Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

CHENNAI

WINTER SEMESTER 2024 - 2025

CSE3040-EXPLORATORY DATA ANALYSIS

REVIEW - 2

A COMPREHENSIVE ANALYSIS ON TRAFFIC ACCIDENTS

TEAM MEMBERS	(TEAM R)
LAKSHARAA A S	23MIA1053
AMRITHA A	23MIA1067
NITISH RAM J	23MIA1095

GUIDED BY

ASNATH VICTY PHAMILA Y

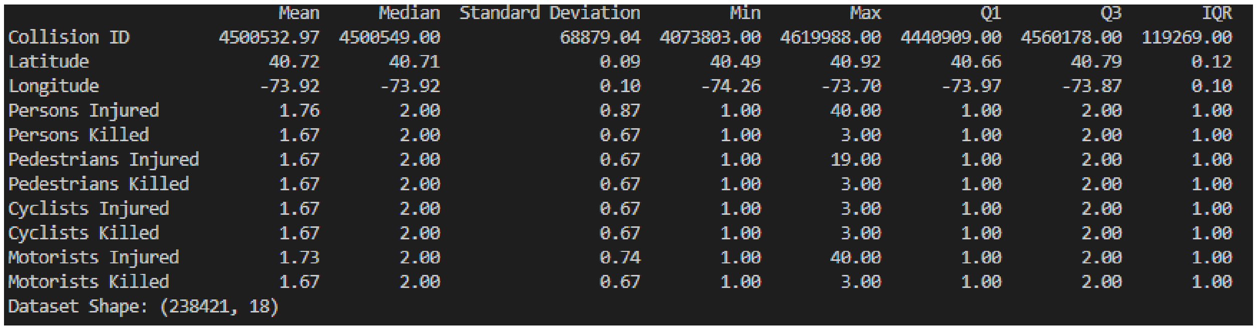
SUBMITTED ON

01/04/2025

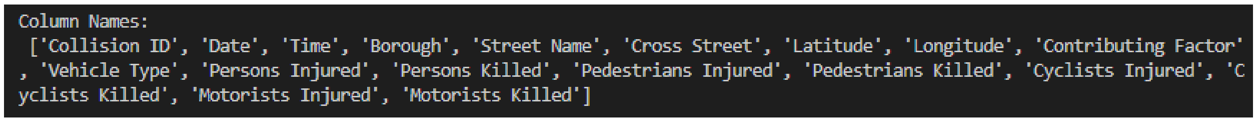
1. **BASIC STATISTICAL ANALYSIS:**

```
import pandas as pd

# Load the dataset
df =
pd.read_csv(r"C:\Users\asoor\Downloads\NYC_Collisions_TimeseriesReady.csv")

# Select only numerical columns
numerical_cols = df.select_dtypes(include=['int64', 'float64']).columns

# Calculate basic statistics
stats = pd.DataFrame(index=numerical_cols)

stats['Mean'] = df[numerical_cols].mean()
stats['Median'] = df[numerical_cols].median()
stats['Standard Deviation'] = df[numerical_cols].std()
stats['Min'] = df[numerical_cols].min()
stats['Max'] = df[numerical_cols].max()
stats['Q1'] = df[numerical_cols].quantile(0.25)
stats['Q3'] = df[numerical_cols].quantile(0.75)
stats['IQR'] = stats['Q3'] - stats['Q1']

# Display the result
print(stats.round(2))

# Convert 'Date' column to datetime for any future time-based analysis
df["Date"] = pd.to_datetime(df["Date"])

# Dataset summary
print("Dataset Shape:", df.shape)
print("\nColumn Names:\n", df.columns.tolist())

# Check for missing values
print("\nMissing Values:\n", df.isnull().sum())

# Descriptive statistics for numerical features
print("\nDescriptive Statistics (Numerical Columns):\n", df.describe())

# Value counts for selected categorical columns
print("\nValue Counts - Borough:\n", df['Borough'].value_counts())
print("\nValue Counts - Contributing Factor (Top 10):\n", df['Contributing
Factor'].value_counts().head(10))
print("\nValue Counts - Vehicle Type (Top 10):\n", df['Vehicle
Type'].value_counts().head(10))
```

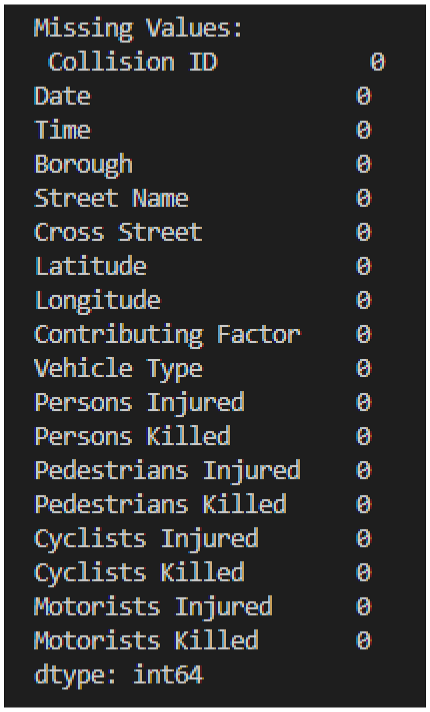
```python
# Correlation matrix of numerical columns
numerical_cols = df.select_dtypes(include=['int64', 'float64']).columns
correlation_matrix = df[numerical_cols].corr()
print("\nCorrelation Matrix:\n", correlation_matrix)
```



