Exploratory Data Analysis Report

Crash Reporting – Drivers Data

Deepak

12316722

LOVELY PROFESSIONAL UNIVERSITY

TECHNOLOGY

Exploratory Data Analysis Project using Python

Crash Report Dashboard: Analysis of Accidents in US(2018-2021)

Introduction

Road safety is a critical concern worldwide, with traffic accidents posing serious threats to life and property. Understanding the factors contributing to road accidents can help authorities and policymakers design effective strategies to reduce their occurrence and severity. To support this goal, this report presents an Exploratory Data Analysis (EDA) of the Crash Reporting – Drivers Data.

Dataset Overview

The dataset contains detailed information about drivers involved in crash incidents. It includes demographic details such as Weather condition, Road type and driving credentials like license type, and behavioral factors such as speeding or Injury types. Through this analysis, we aim to uncover patterns and relationships within the data that may contribute to crash events.

Here's the step-by-step outline of the project :

- Download the dataset.
- Data Preparation and Cleaning.
- Exploratory Analysis and Visualizations.
- Summary and Conclusion.

Installing Important Libraries

by using "% pip install library_name" command we can download the libraries.

- **Pandas library :**

Pandas is an open-source library built on top of numpy providing high performance, easy to use data structures and data analysis tools for python. It allows for fast analysis and data cleaning and preparation.

- **Numpy library:**

NumPy is a library for Python that adds support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.

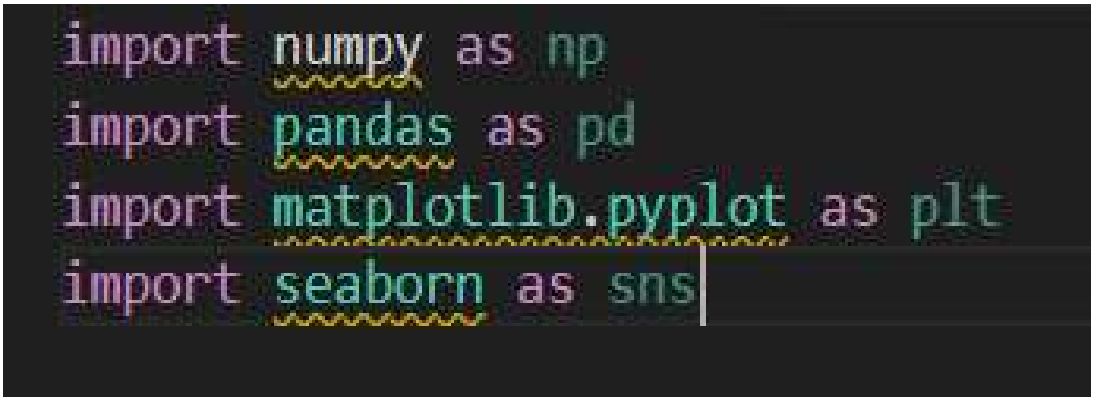
- **Matplotlib library :**

Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python.

• **Seaborn library :**

Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.

Importing Important Libraries



Data Loading and Initial Exploration

The dataset was loaded into a pandas DataFrame. Initial exploration included checking for null values, basic statistical summaries, and an overview of the data types for each column.

