

US Accidents Analysis

Group: Maitreyi Ahire, Sai Sankar Sriram, Vedant Patil

Introduction:

This Python project focuses on analyzing US Accidents data for the year 2020, utilizing three key datasets: People Involved, Vehicle Involved, and a specialized dataset concentrating on accidents in New York City. In order to implement effective preventative measures and emergency response, it is critical to comprehend the complexities of road accidents, which present serious challenges to public safety. An essential component of our study is the People Involved dataset, which offers information on fatalities, injuries, and other human aspects. To provide a comprehensive knowledge of incidents, the Vehicle Involved dataset provides a complete viewpoint on the different types, circumstances, and contributing variables of vehicles. Given the distinct characteristics of urban transportation, a more specialized analysis that addresses the particular difficulties faced by the city is made possible by the NYC Accidents dataset. The objective of this project is to derive significant patterns and trends from the databases, providing insightful information to both national and local authorities.

Dataset:

The datasets used in this project was taken from the government open dataset website and kaggle:

<https://data.gov/>

The particular datasets used:

<https://www.kaggle.com/datasets/jonbown/us-2020-traffic-accidents>

This dataset includes three datasets out of which two were used: pers_20.csv and veh_20.csv

The NYC data used in the project comes from:

<https://catalog.data.gov/dataset/motor-vehicle-collisions-crashes>

The people and vehicle dataset are structured data in the format of .csv and the NYC dataset is a semi-structured dataset which is in JSON format.

The vehicle_df dataset contains the following columns:

'Case No.' - Unique case id
'Strat Injuries' - Injuries
'Region' - East, West, North, South
'Rural/Urban' - Area of accident
'Month' - Month
'Hour' - Time
'Harmed' - Collateral Damage
'Collision Type' - Type of collision
'Hit & Run' - Yes/No
'Truck?' - Type of truck
'Bus?' - Type of Bus
'Speed of car' - in MPH
'Damage' - Type of damage
'Injury' - Type of injury
'In Vehicle Injury' - People injured
'Speed Limit' - Road speed limit
'Road Condition' - Road condition
'Vehicle type' - Type of vehicle

The person_df contains the following columns:

'Case No.', 'Person No.', 'Region', 'PSU', 'Rural/Urban', 'Month', 'Harmed', 'Collision Type', 'Body Type', 'Car Model Year', 'Impact', 'Age', 'Sex', 'Seat Position', 'Air Bag', 'Alcohol', 'Drugs', 'Hospital', 'Vehicle Type'

The nyc_df dataset includes:

Collision_ID
Crash_Date
Crash_Time
Number_of_Persons_Injured
Number_of_Persons_Killed
Number_of_Pedestrians_Injured

Number_of_Pedestrians_Killed
Number_of_Cyclist_Injured
Number_of_Cyclist_Killed
Number_of_Motorist_Injured
Number_of_Motorist_Killed
Contributing_Factor_Vehicle_1
Contributing_Factor_Vehicle_2
Vehicle_Type_Code_1
Vehicle_Type_Code_2
Year

Data Loading & Preprocessing:

Structured Data (CSV)

The CSV dataset contains information about vehicle accidents, including details about the case number, stratum, region, urbanicity, month, hour, harm event, and collision type. The data is loaded into a DataFrame, and columns are appropriately renamed for clarity.

Semi-Structured Data (JSON)

The JSON dataset is retrieved from the NYC Open Data API, containing information about motor vehicle collisions in New York City. The data is loaded, and relevant columns are selected for further analysis.

Data Processing:

Cleaning and preprocessing the original dataset was necessary in order to conduct a thorough analysis. Finding columns with null or missing values was the first step in correcting incomplete or missing data. Actions including the removal of entire columns or the use of reasonable guesses to fill in the missing data were taken. Moreover, a few columns displayed data in different formats, such as texts or numeric values, therefore standardization was required to keep the dataset consistent. In order to ensure that the data would be accurate and reliable for further analysis, this data cleansing procedure was essential.

Description of Data analysis and data cleaning:

Data Cleaning:

Initially we are reading the content of csv file into our python program using the read_csv function. Then we are filtering out our necessary columns and creating a new data frame.

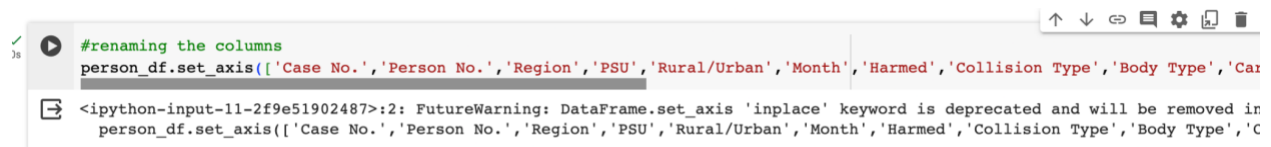
```
[7] #reading the csv file containing people's information
df_raw_per = pd.read_csv('/content/drive/MyDrive/final project scripting/Datasets/pers_20.csv', sep=",", encoding = 'ISO-8859-1')

#filtering out necessary column for analysis
person_df = df_raw_per[['CASENUM', 'PER_NO', 'REGIONNAME', 'PSU_VAR', 'URBANICITYNAME', 'MONTHNAME', 'HARM_EVNAME', 'MAN_COLLNAME', 'I
person_df.columns.to_list()
```



Next we are renaming the columns to understand better and smoothly analyze our data.

```
#renaming the columns
person_df.set_axis(['Case No.', 'Person No.', 'Region', 'PSU', 'Rural/Urban', 'Month', 'Harmed', 'Collision Type', 'Body Type', 'Car
<ipython-input-11-2f9e51902487>:2: FutureWarning: DataFrame.set_axis 'inplace' keyword is deprecated and will be removed in
person_df.set_axis(['Case No.', 'Person No.', 'Region', 'PSU', 'Rural/Urban', 'Month', 'Harmed', 'Collision Type', 'Body Type', 'C
```



To understand what type of data we are working with, we use the head function to explore the first few rows.

```
#exploring the first few columns of person data
person_df.head()
```

