CS 5764: Information Visualization

Author: Jay Sarode

Instructor: Dr. Reza Jafari

Date: 12/6/2023

Table of Contents

Table of Figures

Table of Tables

Abstract

In this project, we delve into the complexities of road traffic incidents, focusing on the interrelations between vehicle types, accident scenarios, and the consequent human impact. The study leverages cutting-edge information visualization techniques to decode and display intricate data patterns, offering fresh insights that could revolutionize road safety strategies.

Globally, road accidents constitute a major public health concern, with a significant human and economic toll. A deeper understanding of how different vehicle categories contribute to these accidents and the severity of the resulting casualties is vital for crafting effective safety measures. This project seeks to shed light on these aspects, providing a nuanced analysis that can inform stakeholders and influence policymaking.

Employing a comprehensive dataset that encompasses diverse aspects of road accidents over several years, our approach stands out in its use of sophisticated visualization tools. We incorporate multi-dimensional data representations, including dynamic heat maps, geospatial analysis, to uncover hidden patterns and correlations within the data. This method allows us to present a layered and detailed exploration of the data, accessible to both experts and laypersons.

The analysis uncovers previously obscured trends, such as specific risk profiles associated with different vehicle types across various urban and rural contexts. We also identify critical temporal patterns, linking accident rates to external factors like environmental conditions and peak traffic periods. These findings offer compelling evidence for targeted interventions.

The project transcends traditional statistical analyses of road accidents, providing an intuitively understandable and visually engaging exploration of the data. This approach not only enriches academic research in the field of traffic safety but also serves as a valuable tool for policymakers, safety advocates, and the broader community. The insights gained can directly influence the development of more nuanced and effective traffic safety regulations and educational initiatives.

Building on this foundation, we aim to explore predictive modeling techniques to identify potential accident hotspots and forecast casualty rates under various hypothetical scenarios. This proactive approach seeks to not just understand but also anticipate and mitigate risks, further contributing to the advancement of road safety measures.

Introduction

The Final Term Project (FTP) presents an opportunity to synthesize the Python visualization skills acquired during the course and apply them to a real-world dataset. The objective is to develop a Python-based, web-enabled dashboard for interactive information visualization and deploy this application using Google Cloud Platform (GCP). The project unfolds in three distinct phases, each building upon the last, to create a comprehensive and user-friendly visualization tool.

Project Objectives and Procedures

The core objective of the FTP is twofold: to demonstrate proficiency in Python visualization techniques and to effectively utilize these skills in creating a dynamic, web-based dashboard. This dashboard will serve as an interactive platform for users to engage with the dataset, offering insights through various visual representations.

Phase I: Static Graphs & Tables

- *Static Visualization*: Utilizing Python's robust visualization libraries, this phase focuses on generating static graphs and tables. These visualizations aim to reveal underlying patterns and relationships within the dataset, laying the groundwork for more complex, interactive displays.

Phase II: Interactive Web-based Dashboard

- *Dashboard Development*: In this phase, we will leverage the Dash framework to create an interactive web-based dashboard. This framework, with its core and HTML components, allows for the development of a highly interactive user interface.

Phase III: Deployment

- *Dockerization*: The developed application will be containerized using Docker, ensuring that it remains consistent across various computing environments.
- *Deployment on GCP*: Finally, the dockerized application will be deployed via the Google Cloud Platform, making it accessible for public use. This step involves setting up the necessary cloud infrastructure to support the application, ensuring its availability and scalability.

Description of the Dataset

The dataset under consideration provides a comprehensive overview of personal injury road accidents in Great Britain (GB) from 2005 to 2014. It is a rich repository of information, encapsulating various dimensions of road safety and accident details. This dataset is structured into four main files: the Accident file, the Vehicle file, the Casualty file, and the Lookup file. Each of these files offers unique insights into different aspects of road accidents.

Accident File

- *Contents*: This primary dataset encompasses extensive details about each accident. It includes data on accident severity, weather conditions, geographic location, date and time, day of the week, and road types.
- *Applications*: This file is crucial for understanding the environmental and temporal factors contributing to road accidents, assisting in identifying high-risk factors and areas.

Vehicle File

- *Contents*: This file delves into specifics about the vehicles involved in the accidents. It covers information on the vehicle type, model, engine size, as well as the age and sex of the driver, and the age of the car.
- *Applications*: Analysis of this data can unveil patterns in vehicle-related factors in accidents, aiding in vehicle safety design and targeted driver safety programs.

Casualty File

- *Contents*: Focused on individuals affected by the accidents, this file records casualty severity, age, sex, social class, and whether the individual was a pedestrian, driver, or passenger.
- *Applications*: This file is essential for understanding the human impact of road accidents, informing healthcare and emergency response strategies.

Lookup File

- *Contents*: Serving as a key to interpret the datasets, this file provides textual descriptions of all the variable codes used in the three main files, ensuring clarity and ease of understanding for users.
- *Applications*: It is vital for accurate data interpretation, making the dataset accessible to a wider range of users, including those without a technical background.

The files of accidents, vehicles and casualties are merged together for highly rigorous, combined analysis.

Dependent Variable: The dataset's dependent variable is typically the accident severity, or casualty severity. There might be some features that can be derived from the base features in the dataset.

The dataset on road safety from 2005 to 2014 is of significant value across various industries. Transport authorities can leverage it to refine road safety measures, develop efficient traffic management systems, and design safer roadways. For the automobile industry, data is pivotal in enhancing vehicle safety features and constructing cars that better mitigate injury risks in accidents. Insurance companies benefit from this dataset by gaining insights for more accurate risk assessments, policy pricing, and understanding patterns in accident occurrences. In the healthcare sector, the data aids in formulating improved emergency response strategies and allocating resources effectively in high-risk areas. Lastly, policy makers and researchers can utilize this rich dataset for creating more focused road safety policies and conducting comprehensive safety research, thereby contributing to overall road safety improvements.

Pre-processing of the Dataset

Each of the three files are loaded, i.e., the accidents file, the vehicles file and the casualty file.

Accident File:

```
(1640597, 32)
  Accident_Index  ...  LSOA_of_Accident_Location
0  200501BS00001  ...                 E01002849
1  200501BS00002  ...                 E01002909
2  200501BS00003  ...                 E01002857
3  200501BS00004  ...                 E01002840
4  200501BS00005  ...                 E01002863
```

Vehicle File:

```
(3004425, 22)
  Accident_Index  Vehicle_Reference  ...  Driver_IMD_Decile  Driver_Home_Area_Type
0  200501BS00001                  1  ...                  7                     1
1  200501BS00002                  1  ...                 -1                    -1
2  200501BS00003                  1  ...                  2                     1
3  200501BS00003                  2  ...                  1                     1
4  200501BS00004                  1  ...                  2                     1
```

Casualty File:

```
(3004425, 53)
  Accident_Index  ...  Driver_Home_Area_Type
0  200501BS00001  ...                     1
1  200501BS00002  ...                    -1
2  200501BS00003  ...                     1
3  200501BS00003  ...                     1
4  200501BS00004  ...                     1
```

These files are merged together based on the common ‘Accident_Index’ feature, and the join is done through inner join.

Final Dataset:

```
Final Dataset:
(4287593, 67)
  Accident_Index  Location_Easting_OSGR  ...  Casualty_Type  Casualty_Home_Area_Type
0  200501BS00001               525680.0  ...              0                        1
1  200501BS00002               524170.0  ...             11                        1
2  200501BS00003               524520.0  ...              9                        1
3  200501BS00003               524520.0  ...              9                        1
4  200501BS00004               526900.0  ...              0                        1
```

The final dataset shape is: 4287593 samples and 67 features.

There are some categorical columns, which are already converted to numerical. But for our exploratory data analysis, the features need to be mapped back with a help of a lookup file which has the labels and value pairs of features.

Through the fuzzy matching we map the sheet name with the exact columns name, having the feature similarity score of greater than 80.

A sample can be shown here:

```
                          Actual Column Name  ... Similarity Score
0                                Police_Force  ...               92
1                            Accident_Severity  ...               94
2                                 Day_of_Week  ...               82
3                  Local_Authority_(District)  ...               92
4                   Local_Authority_(Highway)  ...               92
```

The new dataset is saved into the new csv file, which is loaded into the new python file.

In the new python file, the data loaded is loaded into chunks so that the data loads quickly and efficiently.

Here is the gist of the how the data was loaded:

```
Number of chunks read:  (200000, 67)
Number of chunks read:  (200000, 67)
Number of chunks read:  (200000, 67)
Number of chunks read:  (200000, 67)
Number of chunks read:  (200000, 67)
Number of chunks read:  (200000, 67)
```

```
Time taken to read the data:  50.28195023536682
```

	Accident_Index	Location_Easting_OSGR	...	Casualty_Type	Casualty_Home_Area_Type
0	200501BS00001	525680.0	...	0	1
1	200501BS00002	524170.0	...	11	1
2	200501BS00003	524520.0	...	9	1
3	200501BS00003	524520.0	...	9	1
4	200501BS00004	526900.0	...	0	1

The data which has been loaded, was checked for null values. The dataset contained null values, which were dropped.

There are some the columns that had multiple special characters in it. These special characters were replaced by more meaningful characters.

After the data preprocessing, the data is prepared for exploratory data analysis.

Phase 1: Static Plots and Tables

Line Plots:

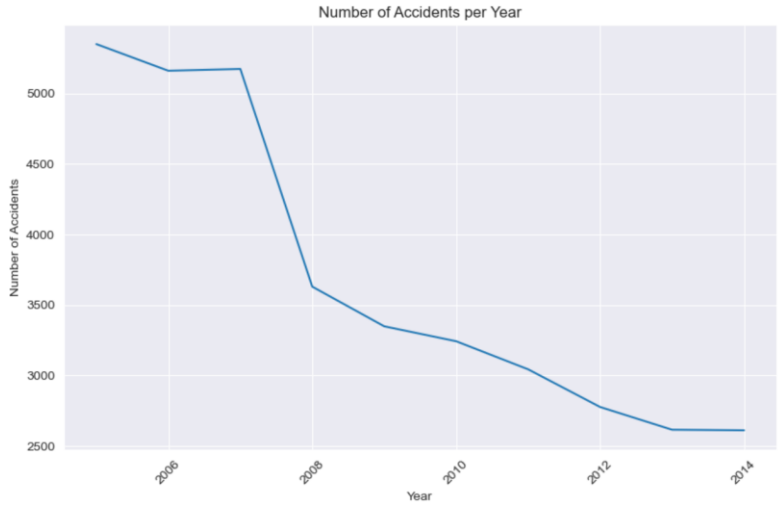

Fig. 1: Line graph of no. of accidents per year

Observations: The graph shows a consistent decline in road accidents from 2005 to 2014, with a steep drop between 2007 and 2008, followed by a more gradual decrease and a potential leveling off in the latter years, suggesting improved road safety or other influencing factors during this period.

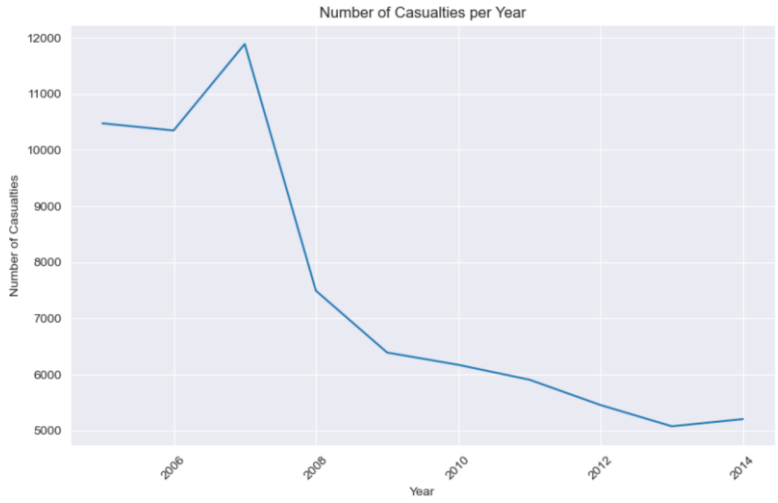

Fig. 2: Line graph of no. of casualties per year

Observations: The graph illustrates a dramatic decline in the number of casualties from road accidents from a peak around 2006 to 2014, with a sharp decrease after 2007 and a general downward trend throughout the period, indicating potential improvements in road safety and emergency response.

Bar Plots:

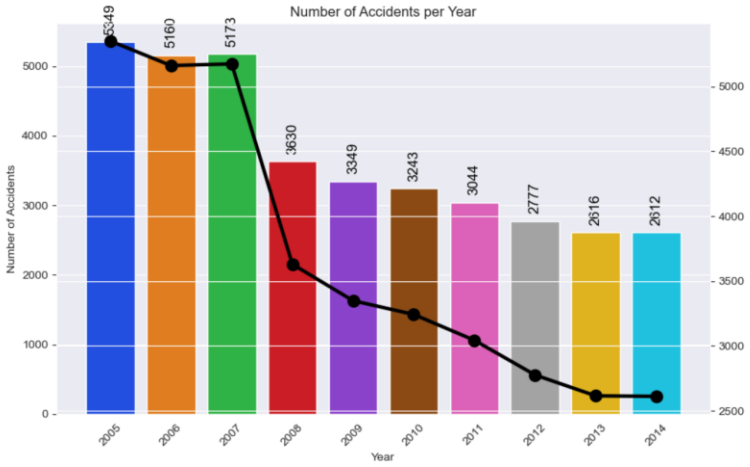

Fig. 3: Bar graph of no. of accidents per year

Observations: The graph depicts a continuous decline in the number of road accidents from 2005 to 2014, with colored bars representing each year and a black line indicating the trend. The sharp decrease between 2007 and 2008 stands out, and the overall trend line shows a steady reduction in accidents over the decade.

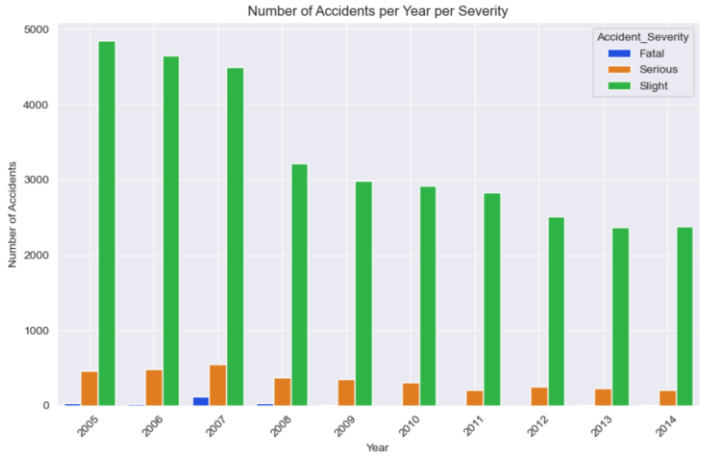

Fig. 4: Bar graph of no. of accidents per year per severity

Observations: The graph presents the number of road accidents from 2005 to 2014, categorized by severity: fatal, serious, and slight. Across the years, there's a noticeable decrease in all types of accidents, with slight accidents being the most common, followed by serious, and then fatal, which are the least frequent. The overall trend shows a significant reduction in the total number of accidents, indicating potential improvements in road safety.

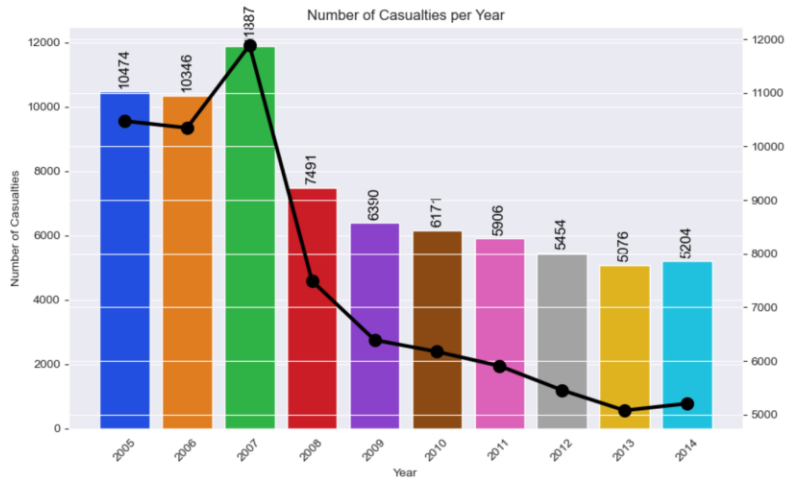

Fig. 5: Bar graph of no. of casualties per year

Observations: The graph indicates a significant downward trend in the number of casualties from road accidents in Great Britain from 2005 to 2014. Notably, there is a sharp decline after 2007. The overall reduction in casualties suggests enhanced road safety and possibly more effective emergency responses during this period.

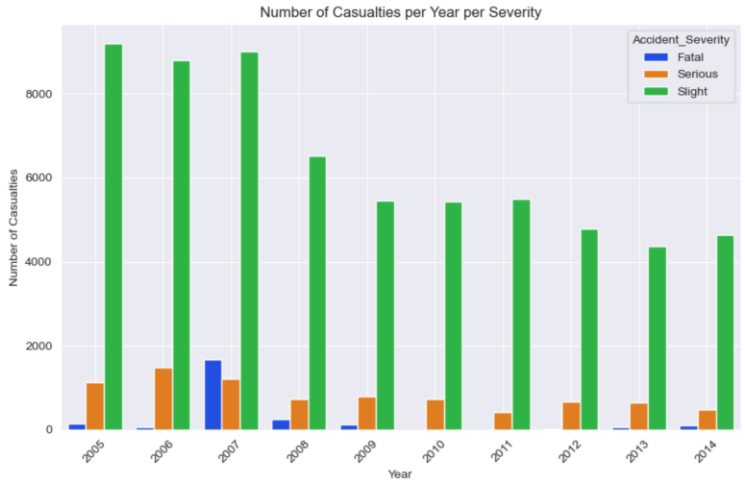

Fig. 6: Bar graph of no. of casualties per year per severity

Observations: The graph shows a consistent decline in road casualties in Great Britain from 2005 to 2014, categorized by severity—fatal, serious, and slight. Slight casualties are the most common, serious casualties less so, and fatal casualties the least, with all categories showing a downward trend over the ten-year period.

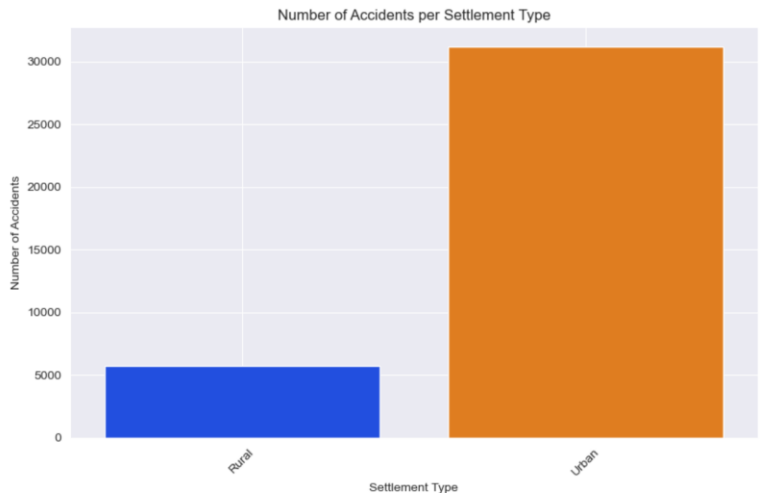

Fig. 7: Bar graph of no. of accidents per year per settlements

Observations: The graph compares the number of road accidents in rural versus urban settings. It shows a significantly higher number of accidents in urban areas compared to rural areas, indicating that urban centers may have more risk factors contributing to road accidents, or simply a higher volume of traffic leading to a greater incidence of accidents.

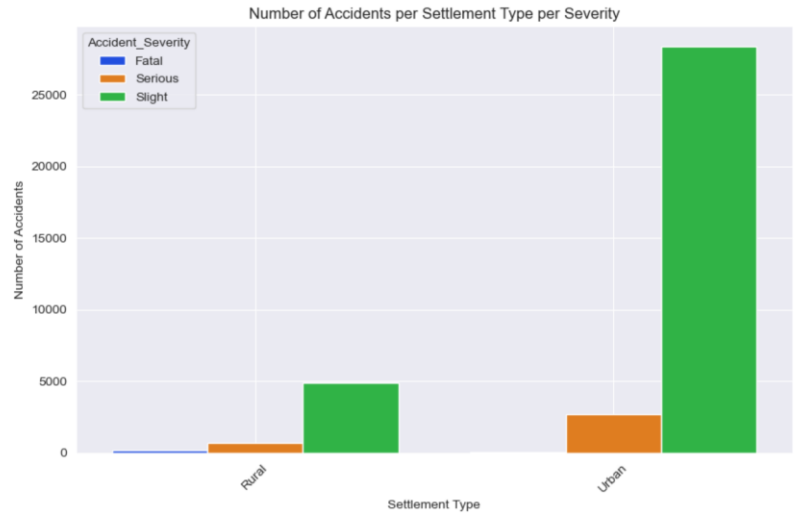

Fig. 8: Bar graph of no. of accidents per year per settlements per severity

Observations: The graph illustrates the distribution of road accidents by severity in rural and urban settings. In both environments, accidents of slight severity are most common, but the total number of accidents is markedly higher in urban areas across all severity levels. Fatal accidents are relatively low in both settings, yet they, along with serious accidents, contribute to the overall higher accident count in urban areas.

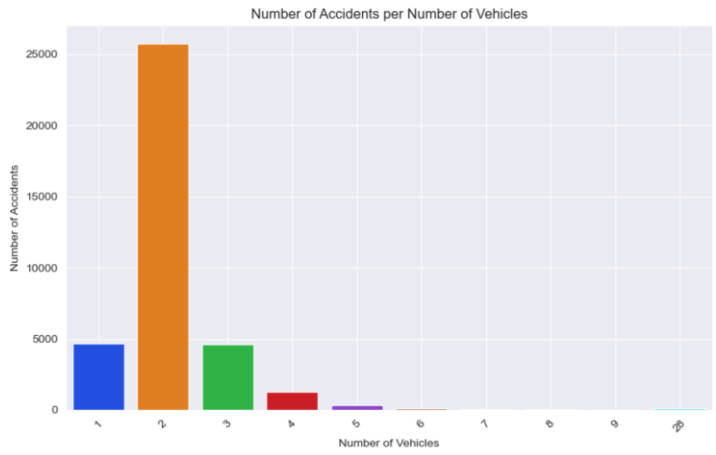

Fig. 9: Bar graph of no. of vehicles affected in accidents.

Observations: The graph shows the number of road accidents correlated to the number of vehicles involved. Most accidents involve two vehicles, followed by accidents with a single vehicle. The frequency of accidents decreases as the number of vehicles involved increases, with incidents involving more than two vehicles becoming progressively rarer. This pattern suggests that the most common road accidents are likely to be between two vehicles or involve a single vehicle.

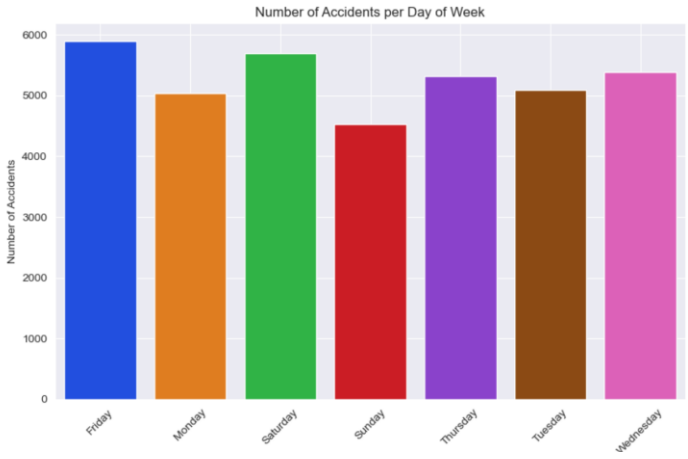

Fig. 10: Bar graph of accidents occurred in weekends/weekdays

Observations: The graph shows the distribution of road accidents across different days of the week. Friday has the highest number of accidents, followed by a somewhat uniform distribution among the other weekdays. Sunday is observed to have the lowest number of accidents. This pattern may indicate varying traffic conditions and behaviors, with Friday potentially having more traffic due to weekend plans and Sunday being quieter on the roads.

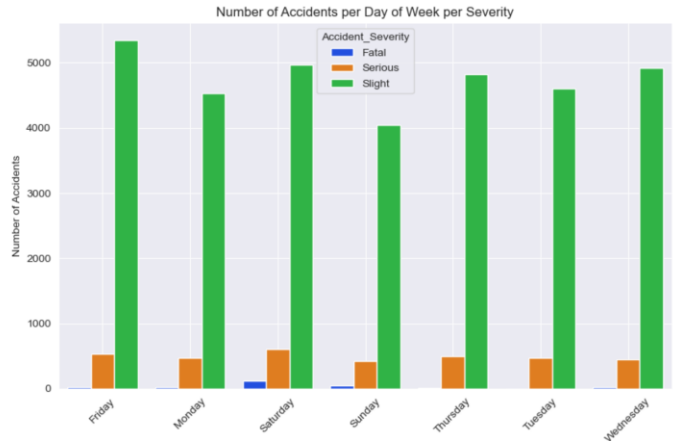

Fig. 11: Bar graph of accidents occurring in weekends/weekdays per severity

Observations: The graph illustrates the number of road accidents by day of the week, categorized by severity (fatal, serious, slight). Slight accidents are the most frequent on all days, with the highest overall number of accidents occurring midweek. Fatal accidents are the least frequent but appear slightly more often during the weekend. This suggests that while slight accidents are common throughout the week, the weekends may pose a higher risk for more severe accidents.

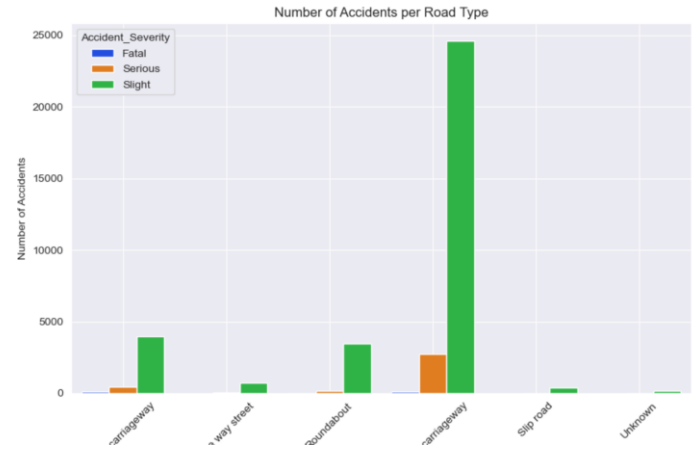

Fig. 12: Bar graph of accidents per road type

Observations: The graph categorizes road accidents by road type and severity. Single carriageways see the highest number of accidents, particularly those classified as slight. Dual carriageways, roundabouts, and slip roads feature fewer accidents, with the least on slip roads. Fatal accidents occur least frequently across all road types. The data indicates that single carriageways might be hotspots for traffic incidents, warranting further safety analysis and interventions.

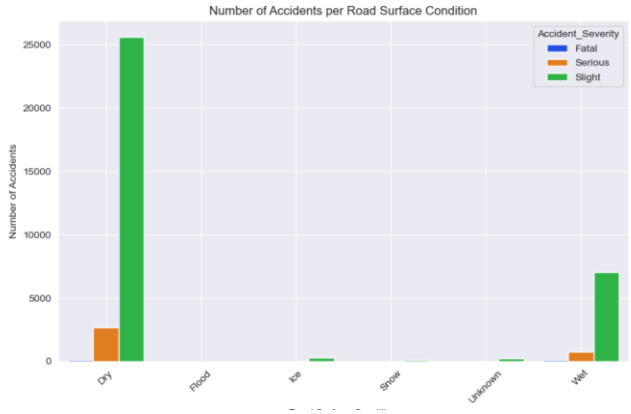

Fig. 13: Bar graph of accidents per road surface conditions

Observations: The graph shows the number of road accidents according to road surface conditions. The majority of accidents occur on dry roads, followed by wet road conditions. Accidents on icy, snowy, or flooded surfaces are comparatively fewer. The 'Unknown' category suggests a non-negligible number of accidents where the road condition wasn't reported or determined. This data may reflect that while adverse weather conditions are risk factors, the sheer volume of traffic on dry roads results in a higher frequency of accidents.

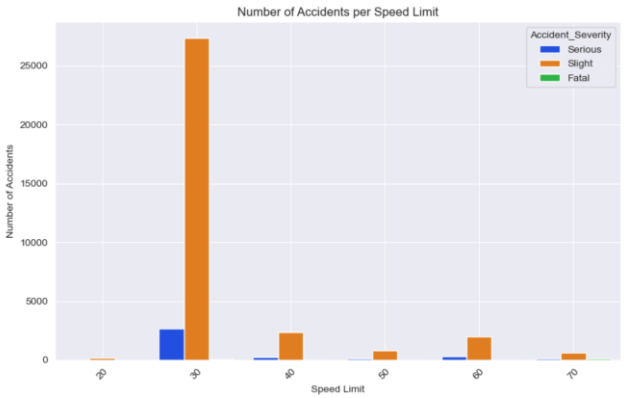

Fig. 14: Bar graph of accidents due to speed limit

Observations: The graph shows the number of road accidents in relation to different speed limits. Most accidents occur in areas with a speed limit of 30 mph, which are typically urban or residential areas. The frequency of accidents decreases significantly as the speed limit increases, with the fewest accidents occurring in the 70 mph speed limit zones. This suggests that lower speed limit areas, despite the reduced speed, may have a higher incidence of accidents, possibly due to increased traffic density or intersections.

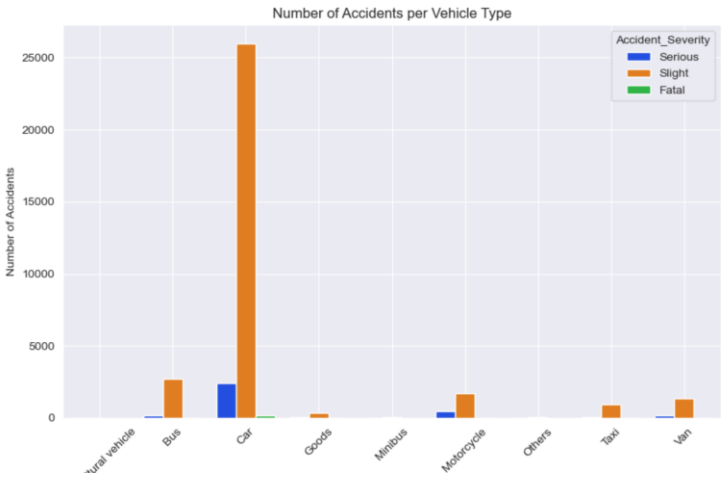

Fig. 15: Bar graph of unique vehicles affected in accidents.

Observations: The graph displays the number of road accidents categorized by vehicle type, along with the severity of the accidents (fatal, serious, slight). Cars are involved in the vast majority of accidents, with a high number of slight severity, followed by serious and fewer fatal accidents. All other vehicle types show significantly fewer accidents by comparison. This data highlights that cars are the most common participants in road accidents, which could be reflective of their prevalence on the roads.

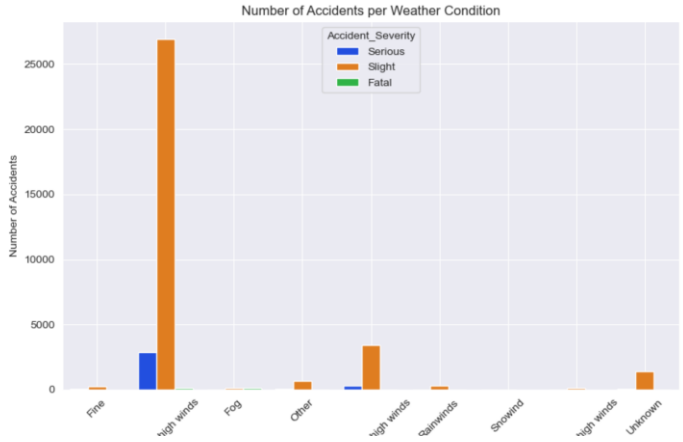

Fig. 16: Bar graph of weather conditions per severity

Observations: The graph depicts road accidents in relation to different weather conditions. The highest number of accidents occur during fine, clear weather, with a significantly smaller number occurring during rain and other weather conditions. Incidents during snow and high winds are even less common. Fatal accidents are relatively low across all weather conditions. This could suggest that although adverse weather conditions are typically considered hazardous, most accidents happen during clear weather conditions.