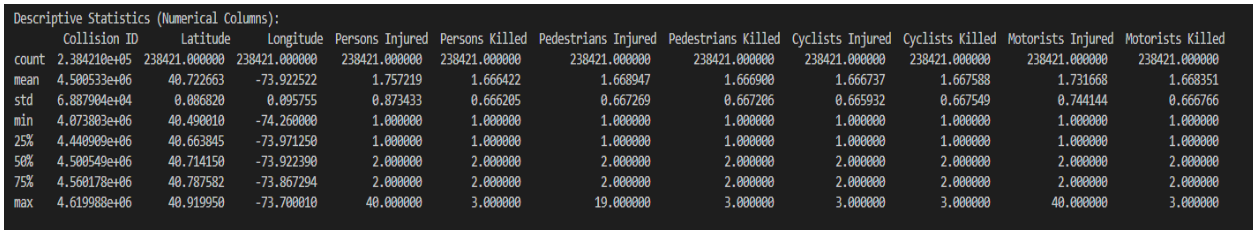
```
                          Mean      Median  Standard Deviation         Min         Max          Q1          Q3        IQR
Collision ID        4500532.97  4500549.00            68879.04  4073803.00  4619988.00  4440909.00  4560178.00  119269.00
Latitude                 40.72       40.71                0.09       40.49       40.92       40.66       40.79       0.12
Longitude               -73.92      -73.92                0.10      -74.26      -73.70      -73.97      -73.87       0.10
Persons Injured           1.76        2.00                0.87        1.00       40.00        1.00        2.00       1.00
Persons Killed            1.67        2.00                0.67        1.00        3.00        1.00        2.00       1.00
Pedestrians Injured       1.67        2.00                0.67        1.00       19.00        1.00        2.00       1.00
Pedestrians Killed        1.67        2.00                0.67        1.00        3.00        1.00        2.00       1.00
Cyclists Injured          1.67        2.00                0.67        1.00        3.00        1.00        2.00       1.00
Cyclists Killed           1.67        2.00                0.67        1.00        3.00        1.00        2.00       1.00
Motorists Injured         1.73        2.00                0.74        1.00       40.00        1.00        2.00       1.00
Motorists Killed          1.67        2.00                0.67        1.00        3.00        1.00        2.00       1.00
Dataset Shape: (238421, 18)
```



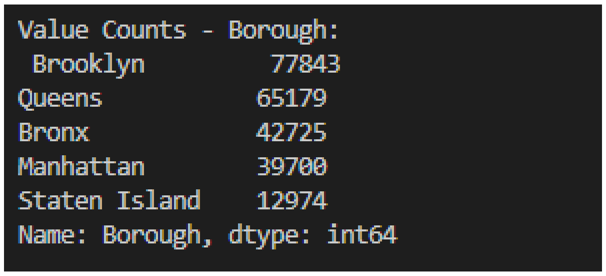
```
Column Names:
 ['Collision ID', 'Date', 'Time', 'Borough', 'Street Name', 'Cross Street', 'Latitude', 'Longitude', 'Contributing Factor'
, 'Vehicle Type', 'Persons Injured', 'Persons Killed', 'Pedestrians Injured', 'Pedestrians Killed', 'Cyclists Injured', 'C
yclists Killed', 'Motorists Injured', 'Motorists Killed']
```



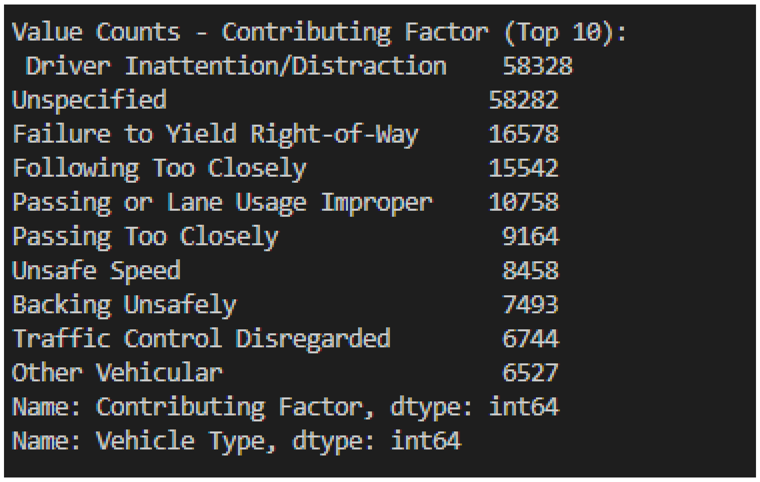
```
Missing Values:
 Collision ID          0
Date                   0
Time                   0
Borough                0
Street Name            0
Cross Street           0
Latitude               0
Longitude              0
Contributing Factor    0
Vehicle Type           0
Persons Injured        0
Persons Killed         0
Pedestrians Injured    0
Pedestrians Killed     0
Cyclists Injured       0
Cyclists Killed        0
Motorists Injured      0
Motorists Killed       0
dtype: int64
```