	Case No.	Person No.	Region	PSU	Rural/Urban	Month	Harmed	Collision Type	Body Type	Car Model Year	Impact	Age	Sex	Seat Position	A
0	202002121240	1	West (MT, ID, WA, OR, CA, NV, NM, AZ, UT, CO, ...)	20	Rural Area	January	Motor Vehicle In-Transport	Angle	4-door sedan, hardtop	2018.0	8 Clock Point	61	Female	Front Seat, Left Side	D
1	202002121240	1	West (MT, ID, WA, OR, CA, NV, NM, AZ, UT, CO, ...)	20	Rural Area	January	Motor Vehicle In-Transport	Angle	Light Pickup	2007.0	12 Clock Point	26	Male	Front Seat, Left Side	D
			South												

To begin our data cleaning, we first check if the data contains any null values using the `isna()` function.

```
#checking for null values in the dataset
null_count = person_df.isna().sum()
print(null_count)
```

```
Case No.      0
Person No.    0
Region        0
PSU           0
Rural/Urban   0
Month         0
Harmed        0
Collision Type 0
Body Type     5077
Car Model Year 5077
Impact       5077
```

We noticed that a few columns contain null values. To fix the issue we are replacing the null values with 'Unspecified' using the `fillna()` function, so that our data doesn't have any null value.

```
▶ #replacing the null value with Unspecified to clean the data
person_df['Body Type'].fillna('Unspecified', inplace=True)
person_df['Car Model Year'].fillna('Unspecified', inplace=True)
person_df['Impact'].fillna('Unspecified', inplace=True)
person_df['Vehicle Type'].fillna('Unspecified', inplace=True)
```

Once the null values are fixed and columns have been appropriately filtered we can begin our data exploration.

These steps were repeated for the vehicle_df dataframe.

For the nyc_df dataframe, first the columns were selected to filter out.

```
▶ columns_to_keep = ['COLLISION_ID', 'CRASH DATE', 'CRASH TIME', 'NUMBER OF PERSONS INJURED',
                    'NUMBER OF PERSONS KILLED', 'NUMBER OF PEDESTRIANS INJURED',
                    'NUMBER OF PEDESTRIANS KILLED', 'NUMBER OF CYCLIST INJURED',
                    'NUMBER OF CYCLIST KILLED', 'NUMBER OF MOTORIST INJURED',
                    'NUMBER OF MOTORIST KILLED', 'CONTRIBUTING FACTOR VEHICLE 1',
                    'CONTRIBUTING FACTOR VEHICLE 2',
                    'VEHICLE TYPE CODE 1', 'VEHICLE TYPE CODE 2']

nyc_df = nyc_df[columns_to_keep]
nyc_df.head(10)
```

Then, the head() function was used to check the first few rows of the data and make sure it is the filtered data required.

Data Exploration:

We begin our data exploration by trying to understand the data types of the data we are working with.

```
#exploring the data types present in the dataset
print(person_df.dtypes)
print(vehicle_df.dtypes)
```

```
Case No.          int64
Person No.        int64
Region            object
PSU               int64
Rural/Urban       object
Month             object
Harmed            object
Collision Type     object
Body Type         object
Car Model Year    object
Impact            object
Age              int64
Sex              object
Seat Position     object
Air Bag           object
Alcohol           object
Drugs             object
--          --
```

We explore the first few rows of the cleaned data using the head function.

```
#exploring the data set
person_df.head()
```

	Case No.	Person No.	Region	PSU	Rural/Urban	Month	Harmed	Collision Type	Body Type	Car Model Year	Impact	Age	Sex	Seat Position
0	202002121240	1	West (MT, ID, WA, OR, CA, NV, NM, AZ, UT, CO, ...)	20	Rural Area	January	Motor Vehicle In-Transport	Angle	4-door sedan, hardtop	2018.0	8 Clock Point	61	Female	Front Seat, Left Side
1	202002121240	1	West (MT, ID, WA, OR, CA, NV, NM, AZ, UT, CO, ...)	20	Rural Area	January	Motor Vehicle In-Transport	Angle	Light Pickup	2007.0	12 Clock Point	26	Male	Front Seat, Left Side
			South											

Next we try to find the summary statistics of each column to understand the measure of data present in each column using the describe() function.

```
#Checking the summary statistics of data
person_df.describe(include='all')
```

	Case No.	Person No.	Region	PSU	Rural/Urban	Month	Harmed	Collision Type	Body Type	Car Model Year	Impact
count	1.319620e+05	131962.000000	131962	131962.000000	131962	131962	131962	131962	131962	131962.0	131962
unique	NaN	NaN	4	NaN	2	12	56	11	66	73.0	26
top	NaN	NaN	South (MD, DE, DC, WV, VA, KY, TN, NC, SC, GA,...	NaN	Urban Area	October	Motor Vehicle In-Transport	Front-to-Rear	4-door sedan, hardtop	2017.0	12 Clock Point
freq	NaN	NaN	71024	NaN	100574	14994	100007	40777	43008	10119.0	52383
mean	2.020026e+11	1.360232	NaN	52.376927	NaN	NaN	NaN	NaN	NaN	NaN	NaN

These steps were repeated for the vehicle_df dataframe.

For the nyc_df dataframe, first the columns were selected to filter out.

```
columns_to_keep = ['COLLISION_ID', 'CRASH DATE', 'CRASH TIME', 'NUMBER OF PERSONS INJURED',
                   'NUMBER OF PERSONS KILLED', 'NUMBER OF PEDESTRIANS INJURED',
                   'NUMBER OF PEDESTRIANS KILLED', 'NUMBER OF CYCLIST INJURED',
                   'NUMBER OF CYCLIST KILLED', 'NUMBER OF MOTORIST INJURED',
                   'NUMBER OF MOTORIST KILLED', 'CONTRIBUTING FACTOR VEHICLE 1',
                   'CONTRIBUTING FACTOR VEHICLE 2',
                   'VEHICLE TYPE CODE 1', 'VEHICLE TYPE CODE 2']

nyc_df = nyc_df[columns_to_keep]

nyc_df.head(10)
```

Then, the head() function was used to check the first few rows of the data and make sure it is the filtered data required.

