ADITYA LALWANI D15C ROLL NO.30

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.

Step 2: Run the following command:

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

This command reads the dataset.csv file into a pandas DataFrame (df)

ADITYA LALWANI D15C ROLL NO.30

Description of the dataset:-

Command 1: print(df.head())

This command prints the first five rows of the DataFrame df

```
print(df.head())
```

```
   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, etc.

```
print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 484302 entries, 0 to 484301
Data columns (total 18 columns):
 #   Column                  Non-Null Count   Dtype
---  ------                  --------------   -----
 0   DR Number               484302 non-null  int64
 1   Date Reported           484302 non-null  object
 2   Date Occurred           484302 non-null  object
 3   Time Occurred           484302 non-null  int64
 4   Area ID                 484302 non-null  int64
 5   Area Name               484302 non-null  object
 6   Reporting District      484302 non-null  int64
 7   Crime Code              484302 non-null  int64
 8   Crime Code Description  484302 non-null  object
 9   MO Codes                399199 non-null  object
 10  Victim Age              414993 non-null  float64
 11  Victim Sex              476778 non-null  object
 12  Victim Descent          476145 non-null  object
 13  Premise Code            483355 non-null  float64
 14  Premise Description     483354 non-null  object
 15  Address                 484301 non-null  object
 16  Cross Street            461357 non-null  object
 17  Location                484301 non-null  object
dtypes: float64(2), int64(5), object(11)
memory usage: 66.5+ MB
None
```

Command 3: print(df.describe())
This command generates summary statistics for numerical columns in the DataFrame df.

```
print(df.describe())
```

```
          DR Number  Time Occurred        Area ID  Reporting District  \
count  4.843020e+05  484302.000000  484302.000000       484302.000000
mean   1.512948e+08    1355.227532      10.797389         1125.574136
std    3.396622e+07     601.640164       5.829521          584.070363
min    1.001000e+08       1.000000       1.000000          100.000000
25%    1.215053e+08     930.000000       6.000000          647.000000
50%    1.507093e+08    1430.000000      11.000000         1127.000000
75%    1.711077e+08    1820.000000      15.000000         1591.000000
max    2.421108e+08    2359.000000      21.000000         2198.000000

       Crime Code     Victim Age   Premise Code
count    484302.0  414993.000000  483355.000000
mean        997.0      41.138188     102.450040
std           0.0      16.507747      23.866423
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.

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

```
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                85103
Victim Age              69309
Victim Sex               7524
Victim Descent           8157
Premise Code              947
Premise Description       948
Address                     1
Cross Street            22945
Location                    1
dtype: int64
```

ADITYA LALWANI D15C ROLL NO.30

Drop columns that aren't useful:-

columns_to_drop = ['Crime Code Description', 'MO Codes', 'Address', 'Cross Street']

df.drop(columns=columns_to_drop, inplace=True)
print(df.head())

this drops the columns ('Crime Code Description', 'MO Codes', 'Address', and 'Cross Street') from the DataFrame and updates the DataFrame in place and then displays the first five rows of the modified DataFrame.

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

```
   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 that have fewer non-null values.

ADITYA LALWANI D15C ROLL NO.30

Take care of missing data:-

```
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())
```

Numerical missing values in the 'Victim Age' column are filled with the median.
Categorical missing values in the specified columns ('Victim Sex', 'Victim Descent', 'Premise Description', 'Premise Code') are filled with the mode. print(df.isnull().sum()) is used to verify that there are no more missing values.

```
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
```

ADITYA LALWANI D15C ROLL NO.30

Create dummy variables:-

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.

```
df = pd.get_dummies(df, columns=['Area Name', 'Victim Sex', 'Premise Description'], drop_first=True)
print(df.head())
```

```
   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

   Premise Description_TRUCK, COMMERICAL  Premise Description_TUNNEL  \
0                                  False                       False
1                                  False                       False
2                                  False                       False
3                                  False                       False
4                                  False                       False
```

Find out outliers (manually):-

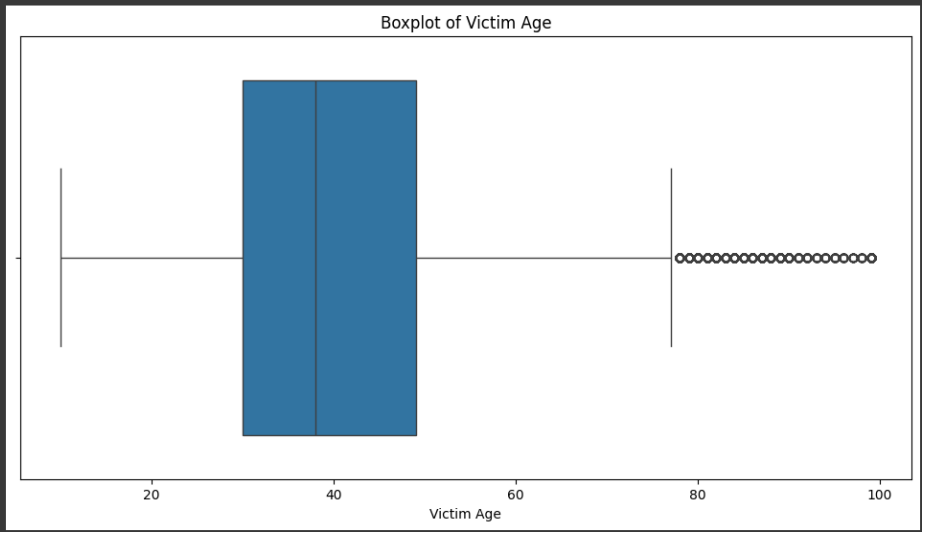
Command 1:

```
plt.figure(figsize=(12,6))
sns.boxplot(x=df['Victim Age'])
plt.title('Boxplot of Victim Age')
plt.show()
```

Use boxplot to visually identify outliers in the 'Victim Age' column by showing the distribution and extreme values.

ADITYA LALWANI D15C ROLL NO.30

```
plt.figure(figsize=(12,6))
sns.boxplot(x=df['Victim Age'])
plt.title('Boxplot of Victim Age')
plt.show()
```



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))

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

```
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))
```

```
Number of Outliers in Victim Age: 12196
```

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 values.

```
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']])
```

```
        Victim Age
100           84.0
101           99.0
141           99.0
146           88.0
152           90.0
...            ...
484182        99.0
484207        99.0
484224        99.0
484232        99.0
484250        99.0

[12196 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' 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'.

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
df['Victim Age Standardized'] = scaler.fit_transform(df[['Victim Age']])
print(df[['Victim Age Standardized', 'Victim Age']])
```

```
        Victim Age Standardized  Victim Age
0                     -1.219863        22.0
1                     -0.697695        30.0
2                     -0.175527        38.0
3                     -1.285134        21.0
4                      0.542454        49.0
...                         ...         ...
484296                -1.024050        25.0
484297                -0.110256        39.0
484298                 0.281370        45.0
484299                -0.893508        27.0
484300                 0.868809        54.0

[484279 rows x 2 columns]
```

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.

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
df['Victim Age Normalized'] = scaler.fit_transform(df[['Victim Age']])
print(df[['Victim Age Normalized', 'Victim Age']])
```

```
        Victim Age Normalized  Victim Age
0                    0.134831        22.0
1                    0.224719        30.0
2                    0.314607        38.0
3                    0.123596        21.0
4                    0.438202        49.0
...                       ...         ...
484296               0.168539        25.0
484297               0.325843        39.0
484298               0.393258        45.0
484299               0.191011        27.0
484300               0.494382        54.0

[484279 rows x 2 columns]
```