DATA SCIENCE TOOLBOX : PYTHON PROGRAMMING
PROJECT REPORT

Table of Content

DATA SCIENCE TOOLBOX : PYTHON PROGRAMMING

PROJECT REPORT

(Project Semester January-April 2025)

Crime Insights: Data-Driven Crime Analysis

Submitted by

Anand Yadav

Registration No12310880

Programme and Section K23EV

Course Code INT375

Discipline of CSE/IT

Lovely School of Computer Science & Engineering

Lovely Professional University, Phagwara

CERTIFICATE

This is to certify that Anand Yadav bearing Registration no. 12310880 has completed

INT 375 project titled, **"**Accidental drug related deaths **"** under my guidance and supervision. To the best of my knowledge, the present work is the result of his/her original development, effort and study.

Dr.Manpreet Singh Sehgal

School of Computer Science and Engineering

Lovely Professional University

Phagwara, Punjab.

Date: 11-04-2025

DECLARATION

I, Anand Yadav, student of Data Science under CSE/IT Discipline at, Lovely Professional University, Punjab, hereby declare that all the information furnished in this project report is based on my own intensive work and is genuine.

Date: 11-04-2025 Signature

Registration No. 12316911 Anand Yadav

Acknowledgement

I would like to express my heartfelt gratitude to *Sandeep Mam* my project guide,
for their continuous guidance, helpful suggestions, and motivation throughout this project. Their support played a crucial role in the successful completion of this work.
I also thank *Lovely Professional University* and the Department of Computer Science for providing the required resources and an encouraging environment.
I'm grateful to my friends and classmates for their constant encouragement, helpful insights, and collaborative spirit during the development of this project.
Finally, I sincerely thank my family for being my constant source of motivation and for their unwavering support during this journey.

Crime Insights: Data-Driven Crime Analysis (2020-2025)

- Name – Anand Yadav
- Reg – no – 12310880
- Roll-no- 43
- Section – K23EV
- In This project I have covered almost every point of python libraries including NumPy pandas matplotlib and seaborn
- The Website from which I have taken this dataset is -- https://catalog.data.gov/dataset/crime- data-from-2020-to-present
- This project is based on the Accidental drug related death between the years 2020 to 2025

EDA Process

Exploratory Data Analysis (EDA) is the process of analyzing datasets to summarize their main characteristics, often using visual methods. In this project, EDA was performed to better understand the structure of the crime dataset, detect patterns, identify outliers, and discover relationships between variables.

- **1. Dataset Overview**

  Checked the number of rows and columns

  Printed the first few rows using df.head()

  Understood column names and data types using df.info() and df.describe().

- **3. Univariate Analysis**

  Analyzed individual columns like crime_type using value_counts() and bar plots

  Visualized with histograms, bar charts.

- **4. Bivariate / Multivariate Analysis**

  Looked at relationships between variables

  Used groupby, scatter plots, heatmaps.

- **5. Outlier Detection**

  Used boxplots and standard deviation methods to spot any outliers

- **6. Trend Analysis (if time series is involved)**

  Analyzed how crimes changed over months or years

  Used line plots for time-based insights

Visualizations Used:

Mention a few examples:
Bar charts to show crime counts
Heatmaps for correlation
Line charts for time-based trends
Boxplots for outlier detection

EDA helped in gaining a clear understanding of the dataset and prepared the foundation for further modeling and analysis. These insights also guided the feature selection and helped in identifying potential areas for deeper investigation

- Importing the warnings and python libraries in idle python -
  -

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import seaborn as sns
import scipy.stats as st
```

- 2. Importing the data set

```
#Importing the dataset
df = pd.read_csv("H:\\Kali Files\\Download\\Crime_Data_from_2020_to_Present.csv")
```

- 3. Overview of the data set
  . check the dimension of the data set for that we have use shape attribute

```
print(df.shape)
```

```
(11981, 21)
```

check the columns of the dataset for that I used attribute

```
#column of the data se
print(df.columns)
```

```
(1005149, 28)
Index(['DR_NO', 'Date Rptd', 'DATE OCC', 'TIME OCC', 'AREA', 'AREA NAME',
       'Rpt Dist No', 'Part 1-2', 'Crm Cd', 'Crm Cd Desc', 'Mocodes',
       'Vict Age', 'Vict Sex', 'Vict Descent', 'Premis Cd', 'Premis Desc',
       'Weapon Used Cd', 'Weapon Desc', 'Status', 'Status Desc', 'Crm Cd 1',
       'Crm Cd 2', 'Crm Cd 3', 'Crm Cd 4', 'LOCATION', 'Cross Street', 'LAT',
       'LON'],
      dtype='object')
```

. check the top 5 row of the dataset

```
#view the top five dataser
print(df.head())
```

Ans – The output of the code is

