Name: Anish Kulkarni Roll No.: 29 Class: D15C AY: 2024-25

Experiment 1

Aim: Introduction to Data science and Data preparation using Pandas steps. Solve the following questions:-

- Load data in Pandas.
- Description of the dataset.
- Drop columns that aren't useful.
- Drop rows with maximum missing values.
- Take care of missing data.
- Create dummy variables.
- Find out outliers (manually)
- Standardization and Normalization of columns

Steps:

- Load data in Pandas:-

Step 1: Run the following commands:

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

These commands import essential libraries for data manipulation (pandas), numerical computations (numpy), and data visualization using matplotlib for basic plotting and seaborn for enhanced statistical graphics.

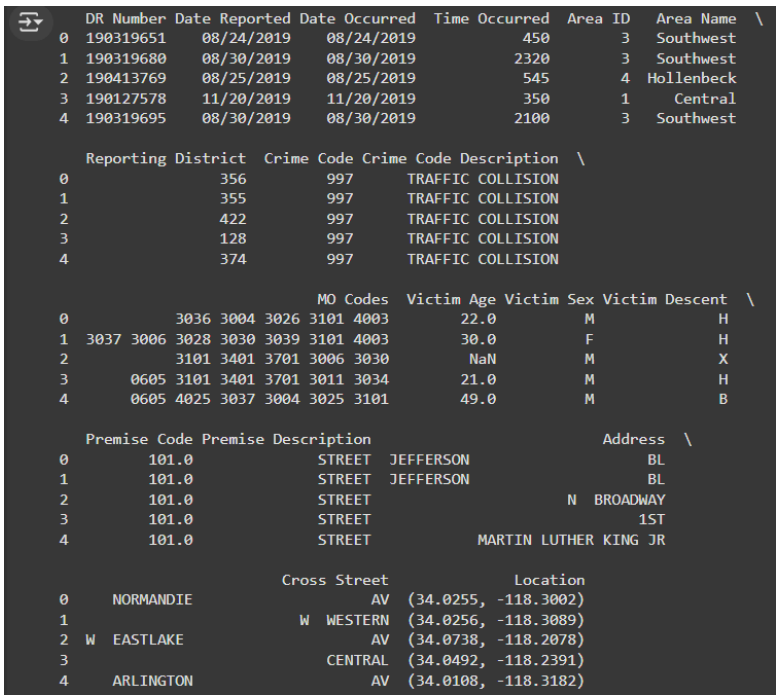
Step 2: Run the following command:

```
df = pd.read_csv('Traffic_Collision_Data_from_2010_to_Present.csv')
```

This command reads the Traffic_Collision_Data_from_2010_to_Present.csv file into a pandas DataFrame (df), allowing for structured data manipulation and analysis.

- Description of the dataset:-

Command 1: print(df.head())
This command prints the first five rows of the DataFrame df, giving a quick preview of the dataset's structure.

