 'CHANDIGARH', 'D&N HAVELI',
       'DAMAN & DIU', 'DELHI UT', 'LAKSHADWEEP', 'PUDUCHERRY',
       'TELANGANA', 'A&N ISLANDS'], dtype=object)
```

```
df.isnull().sum()
```

	0
STATE/UT	0
DISTRICT	0
YEAR	0
MURDER	0
ATTEMPT TO MURDER	0
CULPABLE HOMICIDE NOT AMOUNTING TO MURDER	0
RAPE	0
CUSTODIAL RAPE	0
OTHER RAPE	0
KIDNAPPING & ABDUCTION	0
KIDNAPPING AND ABDUCTION OF WOMEN AND GIRLS	0
KIDNAPPING AND ABDUCTION OF OTHERS	0
DACOITY	0
PREPARATION AND ASSEMBLY FOR DACOITY	0
ROBBERY	0
BURGLARY	0
THEFT	0
AUTO THEFT	0
OTHER THEFT	0
RIOTS	0
CRIMINAL BREACH OF TRUST	0
CHEATING	0
COUNTERFIETING	0
ARSON	0
HURT/GREVIOUS HURT	0
DOWRY DEATHS	0
ASSAULT ON WOMEN WITH INTENT TO OUTRAGE HER MODESTY	0
INSULT TO MODESTY OF WOMEN	0
CRUELTY BY HUSBAND OR HIS RELATIVES	0
IMPORTATION OF GIRLS FROM FOREIGN COUNTRIES	0
CAUSING DEATH BY NEGLIGENCE	0
OTHER IPC CRIMES	0
TOTAL IPC CRIMES	0

```
dtype: int64
```

we dont have any null values.

```
df['YEAR'].unique()
```

```
array([2001, 2002, 2003, 2004, 2005, 2006, 2007, 2008, 2009, 2010, 2011,
       2012, 2013, 2014])
```

seems like we have data from 2001 to 2014

```
df.describe()
```

	YEAR	MURDER	ATTEMPT TO MURDER	CULPABLE HOMICIDE NOT AMOUNTING TO MURDER	RAPE	CUSTODIAL RAPE	OTHER RAPE	KIDNAPPING & ABDUCTION	KIDNAPPING AND ABDUCTION OF WOMEN AND GIRLS	KIDN ABD OF
count	10678.000000	10678.000000	10678.000000	10678.000000	10678.000000	10678.000000	10678.000000	10678.000000	10678.000000	10678.
mean	2007.698539	88.008616	80.406818	9.616595	58.348380	0.042330	57.865518	83.861023	62.821128	23.
std	4.047144	323.658451	317.038205	58.251920	216.304175	1.898937	213.121800	351.295301	272.733640	104.
min	2001.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.
25%	2004.000000	18.000000	10.000000	0.000000	8.000000	0.000000	8.000000	8.000000	5.000000	1.
50%	2008.000000	37.000000	28.000000	2.000000	22.000000	0.000000	22.000000	25.000000	18.000000	5.
75%	2011.000000	66.000000	58.000000	6.000000	45.000000	0.000000	44.000000	58.000000	45.000000	14.
max	2014.000000	7601.000000	7964.000000	1616.000000	5076.000000	189.000000	4792.000000	11183.000000	9737.000000	3183.

8 rows × 31 columns

EDA Questions

Q) Determine the total number of crimes recorded across all districts and the average number of murders per district.

```
crimes = df['TOTAL IPC CRIMES'].sum()
print(f"Total crimes recorded across all districts: {crimes}")
```

```
Total crimes recorded across all districts: 58894630
```

```
average_murders = df['MURDER'].mean()
print(f'Average number of murders per district: {average_murders}')
```

```
Average number of murders per district: 88.00861584566398
```

Q) Examine how crime distributions vary across different states, and identify the top 5 districts with the highest total IPC crimes.

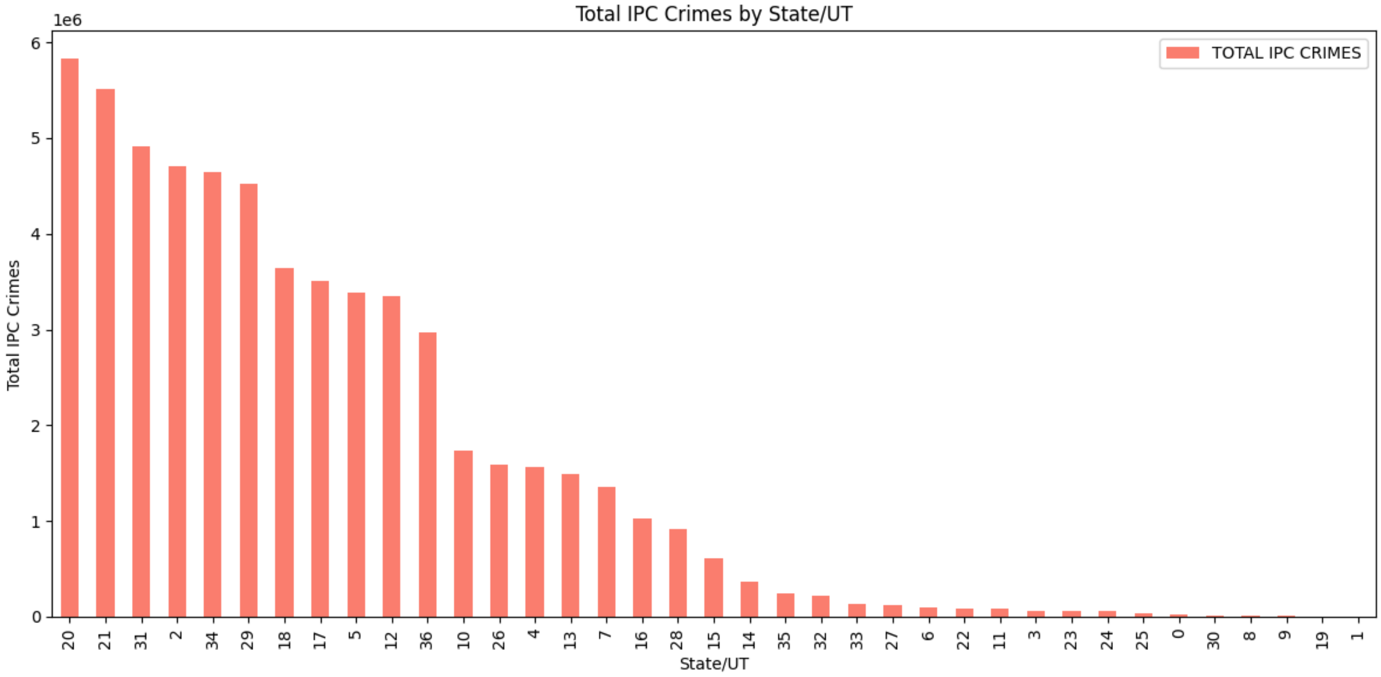
```
df['STATE/UT'].unique()
```

```
array(['ANDHRA PRADESH', 'ARUNACHAL PRADESH', 'ASSAM', 'BIHAR',
       'CHHATTISGARH', 'GOA', 'GUJARAT', 'HARYANA', 'HIMACHAL PRADESH',
       'JAMMU & KASHMIR', 'JHARKHAND', 'KARNATAKA', 'KERALA',
       'MADHYA PRADESH', 'MAHARASHTRA', 'MANIPUR', 'MEGHALAYA', 'MIZORAM',
       'NAGALAND', 'ODISHA', 'PUNJAB', 'RAJASTHAN', 'SIKKIM',
       'TAMIL NADU', 'TRIPURA', 'UTTAR PRADESH', 'UTTARAKHAND',
       'WEST BENGAL', 'A & N ISLANDS', 'CHANDIGARH', 'D&N HAVELI',
       'DAMAN & DIU', 'DELHI UT', 'LAKSHADWEEP', 'PUDUCHERRY',
       'TELANGANA', 'A&N ISLANDS'], dtype=object)
```

```
# Top 5 Districts with Highest Total IPC Crimes
district_crime_totals = df.groupby('DISTRICT')['TOTAL IPC CRIMES'].sum().reset_index()
district_crime_totals = district_crime_totals.sort_values(by='TOTAL IPC CRIMES', ascending=False)
district_crime_totals = district_crime_totals.reset_index(drop=True) # reassign to the variable
top_5_districts = district_crime_totals.head(6)
top_5_districts.drop(0,inplace=True)

print("\nTop 5 Districts with Highest Total IPC Crimes:\n", top_5_districts.to_markdown(index=False, numalign="left", stralign="left"))

# Crime Distribution by State
state_crime_totals = df.groupby('STATE/UT')['TOTAL IPC CRIMES'].sum().reset_index()
state_crime_totals = state_crime_totals.sort_values(by='TOTAL IPC CRIMES', ascending=False)

print("\nCrime Distribution by State:\n", state_crime_totals.to_markdown(index=False, numalign="left", stralign="left"))
```

```
Top 5 Districts with Highest Total IPC Crimes:
 | DISTRICT         | TOTAL IPC CRIMES   |
|:-----------------|:-------------------|
| DELHI UT TOTAL   | 633174             |
| BANGALORE COMMR. | 380665             |
| MUMBAI COMMR.    | 297871            |
| INDORE           | 250639            |
| AHMEDABAD COMMR. | 239263            |

Crime Distribution by State:
 | STATE/UT          | TOTAL IPC CRIMES   |
|:------------------|:-------------------|
| MADHYA PRADESH    | 5827292            |
| MAHARASHTRA       | 5515310            |
| TAMIL NADU        | 4913910            |
| ANDHRA PRADESH    | 4703200            |
| UTTAR PRADESH     | 4649988            |
| RAJASTHAN         | 4525116            |
| KERALA            | 3641164            |
| KARNATAKA         | 3510180            |
| BIHAR             | 3382686            |
| GUJARAT           | 3349190            |
| WEST BENGAL       | 2963774            |
| DELHI UT          | 1738024            |
| ODISHA            | 1588466            |
| ASSAM             | 1558574            |
| HARYANA           | 1494696            |
| CHHATTISGARH      | 1352194            |
| JHARKHAND         | 1031788            |
| PUNJAB            | 915920             |
| JAMMU & KASHMIR   | 616786             |
| HIMACHAL PRADESH  | 365716             |
| UTTARAKHAND       | 243812             |
| TELANGANA         | 213660             |
| TRIPURA           | 128886             |
| PUDUCHERRY        | 122912             |
| CHANDIGARH        | 96210              |
| MANIPUR           | 83782              |
| GOA               | 81658              |
| ARUNACHAL PRADESH | 66542              |
| MEGHALAYA         | 64374              |
| MIZORAM           | 59990              |
| NAGALAND          | 31012              |
| A & N ISLANDS     | 19428              |
| SIKKIM            | 17832              |
| D&N HAVELI        | 10484              |
| DAMAN & DIU       | 6854               |
| LAKSHADWEEP       | 1728               |
| A&N ISLANDS       | 1492               |
<ipython-input-35-3187ba42f254>:6: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a
```

```
#%pip install matplotlib
#%pip install seaborn
import matplotlib.pyplot as plt
import seaborn as sns