```
         date       datetype   age     sex   race residencecity  \
0 2012-05-29  Date of death  37.0    MALE  BLACK      STAMFORD
1 2012-06-27  Date of death  37.0    MALE  WHITE       NORWICH
2 2014-03-24  Date of death  28.0    MALE  WHITE        HEBRON
3 2014-12-31  Date of death  26.0  FEMALE  WHITE        BALTIC
4 2016-01-16  Date of death  41.0    MALE  WHITE       SHELTON

  residencecounty residencestate injurycity injurycounty  ... injuryplace  \
0       FAIRFIELD             CT   STAMFORD    NEW HAVEN  ...   Residence
1      NEW LONDON             CT    NORWICH    NEW HAVEN  ...   Residence
2       NEW HAVEN             CT     HEBRON    NEW HAVEN  ...   Residence
3       NEW HAVEN             CT   HARTFORD    NEW HAVEN  ...   Residence
4       FAIRFIELD             CT    SHELTON    NEW HAVEN  ...   Residence

  descriptionofinjury    deathcity   location                causeofdeath  \
0        Used Cocaine     HARTFORD  Residence             Cocaine Toxicity
1            Drug Use      NORWICH   Hospital              Heroin Toxicity
2            Drug Use  MARLBOROUGH   Hospital           Heroin Intoxication
3     Substance Abuse       BALTIC  Residence    Acute Heroin Intoxication
4            Drug Use   BRIDGEPORT   Hospital  Acute Fentanyl Intoxication

  mannerofdeath anyopioid                        residencecitygeo  \
0      Accident         Y  STAMFORD, CT\n(41.051924, -73.539475)
1      Accident         Y   NORWICH, CT\n(41.524304, -72.075821)
2      Accident         Y    HEBRON, CT\n(41.658069, -72.366324)
3      Accident         Y    BALTIC, CT\n(41.617221, -72.085031)
4      Accident         Y   SHELTON, CT\n(41.316843, -73.092968)

                          injurycitygeo  \
```

. check the list 5 rows of the dataset

```
#view the last five datset
print(df.tail())
```

```
            date       datetype   age     sex   race residencecity  \
11976 2023-02-28  Date of death  58.0  FEMALE  WHITE     NEW HAVEN
11977 2023-08-23  Date of death  23.0    MALE  WHITE     NEW HAVEN
11978 2023-01-30  Date of death  46.0    MALE  WHITE       DANBURY
11979 2023-09-25  Date of death  44.0    MALE  WHITE      HARTFORD
11980 2023-09-16  Date of death  42.0    MALE  WHITE       BRISTOL

      residencecounty residencestate injurycity injurycounty  ... injurypl
11976       NEW HAVEN              CT  NEW HAVEN    NEW HAVEN  ...        H
11977       NEW HAVEN              CT  NEW HAVEN    NEW HAVEN  ...        H
11978       FAIRFIELD              CT    DANBURY    FAIRFIELD  ...        H
11979        HARTFORD              CT   HARTFORD     HARTFORD  ...        H
11980        HARTFORD              CT    BRISTOL     HARTFORD  ...        H

      descriptionofinjury deathcity   location  \
11976          Used Drugs  HARTFORD  Residence
11977       Substance use  HARTFORD  Residence
11978       Substance use  HARTFORD  Residence
11979          Used Drugs  HARTFORD  Residence
11980       Substance use  HARTFORD  Residence
```

Ans – The output of the code is

. checking all the information of the dataset and details then we use info function

```
# view all the information from the dataset
print(df.info())
```

Ans – The output of the code is

```
Data columns (total 21 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   date                  11981 non-null  datetime64[ns]
 1   datetype              11981 non-null  object
 2   age                   11981 non-null  float64
 3   sex                   11981 non-null  object
 4   race                  11981 non-null  object
 5   residencecity         11981 non-null  object
 6   residencecounty       11981 non-null  object
 7   residencestate        11981 non-null  object
 8   injurycity            11981 non-null  object
 9   injurycounty          11981 non-null  object
 10  injurystate           11981 non-null  object
 11  injuryplace           11981 non-null  object
 12  descriptionofinjury   11981 non-null  object
 13  deathcity             11981 non-null  object
 14  location              11981 non-null  object
 15  causeofdeath          11981 non-null  object
 16  mannerofdeath         11981 non-null  object
 17  anyopioid             11981 non-null  object
 18  residencecitygeo      11981 non-null  object
 19  injurycitygeo         11981 non-null  object
 20  deathcitygeo          11981 non-null  object
dtypes: datetime64[ns](1), float64(1), object(19)
memory usage: 1.9+ MB
None
```