Then the data was filtered out to match our other datasets that were from the year 2020. For this we converted the Crash Date column to datetime datatype. We then separated out the year from the date and added it to the new column. Then using this column, we overwrite the nyc_df to just include incidents from the year 2020.


```

# As we just want to focus on data for 2020, we are filtering and just keeping the data for year 2022 in nyc_df

# Converting 'CRASH DATE' to datetime format
nyc_df['CRASH DATE'] = pd.to_datetime(nyc_df['CRASH DATE'])

# Creating a new column for the year
nyc_df['YEAR'] = nyc_df['CRASH DATE'].dt.year

# Filtering the dataset to keep only the rows for the year 2020
nyc_df = nyc_df[nyc_df['YEAR'] == 2020]

nyc_df

```

After this, the null values were checked and fixed using the fillna() function.

```

#checking for null values

nyc_null_count = nyc_df.isna().sum()
print(nyc_null_count)

```

COLLISION_ID	0
CRASH DATE	0
CRASH TIME	0
NUMBER OF PERSONS INJURED	0
NUMBER OF PERSONS KILLED	0
NUMBER OF PEDESTRIANS INJURED	0
NUMBER OF PEDESTRIANS KILLED	0
NUMBER OF CYCLIST INJURED	0
NUMBER OF CYCLIST KILLED	0
NUMBER OF MOTORIST INJURED	0
NUMBER OF MOTORIST KILLED	0
CONTRIBUTING FACTOR VEHICLE 1	512
CONTRIBUTING FACTOR VEHICLE 2	24624
VEHICLE TYPE CODE 1	1072
VEHICLE TYPE CODE 2	34128
YEAR	0

```

dtype: int64

# replacing missing values with 'Unspecified' value
nyc_df['CONTRIBUTING FACTOR VEHICLE 1'].fillna('Unspecified', inplace=True)
nyc_df['CONTRIBUTING FACTOR VEHICLE 2'].fillna('Unspecified', inplace=True)
nyc_df['VEHICLE TYPE CODE 1'].fillna('Unspecified', inplace=True)
nyc_df['VEHICLE TYPE CODE 2'].fillna('Unspecified', inplace=True)

#checking for missing values again
nyc_null_count = nyc_df.isna().sum()
print(nyc_null_count)

```

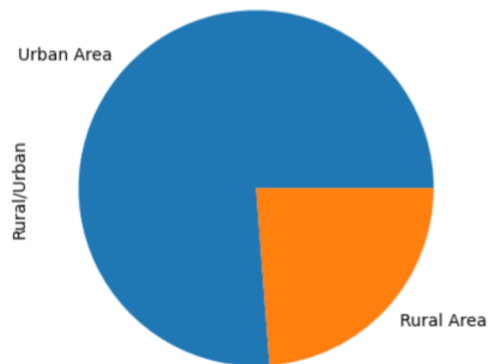
Method of Analysis:

The methods of analysis used in this case involve data analysis and visualization techniques to answer the following questions.

- 1) Which type of cities are most affected by road accidents in the US?

```
#calculating accident based on city and plotting pie chart to visualize
city = person_df['Rural/Urban'].value_counts().nlargest(5)
print(city)
city.plot.pie()
```

```
Urban Area    100574
Rural Area    31388
Name: Rural/Urban, dtype: int64
<Axes: ylabel='Rural/Urban'>
```



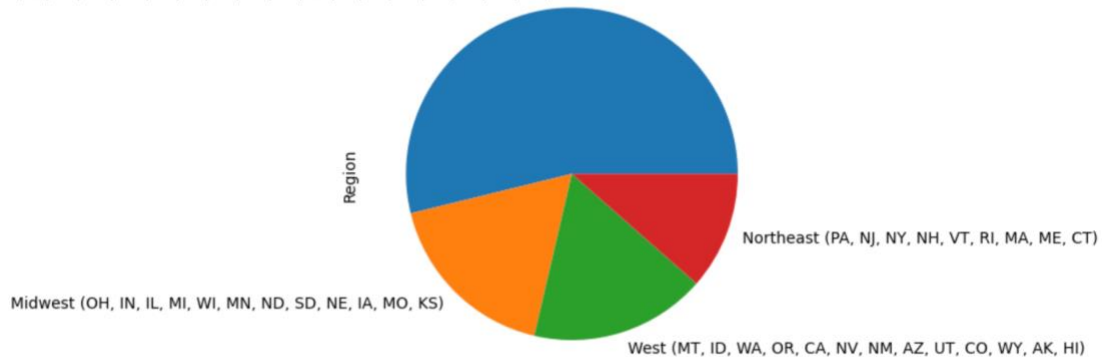
To analyze the cities that have most accidents we use the value count function on the rural/urban column. Once the series is generated, we then use a pie chart to visualize our findings.

2) How are road accidents spread across different regions in the US?

```
#Calculating the road accidents based on regions
Region=person_df['Region'].value_counts()
Region.plot.pie()
```

```
<Axes: ylabel='Region'>
```

South (MD, DE, DC, WV, VA, KY, TN, NC, SC, GA, FL, AL, MS, LA, AR, OK, TX)