state_crime_totals.plot(kind='bar', figsize=(12,6), color='salmon')
plt.title('Total IPC Crimes by State/UT')
plt.xlabel('State/UT')
plt.ylabel('Total IPC Crimes')
plt.xticks(rotation=90)
plt.tight_layout()
plt.show()
```

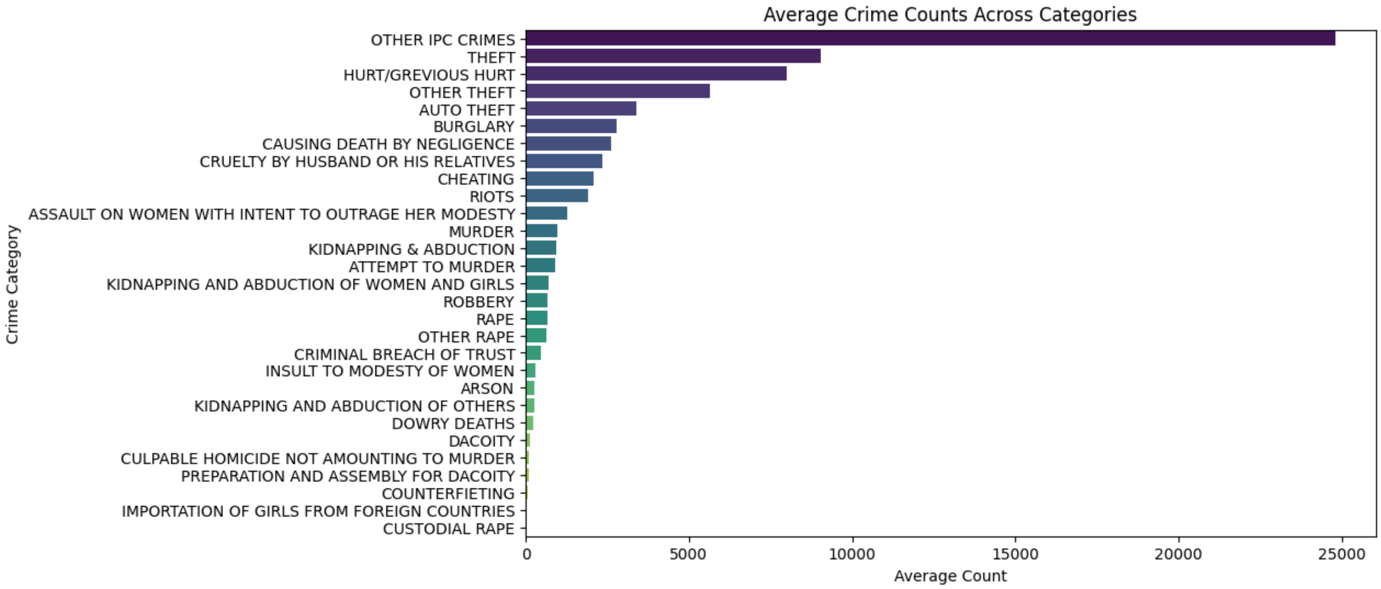


Delhi has the highest district-level crime, followed by major cities like Bangalore and Mumbai. Maharashtra and Madhya Pradesh report the highest state-level crime, while smaller territories like Lakshadweep have the lowest. Crime rates vary significantly across India, influenced by population and socioeconomic factors.

Q) Further, analyze how crime patterns differ across various crime categories in urban vs. rural districts (or using a proxy like population if urban/rural data is unavailable) and investigate whether there is a correlation between different crime types such as murder and theft.

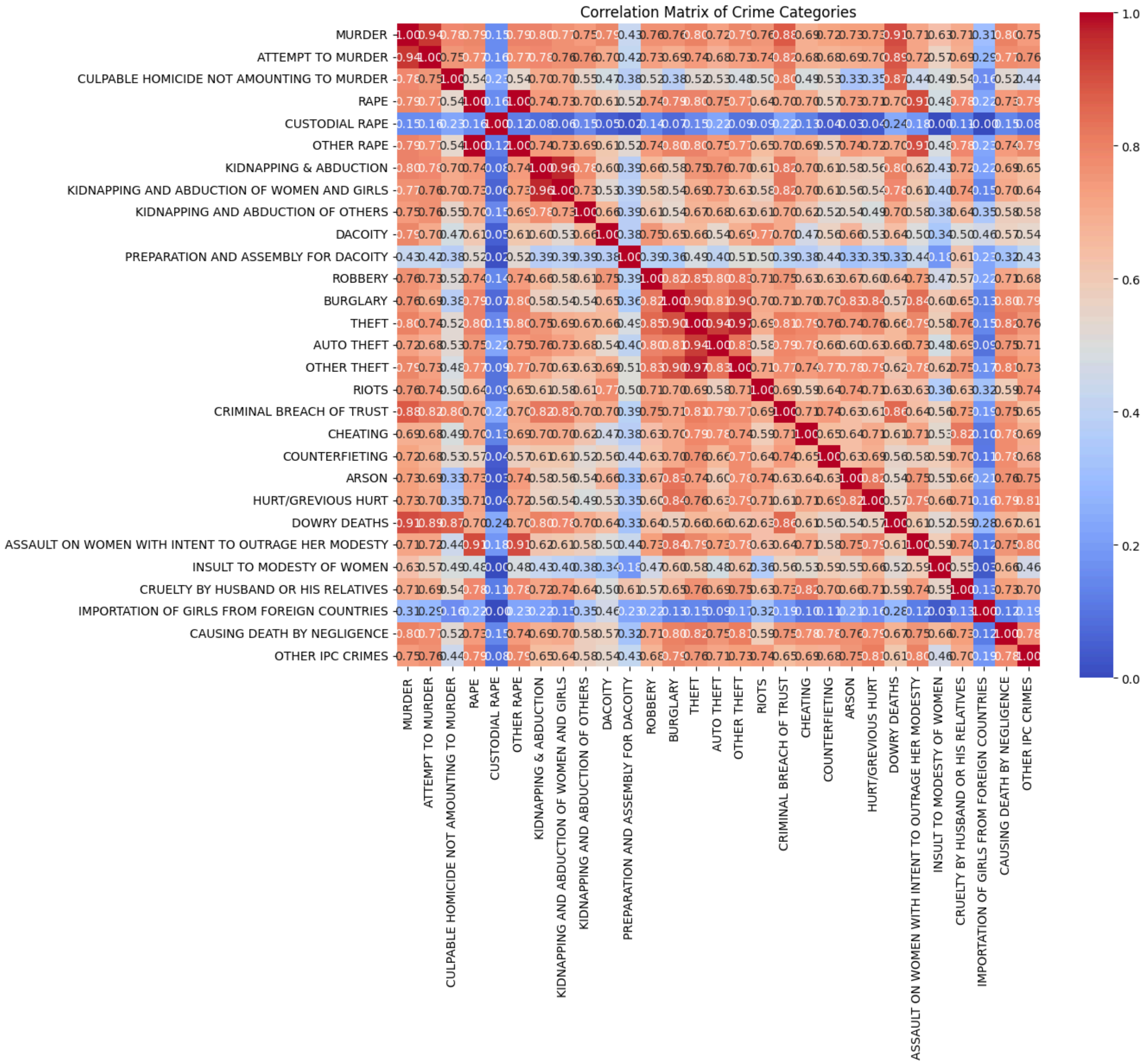
```
crime_categories = [
    'MURDER', 'ATTEMPT TO MURDER', 'CULPABLE HOMICIDE NOT AMOUNTING TO MURDER',
    'RAPE', 'CUSTODIAL RAPE', 'OTHER RAPE', 'KIDNAPPING & ABDUCTION',
    'KIDNAPPING AND ABDUCTION OF WOMEN AND GIRLS', 'KIDNAPPING AND ABDUCTION OF OTHERS',
    'DACOITY', 'PREPARATION AND ASSEMBLY FOR DACOITY', 'ROBBERY', 'BURGLARY',
    'THEFT', 'AUTO THEFT', 'OTHER THEFT', 'RIOTS', 'CRIMINAL BREACH OF TRUST',
    'CHEATING', 'COUNTERFIETING', 'ARSON', 'HURT/GREVIOUS HURT', 'DOWRY DEATHS',
    'ASSAULT ON WOMEN WITH INTENT TO OUTRAGE HER MODESTY', 'INSULT TO MODESTY OF WOMEN',
    'CRUELTY BY HUSBAND OR HIS RELATIVES', 'IMPORTATION OF GIRLS FROM FOREIGN COUNTRIES',
    'CAUSING DEATH BY NEGLIGENCE', 'OTHER IPC CRIMES'
]

# Group by district and compute total crimes per category
district_crime_pattern = df.groupby('DISTRICT')[crime_categories].sum()


# Plot average crime type distribution across districts
average_crime_distribution = district_crime_pattern.mean().sort_values(ascending=False)

plt.figure(figsize=(10,6))
sns.barplot(x=average_crime_distribution.values, y=average_crime_distribution.index, palette="viridis")
plt.title('Average Crime Counts Across Categories')
plt.xlabel('Average Count')
plt.ylabel('Crime Category')
plt.show()
```

```
<ipython-input-37-95584e9aba6d>:21: FutureWarning:


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



From this we can see that theft is the most happening crime compared to all other crimes with cases nearing to 25000.

```
# Compute correlation matrix for all major crime categories
correlation_matrix = df[crime_categories].corr()

plt.figure(figsize=(12,10))
sns.heatmap(correlation_matrix, cmap='coolwarm', annot=True, fmt='.2f', square=True)
plt.title('Correlation Matrix of Crime Categories')
plt.show()

# Direct correlation between Murder and Theft
murder_theft_corr = df['MURDER'].corr(df['THEFT'])
print(f"Correlation between Murder and Theft: {murder_theft_corr:.2f}")
```



```
Correlation between Murder and Theft: 0.80
```

The correlation matrix highlights strong positive relationships among crime types, particularly within theft-related, violent, and crimes-against-women categories, suggesting interconnected patterns. "Importation of girls" shows weak correlations, indicating a distinct issue. Overall, crime types tend to co-occur, though correlation doesn't imply causation. As the number of murders increases, the number of thefts also tends to increase. As we can say that when ever a murder happens there is a chance that theft also happen.

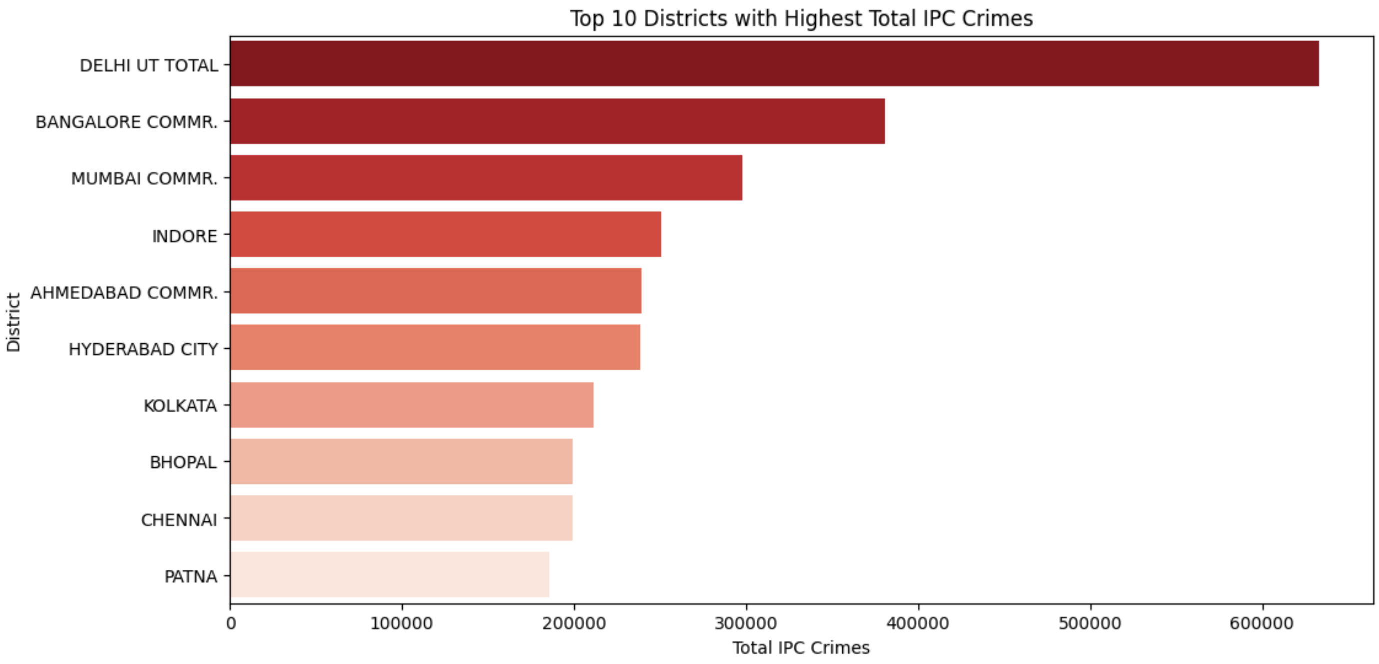
Visualization Questions

Q)How can visualizations be used to explore crime patterns in India by identifying the top 10 districts with the highest crime rates, understanding the overall distribution of total IPC crimes, analyzing crime density across different states, and comparing trends in violent crimes such as murder and rape across various districts?