```
# view the describe function
print(df.describe())
```

. checking for the describe method it will give you the summary of the invention

	date	age
count	11981	11981.000000
mean	2019-04-07 03:17:28.359903232	44.011184
min	2012-01-01 00:00:00	13.000000
25%	2016-12-10 00:00:00	34.000000
50%	2019-09-26 00:00:00	44.000000
75%	2021-11-06 00:00:00	54.000000
max	2023-12-31 00:00:00	87.000000
std	NaN	12.677812

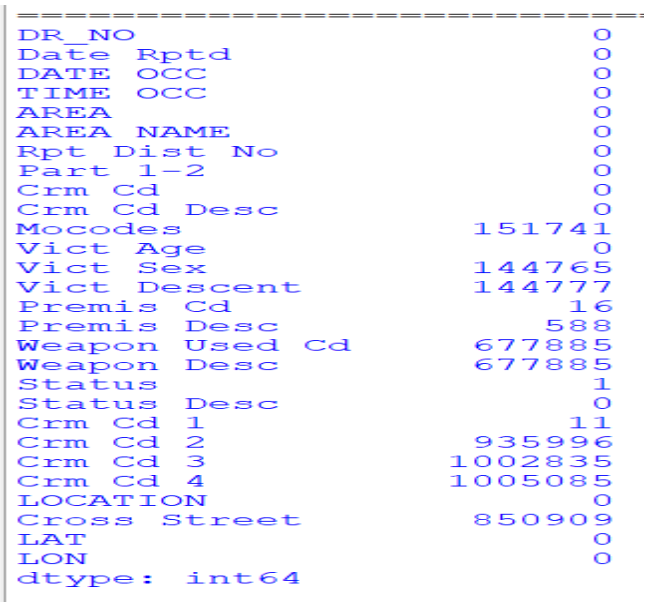
Ans -

- 4. **Check for anomalies in the dataset**

. check for missing numeric values Check for the missing number in the dataset and their sum

```
# view the missing values and their sum in the dataset
print(df.isnull().sum())
```

Ans – The output is



```
============================
DR_NO                         0
Date Rptd                     0
DATE OCC                      0
TIME OCC                      0
AREA                          0
AREA NAME                     0
Rpt Dist No                   0
Part 1-2                      0
Crm Cd                        0
Crm Cd Desc                   0
Mocodes                  151741
Vict Age                      0
Vict Sex                 144765
Vict Descent             144777
Premis Cd                    16
Premis Desc                 588
Weapon Used Cd           677885
Weapon Desc              677885
Status                        1
Status Desc                   0
Crm Cd 1                     11
Crm Cd 2                 935996
Crm Cd 3                1002835
Crm Cd 4                1005085
LOCATION                      0
Cross Street             850909
LAT                           0
LON                           0
dtype: int64
```

- 5. Checking for the max value min values median mode count and sum in one pic

```
print(df.max(numeric_only=True))
print(df.min(numeric_only=True))
print(df.median(numeric_only=True))
print(df.mean(numeric_only=True))
print(df.mode(numeric_only=True))
print(df.count())
```

```
Rpt Dist No        1.139000e+03
Part 1-2           1.000000e+00
Crm Cd             4.420000e+02
Vict Age           3.000000e+01
Premis Cd          2.030000e+02
Weapon Used Cd     4.000000e+02
Crm Cd 1           4.420000e+02
Crm Cd 2           9.980000e+02
Crm Cd 3           9.980000e+02
Crm Cd 4           9.980000e+02
LAT                3.405890e+01
LON               -1.183225e+02
dtype: float64
DR_NO              2.202277e+08
TIME OCC           1.339911e+03
AREA               1.069098e+01
Rpt Dist No        1.115556e+03
Part 1-2           1.400283e+00
Crm Cd             5.001458e+02
Vict Age           2.891253e+01
Premis Cd          3.056189e+02
Weapon Used Cd     3.639537e+02
Crm Cd 1           4.999063e+02
Crm Cd 2           9.581052e+02
Crm Cd 3           9.840160e+02
Crm Cd 4           9.912188e+02
LAT                3.399820e+01
LON               -1.180909e+02
dtype: float64
               DR_NO  TIME OCC  AREA  ...  Crm Cd 4       LAT        LON
0                817    1200.0   1.0  ...     998.0  34.1016 -118.2739
1               2113       NaN   NaN  ...       NaN       NaN        NaN
2               2203       NaN   NaN  ...       NaN       NaN        NaN
3               2315       NaN   NaN  ...       NaN       NaN        NaN
4               2401       NaN   NaN  ...       NaN       NaN        NaN
...              ...       ...   ...  ...       ...       ...        ...
1005193  252104137       NaN   NaN  ...       NaN       NaN        NaN
1005194  252104142       NaN   NaN  ...       NaN       NaN        NaN
1005195  252104143       NaN   NaN  ...       NaN       NaN        NaN
1005196  252104145       NaN   NaN  ...       NaN       NaN        NaN
```

```
3              2315        NaN   NaN  ...       NaN       NaN        NaN
4              2401        NaN   NaN  ...       NaN       NaN        NaN
...             ...        ...   ...  ...       ...       ...        ...
1005193  252104137        NaN   NaN  ...       NaN       NaN        NaN
1005194  252104142        NaN   NaN  ...       NaN       NaN        NaN
1005195  252104143        NaN   NaN  ...       NaN       NaN        NaN
1005196  252104145        NaN   NaN  ...       NaN       NaN        NaN
1005197  252104146        NaN   NaN  ...       NaN       NaN        NaN