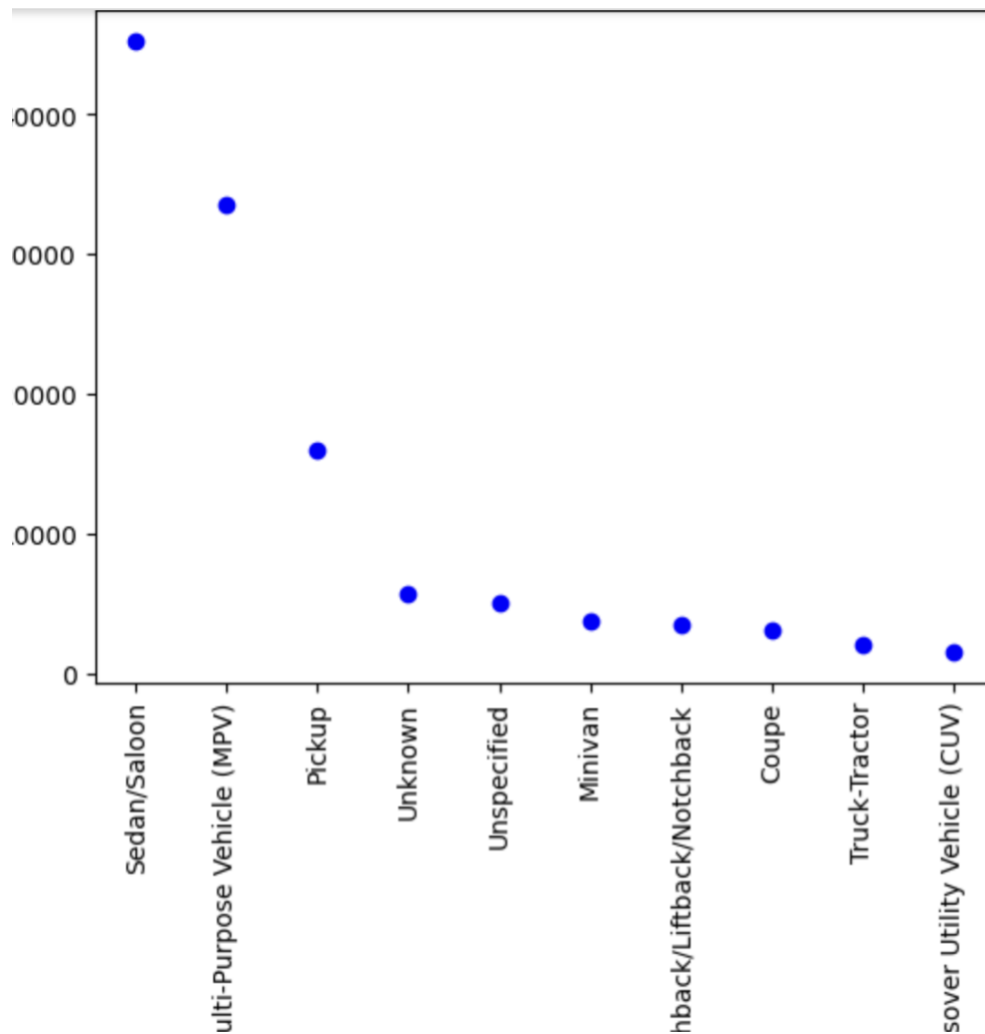


To explore the regions that have most accidents we use the value count function on the rural/urban column. Once the series is generated, we then use a pie chart to visualize our findings.

3) Which type of vehicles are most involved in road accidents?

```
#grouping the data based on vehicle type
group=person_df.groupby('Vehicle Type')['Case No.'].count().sort_values(ascending=False)

#printing the TOP 10 most affected vehicle types
T10= group.head(10)
plt.xticks(rotation=90)
plt.scatter(T10.index,T10.values, color='blue', marker='o', label='Scatter Plot')
```



To find which vehicle types are most involved in road accidents, we first group by the data based on vehicle type and count the number of cases involved in each group.

This gives us a value count of how many accidents have happened for each vehicle type. Next, we use the scatter plot to visualize the top 10 most affected vehicles in our findings.

4) Which age groups are most involved in road accidents?

```
#storing the age column into a new variable
age=person_df['Age']

#creating function to create age groups
def categorize_age(age):
    if age <= 10:
        return '0-10 years'
    elif 11 <= age <= 25:
        return '11-25 years'
    elif 26 <= age <= 40:
        return '26-40 years'
    elif 41 <= age <= 60:
        return '41-60 years'
    elif 61 <= age <= 75:
        return '61-75 years'
    else:
        return '75+ years'

#creating new column to store the age groups
person_df['age_group']=person_df['Age'].apply(categorize_age)

#calculating the accidents based on each age group
group_age=person_df.groupby('age_group')['Case No.'].count()

print(group_age)

colors = ['red', 'green', 'blue', 'orange', 'black', 'purple']

# Create a bar chart
plt.bar(group_age.index, group_age.values, color=colors)

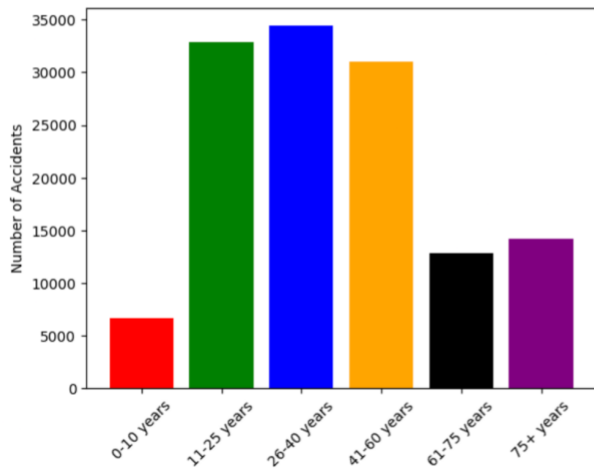
plt.xticks(rotation=45)
# Set labels and title
plt.xlabel('Age Group')
plt.ylabel('Number of Accidents')

# Display the plot
plt.show()
```

```

age_group
0-10 years      6633
11-25 years    32845
26-40 years    34414
41-60 years    30975
61-75 years    12850
75+ years      14245
Name: Case No., dtype: int64

```



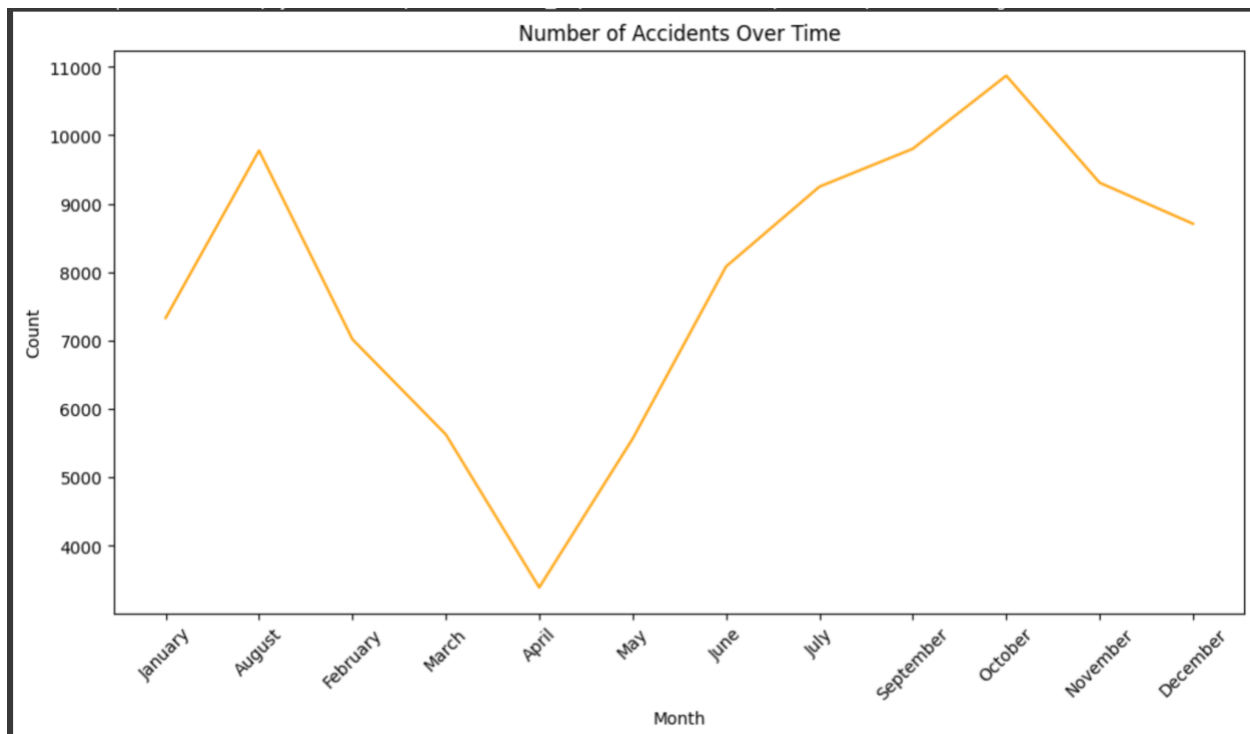
To find out the age groups that are most affected in accidents, we first segregate the data into various age groups. We developed a function which will assign the age group according to the age criteria and then stored it as a new column in our data set. Once an age group was assigned, we used the group by function to calculate the number of accidents involved in each age group and then visualized the same using a bar chart.