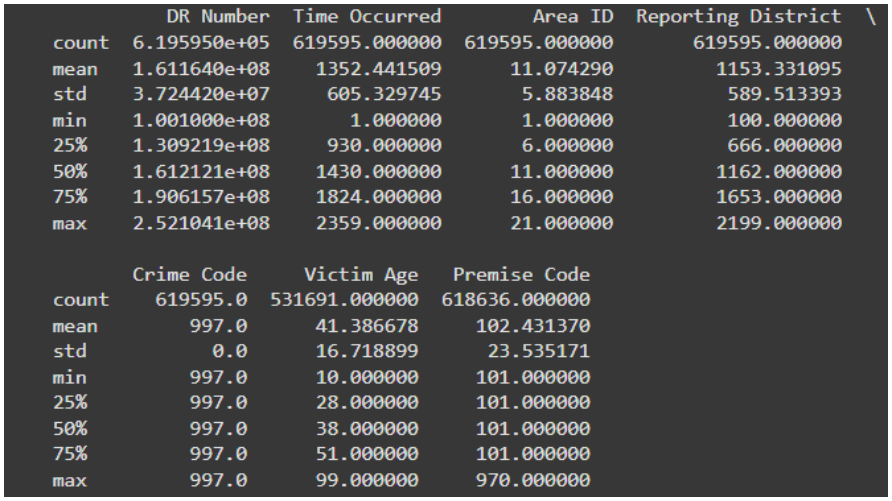


```
   DR Number Date Reported Date Occurred  Time Occurred  Area ID   Area Name  \
0  190319651    08/24/2019    08/24/2019            450        3   Southwest
1  190319680    08/30/2019    08/30/2019           2320        3   Southwest
2  190413769    08/25/2019    08/25/2019            545        4  Hollenbeck
3  190127578    11/20/2019    11/20/2019            350        1     Central
4  190319695    08/30/2019    08/30/2019           2100        3   Southwest

   Reporting District  Crime Code Crime Code Description  \
0                 356         997      TRAFFIC COLLISION
1                 355         997      TRAFFIC COLLISION
2                 422         997      TRAFFIC COLLISION
3                 128         997      TRAFFIC COLLISION
4                 374         997      TRAFFIC COLLISION

                          MO Codes  Victim Age Victim Sex Victim Descent  \
0           3036 3004 3026 3101 4003        22.0          M              H
1  3037 3006 3028 3030 3039 3101 4003        30.0          F              H
2           3101 3401 3701 3006 3030         NaN          M              X
3      0605 3101 3401 3701 3011 3034        21.0          M              H
4      0605 4025 3037 3004 3025 3101        49.0          M              B

   Premise Code Premise Description                          Address  \
0         101.0              STREET  JEFFERSON                    BL
1         101.0              STREET  JEFFERSON                    BL
2         101.0              STREET                     N  BROADWAY
3         101.0              STREET                           1ST
4         101.0              STREET            MARTIN LUTHER KING JR

                     Cross Street             Location
0    NORMANDIE                 AV  (34.0255, -118.3002)
1                    W  WESTERN  (34.0256, -118.3089)
2  W  EASTLAKE                 AV  (34.0738, -118.2078)
3                       CENTRAL  (34.0492, -118.2391)
4    ARLINGTON                 AV  (34.0108, -118.3182)
```