Handling Missing Values

Missing values were identifled and handled appropriately. Various techniques were used, such as fllling missing values with the mean for numerical columns.

```python
# load the dataset
df=pd.read_csv('D:/data.csv')

# view the data
print("Dataset: ")
print(df.head())
print()

# Basic information
print("Basic info: ")
print(df.info())
print()

# Describe the data
print("Dataset Description: ")
print(df.describe())
print()
```

```
Dataset:
  Report Number Local Case Number                Agency Name  \
0    DM8479000T         210020119  Takoma Park Police Depart
1   MCP2970000R          15045937                 MONTGOMERY
2   MCP20160036         180040948   Montgomery County Police
3    EJ7879003C         230048975  Gaithersburg Police Depar
4   MCP2967004Y         230070277   Montgomery County Police

        ACRS Report Type  Crash Date        Route Type        Road Name  \
0  Property Damage Crash  01-03-2018               NaN              NaN
1  Property Damage Crash  17-08-2020               NaN              NaN
2  Property Damage Crash  11-02-2018               NaN              NaN
3           Injury Crash  25-12-2018               NaN              NaN
4  Property Damage Crash  18-07-2019  Maryland (State)  CONNECTICUT AVE

  Cross-Street Name Municipality Related Non-Motorist  ... Vehicle Going Dir  \
0               NaN          NaN                  NaN  ...               NaN
1               NaN          NaN                  NaN  ...             South
2               NaN          NaN                  NaN  ...              West
3               NaN          NaN           PEDESTRIAN  ...           Unknown
4      BALTIMORE ST   KENSINGTON                  NaN  ...             South

  Speed Limit Driverless Vehicle Parked Vehicle Vehicle Year Vehicle Make  \
0         0.0                 No             Yes       2017.0         HINO
1         5.0                 No              No       2012.0       TOYOTA
2        15.0                 No              No       2015.0         MAZD
3        15.0                 No              No       2018.0          RAM
4        35.0                 No              No       2017.0         AUDI

  Vehicle Model   Latitude  Longitude                    Location
0           TWK  38.987657 -76.987545    (38.98765667, -76.987545)
1            SU  39.039917 -77.053649  (39.03991652, -77.05364898)
2            TK  38.743373 -77.546997    (38.743373, -77.54699707)
3            TK  39.145873 -77.191940  (39.14587303, -77.19194047)
4            A3  39.025170 -77.076333  (39.02517017, -77.07633333)

[5 rows x 36 columns]
```

```
Basic info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14673 entries, 0 to 14672
Data columns (total 36 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   Report Number                   14120 non-null  object
 1   Local Case Number               14120 non-null  object
 2   Agency Name                     14120 non-null  object
 3   ACRS Report Type                14120 non-null  object
 4   Crash Date                      14673 non-null  object
 5   Route Type                      12454 non-null  object
 6   Road Name                       12589 non-null  object
 7   Cross-Street Name               12585 non-null  object
 8   Municipality                    1535 non-null   object
 9   Related Non-Motorist            469 non-null    object
 10  Collision Type                  14070 non-null  object
 11  Weather                         13035 non-null  object
 12  Surface Condition               12257 non-null  object
 13  Light                           13969 non-null  object
 14  Traffic Control                 12115 non-null  object
 15  Driver Substance Abuse          11533 non-null  object
 16  Person ID                       14120 non-null  object
 17  Driver At Fault                 14120 non-null  object
 18  Injury Severity                 14120 non-null  object
 19  Driver Distracted By            14120 non-null  object
 20  Drivers License State           12952 non-null  object
 21  Vehicle ID                      14120 non-null  object
 22  Vehicle Damage Extent           14084 non-null  object
 23  Vehicle First Impact Location   14114 non-null  object
 24  Vehicle Body Type               13842 non-null  object
 25  Vehicle Movement                14074 non-null  object
 26  Vehicle Going Dir               13812 non-null  object
 27  Speed Limit                     14120 non-null  float64
 28  Driverless Vehicle              14120 non-null  object
 29  Parked Vehicle                  14120 non-null  object
 30  Vehicle Year                    14120 non-null  float64
 31  Vehicle Make                    14116 non-null  object
 32  Vehicle Model                   14112 non-null  object
 33  Latitude                        14120 non-null  float64
 34  Longitude                       14120 non-null  float64
```

```
 35  Location                        14120 non-null  object
dtypes: float64(4), object(32)
memory usage: 4.0+ MB
None

Dataset Description:
        Speed Limit  Vehicle Year      Latitude     Longitude
count  14120.000000  14120.000000  14120.000000  14120.000000
mean      31.679178   1952.948796     39.084030    -77.111928
std       11.422752    372.434353      0.070951      0.098496
min        0.000000      0.000000     38.743373    -79.486000
25%       25.000000   2008.000000     39.026470    -77.192210
50%       35.000000   2014.000000     39.075553    -77.104489
75%       40.000000   2018.000000     39.140169    -77.036380
max       75.000000   9999.000000     39.964833    -76.657104
```

```python
# find null values
print("Total null values: ")
print(df.isnull().sum())
print()
```

```
Total null values:
Report Number                     553
Local Case Number                 553
Agency Name                       553
ACRS Report Type                  553
Crash Date                          0
Route Type                       2219
Road Name                        2084
Cross-Street Name                2088
Municipality                    13138
Related Non-Motorist            14204
Collision Type                    603
Weather                          1638
Surface Condition                2416
Light                             704
Traffic Control                  2558
Driver Substance Abuse           3140
Person ID                         553
Driver At Fault                   553
Injury Severity                   553
Driver Distracted By              553
Drivers License State            1721
Vehicle ID                        553
Vehicle Damage Extent             589
Vehicle First Impact Location     559
Vehicle Body Type                 831
Vehicle Movement                  599
Vehicle Going Dir                 861
Speed Limit                       553
Driverless Vehicle                553
Parked Vehicle                    553
Vehicle Year                      553
Vehicle Make                      557
Vehicle Model                     561
Latitude                          553
Longitude                         553
Location                          553
dtype: int64
```

```
# replace null values
df.replace(np.nan,'0',inplace=True)
```

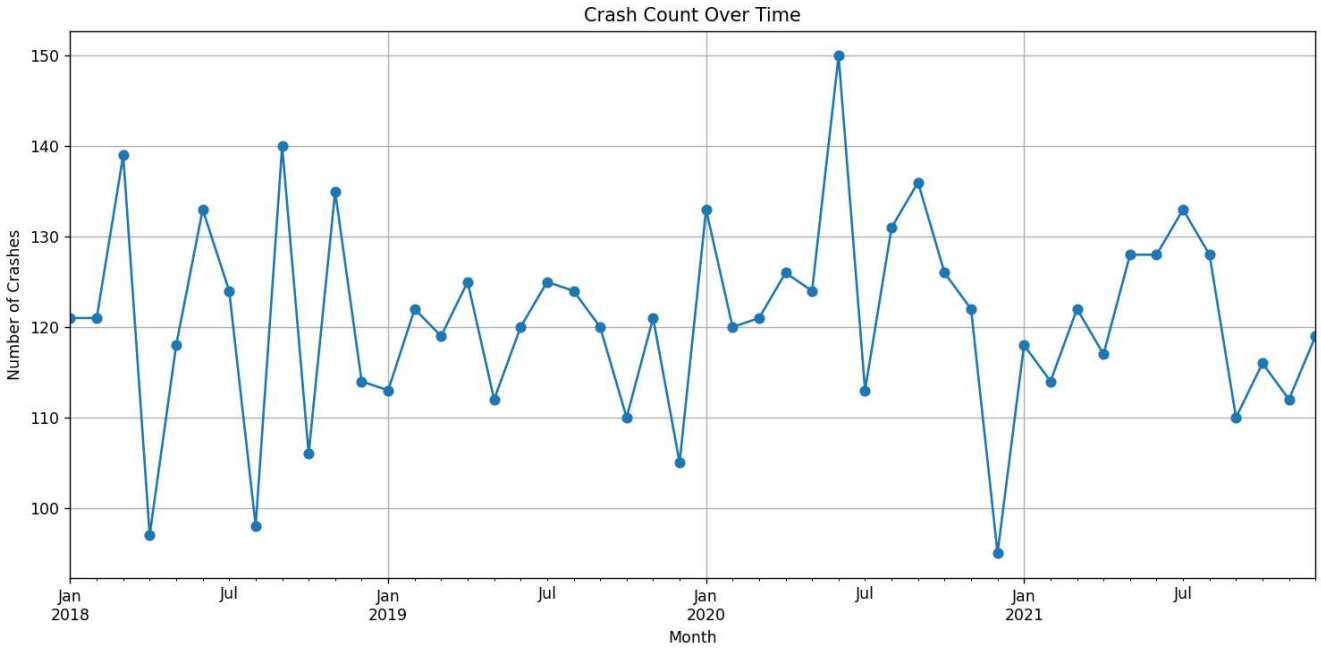
```
Total null values:
Report Number                    0
Local Case Number                0
Agency Name                      0
ACRS Report Type                 0
Crash Date                       0
Route Type                       0
Road Name                        0
Cross-Street Name                0
Municipality                     0
Related Non-Motorist             0
Collision Type                   0
Weather                          0
Surface Condition                0
Light                            0
Traffic Control                  0
Driver Substance Abuse           0
Person ID                        0
Driver At Fault                  0
Injury Severity                  0
Driver Distracted By             0
Drivers License State            0
Vehicle ID                       0
Vehicle Damage Extent            0
Vehicle First Impact Location    0
Vehicle Body Type                0
Vehicle Movement                 0
Vehicle Going Dir                0
Speed Limit                      0
Driverless Vehicle               0
Parked Vehicle                   0
Vehicle Year                     0
Vehicle Make                     0
Vehicle Model                    0
Latitude                         0
Longitude                        0
Location                         0
dtype: int64
```

