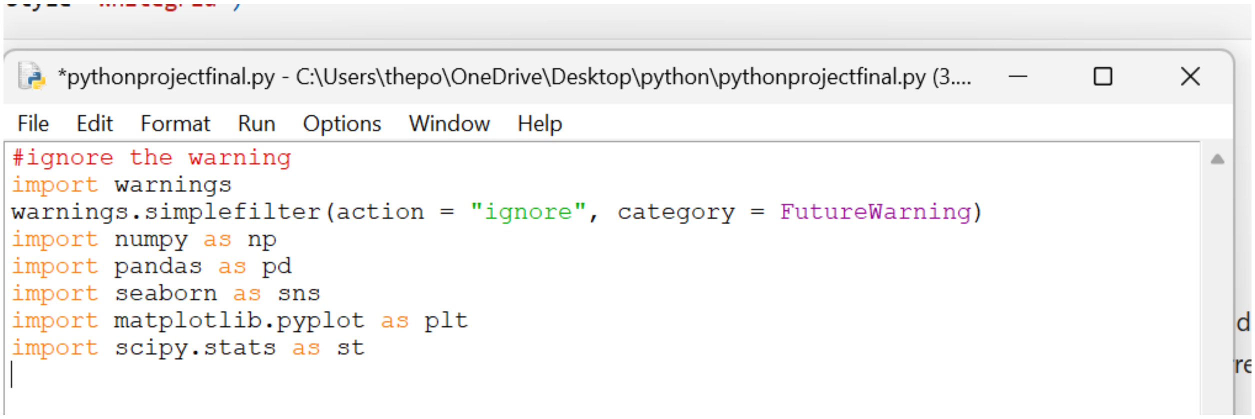
Crime Insights: Data-Driven Crime Analysis

(2020-2025)

- Name – Rohit Kumar
- Reg – no – 12311345
- Roll-no- 47
- Section – K23ED
- In This project I have covered almost every point of python libraries including NumPy pandas mat plot 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 crime dataset between the years 2020 to 2025

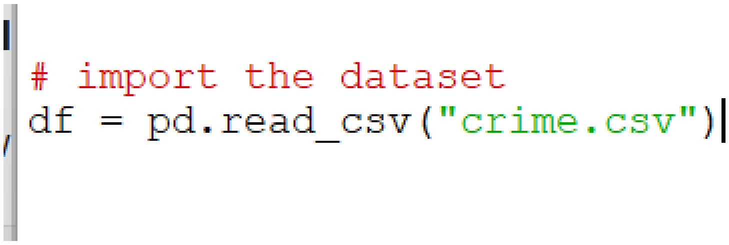
- 1. Importing the warnings and python libraries in idle python --

z


*pythonprojectfinal.py - C:\Users\thepo\OneDrive\Desktop\python\pythonprojectfinal.py (3....
File Edit Format Run Options Window Help

```
#ignore the warning
import warnings
warnings.simplefilter(action = "ignore", category = FutureWarning)
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import scipy.stats as st
```

➢ 2. Importing the data set

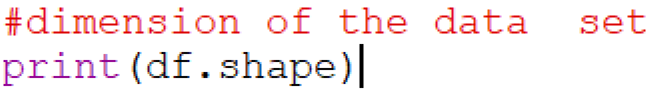


```
# import the dataset
df = pd.read_csv("crime.csv")
```

# Using Pandas

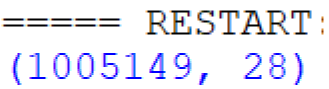
➢ 3. Overview of the data set

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



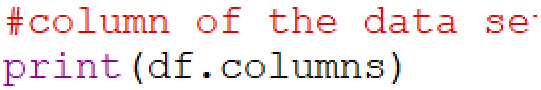
```
#dimension of the data  set
print(df.shape)
```

Ans – The output of the code is the



```
===== RESTART:
(1005149, 28)
```

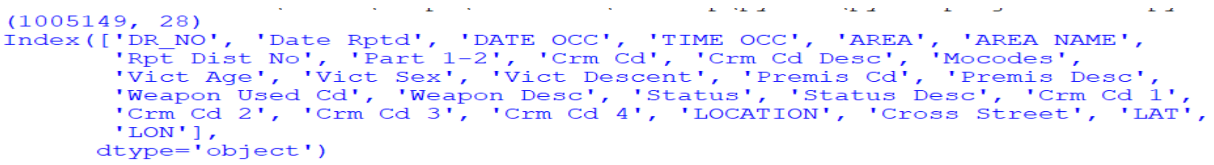
. check the columns of the dataset for that I used attribute



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

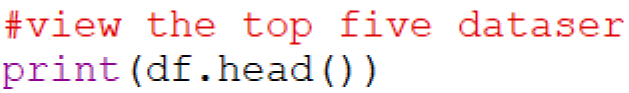
Ans – The output of the code is the

z



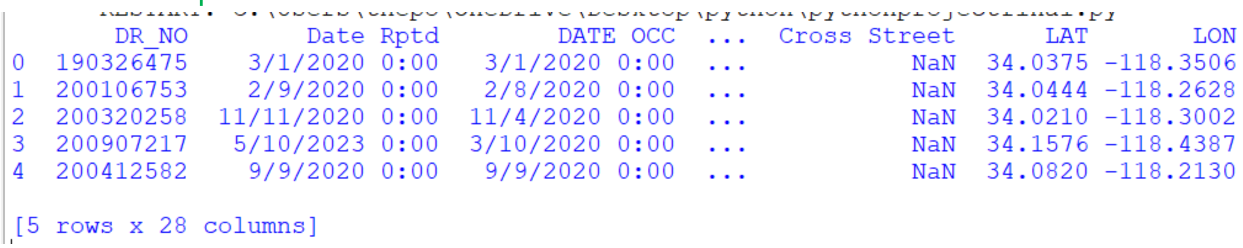
```
(1005149, 28)
Index(['DR_NO', 'Date Rptd', 'DATE OCC', 'TIME OCC', 'AREA', 'AREA NAME',
       'Rpt Dist No', 'Part 1-2', 'Crm Cd', 'Crm Cd Desc', 'Mocodes',
       'Vict Age', 'Vict Sex', 'Vict Descent', 'Premis Cd', 'Premis Desc',
       'Weapon Used Cd', 'Weapon Desc', 'Status', 'Status Desc', '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



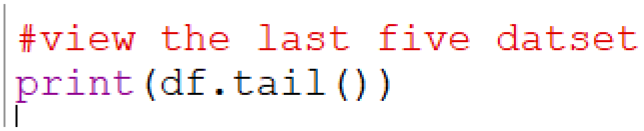
```
#view the top five dataser
print(df.head())
```

Ans – The output of the code is



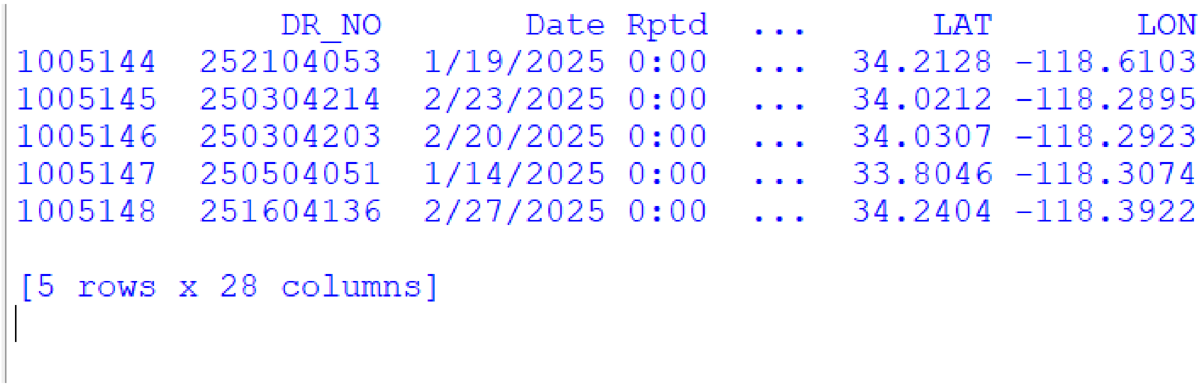
```
       DR_NO         Date Rptd         DATE OCC  ...  Cross Street      LAT       LON
0  190326475    3/1/2020 0:00    3/1/2020 0:00  ...           NaN  34.0375 -118.3506
1  200106753    2/9/2020 0:00    2/8/2020 0:00  ...           NaN  34.0444 -118.2628
2  200320258  11/11/2020 0:00  11/4/2020 0:00  ...           NaN  34.0210 -118.3002
3  200907217   5/10/2023 0:00  3/10/2020 0:00  ...           NaN  34.1576 -118.4387
4  200412582    9/9/2020 0:00    9/9/2020 0:00  ...           NaN  34.0820 -118.2130