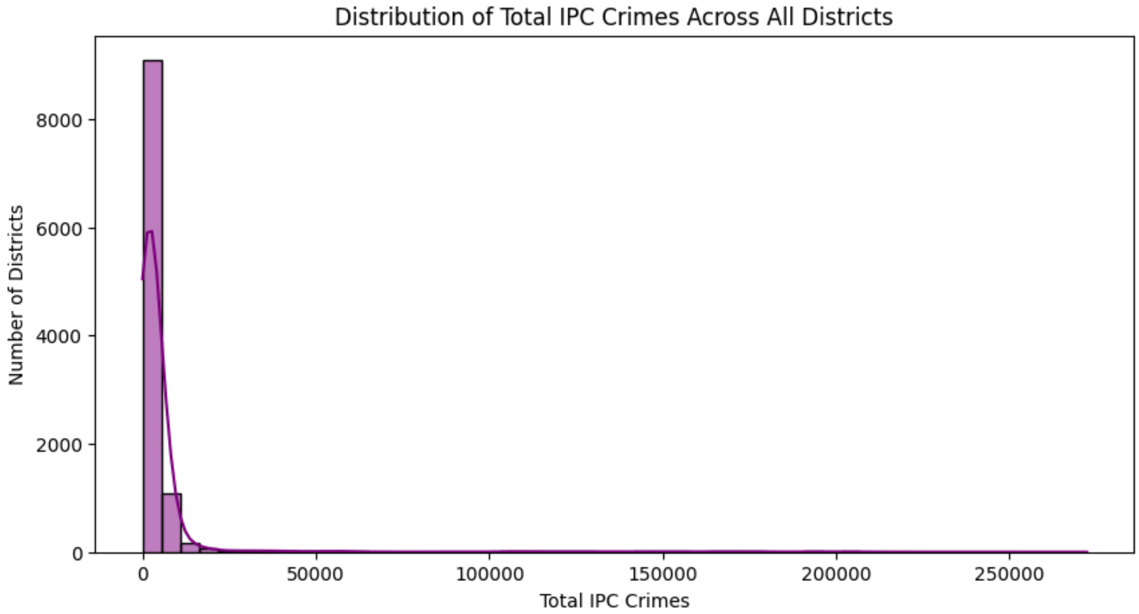
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

```
# Group data by district and sum total IPC crimes
top_districts = df.groupby('DISTRICT')['TOTAL IPC CRIMES'].sum().sort_values(ascending=False).head(11)
top_districts.drop('TOTAL',inplace=True)

# Plot
plt.figure(figsize=(12,6))
sns.barplot(x=top_districts.values, y=top_districts.index, palette='Reds_r')
plt.title('Top 10 Districts with Highest Total IPC Crimes')
plt.xlabel('Total IPC Crimes')
plt.ylabel('District')
plt.show()
```

```
<ipython-input-39-e18b7ba91529>:11: FutureWarning:


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

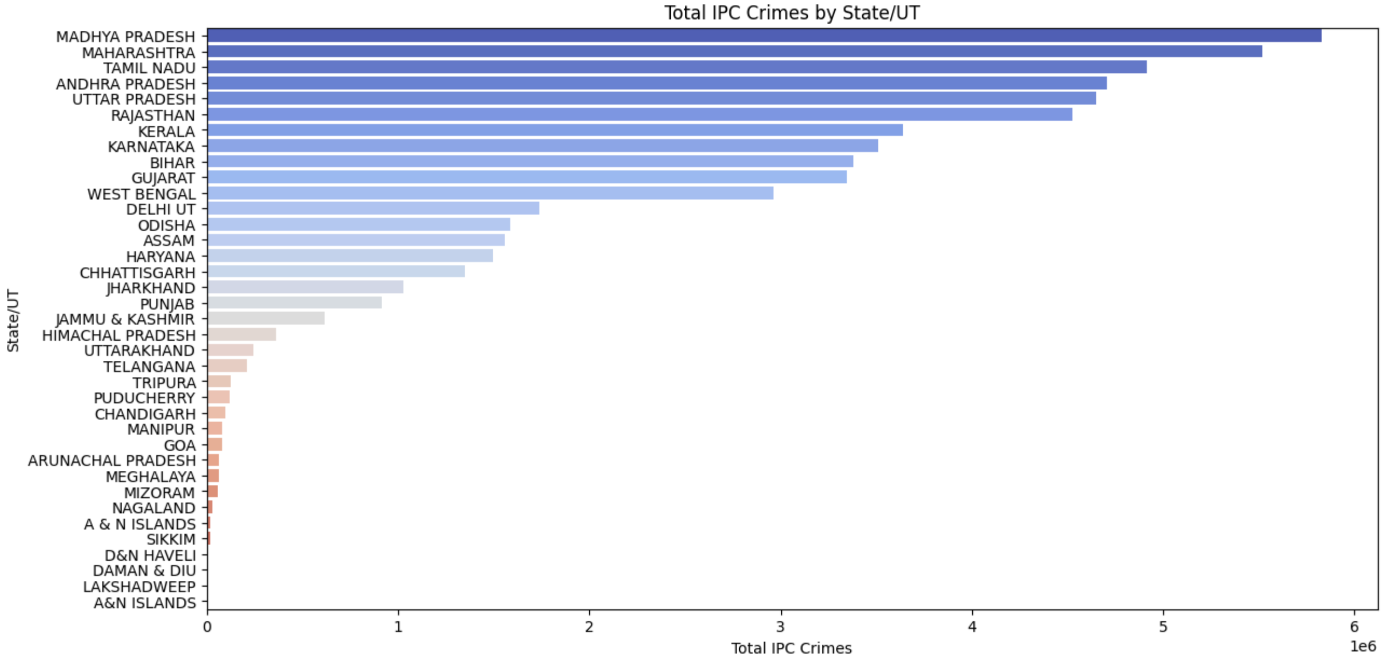


```
plt.figure(figsize=(10,5))
sns.histplot(df['TOTAL IPC CRIMES'], bins=50, kde=True, color='purple')
plt.title('Distribution of Total IPC Crimes Across All Districts')
plt.xlabel('Total IPC Crimes')
plt.ylabel('Number of Districts')
plt.show()
```

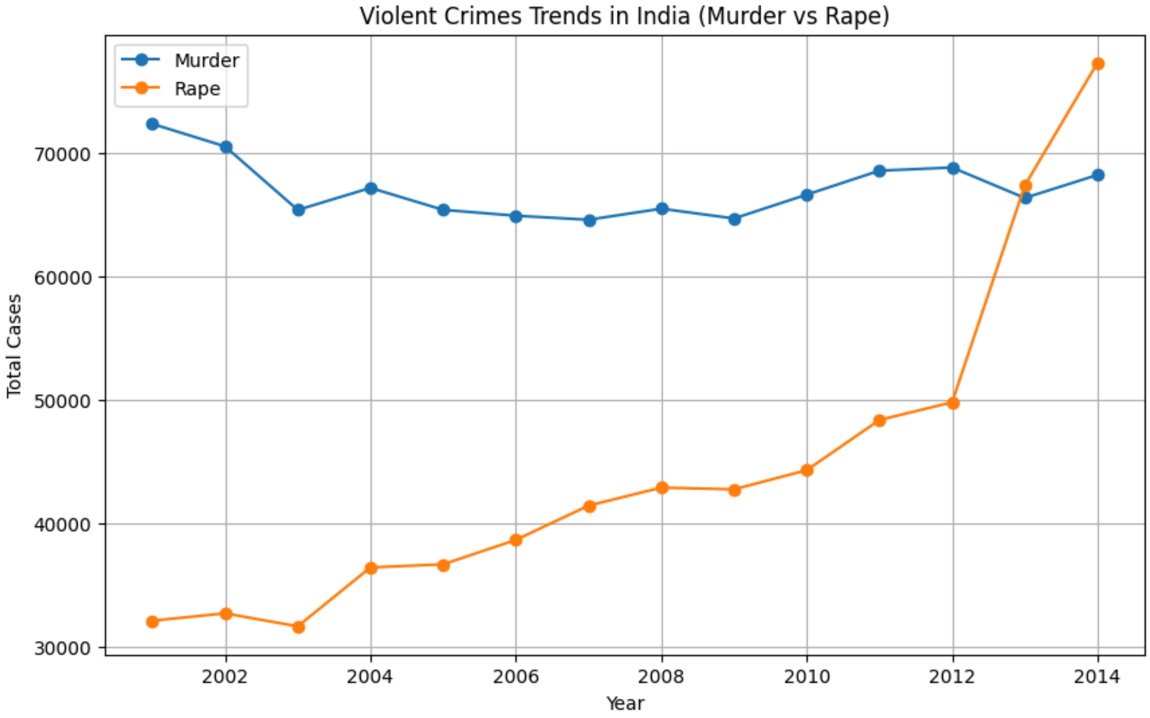

Distribution of Total IPC Crimes Across All Districts

```
state_crime = df.groupby('STATE/UT')['TOTAL IPC CRIMES'].sum().sort_values(ascending=False)

plt.figure(figsize=(14,7))
sns.barplot(x=state_crime.values, y=state_crime.index, palette='coolwarm')
plt.title('Total IPC Crimes by State/UT')
plt.xlabel('Total IPC Crimes')
plt.ylabel('State/UT')
plt.show()
```

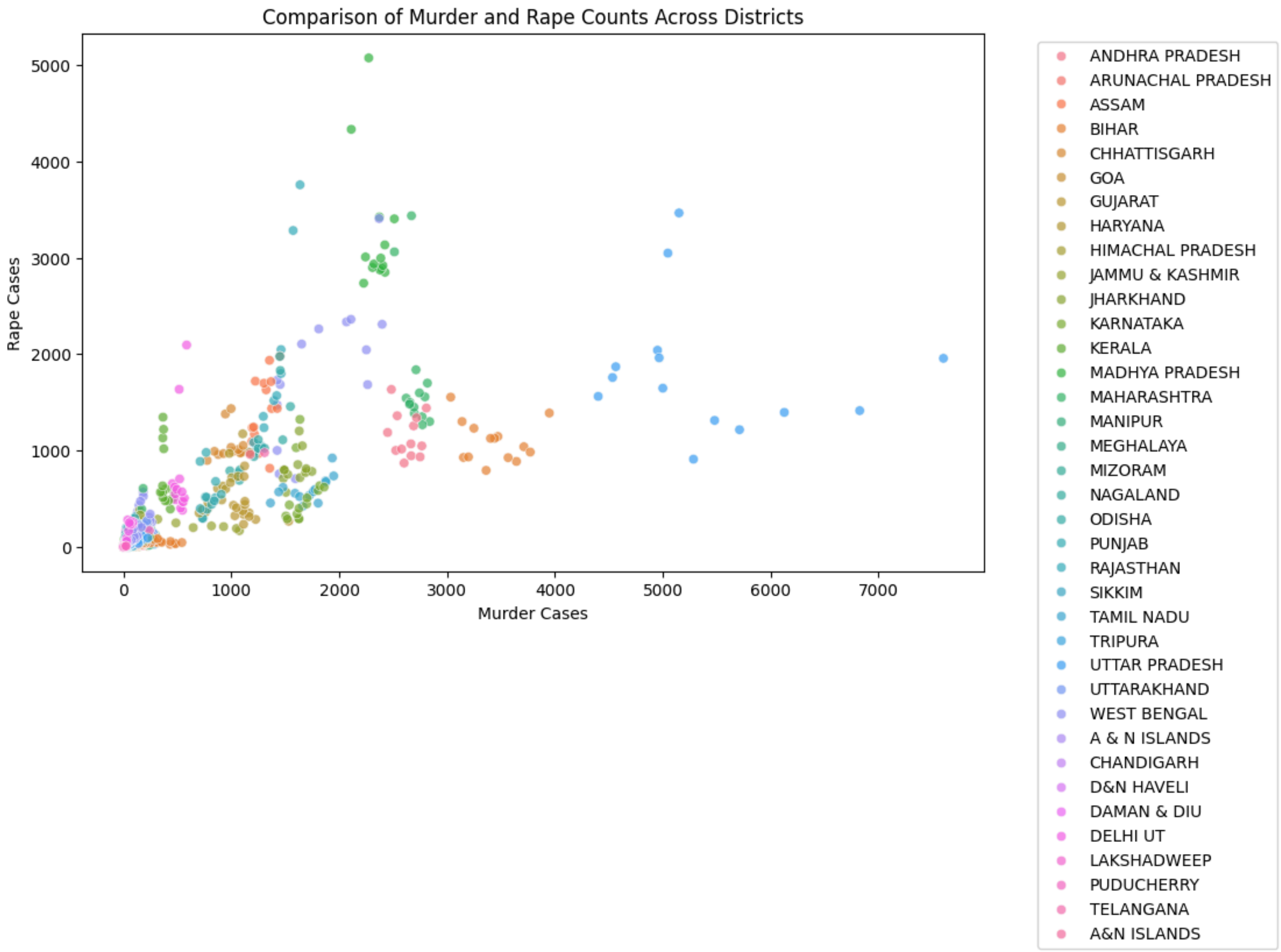
```
<ipython-input-41-561276aa092b>:4: FutureWarning:


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


Total IPC Crimes by State/UT

```
violent_crimes = df.groupby(['YEAR'])[['MURDER', 'RAPE']].sum()

plt.figure(figsize=(10,6))
plt.plot(violent_crimes.index, violent_crimes['MURDER'], marker='o', label='Murder')
plt.plot(violent_crimes.index, violent_crimes['RAPE'], marker='o', label='Rape')
plt.title('Violent Crimes Trends in India (Murder vs Rape)')
plt.xlabel('Year')
plt.ylabel('Total Cases')
plt.legend()
plt.grid(True)
plt.show()
```



```
plt.figure(figsize=(10,6))
sns.scatterplot(data=df, x='MURDER', y='RAPE', hue='STATE/UT', alpha=0.7)
plt.title('Comparison of Murder and Rape Counts Across Districts')
plt.xlabel('Murder Cases')
plt.ylabel('Rape Cases')
plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left')
plt.show()
```