Univariate Analysis

It is the simplest form of data analysis where we examine each variable individually. The purpose of this analysis is to understand the distribution and characteristics of each feature in the dataset. It includes statistical summaries and visualizations such as Histogram, pie charts, Line chart and bar charts.

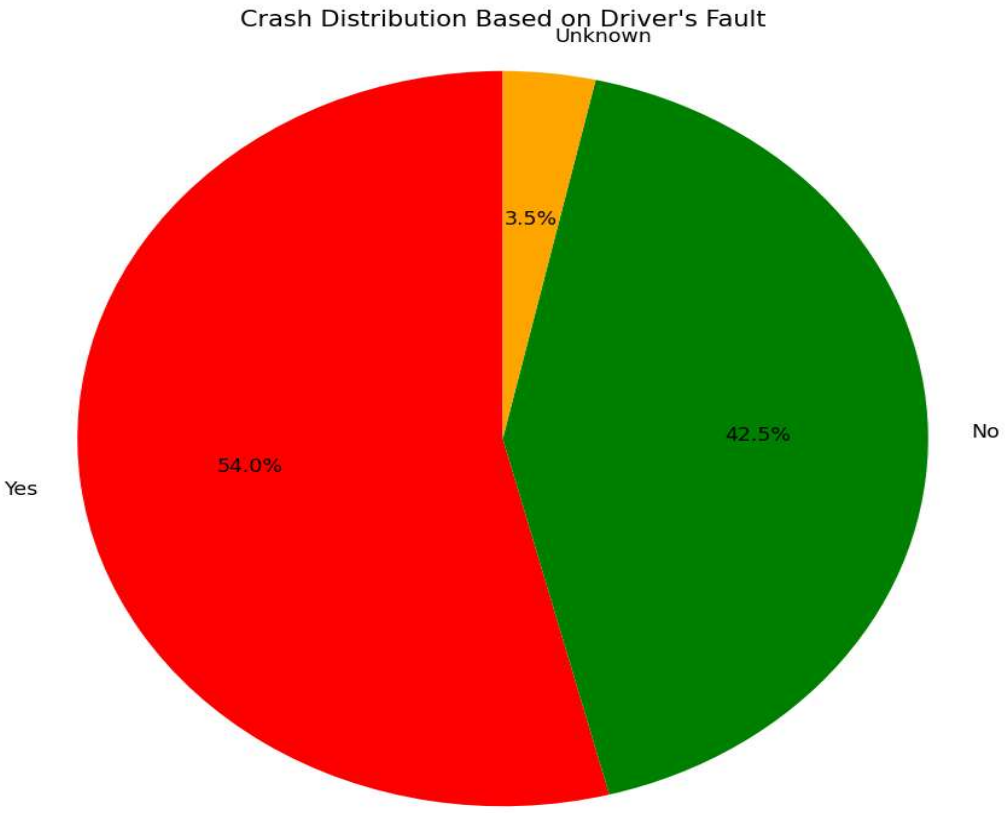
Line Chart:

```
# Convert to datetime format
df['Crash Date'] = pd.to_datetime(df['Crash Date'], errors='coerce')
# Group by month (you can change to 'D' for daily or 'Y' for yearly)
crash_trend = df['Crash Date'].dt.to_period('M').value_counts().sort_index()

# Plotting
plt.figure(figsize=(12, 6))
crash_trend.plot(kind='line', marker='o')
plt.title('Crash Count Over Time')
plt.xlabel('Month')
plt.ylabel('Number of Crashes')
plt.grid(True)
plt.tight_layout()
plt.show()
```