[5 rows x 28 columns]
```

. check the list 5 rows of the dataset



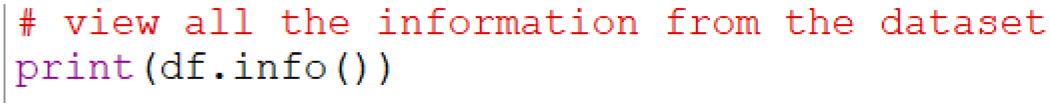
```
#view the last five datset
print(df.tail())
```

Ans – The output of the code is



```
             DR_NO        Date Rptd  ...      LAT       LON
1005144  252104053  1/19/2025 0:00  ...  34.2128 -118.6103
1005145  250304214  2/23/2025 0:00  ...  34.0212 -118.2895
1005146  250304203  2/20/2025 0:00  ...  34.0307 -118.2923
1005147  250504051  1/14/2025 0:00  ...  33.8046 -118.3074
1005148  251604136  2/27/2025 0:00  ...  34.2404 -118.3922

[5 rows x 28 columns]
```

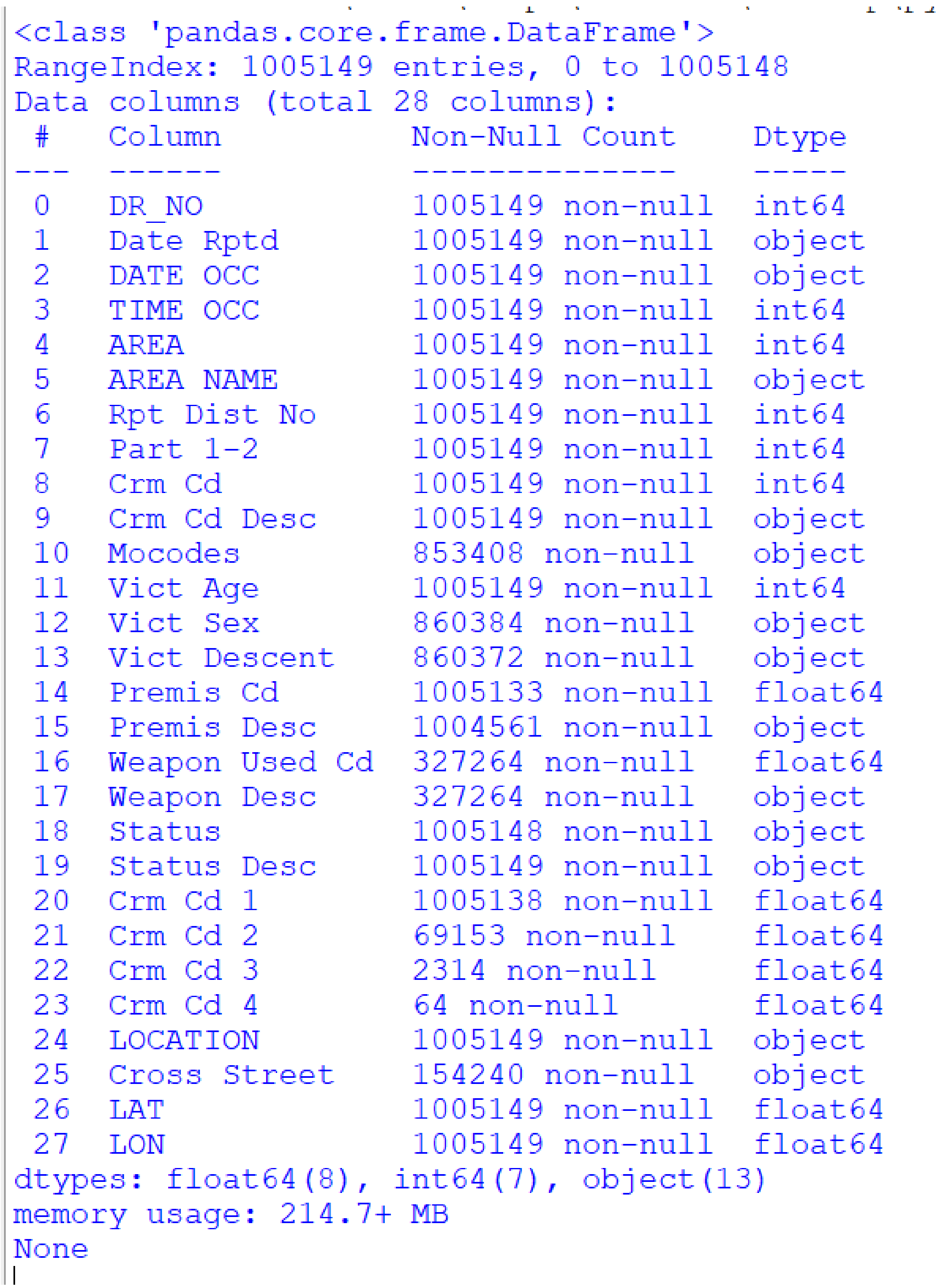
. checking all the information of the dataset and details then we use info function



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

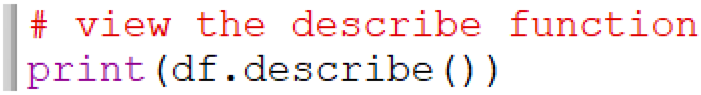
Ans – The output of the code is

z