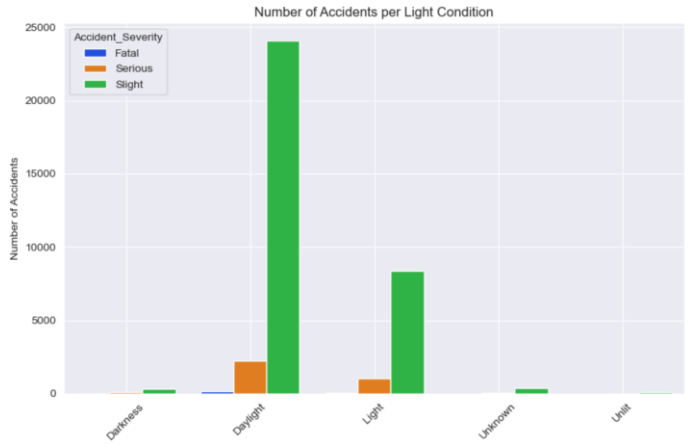

Fig. 17: Bar graph of light conditions per severity

Observations: The chart displays road accidents based on light conditions, showing that the majority occur in daylight, with a significantly smaller number happening in darkness and even fewer when streetlights are present or there is lighting. Accidents under unknown lighting conditions are minimal. The severity of accidents follows a similar pattern across different lighting, with slight accidents being the most common, followed by serious, and then fatal accidents being the least common in all lighting conditions.

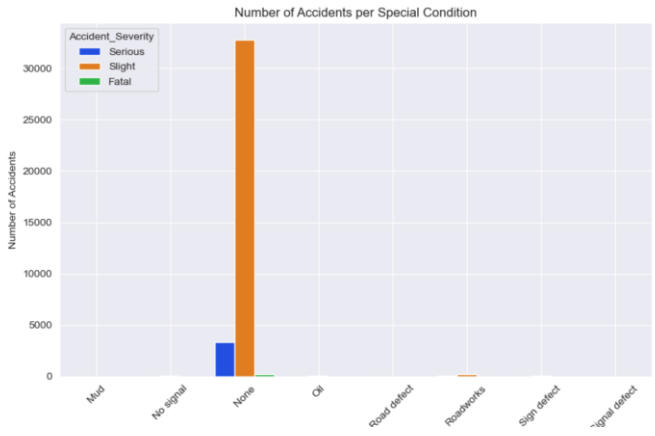

Fig. 18: Bar graph of special conditions per severity

Observations: The chart categorizes road accidents based on special road conditions. The overwhelming majority of accidents occur under no special conditions, indicating that typical driving environments are where most incidents happen. Accidents in conditions like mud, oil, road defects, roadworks, sign defects, and signal defects are significantly less frequent.

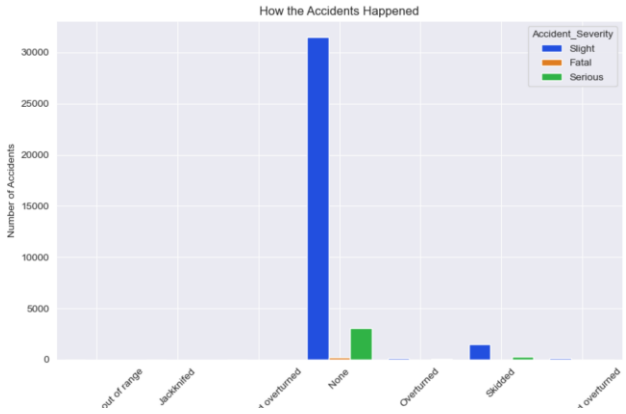

Fig. 19: Bar graph of how accidents happened.

Observations: The chart categorizes road accidents by the manner in which they occurred. A significant majority of the accidents are not specified by any particular manner, indicating a 'None' category. Among the specified categories, 'Overturned' and 'Skidded' represent a small fraction of the accidents. Fatal accidents are the least common in all categories. The dominance of the 'None' category could suggest either a lack of detailed reporting in the data or that most accidents don't involve the vehicle overturning or skidding.

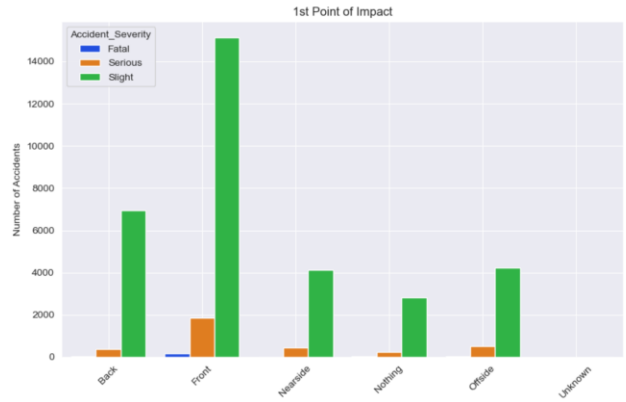

Fig. 20: Bar graph of first point of impact

Observations: The graph categorizes road accidents by the first point of impact on the vehicle. The front of the vehicle is the most common point of impact, with a significant number of both slight and serious accidents, and a smaller number of fatal accidents. The back and offside of the vehicle are the next most common points of impact, but with considerably fewer accidents than the front. Nearside impacts and accidents without a specified impact point (Nothing) are less common, while a very small number of accidents have an unknown point of impact. This suggests that head-on or frontal impacts are the predominant type of accidents.

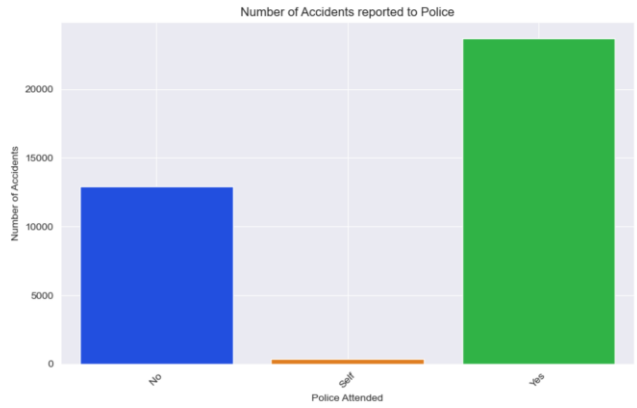

Fig. 21: Bar graph of how accidents happened.

Observations: The graph displays the number of road accidents that were reported to the police. A significant majority of accidents were reported and attended by police, whereas a much smaller number were not reported. There is a negligible category labeled "Self", which could indicate incidents that were self-reported by the involved parties without requiring police at the scene.

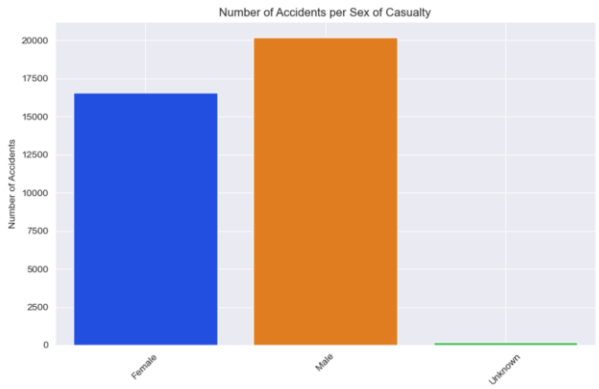

Fig. 22: Bar graph of no. of accidents to sex of casualty

Observations: The chart shows the number of road accidents categorized by the sex of the casualty. There is a relatively even distribution between female and male casualties, with male casualties being slightly higher. A very small number of accidents involve casualties whose sex was unknown.

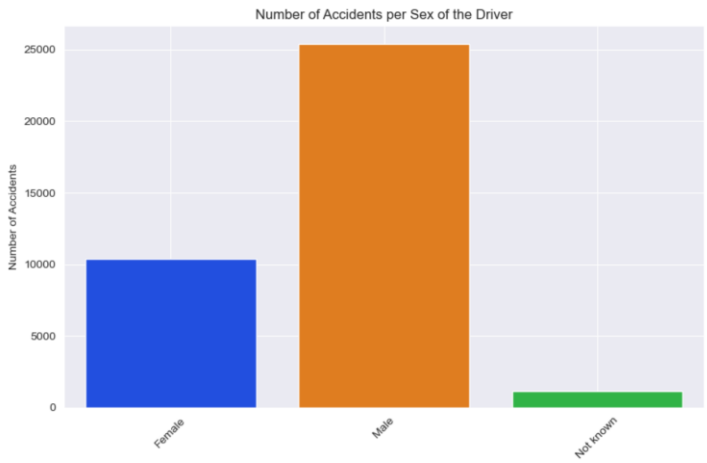

Fig. 23: Bar graph of no. of accidents to sex of the driver

Observations: The graph illustrates the number of road accidents based on the sex of the driver involved. It shows that male drivers are involved in a significantly higher number of accidents than female drivers. There is also a small number of accidents where the sex of the driver is not known.

Pie Charts:

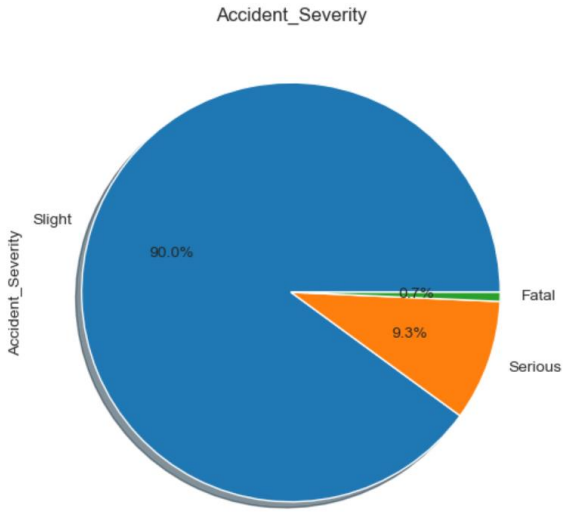

Fig. 24: Pie chart of accident severity

Observations: The pie chart represents the distribution of road accidents by severity. The vast majority, 90%, are classified as slight accidents, indicating minor injuries or damage. Serious accidents account for 9.3% of the total, signifying more significant harm or potential for long-term consequences. Fatal accidents comprise the smallest portion, at 0.7%, indicating a relatively low incidence of accidents resulting in death compared to non-fatal incidents.

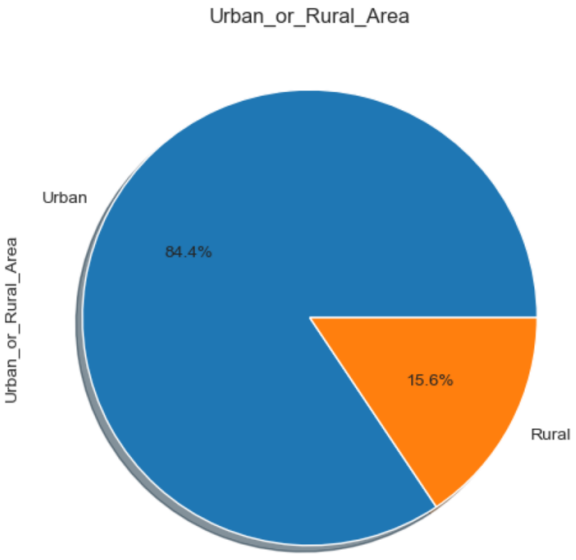

Fig. 25: Pie chart of settlement type

Observations: The pie chart illustrates the distribution of road accidents between urban and rural areas. A significant majority, 84.4%, of accidents occur in urban areas, while the remaining 15.6% take place in rural areas.

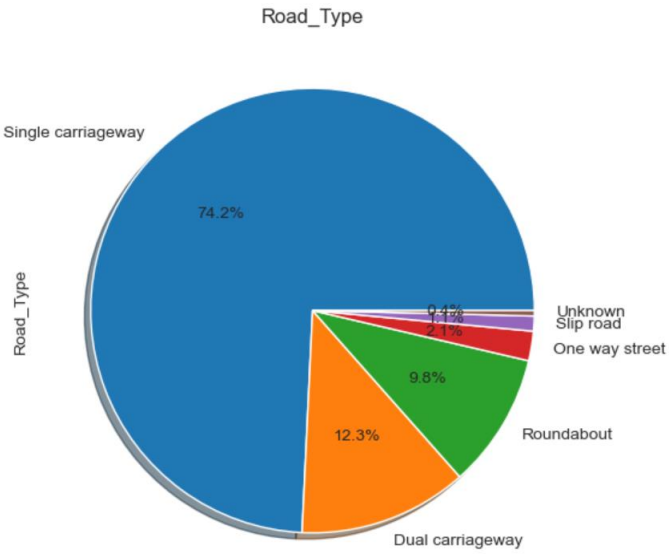

Fig. 26: Pie chart of road type

Observations: The pie chart presents the distribution of road accidents by road type. Most accidents occur on single carriageways, accounting for 74.2% of the total. Dual carriageways are the next most common site for accidents, with 12.3%. Roundabouts see a smaller proportion of accidents, represented by 9.8%. Accidents on one-way streets and slip roads, as well as those on roads of unknown types, make up a very small fraction of the total.

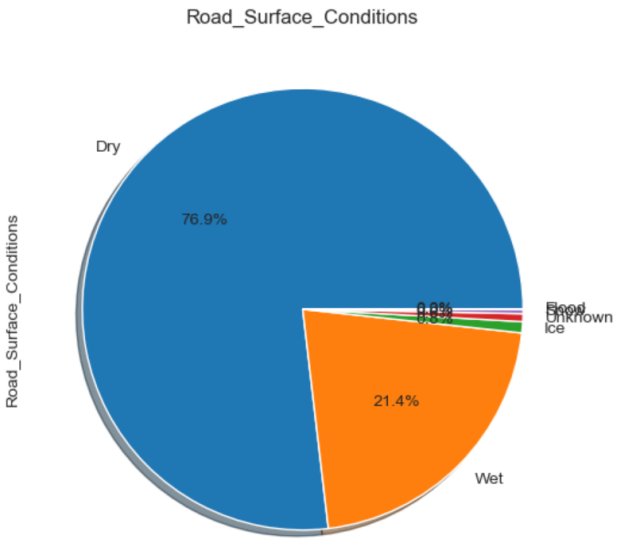

Fig. 27: Pie chart of road surface conditions

Observations: The pie chart indicates the proportion of road accidents occurring under different road surface conditions. A large majority, 76.9%, of accidents happen on dry road surfaces. Wet road conditions account for 21.4% of accidents. A very small percentage occurs in snowy or icy conditions, and an even smaller fraction is categorized under unknown conditions.

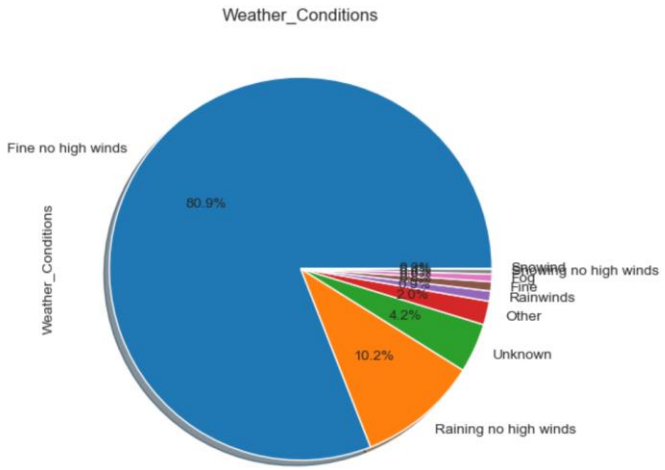

Fig. 28: Pie chart of weather conditions

Observations: The pie chart displays the distribution of road accidents by weather conditions. Most accidents, 80.9%, occur during fine weather with no high winds. Accidents during rainy weather with no high winds account for 10.2%, followed by a small percentage of accidents in other specified weather conditions such as snow and fog. A very small fraction is recorded under unknown weather conditions.

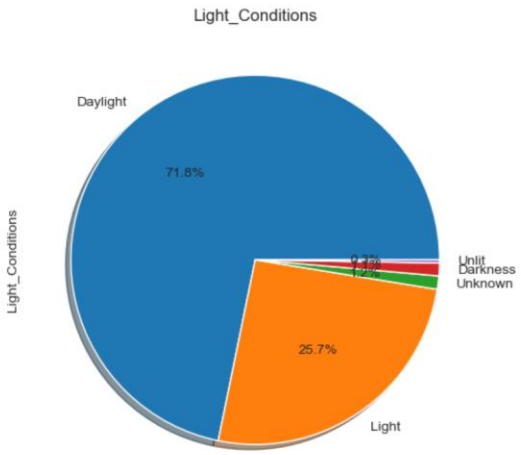

Fig. 29: Pie chart of light conditions

Observations: The pie chart shows the breakdown of road accidents according to the light conditions at the time they occurred. A majority of accidents, 71.8%, happen in daylight. Accidents in light conditions, which may refer to street-lit conditions, account for 25.7%. A very small portion occurs in unlit darkness, and an even smaller fraction is under unknown lighting conditions.

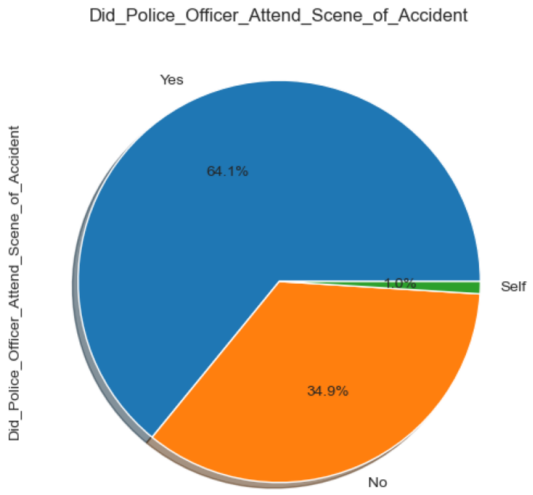

Fig. 30: Pie chart of accidents reported to police.

Observations: The pie chart represents the frequency of police attendance at the scene of road accidents. In 64.1% of the cases, a police officer did attend the scene of the accident. There were 34.9% of accidents where no police officer attended, and a very small

percentage of accidents were reported as 'Self', which may indicate self-reporting to the police without their attendance at the scene.

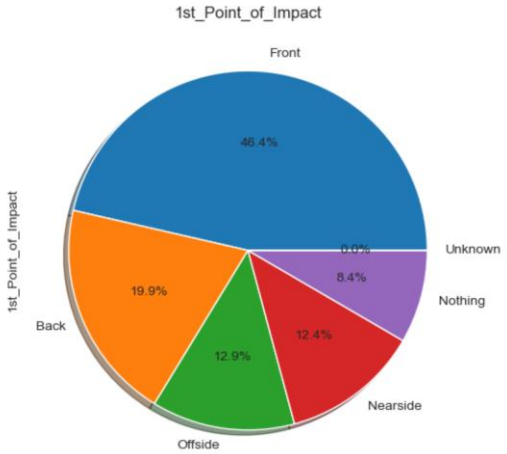

Fig. 31: Pie chart of first point of impact

Observations: The pie chart displays the distribution of the first point of impact in a dataset. The largest segment, colored in blue, represents the 'Front' impact, accounting for 46.4% of the cases. This is followed by 'Back' and 'Offside' impacts, which are 19.9% and 12.9%, respectively. 'Nearside' impacts are at 12.4%, while there are 8.4% of cases with 'Nothing' reported.

Count Plot:

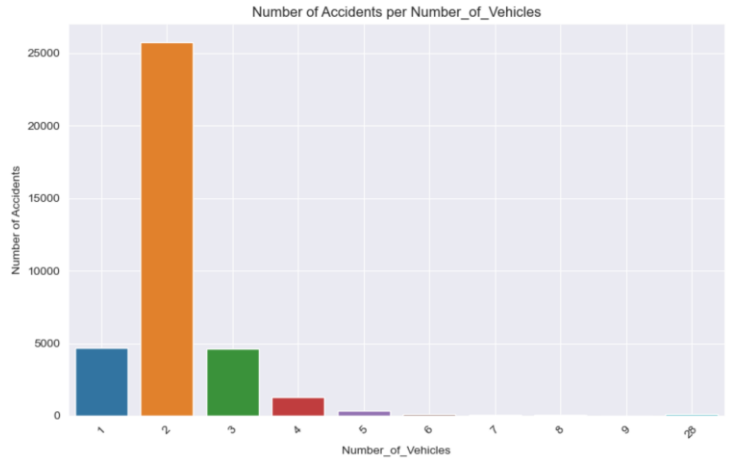

Fig. 32: Count plot of no. of vehicles affected.

Observations: The count plot illustrates the number of accidents correlated with the number of vehicles involved. Accidents involving two vehicles are the most common by a significant margin, with the bar reaching close to 25,000 incidents. Single-vehicle accidents follow, indicated by a blue bar, but with substantially fewer cases. The frequency of accidents decreases as the number of vehicles involved increases, with three or more vehicle accidents being relatively rare.

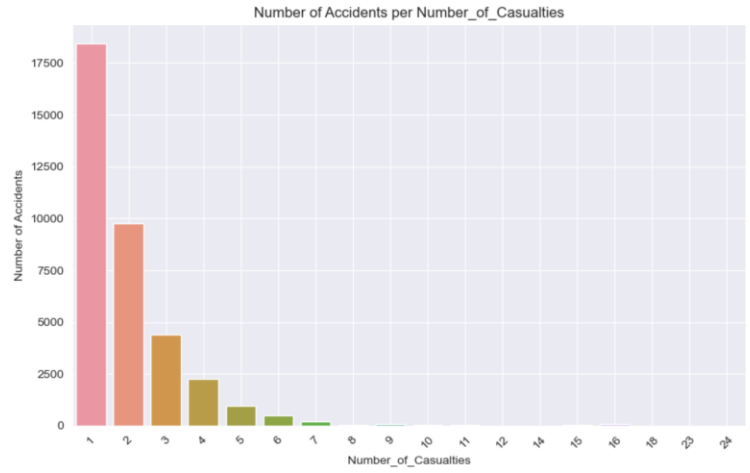

Fig. 33: Count plot of no. of vehicles affected.

Observations: The count plot depicts the number of accidents sorted by the number of casualties involved. Single-casualty incidents dominate, with the count significantly higher than for other categories. There's a clear decreasing trend as the number of casualties per

accident increases, with two casualties per accident being the second most common scenario. The frequency of accidents continues to decline sharply for three or more casualties, indicating such events are much less common.

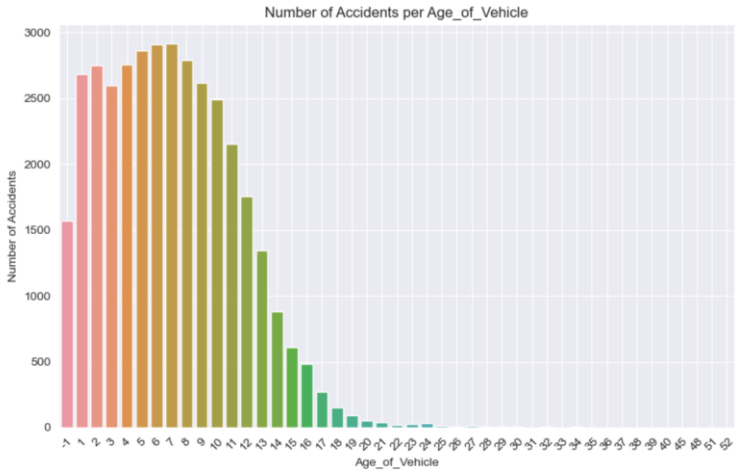

Fig. 34: Count plot of type of vehicles affected in accidents

Observations: The count plot represents the number of accidents in relation to the age of vehicles involved. The trend suggests a decrease in the number of accidents as the age of the vehicle increases. The highest number of accidents involves vehicles that are 0-1 years old, with a gradual decline observed as vehicles age. Vehicles aged 6 years or older are involved in significantly fewer accidents.

Pair Plot:

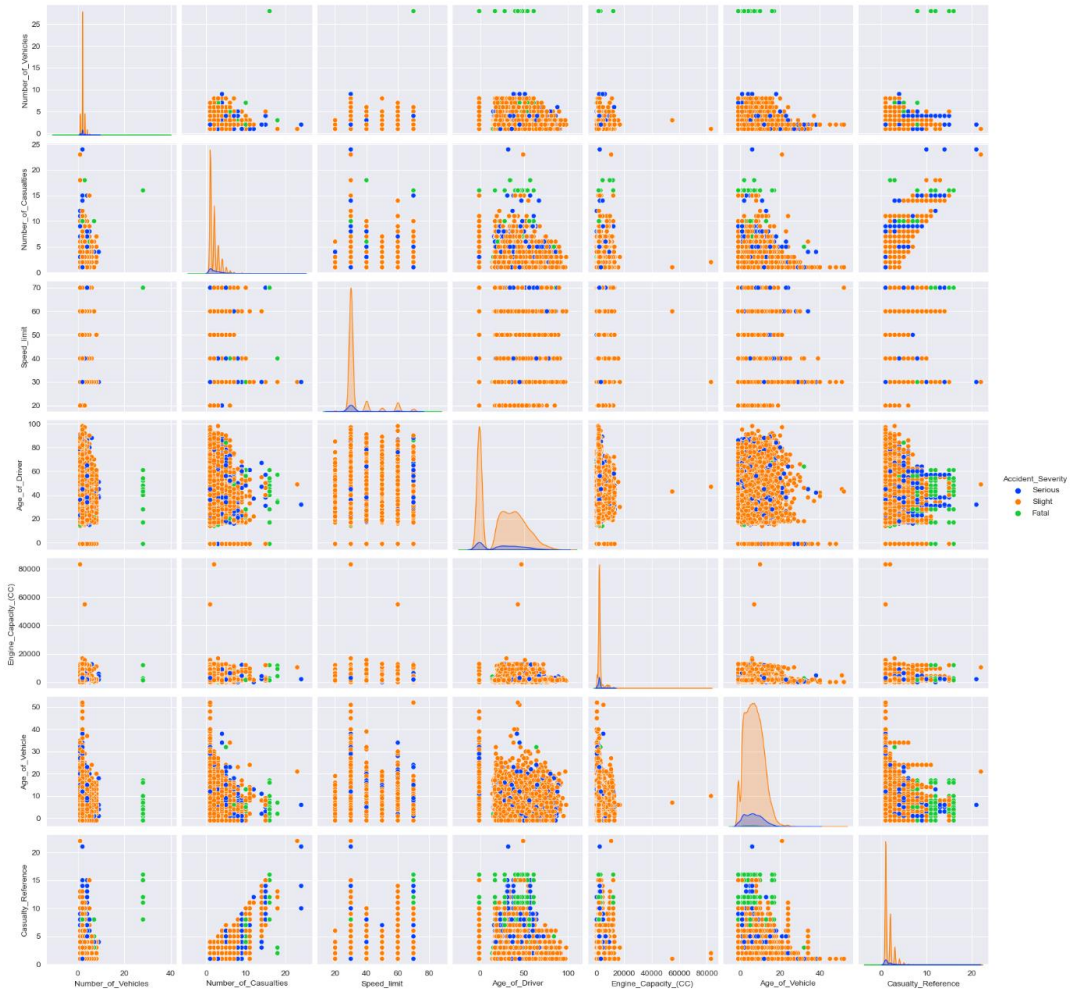

Fig. 35: Pairplot of numerical features

Observations: the pairplot shows the comprehensive overview of how each variable in a dataset relates to the others in the dataset. The diagonal plots, typically histograms, show the distribution of individual variables, which can reveal skewness, peaks, and the range of values. The scatter plots show how variables correlate with each other. Tight clusters may indicate strong correlations, while more dispersed points suggest weaker relationships. Any points that lie far away from the others could be outliers. Here we can see the engine capacity has the outliers.

Heatmap:

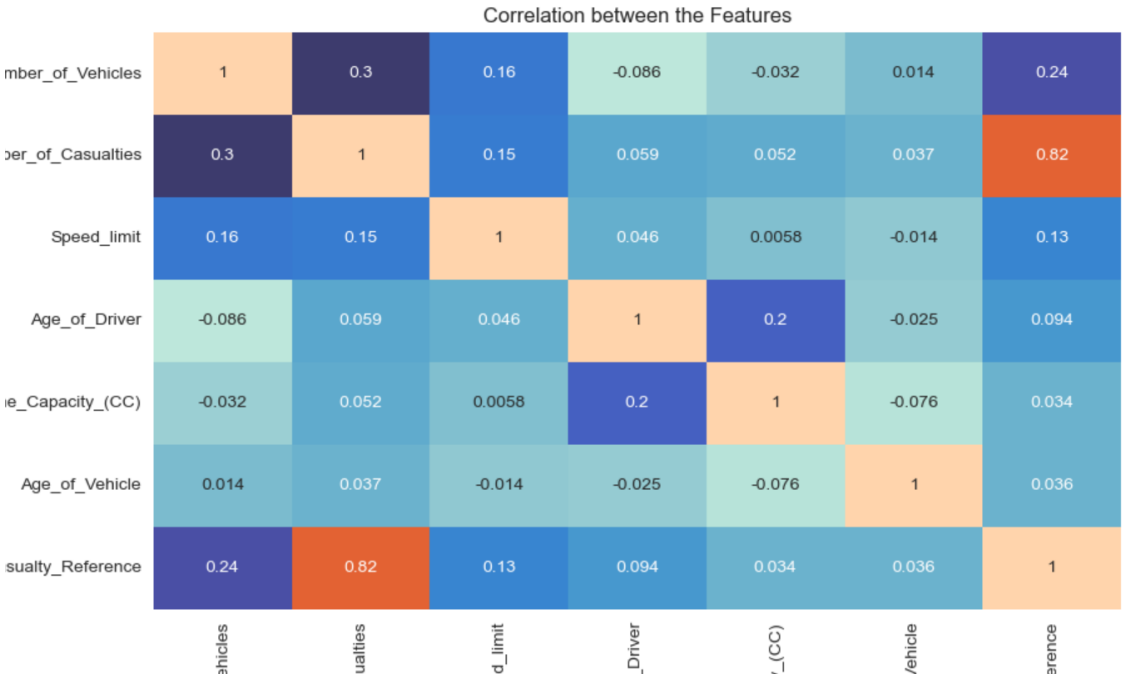

Fig. 36: Heatmap of numerical features

Observations: The heatmap visualizes the correlation coefficients between various variables related to vehicle accidents. The values range from -1 to 1, where 1 indicates a perfect positive correlation, -1 a perfect negative correlation, and 0 no correlation. The strongest positive correlation is between the number of vehicles and casualty reference, while a notable positive correlation is also observed between the number of casualties and casualty reference. Most variables show low to moderate correlation with each other, as indicated by the mix of warm and cool colors.

Subplots:

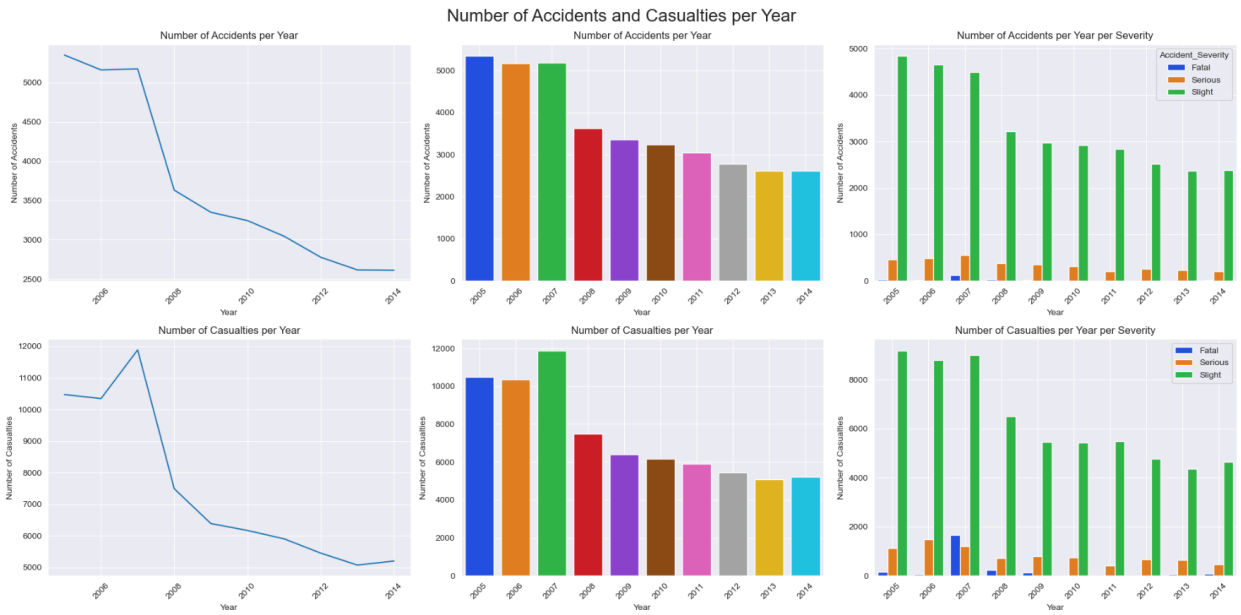

Fig. 37: Subplot of number of accidents per year and no.of casualties per year

Observations:

1. **Number of Accidents per Year:** These line and bar charts show a decreasing trend in the number of accidents over time, with a sharp decline after the initial year shown.
2. **Number of Casualties per Year:** Similar to accidents, the number of casualties per year also decreases. There's an initial spike, followed by a decline.
3. **Number of Accidents per Year per Severity:**This bar chart breaks down accidents by severity (fatal, serious, slight) per year. It shows that slight accidents are the most common, followed by serious, and then fatal, which are the least common.
4. **Number of Casualties per Year per Severity:** This chart categorizes casualties by the same severity categories and displays a consistent pattern of more slight casualties compared to serious and fatal ones.