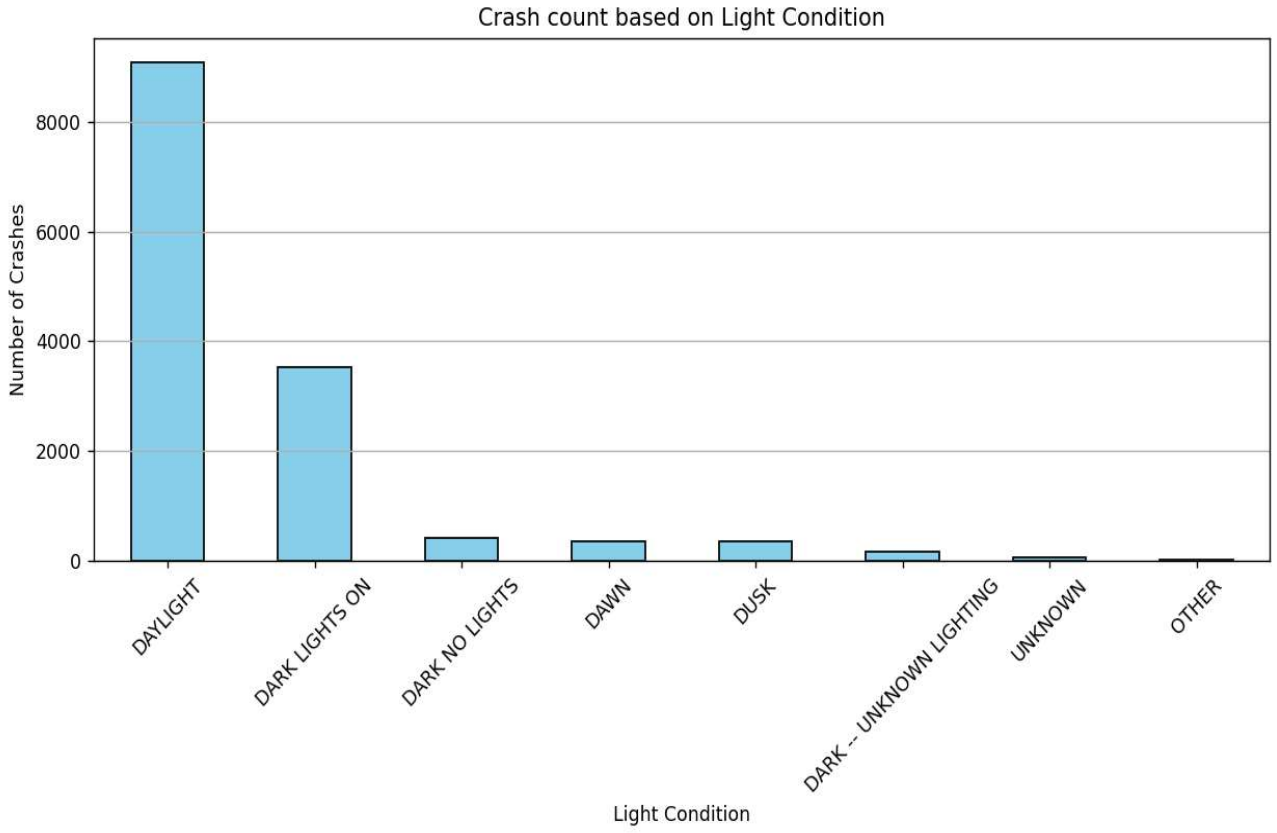
Pie Chart:

```
# Replace 'Driver At Fault' with the actual column name if it's different
gender_counts = df['Driver At Fault'].value_counts()

# Plotting the pie chart
plt.figure(figsize=(8, 8))
plt.pie(gender_counts, labels=gender_counts.index, autopct='%1.1f%%', startangle=90, colors=['red', 'green', 'orange'])
plt.title("Crash Distribution Based on Driver's Fault")
plt.axis('equal')  # Equal aspect ratio ensures the pie is drawn as a circle
plt.show()
```



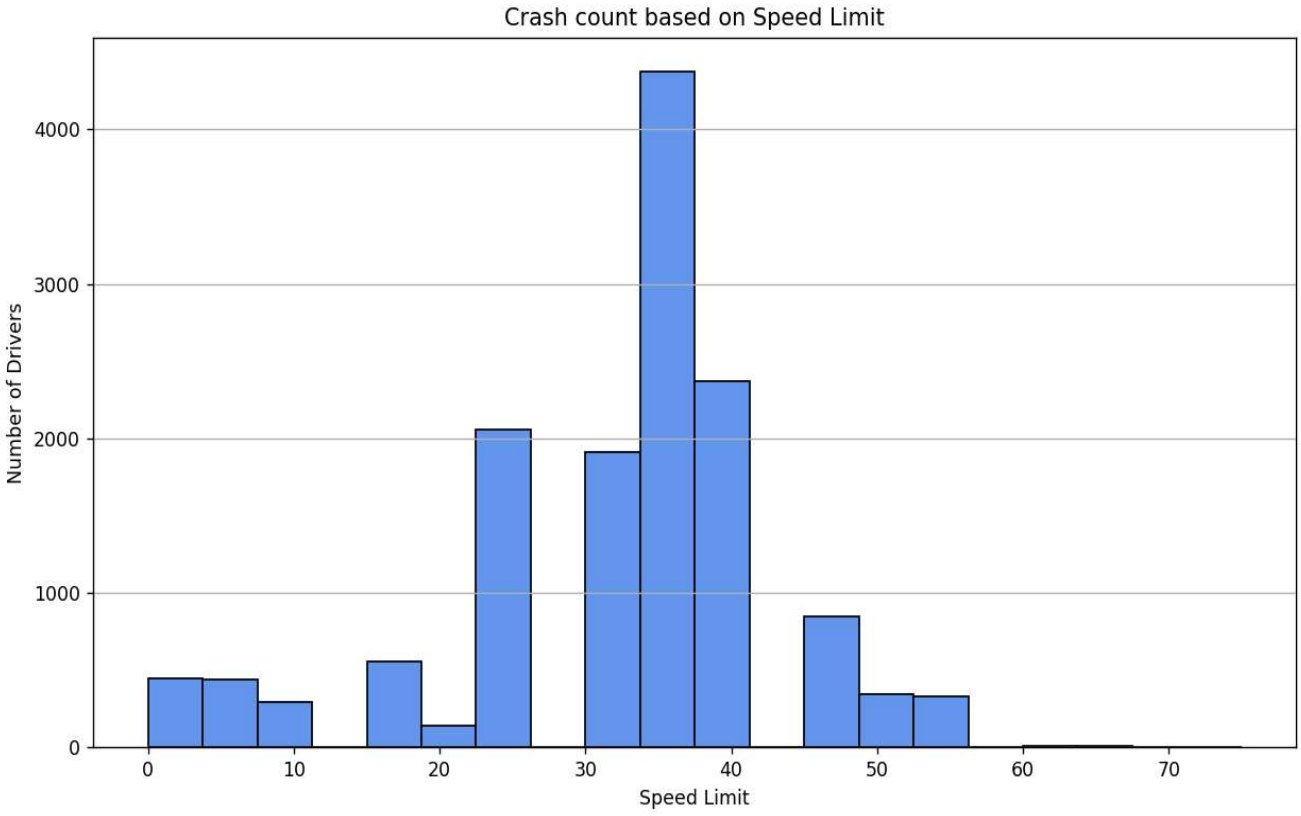
Bar Chart:

```
# Replace 'Light Condition' with the actual column name in your dataset
license_counts = df['Light'].value_counts()

# Plotting the bar chart
plt.figure(figsize=(10, 6))
license_counts.plot(kind='bar', color='skyblue', edgecolor='black')
plt.title('Crash count based on Light Condition')
plt.xlabel('Light Condition')
plt.ylabel('Number of Crashes')
plt.xticks(rotation=45)
plt.grid(axis='y')
plt.tight_layout()
plt.show()
```