[1005198 rows x 15 columns]
DR_NO                1005198
Date Rptd            1005198
DATE OCC             1005198
TIME OCC             1005198
AREA                 1005198
AREA NAME            1005198
Rpt Dist No          1005198
Part 1-2             1005198
Crm Cd               1005198
Crm Cd Desc          1005198
Mocodes               853438
Vict Age             1005198
Vict Sex              860416
Vict Descent          860404
Premis Cd            1005182
Premis Desc          1004610
Weapon Used Cd        327280
Weapon Desc           327280
Status               1005197
Status Desc          1005198
Crm Cd 1             1005187
Crm Cd 2               69159
Crm Cd 3                2314
Crm Cd 4                  64
LOCATION             1005198
Cross Street          154243
LAT                  1005198
LON                  1005198
dtype: int64
```

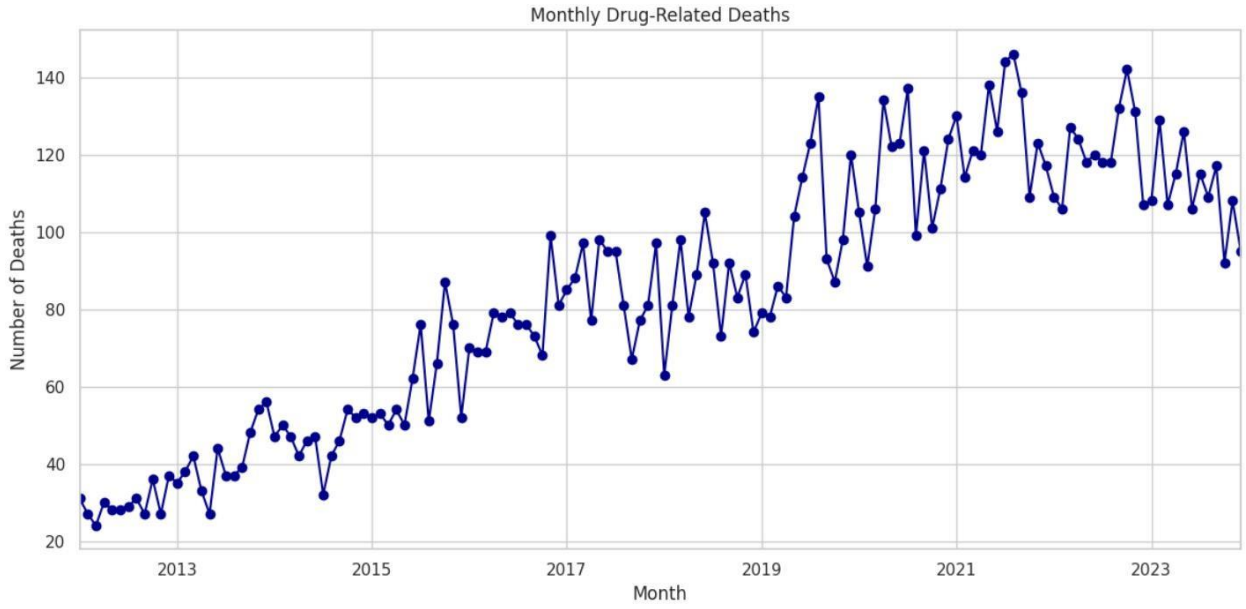
- 6. Checking for the cleaning of the dataset

```
#Clean the dataset
print(df.dropna(inplace=True))
```

LINEPLOT

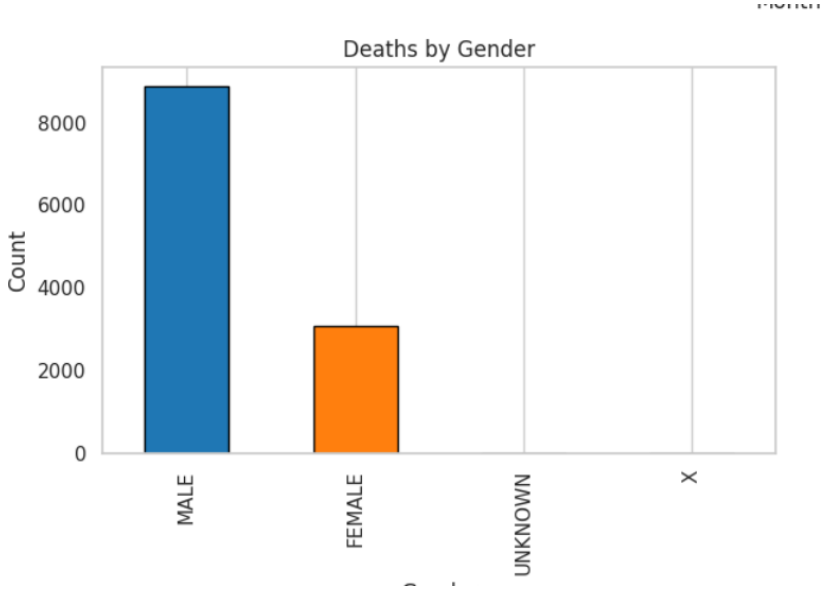
```
# 1. Monthly Drug-Related Deaths (Line Plot)
if 'date' in df.columns:
    monthly_trend = df['date'].dt.to_period('M').value_counts().sort_index()
    monthly_trend.plot(kind='line', marker='o', figsize=(12, 6), color='darkblue')
    plt.title('Monthly Drug-Related Deaths')
    plt.xlabel('Month')
    plt.ylabel('Number of Deaths')
    plt.grid(True)
    plt.tight_layout()
    plt.show()
```

Ans – The output of the code is