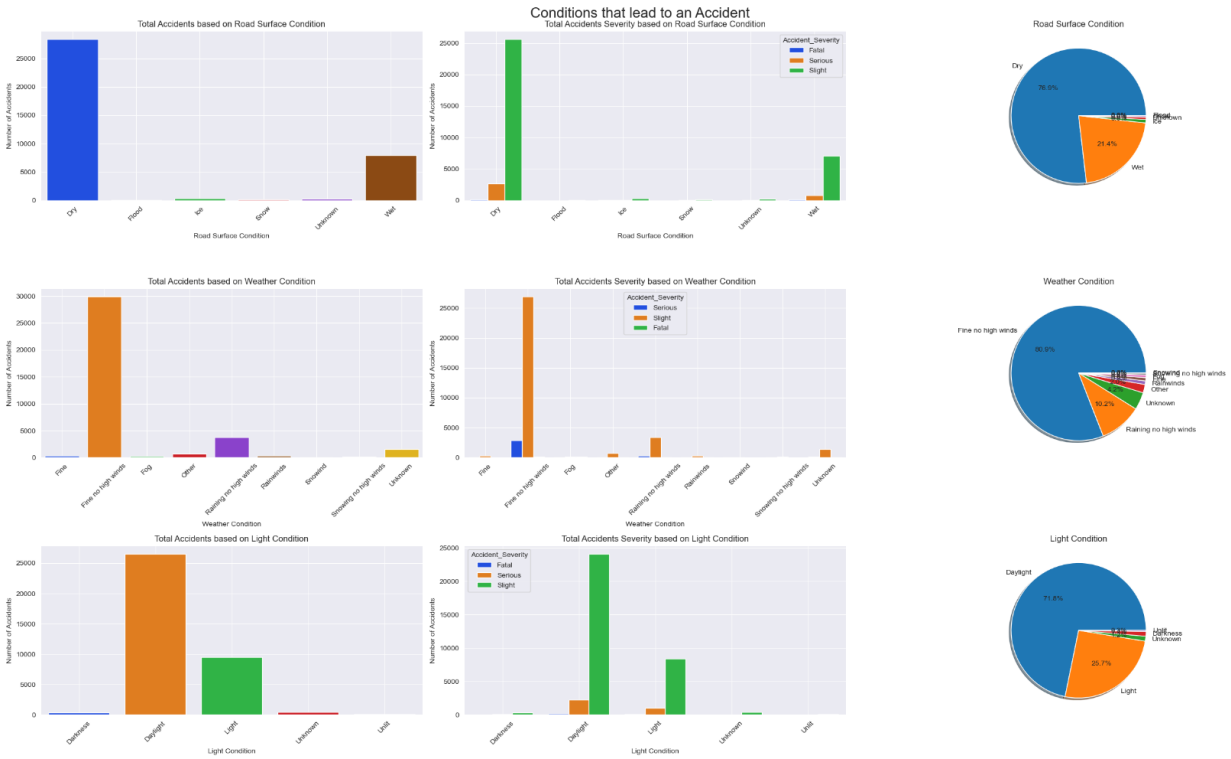

Fig. 38: Subplot for the different conditions that lead to an accident.

Observations:

1. **Road Surface Condition:** The bar graph shows that the majority of accidents occur on dry road conditions, with wet conditions being the second most common.
2. **Weather Condition:** The bar graph indicates most accidents occur in fine weather, with other conditions like rain and snow leading to fewer accidents.
3. **Light Condition:** Most accidents happen in daylight, with dark but lighted conditions being the second most common scenario for accidents.
4. **Severity Based on Road Surface Condition:** The stacked bar graph details that most accidents on all types of road surfaces are of slight severity, with serious and fatal accidents occurring less frequently.
5. **Severity Based on Weather Condition:** Similarly, this graph illustrates that in all weather conditions, slight accidents are the most common.
6. **Severity Based on Light Condition:** Daylight sees primarily slight accidents, with serious and fatal accidents following.

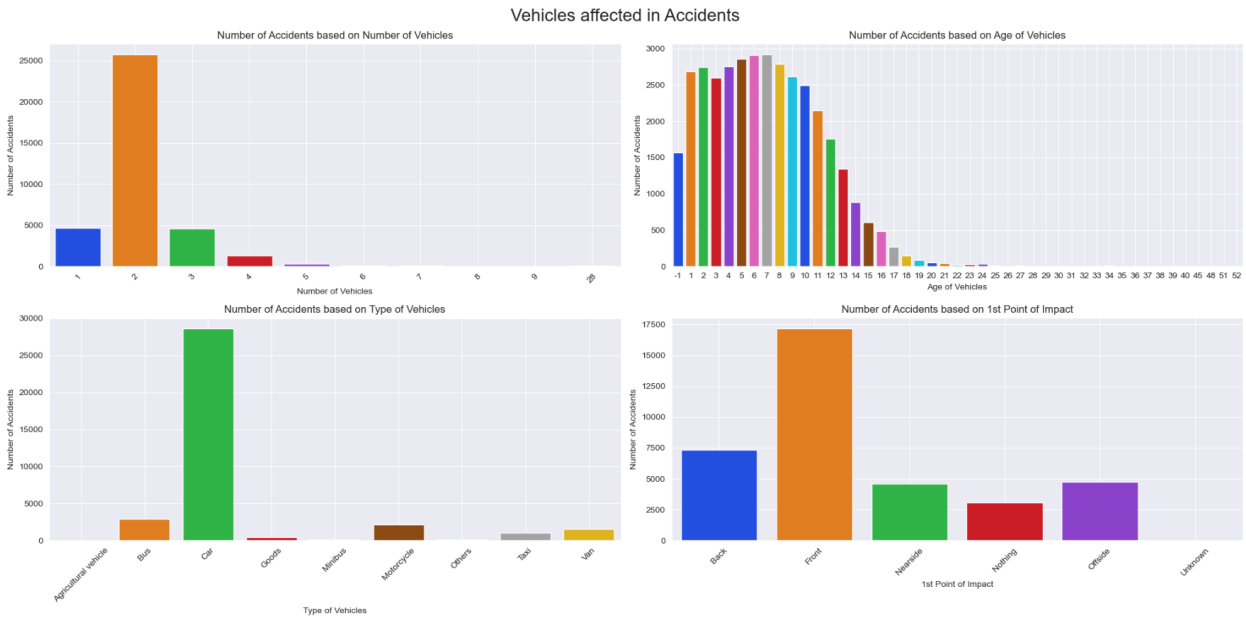

Fig. 39: Subplot depicting the damage to the vehicles.

Observations:

1. **Number of Accidents based on Number of Vehicles:** This bar chart indicates that accidents involving two vehicles are the most common, with a steep drop-off in accidents as the number of vehicles involved increases.
2. **Number of Accidents based on Age of Vehicles:** The bar chart shows that newer vehicles (0-3 years old) are involved in more accidents than older vehicles, with a gradual decline as the age increases.
3. **Number of Accidents based on Type of Vehicles:** Most accidents involve cars, with the number involving other types of vehicles like buses, motorcycles, and goods vehicles being significantly lower.
4. **Number of Accidents based on 1st Point of Impact:** The bar chart shows that the front of vehicles is the most common first point of impact in accidents, followed by the rear, nearside, and offside, with a small number of cases where the first point of impact is unknown.

Displot, Histogram with KDE, KDE plot, Q-Q plot:

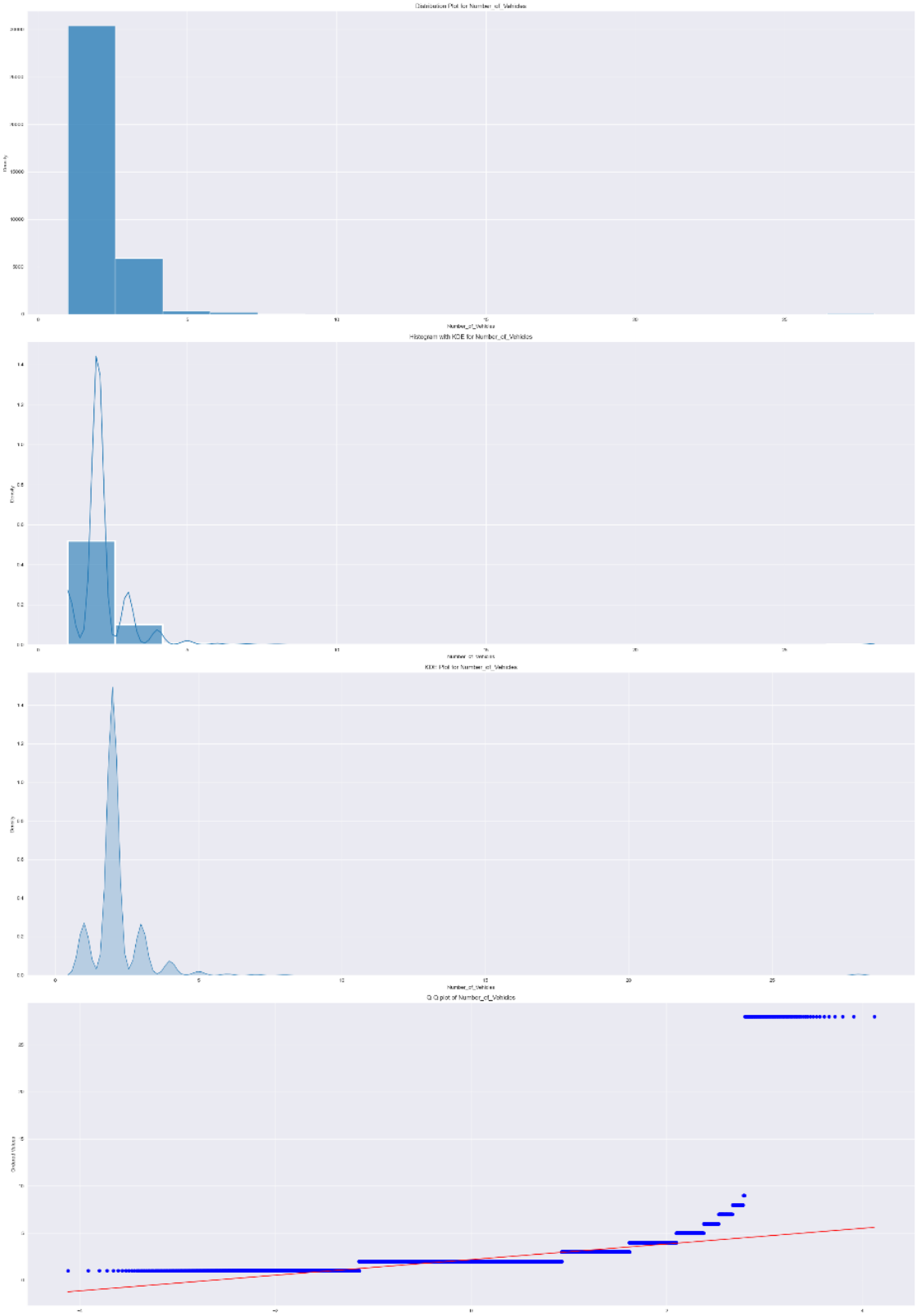

Fig. 40: Displot, histogram with KDE, KDE plot, Q-Q plot of no. of vehicles

Observations:

1. **Displot**: Shows a highly skewed distribution with a strong peak at '1', indicating that most of the accidents involve a single vehicle. The frequency drastically decreases as the number of vehicles increases.
2. **Histogram with KDE**: The KDE curve emphasizes the skewness of the data, with a peak at '1' that rapidly tapers off, suggesting that single-vehicle accidents are far more common than multi-vehicle accidents.
3. **KDE Plot**: Offers a smooth representation of the data distribution. The sharp peak at '1' reaffirms the prevalence of accidents involving a single vehicle, with very low probability of accidents involving more than one vehicle.
4. **QQ Plot**: Compares the distribution of 'Number of Vehicles' to a normal distribution. The pronounced deviation from the straight line indicates that the data is not normally distributed, with a skew towards the lower end.

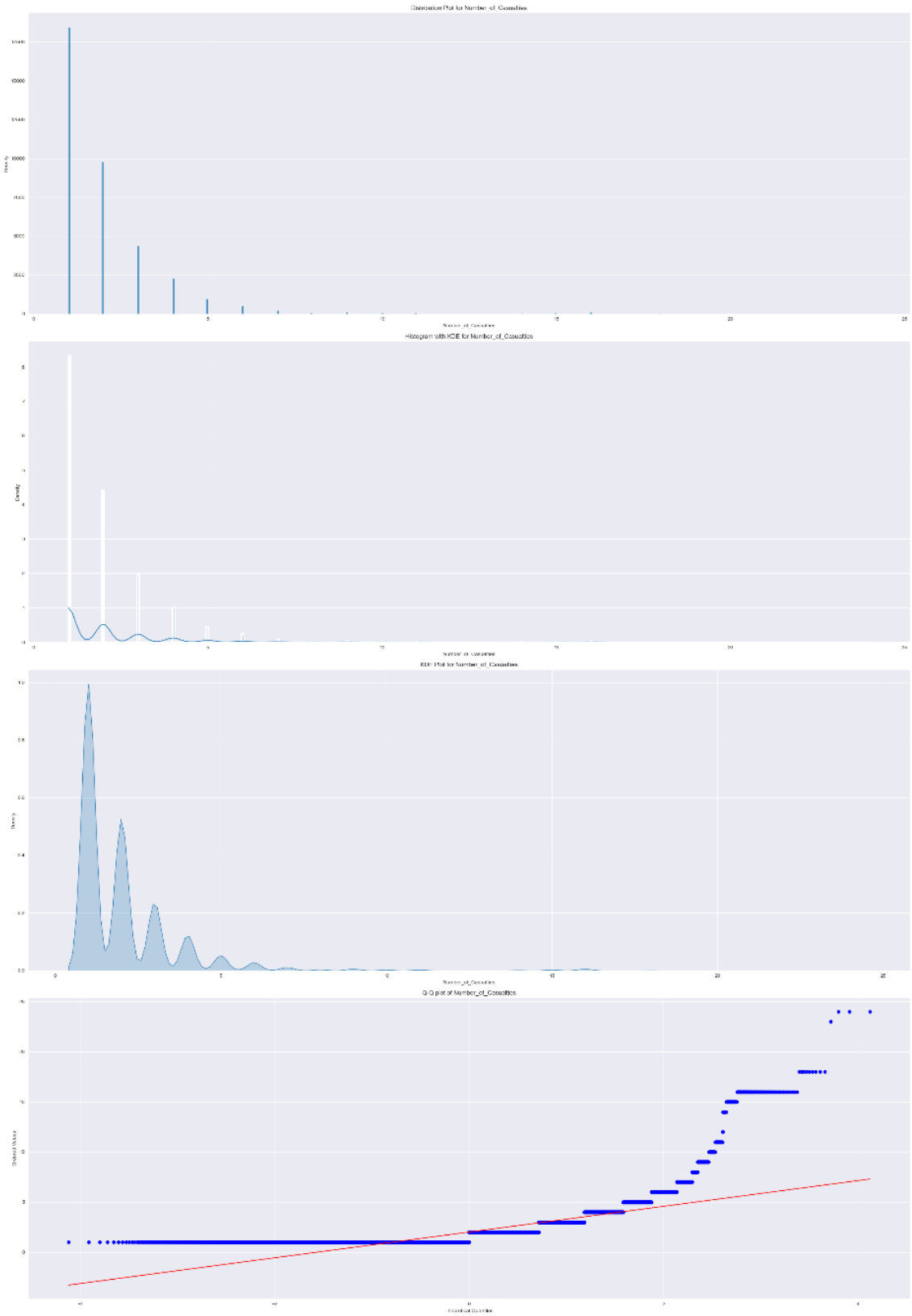

Fig. 41: Displot, histogram with KDE, KDE plot, Q-Q plot of no. of casualties

Observations:

1. **Displot**: Displays a steep decline in frequency as the number of casualties increases, with the highest number of incidents involving only one casualty.
2. **Histogram with KDE**: The KDE curve underscores the skewness towards accidents with fewer casualties, peaking sharply at one casualty and tapering off quickly thereafter.
3. **KDE Plot**: This smoothed curve highlights the concentration of data around lower casualty numbers, indicating that incidents with a high number of casualties are rare.
4. **QQ Plot**: This plot indicates that the 'Number of Casualties' data does not follow a normal distribution, as evidenced by the significant deviation from the straight line, particularly with higher casualty numbers.

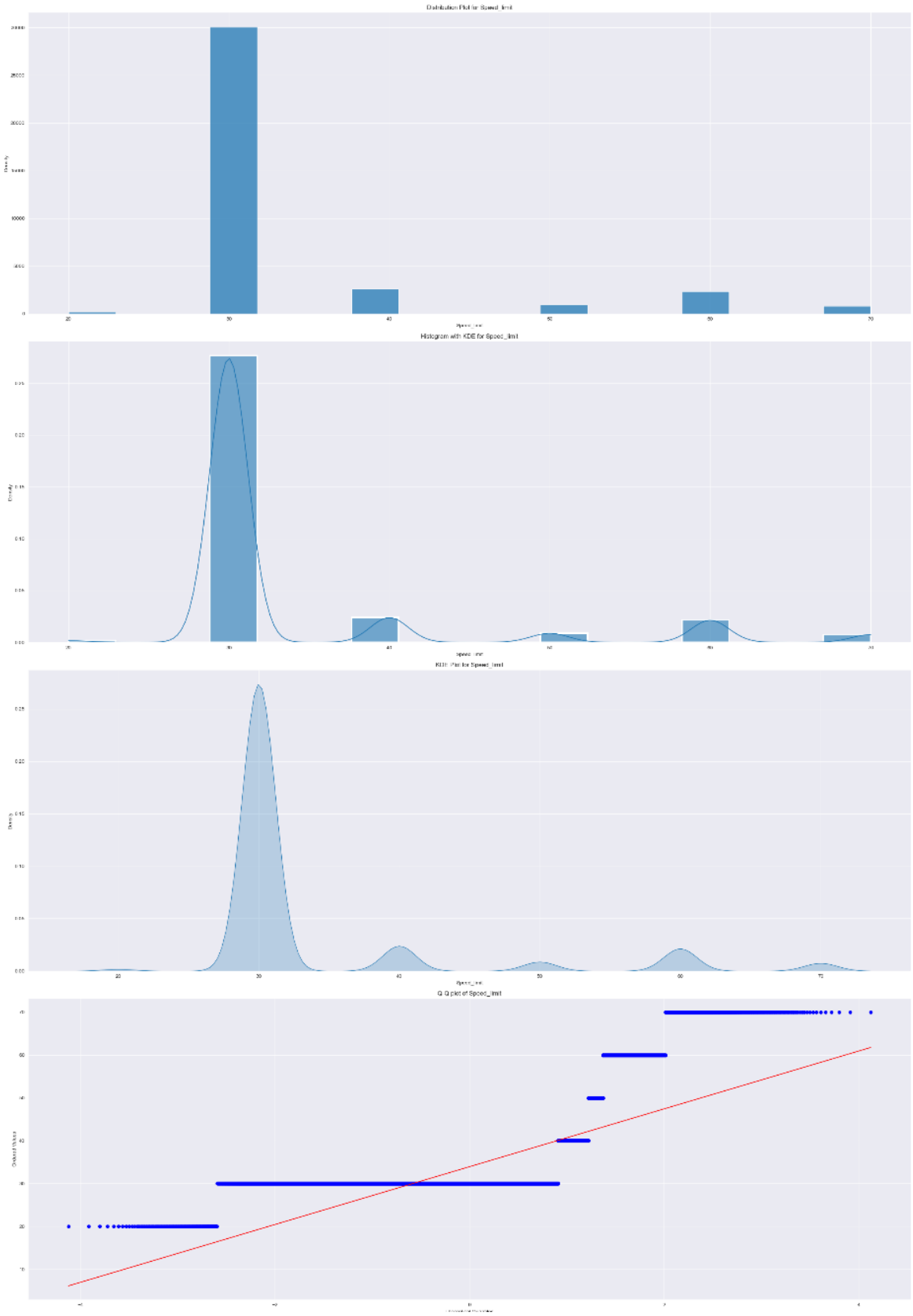

Fig. 42: Displot, histogram with KDE, KDE plot, Q-Q plot of speed limit

Observations:

1. **Displot**: This plot shows that most data points are concentrated around a specific speed limit value, which appears to be the most common speed limit where accidents occur. The distribution is sparse for higher and lower speed limits.
2. **Histogram with KDE**: The KDE overlay on the histogram shows a clear peak at this common speed limit, suggesting that most accidents occur at this particular speed limit.
3. **KDE Plot**: The plot presents a smoothed probability density of the speed limit data, again with a sharp peak at the common speed limit, and smaller peaks at higher speed limits, indicating less frequent occurrences of accidents at those speeds.
4. **QQ Plot**: The deviation of the plotted data points from the red line indicates that the distribution of speed limits in the data does not follow a normal distribution, especially at the tails of the distribution.

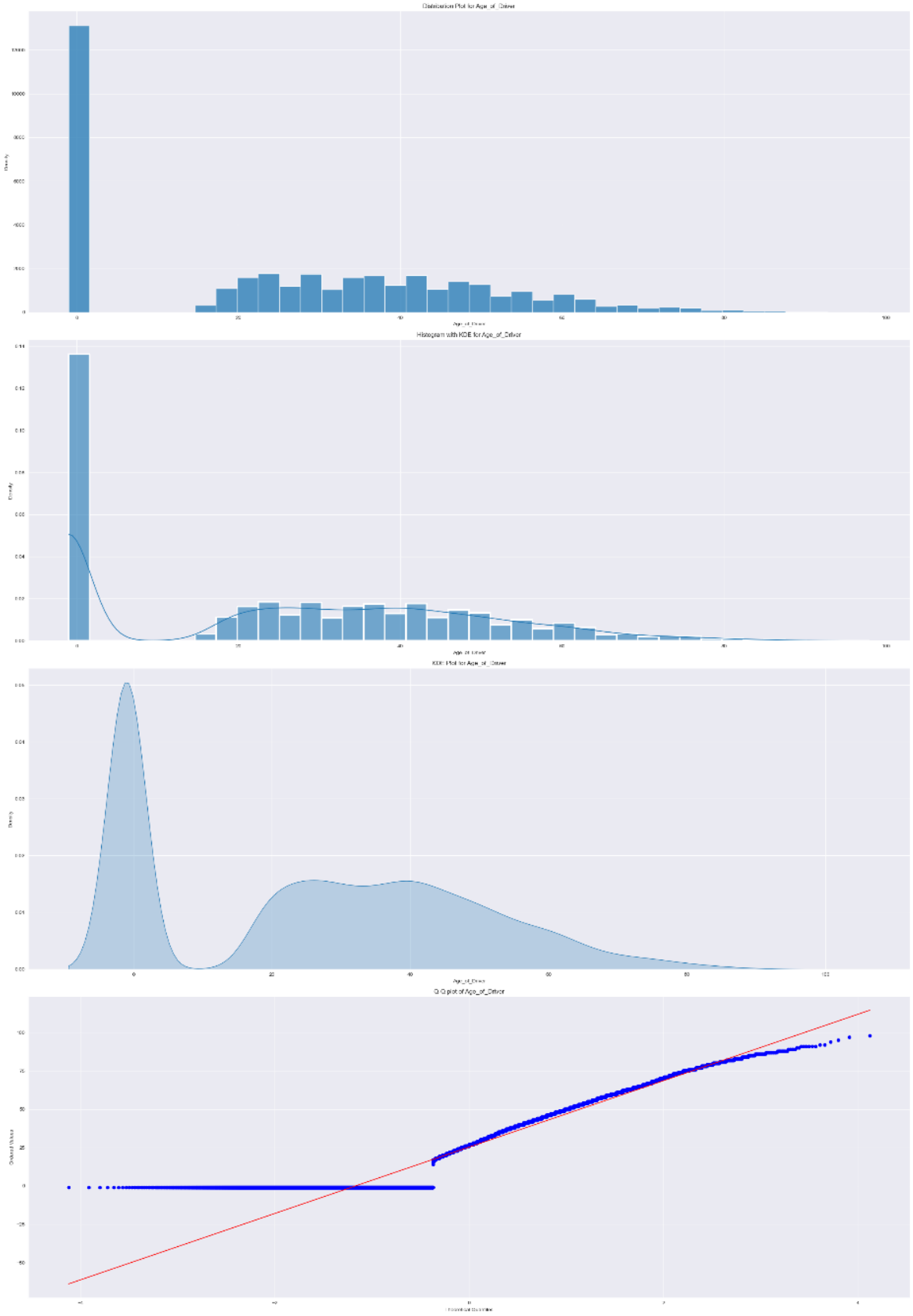

Fig. 43: Displot, histogram with KDE, KDE plot, Q-Q plot of age of driver

Observations:

1. **Displot**: The data shows many young drivers involved in accidents, with the frequency of accidents decreasing with the age of the driver.
2. **Histogram with KDE**: This plot further illustrates the distribution, with a high peak at the lower age range, indicating a high frequency of accidents among younger drivers.
3. **KDE Plot**: The curve shows a significant peak at the lower end, highlighting that most drivers involved in accidents are young, with a long tail indicating that accidents involving older drivers occur but are less frequent.
4. **QQ Plot**: The distribution of the age of drivers is not normal, as shown by the deviation from the red line, particularly at the lower and higher ends of the age range, which may indicate a larger number of young drivers and a smaller number of older drivers involved in accidents than what would be expected in a normal distribution.

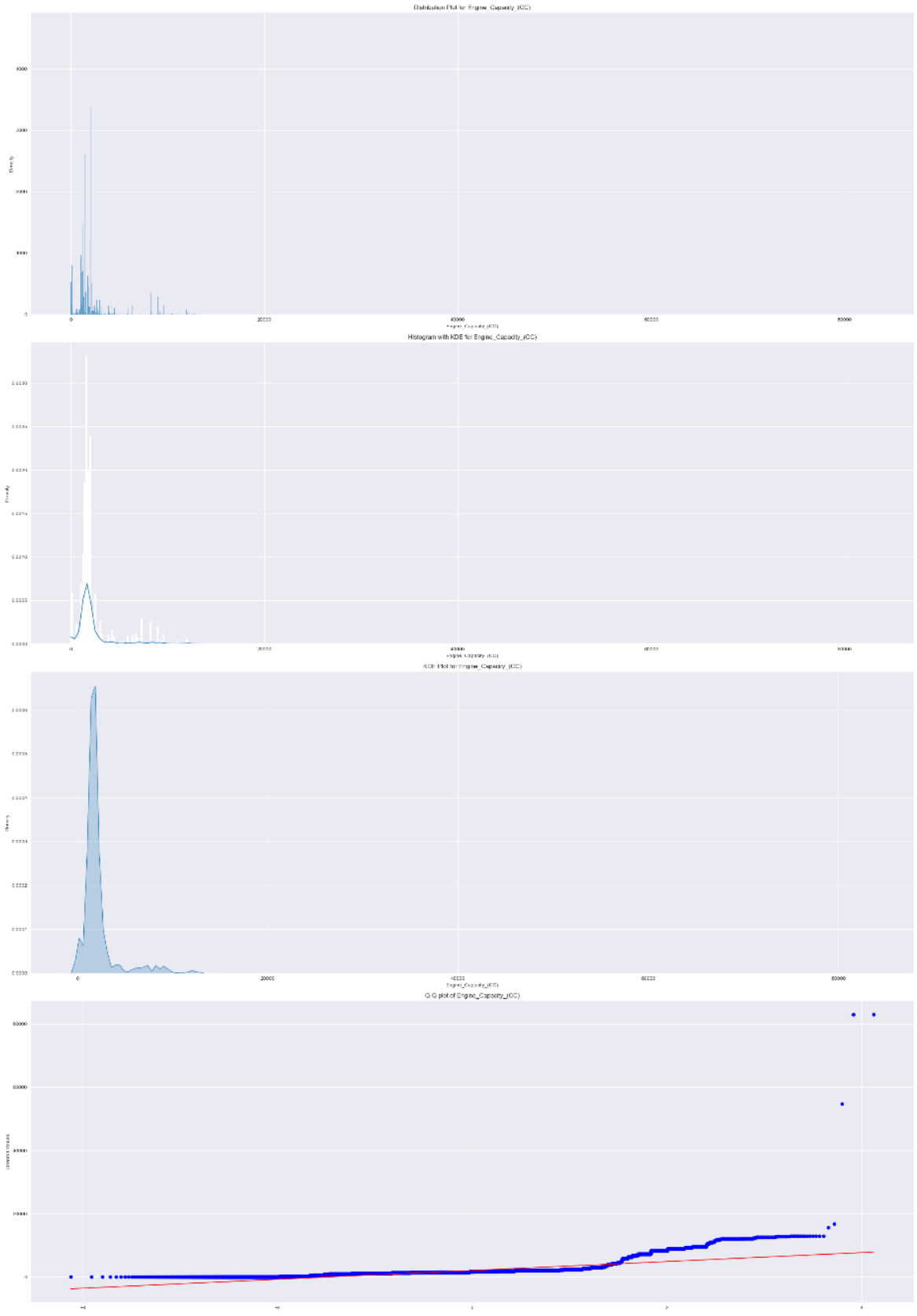

Fig. 44: Displot, histogram with KDE, KDE plot, Q-Q plot of engine capacity

Observations:

1. **Displot**: This plot shows a highly concentrated number of accidents occurring with vehicles having a lower engine capacity, with the frequency dramatically decreasing as the engine capacity increases.
2. **Histogram with KDE**: The KDE curve on this histogram suggests that the majority of accidents involve vehicles with smaller engine capacities, highlighting a steep peak at the lower end of the engine capacity range.
3. **KDE Plot**: The KDE plot reveals a pronounced peak at lower engine capacities, with a long tail towards the higher capacities, suggesting that accidents involving vehicles with high engine capacities are relatively rare.
4. **QQ Plot**: The data points significantly deviate from the straight line, especially at the lower end, indicating that the distribution of engine capacities in accidents does not follow a normal distribution and is skewed towards vehicles with lower engine capacities.

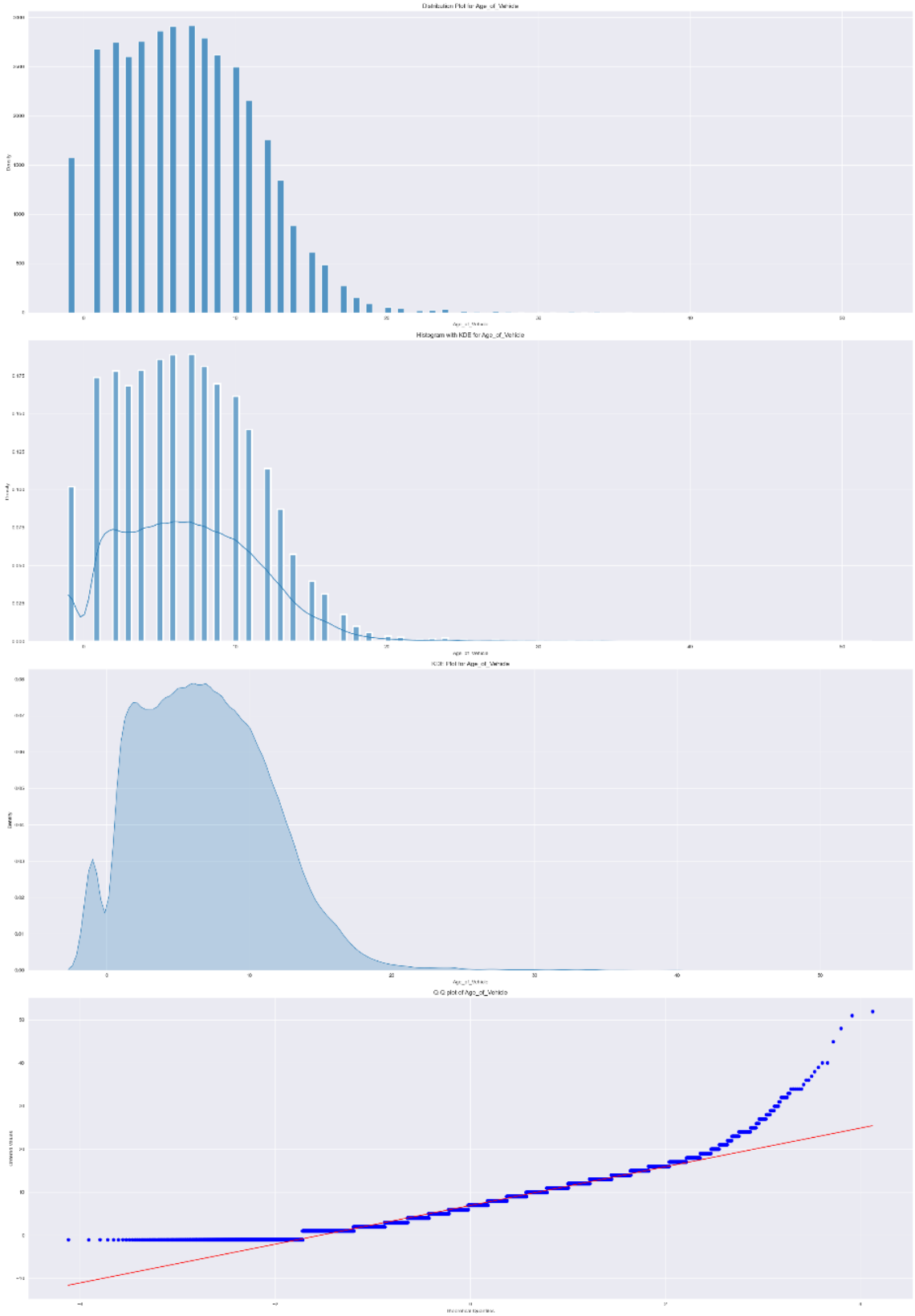

Fig. 45: Displot, histogram with KDE, KDE plot, Q-Q plot of age of vehicle

Observations:

1. **Displot**: Displays a distribution that skews towards younger vehicle ages, indicating that newer vehicles are more commonly involved in accidents. The number of accidents decreases as the age of the vehicle increases.
2. **Histogram with KDE**: This overlay shows a smooth curve that peaks at the lower end of the vehicle age spectrum, reinforcing the concentration of accidents among newer vehicles.
3. **KDE Plot**: The density plot provides a clear visualization of the probability density of accidents by vehicle age, with a significant peak indicating that vehicles of a particular younger age range are most likely to be involved in accidents.
4. **QQ Plot**: The deviation of the data points from the straight line suggests that the distribution of vehicle ages in accidents is not normally distributed. The data is skewed towards newer vehicles, with fewer older vehicles involved in accidents than would be expected if the data were normally distributed.