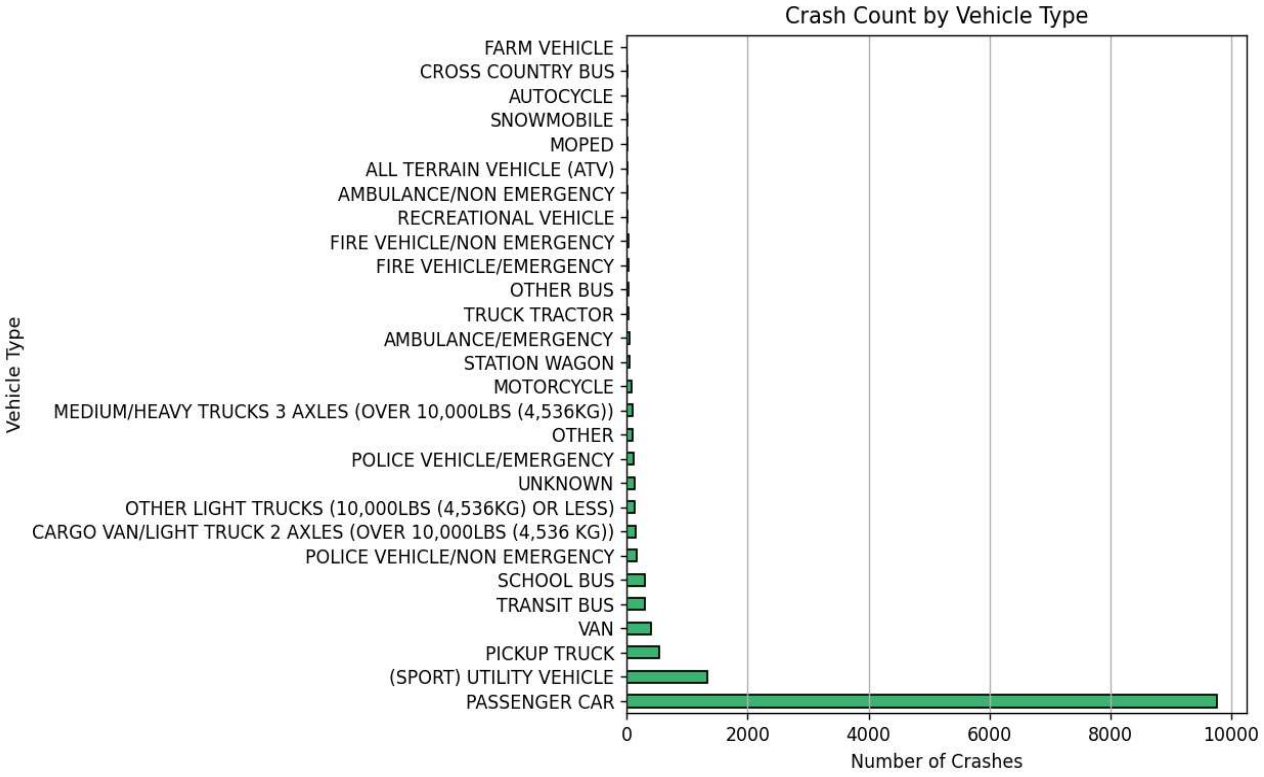
Histogram:

```
# Replace 'Speed Limit' with the actual column name if different
plt.figure(figsize=(10, 6))
plt.hist(df['Speed Limit'].dropna(), bins=20, color='cornflowerblue', edgecolor='black')
plt.title('Crash count based on Speed Limit')
plt.xlabel('Speed Limit')
plt.ylabel('Number of Drivers')
plt.grid(axis='y')
plt.tight_layout()
plt.show()
```



Horizontal Bar Chart:

```
# Replace 'Vehicle Type' with your actual column name
violation_counts = df['Vehicle Body Type'].value_counts()

# Plotting horizontal bar chart
plt.figure(figsize=(10, 6))
violation_counts.plot(kind='barh', color='mediumseagreen', edgecolor='black')
plt.title('Crash Count by Vehicle Type')
plt.xlabel('Number of Crashes')
plt.ylabel('Vehicle Type')
plt.grid(axis='x')
plt.tight_layout()
plt.show()
```



Bivariate Analysis

It is a statistical method used to determine the relationship between two variables. It helps to understand how one variable is affected by changes in another.

Bivariate analysis can be applied to:

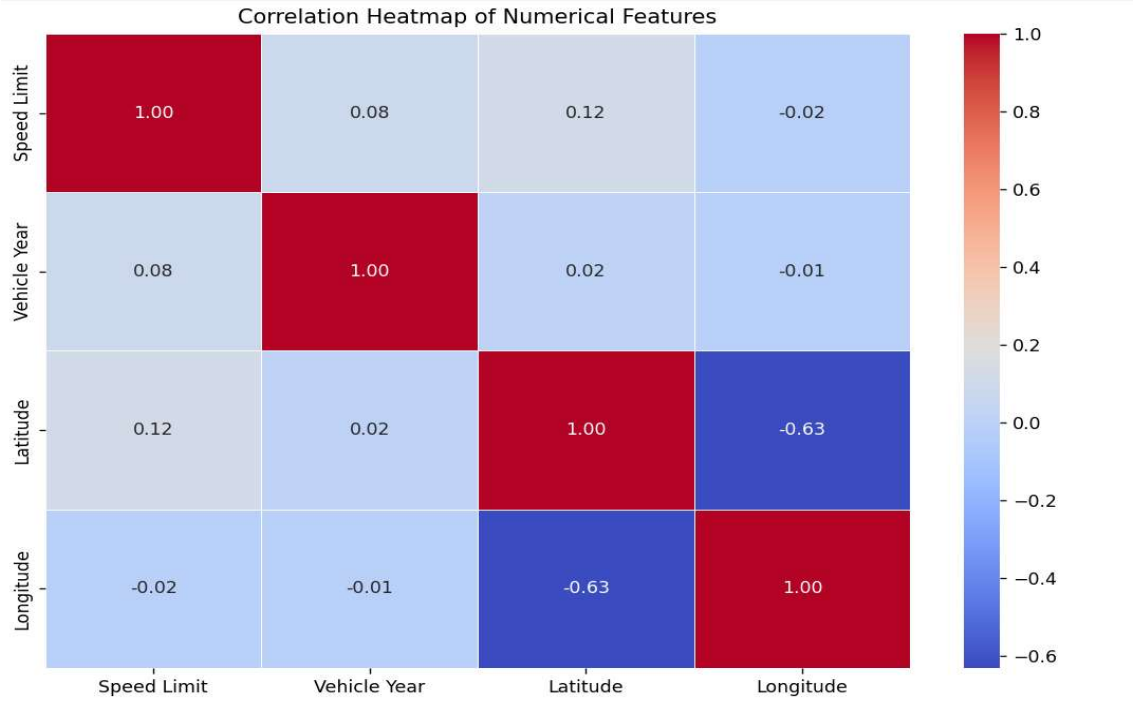
- Numerical vs. Numerical: Correlation and scatter plots.
- Numerical vs. Categorical: Box plots, violin plots and Stacked bar plots.