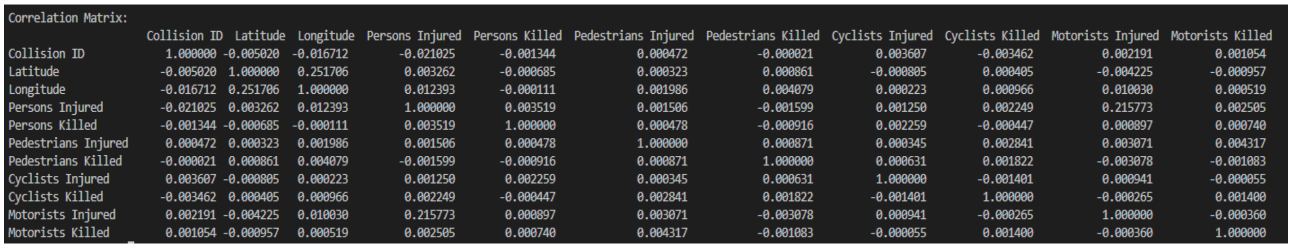
```
Descriptive Statistics (Numerical Columns):
       Collision ID     Latitude    Longitude  Persons Injured  Persons Killed  Pedestrians Injured  Pedestrians Killed  Cyclists Injured  Cyclists Killed  Motorists Injured  Motorists Killed
count  2.384210e+05  238421.000000  238421.000000    238421.000000   238421.000000        238421.000000       238421.000000     238421.000000    238421.000000      238421.000000     238421.000000
mean   4.500533e+06      40.722663     -73.922522         1.757219        1.666422             1.668947            1.666900          1.666737         1.667588           1.731668          1.668351
std    6.887904e+04       0.086820       0.095755         0.873433        0.666205             0.667269            0.667206          0.665932         0.667549           0.744144          0.666766
min    4.073803e+06      40.490010     -74.260000         1.000000        1.000000             1.000000            1.000000          1.000000         1.000000           1.000000          1.000000
25%    4.440909e+06      40.663845     -73.971250         1.000000        1.000000             1.000000            1.000000          1.000000         1.000000           1.000000          1.000000
50%    4.500549e+06      40.714150     -73.922390         2.000000        2.000000             2.000000            2.000000          2.000000         2.000000           2.000000          2.000000
75%    4.560178e+06      40.787582     -73.867294         2.000000        2.000000             2.000000            2.000000          2.000000         2.000000           2.000000          2.000000
max    4.619988e+06      40.919950     -73.700010        40.000000        3.000000            19.000000            3.000000          3.000000         3.000000          40.000000          3.000000
```



```
Value Counts - Borough:
 Brooklyn         77843
Queens           65179
Bronx            42725
Manhattan        39700
Staten Island    12974
Name: Borough, dtype: int64
```



```
Value Counts - Contributing Factor (Top 10):
 Driver Inattention/Distraction    58328
Unspecified                       58282
Failure to Yield Right-of-Way     16578
Following Too Closely             15542
Passing or Lane Usage Improper    10758
Passing Too Closely                9164
Unsafe Speed                       8458
Backing Unsafely                   7493
Traffic Control Disregarded        6744
Other Vehicular                    6527
Name: Contributing Factor, dtype: int64
Name: Vehicle Type, dtype: int64
```



```
Correlation Matrix:
                     Collision ID  Latitude  Longitude  Persons Injured  Persons Killed  Pedestrians Injured  Pedestrians Killed  Cyclists Injured  Cyclists Killed  Motorists Injured  Motorists Killed
Collision ID             1.000000 -0.005020  -0.016712        -0.021025       -0.001344             0.000472           -0.000021          0.003607        -0.003462           0.002191          0.001054
Latitude                -0.005020  1.000000   0.251706         0.003262       -0.000685             0.000323            0.000861         -0.000805         0.000405          -0.004225         -0.000957
Longitude               -0.016712  0.251706   1.000000         0.012393       -0.000111             0.001986            0.004079          0.000223         0.000966           0.010030          0.000519
Persons Injured         -0.021025  0.003262   0.012393         1.000000        0.003519             0.001506           -0.001599          0.001250         0.002249           0.215773          0.002505
Persons Killed          -0.001344 -0.000685  -0.000111         0.003519        1.000000             0.000478           -0.000916          0.002259        -0.000447           0.000897          0.000740
Pedestrians Injured      0.000472  0.000323   0.001986         0.001506        0.000478             1.000000            0.000871          0.000345         0.002841           0.003071          0.004317
Pedestrians Killed      -0.000021  0.000861   0.004079        -0.001599       -0.000916             0.000871            1.000000          0.000631         0.001822          -0.003078         -0.001083
Cyclists Injured         0.003607 -0.000805   0.000223         0.001250        0.002259             0.000345            0.000631          1.000000        -0.001401           0.000941         -0.000055
Cyclists Killed         -0.003462  0.000405   0.000966         0.002249       -0.000447             0.002841            0.001822         -0.001401         1.000000          -0.000265          0.001400
Motorists Injured        0.002191 -0.004225   0.010030         0.215773        0.000897             0.003071           -0.003078          0.000941        -0.000265           1.000000         -0.000360
Motorists Killed         0.001054 -0.000957   0.000519         0.002505        0.000740             0.004317           -0.001083         -0.000055         0.001400          -0.000360          1.000000
```