```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1005149 entries, 0 to 1005148
Data columns (total 28 columns):
 #   Column          Non-Null Count    Dtype
---  ------          --------------    -----
 0   DR_NO           1005149 non-null  int64
 1   Date Rptd       1005149 non-null  object
 2   DATE OCC        1005149 non-null  object
 3   TIME OCC        1005149 non-null  int64
 4   AREA            1005149 non-null  int64
 5   AREA NAME       1005149 non-null  object
 6   Rpt Dist No     1005149 non-null  int64
 7   Part 1-2        1005149 non-null  int64
 8   Crm Cd          1005149 non-null  int64
 9   Crm Cd Desc     1005149 non-null  object
 10  Mocodes         853408 non-null   object
 11  Vict Age        1005149 non-null  int64
 12  Vict Sex        860384 non-null   object
 13  Vict Descent    860372 non-null   object
 14  Premis Cd       1005133 non-null  float64
 15  Premis Desc     1004561 non-null  object
 16  Weapon Used Cd  327264 non-null   float64
 17  Weapon Desc     327264 non-null   object
 18  Status          1005148 non-null  object
 19  Status Desc     1005149 non-null  object
 20  Crm Cd 1        1005138 non-null  float64
 21  Crm Cd 2        69153 non-null    float64
 22  Crm Cd 3        2314 non-null     float64
 23  Crm Cd 4        64 non-null       float64
 24  LOCATION        1005149 non-null  object
 25  Cross Street    154240 non-null   object
 26  LAT             1005149 non-null  float64
 27  LON             1005149 non-null  float64
dtypes: float64(8), int64(7), object(13)
memory usage: 214.7+ MB
None
```

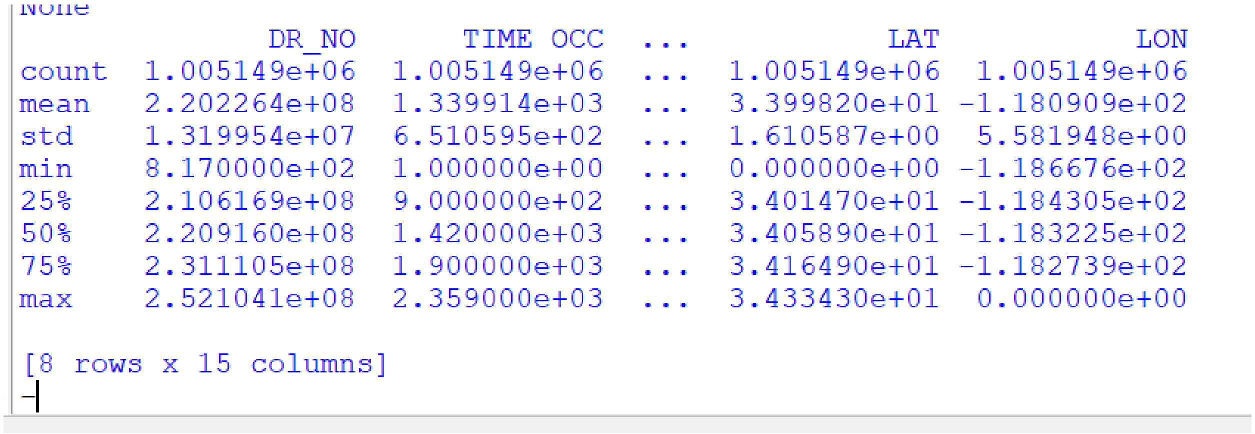
. checking for the describe method it will give you the summary of the invention



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

z

Ans -



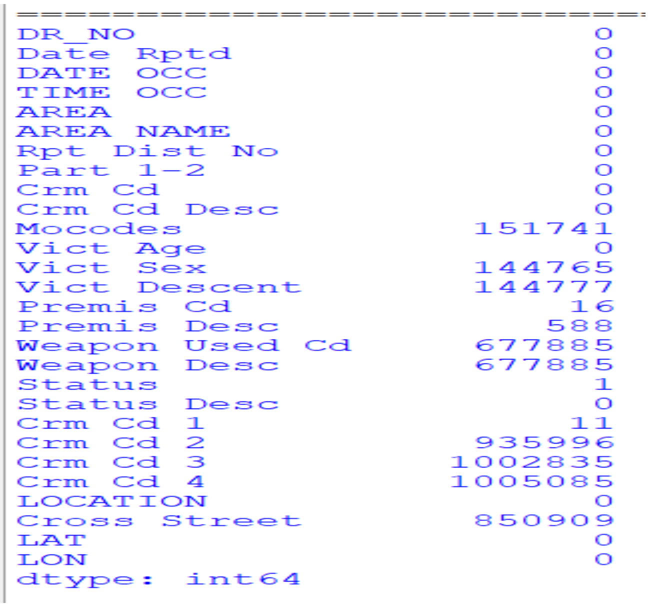
```
None
                   DR_NO     TIME OCC  ...           LAT           LON
count  1.005149e+06  1.005149e+06  ...  1.005149e+06  1.005149e+06
mean   2.202264e+08  1.339914e+03  ...  3.399820e+01 -1.180909e+02
std    1.319954e+07  6.510595e+02  ...  1.610587e+00  5.581948e+00
min    8.170000e+02  1.000000e+00  ...  0.000000e+00 -1.186676e+02
25%    2.106169e+08  9.000000e+02  ...  3.401470e+01 -1.184305e+02
50%    2.209160e+08  1.420000e+03  ...  3.405890e+01 -1.183225e+02
75%    2.311105e+08  1.900000e+03  ...  3.416490e+01 -1.182739e+02
max    2.521041e+08  2.359000e+03  ...  3.433430e+01  0.000000e+00

[8 rows x 15 columns]
```

➢ 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



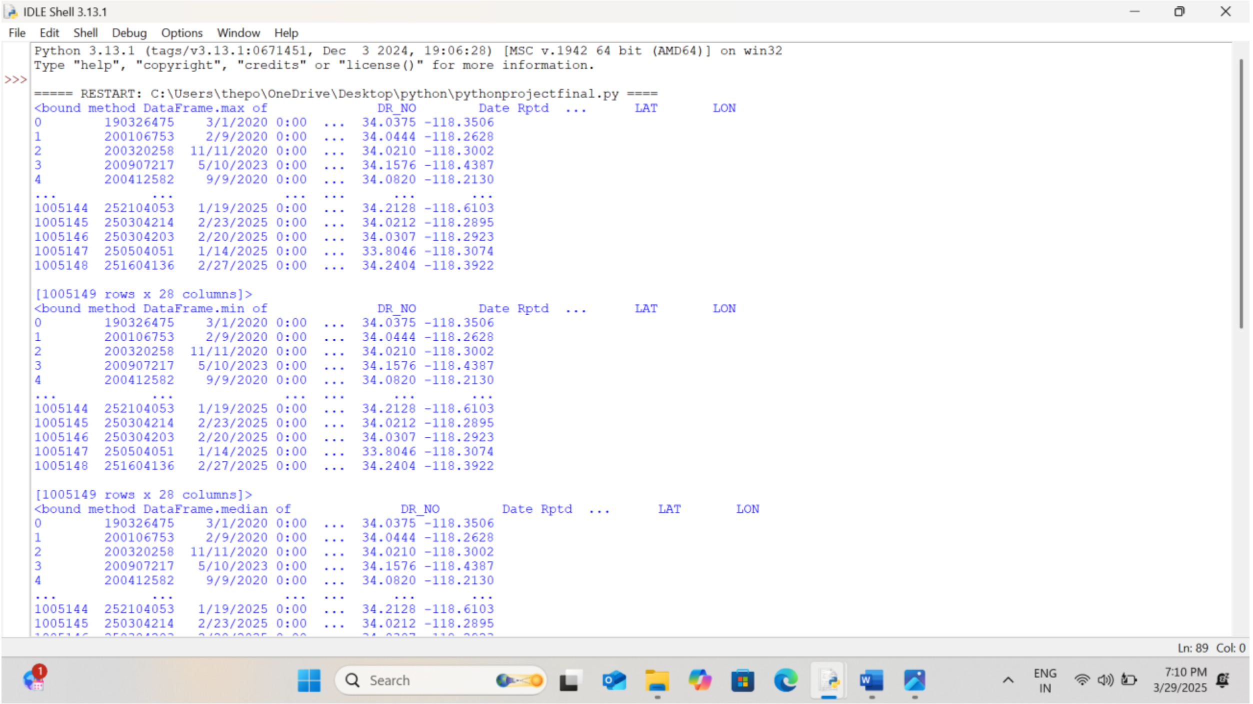
```
==============================
DR_NO                    0
Date Rptd                0
DATE OCC                 0
TIME OCC                 0
AREA                     0
AREA NAME                0
Rpt Dist No              0
Part 1-2                 0
Crm Cd                   0
Crm Cd Desc              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