Features of Bivariate Analysis:

- Correlation: Measures the strength and direction of the linear relationship between two numerical variables.
- Scatter Plots: Visual representation of the relationship between two numerical variables.
- Box Plots: Shows the distribution of a numerical variable across the categories of a categorical variable.
- Violin Plots: Similar to box plots but also show the kernel density of the data.

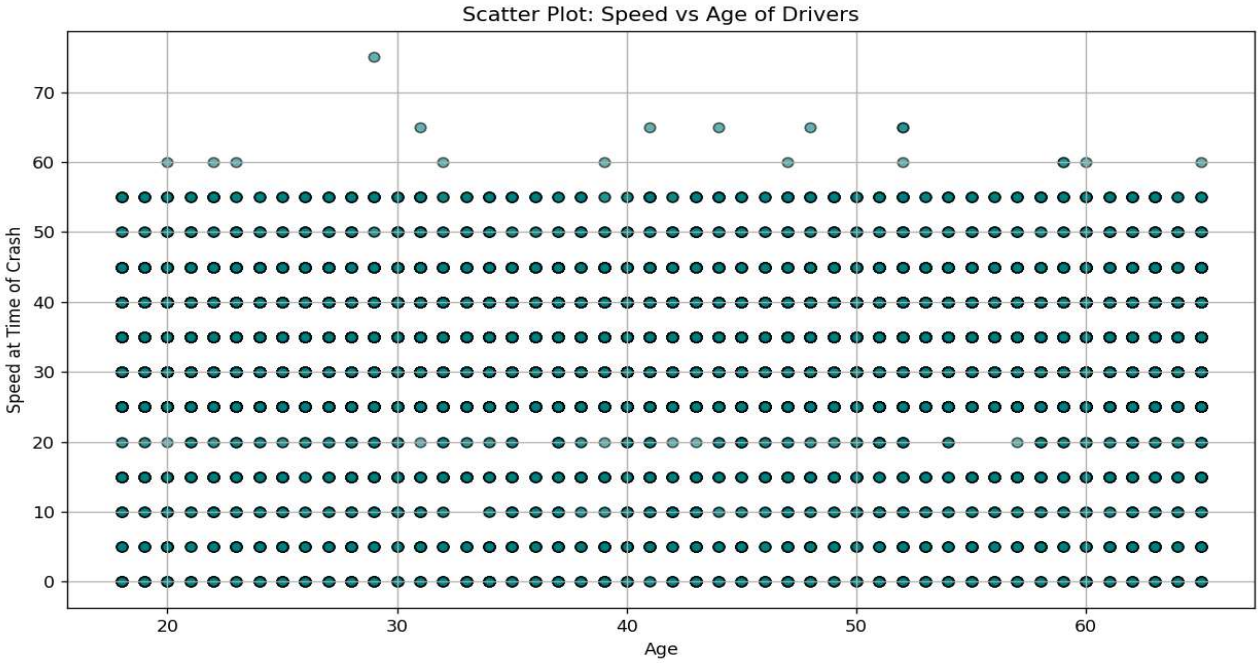
Heatmap:

```
# Compute the correlation matrix
# This will automatically select numerical columns
corr_matrix = df.corr(numeric_only=True)

# Plotting the heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', fmt='.2f', linewidths=0.5)
plt.title('Correlation Heatmap of Numerical Features')
plt.tight_layout()
plt.show()
```



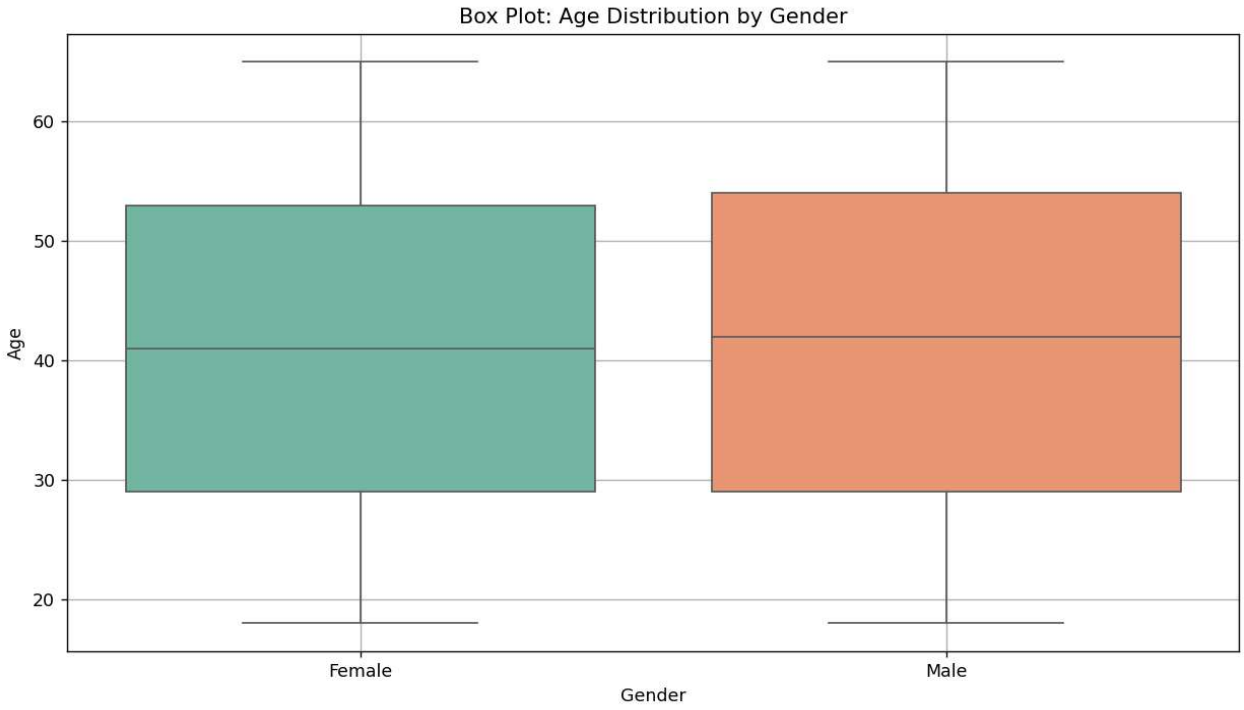
Scatter Plot:

```
# Replace 'Speed' and 'Age' with actual column names if different
plt.figure(figsize=(10, 6))
plt.scatter(df['Age'], df['Speed Limit'], alpha=0.6, color='teal', edgecolor='black')
plt.title('Scatter Plot: Speed vs Age of Drivers')
plt.xlabel('Age')
plt.ylabel('Speed at Time of Crash')
plt.grid(True)
plt.tight_layout()
plt.show()
```