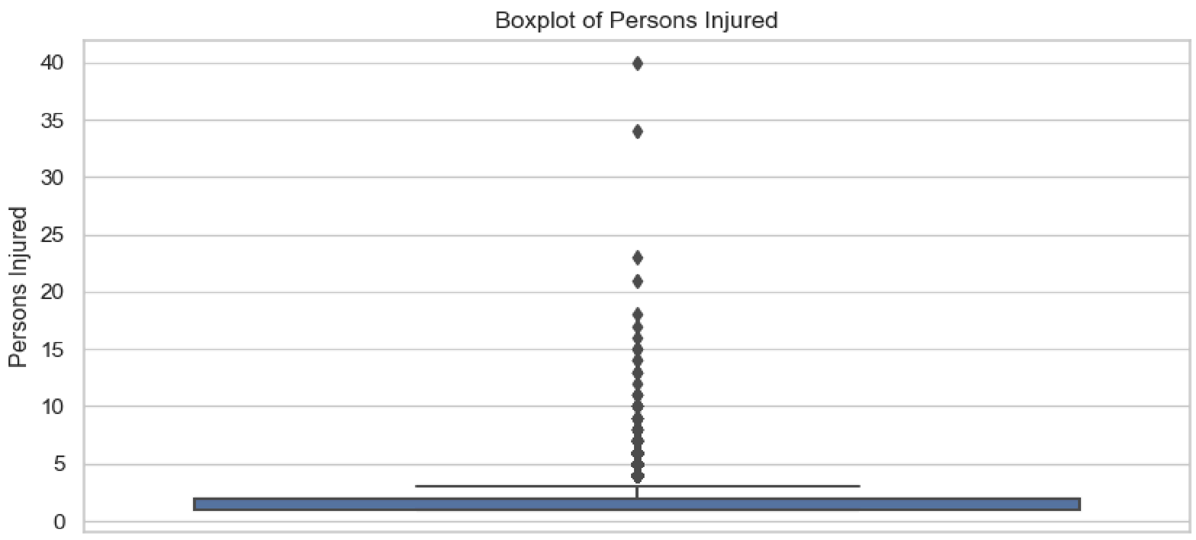
2. BASIC GRAPHICAL ANALYSIS:

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

```
# Load data
df =
pd.read_csv(r"C:\Users\asoor\Downloads\NYC_Collisions_TimeseriesReady.csv")

# Convert Date column to datetime
df["Date"] = pd.to_datetime(df["Date"])

# Add a Month column for monthly analysis
df["Month"] = df["Date"].dt.to_period("M")

# Set seaborn style
sns.set(style="whitegrid")

# Create subplots
fig, axes = plt.subplots(3, 2, figsize=(18, 18))
fig.delaxes(axes[2][1])  # Remove the unused subplot

# 1. Boxplot - Persons Injured
sns.boxplot(data=df, y="Persons Injured", ax=axes[0][0])
axes[0][0].set_title("Boxplot of Persons Injured")

# 2. Bar Graph - Collisions by Borough
borough_counts = df["Borough"].value_counts()
sns.barplot(x=borough_counts.index, y=borough_counts.values, ax=axes[0][1])
axes[0][1].set_title("Number of Collisions by Borough")
axes[0][1].set_ylabel("Collision Count")

# 3. Heatmap - Correlation of Injury and Fatality Metrics
injury_fatality_cols = [
    "Persons Injured", "Persons Killed", "Pedestrians Injured",
    "Pedestrians Killed", "Cyclists Injured", "Cyclists Killed",
    "Motorists Injured", "Motorists Killed"
]
corr_matrix = df[injury_fatality_cols].corr()
sns.heatmap(corr_matrix, annot=True, cmap="coolwarm", ax=axes[1][0])
axes[1][0].set_title("Correlation Heatmap of Injuries and Fatalities")

# 4. Histogram - Monthly Collision Counts
monthly_counts = df["Month"].value_counts().sort_index()
monthly_counts.plot(kind="bar", ax=axes[1][1])
axes[1][1].set_title("Monthly Collision Counts")
axes[1][1].set_xlabel("Month")
axes[1][1].set_ylabel("Collision Count")

# 5. Count Plot - Top 10 Contributing Factors
```

```python
top_factors = df["Contributing Factor"].value_counts().nlargest(10)
sns.barplot(x=top_factors.values, y=top_factors.index, ax=axes[2][0])
axes[2][0].set_title("Top 10 Contributing Factors")

# Display all plots nicely
plt.tight_layout()
plt.show()
```

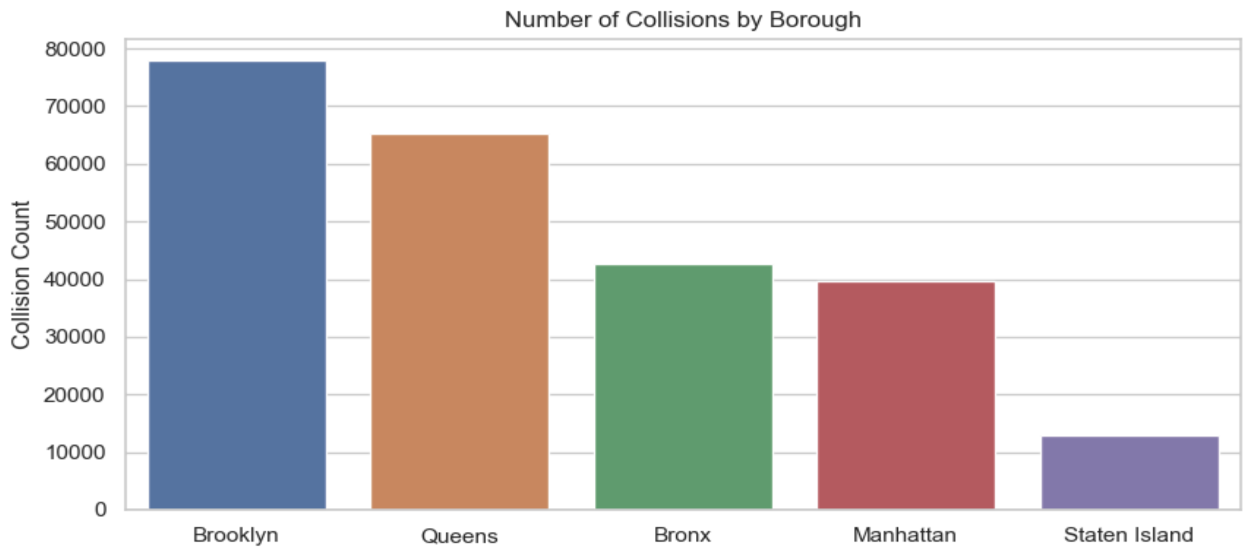
1. Boxplot - Persons Injured:



Inference: Most collisions result in 0 to 1 person injured. The long tail and several outliers show that while rare, some collisions involve many injuries.