z

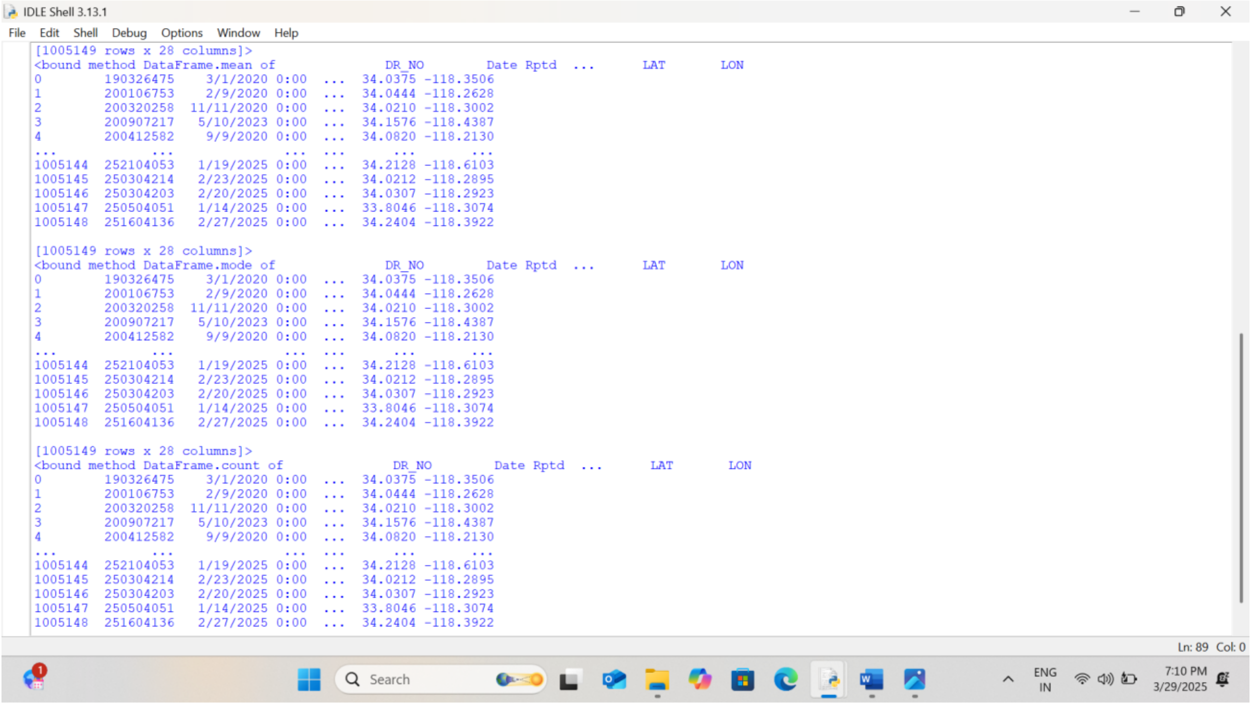
```
#View the max values
print(df.max)
#View the min values
print(df.min)
#view the median values
print(df.median)
#view the mean value
print(df.mean)
#view the mode value
print(df.mode)
#view the count value
print(df.count)
```

Ans - The output of the code is



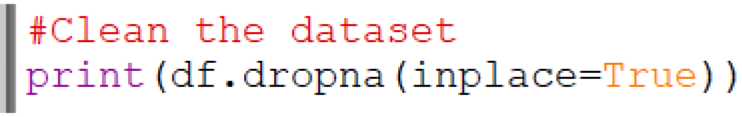
```
Python 3.13.1 (tags/v3.13.1:0671451, Dec  3 2024, 19:06:28) [MSC v.1942 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> 
===== RESTART: C:\Users\thepo\OneDrive\Desktop\python\pythonprojectfinal.py ====
<bound method DataFrame.max of              DR_NO        Date Rptd  ...      LAT       LON
0        190326475    3/1/2020 0:00  ...  34.0375 -118.3506
1        200106753    2/9/2020 0:00  ...  34.0444 -118.2628
2        200320258  11/11/2020 0:00  ...  34.0210 -118.3002
3        200907217   5/10/2023 0:00  ...  34.1576 -118.4387
4        200412582    9/9/2020 0:00  ...  34.0820 -118.2130
...            ...              ...  ...      ...       ...
1005144  252104053   1/19/2025 0:00  ...  34.2128 -118.6103
1005145  250304214   2/23/2025 0:00  ...  34.0212 -118.2895
1005146  250304203   2/20/2025 0:00  ...  34.0307 -118.2923
1005147  250504051   1/14/2025 0:00  ...  33.8046 -118.3074
1005148  251604136   2/27/2025 0:00  ...  34.2404 -118.3922

[1005149 rows x 28 columns]>
<bound method DataFrame.min of              DR_NO        Date Rptd  ...      LAT       LON
0        190326475    3/1/2020 0:00  ...  34.0375 -118.3506
1        200106753    2/9/2020 0:00  ...  34.0444 -118.2628
2        200320258  11/11/2020 0:00  ...  34.0210 -118.3002
3        200907217   5/10/2023 0:00  ...  34.1576 -118.4387
4        200412582    9/9/2020 0:00  ...  34.0820 -118.2130
...            ...              ...  ...      ...       ...
1005144  252104053   1/19/2025 0:00  ...  34.2128 -118.6103
1005145  250304214   2/23/2025 0:00  ...  34.0212 -118.2895
1005146  250304203   2/20/2025 0:00  ...  34.0307 -118.2923
1005147  250504051   1/14/2025 0:00  ...  33.8046 -118.3074
1005148  251604136   2/27/2025 0:00  ...  34.2404 -118.3922

[1005149 rows x 28 columns]>
<bound method DataFrame.median of              DR_NO        Date Rptd  ...      LAT       LON
0        190326475    3/1/2020 0:00  ...  34.0375 -118.3506
1        200106753    2/9/2020 0:00  ...  34.0444 -118.2628
2        200320258  11/11/2020 0:00  ...  34.0210 -118.3002
3        200907217   5/10/2023 0:00  ...  34.1576 -118.4387
4        200412582    9/9/2020 0:00  ...  34.0820 -118.2130
...            ...              ...  ...      ...       ...
1005144  252104053   1/19/2025 0:00  ...  34.2128 -118.6103
1005145  250304214   2/23/2025 0:00  ...  34.0212 -118.2895
```

z



```
[1005149 rows x 28 columns]>
<bound method DataFrame.mean of              DR_NO        Date Rptd  ...      LAT       LON
0         190326475    3/1/2020 0:00  ...  34.0375 -118.3506
1         200106753    2/9/2020 0:00  ...  34.0444 -118.2628
2         200320258  11/11/2020 0:00  ...  34.0210 -118.3002
3         200907217   5/10/2023 0:00  ...  34.1576 -118.4387
4         200412582    9/9/2020 0:00  ...  34.0820 -118.2130
...             ...              ...  ...      ...       ...
1005144  252104053   1/19/2025 0:00  ...  34.2128 -118.6103
1005145  250304214   2/23/2025 0:00  ...  34.0212 -118.2895
1005146  250304203   2/20/2025 0:00  ...  34.0307 -118.2923
1005147  250504051   1/14/2025 0:00  ...  33.8046 -118.3074
1005148  251604136   2/27/2025 0:00  ...  34.2404 -118.3922

[1005149 rows x 28 columns]>
<bound method DataFrame.mode of              DR_NO        Date Rptd  ...      LAT       LON
0         190326475    3/1/2020 0:00  ...  34.0375 -118.3506
1         200106753    2/9/2020 0:00  ...  34.0444 -118.2628
2         200320258  11/11/2020 0:00  ...  34.0210 -118.3002
3         200907217   5/10/2023 0:00  ...  34.1576 -118.4387
4         200412582    9/9/2020 0:00  ...  34.0820 -118.2130
...             ...              ...  ...      ...       ...
1005144  252104053   1/19/2025 0:00  ...  34.2128 -118.6103
1005145  250304214   2/23/2025 0:00  ...  34.0212 -118.2895
1005146  250304203   2/20/2025 0:00  ...  34.0307 -118.2923
1005147  250504051   1/14/2025 0:00  ...  33.8046 -118.3074
1005148  251604136   2/27/2025 0:00  ...  34.2404 -118.3922

[1005149 rows x 28 columns]>
<bound method DataFrame.count of              DR_NO        Date Rptd  ...      LAT       LON
0         190326475    3/1/2020 0:00  ...  34.0375 -118.3506
1         200106753    2/9/2020 0:00  ...  34.0444 -118.2628
2         200320258  11/11/2020 0:00  ...  34.0210 -118.3002
3         200907217   5/10/2023 0:00  ...  34.1576 -118.4387
4         200412582    9/9/2020 0:00  ...  34.0820 -118.2130
...             ...              ...  ...      ...       ...
1005144  252104053   1/19/2025 0:00  ...  34.2128 -118.6103
1005145  250304214   2/23/2025 0:00  ...  34.0212 -118.2895
1005146  250304203   2/20/2025 0:00  ...  34.0307 -118.2923
1005147  250504051   1/14/2025 0:00  ...  33.8046 -118.3074
1005148  251604136   2/27/2025 0:00  ...  34.2404 -118.3922
```

- 6. Checking for the cleaning of the dataset



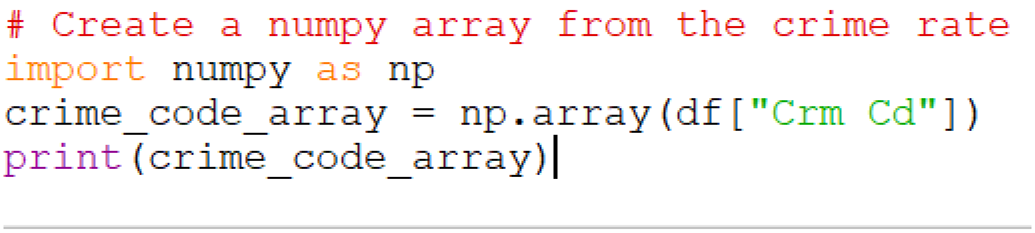
```
#Clean the dataset
print(df.dropna(inplace=True))
```

Ans – The output of the code is the

```
None
```

# CREATION OF NUMPY ARRAY

1.