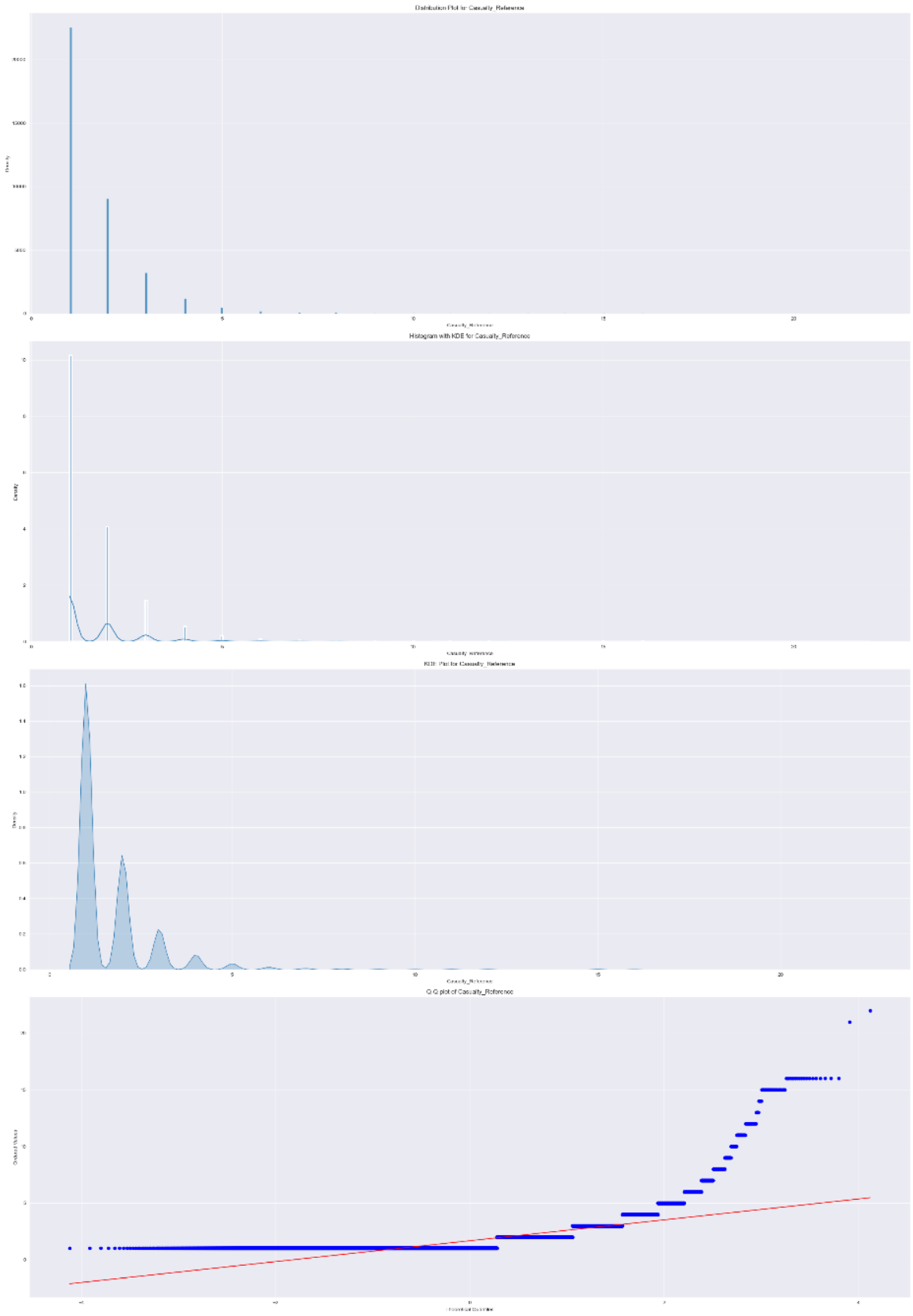

Fig. 46: Displot, histogram with KDE, KDE plot, Q-Q plot of casualty reference

Observations:

1. **Displot**: The histogram shows that the frequency of casualty references is highest at the lower end, indicating that most accidents involve a single casualty or a low number of casualties.
2. **Histogram with KDE**: The KDE overlay suggests that the distribution of casualty references is skewed towards lower numbers, with a peak at the beginning that rapidly decreases.
3. **KDE Plot**: The KDE plot exhibits a pronounced peak for lower casualty reference numbers, confirming that most accidents involve fewer casualties.
4. **QQ Plot**: The QQ plot illustrates that the distribution of casualty references does not follow a normal distribution, with a steep deviation from the expected line at the lower end, which indicates a higher frequency of lower casualty reference numbers than would be expected in a normal distribution.

Area Plots:

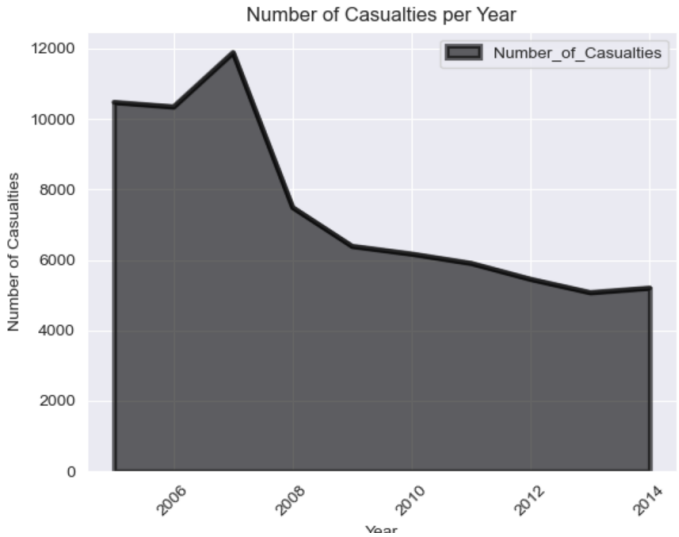

Fig. 47: Area plot of casualties per year

Observations: The area plot depicts the trend in the number of casualties from accidents over a period from 2005 to 2014. The shaded area under the line represents the number of casualties each year. There is a noticeable peak early in the period, followed by a general downward trend. This suggests a decrease in the number of casualties over time, indicating either fewer accidents, safer road conditions, more effective safety measures, or a combination of these factors.

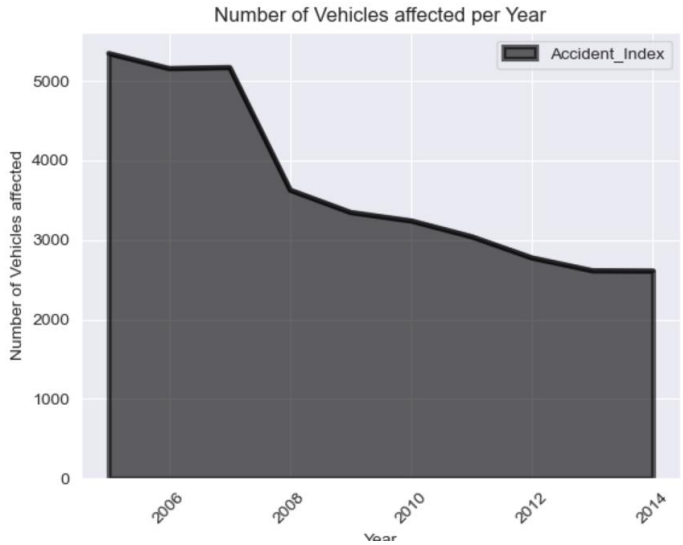

Fig. 48: Area plot of no. of vehicles affected per year.

Observations: The area plot shows the trend in the number of vehicles affected by accidents each year from 2005 to 2014. There is a notable decline over the period, with the most significant drop occurring after 2006. The gradual decrease suggests that there might be improvements in road safety, vehicle safety features, or other factors contributing to fewer vehicles being involved in accidents over time.

Joint plot with KDE and scatter representation:

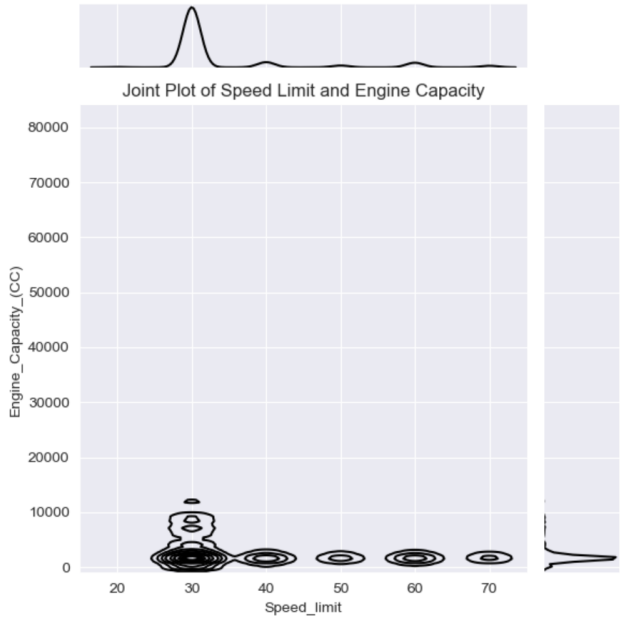

Fig. 49: Joint plot with KDE and scatter representation of speed limit and engine capacity

Observations: This joint plot shows the relationship between speed limit and engine capacity (CC). The concentration of data points is densest at lower speed limits and smaller engine capacities. There are several outliers with high engine capacities across various speed limits. The marginal histograms on the top and right show the univariate distributions of speed limit and engine capacity, respectively, indicating a single peak for speed limits around 30 and a wider distribution for engine capacities with a slight peak around 1000 CC.

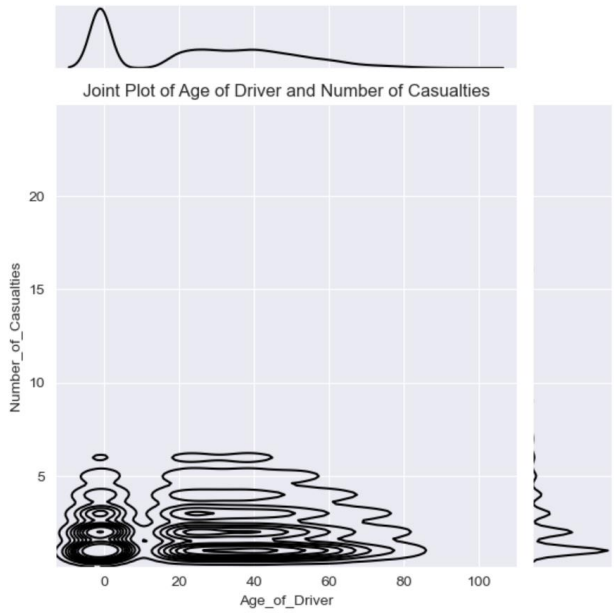

Fig. 50: Joint plot with KDE and scatter representation of age of driver and number of casualties

Observations: The joint plot displays the distribution of the number of casualties in relation to the age of the driver. The data points are most densely packed at younger ages, particularly under 40 years old, with fewer casualties (mostly below 5). There are sporadic occurrences of higher casualty counts across the age spectrum. The marginal histogram for the age of the driver shows a bimodal distribution with peaks around 20 and another smaller peak around 50. The marginal distribution for the number of casualties indicates that instances of 1 or 2 casualties are the most common.

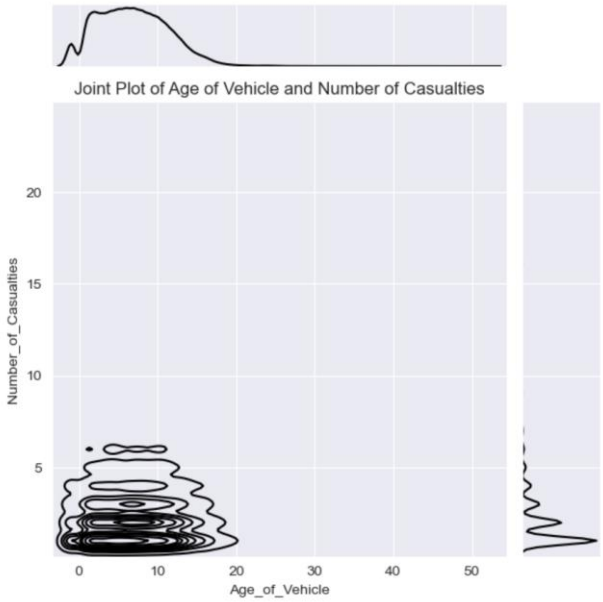

Fig. 51: Joint plot with KDE and scatter representation of age of vehicle and number of casualties

Observations: The joint plot indicates that the number of casualties tends to be lower with newer vehicles, with a dense concentration of data points for vehicles aged 0-10 years. The occurrence of casualties decreases significantly as the age of the vehicle increases beyond 10 years. The marginal histogram for the age of the vehicle shows a decline in frequency as vehicle age increases, with a small peak at 0 indicating the highest number of newer vehicles involved. The distribution for the number of casualties is skewed towards the lower end, with most incidents involving 1 or 2 casualties, as shown by the marginal histogram on the right.

Rug plots:

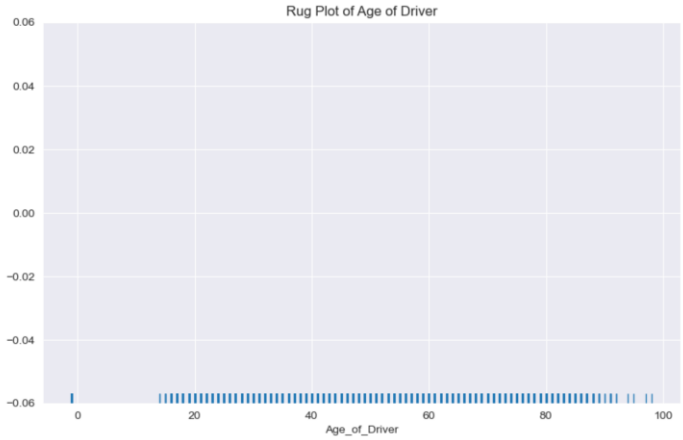

Fig. 52: Rug plot of age of driver

Observations: The rug plot provides a one-dimensional representation of the distribution of drivers' ages. The ticks along the horizontal axis represent individual data points. This plot shows a higher concentration of data points at the younger end of the age spectrum, with a noticeable thinning as age increases. There's a particularly high frequency of drivers in the 20-30 age range, with fewer instances over the age of 60, indicating that the dataset might have a larger proportion of younger drivers.

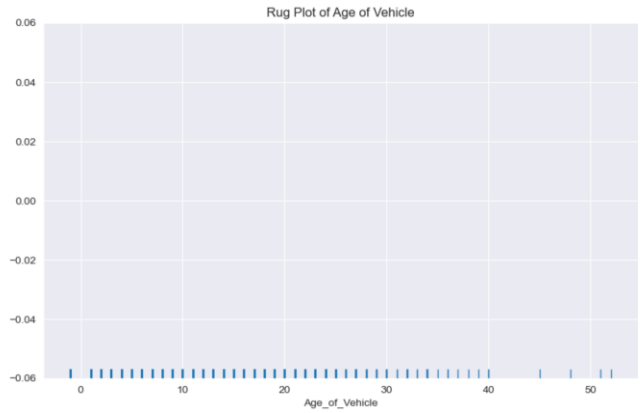

Fig. 53: Rug plot of age of vehicle

Observations: The rug plot shows the age distribution of vehicles. The data points are densely populated at the lower end, indicating a higher frequency of newer vehicles, with very few data points representing vehicles over 10 years old. The graph suggests that the dataset likely contains a greater number of newer vehicles compared to older ones, with the number of vehicles sharply decreasing as age increases. This could imply that either newer vehicles are more commonly in use or that the data collection focused more on newer vehicles.

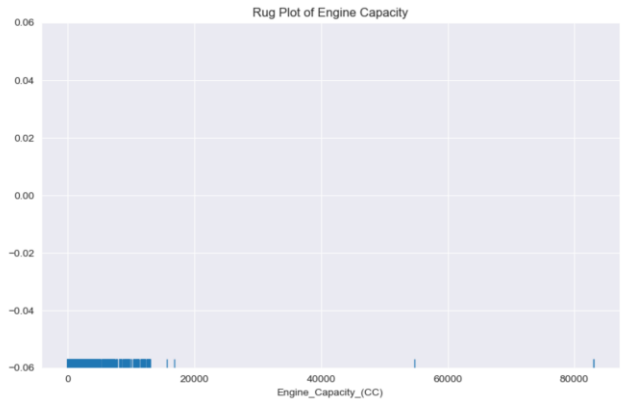

Fig. 54: Rug plot of engine capacity

Observations: The rug plot for engine capacity shows a concentration of data points at the lower end of the engine capacity spectrum, with most vehicles having an engine capacity of 10,000 CC or less. There are very few vehicles with an engine capacity greater than this, as indicated by the sparse ticks. This suggests that the dataset likely contains most vehicles with smaller engine capacities, which are typical for standard consumer vehicles, as opposed to those with larger engines that might be found in commercial or performance vehicles.

Cluster map:

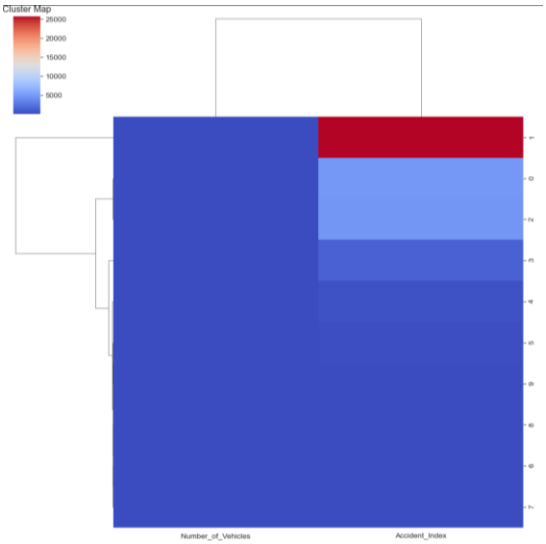

Fig. 55: Cluster map of the dataset

Observations: The image depicts a cluster map, a form of heat map that includes clustering on the sides to show the grouping of similar points. The intensity of the color indicates the number of vehicles involved in accidents, with darker shades representing a higher number, as suggested by the color scale on the left. The red section signifies the highest concentration of data points in that cluster.

Hexbin plots:

Fig. 56: Hexbin plot of age of driver against no. of casualties

Observations: In this Hexbin plot, the Age of Driver is plotted against the Number of Casualties. Most of the data is concentrated at the bottom left corner, indicating that younger drivers are associated with incidents resulting in fewer casualties.

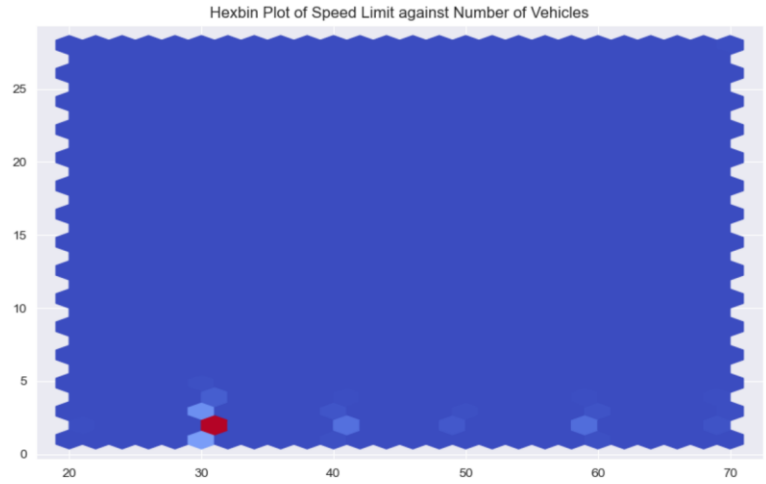

Fig. 57: Hexbin plot of age of driver against no. of casualties

Observations: In this Hexbin plot, Speed Limit is plotted against the Number of Vehicles involved in incidents. There is a concentration of data at the lower end of both axes, with most incidents involving lower speed limits and fewer vehicles. The red hexagon marks a higher density of points, suggesting a particular speed limit where incidents with more vehicles are more frequent.

Strip plots:

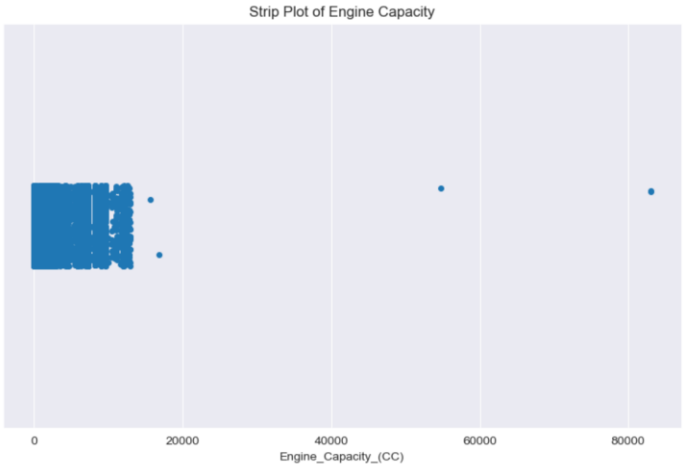

Fig. 58: Strip plot of Engine Capacity

Observations: The strip plot visualizes individual data points for engine capacity (CC) without any binning or aggregation. The data is heavily concentrated at the lower end of the engine capacity range, suggesting that most vehicles in the dataset have a relatively small engine capacity. There are a few isolated points at higher engine capacities, indicating a much smaller number of vehicles with larger engines. The clustering of points near the zero mark may also suggest a high frequency of a specific small engine capacity or a range of capacities within the dataset.

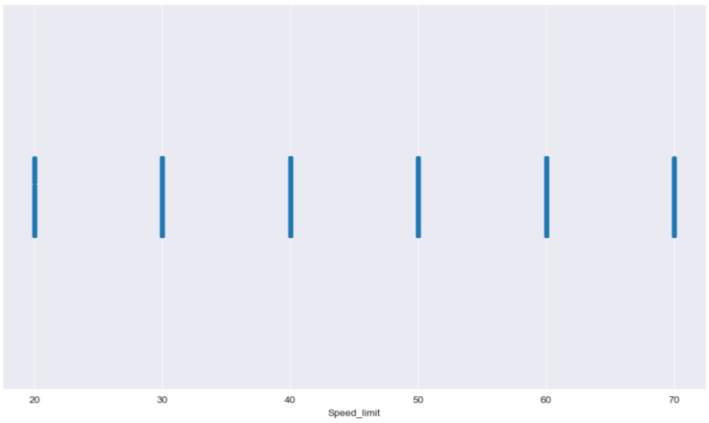

Fig. 59: Strip plot of speed limit

Observations: The strip plot visualizes individual data points for speed limit without any binning or aggregation. The data is not concentrated at all. It is observed that the data is evenly spaced along the x-axis, which corresponds to different speed limit values, and there is no overlap of data points within each category.

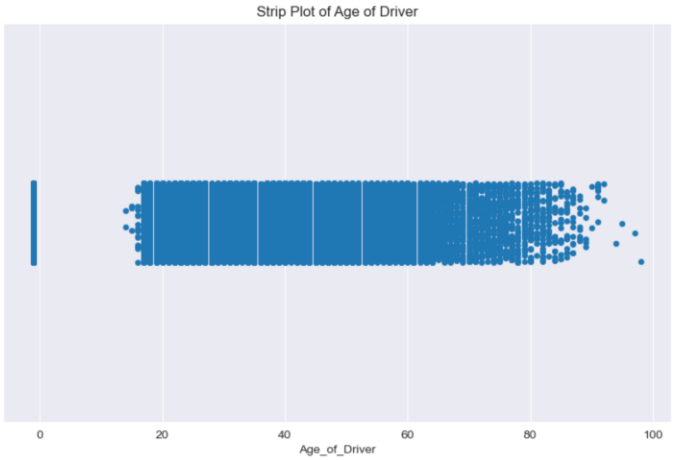

Fig. 60: Strip plot age of driver

Observations: The strip plot displays individual occurrences or data points according to the age of drivers. There is a high density of data points for drivers in the younger age groups, with a gradual decrease in frequency as age increases. The plot suggests that the majority of drivers are between 20 to 40 years old, with fewer drivers in the older age brackets. Additionally, there's a visible scatter of data points at higher ages, indicating variability in the age of drivers but with less frequency compared to the younger age groups.

Violin plots and swarm plots:

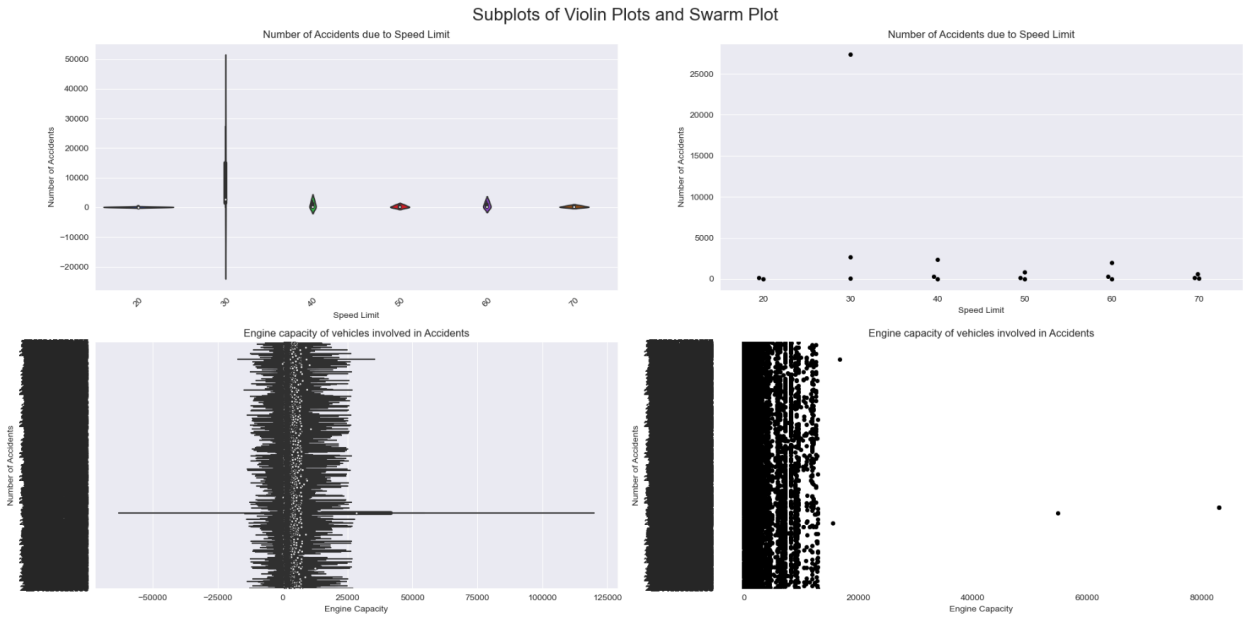

Fig. 61: Violin plots and swarm plots of speed limit and engine capacity

Observations:

Violin Plot: This plot reveals the distribution of accidents across various speed limits, suggesting most accidents occur within specific speed ranges. The widest parts of the violins indicate the speed limits with the highest frequency of accidents.

Violin Plot: The data on engine capacity of vehicles involved in accidents is displayed, which suggests a wide range of engine capacities are involved, with outliers on both ends indicating some accidents involve vehicles with particularly high or low engine capacities.

Swarm Plot: This plot shows accidents spread across different speed limits, which has a concentration of dots at certain speed limits suggests these are common zones where accidents occur, with fewer accidents at the higher speed limits.

Swarm Plot: This plot shows several dense lines indicating common engine capacities involved in accidents. The spread of points suggests variability in the engine capacities involved, which are outliers.

3D plots:

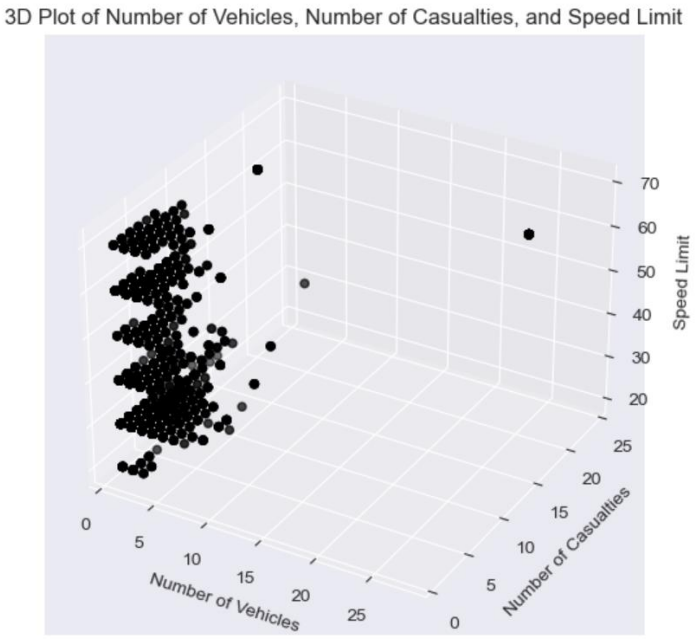

Fig. 62: 3D plot of no. of vehicles, no. of casualties, and speed limit

Observations: The 3D plot displays a concentration of data points where the number of vehicles involved in incidents is low, indicating that most accidents involve fewer vehicles. There's a decrease in the number of incidents as the number of vehicles increases, suggesting a possible relationship between fewer vehicles and higher incidents. Additionally, incidents spread across a range of speed limits, but there's no clear trend visible in this plot between the speed limit and the number of vehicles or casualties involved.

Contour plots:

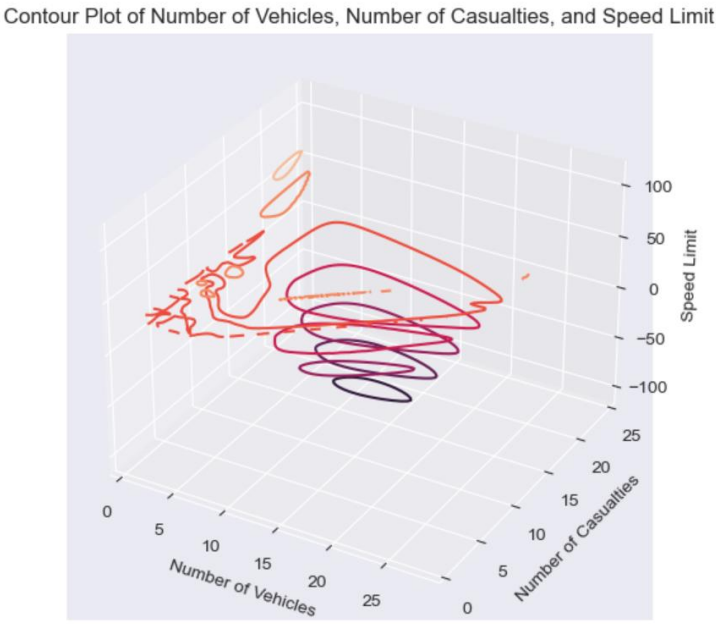

Fig. 63: Contour plot of no. of vehicles, no. of casualties, and speed limit

Observations: The contour plot suggests a high density of incidents involving a smaller number of vehicles and casualties, which is indicated by the tighter contour lines. As the number of vehicles and casualties increases, the contour lines spread out, showing a decrease in incident density.