5) What is the trend of accidents over time?

```

#Creating a line graph using seaborn & matplotlib with the columns Month and count of Caseno
plt.figure(figsize=(12, 6))
sns.lineplot(x='Month', y='Case No.', data=vehicle_df, estimator='count', ci=None, color='orange')
plt.title('Number of Accidents Over Time')
plt.xlabel('Month')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.show()

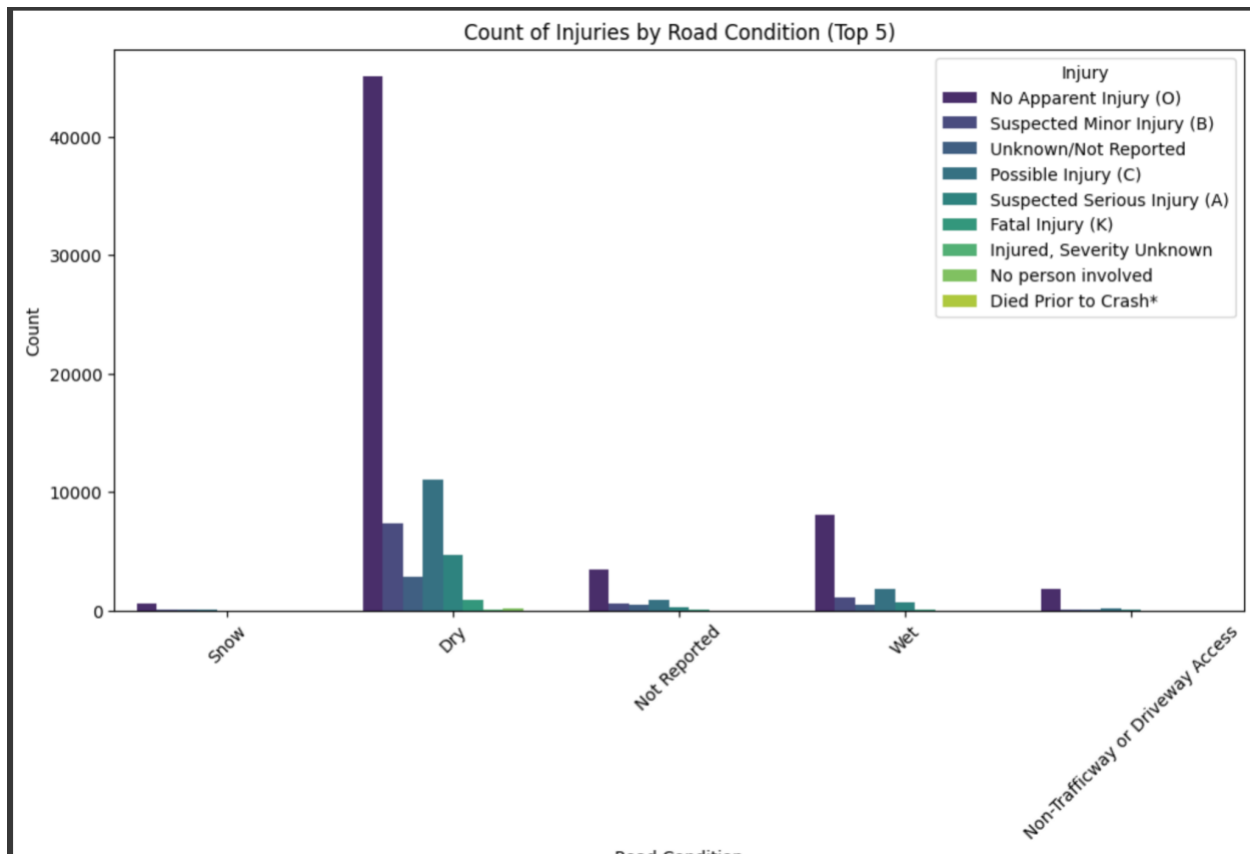
```



To achieve this analysis, a simple comparison of the month column and the count of Case No column was done. Seaborn and matplotlib was used to plot the findings in a line chart.

6) Which type of road conditions lead to the most accidents?