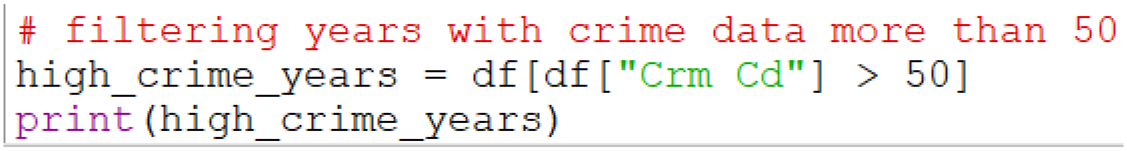
```
# Create a numpy array from the crime rate
import numpy as np
crime_code_array = np.array(df["Crm Cd"])
print(crime_code_array)
```

Ans – The output of the code is

```
[510 330 480 ... 522 210 510]
```

z

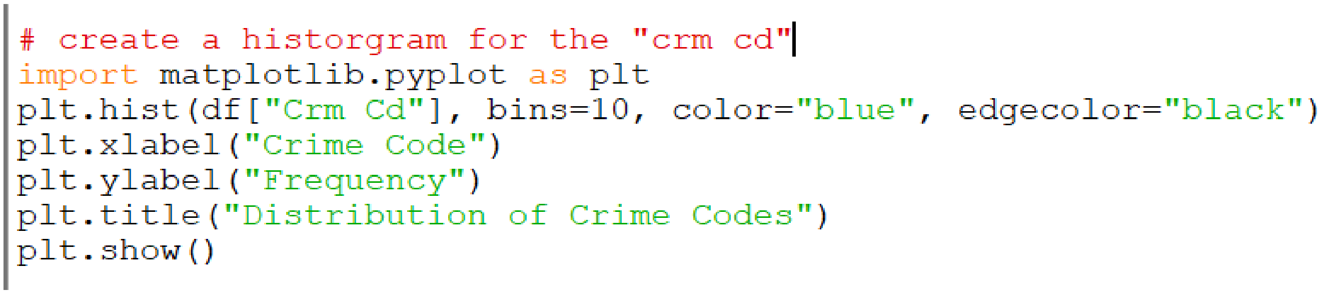
2. filtering years with crime data more than 50



```
# filtering years with crime data more than 50
high_crime_years = df[df["Crm Cd"] > 50]
print(high_crime_years)
```

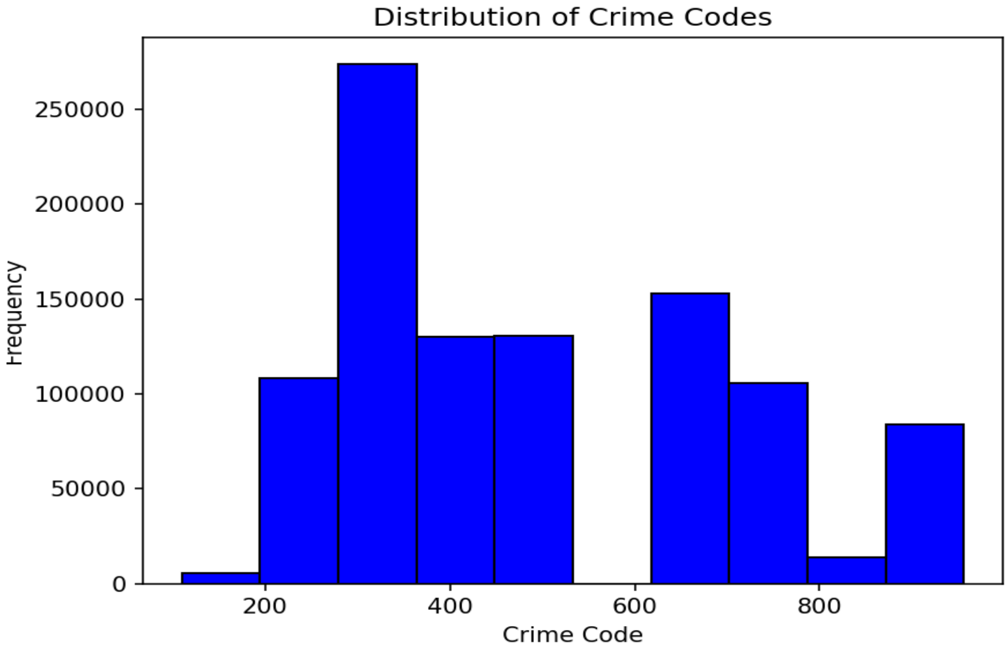
# HISTOGRAM

1. creating a histogram based on crime data



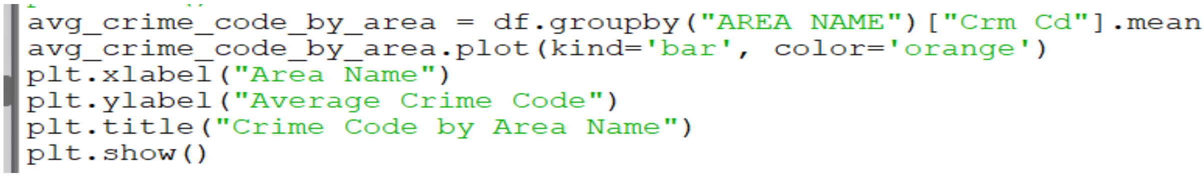
```
# create a historgram for the "crm cd"
import matplotlib.pyplot as plt
plt.hist(df["Crm Cd"], bins=10, color="blue", edgecolor="black")
plt.xlabel("Crime Code")
plt.ylabel("Frequency")
plt.title("Distribution of Crime Codes")
plt.show()
```