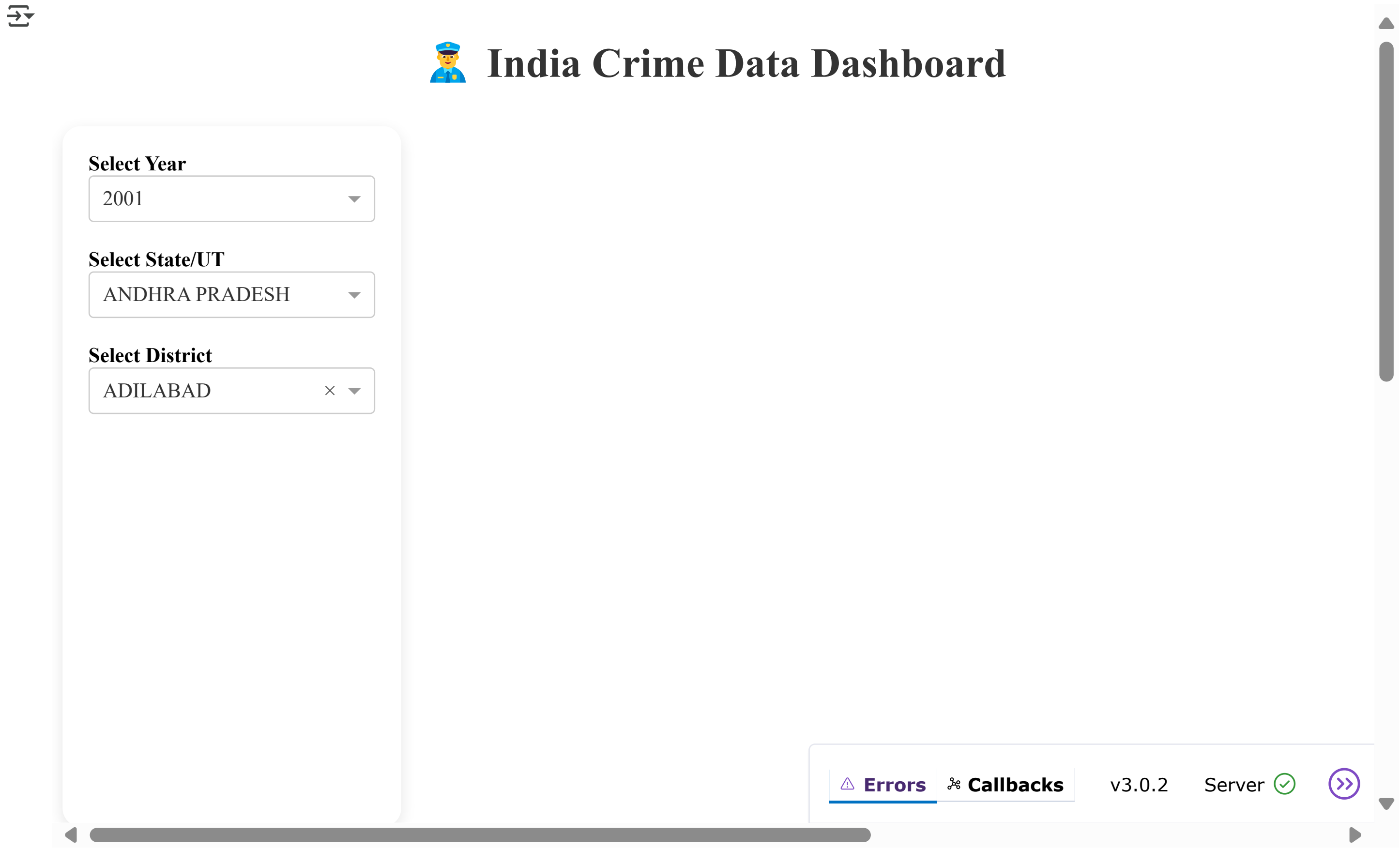


Madhya Pradesh has the highest crimes when states are compared and delhi Ut has high crime rate when districts are compared.

rape cases has been suddenly increased from year 2012 till then murder cases were higher.

Q)Create an interactive dashboard that allows users to filter crime data by year, state, and district.

```
import pandas as pd
import plotly.express as px
#%pip install dash
import dash
from dash import dcc, html
from dash.dependencies import Input, Output


# Initialize the Dash app
app = dash.Dash(__name__)
app.title = "India Crime Data Dashboard"

# App layout with enhanced UI
app.layout = html.Div([
    html.Div([
        html.H1("\U0001F46E♂️ India Crime Data Dashboard", style={'textAlign': 'center', 'color': '#333', 'marginBottom': '30px'}),

        html.Div([
            html.Label("Select Year", style={'fontWeight': 'bold'}),
            dcc.Dropdown(
                id='year-dropdown',
                options=[{'label': year, 'value': year} for year in sorted(df['YEAR'].unique())],
                value=df['YEAR'].min(),
                clearable=False,
                style={'marginBottom': '20px'}
            ),

            html.Label("Select State/UT", style={'fontWeight': 'bold'}),
            dcc.Dropdown(
                id='state-dropdown',
                options=[{'label': state, 'value': state} for state in sorted(df['STATE/UT'].unique())],
                value=df['STATE/UT'].unique()[0],
                clearable=False,
                style={'marginBottom': '20px'}
            ),

            html.Label("Select District", style={'fontWeight': 'bold'}),
            dcc.Dropdown(
                id='district-dropdown',
                style={'marginBottom': '20px'}
            )
        ], style={
            'width': '22%', 'display': 'inline-block', 'verticalAlign': 'top',
            'padding': '20px', 'boxShadow': '0 4px 12px rgba(0,0,0,0.1)',
            'borderRadius': '14px', 'backgroundColor': '#fafafa',
            'minHeight': '500px'
        }),

        html.Div([
            dcc.Graph(id='crime-graph', config={'displayModeBar': False})
        ], style={
            'width': '76%', 'display': 'inline-block', 'padding': '20px',
            'verticalAlign': 'top', 'minHeight': '700px'
        })
    ], style={'maxWidth': '1800px', 'margin': '0 auto'})
])

# Callback to populate district dropdown based on state selection
@app.callback(
    Output('district-dropdown', 'options'),
    Output('district-dropdown', 'value'),
    Input('state-dropdown', 'value')
)
def set_district_options(selected_state):
    filtered_districts = df[df['STATE/UT'] == selected_state]['DISTRICT'].unique()
    options = [{'label': district, 'value': district} for district in sorted(filtered_districts)]
    value = filtered_districts[0] if len(filtered_districts) > 0 else None
    return options, value

# Callback to update graph based on selected filters
@app.callback(
    Output('crime-graph', 'figure'),
    [Input('year-dropdown', 'value'),
     Input('state-dropdown', 'value'),
     Input('district-dropdown', 'value')]
)
def update_graph(selected_year, selected_state, selected_district):
    filtered_df = df[(df['YEAR'] == selected_year) &
                     (df['STATE/UT'] == selected_state) &
                     (df['DISTRICT'] == selected_district)]

    crime_cols = [col for col in df.columns if col not in ['STATE/UT', 'DISTRICT', 'YEAR']]
    melted_df = filtered_df.melt(id_vars=['STATE/UT', 'DISTRICT', 'YEAR'],
                                 value_vars=crime_cols, var_name='Crime Type', value_name='Count')

    fig = px.bar(
        melted_df, x='Crime Type', y='Count',
        title=f'Crime Breakdown for {selected_district}, {selected_state} ({selected_year})',
        labels={'Count': 'Number of Cases'},
        text_auto=True,
        color='Crime Type',
        color_discrete_sequence=px.colors.qualitative.Safe
    )
    fig.update_layout(
        xaxis_tickangle=-45,
        plot_bgcolor='#f9f9f9',
        paper_bgcolor='#fff',
        font=dict(color='#333', size=15),
        title_font_size=24,
        margin=dict(l=30, r=30, t=60, b=30),
        height=700,width=1600
    )

    return fig

if __name__ == '__main__':
    app.run(debug=True)
```



👮 India Crime Data Dashboard

Select Year
2001

Select State/UT
ANDHRA PRADESH

Select District
ADILABAD

Please check in the code to see the dynamic dashboard as its not visible in the pdf report.

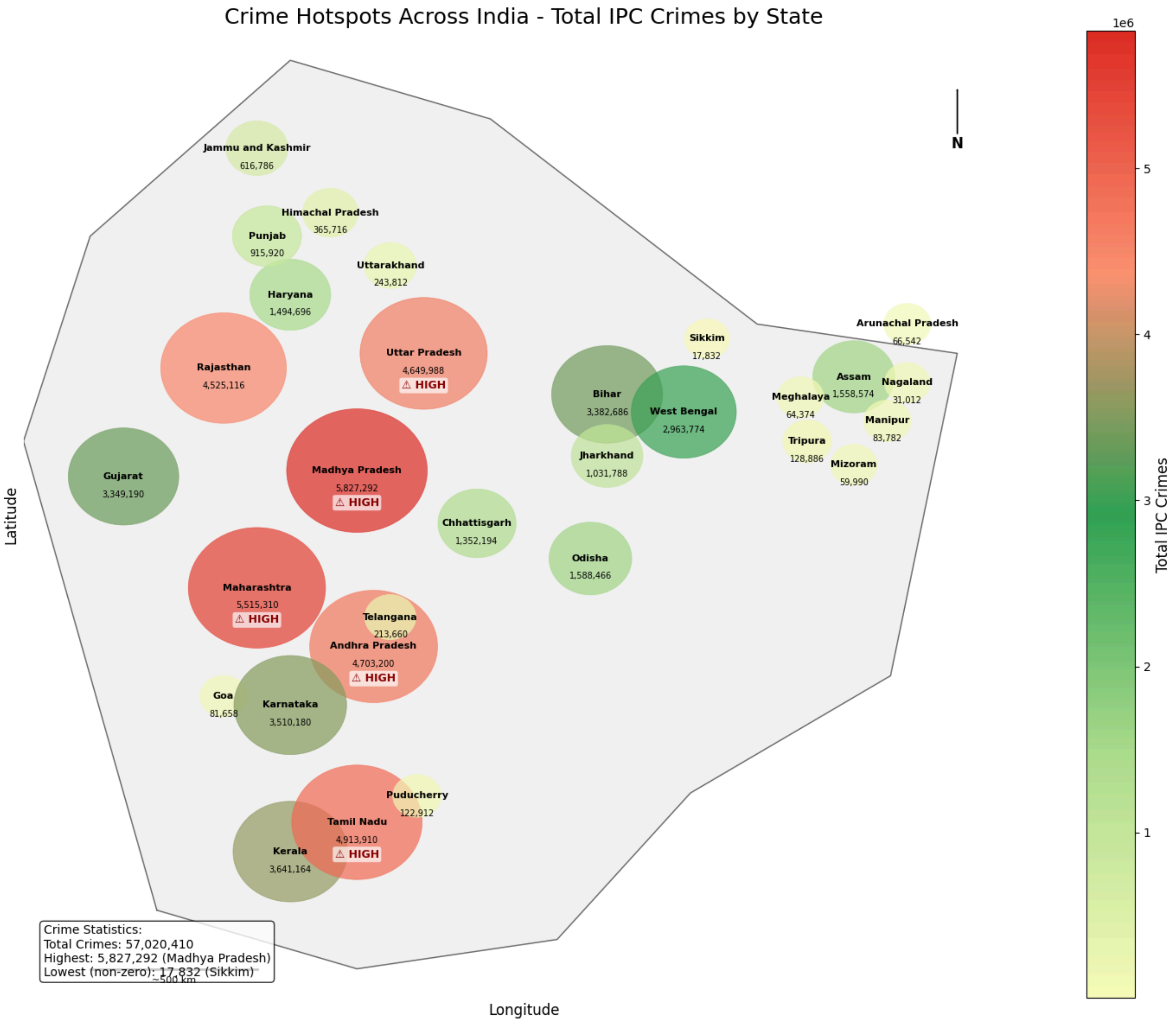
Q) Use a geospatial map to visualize crime hot spots across India. (Matplotlib)

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.colors import LinearSegmentedColormap
from matplotlib.cm import ScalarMappable
from matplotlib.patches import Polygon

# Create state points with coordinates
state_points = {
    'Andhra Pradesh': [78.5, 17.0],
    'Arunachal Pradesh': [94.5, 28.0],
    'Assam': [92.9, 26.2],
    'Bihar': [85.5, 25.6],
    'Chhattisgarh': [81.6, 21.2],
    'Goa': [74.0, 15.3],
    'Gujarat': [71.0, 22.8],
    'Haryana': [76.0, 29.0],
    'Himachal Pradesh': [77.2, 31.8],
    'Jammu and Kashmir': [75.0, 34.0],
    'Jharkhand': [85.5, 23.5],
    'Karnataka': [76.0, 15.0],
    'Kerala': [76.0, 10.0],
    'Madhya Pradesh': [78.0, 23.0],
    'Maharashtra': [75.0, 19.0],
    'Manipur': [93.9, 24.7],
    'Meghalaya': [91.3, 25.5],
    'Mizoram': [92.9, 23.2],
    'Nagaland': [94.5, 26.0],
    'Nagaland': [94.5, 26.0],
    'Odisha': [85.0, 20.0],
    'Punjab': [75.3, 31.0],
    'Rajasthan': [74.0, 26.5],
    'Sikkim': [88.5, 27.5],
    'Tamil Nadu': [78.0, 11.0],
    'Telangana': [79.0, 18.0],
    'Tripura': [91.5, 24.0],
    'Uttar Pradesh': [80.0, 27.0],
    'Uttarakhand': [79.0, 30.0],
    'West Bengal': [87.8, 25.0],
    'Delhi': [77.2, 28.6],
    'Puducherry': [79.8, 11.9]
}

# Simulated crime data based on your DataFrame structure
crime_data = df.groupby('STATE/UT')['TOTAL IPC CRIMES'].sum().reset_index()

# Map state names
state_name_mapping = {
    'ANDHRA PRADESH': 'Andhra Pradesh',
    'ARUNACHAL PRADESH': 'Arunachal Pradesh',
    'ASSAM': 'Assam',
    'BIHAR': 'Bihar',
    'CHHATTISGARH': 'Chhattisgarh',
    'GOA': 'Goa',
    'GUJARAT': 'Gujarat',
    'HARYANA': 'Haryana',
    'HIMACHAL PRADESH': 'Himachal Pradesh',
    'JAMMU & KASHMIR': 'Jammu and Kashmir',
    'JHARKHAND': 'Jharkhand',
    'KARNATAKA': 'Karnataka',
    'KERALA': 'Kerala',
    'MADHYA PRADESH': 'Madhya Pradesh',
    'MAHARASHTRA': 'Maharashtra',
    'MANIPUR': 'Manipur',
    'MEGHALAYA': 'Meghalaya',
    'MIZORAM': 'Mizoram',
    'NAGALAND': 'Nagaland',
    'ODISHA': 'Odisha',
    'PUNJAB': 'Punjab',
    'RAJASTHAN': 'Rajasthan',
    'SIKKIM': 'Sikkim',
    'TAMIL NADU': 'Tamil Nadu',
    'TELANGANA': 'Telangana',
    'TRIPURA': 'Tripura',
    'UTTAR PRADESH': 'Uttar Pradesh',
    'UTTARAKHAND': 'Uttarakhand',
    'WEST BENGAL': 'West Bengal',
    'DELHI': 'Delhi',
    'PUDUCHERRY': 'Puducherry'
}