Scatter Matrix:

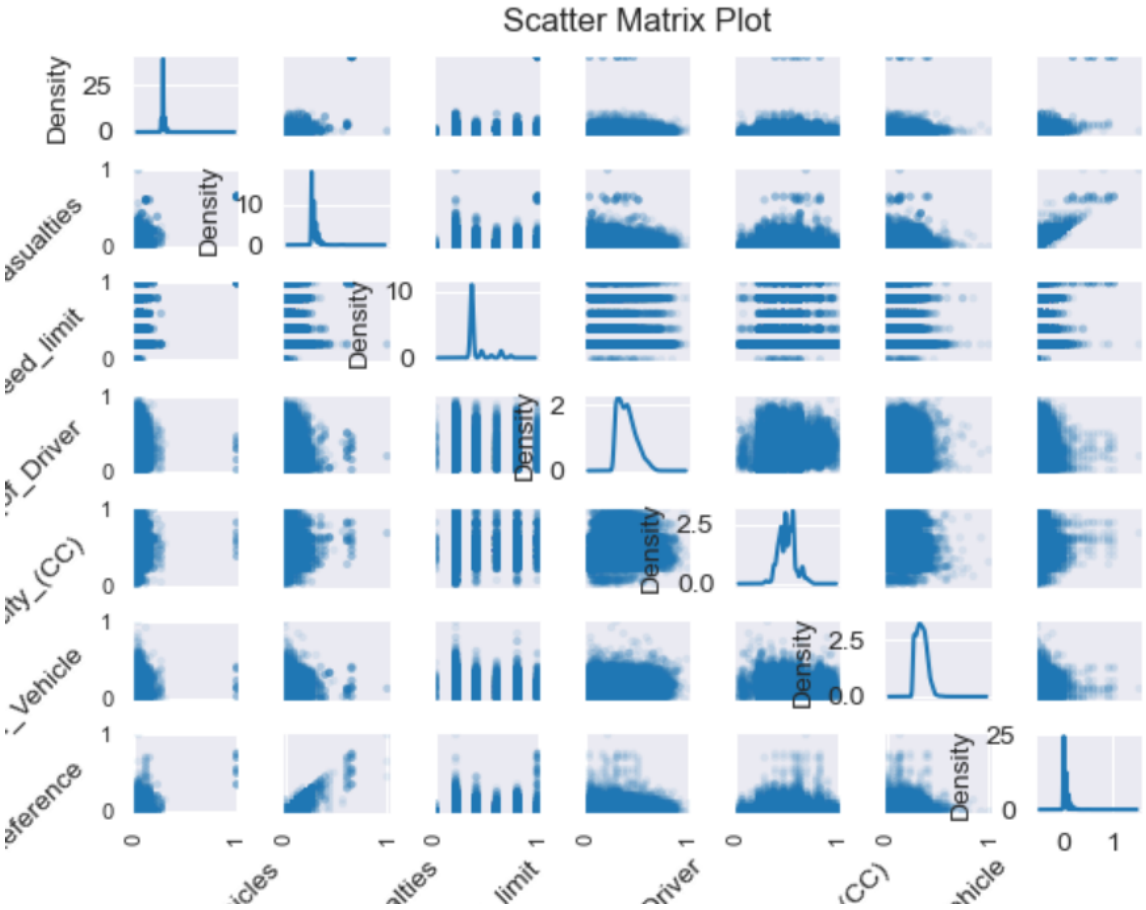

Fig. 64: Scatter Matrix

Observations: The scatter matrix plot displays the pairwise relationships between several continuous variables and their individual distributions. These are:

- The diagonal shows kernel density estimations, revealing the distribution of each variable, which for some appears to be normally distributed, while others are skewed or have a bimodal distribution.
- There is a mix of linear and non-linear relationships between variables, with some showing a clear positive or negative correlation, and others displaying no discernible pattern.
- The plot density indicates varying degrees of correlation strength; denser clusters suggest stronger relationships, while more dispersed plots suggest weaker ones.

Box plots:

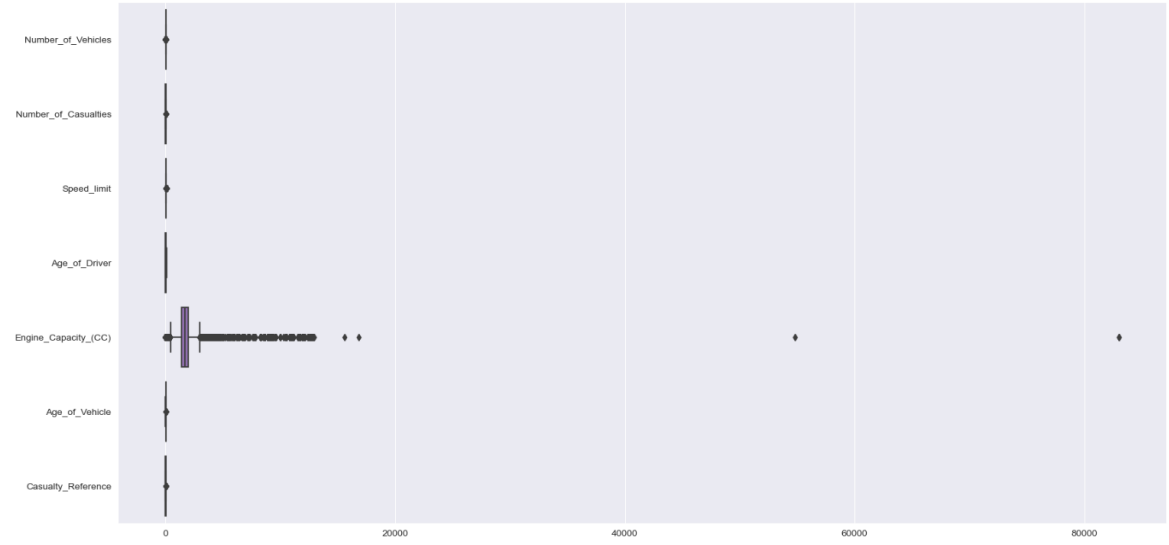

Fig. 65: Box plot of numerical features before Outlier Detection

Observations: The box plot that presents a range of data across various categories, with engine capacity showing the highest number of outliers.

Outlier Detection:

The data is checked for outlier detection. The outliers are removed from the dataset using the “Interquartile Range” method. The interquartile range (IQR) is a statistical measure used to describe the spread or dispersion of data within a dataset. It is a robust measure of variability that is not influenced by extreme outliers or skewed data distributions. The IQR is defined as the range between the first quartile (Q1) and the third quartile (Q3) of a dataset.

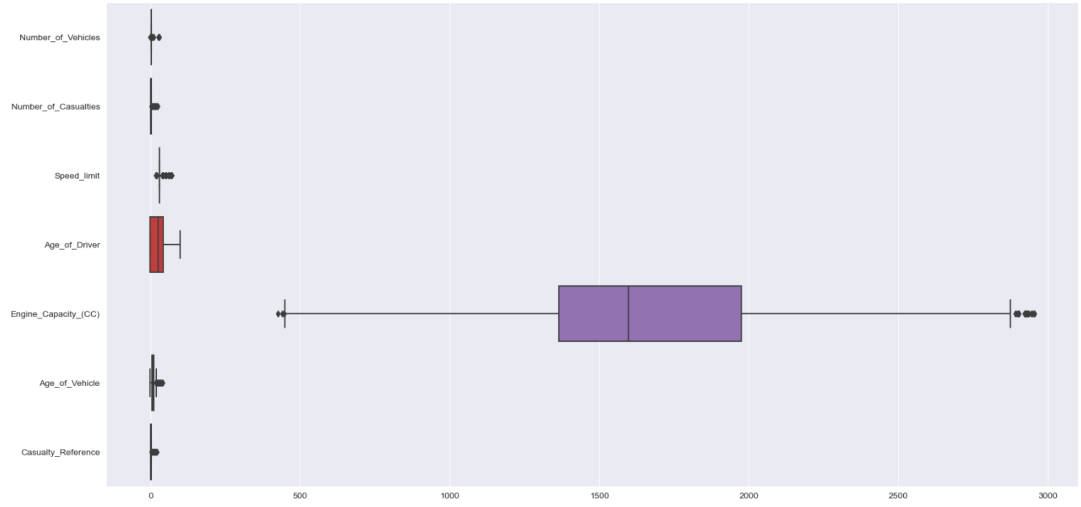

Fig. 66: Box plot of numerical features after Outlier Detection

Observations: The box plot that presents a range of data across various categories. There are some visible outliers in the plot, but these values are not that extreme compared to the former.

Normality Test:

The numerical features in the dataset have been checked for a normal distribution, through the Shapiro test. These are the results:

```
================================================
Shapiro test : Number_of_Vehicles dataset : statistics = 0.29 p-value of =0.00
Number_of_Vehicles dataset is not normal
================================================
Shapiro test : Number_of_Casualties dataset : statistics = 0.63 p-value of =0.00
Number_of_Casualties dataset is not normal
================================================
Shapiro test : Speed_limit dataset : statistics = 0.50 p-value of =0.00
Speed_limit dataset is not normal
================================================
Shapiro test : Age_of_Driver dataset : statistics = 0.89 p-value of =0.00
Age_of_Driver dataset is not normal
================================================
Shapiro test : Engine_Capacity_(CC) dataset : statistics = 0.99 p-value of =0.00
Engine_Capacity_(CC) dataset is not normal
================================================
Shapiro test : Age_of_Vehicle dataset : statistics = 0.98 p-value of =0.00
Age_of_Vehicle dataset is not normal
================================================
Shapiro test : Age_of_Driver dataset : statistics = 0.89 p-value of =0.00
Age_of_Driver dataset is not normal
Best lambda value: -0.19
Best lambda value: -0.66
Best lambda value: -3.22
Best lambda value: 0.04
Best lambda value: 0.67
Best lambda value: 0.55
Best lambda value: 0.04
```

To normalize the data, we can make it non-gaussian data to gaussian data by applying a boxcox transformation.

Transformed data:

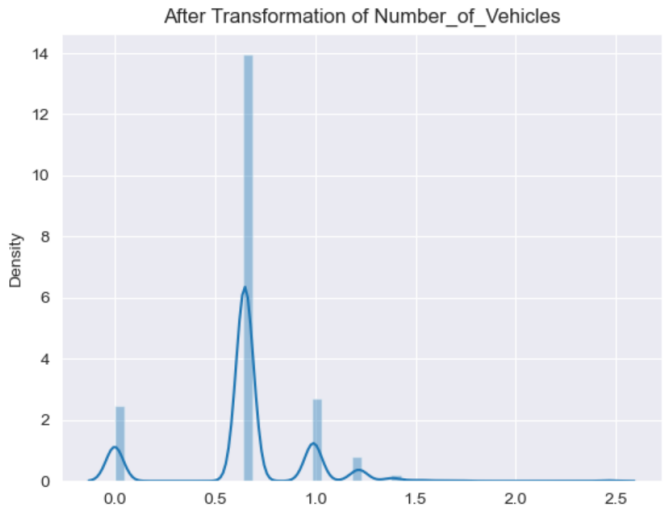

Fig. 67: Transformation of no. of vehicles

Observations: The no. of vehicles feature has been converted from a non-gaussian distribution to a gaussian distribution.

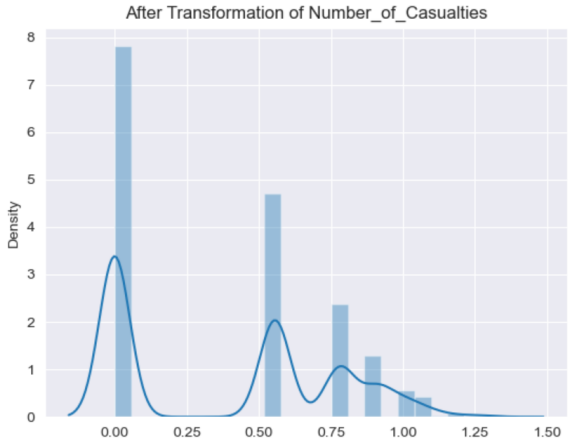

Fig. 68: Transformation of no. of casualties

Observations: The no. of casualties feature has been converted from a non-gaussian distribution to a gaussian distribution.

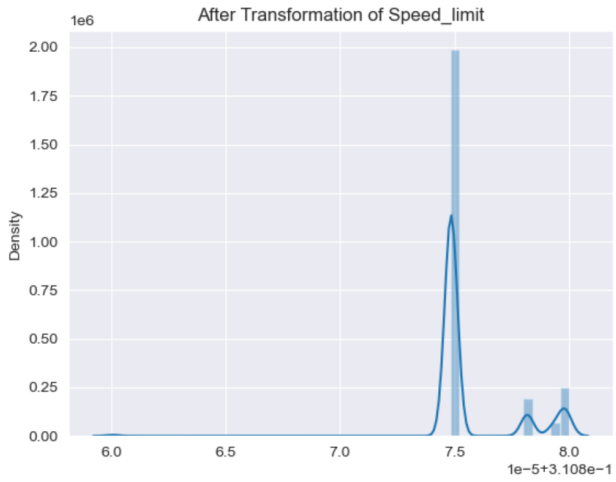

Fig. 69: Transformation of speed limit

Observations: The speed limit feature has been converted from a non-gaussian distribution to a gaussian distribution.

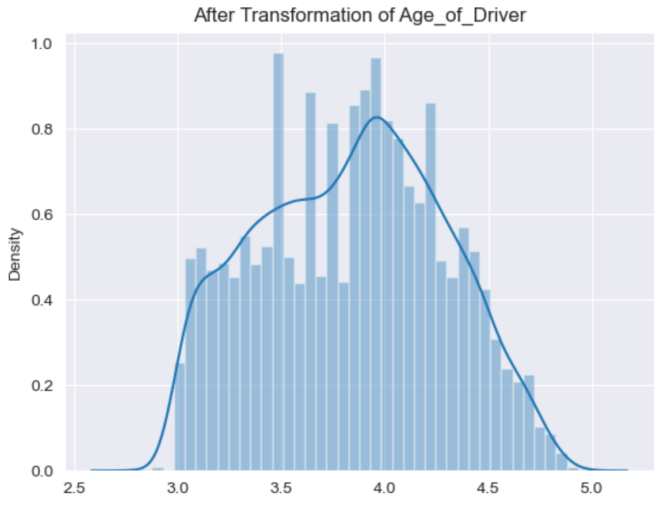

Fig. 70: Transformation of age of driver

Observations: The age of driver feature has been converted from a non-gaussian distribution to a gaussian distribution.

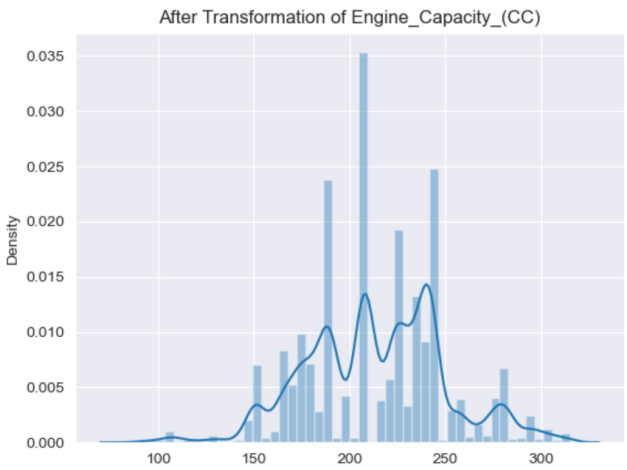

Fig. 71: Transformation of engine capacity

Observations: The engine capacity feature has been converted from a non-gaussian distribution to a gaussian distribution.

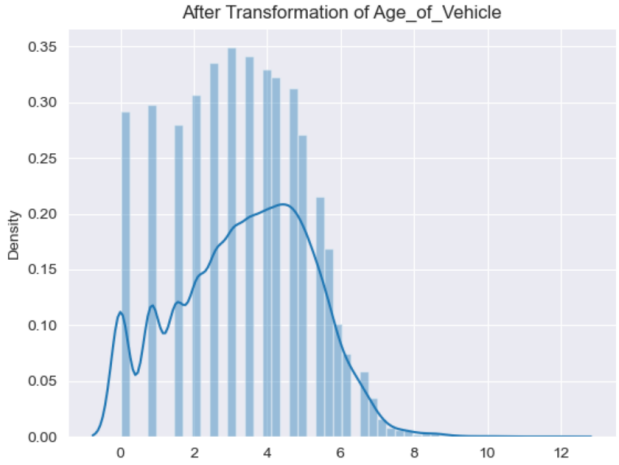

Fig. 72: Transformation of age of vehicle

Observations: The age of vehicle feature has been converted from a non-gaussian distribution to a gaussian distribution.

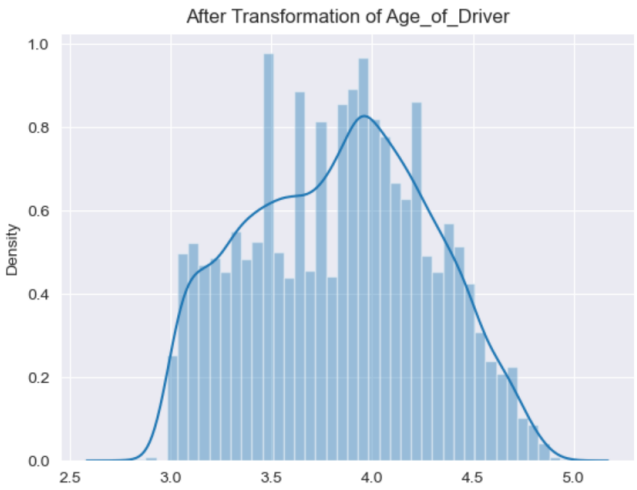

Fig. 73: Transformation of age of driver

Observations: The age of driver feature has been converted from a non-gaussian distribution to a gaussian distribution.

The data now is transformed such that the dataset contains all the numerical features by:

- Encoding the categorical features through one-hot encoding
- Scaling the numerical features through the scale function

Data Balancing:

In many machine learning applications, balancing the target variable is essential because it assures fair and accurate predictions and helps prevent model bias. The model may become biased in favor of the majority class and result in subpar performance on the minority class when the target variable is unbalanced, meaning that one class greatly outnumbers the others. This imbalance may cause the model to mistakenly prioritize accuracy over accurately recognizing uncommon but significant outcomes. Machine learning models are better suited to provide more equitable and dependable predictions across all classes by balancing the target variable, especially in scenarios with imbalanced datasets. This can be done by using appropriate performance metrics like precision-recall instead of accuracy, or by resampling techniques like oversampling the minority class or under sampling the majority class. In this case, the dataset is balanced by using SMOTE, i.e., Synthetic Minority Oversampling Technique, which increases the minority sample in the data, so that the data is balanced.

Here are the results of the data balancing before and after process:

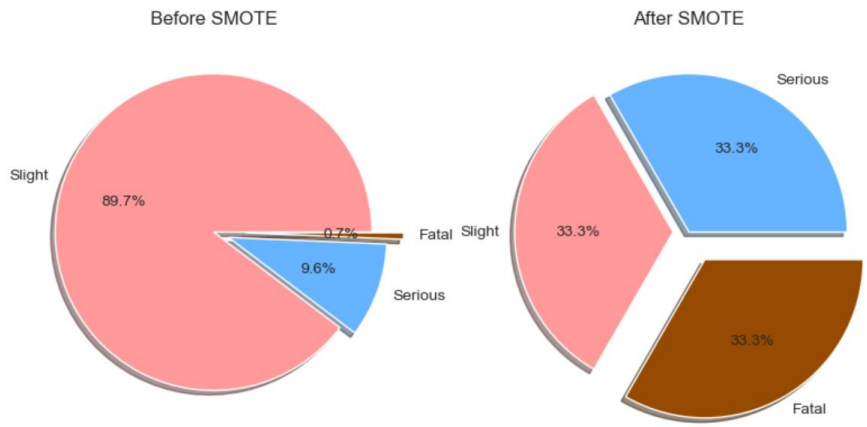

Fig. 74: Data Balancing before and after SMOTE

Observations: The provided pie charts illustrate the distribution of three accident severity categories before and after the application of SMOTE (Synthetic Minority Over-sampling Technique):

- **Before SMOTE**: The chart is overwhelmingly dominated by 'Slight' accidents at 89.7%. 'Serious' accidents make up 9.6%, and 'Fatal' accidents are the least common at only 0.7%.

- **After SMOTE**: The chart shows an equal distribution among the three accident severities, with each category now making up exactly one-third (33.3%) of the data.

Principle Component Analysis:

Principal Component Analysis (PCA) is a dimensionality reduction technique widely used in data analysis and machine learning. Its primary goal is to transform a high-dimensional dataset into a lower-dimensional one while retaining as much of the original data's variability as possible. PCA identifies linear combinations of the original features, known as principal components, which capture the most significant variations in the data. These components are ordered by their importance, with the first component explaining the most variance and subsequent components explaining progressively less.

In this case the dataset is fed to the PCA algorithm, and the features are selected based on higher principal components, the next less component, and so on, for 90% explained cumulative variance.

Here are the results of PCA analysis:

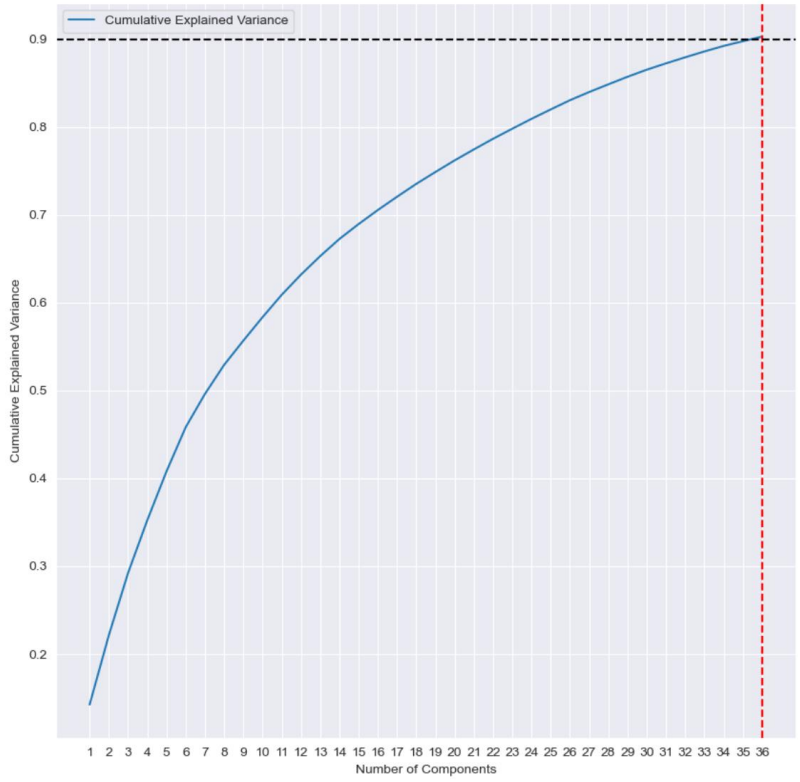

Fig. 75: Principal Component Analysis

Observations: The graph depicts the cumulative explained variance by the number of components, typically used in Principal Component Analysis. The curve increases steeply at the beginning, indicating that the initial components explain a significant amount of variance within the data. As more components are added, the rate of increase in explained variance slows down, but the cumulative explained variance continues to rise.

The plot has a dashed horizontal line close to the 0.9 mark, which is commonly used as a cutoff for deciding how many components to retain. The vertical dashed line intersects the horizontal line at around 36 components, suggesting that 36 components explain approximately 90% of the variance in the data.

Tables:

Year	Number of Accidents
2005	5349
2006	5160
2007	5173
2008	3630
2009	3349
2010	3243
2011	3044
2012	2777
2013	2616
2014	2612

Table 1: Table of Accidents per year

Observations: The table shows a clear downward trend in the number of accidents from 2005 to 2014. The highest value is in 2005 at 5349, with the lowest value in 2014 at 2612. There is a significant drop after 2007.

Year	Number of Casualties
2005	10474
2006	10346
2007	11887
2008	7491
2009	6390
2010	6171
2011	5906
2012	5454
2013	5076
2014	5204

Table 2: Table of Casualties per year

Observations: The table shows a clear downward trend in the number of deaths from 2005 to 2014. The highest value is in 2005 at 10474, with the lowest value in 2014 at 5204. There is a significant drop after 2007.

Carriage Way Hazards	Number of Accidents
Any animal in carriageway (except ridden horse)	27
Data missing or out of range	1
None	36582
Other object on road	102
Pedestrian in carriageway - not injured	115
Previous accident	110
Vehicle load on road	16

Table 3: Table of Carriageway Hazards

Observations: This table represents the vast majority of accidents that occurred with no hazards reported, suggesting that factors other than road obstructions are major contributors to accidents. Animals on the road, other objects, pedestrians not injured, and previous accidents represent a minor proportion, while cases of vehicle load on the road leading to accidents are relatively rare.

Road Surface Condition	Number of Accidents
Dry	28401
Flood	11
Ice	301
Snow	107
Unknown	214
Wet	7919

Table 4: Table of Road Surface Conditions

Observations: This table shows that that a significant majority of accidents occurred on dry road conditions, i.e., 28401, followed by a substantial number on wet roads. Accidents on icy roads are also there, while those on snowy roads and in flood conditions are relatively small in number.

Weather Condition	Number of Accidents
Fine	289
Fine no high winds	29913
Fog	233
Other	738
Raining no high winds	3753
Rainwinds	317
Snowind	16
Snowing no high winds	152
Unknown	1542

Table 5: Table of Weather Conditions

Observations: This table shows that the majority of accidents occurred in fine weather conditions with no high winds. Incidents in rainy weather are also high, whereas accidents in foggy conditions, while less frequent, are still notable. Snowy conditions account for the least number of accidents.

Phase 2: Interactive web-based Dashboard

A dashboard has been created to visualize and analyze the relationships between accidents, vehicles and casualties.

There are 9 tabs in the dashboard app:

Tab 1: About Project: Basic details of the project.

Tab 2: Subplots: Different types of plots clubbed together to analyze the data, which can be selected by the user using the dropdown.

Tab 3: Individual Line Plots: Different types of line plots, which can be selected by the user using the dropdown.

Tab 4: Individual Bar Plots: Different types of bar plots, which can be selected by the user using the dropdown.

Tab 5: Individual Pie Plots: Different types of pie plots, which can be selected by the user using the dropdown.

Tab 6: Individual Count Plots: Different types of count plots, which can be selected by the user using the dropdown.

Tab 7: Individual Histograms: Different types of histogram plots, which can be selected by the user using the dropdown.

Tab 8: Other Miscellaneous Plots: Different miscellaneous plots, which can be selected by the user using checklist.

Tab 9: Geospatial Plots: Different types of geospatial plots, which can be selected by the user using the radio buttons and year through sliding.

Here is the image of the about project tab:

Here is the sample of the subplot tab:

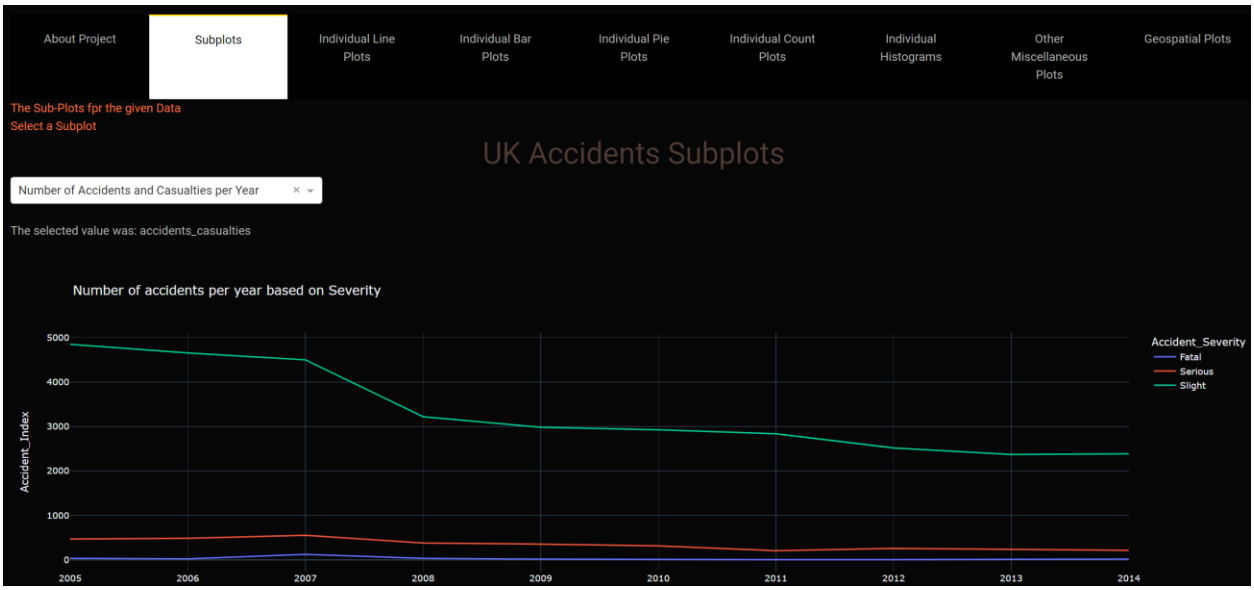

Fig. 76: Sample Plot of Subplot

Observations: In this subplot tab, a line graph displays accident severity from 2005 to 2014, with 'Slight' accidents being the most common and showing a notable declining trend. 'Serious' and 'Fatal' accidents are less frequent, with 'Fatal' accidents remaining relatively stable over the years.

Here is the sample of the individual line plot tab:

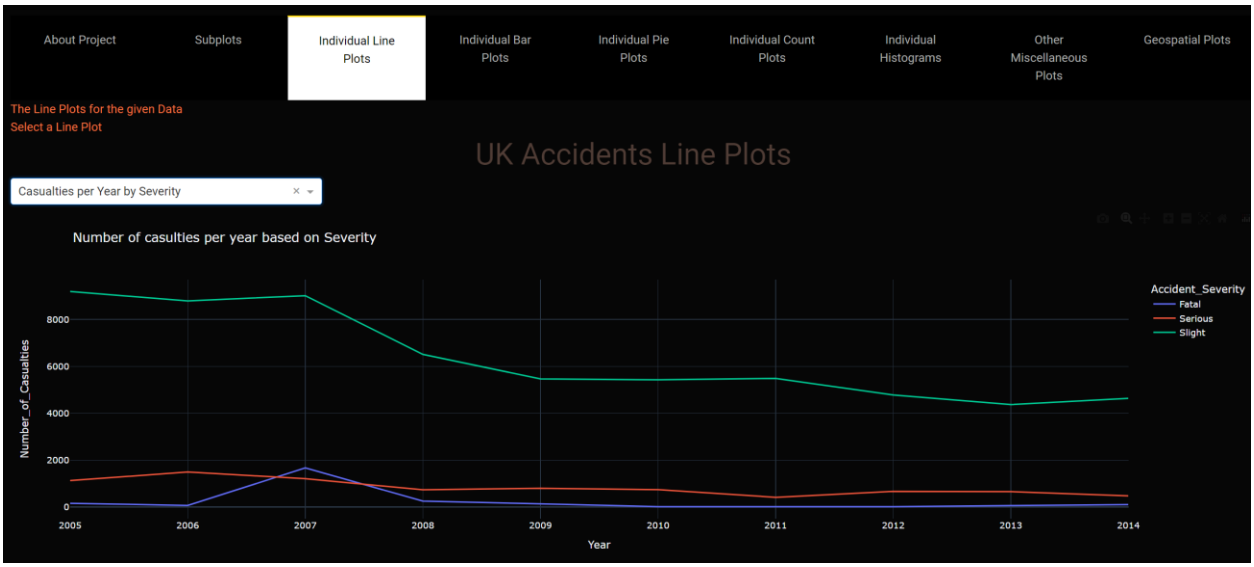

Fig. 77: Sample Plot of Individual Line plot

Observations: In this individual line plot tab, the line plot titled "Number of casualties per year based on Severity" as part of the UK Accidents Line Plots shows that 'Slight' casualties are most common, but have decreased over the years. 'Serious' casualties also exhibit a decrease, with a slight uptick around 2008. 'Fatal' casualties fluctuate year by year but remain the least common, with a slight increasing trend towards 2014.

Here is the sample of the individual bar plot tab:

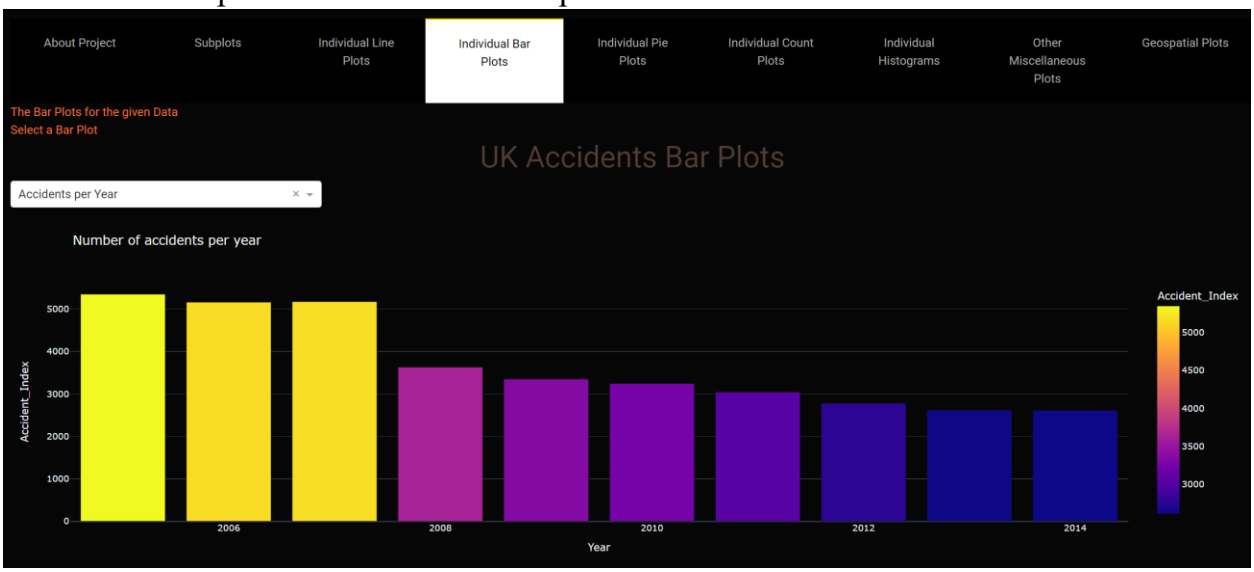

Fig. 78: Sample Plot of Individual Bar plot

Observations: In this individual bar plot tab, the bar plot from the UK Accidents Bar Plots section shows the number of accidents per year from 2005 to 2014. There is a clear downward trend in the number of accidents over the years, with the highest in 2005 and 2006 and a steady decrease to the lowest in 2014.