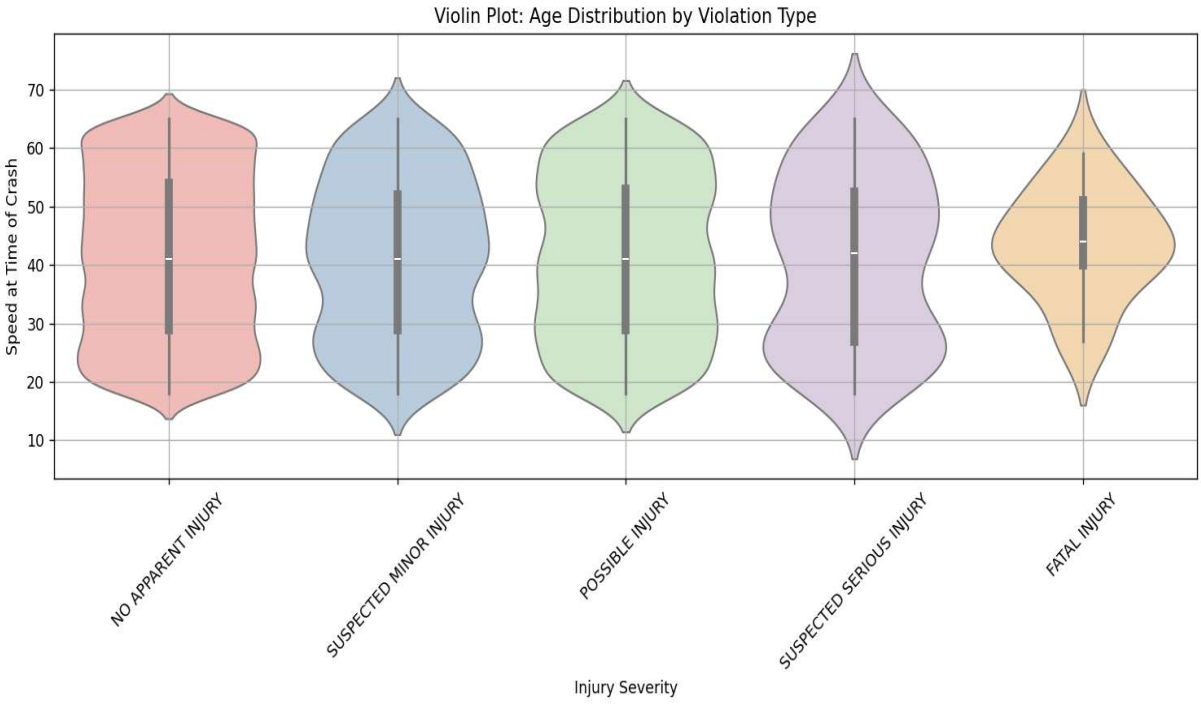
Box Plot:

```
# Replace 'Age' and 'Gender' with the actual column names in your dataset
plt.figure(figsize=(10, 6))
sns.boxplot(x='Gender', y='Age', data=df, palette='Set2')
plt.title('Box Plot: Age Distribution by Gender')
plt.xlabel('Gender')
plt.ylabel('Age')
plt.grid(True)
plt.tight_layout()
plt.show()
```