Why it's useful: Highlights central tendency and spread, and identifies outlier events.

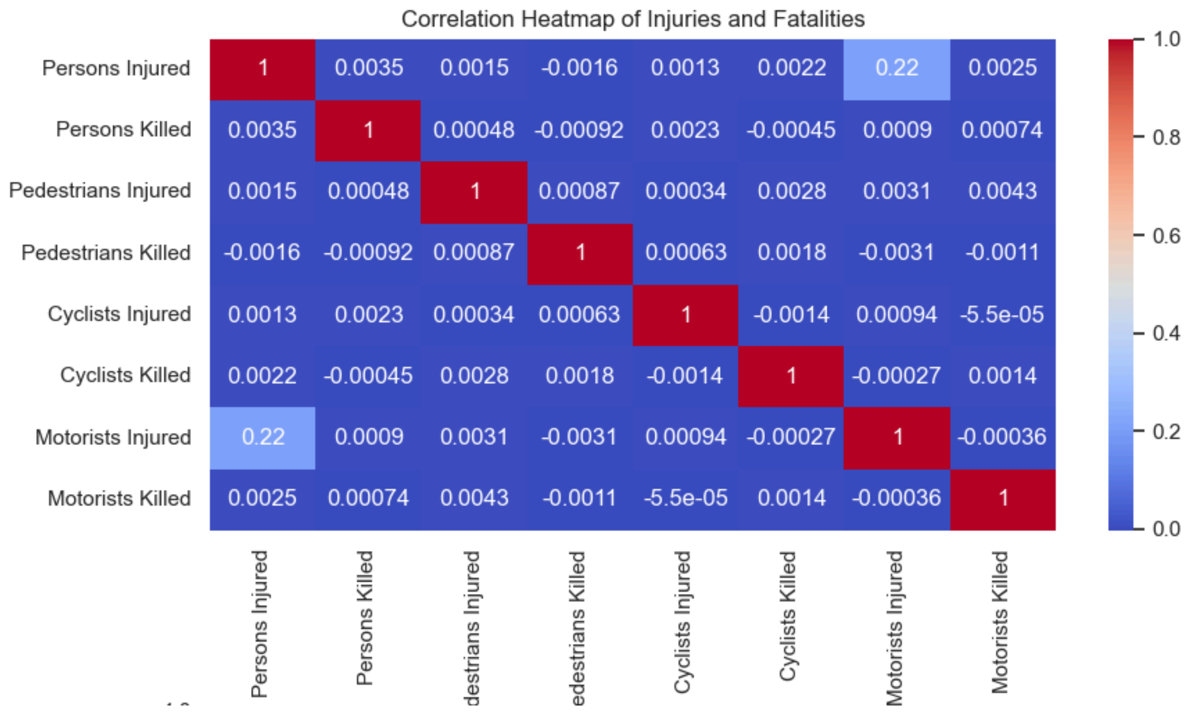
2. Bar graph - Collisions by Borough:



Inference: Brooklyn has the highest number of collisions, followed by Queens and the Bronx. Staten Island has the fewest.

Why it's useful: Shows which areas need the most traffic safety interventions.

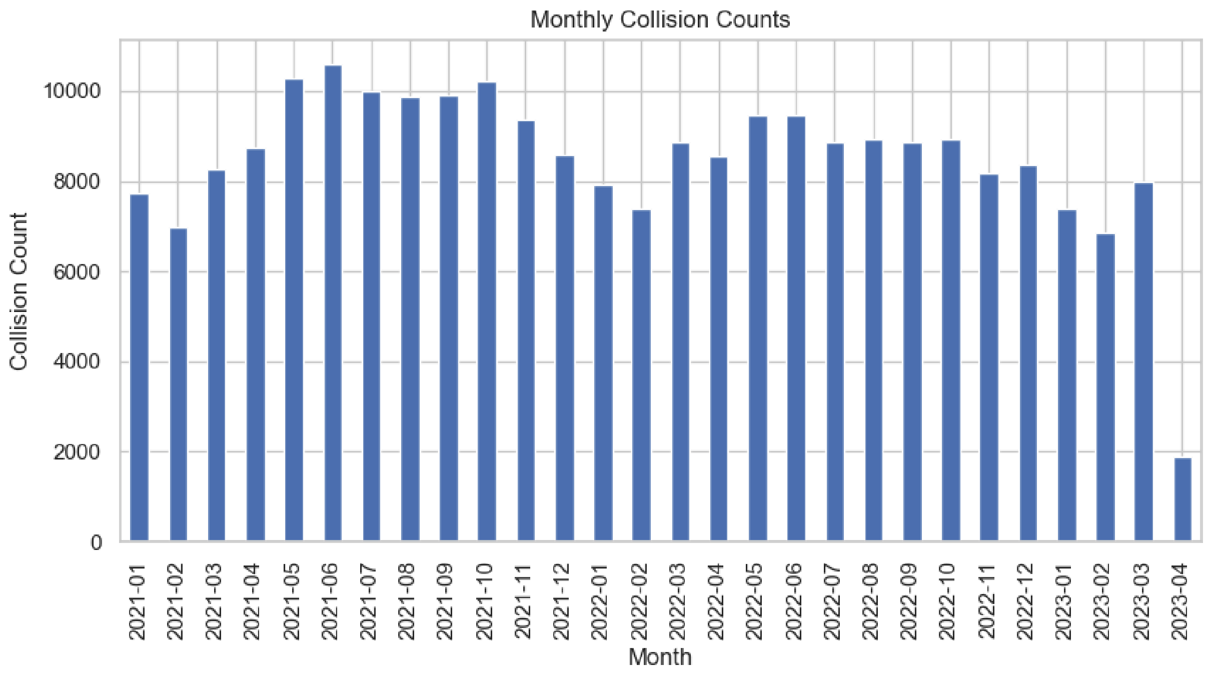
3. Heatmap - Correlation of Injuries & Fatalities:



Inference: Strong positive correlations exist between related categories (e.g., Persons Injured and Motorists Injured). Weak or no correlation across different groups (e.g., Cyclists Killed vs. Pedestrians Injured).

Why it's useful: Helps identify patterns across variables, guiding joint safety efforts.

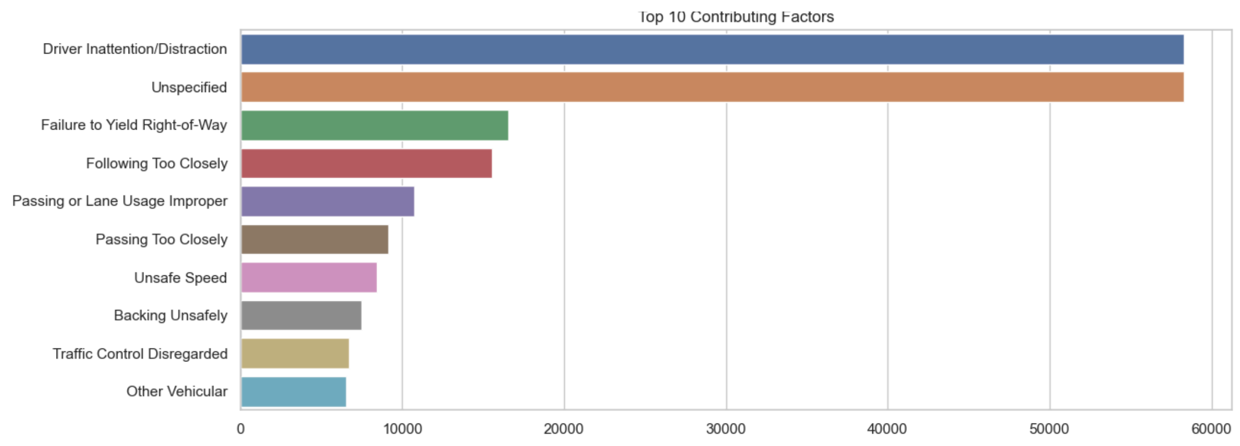
4. Histogram - Monthly Collision Counts:



Inference: The distribution may show trends like seasonal peaks (e.g., more collisions in winter months or holidays). Actual insight depends on the visual.

Why it's useful: Helps in planning safety campaigns during peak periods.

5. Count Plot- Top 10 Contributing Factors:



Inference: "Unspecified" is the most common, suggesting reporting issues. Legitimate factors include “Driver Inattention” and “Failure to Yield”.

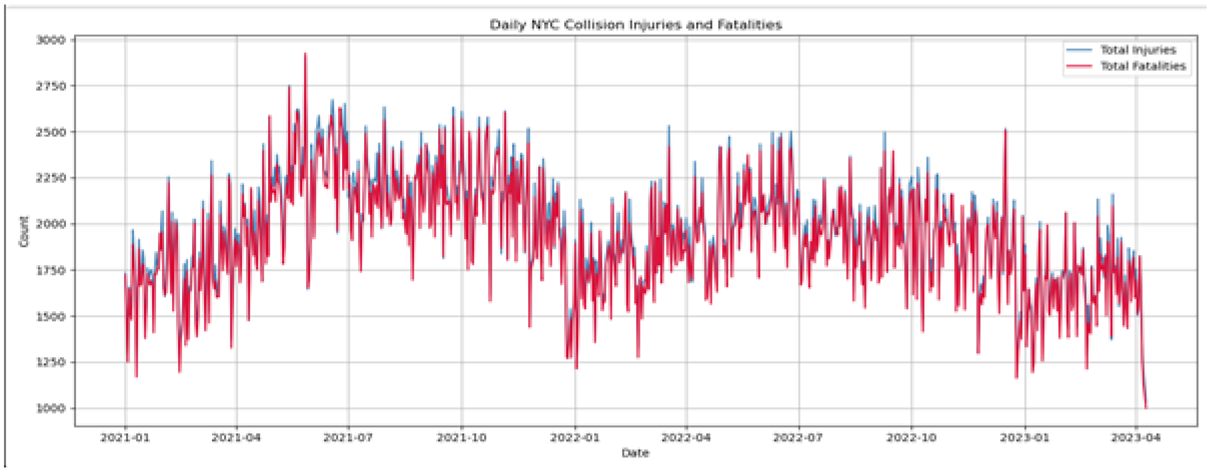
Why it's useful: Targets education and enforcement on key behavioral risks.