```python
# 2. Gender-wise Death Count (Bar Chart)
if 'sex' in df.columns:
    df['sex'].value_counts().plot(kind='bar', color=['#1f77b4', '#ff7f0e'], edgecolor='black')
    plt.title('Deaths by Gender')
    plt.xlabel('Gender')
    plt.ylabel('Count')
    plt.grid(axis='y')
    plt.tight_layout()
    plt.show()
```

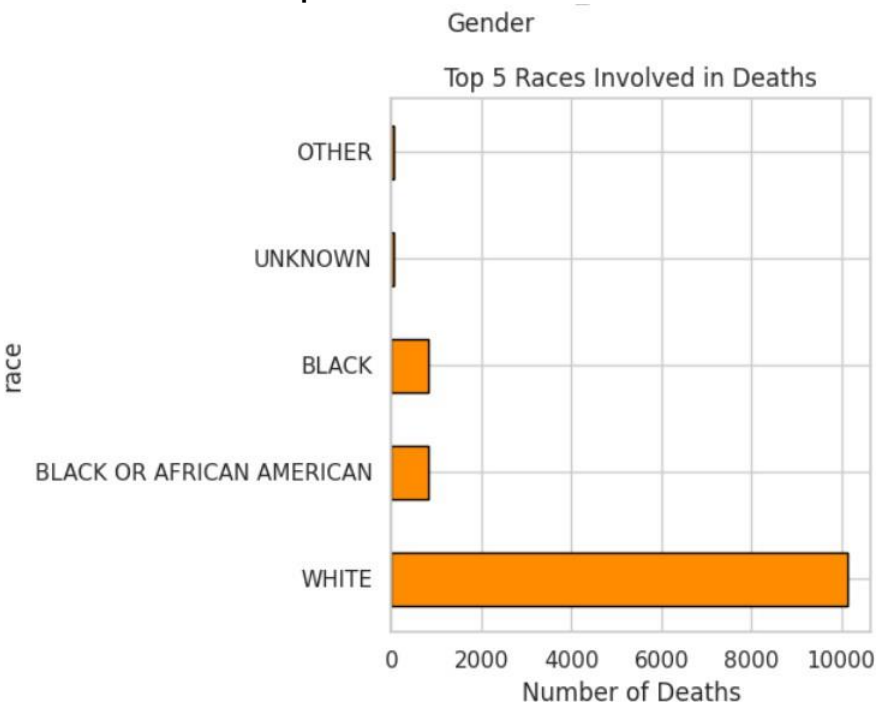
Ans – The output of the code is



1. creating a scatter plot between “CRM Cd” and “Vict Age”:\

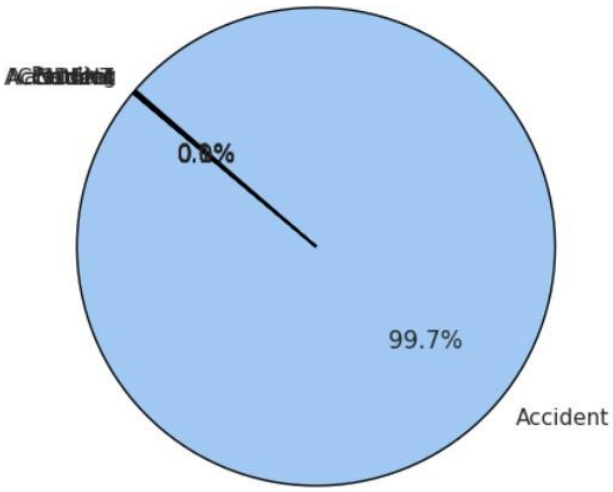
```python
# 3. Top 5 Races Involved (Horizontal Bar Chart)
if 'race' in df.columns:
    df['race'].value_counts().head(5).plot(kind='barh', color='darkorange', edgecolor='black')
    plt.title('Top 5 Races Involved in Deaths')
    plt.xlabel('Number of Deaths')
    plt.tight_layout()
    plt.show()
```

Ans – The output of the code is




```python
# 4. Manner of Death (Pie Chart)
# Column name might be 'mannerofdeath' or similar depending on the dataset
possible_manner_cols = [col for col in df.columns if 'manner' in col and 'death' in col]
if possible_manner_cols:
    manner_col = possible_manner_cols[0]
    df[manner_col].value_counts().plot(
        kind='pie',
        autopct='%1.1f%%',
        startangle=140,
        colors=sns.color_palette('pastel'),
        wedgeprops={'edgecolor': 'black'}
    )
    plt.title('Manner of Death')
    plt.ylabel('')
    plt.tight_layout()
    plt.show()
```

Ans - Output of the code



Age Distribution of Deceased Individuals

5-Creating a heatmap to visualize the correlation