# Create a dictionary mapping state names to crime values
crime_values = {}
for i, row in crime_data.iterrows():
    state_name = state_name_mapping.get(row['STATE/UT'])
    if state_name:
        crime_values[state_name] = row['TOTAL IPC CRIMES']

# Create a figure and axis
fig, ax = plt.subplots(figsize=(15, 12))

# Set India map boundaries
ax.set_xlim(68, 98)
ax.set_ylim(5, 38)

# Draw a simplified India outline
india_outline = np.array([
    [72, 8], [78, 6], [84, 7], [88, 12],
    [94, 16], [96, 27], [90, 28], [82, 35],
    [76, 37], [70, 31], [68, 24], [72, 8]
])
ax.plot(india_outline[:, 0], india_outline[:, 1], 'k-', linewidth=1, alpha=0.5)

# Fill the outline with a light color
ax.add_patch(Polygon(india_outline, facecolor='lightgray', edgecolor='gray', alpha=0.3))
```

```
# Create a custom colormap
colors = ['#f7fcb9', '#addd8e', '#31a354', '#fc9272', '#de2d26']
cmap = LinearSegmentedColormap.from_list('crime_cmap', colors)

# Normalize crime values for sizing and coloring
min_crime = min(crime_data['TOTAL IPC CRIMES'])
max_crime = max(crime_data['TOTAL IPC CRIMES'])
norm = plt.Normalize(min_crime, max_crime)

# Plot each state as a circle sized and colored by crime rate
for state, coords in state_points.items():
    crime_value = crime_values.get(state, 0)
    if crime_value > 0:
        # Scale circle size based on crime value
        size = 100 + (crime_value / max_crime) * 900
        # Plot the circle
        circle = plt.Circle((coords[0], coords[1]),
                            radius=np.sqrt(size) / 15,  # Convert size to radius with sqrt
                            color=cmap(norm(crime_value)),
                            alpha=0.7,
                            edgecolor='black',
                            linewidth=0.5)
        ax.add_patch(circle)

        # Add state name
        ax.annotate(state,
                  xy=(coords[0], coords[1]),
                  ha='center',
                  va='center',
                  fontsize=8,
                  fontweight='bold')

        # Add crime count
        ax.annotate(f"{int(crime_value):,}",
                  xy=(coords[0], coords[1] - 0.7),
                  ha='center',
                  fontsize=7,
                  color='black')

# Create a scatter point for legend (not visible in plot)
sc = ax.scatter([], [], c=[], cmap=cmap, norm=norm, s=0)

# Add colorbar
cbar = plt.colorbar(sc, ax=ax)
cbar.set_label('Total IPC Crimes', size=12)

# Add a text box with crime statistics
state_with_max = max(crime_values.items(), key=lambda x: x[1])[0]
state_with_min = min({k: v for k, v in crime_values.items() if v > 0}.items(), key=lambda x: x[1])[0]

textstr = '\n'.join((
    'Crime Statistics:',
    f'Total Crimes: {int(sum(crime_values.values())):,}',
    f'Highest: {int(max(crime_values.values())):,} ({state_with_max})',
    f'Lowest (non-zero): {int(min([v for v in crime_values.values() if v > 0])):,} ({state_with_min})'
))
props = dict(boxstyle='round', facecolor='white', alpha=0.8)
ax.text(0.02, 0.02, textstr, transform=ax.transAxes, fontsize=10,
        verticalalignment='bottom', bbox=props)

# Mark the top 5 crime states
top_crime_states = sorted(crime_values.items(), key=lambda x: x[1], reverse=True)[:5]
for state, value in top_crime_states:
    coords = state_points[state]
    ax.annotate("⚠️ HIGH",
              xy=(coords[0], coords[1] - 1.2),
              ha='center',
              fontsize=9,
              fontweight='bold',
              color='darkred',
              bbox=dict(facecolor='white', alpha=0.7, edgecolor='none', boxstyle='round,pad=0.2'))

# Add title and labels
plt.title('Crime Hotspots Across India - Total IPC Crimes by State', fontsize=18)
plt.xlabel('Longitude', fontsize=12)
plt.ylabel('Latitude', fontsize=12)
```

```
# Remove axis ticks for cleaner map
plt.xticks([])
plt.yticks([])
for spine in ax.spines.values():
    spine.set_visible(False)

# Add north arrow
arrow_props = dict(facecolor='black', width=0.1, headwidth=0.5, headlength=0.5)
ax.annotate('N', xy=(96, 36), xytext=(96, 34),
          arrowprops=arrow_props, ha='center', fontsize=12, fontweight='bold')

# Add a scale indicator
ax.plot([70, 75], [6, 6], 'k-', linewidth=2)
ax.text(72.5, 5.5, '~500 km', ha='center', fontsize=8)

plt.tight_layout()
plt.show()

# If you want to focus on specific crime types instead of total:
def plot_specific_crime(crime_type):
    """
    Plot a map focused on a specific crime type

    Parameters:
    crime_type (str): Column name of the crime type to visualize
    """
    # In your actual code, you would use your real DataFrame with specific crime columns
    # For demonstration purposes, let's create sample data for one crime type

    # Example for visualizing Murder rates
    sample_data = pd.DataFrame({
        'STATE/UT': crime_data['STATE/UT'],
        crime_type: np.random.randint(100, 5000, size=len(crime_data))
    })

    # Then create visualization similar to above but using sample_data[crime_type] instead of TOTAL IPC CRIMES

# Uncomment to visualize a specific crime type
# plot_specific_crime('MURDER')
```

```
<ipython-input-45-4c370aee17e5>:122: UserWarning:

Setting the 'color' property will override the edgecolor or facecolor properties.
```

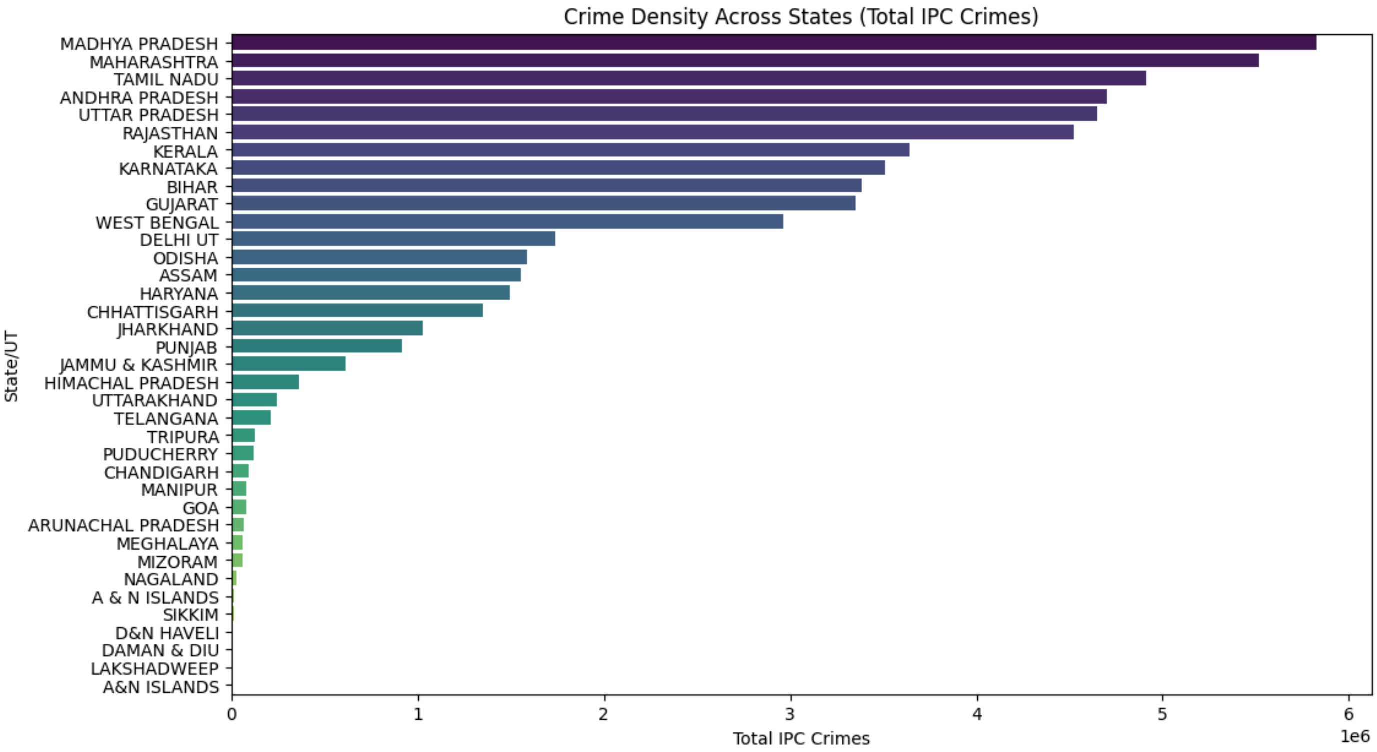


Advanced Questions

Q) Identify the state with the lowest crime rate and analyze why it might be lower than others.

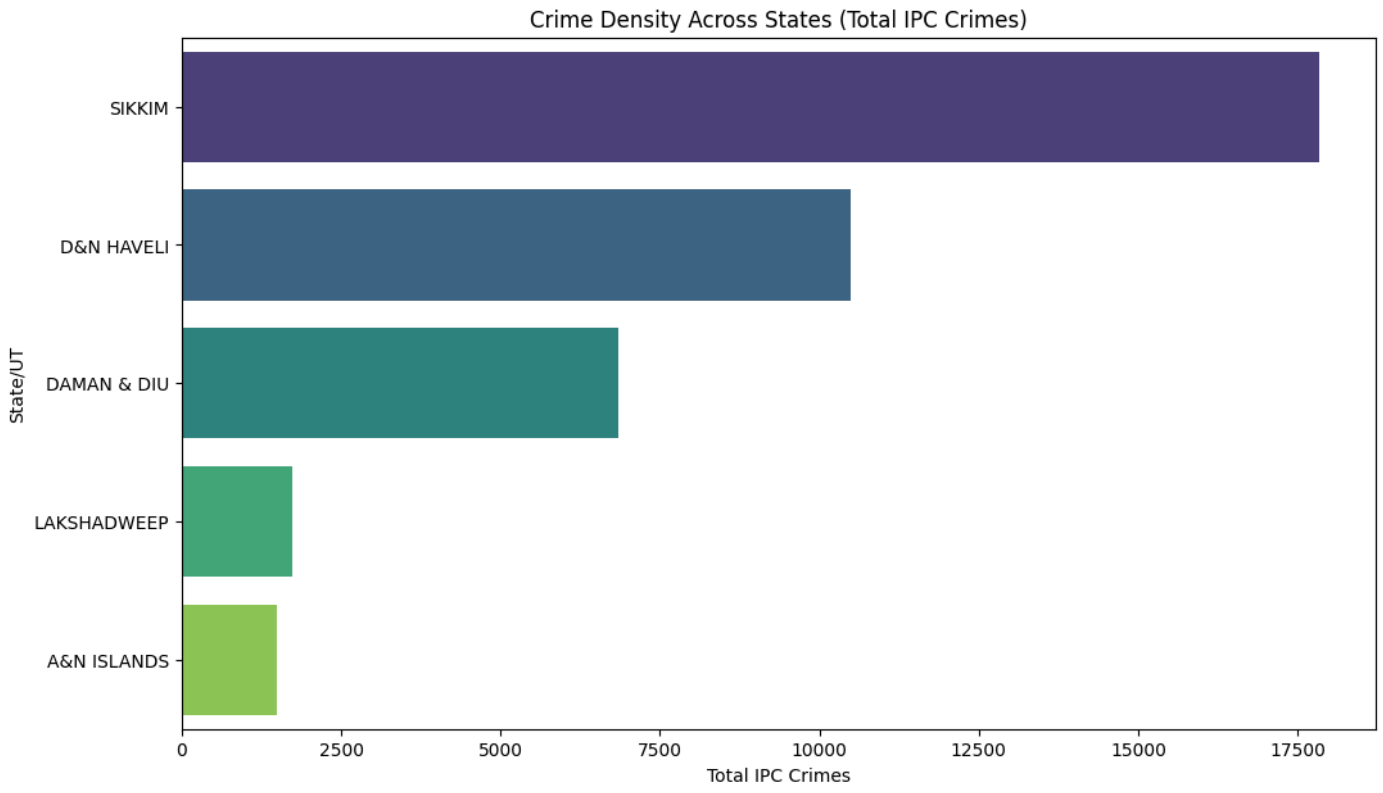
```
# Group by state and sum total crimes
state_crime = df.groupby('STATE/UT')['TOTAL IPC CRIMES'].sum().sort_values(ascending=False)

plt.figure(figsize=(12,7))
sns.barplot(x=state_crime.values, y=state_crime.index, palette='viridis')
plt.title('Crime Density Across States (Total IPC Crimes)')
plt.xlabel('Total IPC Crimes')
plt.ylabel('State/UT')
plt.show()
```

```
<ipython-input-46-a3597140254a>:5: FutureWarning:


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



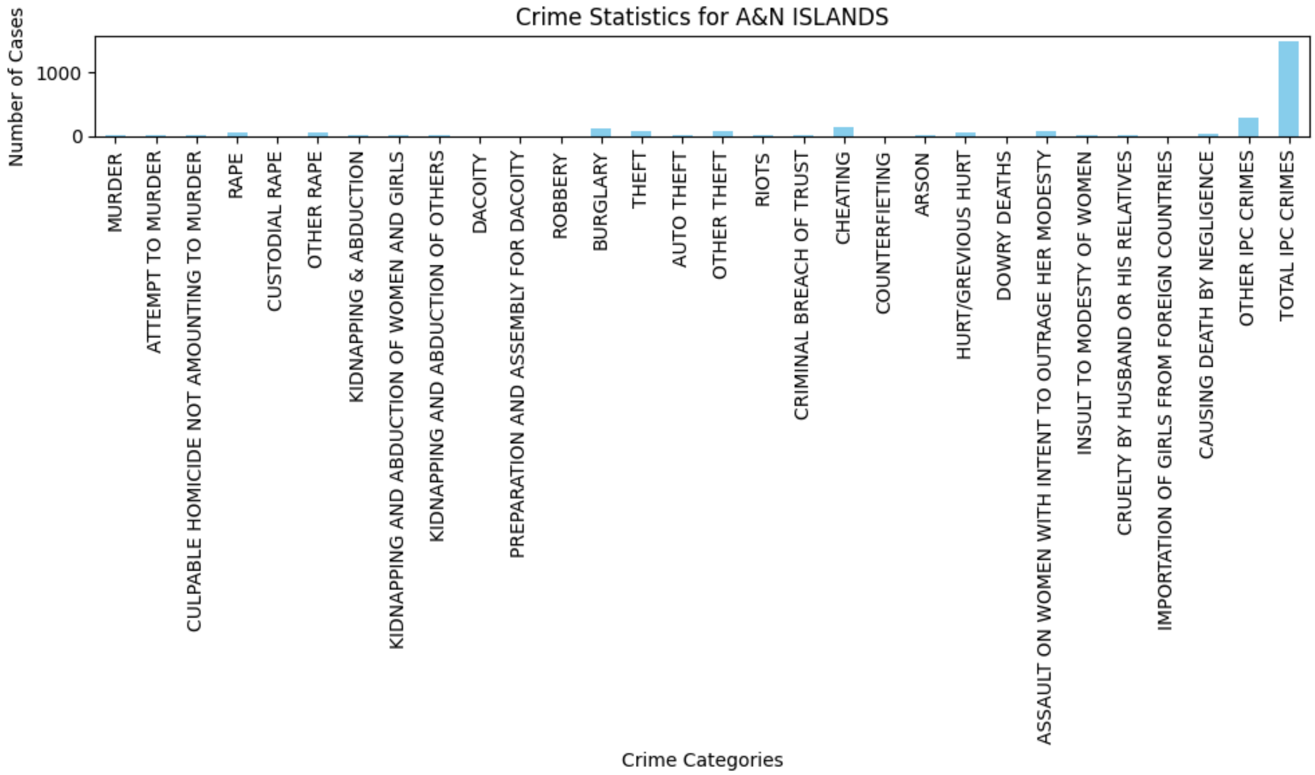
```
state_crime5 = state_crime.tail(5)

plt.figure(figsize=(12,7))
sns.barplot(x=state_crime5.values, y=state_crime5.index, palette='viridis')
plt.title('Crime Density Across States (Total IPC Crimes)')
plt.xlabel('Total IPC Crimes')
plt.ylabel('State/UT')
plt.show()
```

```
<ipython-input-47-70de6c7c3c80>:4: FutureWarning:


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



A&N Islands have the least number of crimes

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

state_name = 'A&N ISLANDS'

# Filter the dataframe for the given state
df_state = df[df['STATE/UT'] == state_name]

# Sum the crimes for each year
df_state_sum = df_state.drop(columns=['STATE/UT', 'DISTRICT', 'YEAR']).sum()

# Plot the crimes for the state
plt.figure(figsize=(10, 6))
df_state_sum.plot(kind='bar', color='skyblue')
plt.title(f"Crime Statistics for {state_name}")
plt.xlabel("Crime Categories")
plt.ylabel("Number of Cases")
plt.xticks(rotation=90)
plt.tight_layout()
plt.show()
```



Reasons:

1.Smaller Population: The A&N Islands have a relatively small population compared to mainland states.

2.Tourism-based Economy: A&N Islands largely rely on tourism, which could lead to a more stable economic environment. When tourism is a major economic driver, local governments might prioritize safety to maintain a positive reputation among tourists.

3.Cultural Values: The cultural values of residents in the A&N Islands might place more emphasis on community, non-violence, and peaceful coexistence, leading to lower crime rates.

4.Underreporting: It's also worth considering the possibility of underreporting of crime in smaller or more isolated areas. Sometimes, residents may not report crimes due to distrust in authorities or fear of repercussions.

Q) Find the most common type of crime committed in each district.

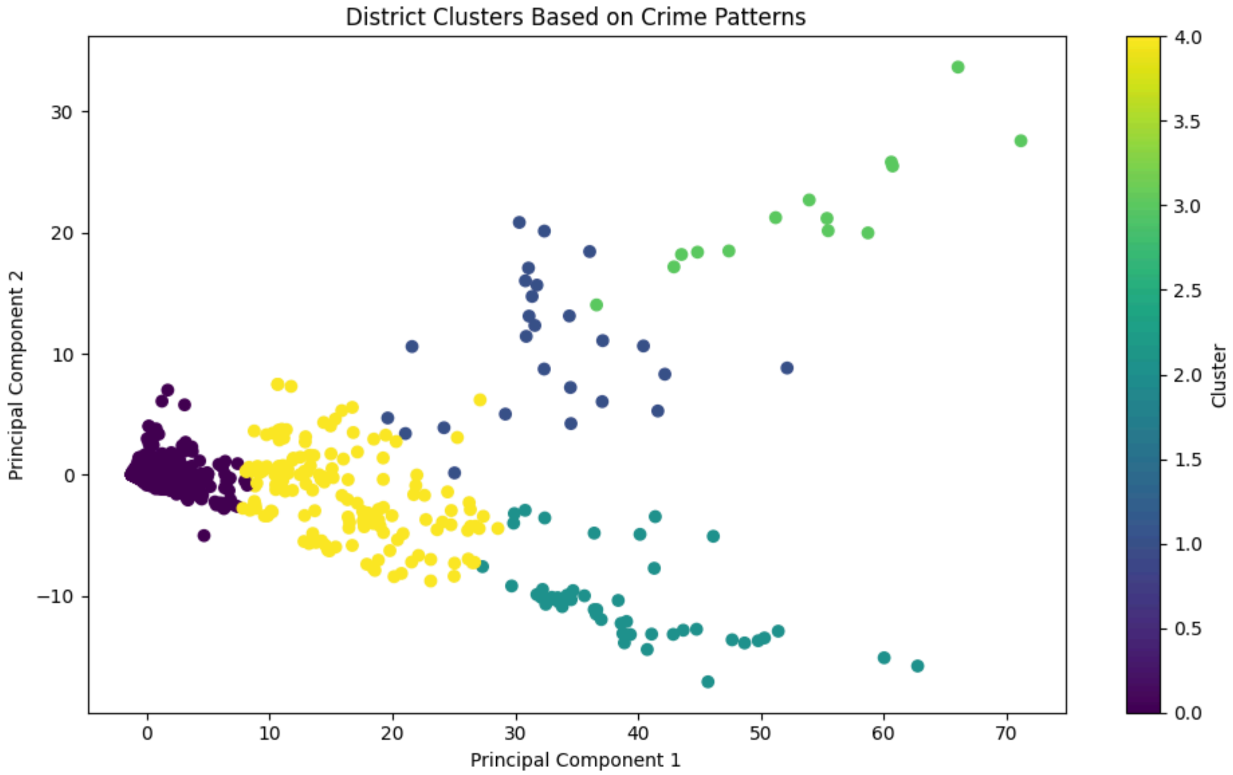
```
import pandas as pd


# List of crime columns to consider
crime_columns = [
    'MURDER', 'ATTEMPT TO MURDER', 'CULPABLE HOMICIDE NOT AMOUNTING TO MURDER',
    'RAPE', 'CUSTODIAL RAPE', 'OTHER RAPE', 'KIDNAPPING & ABDUCTION',
    'KIDNAPPING AND ABDUCTION OF WOMEN AND GIRLS', 'KIDNAPPING AND ABDUCTION OF OTHERS',
    'DACOITY', 'PREPARATION AND ASSEMBLY FOR DACOITY', 'ROBBERY', 'BURGLARY',
    'THEFT', 'AUTO THEFT', 'OTHER THEFT', 'RIOTS', 'CRIMINAL BREACH OF TRUST',
    'CHEATING', 'COUNTERFIETING', 'ARSON', 'HURT/GREVIOUS HURT', 'DOWRY DEATHS',
    'ASSAULT ON WOMEN WITH INTENT TO OUTRAGE HER MODESTY', 'INSULT TO MODESTY OF WOMEN',
    'CRUELTY BY HUSBAND OR HIS RELATIVES', 'IMPORTATION OF GIRLS FROM FOREIGN COUNTRIES',
    'CAUSING DEATH BY NEGLIGENCE'
]

# Group by 'DISTRICT' and sum the crime counts
df2=df.drop(columns=['TOTAL IPC CRIMES','OTHER IPC CRIMES'])
district_crimes = df2.groupby('DISTRICT')[crime_columns].sum()

# Find the most common crime for each district
most_common_crimes = district_crimes.idxmax(axis=1)
#most_common_crimes=most_common_crimes.iloc[:, 1:]

# Create a DataFrame with the most common crime for each district
most_common_crimes_df = pd.DataFrame({'MOST_COMMON_CRIME': most_common_crimes})
```

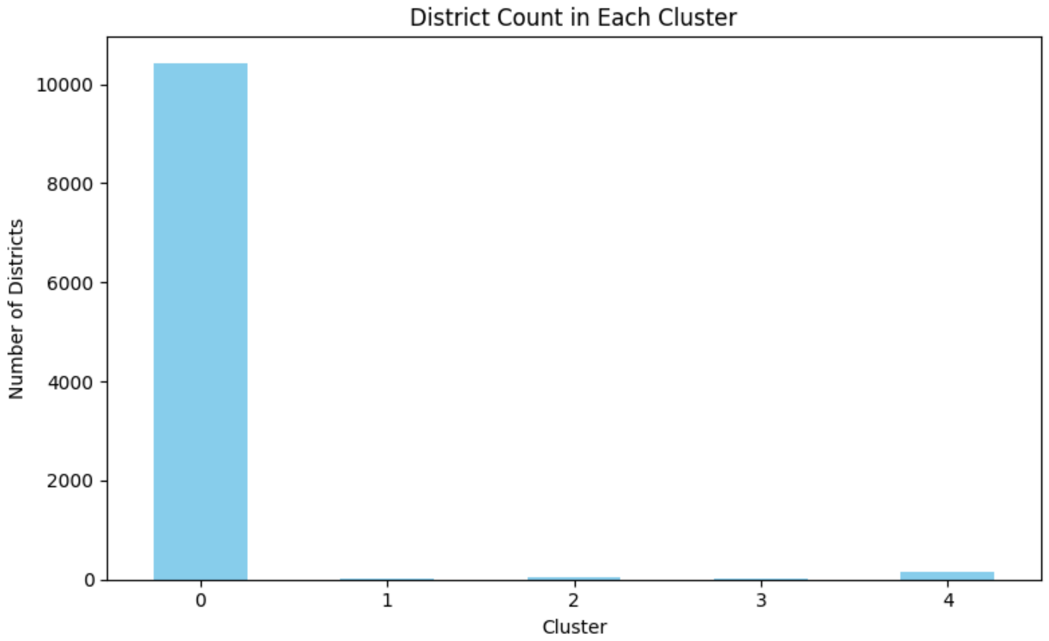
```
# Display the results
print("Most Common crime by district:",most_common_crimes_df)
```

```
Most Common crime by district:                                      MOST_COMMON_CRIME
DISTRICT
24 PARGANAS NORTH                                THEFT
24 PARGANAS SOUTH  CRUELTY BY HUSBAND OR HIS RELATIVES
A and N ISLANDS                                  THEFT
ADILABAD                            HURT/GREVIOUS HURT
AGAR                                HURT/GREVIOUS HURT
...                                                ...
WOKHA                                            THEFT
YADGIRI                             HURT/GREVIOUS HURT
YAMUNANAGAR                                      THEFT
YAVATMAL                                         THEFT
ZUNHEBOTO                                        THEFT

[954 rows x 1 columns]
```

Q) Apply clustering algorithms (e.g., K-Means) to group districts based on crime patterns

```
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA


# 1. Preprocessing
# Filter out non-crime-related columns (like STATE/UT, YEAR, and DISTRICT)
crime_columns = df.columns[3:-1]  # Excluding the first 3 columns (State/UT, District, Year) and the last column (Total IPC Crimes)

# Extract the crime data for clustering
crime_data = df[crime_columns]

# Normalize the crime data using StandardScaler
scaler = StandardScaler()
crime_data_scaled = scaler.fit_transform(crime_data)

# 2. Apply K-Means Clustering
# Choose the number of clusters (e.g., 5) - You can experiment with different values
kmeans = KMeans(n_clusters=5, random_state=42)
df['Cluster'] = kmeans.fit_predict(crime_data_scaled)

# 3. Visualize the clusters using PCA for dimensionality reduction (2D visualization)
pca = PCA(n_components=2)
principal_components = pca.fit_transform(crime_data_scaled)

# Create a DataFrame with PCA results and the cluster labels
pca_df = pd.DataFrame(data=principal_components, columns=['PC1', 'PC2'])
pca_df['Cluster'] = df['Cluster']

# Plot the clusters in 2D
plt.figure(figsize=(10, 6))
plt.scatter(pca_df['PC1'], pca_df['PC2'], c=pca_df['Cluster'], cmap='viridis', marker='o')
plt.title("District Clusters Based on Crime Patterns")
plt.xlabel("Principal Component 1")
plt.ylabel("Principal Component 2")
plt.colorbar(label='Cluster')
plt.tight_layout()
plt.show()

# 4. Plot the district count in each cluster
plt.figure(figsize=(8, 5))
cluster_counts = df['Cluster'].value_counts().sort_index()
cluster_counts.plot(kind='bar', color='skyblue')
plt.title("District Count in Each Cluster")
plt.xlabel("Cluster")
plt.ylabel("Number of Districts")
plt.xticks(rotation=0)
plt.tight_layout()
plt.show()
```