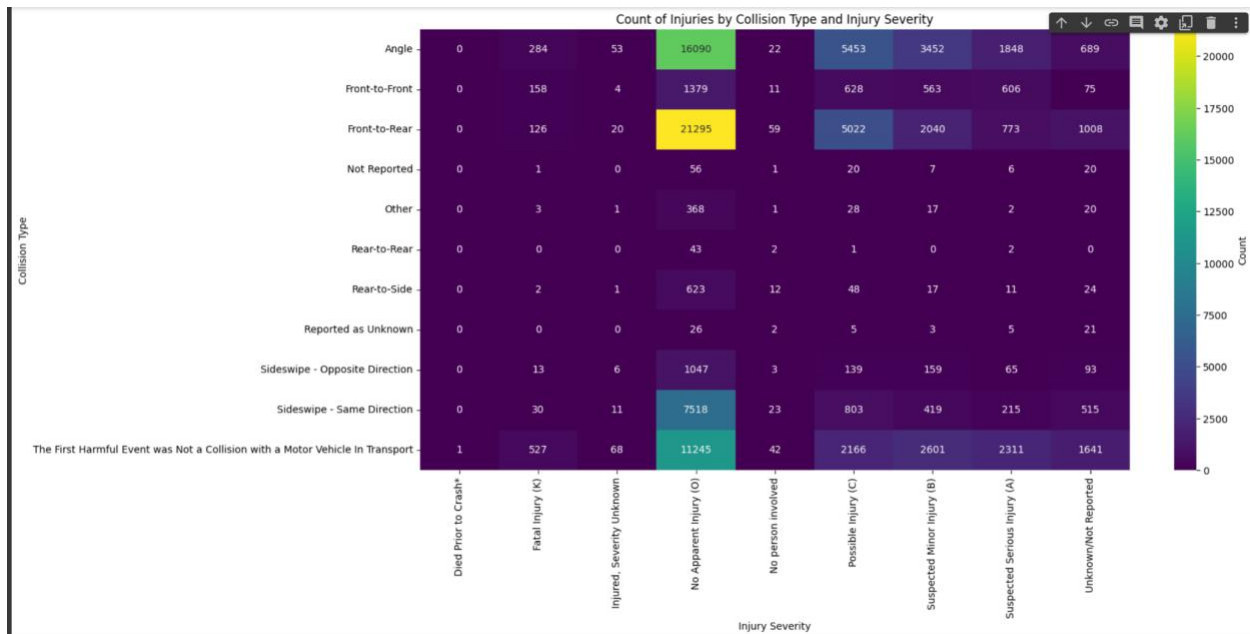
```
#Fetching out the top 5 road conditions
top_5_conditions = vehicle_df['Road Condition'].value_counts().head(5).index
#Filtering out the data for just those top 5 conditions
filtered_df = vehicle_df[vehicle_df['Road Condition'].isin(top_5_conditions)]
#Plotting the data in a bar graph
plt.figure(figsize=(12, 6))
sns.countplot(x='Road Condition', hue='Injury', data=filtered_df, palette='viridis')
plt.title('Count of Injuries by Road Condition (Top 5)')
plt.xlabel('Road Condition')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.show()
```



Here, the top 5 road conditions were taken out to filter the data based on those. Then a filtered_df was created to plot these metrics. This was necessary as there were a large number of road conditions unique values.

7) How does the injury relate to the type of crash?

```
#Plotting the Collision type column and the Injury column in a heatmap
plt.figure(figsize=(16, 8))
#Aggregating values for the matrix
collision_injury_matrix = pd.crosstab(vehicle_df['Collision Type'], vehicle_df['Injury'])
sns.heatmap(collision_injury_matrix, annot=True, cmap='viridis', fmt='d', cbar_kws={'label': 'Count'})
plt.title('Count of Injuries by Collision Type and Injury Severity')
plt.xlabel('Injury Severity')
plt.ylabel('Collision Type')
plt.show()
```



Here to create a heatmap, first a matrix was created using the `crosstab()` function with the Collision type and Injury type columns. Then seaborn was used to plot the heatmap with these two columns showing the relation between the type of collision and the type of injury sustained. A legend at the side helps us guide through the heatmap and makes the analysis easy.

8) What are the primary contributing factors to vehicle collisions in New York City?


```
def plot_top_contributing_factors(column_name, title):
    plt.figure(figsize=(10, 6))

    # Filtering out 'Unspecified' values
    filtered_df = nyc_df[nyc_df[column_name] != 'Unspecified']

    # Getting the top 5 contributing factors
    top_factors = filtered_df[column_name].value_counts().nlargest(5)

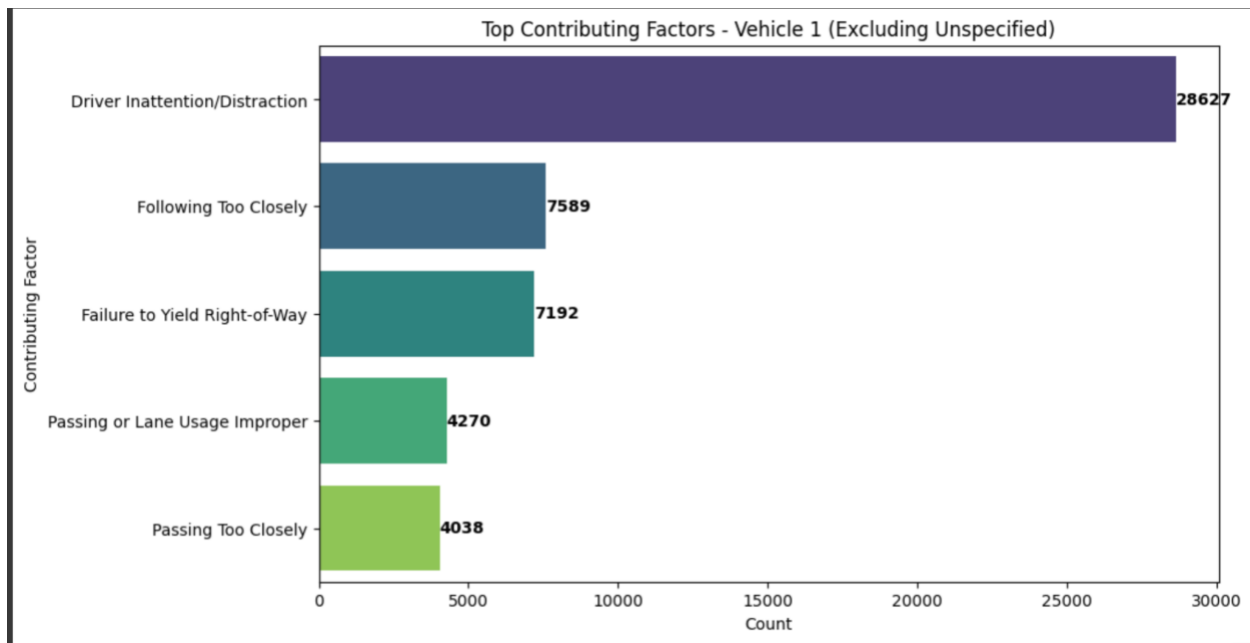
    # Bar plot with a custom color palette
    sns.barplot(y=top_factors.index, x=top_factors.values, palette='viridis')

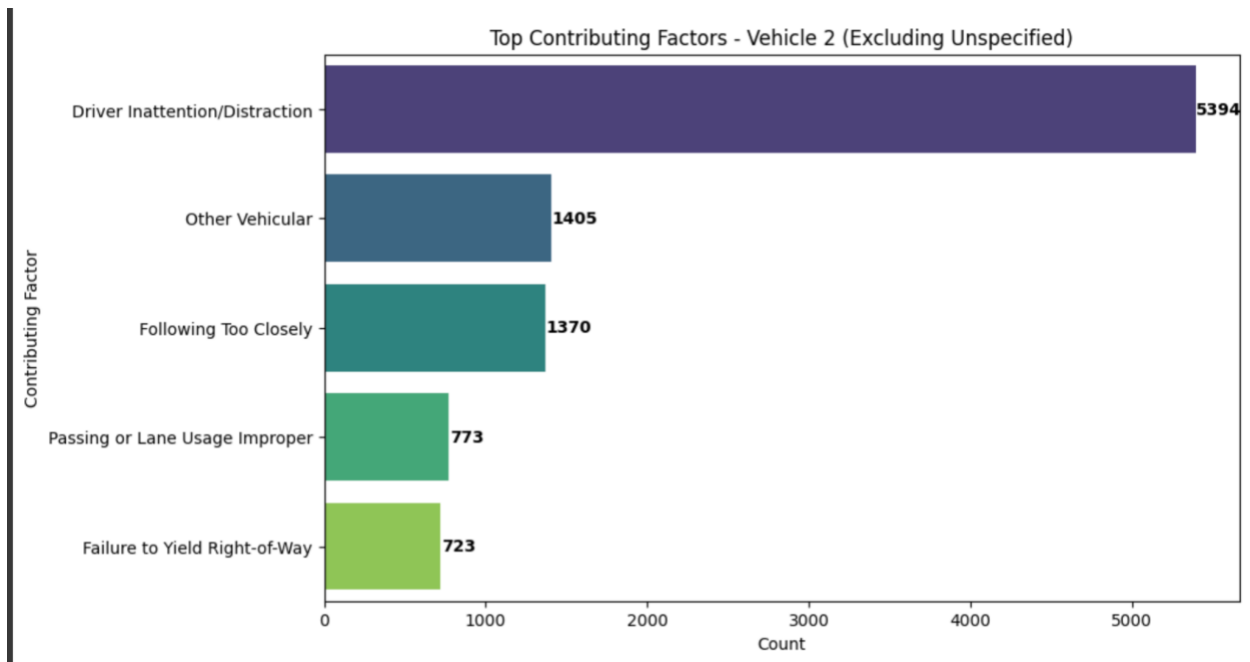
    # Adding data labels to the bars
    for i, v in enumerate(top_factors.values):
        plt.text(v + 5, i, str(v), color='black', va='center', fontweight='bold')

    plt.title(title)
    plt.xlabel('Count')
    plt.ylabel('Contributing Factor')
    plt.show()

# Bar plot for the top 5 contributing factors for Vehicle 1
plot_top_contributing_factors('CONTRIBUTING FACTOR VEHICLE 1', 'Top Contributing Factors - Vehicle 1 (Excluding Unspecified)')

# Bar plot for the top 5 contributing factors for Vehicle 2
plot_top_contributing_factors('CONTRIBUTING FACTOR VEHICLE 2', 'Top Contributing Factors - Vehicle 2 (Excluding Unspecified)')
```