Ans – The output of the code is



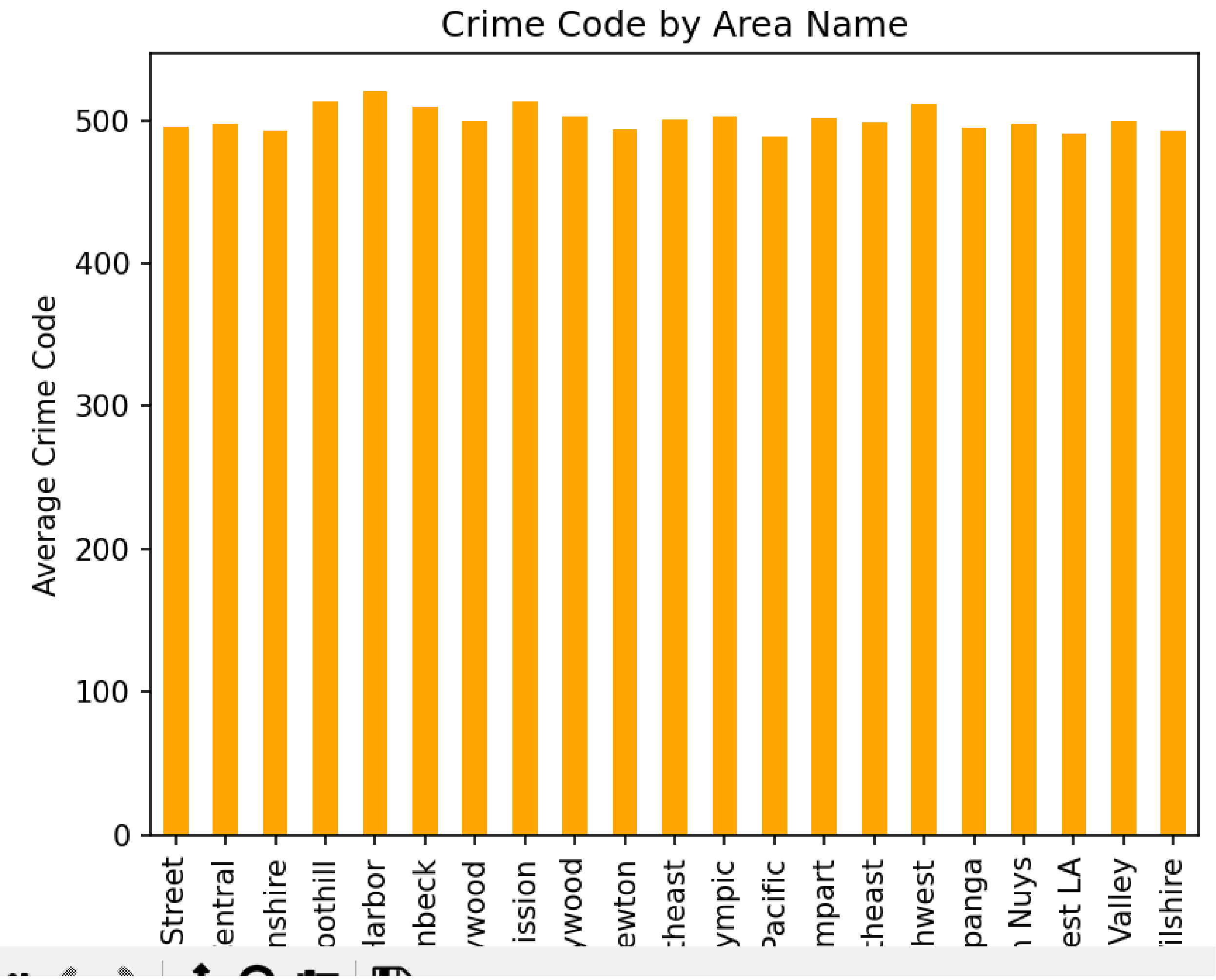
z

2. creating a Bar chart in which Bar chart comparing “ Crm Cd” across “Area Name”



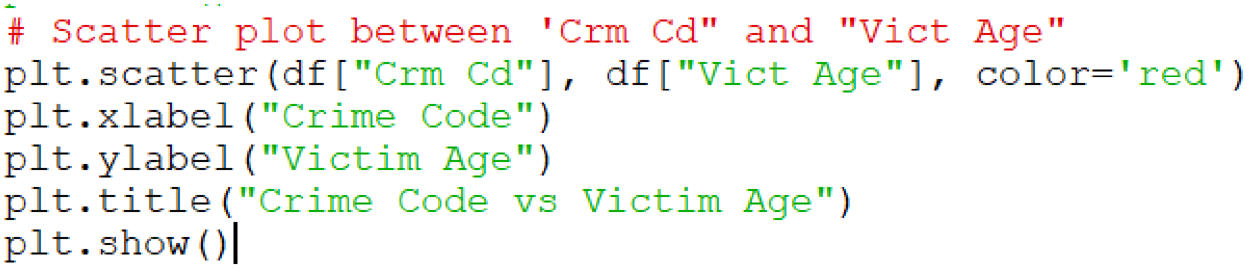
```
avg_crime_code_by_area = df.groupby("AREA NAME")["Crm Cd"].mean
avg_crime_code_by_area.plot(kind='bar', color='orange')
plt.xlabel("Area Name")
plt.ylabel("Average Crime Code")
plt.title("Crime Code by Area Name")
plt.show()
```

Ans – The output of the code is



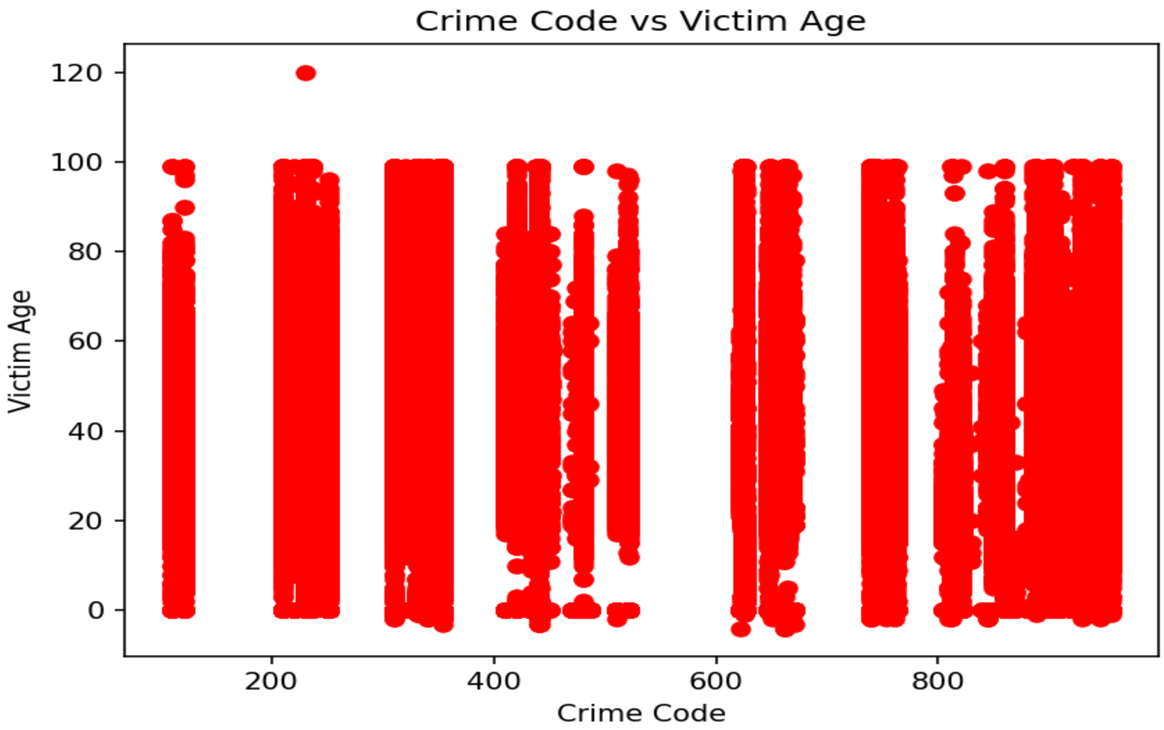
3. creating a scatter plot between “CRM Cd” and “Vict Age”:\

z



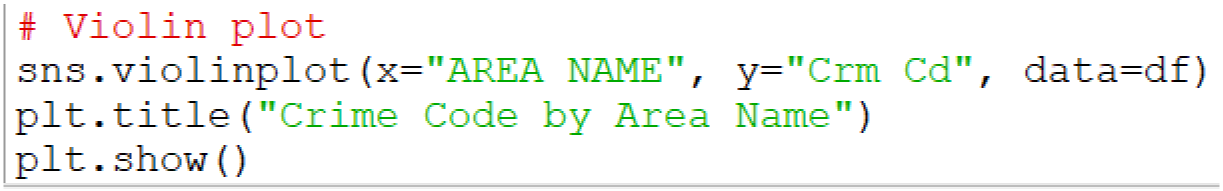
```
# Scatter plot between 'Crm Cd" and "Vict Age"
plt.scatter(df["Crm Cd"], df["Vict Age"], color='red')
plt.xlabel("Crime Code")
plt.ylabel("Victim Age")
plt.title("Crime Code vs Victim Age")
plt.show()
```