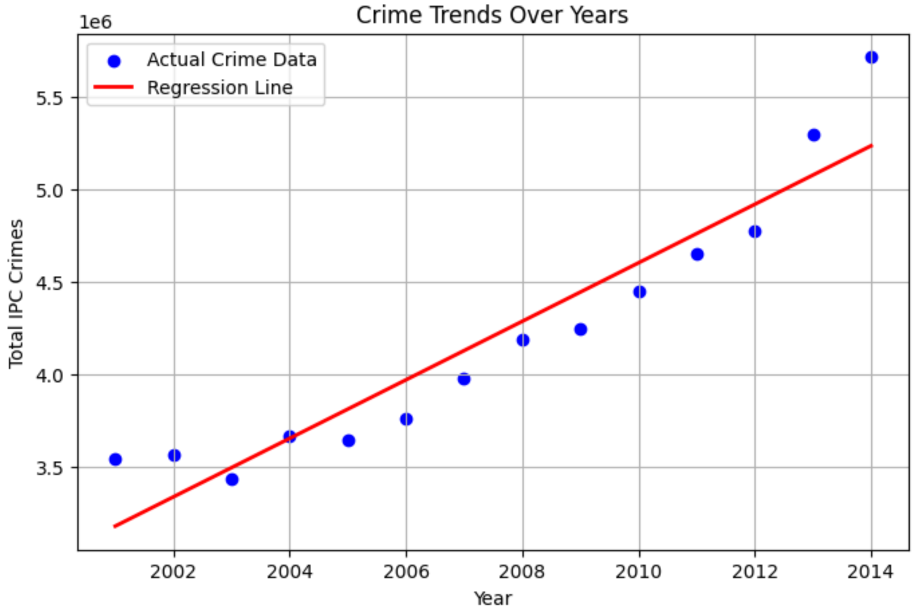
Based on the data we have we have divided them into 5 groups and we can see that most districts comes under 1st cluster.

Q) Predict future crime trends using regression analysis.

```
import pandas as pd
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
import matplotlib.pyplot as plt


# Grouping: Total crimes per year for the entire country (you can also group by State/District)
crime_trend = df.groupby('YEAR')['TOTAL IPC CRIMES'].sum().reset_index()

# Features and target
X = crime_trend[['YEAR']]
```

```
y = crime_trend['TOTAL IPC CRIMES']

# Build Linear Regression model
model = LinearRegression()
model.fit(X, y)

# Predict on known years (for validation)
y_pred = model.predict(X)

# Print model performance
print(f"R² Score: {r2_score(y, y_pred):.3f}")
print(f"Mean Squared Error: {mean_squared_error(y, y_pred):.2f}")

# Plot actual vs predicted
plt.figure(figsize=(8, 5))
plt.scatter(X, y, color='blue', label='Actual Crime Data')
plt.plot(X, y_pred, color='red', linewidth=2, label='Regression Line')
plt.title('Crime Trends Over Years')
plt.xlabel('Year')
plt.ylabel('Total IPC Crimes')
plt.legend()
plt.grid(True)
plt.show()

# Predict Future Years
future_years = pd.DataFrame({'YEAR': np.arange(2025, 2031)})  # Predict for 2025-2030
future_predictions = model.predict(future_years)

# Show predictions
future_years['Predicted Crimes'] = future_predictions.astype(int)
print("\n📌 Predicted Total IPC Crimes for Future Years:")
print(future_years)
```

```
R² Score: 0.896
Mean Squared Error: 47579969790.02
```



```
📌 Predicted Total IPC Crimes for Future Years:
   YEAR  Predicted Crimes
0  2025           6979458
1  2026           7137898
2  2027           7296338
3  2028           7454778
4  2029           7613218
5  2030           7771658
```

89.6% of the variance in the total crime data can be explained by the model. This is a strong result, indicating that linear regression model is doing a good job of fitting the data.

```
crime_trend['Predicted Crimes'] = y_pred
print(crime_trend[['YEAR', 'TOTAL IPC CRIMES', 'Predicted Crimes']].head(10))
```

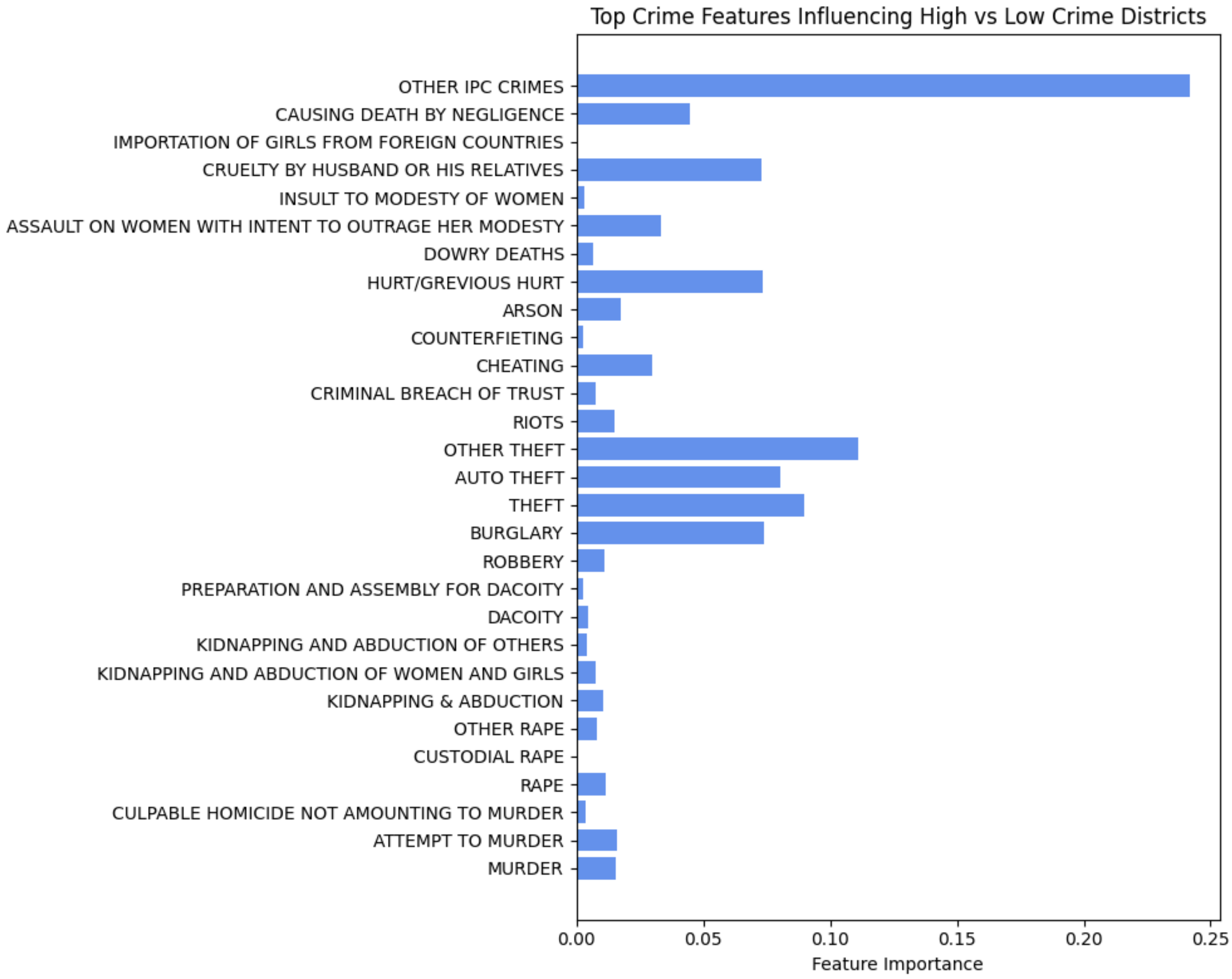
```
   YEAR  TOTAL IPC CRIMES  Predicted Crimes
0  2001           3538616      3.176900e+06
1  2002           3560660      3.335340e+06
2  2003           3432240      3.493780e+06
3  2004           3664020      3.652219e+06
4  2005           3645204      3.810659e+06
5  2006           3756586      3.969099e+06
6  2007           3979346      4.127539e+06
7  2008           4186758      4.285979e+06
8  2009           4242690      4.444419e+06
9  2010           4449662      4.602859e+06
```

the predicted values seems good.

Q) Use a machine learning model to classify high-crime and low-crime districts.

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix
import matplotlib.pyplot as plt

# Create a binary label for High (1) / Low (0) crime based on TOTAL IPC CRIMES
threshold = df['TOTAL IPC CRIMES'].median()
df['Crime_Level'] = (df['TOTAL IPC CRIMES'] >= threshold).astype(int)

# Select numeric features only (exclude state, district, year, target)
X = df.drop(columns=['STATE/UT', 'DISTRICT', 'YEAR', 'TOTAL IPC CRIMES', 'Crime_Level'])
X = X.select_dtypes(include=['int64', 'float64'])  # Make sure only numbers remain

y = df['Crime_Level']

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y
)

# Initialize Random Forest Classifier
model = RandomForestClassifier(n_estimators=100, random_state=42)

# Train the model
model.fit(X_train, y_train)

# Predict on test data
y_pred = model.predict(X_test)

# Evaluation
print("\n=== Confusion Matrix ===\n", confusion_matrix(y_test, y_pred))
print("\n=== Classification Report ===\n", classification_report(y_test, y_pred))

# Plot feature importance
importances = model.feature_importances_
features = X.columns

plt.figure(figsize=(10, 8))
plt.barh(features, importances, color='cornflowerblue')
plt.xlabel('Feature Importance')
plt.title('Top Crime Features Influencing High vs Low Crime Districts')
plt.tight_layout()
plt.show()
```

```
=== Confusion Matrix ===
 [[1037   30]
 [  11 1058]]

=== Classification Report ===
               precision    recall  f1-score   support

           0       0.99      0.97      0.98      1067
           1       0.97      0.99      0.98      1069