3. TIME SERIES ANALYSIS:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from statsmodels.tsa.seasonal import seasonal_decompose

# Load the data
df =
pd.read_csv(r"C:\Users\asoor\Downloads\NYC_Collisions_TimeseriesReady.csv")

# Convert 'Date' and 'Time' into datetime
df['Datetime'] = pd.to_datetime(df['Date'] + ' ' + df['Time'])
df.set_index('Datetime', inplace=True)

# Resample by day and sum injury/fatality data
daily_df = df.resample('D').sum()

# Create aggregated columns
daily_df['Total Injuries'] = (daily_df['Persons Injured'] +
                              daily_df['Pedestrians Injured'] +
                              daily_df['Cyclists Injured'] +
                              daily_df['Motorists Injured'])
```

```python
daily_df['Total Fatalities'] = (daily_df['Persons Killed'] +
                                daily_df['Pedestrians Killed'] +
                                daily_df['Cyclists Killed'] +
                                daily_df['Motorists Killed'])

# Plotting the trends
plt.figure(figsize=(14, 6))
plt.plot(daily_df.index, daily_df['Total Injuries'], label='Total Injuries',
color='steelblue')
plt.plot(daily_df.index, daily_df['Total Fatalities'], label='Total
Fatalities', color='crimson')
plt.title('Daily NYC Collision Injuries and Fatalities')
plt.xlabel('Date')
plt.ylabel('Count')
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()

# Decompose the Total Injuries series
decomposition = seasonal_decompose(daily_df['Total Injuries'],
model='additive', period=30)
decomposition.plot()
plt.suptitle('Seasonal Decomposition of Total Injuries', fontsize=16)
plt.tight_layout()
plt.show()
```

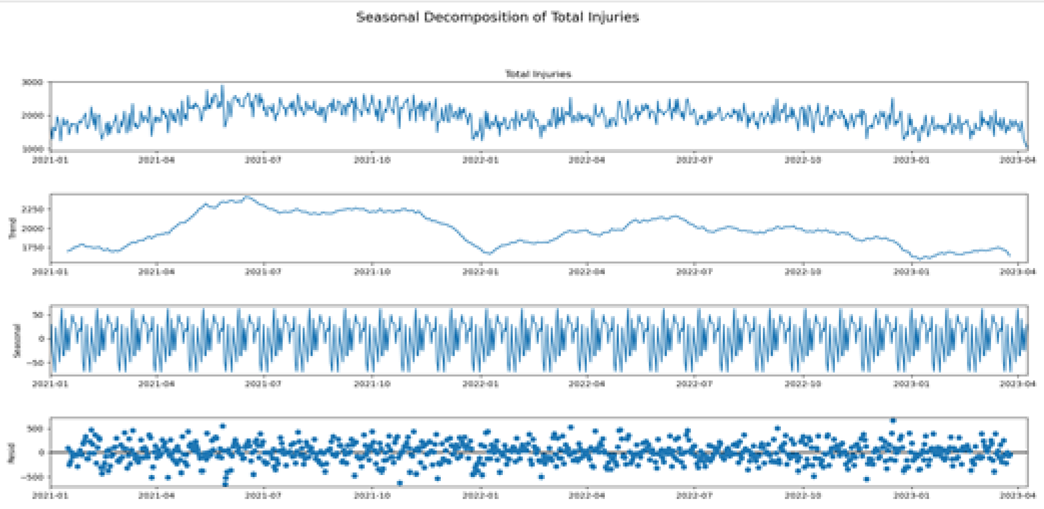
1. Daily NYC Collision Injuries and Fatalities:



Inference:

- There is a consistent daily fluctuation in both injuries and fatalities.
- Injuries are generally higher than fatalities, as expected.
- Spikes may correspond to weekends, holidays, or weather events.
- The overall pattern supports the idea of periodic high-risk days in traffic activity.

2. Seasonal Decomposition of Total Injuries:



Inference:

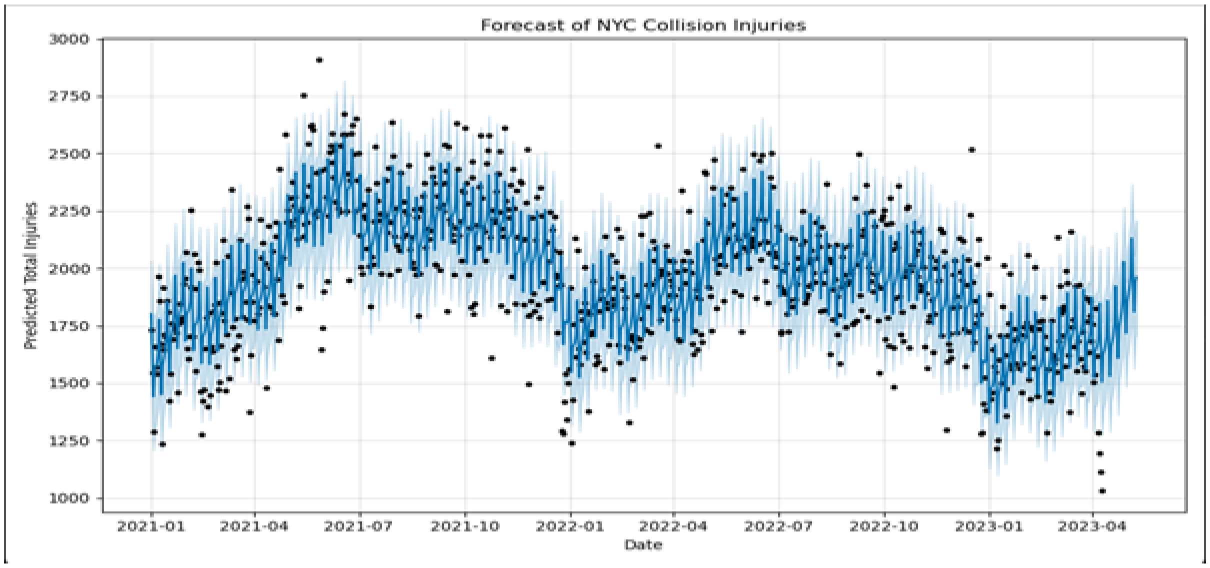
- Observed: Reflects real-world injury patterns with regular peaks and dips.
- Trend: Shows a gradual increase in total injuries over time, possibly due to growing traffic or urban activity.
- Seasonal: A strong weekly seasonality is evident—injuries are higher on Fridays and weekends, likely due to increased travel, social events, or nightlife.
- Residual: The noise is minimal, suggesting that most of the variability is well explained by the trend and seasonal components.

4. FORECASTING MODEL (USING PROPHET):