Ans – The output of the code is



# Seaborn python libraries

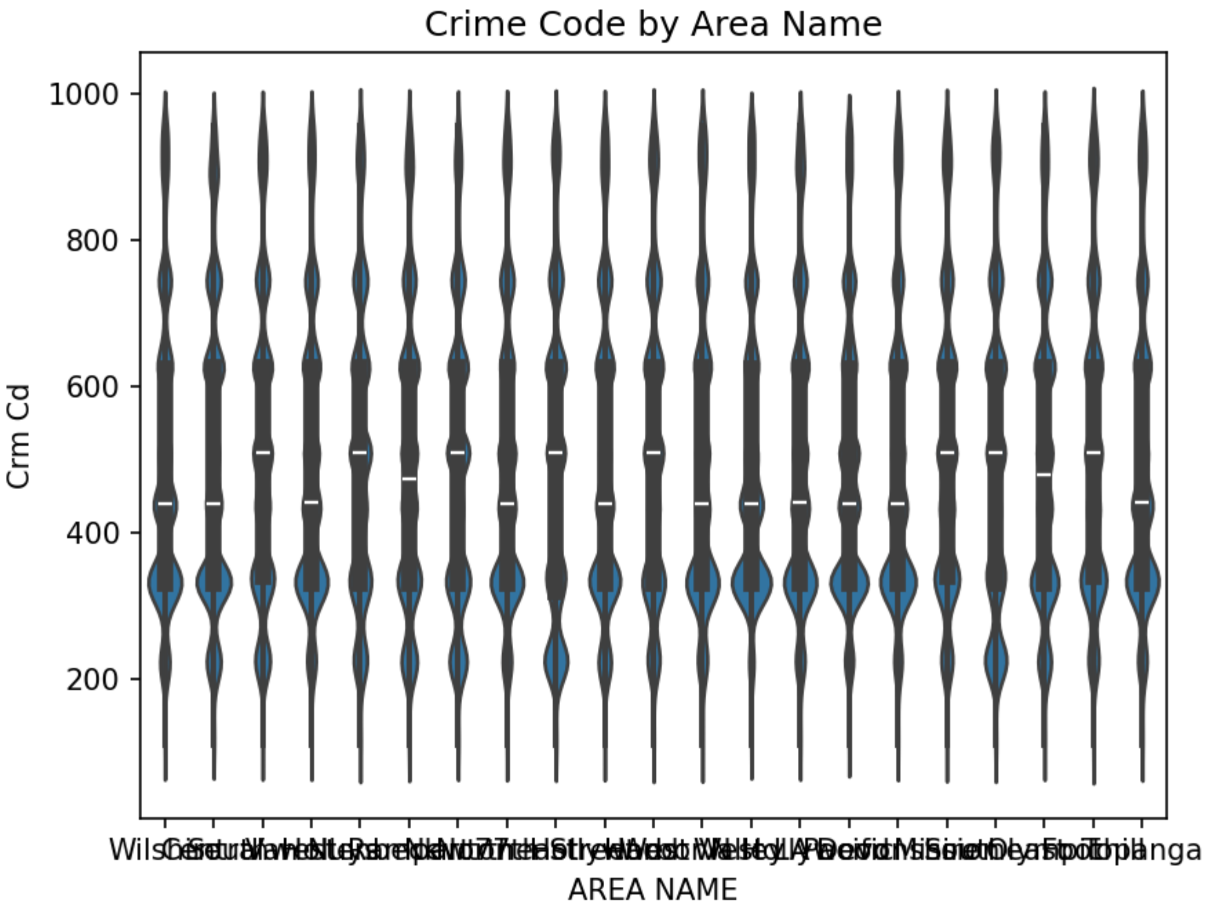
1. Creating a Violin plot for "CRM Cd" grouped by "AREA NAME":



```
# Violin plot
sns.violinplot(x="AREA NAME", y="Crm Cd", data=df)
plt.title("Crime Code by Area Name")
plt.show()
```

Ans – The output of the code is

z

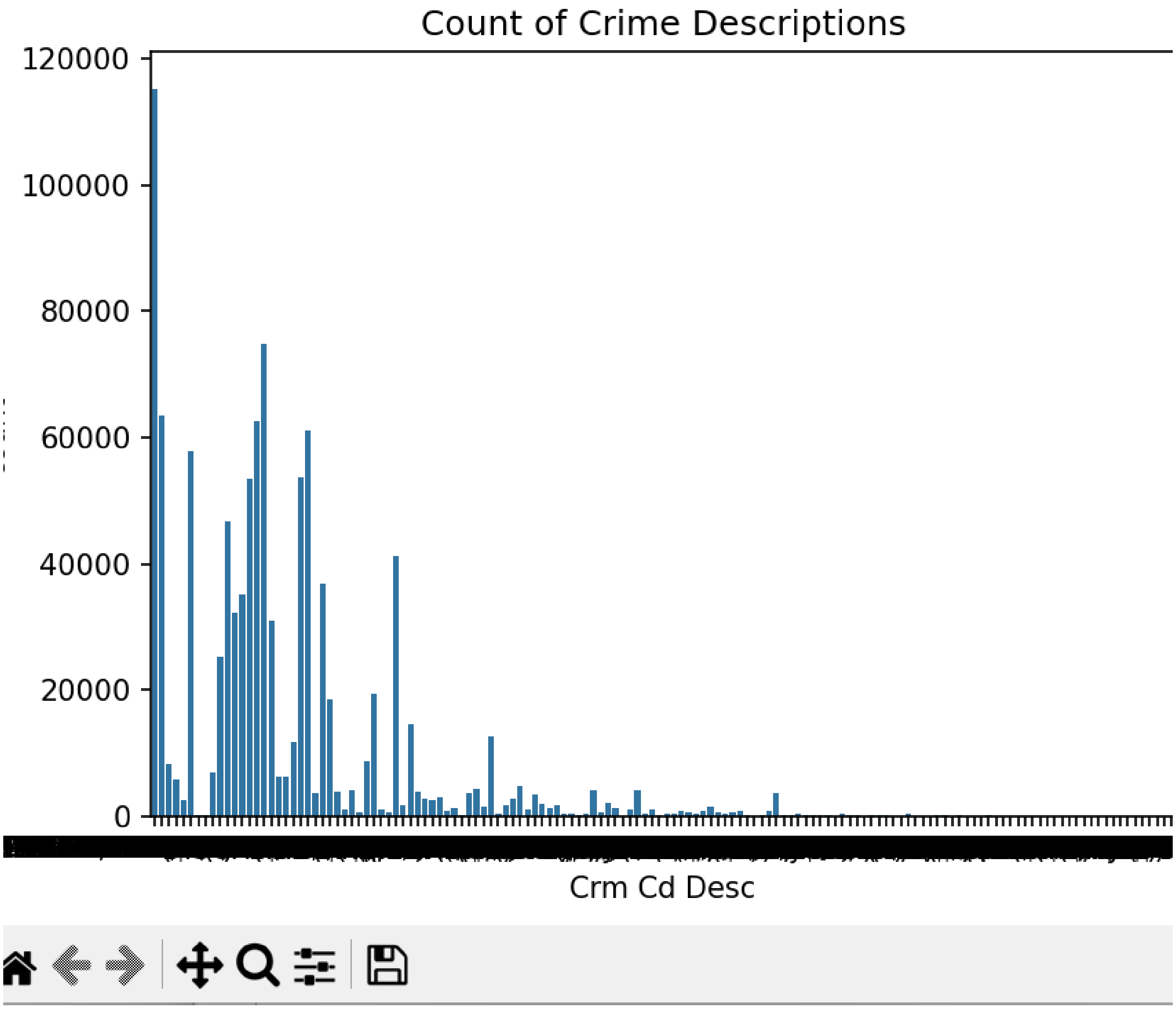


2. Creating a count plot

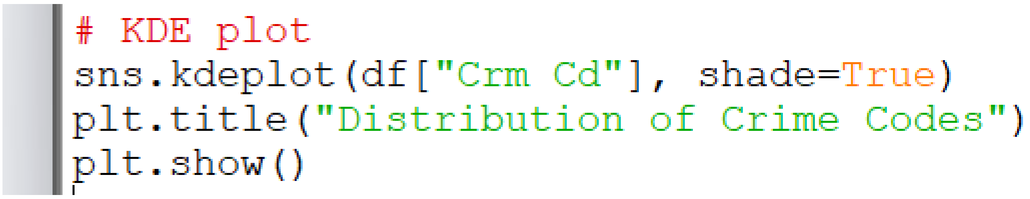
```
# count plot

sns.countplot(x="Crm Cd Desc", data=df)
plt.title("Count of Crime Descriptions")
plt.show()
```