Here is the sample of the individual pie plot tab:

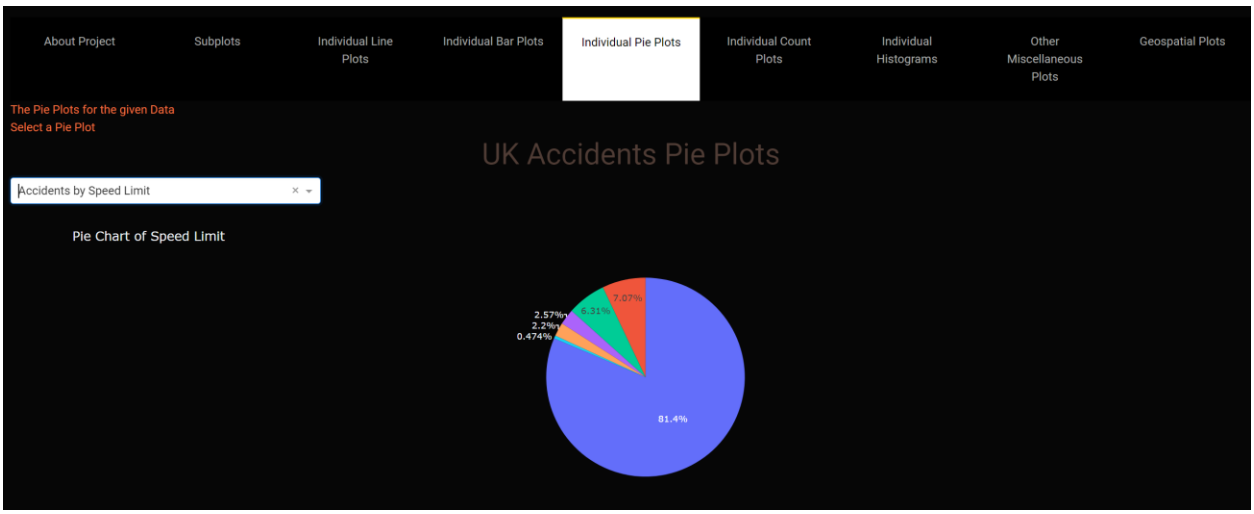

Fig. 79: Sample Plot of Individual Pie plot

Observations: In this individual pie plot tab, the pie plot from the UK Accidents Pie Plots dashboard section displays the distribution of accidents by speed limit. Many accidents, 61.4%, occur at one speed limit, represented by the large blue segment. The remaining accidents are distributed among various other speed limits, with the smallest proportion at 0.47%, shown in green.

Here is the sample of the individual count plot tab:

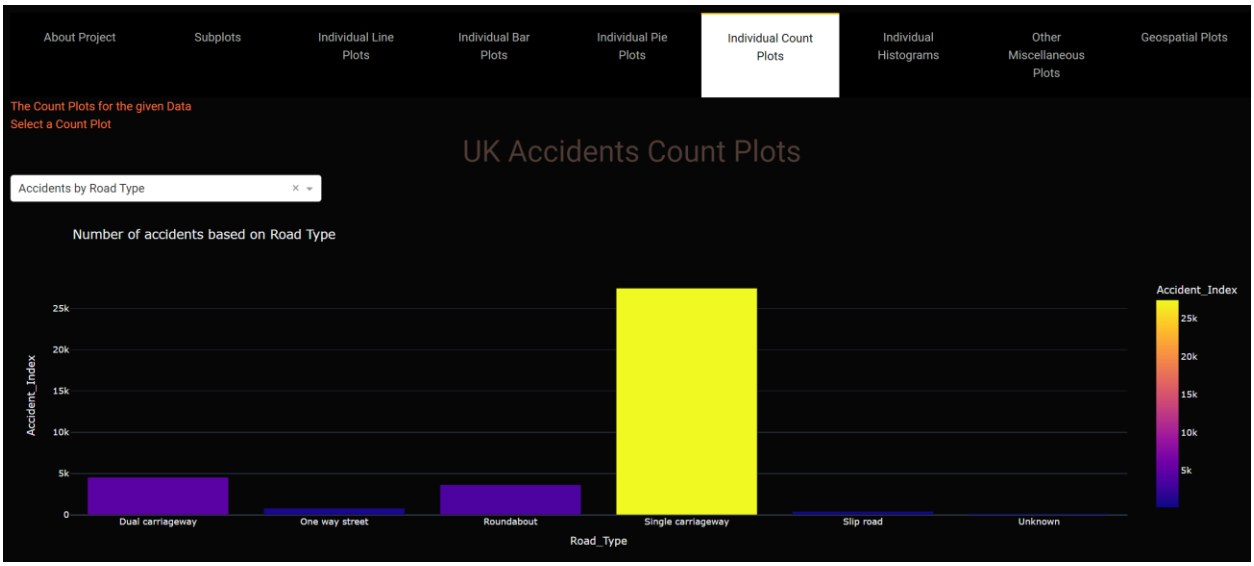

Fig. 80: Sample Plot of Individual Count plot

Observations: In this individual count plot tab, the count plot from the UK Accidents Count Plots section shows the number of accidents categorized by road type. The most accidents occur on a "Single carriageway," as indicated by the tall yellow bar, followed by a significantly lower number on "Dual carriageways," "Roundabouts," and "One way streets." A category labeled "Unknown" has some accidents, while "Slip roads" have the fewest recorded accidents.

Here is the sample of the individual histograms:

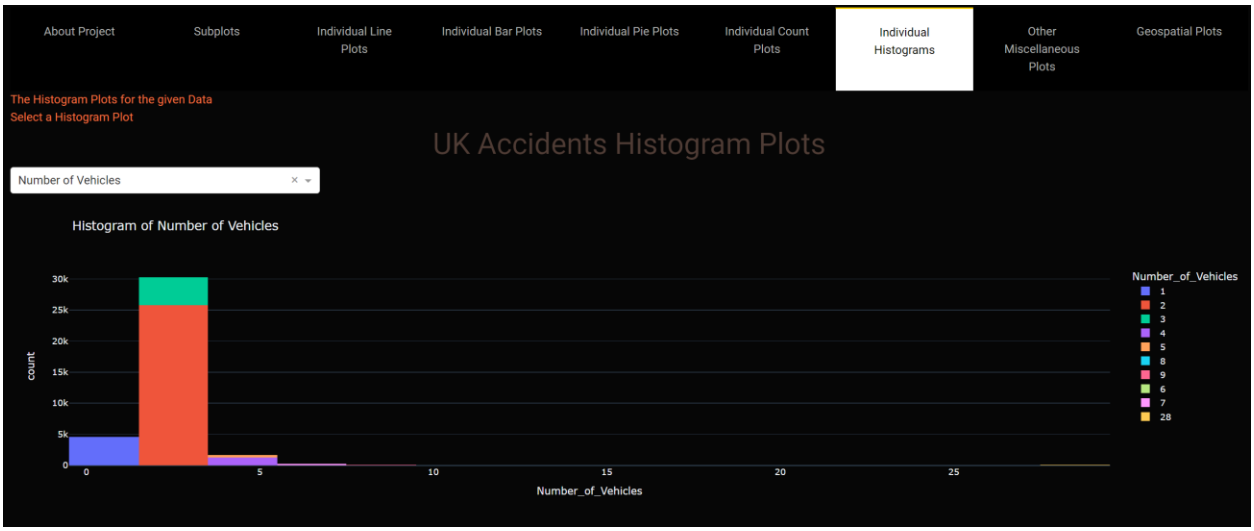

Fig. 81: Sample Plot of Individual Histograms

Observations: In this individual histogram tab, the histogram from the UK Accidents Histogram Plots section displays the count of accidents by the number of vehicles involved. Accidents involving two vehicles are the most common, shown by the tall red bar, followed by accidents with one vehicle. There is a sharp decline in the frequency of accidents as the number of vehicles increases, with accidents involving three or more vehicles being relatively rare.

Here is the sample of the other misc. plots:

Fig. 82: Sample Plot of Other Misc. Plots

Observations: In this other miscellaneous tab, the area plot from the UK Accidents Miscellaneous Plots section illustrates the number of casualties by year, differentiated by accident severity. The largest area, in green, represents slight injuries, which consistently constitute the bulk of casualties each year. Serious injuries, shown in blue, peak around 2007 and then decrease over time.

Here is the sample of geospatial plots:

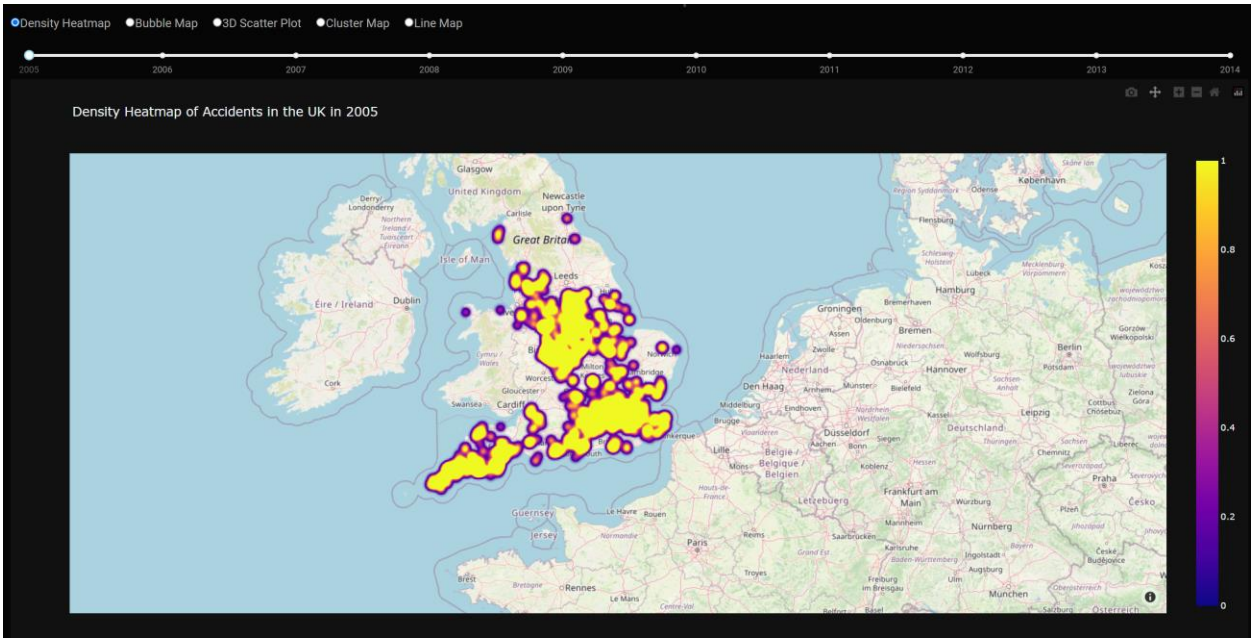

Fig. 83: Sample Plot of Geospatial plots

Observations: In this geospatial plot tab, the density heatmap of accidents in the UK for the year 2005. High concentrations of accidents are indicated by warmer colors (yellow to red), showing significant clusters around major urban areas and regions show moderate to low accident densities, as depicted by cooler colors (purple to blue).

Phase 3: Deployment

The dashboard has been dockerized the dashboard app and deployed on Google Cloud Platform for production.

Here is the link to the deployed app:
https://dashapp-aybkomhtgq-nn.a.run.app/

Conclusion:

This initiative marks a progressive leap in traffic accident analysis. The project harnesses the power of visual data representation to uncover critical insights, which are essential for devising effective road safety measures. The data reveals specific risk factors linked to different vehicle types and situational variables, providing a rich knowledge base for those shaping safety regulations and educational programs. Future directions include the development of predictive models to foresee and address potential traffic risks, underscoring a proactive commitment to road safety enhancement.

a. Insights from Graphs: The graphs demonstrate the prevalence and characteristics of road accidents, vehicles, and casualties and show regional patterns in accident occurrence. This visual approach brings to light the intricate dynamics of road safety.

b. Benefits of the Python Dashboard: The interactive Python dashboard simplifies complex data analysis, allowing users to visualize trends and patterns that may not be immediately apparent in raw data. It transforms data exploration into an interactive experience, making the information more accessible and easier to understand for a broad audience.

c. User-Friendliness: The created app is designed to be user-friendly. It offers an intuitive interface that allows both experts and laypersons to navigate and understand complex traffic accident data through interactive visualizations. The dashboard's capacity to present data through various graphical formats like density map plots, area plots, and dynamic charts likely contributes to an engaging user experience, making the analysis of intricate patterns both accessible and straightforward. This approach emphasizes usability and aims to facilitate a deeper engagement with the data, enhancing the user's ability to gain insights without requiring specialized knowledge in data analysis.

d. App Functionality: The app's functionality is robust, offering a user-friendly environment to explore and interpret traffic data. Its interactivity and flexible visualization capabilities make it a valuable tool for detailed data analysis, suitable for diverse users including researchers, policymakers, and the public.

Appendix:

Phase 1: Static Plots and Tables

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from mpl_toolkits.mplot3d import Axes3D
import time
import statsmodels.api as sm
import scipy.stats as stats
from scipy.stats import shapiro
from scipy.stats import boxcox
from scipy.interpolate import griddata
from imblearn.over_sampling import SMOTE
from sklearn.decomposition import PCA
import warnings

pd.set_option('display.float_format', lambda x: '%.2f' % x)
sns.set_style('darkgrid')
warnings.filterwarnings("ignore")

# Read in the data in chunks in for loop and then convert it to a dataframe
and see the time it takes to read the data
start_time = time.time()
accidents_vehicles_casualties = pd.DataFrame()

for chunk in pd.read_csv('UK_Accidents_Merger.csv', chunksize=200000,
low_memory=False):
    print('Number of chunks read: ', chunk.shape)
    accidents_vehicles_casualties = pd.concat([accidents_vehicles_casualties,
chunk])

print("Time taken to read the data: ", time.time() - start_time)

print("Shape of the dataframe: ", accidents_vehicles_casualties.shape)

#---------------------------------------------
# Data Wrangling
#---------------------------------------------

# Check for missing values
print("Missing Values:")
print(accidents_vehicles_casualties.isnull().sum()/accidents_vehicles_casualt
ies.shape[0])

# Drop or impute missing values
accidents_vehicles_casualties.dropna(inplace=True)

# Save the dataframe to a csv file
# accidents_vehicles_casualties.to_csv('UK_Accidents_Merger_Cleaned.csv',
index=False)

# Check for duplicates
```

```
print("Duplicates:")
print(accidents_vehicles_casualties.duplicated().sum())

# Check for data types
print("Data Types:")
print(accidents_vehicles_casualties.dtypes)

# Count number of numeric and non-numeric columns
print("Number of Numeric Columns:")
print(len(accidents_vehicles_casualties.select_dtypes(include=np.number).colu
mns))

print("Number of Non-Numeric Columns:")
print(len(accidents_vehicles_casualties.select_dtypes(exclude=np.number).colu
mns))

#----------------------------------------
# Viusalizations using Matplotlib, Seaborn
#----------------------------------------

# Convert date to year
accidents_vehicles_casualties['Date'] =
pd.to_datetime(accidents_vehicles_casualties['Date'])
accidents_vehicles_casualties['Year'] =
accidents_vehicles_casualties['Date'].dt.year

# Manipulate the data
accidents_vehicles_casualties['Junction_Detail'] =
accidents_vehicles_casualties['Junction_Detail'].replace({
    'T or staggered junction': 'Staggered Junction',
    'Private drive or entrance': 'Entrance',
    'Other junction': 'Others',
    'More then 4 arms (not roundabout)': 'Fours',
    'Mini-roundabout': 'Miniroundabout',
    'Not at junction or within 20 metres': 'No Junction',
    'Data missing or out of range': 'Unknown',
})

accidents_vehicles_casualties['Junction_Control'] =
accidents_vehicles_casualties['Junction_Control'].replace({
    'Authorised person': 'Authorised',
    'Data missing or out of range': 'Unknown',
    'Give way or uncontrolled': 'Uncontrolled',
    'Stop sign': 'Stop',
    'Auto traffic signal': 'Traffic Signal',
})


accidents_vehicles_casualties['Light_Conditions'] =
accidents_vehicles_casualties['Light_Conditions'].replace({
    'Darkness - no lighting': 'Darkness',
    'Darkness - lighting unknown': 'Unknown',
    'Darkness - lights lit': 'Light',
    'Darkness - lights unlit': 'Unlit'
})

accidents_vehicles_casualties['Weather_Conditions'] =
```

```
accidents_vehicles_casualties['Weather_Conditions'].replace({
    'Fine without high winds': 'Fine',
    'Raining without high winds': 'Rain',
    'Raining + high winds': 'Rainwinds',
    'Fine + high winds': 'Fine',
    'Fog or mist': 'Fog',
    'Snowing without high winds': 'Snow',
    'Snowing + high winds': 'Snowind',
    'Data missing or out of range': 'Unknown',
})

accidents_vehicles_casualties['Road_Surface_Conditions'] =
accidents_vehicles_casualties['Road_Surface_Conditions'].replace({
    'Dry': 'Dry',
    'Wet or damp': 'Wet',
    'Frost or ice': 'Ice',
    'Snow': 'Snow',
    'Flood over 3cm. deep': 'Flood',
    'Data missing or out of range': 'Unknown',
})

accidents_vehicles_casualties['Did_Police_Officer_Attend_Scene_of_Accident']
=
accidents_vehicles_casualties['Did_Police_Officer_Attend_Scene_of_Accident'].
replace(
    'No - accident was reported using a self completion  form (self rep
only)', 'Self')

accidents_vehicles_casualties['Vehicle_Type'] =
accidents_vehicles_casualties['Vehicle_Type'].replace({
    'Bus or coach (17 or more pass seats)': 'Bus',
    'Van / Goods 3.5 tonnes mgw or under': 'Van',
    'Taxi/Private hire car': 'Taxi',
    'Motorcycle 125cc and under': 'Motorcycle',
    'Motorcycle over 500cc': 'Motorcycle',
    'Goods 7.5 tonnes mgw and over': 'Goods',
    'Motorcycle 50cc and under': 'Motorcycle',
    'Motorcycle over 125cc and up to 500cc': 'Motorcycle',
    'Goods over 3.5t. and under 7.5t': 'Goods',
    'Other vehicle': 'Others',
    'Minibus (8 - 16 passenger seats)': 'Minibus',
    'Agricultural vehicle (includes diggers etc.)': 'Agricultural',
    'Motorcycle - unknown cc': 'Motorcycle'
})

accidents_vehicles_casualties['Towing_and_Articulation'] =
accidents_vehicles_casualties['Towing_and_Articulation'].replace({
    'No tow/articulation': 'No',
    'Articulated vehicle': 'Articulated',
    'Single trailer': 'Single',
    'Other tow': 'Others',
    'Double or multiple trailer': 'Multiple',
    'Data missing or out of range': 'Unknown',
})

accidents_vehicles_casualties['Vehicle_Manoeuvre'] =
accidents_vehicles_casualties['Vehicle_Manoeuvre'].replace({
```

```
    'Going ahead other': 'Going ahead',
    'Turning right': 'Right',
    'Waiting to go - held up': 'Waiting',
    'Slowing or stopping': 'Slowing',
    'Turning left': 'Left',
    'Moving off': 'Moving',
    'Waiting to turn right': 'Waiting',
    'Going ahead right-hand bend': 'Going ahead',
    'Going ahead left-hand bend': 'Going ahead',
    'Overtaking moving vehicle - offside': 'Overtaking',
    'Waiting to turn left': 'Waiting',
    'Overtaking static vehicle - offside': 'Overtaking',
    'Changing lane to left': 'Changing lane',
    'Changing lane to right': 'Changing lane',
    'U-turn': 'Uturn',
    'Overtaking - nearside': 'Overtaking',
    'Data missing or out of range': 'Unknown',
})

accidents_vehicles_casualties['Vehicle_Location-Restricted_Lane'] =
accidents_vehicles_casualties['Vehicle_Location-Restricted_Lane'].replace({
    'On main c\way - not in restricted lane': 'Mainlane',
    'Bus lane': 'Buslane',
    'Footway (pavement)': 'Footway',
    'Leaving lay-by or hard shoulder': 'Leavinglayby',
    'On lay-by or hard shoulder': 'Onlayby',
    'Busway (including guided busway)': 'Busway',
    'Cycle lane (on main carriageway)': 'Cyclelane',
    'Tram/Light rail track': 'Tramtrack',
    'Entering lay-by or hard shoulder': 'Enteringlayby',
    'Cycleway or shared use footway (not part of  main carriageway)':
'Cycleway',
    'Data missing or out of range': 'Unknown',
})

accidents_vehicles_casualties['Junction_Location'] =
accidents_vehicles_casualties['Junction_Location'].replace({
    'Approaching junction or waiting/parked at junction approach':
'Approaching',
    'Mid Junction - on roundabout or on main road': 'Mid',
    'Cleared junction or waiting/parked at junction exit': 'Cleared',
    'Entering from slip road': 'Entering',
    'Leaving main road into minor road': 'Leaving',
    'Entering main road from minor road': 'Entering',
    'Leaving roundabout': 'Leaving',
    'Entering roundabout': 'Entering',
    'Data missing or out of range': 'Unknown',
    'Not at or within 20 metres of junction': 'No Junction',
})

accidents_vehicles_casualties['1st_Point_of_Impact'] =
accidents_vehicles_casualties['1st_Point_of_Impact'].replace({
    'Did not impact': 'Nothing',
    'Data missing or out of range': 'Unknown'
})

accidents_vehicles_casualties['Was_Vehicle_Left_Hand_Drive?'] =
```

```
accidents_vehicles_casualties['Was_Vehicle_Left_Hand_Drive?'].replace({
    'Data missing or out of range': 'Unknown'
})

accidents_vehicles_casualties['Propulsion_Code'] =
accidents_vehicles_casualties['Propulsion_Code'].replace({
    'Gas/Bi-fuel': 'Gas',
    'Petrol/Gas (LPG)': 'LPG',
})

accidents_vehicles_casualties['Casualty_Class'] =
accidents_vehicles_casualties['Casualty_Class'].replace({
    'Driver or rider': 'Driver'
})

accidents_vehicles_casualties['Sex_of_Casualty'] =
accidents_vehicles_casualties['Sex_of_Casualty'].replace({
    'Data missing or out of range': 'Unknown'
})

accidents_vehicles_casualties['Car_Passenger'] =
accidents_vehicles_casualties['Car_Passenger'].replace({
    'Data missing or out of range': 'Unknown'
})

accidents_vehicles_casualties['Urban_or_Rural_Area'] =
accidents_vehicles_casualties['Urban_or_Rural_Area'].replace({
    1: 'Urban',
    2: 'Rural',
    3: 'Unallocated'
})

accidents_vehicles_casualties['Special_Conditions_at_Site'] =
accidents_vehicles_casualties['Special_Conditions_at_Site'].replace({
    'Auto traffic signal - out': 'No signal',
    'Auto signal part defective': 'Signal defect',
    'Oil or diesel': 'Oil',
    'Road sign or marking defective or obscured': 'Sign defect',
    'Road surface defective': 'Road defect'
})


# Histogram plots with KDE and probability plot
cont_columns =
list(accidents_vehicles_casualties.select_dtypes(include=[np.number]).columns
.values)

cont_columns.remove('Location_Easting_OSGR')
cont_columns.remove('Location_Northing_OSGR')
cont_columns.remove('Longitude')
cont_columns.remove('Latitude')
cont_columns.remove('Year')
cont_columns.remove('1st_Road_Number')
cont_columns.remove('2nd_Road_Number')
cont_columns.remove('Pedestrian_Crossing-Human_Control')
cont_columns.remove('Pedestrian_Crossing-Physical_Facilities')
cont_columns.remove('Vehicle_Reference_x')
```

```
cont_columns.remove('Vehicle_Reference_y')
cont_columns.remove('Pedestrian_Location')
cont_columns.remove('Pedestrian_Movement')
cont_columns.remove('Pedestrian_Road_Maintenance_Worker')
cont_columns.remove('Casualty_Home_Area_Type')
cont_columns.remove('Age_Band_of_Driver')
cont_columns.remove('Age_Band_of_Casualty')
cont_columns.remove('Driver_Home_Area_Type')
cont_columns.remove('Bus_or_Coach_Passenger')

print("Numerical Columns: \n", cont_columns)

cat_cols =
list(accidents_vehicles_casualties.select_dtypes(include=['object']).columns)

cat_cols.remove('Accident_Severity')

#----------------------------------------
# Line plots
#----------------------------------------

# Create a line graph of the number of accidents per year
accidents_by_year =
accidents_vehicles_casualties.groupby('Year').count()['Accident_Index'].reset
_index()

plt.figure(figsize=(10, 6))
sns.lineplot(x='Year', y='Accident_Index', data=accidents_by_year,
palette='bright')
plt.xlabel('Year')
plt.ylabel('Number of Accidents')
plt.title('Number of Accidents per Year')
plt.xticks(rotation=45)
plt.grid(True)
plt.show()

# Create a line graph of the number of casualties per year
casualties_by_year =
accidents_vehicles_casualties.groupby('Year').sum()['Number_of_Casualties'].r
eset_index()

plt.figure(figsize=(10, 6))
sns.lineplot(x='Year', y='Number_of_Casualties', data=casualties_by_year,
palette='bright')
plt.xlabel('Year')
plt.ylabel('Number of Casualties')
plt.title('Number of Casualties per Year')
plt.xticks(rotation=45)
plt.grid(True)
plt.show()

#----------------------------------------
# Bar graphs
#----------------------------------------

# Create a bar chart of the number of accidents per year with numbers on the
top of plot as well as a line gping thorugh all the plots
```

```
plt.figure(figsize=(10, 6))
sns.barplot(x='Year', y='Accident_Index', data=accidents_by_year,
palette='bright')
plt.xlabel('Year')
plt.ylabel('Number of Accidents')
plt.title('Number of Accidents per Year')
plt.xticks(rotation=45)

for k, v in accidents_by_year['Accident_Index'].items():
    if v > 1000:
        plt.text(k, v + 100, str(v), fontsize=12, color="black", ha="center",
                 rotation=90, va="bottom")

accidents_by_year['Accident_Index'].plot(label='Number of Accidents',
color='black', linewidth=3,
                                         ax=plt.twinx(), marker='o',
markersize=10)
plt.grid(True)
plt.show()

# Create a bar chart of the number of accidents per year per severity
accidents_by_year_severity = accidents_vehicles_casualties.groupby(['Year',
'Accident_Severity']).count()['Accident_Index'].reset_index()

plt.figure(figsize=(10, 6))
sns.barplot(x='Year', y='Accident_Index', hue='Accident_Severity',
data=accidents_by_year_severity, palette='bright')
plt.xlabel('Year')
plt.ylabel('Number of Accidents')
plt.title('Number of Accidents per Year per Severity')
plt.xticks(rotation=45)
plt.grid(True)
plt.show()

# Create a bar chart of casualties per year
plt.figure(figsize=(10, 6))
sns.barplot(x='Year', y='Number_of_Casualties', data=casualties_by_year,
palette='bright')
plt.xlabel('Year')
plt.ylabel('Number of Casualties')
plt.title('Number of Casualties per Year')
plt.xticks(rotation=45)

for k, v in casualties_by_year['Number_of_Casualties'].items():
    if v > 1000:
        plt.text(k, v + 100, str(v), fontsize=12, color="black", ha="center",
                 rotation=90, va="bottom")

casualties_by_year['Number_of_Casualties'].plot(label='Number of Casualties',
color='black', linewidth=3,
                                                ax=plt.twinx(), marker='o',
markersize=10)
plt.grid(True)
plt.show()

# Create a bar chart of casualties per year per severity
casualties_by_year_severity = accidents_vehicles_casualties.groupby(['Year',
```

```
'Accident_Severity']).sum()['Number_of_Casualties'].reset_index()

plt.figure(figsize=(10, 6))
sns.barplot(x='Year', y='Number_of_Casualties', hue='Accident_Severity',
data=casualties_by_year_severity, palette='bright')
plt.xlabel('Year')
plt.ylabel('Number of Casualties')
plt.title('Number of Casualties per Year per Severity')
plt.xticks(rotation=45)
plt.grid(True)
plt.show()

# Create a bar graph that checks whether majority of the accidents happen
during the urban or rural areas, with line graph of casulties based on the
settlements
accidents_by_settlement =
accidents_vehicles_casualties.groupby('Urban_or_Rural_Area').count()['Acciden
t_Index'].reset_index()

plt.figure(figsize=(10, 6))
sns.barplot(x='Urban_or_Rural_Area', y='Accident_Index',
data=accidents_by_settlement, palette='bright')
plt.xlabel('Settlement Type')
plt.ylabel('Number of Accidents')
plt.title('Number of Accidents per Settlement Type')
plt.xticks(rotation=45)
plt.grid(True)
plt.show()

time.sleep(10)
time.sleep(5)

# Create a graph of urban/rural accidents based on severity
accidents_by_settlement_severity =
accidents_vehicles_casualties.groupby(['Urban_or_Rural_Area',
'Accident_Severity']).count()['Accident_Index'].reset_index()

plt.figure(figsize=(10, 6))
sns.barplot(x='Urban_or_Rural_Area', y='Accident_Index',
hue='Accident_Severity', data=accidents_by_settlement_severity,
palette='bright')
plt.xlabel('Settlement Type')
plt.ylabel('Number of Accidents')
plt.title('Number of Accidents per Settlement Type per Severity')
plt.xticks(rotation=45)
plt.grid(True)
plt.show()

# Create a graph explaining of many vehicles were involved in accidents based
on the severity, and how many casulaties were involved in the accidents based
on the severity
accidents_by_vehicles =
accidents_vehicles_casualties.groupby('Number_of_Vehicles').count()['Accident
_Index'].reset_index()

plt.figure(figsize=(10, 6))
sns.barplot(x='Number_of_Vehicles', y='Accident_Index',
```

```
data=accidents_by_vehicles, palette='bright')
plt.xlabel('Number of Vehicles')
plt.ylabel('Number of Accidents')
plt.title('Number of Accidents per Number of Vehicles')
plt.xticks(rotation=45)
plt.grid(True)
plt.show()

# Create a graph to check whether many accidents happened in the weekdays or
the weekends
accidents_by_day =
accidents_vehicles_casualties.groupby('Day_of_Week').count()['Accident_Index'
].reset_index()

plt.figure(figsize=(10, 6))
sns.barplot(x='Day_of_Week', y='Accident_Index', data=accidents_by_day,
palette='bright')
plt.xlabel('Day of Week')
plt.ylabel('Number of Accidents')
plt.title('Number of Accidents per Day of Week')
plt.xticks(rotation=45)
plt.grid(True)
plt.show()

# Create a graph to check whether many accidents happened in the weekdays or
the weekends based on severity
accidents_by_day_severity =
accidents_vehicles_casualties.groupby(['Day_of_Week',
'Accident_Severity']).count()['Accident_Index'].reset_index()

plt.figure(figsize=(10, 6))
sns.barplot(x='Day_of_Week', y='Accident_Index', hue='Accident_Severity',
data=accidents_by_day_severity, palette='bright')
plt.xlabel('Day of Week')
plt.ylabel('Number of Accidents')
plt.title('Number of Accidents per Day of Week per Severity')
plt.xticks(rotation=45)
plt.grid(True)
plt.show()

# Create a bar graph to check whether accident severity is based on the type
of road
accidents_by_road = accidents_vehicles_casualties.groupby(['Road_Type',
'Accident_Severity']).count()['Accident_Index'].reset_index()

plt.figure(figsize=(10, 6))
sns.barplot(x='Road_Type', y='Accident_Index', data=accidents_by_road,
palette='bright', hue='Accident_Severity')
plt.xlabel('Road Type')
plt.ylabel('Number of Accidents')
plt.title('Number of Accidents per Road Type')
plt.xticks(rotation=45)
plt.grid(True)
plt.show()

#Create a groupby function depicting the road surface conditions during the
accidents
```

```
accidents_by_surface =
accidents_vehicles_casualties.groupby('Road_Surface_Conditions').count()['Acc
ident_Index'].reset_index()

# Create a bar graph to check whether accident severity is based on the
surface condition of the road
accidents_by_surface_severity =
accidents_vehicles_casualties.groupby(['Road_Surface_Conditions',
'Accident_Severity']).count()['Accident_Index'].reset_index()

plt.figure(figsize=(10, 6))
sns.barplot(x='Road_Surface_Conditions', y='Accident_Index',
data=accidents_by_surface_severity, palette='bright',
hue='Accident_Severity')
plt.xlabel('Road Surface Condition')
plt.ylabel('Number of Accidents')
plt.title('Number of Accidents per Road Surface Condition')
plt.xticks(rotation=45)
plt.grid(True)
plt.show()