    accuracy                           0.98      2136
   macro avg       0.98      0.98      0.98      2136
weighted avg       0.98      0.98      0.98      2136
```



The macro avg and weighted avg both show values of 0.98 for precision, recall, and F1 score, meaning the model performs well across both high-crime and low-crime classes.

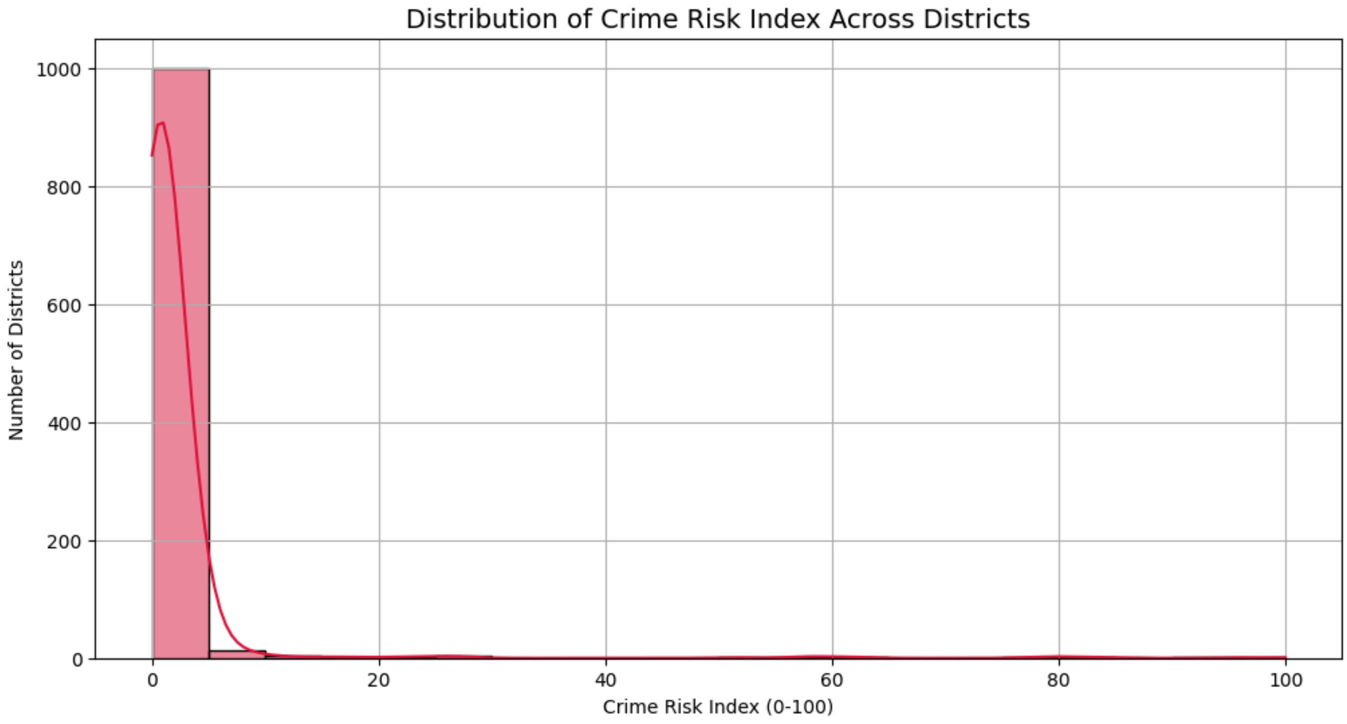
The confusion matrix likely indicates that the model is able to correctly distinguish between high-crime and low-crime districts with only a small number of misclassifications.

Q) Develop a crime risk index for districts based on historical data.

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

# Group data by district to compute total crime over all years
district_crime = df.groupby(['STATE/UT', 'DISTRICT'])['TOTAL IPC CRIMES'].sum().reset_index()

# Normalize to create a Risk Index (0-100 scale)
max_crime = district_crime['TOTAL IPC CRIMES'].max()
```

```
district_crime['Crime_Risk_Index'] = (district_crime['TOTAL IPC CRIMES'] / max_crime) * 100

# Sort: High risk to low risk
district_crime = district_crime.sort_values('Crime_Risk_Index', ascending=False)

# Show top 10 most at-risk districts
print("\n🔥 Top 10 High Crime Risk Districts:\n")
print(district_crime[['STATE/UT', 'DISTRICT', 'Crime_Risk_Index']].head(10))

# Plot Risk Distribution
plt.figure(figsize=(12,6))
sns.histplot(district_crime['Crime_Risk_Index'], bins=20, kde=True, color='crimson')
plt.title('Distribution of Crime Risk Index Across Districts', fontsize=14)
plt.xlabel('Crime Risk Index (0-100)')
plt.ylabel('Number of Districts')
plt.grid(True)
plt.show()
```

```
🔥 Top 10 High Crime Risk Districts:

           STATE/UT DISTRICT  Crime_Risk_Index
552  MADHYA PRADESH    TOTAL        100.000000
604     MAHARASHTRA    TOTAL         94.646192
851      TAMIL NADU    TOTAL         84.325790
37   ANDHRA PRADESH    TOTAL         80.709873
971   UTTAR PRADESH    TOTAL         79.796722
795       RAJASTHAN    TOTAL         77.653840
487          KERALA    TOTAL         62.484667
456       KARNATAKA    TOTAL         60.236899
158           BIHAR    TOTAL         58.049022
285         GUJARAT    TOTAL         57.474209
```



as known maharastra and madhya pradesh has the high crime rate and the risk index also says the same.

Bonus Questions

Q) What percentage of crimes are committed against women?

```
import pandas as pd

# List of columns related to crimes against women
crimes_against_women = [
    'RAPE',
    'CUSTODIAL RAPE',
    'OTHER RAPE',
    'KIDNAPPING AND ABDUCTION OF WOMEN AND GIRLS',
    'ASSAULT ON WOMEN WITH INTENT TO OUTRAGE HER MODESTY',
    'INSULT TO MODESTY OF WOMEN',
    'CRUELTY BY HUSBAND OR HIS RELATIVES',
    'DOWRY DEATHS',
    'IMPORTATION OF GIRLS FROM FOREIGN COUNTRIES'
]

# Sum of crimes against women
total_crimes_against_women = df[crimes_against_women].sum().sum()

# Sum of all crimes (total IPC crimes column)
total_crimes = df['TOTAL IPC CRIMES'].sum()

# Calculate percentage
percentage_crimes_against_women = (total_crimes_against_women / total_crimes) * 100

print(f"Percentage of crimes committed against women: {percentage_crimes_against_women:.2f}%")
```

```
Percentage of crimes committed against women: 9.97%
```

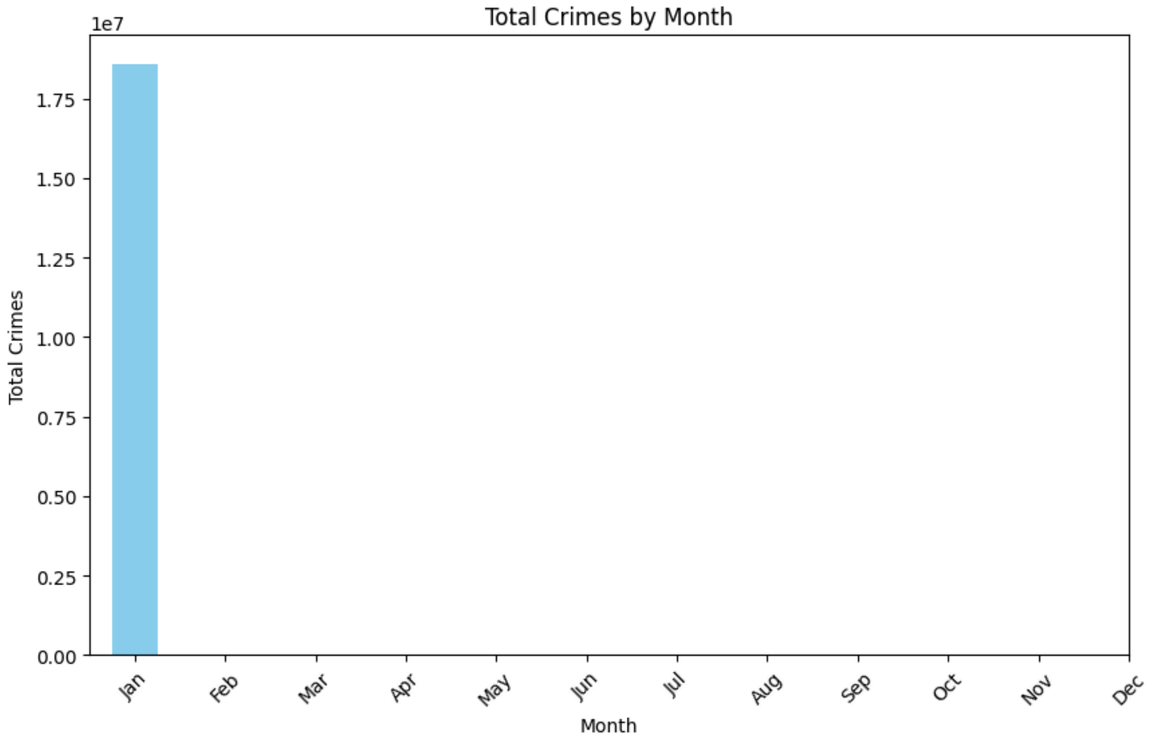
Q) Identify the state with the highest number of dowry deaths.

```
# Group by STATE/UT and sum the dowry deaths
state_dowry_deaths = df.groupby('STATE/UT')['DOWRY DEATHS'].sum()

# Sort the states by dowry deaths in descending order and get the state with the highest number
highest_dowry_deaths_state = state_dowry_deaths.idxmax()
highest_dowry_deaths_count = state_dowry_deaths.max()

print(f"State with the highest number of dowry deaths: {highest_dowry_deaths_state}")
print(f"Number of dowry deaths: {highest_dowry_deaths_count}")
```

```
State with the highest number of dowry deaths: UTTAR PRADESH
Number of dowry deaths: 57256
```

Q) Analyze seasonal variations in crime trends (e.g., do crimes increase during certain months?).

```
import pandas as pd
import matplotlib.pyplot as plt
# Convert the 'YEAR' column to datetime (If the column is 'YEAR' without specific date)
# This will create a dummy date, which we'll use to extract the month.

df['DATE'] = pd.to_datetime(df['YEAR'].astype(str) + '-01-01')  # If no actual date available, set to Jan 1st each year

# Extract Month and Year from the 'DATE' column
df['Month'] = df['DATE'].dt.month
df['Year'] = df['DATE'].dt.year

# Calculate total crimes (sum of specific crime types)
df['Total_Crimes'] = df[['MURDER', 'ATTEMPT TO MURDER', 'RAPE', 'KIDNAPPING & ABDUCTION',
                         'ROBBERY', 'BURGLARY', 'THEFT', 'RIOTS', 'ARSON', 'ASSAULT ON WOMEN WITH INTENT TO OUTRAGE HER MODESTY']].sum(axis

# Group by month and calculate total crimes in each month
monthly_crimes = df.groupby('Month')['Total_Crimes'].sum()

# Plot the results
plt.figure(figsize=(10, 6))
monthly_crimes.plot(kind='bar', color='skyblue')
plt.title('Total Crimes by Month')
plt.xlabel('Month')
plt.ylabel('Total Crimes')
plt.xticks(ticks=range(12), labels=['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec'], rotation=45)
plt.show()
```



The data which we have is yearly and hence we cant get seasonal data based on months.

Q) Examine if there is a link between cities and crime rates

The dataset donot contain Cities, we only have states and Dictrict

Q) Build a time-series model to forecast crime rates for the next five years.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from statsmodels.tsa.arima.model import ARIMA
from statsmodels.tsa.statespace.sarimax import SARIMAX
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
from statsmodels.tsa.stattools import adfuller
from sklearn.metrics import mean_squared_error
import warnings
warnings.filterwarnings('ignore')

# Check YEAR column type
print(f"\nYEAR column type: {df['YEAR'].dtype}")

# Convert YEAR to integer if it's a timestamp or datetime
if pd.api.types.is_datetime64_any_dtype(df['YEAR']):
    print("Converting YEAR from timestamp to integer...")
    df['YEAR'] = df['YEAR'].dt.year
elif not pd.api.types.is_integer_dtype(df['YEAR']):
    df['YEAR'] = df['YEAR'].astype(int)

# Step 1: Aggregate crimes at the national level by year
# Select only numeric columns for summing (excluding 'YEAR')
numeric_cols = [col for col in df.columns if np.issubdtype(df[col].dtype, np.number) and col != 'YEAR']
national_yearly = df.groupby('YEAR')[numeric_cols].sum().reset_index()

#print("\nNational yearly crime totals:")
#print(national_yearly.head())

# Identify the total crime column
# Look for column containing 'TOTAL' and 'CRIME' (case insensitive)
total_crime_col = None
for col in national_yearly.columns:
    if isinstance(col, str) and 'TOTAL' in col.upper() and 'CRIME' in col.upper():
        total_crime_col = col
        break

# If no specific total crime column found, use sum of all numeric columns except YEAR
if total_crime_col is None:
    print("No specific total crime column found. Creating one by summing all numeric columns except YEAR.")
    numeric_cols = [col for col in national_yearly.columns if col != 'YEAR' and np.issubdtype(national_yearly[col].dtype, np.number)]
    national_yearly['TOTAL CRIMES'] = national_yearly[numeric_cols].sum(axis=1)
    total_crime_col = 'TOTAL CRIMES'

print(f"Using '{total_crime_col}' as the total crime column")

# Step 2: Plot the time series of total crimes
plt.figure(figsize=(12, 6))
plt.plot(national_yearly['YEAR'], national_yearly[total_crime_col], marker='o')
plt.title(f'{total_crime_col} in India (Time Series)')
plt.xlabel('Year')
plt.ylabel('Number of Crimes')
plt.grid(True)
plt.xticks(national_yearly['YEAR'][::2])  # Show every second year on x-axis
plt.tight_layout()
plt.savefig('total_crime_trend.png')

# Step 3: We need population data to calculate crime rates
# Since we don't have actual population data in the DataFrame, let's use Census of India estimates
# Create population estimates for each year represented in the data
years = national_yearly['YEAR'].unique()
min_year = int(min(years))
max_year = int(max(years))

# Approximate population base and growth rate for India
population_base = 1_028_000_000  # ~2001 population
growth_rate = 0.0145  # Annual growth rate ~1.45%

# Create population estimates for each year
population_data = []
for year in range(min_year, max_year + 1):
    pop = int(population_base * (1 + growth_rate) ** (year - min_year))
    population_data.append({'YEAR': year, 'POPULATION': pop})

population_df = pd.DataFrame(population_data)
national_yearly = pd.merge(national_yearly, population_df, on='YEAR')

# Calculate crime rate per 100,000 people
national_yearly['CRIME_RATE'] = (national_yearly[total_crime_col] / national_yearly['POPULATION']) * 100000

# Plot crime rate
plt.figure(figsize=(12, 6))
plt.plot(national_yearly['YEAR'], national_yearly['CRIME_RATE'], marker='o', color='red')
plt.title('Crime Rate per 100,000 Population in India')
plt.xlabel('Year')
plt.ylabel('Crime Rate per 100,000 people')
plt.grid(True)
plt.xticks(national_yearly['YEAR'][::2])
plt.tight_layout()
plt.savefig('crime_rate_trend.png')

# Step 4: Check stationarity
def test_stationarity(timeseries):
    result = adfuller(timeseries.values)
    print('ADF Statistic: %f' % result[0])
    print('p-value: %f' % result[1])
    print('Critical Values:')
    for key, value in result[4].items():
        print('\t%s: %.3f' % (key, value))

    # Interpret results
    if result[1] <= 0.05:
        print("Data is stationary (reject null hypothesis)")
    else:
        print("Data is not stationary (fail to reject null hypothesis)")

    return result[1] <= 0.05

print("\nStationarity test for crime rate:")
is_stationary = test_stationarity(national_yearly['CRIME_RATE'])
```