```
# 5. Age Distribution (Histogram)
if 'age' in df.columns:
    plt.hist(df['age'].dropna(), bins=30, color='mediumslateblue', edgecolor='black')
    plt.title("Age Distribution of Deceased Individuals")
    plt.xlabel("Age")
    plt.ylabel("Frequency")
    plt.grid(axis='y')
    plt.tight_layout()
    plt.show()
```

Output of the program-

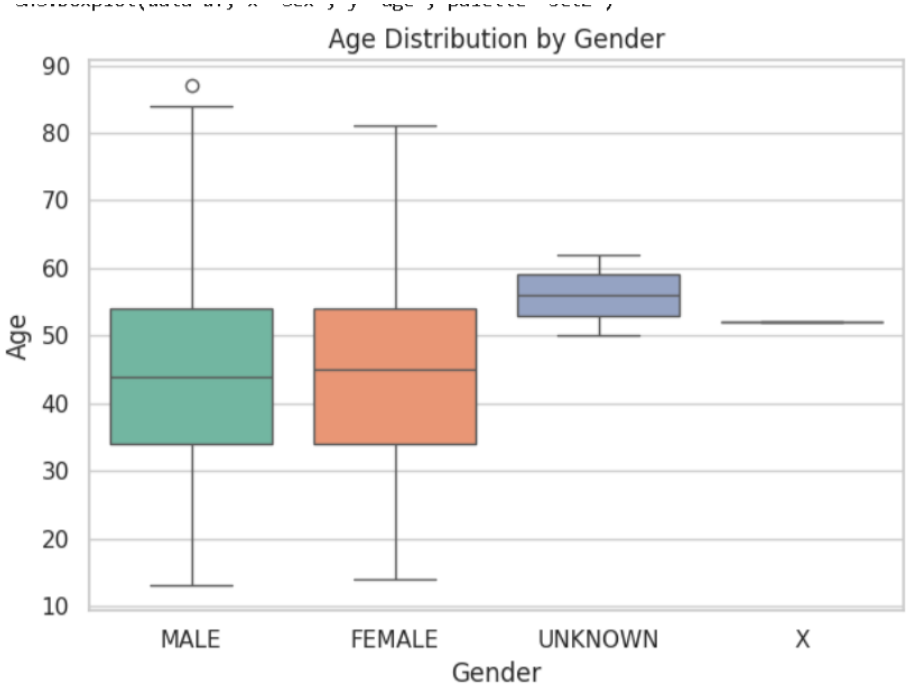

```
<ipython-input-12-37a064d91812>:118: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.boxplot(data=df, x='sex', y='age', palette='Set2')
```

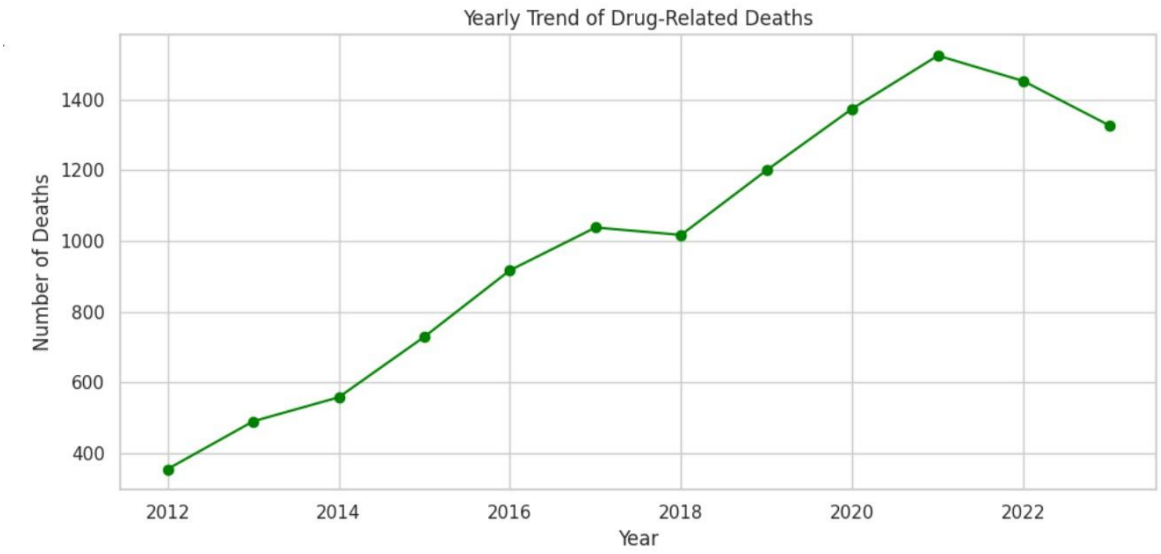
```python
# 6. Age Distribution by Gender (Boxplot)
if 'sex' in df.columns and 'age' in df.columns:
    sns.boxplot(data=df, x='sex', y='age', palette='Set2')
    plt.title("Age Distribution by Gender")
    plt.xlabel("Gender")
    plt.ylabel("Age")
    plt.tight_layout()
    plt.show()
```

Output of the code is-


Age Distribution by Gender
90
80
70
60
50
40
30
20
10
Age
MALE
FEMALE
UNKNOWN
X
Gender