Command 2: print(df.info())
This command displays a summary of the DataFrame df, including the number of rows and columns, data types, and non-null value counts for each column.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 619595 entries, 0 to 619594
Data columns (total 18 columns):
 #   Column                  Non-Null Count   Dtype
---  ------                  --------------   -----
 0   DR Number               619595 non-null  int64
 1   Date Reported           619595 non-null  object
 2   Date Occurred           619595 non-null  object
 3   Time Occurred           619595 non-null  int64
 4   Area ID                 619595 non-null  int64
 5   Area Name               619595 non-null  object
 6   Reporting District      619595 non-null  int64
 7   Crime Code              619595 non-null  int64
 8   Crime Code Description  619595 non-null  object
 9   MO Codes                532293 non-null  object
 10  Victim Age              531691 non-null  float64
 11  Victim Sex              608958 non-null  object
 12  Victim Descent          608007 non-null  object
 13  Premise Code            618636 non-null  float64
 14  Premise Description     618635 non-null  object
 15  Address                 619595 non-null  object
 16  Cross Street            590242 non-null  object
 17  Location                619595 non-null  object
dtypes: float64(2), int64(5), object(11)
memory usage: 85.1+ MB
None
```

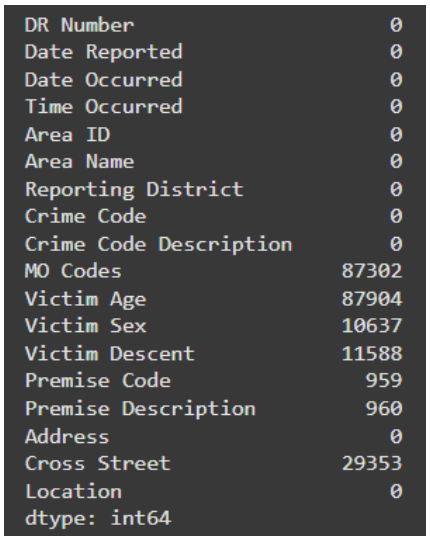
Command 3: print(df.describe())
This command generates summary statistics for numerical columns in the DataFrame df, including count, mean, standard deviation, minimum, maximum, and quartile values.



	DR Number	Time Occurred	Area ID	Reporting District	\
count	6.195950e+05	619595.000000	619595.000000	619595.000000	
mean	1.611640e+08	1352.441509	11.074290	1153.331095	
std	3.724420e+07	605.329745	5.883848	589.513393	
min	1.001000e+08	1.000000	1.000000	100.000000	
25%	1.309219e+08	930.000000	6.000000	666.000000	
50%	1.612121e+08	1430.000000	11.000000	1162.000000	
75%	1.906157e+08	1824.000000	16.000000	1653.000000	
max	2.521041e+08	2359.000000	21.000000	2199.000000	

	Crime Code	Victim Age	Premise Code
count	619595.0	531691.000000	618636.000000
mean	997.0	41.386678	102.431370
std	0.0	16.718899	23.535171
min	997.0	10.000000	101.000000
25%	997.0	28.000000	101.000000
50%	997.0	38.000000	101.000000
75%	997.0	51.000000	101.000000
max	997.0	99.000000	970.000000

Command 4: print(df.isnull().sum())
This command prints the total number of missing (null) values in each column of the DataFrame df, helping identify incomplete data.



DR Number	0
Date Reported	0
Date Occurred	0
Time Occurred	0
Area ID	0
Area Name	0
Reporting District	0
Crime Code	0
Crime Code Description	0
MO Codes	87302
Victim Age	87904
Victim Sex	10637
Victim Descent	11588
Premise Code	959
Premise Description	960
Address	0
Cross Street	29353
Location	0

dtype: int64

- Drop columns that aren't useful:-

Command: columns_to_drop = ['Crime Code Description', 'MO Codes', 'Address', 'Cross Street']
df.drop(columns=columns_to_drop, inplace=True)
print(df.head())

The above drops the columns ('Crime Code Description', 'MO Codes', 'Address', and 'Cross Street') from the DataFrame df, as they are deemed not useful for the analysis, and updates the DataFrame in place. The print(df.head()) command then displays the first five rows of the modified DataFrame.

	DR Number	Date Reported	Date Occurred	Time Occurred	Area ID	Area Name	\
0	190319651	08/24/2019	08/24/2019	450	3	Southwest	
1	190319680	08/30/2019	08/30/2019	2320	3	Southwest	
2	190413769	08/25/2019	08/25/2019	545	4	Hollenbeck	
3	190127578	11/20/2019	11/20/2019	350	1	Central	
4	190319695	08/30/2019	08/30/2019	2100	3	Southwest	

	Reporting District	Crime Code	Victim Age	Victim Sex	Victim Descent	\
0	356	997	22.0	M	H	
1	355	997	30.0	F	H	
2	422	997	NaN	M	X	
3	128	997	21.0	M	H	
4	374	997	49.0	M	B	

	Premise Code	Premise Description	Location
0	101.0	STREET	(34.0255, -118.3002)
1	101.0	STREET	(34.0256, -118.3089)
2	101.0	STREET	(34.0738, -118.2078)
3	101.0	STREET	(34.0492, -118.2391)
4	101.0	STREET	(34.0108, -118.3182)

- Drop rows with maximum missing values:-

Command: threshold = df.shape[1] * 0.7
df.dropna(thresh=threshold, inplace=True)

The above calculates a threshold for the minimum number of non-null values required in a row (70% of the total columns) and drops rows in the DataFrame df that have fewer non-null values than this threshold, updating the DataFrame in place.

```
Dataset shape before dropping rows with maximum missing values: (619595, 14)
Dataset shape after dropping rows with maximum missing values: (619572, 14)
```

- Take care of missing data:-

Command: df.fillna({'Victim Age': df['Victim Age'].median()}, inplace=True)

```
categorical_columns = ['Victim Sex', 'Victim Descent', 'Premise Description', 'Premise Code']
for col in categorical_columns:
  df[col].fillna(df[col].mode()[0], inplace=True)
print(df.isnull().sum())
```

The above code fills missing values in the DataFrame df as follows:

- ☐ Numerical missing values in the 'Victim Age' column are filled with the median of that column.
- ☐ Categorical missing values in the specified columns ('Victim Sex', 'Victim Descent', 'Premise Description', 'Premise Code') are filled with the mode (most frequent value) of each respective column.

print(df.isnull().sum()) is used to verify that there are no more missing values in the DataFrame.