To achieve this analysis, we defined a function passing two parameters: the column name and the title for the graph. We assigned these as they will change with the vehicle. Then the 'unspecified' values were filtered out and the top 5 factors were determined. These factors were then plotted in the function. This function was then called two times as per the column name giving us the two graphs.

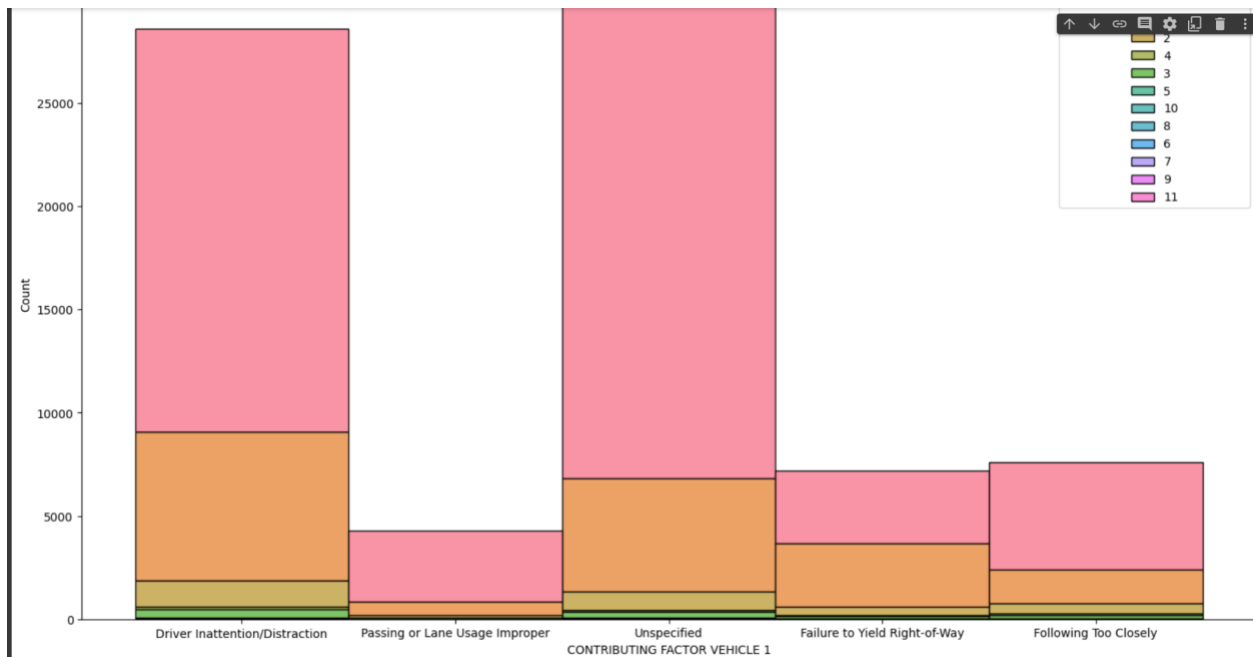
9) What is the distribution of collision severity (in terms of the number of persons injured) across the top 5 contributing factors in vehicle collisions?

```
# Determining the top 5 contributing factors
top_factors = nyc_df['CONTRIBUTING FACTOR VEHICLE 1'].value_counts().nlargest(5).index

# Filtering the DataFrame for only the top 5 contributing factors
filtered_df = nyc_df[nyc_df['CONTRIBUTING FACTOR VEHICLE 1'].isin(top_factors)]

# Setting the figure size to increase the width
plt.figure(figsize=(18, 10)) # Adjust the width (10) and height (6) as needed

# Creating a stacked histogram
sns.histplot(data=filtered_df, x='CONTRIBUTING FACTOR VEHICLE 1', hue='NUMBER OF PERSONS INJURED', multiple='stack')
plt.title('Distribution of Collision Severity by Top 5 Contributing Factors')
plt.show()
```



For this analysis, a multi-stacked histogram was used. First the top five factors were determined and the data related to those factors was filtered. This graph also shows the severity of the accidents and the people injured.