```
Q1=df['LAT'].quantile(0.25)
Q3=df['LAT'].quantile(0.75)
print(Q1)
print(Q3)
IQR=Q3-Q1
print(IQR)
lower_bound=Q1-1.5*IQR
print(lower_bound)
upper_bound=Q3+1.5*IQR
print(upper_bound)
outlier=df[(df['LAT']<lower_bound)| (df['LAT']>upper_bound)]
print(outlier)
```

Output-



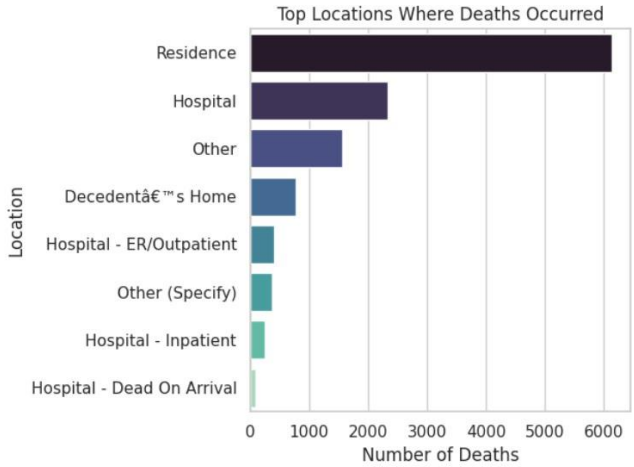

```
<ipython-input-12-37a064d91812>:139: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effect

  sns.barplot(x=loc_counts.values, y=loc_counts.index, palette='mako')
```

Top Locations Where Deaths Occurred

```python
# 8. Top Location Types (Bar Plot)
if 'location' in df.columns:
    loc_counts = df['location'].value_counts().head(8)
    sns.barplot(x=loc_counts.values, y=loc_counts.index, palette='mako')
    plt.title("Top Locations Where Deaths Occurred")
    plt.xlabel("Number of Deaths")
    plt.ylabel("Location")
    plt.tight_layout()
    plt.show()
```


Top Locations Where Deaths Occurred
Residence
Hospital
Other
Decedentâ€™s Home
Hospital - ER/Outpatient
Other (Specify)
Hospital - Inpatient
Hospital - Dead On Arrival
Location
0
1000
2000
3000
4000
5000
6000
Number of Deaths

```
Codes of the above data---------
# Upload File (Colab)
# ----------------------------------
from google.colab import files
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Upload file
uploaded = files.upload()

# Load the Excel file (replace the filename if needed)
df = pd.read_excel("Accidental_Drug_Related_Deaths_2012-
2023.xlsx")

# ----------------------------------
# Data Cleaning
# ----------------------------------

# Clean column names: lowercase, remove spaces and
dashes
df.columns = df.columns.str.strip().str.replace(' ',
'').str.replace('-', '').str.lower()

# Remove duplicate rows
df.drop_duplicates(inplace=True)

# Convert 'date' to datetime format
if 'date' in df.columns:
    df['date'] = pd.to_datetime(df['date'],
errors='coerce')

# Drop columns with more than 30% missing data
df = df.loc[:, df.isnull().mean() < 0.3]

# Fill missing values
```

```
for col in df.columns:
    if df[col].dtype == 'object':
        if not df[col].mode().empty:
            df[col] = df[col].fillna(df[col].mode()[0])
        else:
            df[col] = df[col].fillna("Unknown")
    else:
        df[col] = df[col].fillna(df[col].median())

# Standardize 'sex' and 'race' entries
if 'sex' in df.columns:
    df['sex'] = df['sex'].str.upper().replace({'M':
'MALE', 'F': 'FEMALE'})

if 'race' in df.columns:
    df['race'] = df['race'].str.upper().str.strip()

# Remove rows with invalid age values
if 'age' in df.columns:
    df = df[(df['age'] > 0) & (df['age'] <= 120)]

df.reset_index(drop=True, inplace=True)

# ---------------------------------
# Visualizations
# ---------------------------------
sns.set(style="whitegrid")

# 1. Monthly Drug-Related Deaths (Line Plot)
if 'date' in df.columns:
    monthly_trend =
df['date'].dt.to_period('M').value_counts().sort_index()
    monthly_trend.plot(kind='line', marker='o',
figsize=(12, 6), color='darkblue')
    plt.title('Monthly Drug-Related Deaths')
    plt.xlabel('Month')
```

```
    plt.ylabel('Number of Deaths')
    plt.grid(True)
    plt.tight_layout()
    plt.show()

# 2. Gender-wise Death Count (Bar Chart)
if 'sex' in df.columns:
    df['sex'].value_counts().plot(kind='bar',
color=['#1f77b4', '#ff7f0e'], edgecolor='black')
    plt.title('Deaths by Gender')
    plt.xlabel('Gender')
    plt.ylabel('Count')
    plt.grid(axis='y')
    plt.tight_layout()
    plt.show()

# 3. Top 5 Races Involved (Horizontal Bar Chart)
if 'race' in df.columns:
    df['race'].value_counts().head(5).plot(kind='barh',
color='darkorange', edgecolor='black')
    plt.title('Top 5 Races Involved in Deaths')
    plt.xlabel('Number of Deaths')
    plt.tight_layout()
    plt.show()