# Create a graph depicting severity if accidents based on the speed limits
accidents_by_speed = accidents_vehicles_casualties.groupby(['Speed_limit',
'Accident_Severity']).count()['Accident_Index'].reset_index()

plt.figure(figsize=(10, 6))
sns.barplot(x='Speed_limit', y='Accident_Index', data=accidents_by_speed,
palette='bright', hue='Accident_Severity')
plt.xlabel('Speed Limit')
plt.ylabel('Number of Accidents')
plt.title('Number of Accidents per Speed Limit')
plt.xticks(rotation=45)
plt.grid(True)
plt.show()

# Create a graph depicting which kind of vehicles were involved in accidents
accidents_by_vehicle = accidents_vehicles_casualties.groupby(['Vehicle_Type',
'Accident_Severity']).count()['Accident_Index'].reset_index()

plt.figure(figsize=(10, 6))
sns.barplot(x='Vehicle_Type', y='Accident_Index', data=accidents_by_vehicle,
palette='bright', hue='Accident_Severity')
plt.xlabel('Vehicle Type')
plt.ylabel('Number of Accidents')
plt.title('Number of Accidents per Vehicle Type')
plt.xticks(rotation=45)
plt.grid(True)
plt.show()

#Create a groupby function depicting the weather conditions during the
accidents
accidents_by_weather =
accidents_vehicles_casualties.groupby('Weather_Conditions').count()['Accident
_Index'].reset_index()

# Create a graph depicting the weather conditions during the accidents based
on severity
```

```
accidents_by_weather_severity =
accidents_vehicles_casualties.groupby(['Weather_Conditions',
'Accident_Severity']).count()['Accident_Index'].reset_index()

plt.figure(figsize=(10, 6))
sns.barplot(x='Weather_Conditions', y='Accident_Index',
data=accidents_by_weather_severity, palette='bright',
hue='Accident_Severity')
plt.xlabel('Weather Condition')
plt.ylabel('Number of Accidents')
plt.title('Number of Accidents per Weather Condition')
plt.xticks(rotation=45)
plt.grid(True)
plt.show()

#Create a groupby function depicting the light conditions during the
accidents
accidents_by_light =
accidents_vehicles_casualties.groupby('Light_Conditions').count()['Accident_I
ndex'].reset_index()

# Create a graph depicting the light conditions during the accidents based on
severity
accidents_by_light_severity =
accidents_vehicles_casualties.groupby(['Light_Conditions',
'Accident_Severity']).count()['Accident_Index'].reset_index()

plt.figure(figsize=(10, 6))
sns.barplot(x='Light_Conditions', y='Accident_Index',
data=accidents_by_light_severity, palette='bright', hue='Accident_Severity')
plt.xlabel('Light Condition')
plt.ylabel('Number of Accidents')
plt.title('Number of Accidents per Light Condition')
plt.xticks(rotation=45)
plt.grid(True)
plt.show()

#Create a groupby function depicting the special conditions during the
accidents
accidents_by_special =
accidents_vehicles_casualties.groupby('Special_Conditions_at_Site').count()['
Accident_Index'].reset_index()

# Create a graph depicting the special conditions during the accidents based
on severity
accidents_by_special_severity =
accidents_vehicles_casualties.groupby(['Special_Conditions_at_Site',
'Accident_Severity']).count()['Accident_Index'].reset_index()

plt.figure(figsize=(10, 6))
sns.barplot(x='Special_Conditions_at_Site', y='Accident_Index',
data=accidents_by_special_severity, palette='bright',
hue='Accident_Severity')
plt.xlabel('Special Condition')
plt.ylabel('Number of Accidents')
plt.title('Number of Accidents per Special Condition')
plt.xticks(rotation=45)
```

```
plt.grid(True)
plt.show()


# Create a graph of how the accidents happened
accidents_happen_how =
accidents_vehicles_casualties.groupby(['Skidding_and_Overturning',
'Accident_Severity']).count()['Accident_Index'].reset_index()

plt.figure(figsize=(10, 6))
sns.barplot(x='Skidding_and_Overturning', y='Accident_Index',
data=accidents_happen_how, palette='bright', hue='Accident_Severity')
plt.xlabel('Accident Incidents')
plt.ylabel('Number of Accidents')
plt.title('How the Accidents Happened')
plt.xticks(rotation=45)
plt.grid(True)
plt.show()

# Create a graph of what is the 1st point of impact
accidents_by_impact =
accidents_vehicles_casualties.groupby(['1st_Point_of_Impact',
'Accident_Severity']).count()['Accident_Index'].reset_index()

plt.figure(figsize=(10, 6))
sns.barplot(x='1st_Point_of_Impact', y='Accident_Index',
data=accidents_by_impact, palette='bright', hue='Accident_Severity')
plt.xlabel('1st Point of Impact')
plt.ylabel('Number of Accidents')
plt.title('1st Point of Impact')
plt.xticks(rotation=45)
plt.grid(True)
plt.show()

# Create a graph based on the how many accidents were reported to the police
accidents_by_police =
accidents_vehicles_casualties.groupby('Did_Police_Officer_Attend_Scene_of_Acc
ident').count()['Accident_Index'].reset_index()

plt.figure(figsize=(10, 6))
sns.barplot(x='Did_Police_Officer_Attend_Scene_of_Accident',
y='Accident_Index', data=accidents_by_police, palette='bright')
plt.xlabel('Police Attended')
plt.ylabel('Number of Accidents')
plt.title('Number of Accidents reported to Police')
plt.xticks(rotation=45)
plt.grid(True)
plt.show()

time.sleep(10)
time.sleep(10)
time.sleep(10)

# Create a graph depicting no of accidents to the sex of the casualty
accidents_by_sex =
accidents_vehicles_casualties.groupby('Sex_of_Casualty').count()['Accident_In
dex'].reset_index()
```

```
plt.figure(figsize=(10, 6))
sns.barplot(x='Sex_of_Casualty', y='Accident_Index', data=accidents_by_sex,
palette='bright')
plt.xlabel('Sex of Casualty')
plt.ylabel('Number of Accidents')
plt.title('Number of Accidents per Sex of Casualty')
plt.xticks(rotation=45)
plt.grid(True)
plt.show()

# Create a graph depicting no of accidents to the sex of the driver
accidents_by_sex_driver =
accidents_vehicles_casualties.groupby('Sex_of_Driver').count()['Accident_Inde
x'].reset_index()

plt.figure(figsize=(10, 6))
sns.barplot(x='Sex_of_Driver', y='Accident_Index',
data=accidents_by_sex_driver, palette='bright')
plt.xlabel('Sex of the Driver')
plt.ylabel('Number of Accidents')
plt.title('Number of Accidents per Sex of the Driver')
plt.xticks(rotation=45)
plt.grid(True)
plt.show()

#----------------------------------------
# Pie charts
#----------------------------------------

def pie_plts(df, column):
    plt.figure(figsize=(10, 6))
    df[column].value_counts().plot(kind='pie', autopct='%1.1f%%',
shadow=True)
    plt.title(column)
    plt.show()

# Create a pie chart of accident severity
pie_plts(accidents_vehicles_casualties, 'Accident_Severity')

# Create a pie chart of settlement type
pie_plts(accidents_vehicles_casualties, 'Urban_or_Rural_Area')

# Create a pie chart of road type accidents
pie_plts(accidents_vehicles_casualties, 'Road_Type')

# Create a pie chart of road surface conditions
pie_plts(accidents_vehicles_casualties, 'Road_Surface_Conditions')

# Create a pie chart of weather conditions
pie_plts(accidents_vehicles_casualties, 'Weather_Conditions')

# Create a pie chart of light conditions
pie_plts(accidents_vehicles_casualties, 'Light_Conditions')

# Create a pie chart of whether police attended the scene of accident
pie_plts(accidents_vehicles_casualties,
```

```
'Did_Police_Officer_Attend_Scene_of_Accident')

time.sleep(10)

# Create a pie chart of first point of impact
pie_plts(accidents_vehicles_casualties, '1st_Point_of_Impact')

#----------------------------------------
# Count plots
#----------------------------------------

def count_plts(df, column):
    plt.figure(figsize=(10, 6))
    sns.countplot(x=column, data=df)
    plt.xlabel(column)
    plt.ylabel('Number of Accidents')
    plt.title('Number of Accidents per ' + column)
    plt.xticks(rotation=45)
    plt.grid(True)
    plt.show()

# Count plot of number of vechicles affected
count_plts(accidents_vehicles_casualties, 'Number_of_Vehicles')

# Count plot of number of casualties affected
count_plts(accidents_vehicles_casualties, 'Number_of_Casualties')

# Count plot of age of the vehicles that were involved in accidents
count_plts(accidents_vehicles_casualties, 'Age_of_Vehicle')

time.sleep(10)

#----------------------------------------
# Pairplot
#----------------------------------------

# Create a pairplot of selected features
pairplot_vars = pd.DataFrame(accidents_vehicles_casualties, columns=
cont_columns)
pairplot_vars['Accident_Severity'] =
accidents_vehicles_casualties['Accident_Severity']

sns.pairplot(data=pairplot_vars, hue="Accident_Severity" , palette='bright')
plt.title('Pairplot of Selected Features')
plt.show()

#----------------------------------------
# Heatmap
#----------------------------------------

# Create a heatmap of selected features
heatmap_vars = pd.DataFrame(accidents_vehicles_casualties, columns=
cont_columns)

plt.figure(figsize=(10, 6))
sns.heatmap(heatmap_vars.corr(method='spearman'), annot=True, cmap='icefire',
cbar=False)
```

```
plt.title('Correlation between the Features')
plt.show()

time.sleep(10)

#----------------------------------------
# Subplots
#----------------------------------------

# Create a subplot of number of accidents per year and number of casualties
per year
fig, axes = plt.subplots(2, 3, figsize=(20, 10))
plt.suptitle('Number of Accidents and Casualties per Year', fontsize=20)

# Number of accidents per year line plot
sns.lineplot(x='Year', y='Accident_Index', data=accidents_by_year,
palette='bright', ax=axes[0, 0])
axes[0, 0].set_xlabel('Year')
axes[0, 0].set_ylabel('Number of Accidents')
axes[0, 0].set_title('Number of Accidents per Year')
axes[0, 0].grid(True)
axes[0, 0].set_xticklabels(axes[0, 0].get_xticklabels(), rotation=45)

# Number of accidents per year bar plot per severity
sns.barplot(x='Year', y='Accident_Index', data=accidents_by_year,
palette='bright', ax=axes[0, 1])
axes[0, 1].set_xlabel('Year')
axes[0, 1].set_ylabel('Number of Accidents')
axes[0, 1].set_title('Number of Accidents per Year')
axes[0, 1].grid(True)
axes[0, 1].set_xticklabels(axes[0, 1].get_xticklabels(), rotation=45)

# Number of accidents per year per severity
sns.barplot(x='Year', y='Accident_Index', hue='Accident_Severity',
data=accidents_by_year_severity, palette='bright', ax=axes[0, 2])
axes[0, 2].set_xlabel('Year')
axes[0, 2].set_ylabel('Number of Accidents')
axes[0, 2].set_title('Number of Accidents per Year per Severity')
axes[0, 2].grid(True)
axes[0, 2].set_xticklabels(axes[0, 2].get_xticklabels(), rotation=45)

# Number of casualties per year line plot
sns.lineplot(x='Year', y='Number_of_Casualties', data=casualties_by_year,
palette='bright', ax=axes[1, 0])
axes[1, 0].set_xlabel('Year')
axes[1, 0].set_ylabel('Number of Casualties')
axes[1, 0].set_title('Number of Casualties per Year')
axes[1, 0].grid(True)
axes[1, 0].set_xticklabels(axes[1, 0].get_xticklabels(), rotation=45)

# Number of casualties per year bar plot
sns.barplot(x='Year', y='Number_of_Casualties', data=casualties_by_year,
palette='bright', ax=axes[1, 1])
axes[1, 1].set_xlabel('Year')
axes[1, 1].set_ylabel('Number of Casualties')
axes[1, 1].set_title('Number of Casualties per Year')
axes[1, 1].grid(True)
```

```
axes[1, 1].set_xticklabels(axes[1, 1].get_xticklabels(), rotation=45)

# Number of casualties per year per severity
sns.barplot(x='Year', y='Number_of_Casualties', hue='Accident_Severity',
data=casualties_by_year_severity, palette='bright', ax=axes[1, 2])
axes[1, 2].set_xlabel('Year')
axes[1, 2].set_ylabel('Number of Casualties')
axes[1, 2].set_title('Number of Casualties per Year per Severity')
axes[1, 2].grid(True)
axes[1, 2].set_xticklabels(axes[1, 2].get_xticklabels(), rotation=45)

plt.tight_layout()
plt.legend()
plt.show()

time.sleep(15)

# Create a subplot for the different conditions that lead to an accident
fig, axes = plt.subplots(3, 3, figsize=(25, 15))
plt.suptitle('Conditions that lead to an Accident', fontsize=20)

# Road surface conditions
sns.barplot(x='Road_Surface_Conditions', y='Accident_Index',
data=accidents_by_surface, palette='bright', ax=axes[0, 0])
axes[0, 0].set_xlabel('Road Surface Condition')
axes[0, 0].set_ylabel('Number of Accidents')
axes[0, 0].set_title('Total Accidents based on Road Surface Condition')
axes[0, 0].grid(True)
axes[0, 0].set_xticklabels(axes[0, 0].get_xticklabels(), rotation=45)

# Road surface conditions per severity
sns.barplot(x='Road_Surface_Conditions', y='Accident_Index',
hue='Accident_Severity', data=accidents_by_surface_severity,
palette='bright', ax=axes[0, 1])
axes[0, 1].set_xlabel('Road Surface Condition')
axes[0, 1].set_ylabel('Number of Accidents')
axes[0, 1].set_title('Total Accidents Severity based on Road Surface
Condition')
axes[0, 1].grid(True)
axes[0, 1].set_xticklabels(axes[0, 1].get_xticklabels(), rotation=45)

# Raod surface pie chart
accidents_vehicles_casualties['Road_Surface_Conditions'].value_counts().plot(
kind='pie', autopct='%1.1f%%', shadow=True, ax=axes[0, 2])
axes[0, 2].set_title('Road Surface Condition')
axes[0, 2].set_ylabel('')
axes[0, 2].set_xlabel('')
axes[0, 2].set_xticklabels(axes[0, 2].get_xticklabels(), rotation=45)

# Weather conditions
sns.barplot(x='Weather_Conditions', y='Accident_Index',
data=accidents_by_weather, palette='bright', ax=axes[1, 0])
axes[1, 0].set_xlabel('Weather Condition')
axes[1, 0].set_ylabel('Number of Accidents')
axes[1, 0].set_title('Total Accidents based on Weather Condition')
axes[1, 0].grid(True)
axes[1, 0].set_xticklabels(axes[1, 0].get_xticklabels(), rotation=45)
```

```
# Weather conditions per severity
sns.barplot(x='Weather_Conditions', y='Accident_Index',
hue='Accident_Severity', data=accidents_by_weather_severity,
palette='bright', ax=axes[1, 1])
axes[1, 1].set_xlabel('Weather Condition')
axes[1, 1].set_ylabel('Number of Accidents')
axes[1, 1].set_title('Total Accidents Severity based on Weather Condition')
axes[1, 1].grid(True)
axes[1, 1].set_xticklabels(axes[1, 1].get_xticklabels(), rotation=45)

# Weather conditions pie chart
accidents_vehicles_casualties['Weather_Conditions'].value_counts().plot(kind=
'pie', autopct='%1.1f%%', shadow=True, ax=axes[1, 2])
axes[1, 2].set_title('Weather Condition')
axes[1, 2].set_ylabel('')
axes[1, 2].set_xlabel('')
axes[1, 2].set_xticklabels(axes[1, 2].get_xticklabels(), rotation=45)

# Light conditions
sns.barplot(x='Light_Conditions', y='Accident_Index',
data=accidents_by_light, palette='bright', ax=axes[2, 0])
axes[2, 0].set_xlabel('Light Condition')
axes[2, 0].set_ylabel('Number of Accidents')
axes[2, 0].set_title('Total Accidents based on Light Condition')
axes[2, 0].grid(True)
axes[2, 0].set_xticklabels(axes[2, 0].get_xticklabels(), rotation=45)

# Light conditions per severity
sns.barplot(x='Light_Conditions', y='Accident_Index',
hue='Accident_Severity', data=accidents_by_light_severity, palette='bright',
ax=axes[2, 1])
axes[2, 1].set_xlabel('Light Condition')
axes[2, 1].set_ylabel('Number of Accidents')
axes[2, 1].set_title('Total Accidents Severity based on Light Condition')
axes[2, 1].grid(True)
axes[2, 1].set_xticklabels(axes[2, 1].get_xticklabels(), rotation=45)

# Light conditions pie chart
accidents_vehicles_casualties['Light_Conditions'].value_counts().plot(kind='p
ie', autopct='%1.1f%%', shadow=True, ax=axes[2, 2])
axes[2, 2].set_title('Light Condition')
axes[2, 2].set_ylabel('')
axes[2, 2].set_xlabel('')
axes[2, 2].set_xticklabels(axes[2, 2].get_xticklabels(), rotation=45)

plt.tight_layout()
plt.show()

time.sleep(15)

# Create a subplot depicting the damage to the vehicles

# Age of vehicles
accidents_by_age =
accidents_vehicles_casualties.groupby('Age_of_Vehicle').count()['Accident_Ind
ex'].reset_index()
```

```
# Type of vehicles
accidents_by_vehicle_affected =
accidents_vehicles_casualties.groupby('Vehicle_Type').count()['Accident_Index
'].reset_index()

# 1st point of impact
accidents_by_impact =
accidents_vehicles_casualties.groupby('1st_Point_of_Impact').count()['Acciden
t_Index'].reset_index()

fig, axes = plt.subplots(2, 2, figsize=(20, 10))
plt.suptitle('Vehicles affected in Accidents', fontsize=20)

# Number of vehicles affected
sns.barplot(x='Number_of_Vehicles', y='Accident_Index',
data=accidents_by_vehicles, palette='bright', ax=axes[0, 0])
axes[0, 0].set_xlabel('Number of Vehicles')
axes[0, 0].set_ylabel('Number of Accidents')
axes[0, 0].set_title('Number of Accidents based on Number of Vehicles')
axes[0, 0].grid(True)
axes[0, 0].set_xticklabels(axes[0, 0].get_xticklabels(), rotation=45)

# Age of vehicles affected
sns.barplot(x='Age_of_Vehicle', y='Accident_Index', data=accidents_by_age,
palette='bright', ax=axes[0, 1])
axes[0, 1].set_xlabel('Age of Vehicles')
axes[0, 1].set_ylabel('Number of Accidents')
axes[0, 1].set_title('Number of Accidents based on Age of Vehicles')
axes[0, 1].grid(True)

# Type of vehicles affected
sns.barplot(x='Vehicle_Type', y='Accident_Index',
data=accidents_by_vehicle_affected, palette='bright', ax=axes[1, 0])
axes[1, 0].set_xlabel('Type of Vehicles')
axes[1, 0].set_ylabel('Number of Accidents')
axes[1, 0].set_title('Number of Accidents based on Type of Vehicles')
axes[1, 0].grid(True)
axes[1, 0].set_xticklabels(axes[1, 0].get_xticklabels(), rotation=45)

# 1st point of impact on the vehicles affected
sns.barplot(x='1st_Point_of_Impact', y='Accident_Index',
data=accidents_by_impact, palette='bright', ax=axes[1, 1])
axes[1, 1].set_xlabel('1st Point of Impact')
axes[1, 1].set_ylabel('Number of Accidents')
axes[1, 1].set_title('Number of Accidents based on 1st Point of Impact')
axes[1, 1].grid(True)
axes[1, 1].set_xticklabels(axes[1, 1].get_xticklabels(), rotation=45)

plt.tight_layout()
plt.show()

time.sleep(15)

#----------------------------------------
# Create a subplot for dist plot, histogram with KDE, KDE plot, Q-Q plot
#----------------------------------------
```

```
def plot_continuous_columns(df, columns):
    for col in columns:
        if col in df.columns:

            fig, axes = plt.subplots(4, 1, figsize=(25, 36))

            # Displot
            sns.histplot(df[col], kde=False, ax=axes[0])
            axes[0].set_title(f'Distribution Plot for {col}')
            axes[0].set_xlabel(col)
            axes[0].set_ylabel('Density')

            # Histogram with KDE
            sns.histplot(df[col], kde=True, ax=axes[1], stat="density",
linewidth=3, alpha=0.6)
            axes[1].set_title(f'Histogram with KDE for {col}')
            axes[1].set_xlabel(col)
            axes[1].set_ylabel('Density')

            # KDE plot
            sns.kdeplot(df[col], ax=axes[2], fill=True)
            axes[2].set_title(f'KDE Plot for {col}')
            axes[2].set_xlabel(col)
            axes[2].set_ylabel('Density')

            # Q-Q plot
            stats.probplot(df[col], dist="norm", plot=axes[3])
            axes[3].set_title(f'Q-Q plot of {col}')
            axes[3].set_xlabel('Theoretical Quantiles')
            axes[3].set_ylabel('Ordered Values')

            time.sleep(5)

            plt.tight_layout()
            plt.show()
        else:
            print(f'Column {col} not found in DataFrame')

plot_continuous_columns(accidents_vehicles_casualties, cont_columns)

time.sleep(10)


#----------------------------------------
# Area plots
#----------------------------------------

# Area plots of casualties per year
plt.figure(figsize=(10, 6))
casualties_by_year.plot.area(x='Year', y='Number_of_Casualties',
color='black', linewidth=3, alpha=0.6)
plt.xlabel('Year')
plt.ylabel('Number of Casualties')
plt.title('Number of Casualties per Year')
plt.xticks(rotation=45)
plt.grid(True)
```

```
plt.show()

time.sleep(10)

# Area plot of number of vehicles affected per year
plt.figure(figsize=(10, 6))
accidents_by_year.plot.area(x='Year', y='Accident_Index', color='black',
linewidth=3, alpha=0.6)
plt.xlabel('Year')
plt.ylabel('Number of Vehicles affected')
plt.title('Number of Vehicles affected per Year')
plt.xticks(rotation=45)
plt.grid(True)
plt.show()

time.sleep(10)

#-----------------------------------------------
# Joint plot with KDE and scatter representation
#-----------------------------------------------

# Create a joint plot of speed limit and engine capacity
plt.figure(figsize=(10, 6))
sns.jointplot(x='Speed_limit', y='Engine_Capacity_(CC)',
data=accidents_vehicles_casualties, kind='kde', color='black')
plt.title('Joint Plot of Speed Limit and Engine Capacity')
plt.tight_layout()
plt.show()

time.sleep(5)

# Create a joint plot of age of driver and number of casualties
plt.figure(figsize=(10, 6))
sns.jointplot(x='Age_of_Driver', y='Number_of_Casualties',
data=accidents_vehicles_casualties, kind='kde', color='black')
plt.title('Joint Plot of Age of Driver and Number of Casualties')
plt.tight_layout()
plt.show()

time.sleep(5)

# Create a joint plot of age of vehicle and number of casualties
plt.figure(figsize=(10, 6))
sns.jointplot(x='Age_of_Vehicle', y='Number_of_Casualties',
data=accidents_vehicles_casualties, kind='kde', color='black')
plt.title('Joint Plot of Age of Vehicle and Number of Casualties')
plt.tight_layout()
plt.show()

time.sleep(5)

#----------------------------------------
# Rug plots
#----------------------------------------

# Create a rug plot of age of driver
plt.figure(figsize=(10, 6))
```

```
sns.rugplot(x='Age_of_Driver', data=accidents_vehicles_casualties)
plt.title('Rug Plot of Age of Driver')
plt.show()

time.sleep(5)

# Create a rug plot of age of vehicle
plt.figure(figsize=(10, 6))
sns.rugplot(x='Age_of_Vehicle', data=accidents_vehicles_casualties)
plt.title('Rug Plot of Age of Vehicle')
plt.show()

time.sleep(5)

# Create a reg plot of engine capacity
plt.figure(figsize=(10, 6))
sns.rugplot(x='Engine_Capacity_(CC)', data=accidents_vehicles_casualties)
plt.title('Rug Plot of Engine Capacity')
plt.show()

time.sleep(5)

#----------------------------------------
# Cluster map
#----------------------------------------

# Create a cluster map
plt.figure(figsize=(10, 6))
sns.clustermap(accidents_by_vehicles, cmap='coolwarm')
plt.title('Cluster Map')
plt.show()

time.sleep(5)

#----------------------------------------
# Hexbin plot
#----------------------------------------

# Create a hexbin plot of age of driver against number of casualties
plt.figure(figsize=(10, 6))
plt.hexbin(x='Age_of_Driver', y='Number_of_Casualties',
data=accidents_vehicles_casualties, gridsize=25, cmap='coolwarm')
plt.title('Hexbin Plot of Age of Driver against Number of Casualties')
plt.show()

time.sleep(5)

# Create a hexbin plot of speed limit against number of vehicles
plt.figure(figsize=(10, 6))
plt.hexbin(x='Speed_limit', y='Number_of_Vehicles',
data=accidents_vehicles_casualties, gridsize=25, cmap='coolwarm')
plt.title('Hexbin Plot of Speed Limit against Number of Vehicles')
plt.show()

time.sleep(5)

#----------------------------------------
```

```
# Strip plot
#----------------------------------------

# Create a strip plot of engine capacity
plt.figure(figsize=(10, 6))
sns.stripplot(x='Engine_Capacity_(CC)', data=accidents_vehicles_casualties)
plt.title('Strip Plot of Engine Capacity')
plt.show()

time.sleep(5)

# Create a strip plot of speed limit
plt.figure(figsize=(10, 6))
sns.stripplot(x='Speed_limit', data=accidents_vehicles_casualties)
plt.tight_layout()
plt.show()

time.sleep(5)

# Create a strip plot of age of driver
plt.figure(figsize=(10, 6))
sns.stripplot(x='Age_of_Driver', data=accidents_vehicles_casualties)
plt.title('Strip Plot of Age of Driver')
plt.show()

time.sleep(5)

#----------------------------------------
# Violin plots and swarm plot
#----------------------------------------

# Create a subplot of violin plots and swarm plot for speed limit, engine
capacity

fig, axes = plt.subplots(2, 2, figsize=(20, 10))
plt.suptitle('Subplots of Violin Plots and Swarm Plot', fontsize=20)

# Speed limit
sns.violinplot(x='Speed_limit', y='Accident_Index', data=accidents_by_speed,
palette='bright', ax=axes[0, 0])
axes[0, 0].set_xlabel('Speed Limit')
axes[0, 0].set_ylabel('Number of Accidents')
axes[0, 0].set_title('Number of Accidents due to Speed Limit')
axes[0, 0].set_xticklabels(axes[0, 0].get_xticklabels(), rotation=45)

sns.swarmplot(x='Speed_limit', y='Accident_Index', data=accidents_by_speed,
color='black', ax=axes[0, 1])
axes[0, 1].set_xlabel('Speed Limit')
axes[0, 1].set_ylabel('Number of Accidents')
axes[0, 1].set_title('Number of Accidents due to Speed Limit')
axes[0, 0].set_xticklabels(axes[0, 0].get_xticklabels(), rotation=45)

# Engine capacity
sns.violinplot(x='Engine_Capacity_(CC)', y='Accident_Index',
data=accidents_vehicles_casualties, palette='bright', ax=axes[1, 0])
axes[1, 0].set_xlabel('Engine Capacity')
axes[1, 0].set_ylabel('Number of Accidents')
```

```
axes[1, 0].set_title('Engine capacity of vehicles involved in Accidents')
axes[0, 0].set_xticklabels(axes[0, 0].get_xticklabels(), rotation=45)

sns.swarmplot(x='Engine_Capacity_(CC)', y='Accident_Index',
data=accidents_vehicles_casualties, color='black', ax=axes[1, 1])
axes[1, 1].set_xlabel('Engine Capacity')
axes[1, 1].set_ylabel('Number of Accidents')
axes[1, 1].set_title('Engine capacity of vehicles involved in Accidents')
axes[0, 0].set_xticklabels(axes[0, 0].get_xticklabels(), rotation=45)

plt.tight_layout()
plt.grid(True)
plt.show()

time.sleep(10)

# Violin plot of speed limit
plt.figure(figsize=(10, 6))
sns.violinplot(x='Speed_limit', y='Accident_Index', data=accidents_by_speed,
palette='bright')
plt.xlabel('Speed Limit')
plt.ylabel('Number of Accidents')
plt.title('Number of Accidents due to Speed Limit')
plt.xticks(rotation=45)
plt.grid(True)
plt.show()

time.sleep(5)

accidents_vehicles_casualties.drop(['Location_Easting_OSGR',
'Location_Northing_OSGR', 'Longitude', 'Latitude',
                                    'Police_Force',
'Local_Authority_(District)', 'Local_Authority_(Highway)',
                                    '1st_Road_Number', '2nd_Road_Number',
'Accident_Index', 'Date', 'Time',
                                    'LSOA_of_Accident_Location', 'Year',
'Journey_Purpose_of_Driver', 'Sex_of_Driver',
                                    'Skidding_and_Overturning',
'Special_Conditions_at_Site', 'Carriageway_Hazards',
                                    'Hit_Object_in_Carriageway',
'Hit_Object_off_Carriageway', 'Vehicle_Leaving_Carriageway',
                                    'Hit_Object_off_Carriageway',
'Driver_IMD_Decile', 'Age_of_Casualty', 'Casualty_Type',
                                    'Vehicle_Reference_x',
'Vehicle_Reference_y', 'Pedestrian_Crossing-Human_Control',
                                    'Pedestrian_Crossing-
Physical_Facilities', 'Pedestrian_Road_Maintenance_Worker',
'Casualty_Home_Area_Type',
                                    'Age_Band_of_Driver',
'Age_Band_of_Casualty', 'Pedestrian_Location', 'Pedestrian_Movement',
'Bus_or_Coach_Passenger',
                                    'Driver_Home_Area_Type'], axis=1,
inplace=True)

#----------------------------------------
# Box plot
#----------------------------------------
```

```
plt.figure(figsize=(20, 10))
sns.boxplot(data=accidents_vehicles_casualties, orient='h')
plt.show()

#----------------------------------------
# IQR
#----------------------------------------

# Create a function to calculate the IQR
def iqr(df):
    q1 = df.quantile(0.25)
    q3 = df.quantile(0.75)
    iqr = q3 - q1
    lower = q1 - 1.5 * iqr
    upper = q3 + 1.5 * iqr
    return lower, upper

lower, upper = iqr(accidents_vehicles_casualties['Engine_Capacity_(CC)'])
accidents_vehicles_casualties =
accidents_vehicles_casualties[(accidents_vehicles_casualties['Engine_Capacity
_(CC)'] > lower) & (accidents_vehicles_casualties['Engine_Capacity_(CC)'] <
upper)]

# Create a box plot of the data after removing the outliers
plt.figure(figsize=(20, 10))
sns.boxplot(data=accidents_vehicles_casualties, orient='h')
plt.show()

# ----------------------------------------
# Normality test
# ----------------------------------------

# Create a function to test the normality of the data to check if numerical
data is normally distributed
def shapiro_test(x, title):
    stats, p = shapiro(x.dropna())
    print('=' * 50)
    print(f'Shapiro test : {title} dataset : statistics = {stats:.2f} p-value
of ={p:.2f}' )
    if p > 0.05:
        print(f'{title} dataset is normal')
    else:
        print(f'{title} dataset is not normal')


shapiro_test(accidents_vehicles_casualties['Number_of_Vehicles'],
'Number_of_Vehicles')
shapiro_test(accidents_vehicles_casualties['Number_of_Casualties'],
'Number_of_Casualties')
shapiro_test(accidents_vehicles_casualties['Speed_limit'], 'Speed_limit')
shapiro_test(accidents_vehicles_casualties['Age_of_Driver'], 'Age_of_Driver')
shapiro_test(accidents_vehicles_casualties['Engine_Capacity_(CC)'],
'Engine_Capacity_(CC)')
shapiro_test(accidents_vehicles_casualties['Age_of_Vehicle'],
'Age_of_Vehicle')
shapiro_test(accidents_vehicles_casualties['Age_of_Driver'], 'Age_of_Driver')
```

```
# Remove all the negative values from the numerical columns
accidents_vehicles_casualties =
accidents_vehicles_casualties[(accidents_vehicles_casualties['Age_of_Driver']
> 0) & (accidents_vehicles_casualties['Age_of_Vehicle'] > 0) &
(accidents_vehicles_casualties['Engine_Capacity_(CC)'] > 0)]


# If the data is not normally distributed, we can make it gaussian by
applying a boxcox transformation
def boxcox_transform(x):
    transformed_data, lamda = boxcox(x)
    print(f'Best lambda value: {lamda:.2f}')

    # After transformation
    sns.distplot(transformed_data, kde=True)
    plt.title(f'After Transformation of {x.name}')
    plt.show()

    time.sleep(2)

boxcox_transform(accidents_vehicles_casualties['Number_of_Vehicles'])
boxcox_transform(accidents_vehicles_casualties['Number_of_Casualties'])
boxcox_transform(accidents_vehicles_casualties['Speed_limit'])
boxcox_transform(accidents_vehicles_casualties['Age_of_Driver'])
boxcox_transform(accidents_vehicles_casualties['Engine_Capacity_(CC)'])
boxcox_transform(accidents_vehicles_casualties['Age_of_Vehicle'])
boxcox_transform(accidents_vehicles_casualties['Age_of_Driver'])


cat_cols =
list(accidents_vehicles_casualties.select_dtypes(include=['object']).columns)
cat_cols.remove('Accident_Severity')

