### **Solution Plan: Data Cleaning, Preprocessing, and Visualization**

### **1. Data Cleaning Steps**

**a. Handle Missing Values and Zeros**

* Checked the dataset for missing values (NaN) and zeros in numerical columns.

Replaced zeros and missing values with the mean of the respective column, calculated excluding zeros:  
python  
CopyEdit  
mean\_val = numerical\_df[col][numerical\_df[col] != 0].mean()

df[col] = df[col].replace(0, mean\_val)

df[col] = df[col].fillna(mean\_val)

**b. Removed Outliers**

Used the **Interquartile Range (IQR)** method to identify and remove extreme outliers from numerical columns:  
python  
CopyEdit  
Q1 = df[col].quantile(0.25)

Q3 = df[col].quantile(0.75)

IQR = Q3 - Q1

df = df[~((df[col] < (Q1 - 1.5 \* IQR)) | (df[col] > (Q3 + 1.5 \* IQR)))]

**c. Data Retention Verification**

Ensured at least 80% of the original entries were retained:  
python  
CopyEdit  
retained\_percentage = (len(df) / original\_row\_count) \* 100

assert retained\_percentage >= 80, "Data retention below required threshold."

**d. Saved Cleaned Data**

Saved the cleaned dataset to CLEANED\_FINAL\_DATASET.csv:  
python  
CopyEdit  
df.to\_csv('CLEANED\_FINAL\_DATASET.csv', index=False)

### **2. Preprocessing for XGBoost**

**a. Feature Selection**

* Selected relevant columns for features (X) and target (y).

**b. Data Splitting**

Split the dataset into training and testing sets:  
python  
CopyEdit  
from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

**c. Encoding Categorical Variables**

Encoded categorical columns using **one-hot encoding** if applicable:  
python  
CopyEdit  
X = pd.get\_dummies(X, drop\_first=True)

**d. Normalization**

Normalized numerical features using Min-Max scaling:  
python  
CopyEdit  
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()

X\_train = scaler.fit\_transform(X\_train)

X\_test = scaler.transform(X\_test)

### 

### 

### **3. XGBoost Model Implementation**

**a. Model Training**

Used XGBoost to train on the cleaned and preprocessed data:  
python  
CopyEdit  
from xgboost import XGBClassifier

model = XGBClassifier()

model.fit(X\_train, y\_train)

**b. Predictions**

Made predictions on the test data:  
python  
CopyEdit  
y\_pred = model.predict(X\_test)

**c. Accuracy Evaluation**

Evaluated model accuracy using a confusion matrix and accuracy score:  
python  
CopyEdit  
from sklearn.metrics import accuracy\_score, confusion\_matrix

acc = accuracy\_score(y\_test, y\_pred)

print(f"Accuracy: {acc}")

print(confusion\_matrix(y\_test, y\_pred))

### **4. Visualizations**

**a. Crime Data Analysis**

Created bar plots for specific crimes grouped by states/UTs:  
python  
CopyEdit  
crime\_types = ['Rape other than Custodial', 'Rape\_Gang Rape']

for crime in crime\_types:

df.groupby('States/UTs')[crime].sum().sort\_values(ascending=False).plot(kind='bar', figsize=(12, 6))

**b. Crime Distribution**

Visualized crime distribution across states/UTs using a pie chart:  
python  
CopyEdit  
crime\_dist = df.groupby('States/UTs')['Rape other than Custodial'].sum()

* crime\_dist.plot(kind='pie', autopct='%1.1f%%', figsize=(10, 8), startangle=140)