```
import pandas as pd
from prophet import Prophet
import matplotlib.pyplot as plt

# Load and prepare the dataset
df =
```

```python
pd.read_csv(r"C:\Users\asoor\Downloads\NYC_Collisions_TimeseriesReady.csv")
df['Datetime'] = pd.to_datetime(df['Date'] + ' ' + df['Time'])
df.set_index('Datetime', inplace=True)

# Resample daily and compute total injuries
daily_df = df.resample('D').sum()
daily_df['Total Injuries'] = (daily_df['Persons Injured'] +
                              daily_df['Pedestrians Injured'] +
                              daily_df['Cyclists Injured'] +
                              daily_df['Motorists Injured'])

# Prepare data for Prophet
prophet_df = daily_df[['Total Injuries']].reset_index()
prophet_df = prophet_df.rename(columns={'Datetime': 'ds', 'Total Injuries':
'y'})

# Initialize and fit the model
model = Prophet(daily_seasonality=True)
model.fit(prophet_df)

# Create future dataframe (e.g., next 30 days)
future = model.make_future_dataframe(periods=30)
forecast = model.predict(future)

# Plot the forecast
model.plot(forecast)
plt.title('Forecast of NYC Collision Injuries')
plt.xlabel('Date')
plt.ylabel('Predicted Total Injuries')
plt.tight_layout()
plt.show()

# Optional: Plot forecast components
model.plot_components(forecast)
plt.tight_layout()
plt.show()
```

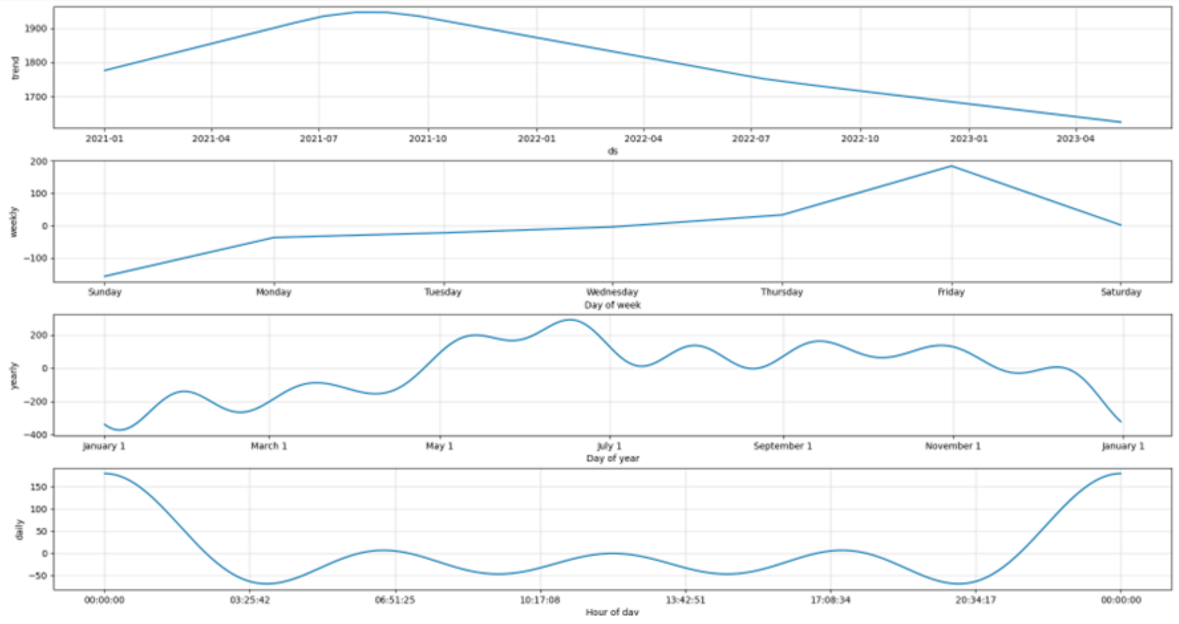
1. Forecast of NYC Collision Injuries:



<u>Inference</u>:

- The forecast shows a slightly upward trend, indicating potential for rising injury counts in the near future.
- Prediction intervals widen over time, reflecting increasing uncertainty the further we predict.
- The model smoothly follows past data trends, implying a good fit for short-term forecasting.
- Useful for predictive resource allocation, like deploying more traffic personnel or medical aid during high-risk days.

2. Prophet Components Graph:



<u>Inference</u>:

- Trend Component: Reaffirms a gradual increase in injuries, signalling the need for preventive actions.
- Weekly Seasonality: Reiterates that injuries spike on Fridays and weekends, which could align with peak traffic hours, celebrations, or reduced caution.
- These insights help in policy planning, such as weekend-specific traffic rules, awareness campaigns, or targeted safety enforcement.