```
DR Number             0
Date Reported         0
Date Occurred         0
Time Occurred         0
Area ID               0
Area Name             0
Reporting District    0
Crime Code            0
Victim Age            0
Victim Sex            0
Victim Descent        0
Premise Code          0
Premise Description   0
Location              0
dtype: int64
```

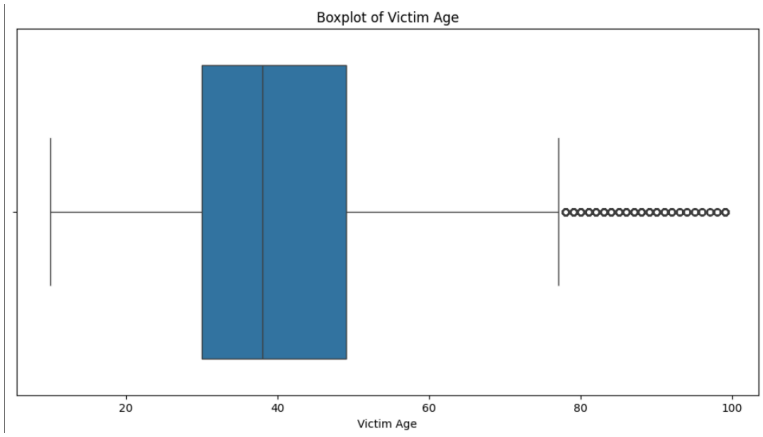
- Create dummy variables:-

Command: df = pd.get_dummies(df, columns=['Area Name', 'Victim Sex', 'Premise Description'], drop_first=True)
The above code changes the categorical columns ('Area Name', 'Victim Sex', and 'Premise Description') into separate columns with 0s and 1s to represent each category, and removes the first category in each to prevent confusion during analysis.

```
   DR Number Date Reported Date Occurred  Time Occurred  Area ID  \
0  190319651    08/24/2019    08/24/2019            450        3
1  190319680    08/30/2019    08/30/2019           2320        3
2  190413769    08/25/2019    08/25/2019            545        4
3  190127578    11/20/2019    11/20/2019            350        1
4  190319695    08/30/2019    08/30/2019           2100        3

   Reporting District  Crime Code  Victim Age Victim Descent  Premise Code  \
0                 356         997        22.0              H         101.0
1                 355         997        30.0              H         101.0
2                 422         997        38.0              X         101.0
3                 128         997        21.0              H         101.0
4                 374         997        49.0              B         101.0

   ... Premise Description_TRAM/STREETCAR(BOXLIKE WAG ON RAILS)*  \
0  ...                                                 False
1  ...                                                 False
2  ...                                                 False
3  ...                                                 False
4  ...                                                 False