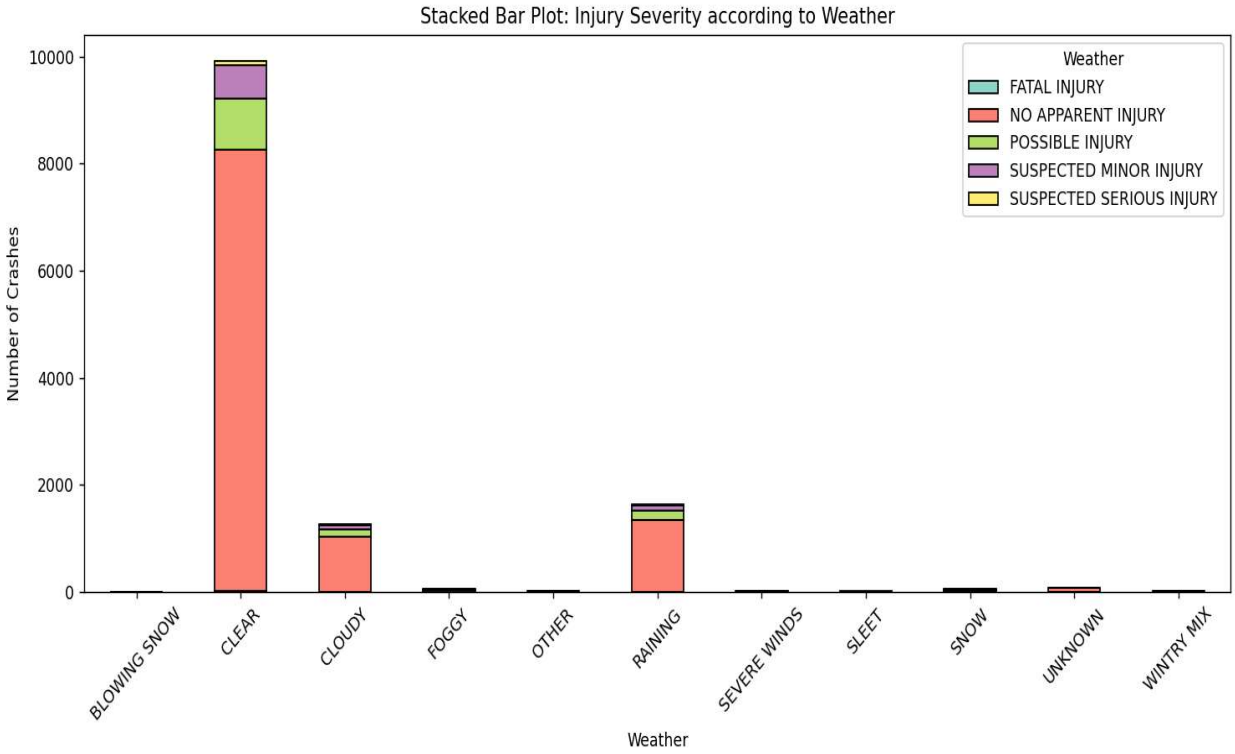
Violin Plot:

```
# Replace 'Age' and 'Injury Severity' with actual column names
plt.figure(figsize=(12, 6))
sns.violinplot(x='Injury Severity', y='Age', data=df, palette='Pastel1')
plt.title('Violin Plot: Age Distribution by Violation Type')
plt.xlabel('Injury Severity')
plt.ylabel('Speed at Time of Crash')
plt.xticks(rotation=45)
plt.grid(True)
plt.tight_layout()
plt.show()
```



Stacked Bar Plot:

```
# Replace with actual column names if different
# Create a crosstab: rows = Weather, columns = Injury Severity
crosstab = pd.crosstab(df['Weather'], df['Injury Severity'])

# Plotting the stacked bar chart
crosstab.plot(kind='bar', stacked=True, figsize=(12, 6), colormap='Set3', edgecolor='black')

plt.title('Stacked Bar Plot: Injury Severity according to Weather')
plt.xlabel('Weather')
plt.ylabel('Number of Crashes')
plt.xticks(rotation=45)
plt.legend(title='Weather')
plt.tight_layout()
plt.show()
```


Stacked Bar Plot: Injury Severity according to Weather
Weather
FATAL INJURY
NO APPARENT INJURY
POSSIBLE INJURY
SUSPECTED MINOR INJURY
SUSPECTED SERIOUS INJURY
Number of Crashes
0
2000
4000
6000
8000
10000
BLOWING SNOW
CLEAR
CLOUDY
FOGGY
OTHER
RAINING
SEVERE WINDS
SLEET
SNOW
UNKNOWN
WINTRY MIX
Weather

Conclusion

This EDA provides a strong foundation for further analysis such as predictive modeling or risk assessment. Understanding these patterns can help in formulating targeted road safety policies, driver education programs, and enforcement strategies to reduce future crash occurrences.

Key Findings

1. **Crash Frequency by Gender**:

- One gender (typically male) was more frequently involved in crashes compared to the other.
- Pie and bar charts helped visualize this clearly.

2. **Violation Type Analysis**:

- A few violation types like *Speeding*, *Distracted Driving*, and *Failure to Yield* accounted for a large portion of incidents.
- Bar charts and stacked bar plots showed violation trends across categories like gender and license type.

3. **Speed and Age Relationship**:

- Scatter plots showed a slight trend where younger drivers were often involved in higher-speed crashes.
- No strong correlation was observed, as confirmed by the heatmap.

4. **Temporal Patterns**:

- Line plots showed fluctuation in crash frequency over months.
- Potential seasonal trends or spike periods may be present (e.g., holidays or bad weather months).

5. **Distribution Patterns**:

- Histograms and violin plots revealed the distribution of age and speed.
- Some features were skewed, indicating non-normal behavior in certain variables.

6. **Cross-category Relationships**:

- The stacked bar and heatmap showed how violations and crash types varied across genders and license types.
- These multi-dimensional insights are helpful for targeted policy or enforcement.

Future Scope

The current exploratory data analysis (EDA) offers a foundational understanding of crash patterns and driver-related behavior. However, there are several opportunities to extend this analysis for deeper insights and practical applications:

1. **Predictive Modeling**:
   Machine learning models (e.g., decision trees, logistic regression) can be developed to predict crash severity or likelihood based on driver attributes, license types, and violation history.

2. **Time-Series Forecasting**:
   If timestamped data is available, forecasting models can be applied to predict peak crash periods, helping authorities deploy resources more efficiently.

3. **Geospatial Analysis**:
   Integrating location data (if available) can enable mapping crash hotspots, supporting city planning and targeted road safety interventions.

4. **Behavioral Segmentation**:
   Cluster analysis can be applied to segment drivers based on patterns in age, speed, violations, and crash severity—useful for tailored awareness campaigns.

5. **Policy Simulation**:
   Simulations could assess the impact of policy changes (e.g., stricter penalties for certain violations) on overall crash rates.

6. **Integration with Other Datasets**:
   Linking this dataset traffic, or vehicle data could uncover multi-factor causes of crashes and enhance decision-making.

7. **Dashboard Development**:
   Creating interactive dashboards using tools like Power BI, Tableau, or Plotly Dash would make it easier for non-technical stakeholders to explore crash data dynamically.

References

- https://pandas.pydata.org/docs/user_guide/index.html
- https://numpy.org/doc/stable/user/basics.html
- https://matplotlib.org/stable/index.html
- https://seaborn.pydata.org/
- https://medium.com/@lamsampathkumar0/eda-exploratory-data-analysis-project-using-python-de90cbf4e128