- 10) How does the distribution of the top N vehicle types arriving after crash at hospitals vary, and which experience the highest volume of cases for these specific vehicle types?

```
# Merging person_df and vehicle_df on 'Case No.'
combined_df = pd.merge(person_df, vehicle_df, on='Case No.')
combined_df.columns

top_n_vehicle_types = 5

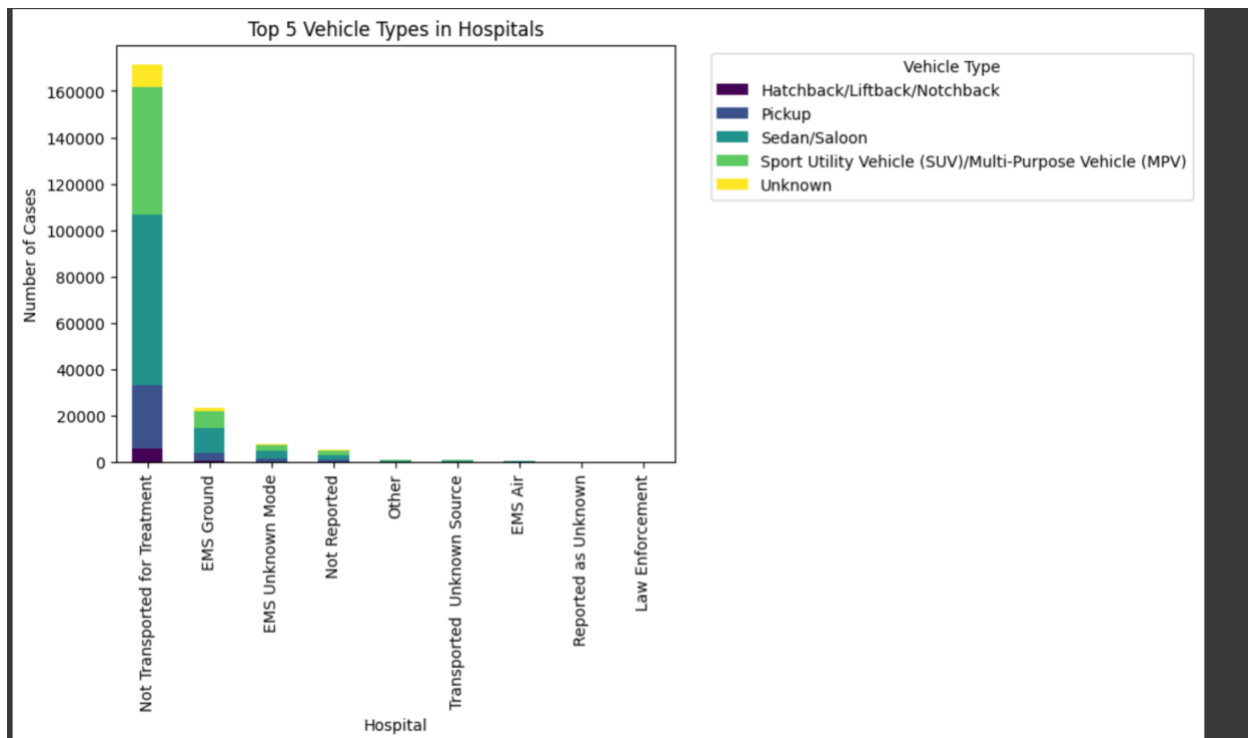
# Calculating the top N most frequent vehicle types
top_vehicle_types = combined_df['Vehicle type'].value_counts().nlargest(top_n_vehicle_types).index

# Filtering the DataFrame for the top N vehicle types
filtered_df = combined_df[combined_df['Vehicle type'].isin(top_vehicle_types)]

# Creating a table of counts for each hospital and vehicle type
hospital_vehicle_counts = filtered_df.groupby('Hospital')['Vehicle type'].value_counts().unstack(fill_value=0)

# Sorting the values for better visualization
hospital_vehicle_counts = hospital_vehicle_counts.reindex(hospital_vehicle_counts.sum(axis=1).sort_values(ascending=False).index)

# Plotting
plt.figure(figsize=(12, 6))
hospital_vehicle_counts.plot(kind='bar', stacked=True, colormap='viridis')
plt.title(f'Top {top_n_vehicle_types} Vehicle Types in Hospitals')
plt.xlabel('Hospital')
plt.ylabel('Number of Cases')
plt.legend(title='Vehicle Type', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.show()
```



For this, we had to merge the two datasets used: `person_df` and `vehicle_df`. We used the `merge()` function on the `Case No` column which was a common column between these two datasets. For this analysis we considered the top 5 vehicle types with accidents. After filtering out the data for these vehicles, we created a table for the hospital counts i.e type of care. This table was then plotted using stacked bar chart which we can see above.

Description of program:

The program consists of several steps including data loading, cleaning, exploration, analysis and visualizations. Each step is clearly marked with comments and headings. Each analysis starts with a question and is followed by the steps taken to achieve that analysis. There are two types of datasets used in this project: structured and semi-structured.

Conclusion of Analysis:

- The analysis provides various factors which are associated with road accidents. The findings and trends found in this analysis could be used by the government to take precautionary measures in future road constructions and change the rules.
- The Findings also highlight the severity of damage incurred in an accident, which can be taken into consideration by the road safety organizations to be prepared and have sufficient supply based on the impactful zones.
- Daily commuters can also use this analysis to be cautious while driving in particular areas and decide on the type of vehicle which they find most comfortable and secure to travel.
- Road organizations can also use this data to analyze the road types which have the most occurrence of accidents and help provide alternative solutions which may avoid crowding and accidents.

Tasks Assigned:

Maitreyi Ahire	Loading, Cleaning and Analysis of the NYC data
Sai Sankar Sriram	Loading, Cleaning and Analysis of the People data
Vedant Patil	Loading, Cleaning and Analysis of the Vehicle data

Group	Merging and Analysis of person_df and vehicle_df Report creation
-------	---