   Premise Description_TRANSPORTATION FACILITY (AIRPORT)  \
0                                              False
1                                              False
2                                              False
3                                              False
4                                              False
```

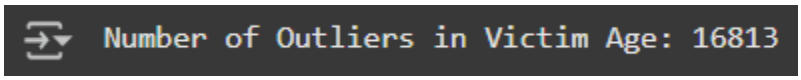
- Find out outliers (manually):-

Command 1: plt.figure(figsize=(12,6))
sns.boxplot(x=df['Victim Age'])
plt.title('Boxplot of Victim Age')
plt.show()
The first method uses a boxplot to visually identify outliers in the 'Victim Age' column by showing the distribution and extreme values.



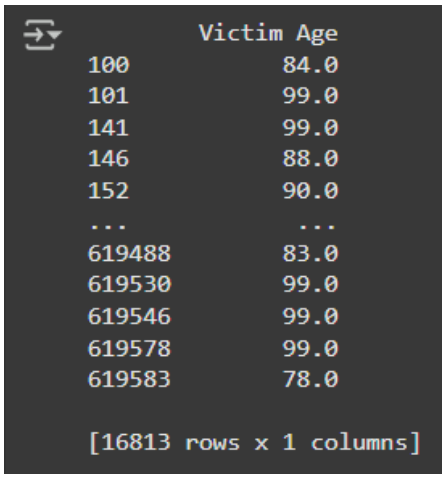
Command 2: Q1 = df['Victim Age'].quantile(0.25)
Q3 = df['Victim Age'].quantile(0.75)
IQR = Q3 - Q1
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR
outliers = df[(df['Victim Age'] < lower_bound) | (df['Victim Age'] > upper_bound)]
print("Number of Outliers in Victim Age:", len(outliers))

The second method calculates the IQR (Interquartile Range), defines a range using 1.5 times the IQR, and identifies outliers numerically by checking values outside this range.



Command 3: Q1 = df['Victim Age'].quantile(0.25)
Q3 = df['Victim Age'].quantile(0.75)
IQR = Q3 - Q1
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR
outliers = df[(df['Victim Age'] < lower_bound) | (df['Victim Age'] > upper_bound)]
print(outliers[['Victim Age']])

Instead of just counting the outliers, we can also list the actual outlier values from the 'Victim Age' column as seen below.

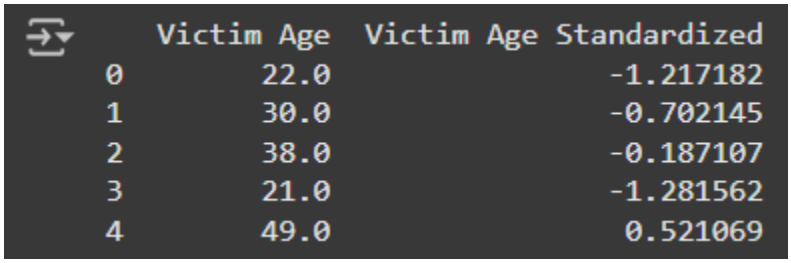


	Victim Age
100	84.0
101	99.0
141	99.0
146	88.0
152	90.0
...	...
619488	83.0
619530	99.0
619546	99.0
619578	99.0
619583	78.0

[16813 rows x 1 columns]

- Standardization of columns:-

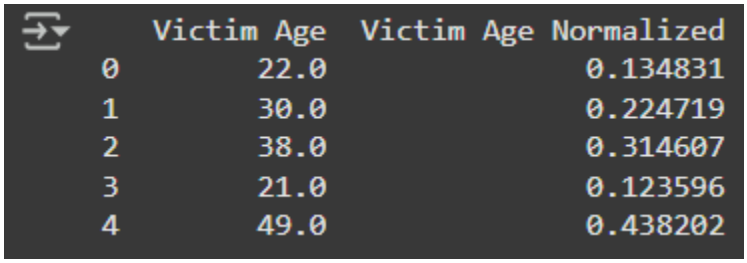
Command: from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
df['Victim Age Standardized'] = scaler.fit_transform(df[['Victim Age']])
The above code applies standardization to the 'Victim Age' column using StandardScaler from scikit-learn, transforming the data so that it has a mean of 0 and a standard deviation of 1, and stores the standardized values in a new column 'Victim Age Standardized' for better comparability and normalization.



	Victim Age	Victim Age Standardized
0	22.0	-1.217182
1	30.0	-0.702145
2	38.0	-0.187107
3	21.0	-1.281562
4	49.0	0.521069

- Normalization of columns:-

Command: from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
df['Victim Age Normalized'] = scaler.fit_transform(df[['Victim Age']])
The above code applies normalization, a technique that rescales the values of the 'Victim Age' column to a range between 0 and 1, using MinMaxScaler from scikit-learn, and stores the normalized values in a new column 'Victim Age Normalized' to ensure the data is on a consistent scale for improved model performance.



	Victim Age	Victim Age Normalized
0	22.0	0.134831
1	30.0	0.224719
2	38.0	0.314607
3	21.0	0.123596
4	49.0	0.438202

Conclusion:

1. In this experiment, we learned how to preprocess data using Pandas steps.
2. By loading data into Pandas, we successfully imported the dataset into a Pandas DataFrame for analysis.
3. By describing the dataset, we understood the structure and statistics of the data using .info() and .describe().

4. By dropping unnecessary columns, we removed columns that were not relevant to the analysis to reduce complexity.
5. By dropping rows with maximum missing values, we eliminated rows where a high percentage of data was missing to improve data quality.
6. By handling missing data, we filled missing values appropriately to ensure data consistency.
7. By creating dummy variables, we converted categorical columns ('Area Name', 'Victim Sex', and 'Premise Description') into numerical representations using one-hot encoding.
8. By detecting outliers, we identified outliers using a Boxplot and Interquartile Range (IQR). We also displayed the outliers.
9. By standardizing and normalizing a numerical column, we ensured uniform data distribution by applying feature scaling techniques (standardization and normalization).
10. After preprocessing the data (removing rows with maximum missing values, removing outliers, etc), the number of rows in the dataset reduced from 619,595 to 602,702 (decrease of 2.72%).