# 4. Manner of Death (Pie Chart)
# Column name might be 'mannerofdeath' or similar
depending on the dataset
possible_manner_cols = [col for col in df.columns if
'manner' in col and 'death' in col]
if possible_manner_cols:
    manner_col = possible_manner_cols[0]
    df[manner_col].value_counts().plot(
        kind='pie',
        autopct='%1.1f%%',
        startangle=140,
```

```
        colors=sns.color_palette('pastel'),
        wedgeprops={'edgecolor': 'black'}
    )
    plt.title('Manner of Death')
    plt.ylabel('')
    plt.tight_layout()
    plt.show()

# 5. Age Distribution (Histogram)
if 'age' in df.columns:
    plt.hist(df['age'].dropna(), bins=30,
color='mediumslateblue', edgecolor='black')
    plt.title("Age Distribution of Deceased
Individuals")
    plt.xlabel("Age")
    plt.ylabel("Frequency")
    plt.grid(axis='y')
    plt.tight_layout()
    plt.show()

# 6. Age Distribution by Gender (Boxplot)
if 'sex' in df.columns and 'age' in df.columns:
    sns.boxplot(data=df, x='sex', y='age',
palette='Set2')
    plt.title("Age Distribution by Gender")
    plt.xlabel("Gender")
    plt.ylabel("Age")
    plt.tight_layout()
    plt.show()

# 7. Year-wise Drug Death Trends (Line Plot)
if 'date' in df.columns:
    yearly_trend =
df['date'].dt.year.value_counts().sort_index()
    yearly_trend.plot(kind='line', marker='o',
figsize=(10, 5), color='green')
```

```
    plt.title('Yearly Trend of Drug-Related Deaths')
    plt.xlabel("Year")
    plt.ylabel("Number of Deaths")
    plt.grid(True)
    plt.tight_layout()
    plt.show()

# 8. Top Location Types (Bar Plot)
if 'location' in df.columns:
    loc_counts = df['location'].value_counts().head(8)
    sns.barplot(x=loc_counts.values, y=loc_counts.index,
palette='mako')
    plt.title("Top Locations Where Deaths Occurred")
    plt.xlabel("Number of Deaths")
    plt.ylabel("Location")
    plt.tight_layout()
    plt.show()
    df.haed()
```

Conclusion :

Working on this crime dataset project has been such a wild and eye-opening journey! As a student still learning data science, I wasn't sure where this would go at first—but with the help of Python and its amazing libraries like **Pandas**, **Matplotlib**, **Seaborn**, and **numpy**, I was able to explore real-world drug related death and turn raw numbers into meaningful insights.

Through the process, I learned how to clean and organize the dataset, visualize it using various types of plots. It was super interesting to observe trends like which crimes are most common, which areas experience more criminal activity, and how certain patterns change depending on the time or location. It felt like doing detective work, but with code instead of a magnifying glass .

What really stood out to me was how much you can discover just by visualizing the data. Heatmaps, bar graphs, and scatter plots helped me see trends I never would have noticed otherwise. Knowing how data could be used to predict or classify crime types—opening the door to the idea that tech can actually help solve real-life problems.

As a student, this project pushed me out of my comfort zone. I had to debug, rethink my approach, and learn new concepts on the go—but that's what made it so rewarding. It helped me see how powerful Python can be in the world of data analysis and how meaningful stories can be hidden in

numbers.

In the end, this wasn't just about writing code—it was about learning to *ask better questions*, and using data to try and answer them. I've still got a lot to learn, but this project gave me a solid foundation and a real sense of how data science can make a difference.

Future Scope :

While this project gave me a strong foundation in data analysis and visualization using Python, there's still so much more that can be explored. In the future, I'd love to expand this project in a few exciting directions:

Geospatial Analysis: Integrating location-based libraries like **Folium** or **GeoPandas** could help visualize crime data on interactive maps, making it easier to identify crime hotspots and patterns geographically.

Time Series Forecasting: By focusing more on the time aspect of drug death related data, I could try building time series models (using libraries like **statsmodels** or **Prophet**) to forecast death trends in the future. That could actually be useful for law enforcement planning!

Bigger, Cleaner Datasets: In the future, I'd love to work with larger and more recent datasets that include additional features like demographic data, weather info, or socio-economic factors. This could help find deeper insights and stronger correlations.

Machine Learning Models and NLP : can be used to find more accurate pattern in crimead eventually will give good results.

Real-time Dashboards: It'd be super cool to build a live dashboard using **Dash** so that others can interact with the visualizations and get insights in real-time.

Overall, this project is just a starting point. As I continue learning and building my skills, I hope to make this kind of data analysis more detailed, more useful, and maybe even a bit more impactful.

References

Catalog, “Crime data from 20220 to present ”, catalog.data.gov,2020. [Online].
Available: https://catalog.data.gov/dataset/crime-%20data-from-2020-to-present
https://www.linkedin.com/in/anand003
[Accessed: Apr. 2- 2025].