Ans – The output of the code is

z



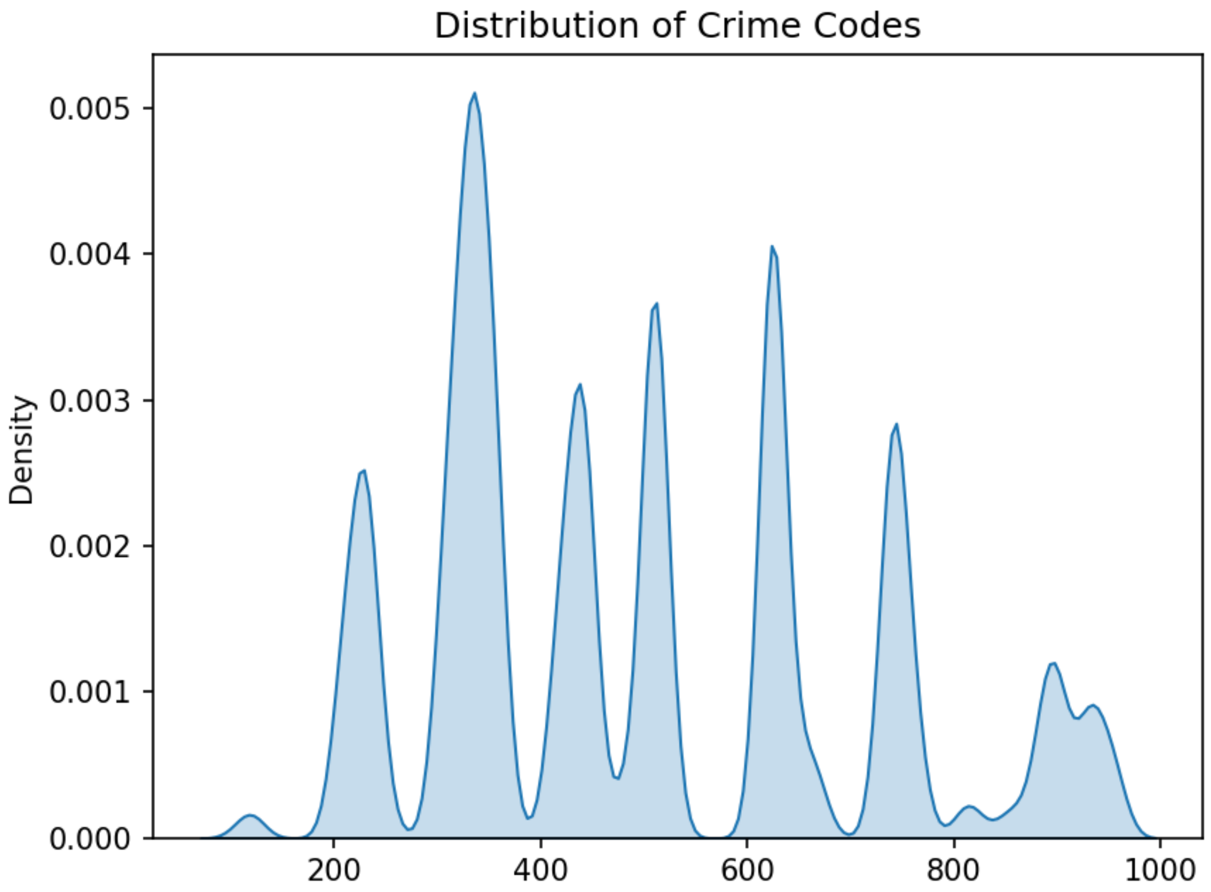
3. Creating a KDE Graph



```
# KDE plot
sns.kdeplot(df["Crm Cd"], shade=True)
plt.title("Distribution of Crime Codes")
plt.show()
```

Ans – The output of the code is the

z



BELOW I HAVE WRITTEN ALL THE CODE WHICH I HAVE WRITE IN IDLE PYTHON

```
import warnings
warnings.simplefilter(action="ignore", category=FutureWarning)

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

# Importing the dataset
df = pd.read_csv("crime.csv")

# Checking the dimensions of the dataset
# print(df.shape)

# Listing the columns of the dataset
```

z

```
# print(df.columns)

# Viewing the first five rows of the dataset
# print(df.head())

# Viewing the last five rows of the dataset
# print(df.tail())

# Viewing all the information about the dataset
# print(df.info())

# Descriptive statistics of the dataset
# print(df.describe())

# Checking for missing values in the dataset and their total count
# print(df.isnull().sum())

# Viewing the maximum values in the dataset
# print(df.max())

# Viewing the minimum values in the dataset
# print(df.min())

# Viewing the median values in the dataset
# print(df.median())

# Viewing the mean values in the dataset
# print(df.mean())

# Viewing the mode values in the dataset
# print(df.mode())

# Counting non-null values in each column
# print(df.count())

# Cleaning the dataset by dropping rows with missing values
# print(df.dropna(inplace=True))

# Creating a numpy array from the crime rate
# crime_code_array = np.array(df["Crm Cd"])
```

z

```
# print(crime_code_array)

# Filtering years with crime data greater than 50
high_crime_years = df[df["Crm Cd"] > 50]
# print(high_crime_years)

# Creating a histogram for the "Crm Cd" column
# plt.hist(df["Crm Cd"], bins=10, color="blue", edgecolor="black")
# plt.xlabel("Crime Code")
# plt.ylabel("Frequency")
# plt.title("Distribution of Crime Codes")
# plt.show()

# Creating a bar chart to show the average crime code by area
# avg_crime_code_by_area = df.groupby("AREA NAME")["Crm Cd"].mean()
# avg_crime_code_by_area.plot(kind='bar', color='orange')
# plt.xlabel("Area Name")
# plt.ylabel("Average Crime Code")
# plt.title("Crime Code by Area Name")
# plt.show()

# Creating a line graph to show the trend of crime code across dates
# plt.plot(df["DATE OCC"], df["Crm Cd"], marker='o')
# plt.xlabel("Date of Occurrence")
# plt.ylabel("Crime Code")
# plt.title("Trend of Crime Code Across Dates")
# plt.show()

# Scatter plot between 'Crm Cd' and 'Vict Age'
# plt.scatter(df["Crm Cd"], df["Vict Age"], color='red')
# plt.xlabel("Crime Code")
# plt.ylabel("Victim Age")
# plt.title("Crime Code vs Victim Age")
# plt.show()

# Boxplot for "Crm Cd" distribution by year
# import seaborn as sns
# sns.boxplot(x="Year", y="Crm Cd", data=df)
# plt.title("Crime Code Distribution by Year")
# plt.show()
```

z

```
# Creating a heatmap to visualize the correlation between features
# sns.heatmap(df.corr(), annot=True, cmap="coolwarm")
# plt.title("Feature Correlation Heatmap")
# plt.show()

# Violin plot for crime code distribution by area
# sns.violinplot(x="AREA NAME", y="Crm Cd", data=df)
# plt.title("Crime Code by Area Name")
# plt.show()

# Swarm plot for crime code distribution across years
# sns.swarmplot(x="Year", y="Crm Cd", data=df)
# plt.xticks(rotation=90)
# plt.title("Crime Code Across Years")
# plt.show()

# Pair plot for selected columns
# sns.pairplot(df[["Crm Cd", "Vict Age", "Year"]])
# plt.show()

# Count plot for crime descriptions
# sns.countplot(x="Crm Cd Desc", data=df)
# plt.title("Count of Crime Descriptions")
# plt.show()

# KDE plot for the distribution of crime codes
# sns.kdeplot(df["Crm Cd"], shade=True)
# plt.title("Distribution of Crime Codes")
# plt.show()
```

z