#----------------------------------------
# 3d plot and contour plot
#----------------------------------------

# Create a 3d plot of number of vehicles, number of casualties, and speed
limit
fig = plt.figure(figsize=(10, 6))
ax = fig.add_subplot(111, projection='3d')
ax.scatter(xs=accidents_vehicles_casualties['Number_of_Vehicles'],
ys=accidents_vehicles_casualties['Number_of_Casualties'],
zs=accidents_vehicles_casualties['Speed_limit'], c='black', marker='o')
ax.set_xlabel('Number of Vehicles')
ax.set_ylabel('Number of Casualties')
ax.set_zlabel('Speed Limit')
plt.title('3D Plot of Number of Vehicles, Number of Casualties, and Speed
Limit')
plt.show()

time.sleep(5)

# Create a contour plot of number of vehicles, number of casualties, and
speed limit
accidents_axis = accidents_vehicles_casualties[['Number_of_Vehicles',
```

```
'Number_of_Casualties', 'Speed_limit']]

X, Y = np.meshgrid(np.linspace(accidents_axis['Number_of_Vehicles'].min(),
accidents_axis['Number_of_Vehicles'].max(), 100),
                   np.linspace(accidents_axis['Number_of_Casualties'].min(),
accidents_axis['Number_of_Casualties'].max(), 100))

Z = griddata((accidents_axis['Number_of_Vehicles'],
accidents_axis['Number_of_Casualties']),
             accidents_axis['Speed_limit'],
             (X, Y), method='cubic')

fig = plt.figure(figsize=(10, 6))
ax = fig.add_subplot(111, projection='3d')
ax.contour(X, Y, Z)
ax.set_xlabel('Number of Vehicles')
ax.set_ylabel('Number of Casualties')
ax.set_zlabel('Speed Limit')
plt.title('Contour Plot of Number of Vehicles, Number of Casualties, and
Speed Limit')
plt.show()



# One-hot encode the categorical columns
accidents_vehicles_casualties = pd.get_dummies(accidents_vehicles_casualties,
columns=cat_cols, drop_first=True)

# Scale the continuous columns
def scale(x):
    return (x - x.min()) / (x.max() - x.min())

accidents_vehicles_casualties[cont_columns] =
accidents_vehicles_casualties[cont_columns].apply(scale, axis=0)


# Divide the data into features and target
X, y = accidents_vehicles_casualties.drop('Accident_Severity', axis=1),
accidents_vehicles_casualties['Accident_Severity']

#Oversample the minority class
oversample = SMOTE()
X_smote, y_smote = oversample.fit_resample(X, y)

# Check the balance of the classes
print('Processisng SMOTE...')
fig, ax = plt.subplots(1, 2, figsize = (10, 5))
y.value_counts().plot.pie(explode = [0, 0.1, 0.2], autopct = "%1.1f%%", ax =
ax[0], colors = ['#ff9999', '#66b3ff', '#964B00'],
                          shadow = True)
ax[0].set_title("Before SMOTE")
ax[0].set_ylabel('')
y_smote.value_counts().plot.pie(explode = [0, 0.1, 0.2], autopct = "%1.1f%%",
ax = ax[1],  colors = ['#66b3ff', '#ff9999', '#964B00'],
                                shadow = True)
ax[1].set_title("After SMOTE")
ax[1].set_ylabel('')
```

```
print("Overall dataset values have been increased.")
print("Original size:\n", y.value_counts())
print("New size:\n", y_smote.value_counts())
plt.show()

X_smote = sm.add_constant(X_smote)

print('New size of dataset:', X_smote.shape, y_smote.shape)

#----------------------------------------
# PCA
#----------------------------------------

# Start PCA
pca = PCA(svd_solver="full", n_components=0.9, random_state=5764)
X_pca = pca.fit_transform(X_smote)
print("Original Shape: ", X_smote.shape)
print("Reduced Shape: ", X_pca.shape)

print("Number of features needed to explain more than 90% of the dependent
variance:",
    np.where(np.cumsum(pca.explained_variance_ratio_) > 0.9)[0][0] + 1, )

plt.figure(figsize=(10, 10))
plt.plot(
    np.arange(1, len(np.cumsum(pca.explained_variance_ratio_)) + 1, 1),
    np.cumsum(pca.explained_variance_ratio_), label="Cumulative Explained
Variance",)
plt.xticks(np.arange(1, len(np.cumsum(pca.explained_variance_ratio_)) + 1,
1))
plt.xlabel("Number of Components")
plt.ylabel("Cumulative Explained Variance")
plt.grid(True)
plt.axvline(x=(np.where(np.cumsum(pca.explained_variance_ratio_) > 0.9)[0][0]
+ 1, ),
            color="red", linestyle="--")
plt.axhline(y=0.9, color="black", linestyle="--")
plt.legend()
plt.show()

# Conditional Number of Original and Reduced data
print('Condition Number of Original data:', np.linalg.cond(X_smote))
print('Condition Number of Reduced data:', np.linalg.cond(X_pca))

#----------------------------------------
#End of Exploratory Data Analysis
#----------------------------------------
```

Phase 2: Interactive Dashboard

```
import pandas as pd
import time
import plotly.express as px
from dash import Dash
from dash import dcc, html
from dash.dependencies import Input, Output, State
import dash_bootstrap_components as dbc
from dash_bootstrap_templates import load_figure_template
from dash.exceptions import PreventUpdate
import warnings

pd.set_option('display.float_format', lambda x: '%.2f' % x)
load_figure_template('CYBORG')
warnings.filterwarnings("ignore")

# Read in the data
start_time = time.time()
accidents_vehicles_casualties = pd.DataFrame()

for chunk in pd.read_csv('UK_Accidents_Merger_Cleaned.csv', chunksize=200000,
low_memory=False):
    print('Number of chunks read: ', chunk.shape)
    accidents_vehicles_casualties = pd.concat([accidents_vehicles_casualties,
chunk])

print("Time taken to read the data: ", time.time() - start_time)

# Drop or impute missing values
accidents_vehicles_casualties.dropna(inplace=True)

# Viusalizations using Plotly

# Convert date to year
accidents_vehicles_casualties['Date'] =
pd.to_datetime(accidents_vehicles_casualties['Date'])
accidents_vehicles_casualties['Year'] =
accidents_vehicles_casualties['Date'].dt.year

# Remove dataset features
accidents_vehicles_casualties.drop(['Location_Easting_OSGR',
'Location_Northing_OSGR','Local_Authority_(District)',
                                    'Local_Authority_(Highway)',
'1st_Road_Number', '2nd_Road_Number',
                                    'Pedestrian_Crossing-Human_Control',
                                    'Pedestrian_Crossing-
Physical_Facilities', 'Vehicle_Reference_x', 'Vehicle_Reference_y',
                                    'Pedestrian_Location',
'Pedestrian_Movement', 'Pedestrian_Road_Maintenance_Worker',
                                    'Casualty_Home_Area_Type',
'Age_Band_of_Driver', 'Age_Band_of_Casualty'], axis=1, inplace=True)

accidents_vehicles_casualties['Junction_Detail'] =
accidents_vehicles_casualties['Junction_Detail'].replace({
    'T or staggered junction': 'Staggered Junction',
    'Private drive or entrance': 'Entrance',
```

```
    'Other junction': 'Others',
    'More then 4 arms (not roundabout)': 'Fours',
    'Mini-roundabout': 'Miniroundabout',
    'Not at junction or within 20 metres': 'No Junction',
    'Data missing or out of range': 'Unknown',
})

accidents_vehicles_casualties['Junction_Control'] =
accidents_vehicles_casualties['Junction_Control'].replace({
    'Authorised person': 'Authorised',
    'Data missing or out of range': 'Unknown',
    'Give way or uncontrolled': 'Uncontrolled',
    'Stop sign': 'Stop',
    'Auto traffic signal': 'Traffic Signal',
})


accidents_vehicles_casualties['Light_Conditions'] =
accidents_vehicles_casualties['Light_Conditions'].replace({
    'Darkness - no lighting': 'Darkness',
    'Darkness - lighting unknown': 'Unknown',
    'Darkness - lights lit': 'Light',
    'Darkness - lights unlit': 'Unlit'
})

accidents_vehicles_casualties['Weather_Conditions'] =
accidents_vehicles_casualties['Weather_Conditions'].replace({
    'Fine without high winds': 'Fine',
    'Raining without high winds': 'Rain',
    'Raining + high winds': 'Rainwinds',
    'Fine + high winds': 'Fine',
    'Fog or mist': 'Fog',
    'Snowing without high winds': 'Snow',
    'Snowing + high winds': 'Snowind',
    'Data missing or out of range': 'Unknown',
})

accidents_vehicles_casualties['Road_Surface_Conditions'] =
accidents_vehicles_casualties['Road_Surface_Conditions'].replace({
    'Dry': 'Dry',
    'Wet or damp': 'Wet',
    'Frost or ice': 'Ice',
    'Snow': 'Snow',
    'Flood over 3cm. deep': 'Flood',
    'Data missing or out of range': 'Unknown',
})

accidents_vehicles_casualties['Did_Police_Officer_Attend_Scene_of_Accident']
=
accidents_vehicles_casualties['Did_Police_Officer_Attend_Scene_of_Accident'].
replace(
    'No - accident was reported using a self completion  form (self rep
only)', 'Self')

accidents_vehicles_casualties['Vehicle_Type'] =
accidents_vehicles_casualties['Vehicle_Type'].replace({
    'Bus or coach (17 or more pass seats)': 'Bus',
```

```
    'Van / Goods 3.5 tonnes mgw or under': 'Van',
    'Taxi/Private hire car': 'Taxi',
    'Motorcycle 125cc and under': 'Motorcycle',
    'Motorcycle over 500cc': 'Motorcycle',
    'Goods 7.5 tonnes mgw and over': 'Goods',
    'Motorcycle 50cc and under': 'Motorcycle',
    'Motorcycle over 125cc and up to 500cc': 'Motorcycle',
    'Goods over 3.5t. and under 7.5t': 'Goods',
    'Other vehicle': 'Others',
    'Minibus (8 - 16 passenger seats)': 'Minibus',
    'Agricultural vehicle (includes diggers etc.)': 'Agricultural',
    'Motorcycle - unknown cc': 'Motorcycle'
})

accidents_vehicles_casualties['Towing_and_Articulation'] =
accidents_vehicles_casualties['Towing_and_Articulation'].replace({
    'No tow/articulation': 'No',
    'Articulated vehicle': 'Articulated',
    'Single trailer': 'Single',
    'Other tow': 'Others',
    'Double or multiple trailer': 'Multiple',
    'Data missing or out of range': 'Unknown',
})

accidents_vehicles_casualties['Vehicle_Manoeuvre'] =
accidents_vehicles_casualties['Vehicle_Manoeuvre'].replace({
    'Going ahead other': 'Going ahead',
    'Turning right': 'Right',
    'Waiting to go - held up': 'Waiting',
    'Slowing or stopping': 'Slowing',
    'Turning left': 'Left',
    'Moving off': 'Moving',
    'Waiting to turn right': 'Waiting',
    'Going ahead right-hand bend': 'Going ahead',
    'Going ahead left-hand bend': 'Going ahead',
    'Overtaking moving vehicle - offside': 'Overtaking',
    'Waiting to turn left': 'Waiting',
    'Overtaking static vehicle - offside': 'Overtaking',
    'Changing lane to left': 'Changing lane',
    'Changing lane to right': 'Changing lane',
    'U-turn': 'Uturn',
    'Overtaking - nearside': 'Overtaking',
    'Data missing or out of range': 'Unknown',
})

accidents_vehicles_casualties['Vehicle_Location-Restricted_Lane'] =
accidents_vehicles_casualties['Vehicle_Location-Restricted_Lane'].replace({
    'On main c\way - not in restricted lane': 'Mainlane',
    'Bus lane': 'Buslane',
    'Footway (pavement)': 'Footway',
    'Leaving lay-by or hard shoulder': 'Leavinglayby',
    'On lay-by or hard shoulder': 'Onlayby',
    'Busway (including guided busway)': 'Busway',
    'Cycle lane (on main carriageway)': 'Cyclelane',
    'Tram/Light rail track': 'Tramtrack',
    'Entering lay-by or hard shoulder': 'Enteringlayby',
    'Cycleway or shared use footway (not part of  main carriageway)':
```

```
'Cycleway',
    'Data missing or out of range': 'Unknown',
})

accidents_vehicles_casualties['Junction_Location'] =
accidents_vehicles_casualties['Junction_Location'].replace({
    'Approaching junction or waiting/parked at junction approach':
'Approaching',
    'Mid Junction - on roundabout or on main road': 'Mid',
    'Cleared junction or waiting/parked at junction exit': 'Cleared',
    'Entering from slip road': 'Entering',
    'Leaving main road into minor road': 'Leaving',
    'Entering main road from minor road': 'Entering',
    'Leaving roundabout': 'Leaving',
    'Entering roundabout': 'Entering',
    'Data missing or out of range': 'Unknown',
    'Not at or within 20 metres of junction': 'No Junction',
})

accidents_vehicles_casualties['1st_Point_of_Impact'] =
accidents_vehicles_casualties['1st_Point_of_Impact'].replace({
    'Did not impact': 'Nothing',
    'Data missing or out of range': 'Unknown'
})

accidents_vehicles_casualties['Was_Vehicle_Left_Hand_Drive?'] =
accidents_vehicles_casualties['Was_Vehicle_Left_Hand_Drive?'].replace({
    'Data missing or out of range': 'Unknown'
})

accidents_vehicles_casualties['Propulsion_Code'] =
accidents_vehicles_casualties['Propulsion_Code'].replace({
    'Gas/Bi-fuel': 'Gas',
    'Petrol/Gas (LPG)': 'LPG',
})

accidents_vehicles_casualties['Casualty_Class'] =
accidents_vehicles_casualties['Casualty_Class'].replace({
    'Driver or rider': 'Driver'
})

accidents_vehicles_casualties['Sex_of_Casualty'] =
accidents_vehicles_casualties['Sex_of_Casualty'].replace({
    'Data missing or out of range': 'Unknown'
})

accidents_vehicles_casualties['Car_Passenger'] =
accidents_vehicles_casualties['Car_Passenger'].replace({
    'Data missing or out of range': 'Unknown'
})

accidents_vehicles_casualties['Urban_or_Rural_Area'] =
accidents_vehicles_casualties['Urban_or_Rural_Area'].replace({
    1: 'Urban',
    2: 'Rural',
    3: 'Unallocated'
})
```

```
accidents_vehicles_casualties['Special_Conditions_at_Site'] =
accidents_vehicles_casualties['Special_Conditions_at_Site'].replace({
    'Auto traffic signal - out': 'No signal',
    'Auto signal part defective': 'Signal defect',
    'Oil or diesel': 'Oil',
    'Road sign or marking defective or obscured': 'Sign defect',
    'Road surface defective': 'Road defect'
})

# Create a dash app
uk_accidents_app = Dash("UK Accident Dashboard",
external_stylesheets=[dbc.themes.CYBORG])

# Layout of the app
uk_accidents_app.layout = html.Div(style={'padding': '10px 10px 10px 10px'},
    children=[

    html.H1("UK Accidents Dashboard", style={'textAlign': 'center', 'color':
'#503D36'}),

    html.Div("Explore the relationships between UK Accidents, Casualties, and
Vehicles...", style={'textAlign': 'center', 'color': '#F57241'}),

    html.Br(),

    # Put tabs in left side not in middle
    dcc.Tabs(id='Tabs', value='about-tool', children=[
        dcc.Tab(label='About Project', id='About tool', value='about-tool'),
        dcc.Tab(label='Subplots', id='Subplot tool', value='subplots-tool'),
        dcc.Tab(label='Individual Line Plots', id='Scatter tool',
value='line-plot-tool'),
        dcc.Tab(label='Individual Bar Plots', id='Bar tool', value='bar-plot-
tool'),
        dcc.Tab(label='Individual Pie Plots', id='Pie tool', value='pie-plot-
tool'),
        dcc.Tab(label='Individual Count Plots', id='Count tool',
value='count-plot-tool'),
        dcc.Tab(label='Individual Histograms', id='Histogram tool',
value='histogram-tool'),
        dcc.Tab(label="Other Miscellaneous Plots", id="Misc tool",
value="misc-plot-tool"),
        dcc.Tab(label='Geospatial Plots', id='Geospatial tool',
value='geospatial-tool')
        ],
    colors={
        "border": "black",
        "primary": "gold",
        "background": "black"
    },
    ),
    html.Div(id='tabs-content')
    ]
)

@uk_accidents_app.callback(
    Output(component_id='tabs-content', component_property='children'),
```

```
    [Input(component_id='Tabs', component_property='value')])

def render_content(tab):

    if tab == 'about-tool':
        return tab_about

    elif tab == 'subplots-tool':
        return sub_plots

    elif tab == 'line-plot-tool':
        return line_plots

    elif tab == 'bar-plot-tool':
        return bar_plots

    elif tab == 'pie-plot-tool':
        return pie_plots

    elif tab == 'count-plot-tool':
        return count_plots

    elif tab == 'histogram-tool':
        return histogram_plots

    elif tab == 'misc-plot-tool':
        return misc_plots

    elif tab == 'geospatial-tool':
        return geo_plots

#---------------------------------
# About Project
#---------------------------------

# Create about project tab
tab_about = html.Div(
    children=[

    html.H4('About Project', style={'textAlign': 'center', 'color':
'#503D36', 'font-size': '40px'}),


html.Img(src='https://t3.ftcdn.net/jpg/03/72/46/46/360_F_372464646_Ks082AREON
EjY5XYhWSexdDGFQ9tHr8S.jpg', style={'width': '70%',

'display': 'block',

'margin-left': 'auto',

'margin-right': 'auto',

'padding': '10px'}),
    html.P(
        'This project is about exploring the relationships between UK
Accidents, Vehicles, and Casualties. The data is from the UK government
website.',
```

```
        style={'margin': '10px 0', 'color': 'white'}
    ),
    html.P(
        'The United Kingdom Police Forces collects data on every vehicle
collision in the UK on a form called Stats19.',
        style={'margin': '10px 0', 'color': 'white'}
    ),
    html.P([
        'Data from this form ends up at the DFT and is published at ',
        html.A('https://data.gov.uk/dataset/road-accidents-safety-data',
               href='https://data.gov.uk/dataset/road-accidents-safety-data',
               target='_blank',
               style={'color': 'lightblue'}),
    ], style={'margin': '10px 0', 'color': 'white'}),
    html.P(
        'This project aims to delve into the intricate relationships and
patterns that exist among accidents, vehicles, and casualties in the UK,
spanning a substantial period from 2005 to 2014.',
        style={'margin': '10px 0', 'color': 'white'}
    ),
    html.P(
        'The analysis of such a dataset can reveal multi-faceted insights and
correlations that are crucial for understanding the dynamics of road safety
and accident causation in the UK.',
        style={'margin': '10px 0', 'color': 'white'}
    ),
    html.P([
        'The dataset was found on Kaggle at ',
        html.A('https://www.kaggle.com/datasets/benoit72/uk-accidents-10-
years-history-with-many-variables',
               href='https://www.kaggle.com/datasets/benoit72/uk-accidents-
10-years-history-with-many-variables',
               target='_blank',
               style={'color': 'lightblue'}),
    ], style={'margin': '10px 0', 'color': 'white'}),
  ], style={'padding': '20px'}
)

@uk_accidents_app.callback(
    Output(component_id='tab_about', component_property='children'),
    [Input(component_id='tab_about', component_property='figure')])

def render_content(tab_about):

       return tab_about

#-------------------------------
# Subplots
#-------------------------------

# Create a dropdown of subplots:
# 1. Number of Accidents and Casualties per Year
# 2. Conditions that lead to an Accident
# 3. Vehicles affected in Accidents

# Allow only one subplot to be selected at a time
```

```
accidents_by_surface =
accidents_vehicles_casualties.groupby('Road_Surface_Conditions').count()['Acc
ident_Index'].reset_index()
accidents_by_surface_severity =
accidents_vehicles_casualties.groupby(['Road_Surface_Conditions',
'Accident_Severity']).count()['Accident_Index'].reset_index()

accidents_by_weather =
accidents_vehicles_casualties.groupby('Weather_Conditions').count()['Accident
_Index'].reset_index()
accidents_by_weather_severity =
accidents_vehicles_casualties.groupby(['Weather_Conditions',
'Accident_Severity']).count()['Accident_Index'].reset_index()

accidents_by_light =
accidents_vehicles_casualties.groupby('Light_Conditions').count()['Accident_I
ndex'].reset_index()
accidents_by_light_severity =
accidents_vehicles_casualties.groupby(['Light_Conditions',
'Accident_Severity']).count()['Accident_Index'].reset_index()

accidents_by_age =
accidents_vehicles_casualties.groupby('Age_of_Vehicle').count()['Accident_Ind
ex'].reset_index()

accidents_by_vehicle_affected =
accidents_vehicles_casualties.groupby('Vehicle_Type').count()['Accident_Index
'].reset_index()

accidents_by_impact =
accidents_vehicles_casualties.groupby('1st_Point_of_Impact').count()['Acciden
t_Index'].reset_index()

sub_plots = html.Div(
    children=[

    html.Header('The Sub-Plots fpr the given Data', style={'color':
'#F57241'}),

    html.Label('Select a Subplot', style={'color': '#F57241'}),

    html.H4('UK Accidents Subplots', style={'textAlign': 'center', 'color':
'#503D36', 'font-size': 40}),

    dcc.Dropdown(id='dropdown',
                 options=[
                     {'label': 'Number of Accidents and Casualties per Year',
'value': 'accidents_casualties'},
                     {'label': 'Conditions that lead to an Accident',
'value': 'conditions'},
                     {'label': 'Vehicles affected in Accidents', 'value':
'vehicles'}
                 ],
                 value='accidents_casualties',
                 style={'width': '50%'},
                 multi = False,
                 ),
```

```
    html.Br(),

    #Add loading
    dcc.Loading(
        id="loading-1",
        type="default",
        children=html.Div(id='output_container', children=[])
    ),
    html.Br(),

    dcc.Loading(
        id="loading-2",
        type="default",
        children=html.Div(id='graph_container', children=[])
    ),
  ]
)

@uk_accidents_app.callback(
    [Output('output_container', 'children'),
     Output('graph_container', 'children')],
    [Input('dropdown', 'value')]
)

def select_dropdown(value):

    container = "The selected value was: {}".format(value)

    graphs = html.Div()

    if value == 'accidents_casualties':

        fig1 = px.line(accidents_by_year_severity, x='Year',
y='Accident_Index', color='Accident_Severity', height=500,
                           title='Number of accidents per year based on
Severity', template='plotly_dark').update_layout(
                {'plot_bgcolor': 'rgba(0, 0, 0, 0)', 'paper_bgcolor':
'rgba(0, 0, 0, 0)'})

        fig1_bar = px.bar(year_accidents, x='Year', y='Accident_Index',
color='Accident_Index', height=500,
                          title='Number of accidents per year',
template='plotly_dark').update_layout(
                {'plot_bgcolor': 'rgba(0, 0, 0, 0)', 'paper_bgcolor':
'rgba(0, 0, 0, 0)'})

        fig2 = px.line(casualties_by_year_severity, x='Year',
y='Number_of_Casualties', color='Accident_Severity', height=500,
                               title='Number of casulties per year based on
Severity', template='plotly_dark').update_layout(
                {'plot_bgcolor': 'rgba(0, 0, 0, 0)', 'paper_bgcolor':
'rgba(0, 0, 0, 0)'})

        fig2_bar = px.bar(casualties_by_year, x='Year',
y='Number_of_Casualties', color='Number_of_Casualties', height=500,
                               title='Number of casulties per year',
```

```
template='plotly_dark').update_layout(
                {'plot_bgcolor': 'rgba(0, 0, 0, 0)', 'paper_bgcolor':
'rgba(0, 0, 0, 0)'})

        graphs = html.Div(
            children=[
                dcc.Graph(
                    figure=fig1
                ),
                dcc.Graph(
                    figure=fig1_bar
                ),
                dcc.Graph(
                    figure=fig2
                ),
                dcc.Graph(
                    figure=fig2_bar
                ),
            ]
        )


    elif value == 'conditions':

        fig1 = px.bar(accidents_by_surface_severity,
x='Road_Surface_Conditions', y='Accident_Index', color='Accident_Severity',
height=500,
                      title='Number of accidents based on Road Surface
Conditions and Severity', template='plotly_dark').update_layout(
            {'plot_bgcolor': 'rgba(0, 0, 0, 0)', 'paper_bgcolor': 'rgba(0, 0,
0, 0)'})

        fig1_pie = px.pie(accidents_by_surface, values='Accident_Index',
names='Road_Surface_Conditions',
                          title='Pie Chart of Road Surface Conditions',
                          template='plotly_dark').update_layout(
            {'plot_bgcolor': 'rgba(0, 0, 0, 0)', 'paper_bgcolor': 'rgba(0, 0,
0, 0)', 'showlegend': False})

        fig2 = px.bar(accidents_by_weather_severity, x='Weather_Conditions',
y='Accident_Index', color='Accident_Severity', height=500,
                      title='Number of accidents based on Weather Conditions
and Severity', template='plotly_dark').update_layout(
            {'plot_bgcolor': 'rgba(0, 0, 0, 0)', 'paper_bgcolor': 'rgba(0, 0,
0, 0)'})

        fig2_pie = px.pie(accidents_by_weather, values='Accident_Index',
names='Weather_Conditions',
                          title='Pie Chart of Weather Conditions',
                          template='plotly_dark').update_layout(
            {'plot_bgcolor': 'rgba(0, 0, 0, 0)', 'paper_bgcolor': 'rgba(0, 0,
0, 0)', 'showlegend': False})

        fig3 = px.bar(accidents_by_light_severity, x='Light_Conditions',
y='Accident_Index', color='Accident_Severity', height=500,
                      title='Number of accidents based on Light Conditions
and Severity', template='plotly_dark').update_layout(
```

```
            {'plot_bgcolor': 'rgba(0, 0, 0, 0)', 'paper_bgcolor': 'rgba(0, 0,
0, 0)'})

        fig3_pie = px.pie(accidents_by_light, values='Accident_Index',
names='Light_Conditions',
                          title='Pie Chart of Light Conditions',
                          template='plotly_dark').update_layout(
            {'plot_bgcolor': 'rgba(0, 0, 0, 0)', 'paper_bgcolor': 'rgba(0, 0,
0, 0)', 'showlegend': False})

        graphs = html.Div(
            children=[
                dcc.Graph(
                    figure=fig1
                ),
                dcc.Graph(
                    figure=fig1_pie
                ),
                dcc.Graph(
                    figure=fig2
                ),
                dcc.Graph(
                    figure=fig2_pie
                ),
                dcc.Graph(
                    figure=fig3
                ),
                dcc.Graph(
                    figure=fig3_pie
                ),
            ]
        )
        return container, graphs

    elif value == 'vehicles':

        fig1 = px.bar(accidents_by_age, x='Age_of_Vehicle',
y='Accident_Index', color='Accident_Index', height=500,
                      title='Number of accidents based on Age of Vehicle',
template='plotly_dark').update_layout(
            {'plot_bgcolor': 'rgba(0, 0, 0, 0)', 'paper_bgcolor': 'rgba(0, 0,
0, 0)'})

        fig2 = px.bar(accidents_by_vehicle_affected, x='Vehicle_Type',
y='Accident_Index', color='Accident_Index', height=500,
                      title='Number of accidents based on Vehicle Type',
template='plotly_dark').update_layout(
            {'plot_bgcolor': 'rgba(0, 0, 0, 0)', 'paper_bgcolor': 'rgba(0, 0,
0, 0)'})

        fig3 = px.bar(accidents_by_impact, x='1st_Point_of_Impact',
y='Accident_Index', color='Accident_Index', height=500,
                      title='Number of accidents based on First Point of
Impact', template='plotly_dark').update_layout(
            {'plot_bgcolor': 'rgba(0, 0, 0, 0)', 'paper_bgcolor': 'rgba(0, 0,
0, 0)'})
```

```
        graphs = html.Div(
            children=[
                dcc.Graph(
                    figure=fig1
                ),
                dcc.Graph(
                    figure=fig2
                ),
                dcc.Graph(
                    figure=fig3
                ),
            ]
        )

    return container, graphs


#---------------------------------
# Line Plots
#---------------------------------

# Create line graphs such that there should be a dropdown with a download
button to all plots:
# 1. Number of Accidents per Year based on Severity
# 2. Number of Casualties per Year based on Severity

#
year_accidents =
accidents_vehicles_casualties.groupby('Year')['Accident_Index'].count().reset
_index()
accidents_by_year_severity = accidents_vehicles_casualties.groupby(['Year',
'Accident_Severity']).count()['Accident_Index'].reset_index()

casualties_by_year =
accidents_vehicles_casualties.groupby('Year').sum()['Number_of_Casualties'].r
eset_index()
casualties_by_year_severity = accidents_vehicles_casualties.groupby(['Year',
'Accident_Severity']).sum()['Number_of_Casualties'].reset_index()

line_plots = html.Div(
    children=[

    html.Header('The Line Plots for the given Data', style={'color':
'#F57241'}),

    html.Label('Select a Line Plot', style={'color': '#F57241'}),

    html.H4('UK Accidents Line Plots', style={'textAlign': 'center', 'color':
'#503D36', 'font-size': 40}),

    dcc.Dropdown(
        id='graph-dropdown',
        options=[
            {'label': 'Accidents per Year by Severity', 'value':
'accidents'},
            {'label': 'Casualties per Year by Severity', 'value':
'casualties'}
```

```
        ],
        style={'width': '50%'},
        multi=False,
        value='accidents',
    ),

    dcc.Graph(id='display-graph'),

    html.Button('Download Graph', id='download-graph-button'),

    dcc.Download(id='download1'),

    html.Br(),
    ]
)

@uk_accidents_app.callback(
    Output('display-graph', 'figure'),
    [Input('graph-dropdown', 'value')])

def update_graph(graph_type):

    if graph_type == 'accidents':

        fig1 = px.line(accidents_by_year_severity, x='Year',
y='Accident_Index', color='Accident_Severity', height=500,
                           title='Number of accidents per year based on
Severity', template='plotly_dark').update_layout(
                    {'plot_bgcolor': 'rgba(0, 0, 0, 0)', 'paper_bgcolor':
'rgba(0, 0, 0, 0)'})

        return fig1

    else:

        fig2 = px.line(casualties_by_year_severity, x='Year',
y='Number_of_Casualties', color='Accident_Severity', height=500,
                           title='Number of casulties per year based on
Severity', template='plotly_dark').update_layout(
                    {'plot_bgcolor': 'rgba(0, 0, 0, 0)', 'paper_bgcolor':
'rgba(0, 0, 0, 0)'})

        return fig2

@uk_accidents_app.callback(
    Output('download-graph', 'data'),
    [Input('download-graph-button', 'n_clicks'),
     Input('graph-dropdown', 'value')],prevent_initial_call=True)

def download_graph(n_clicks, graph_type):

    if not n_clicks:
        raise PreventUpdate

    fig = update_graph(graph_type)
    return dcc.send_bytes(fig.to_image(format="PNG"),
filename=f"{graph_type}_graph.png")
```

```
#-------------------------------
# Bar Plots
#-------------------------------

# Create bar graphs such that there should be a dropdown:
# 1. Number of Accidents per Year
# 2. Number of Accidents per Year based on Severity
# 3. Number of Casualties per Year
# 4. Number of Casualties per Year based on Severity
# 5. Number of Vehicles involved in Accidents
# 6. Number of Accidents based on Settlement
# 7. Number of Accidents based on Settlement and Severity
# 7. Number of Accidents based on Police Attending Scene



accidents_by_vehicles =
accidents_vehicles_casualties.groupby('Number_of_Vehicles').count()['Accident
_Index'].reset_index()

accidents_by_settlement =
accidents_vehicles_casualties.groupby('Urban_or_Rural_Area').count()['Acciden
t_Index'].reset_index()
accidents_by_settlement_severity =
accidents_vehicles_casualties.groupby(['Urban_or_Rural_Area',
'Accident_Severity']).count()['Accident_Index'].reset_index()

accidents_by_police =
accidents_vehicles_casualties.groupby('Did_Police_Officer_Attend_Scene_of_Acc
ident').count()['Accident_Index'].reset_index()


bar_plots = html.Div(
    children=[

    html.Header('The Bar Plots for the given Data', style={'color':
'#F57241'}),

    html.Label('Select a Bar Plot', style={'color': '#F57241'}),

    html.H4('UK Accidents Bar Plots', style={'textAlign': 'center', 'color':
'#503D36', 'font-size': 40}),

        dcc.Dropdown(
            id='graph-bar-dropdown',
            options=[
                {'label': 'Accidents per Year', 'value': 'accidents'},
                {'label': 'Accidents per Year by Severity', 'value':
'accidents_severity'},
                {'label': 'Casualties per Year', 'value': 'casualties'},
                {'label': 'Casualties per Year by Severity', 'value':
'casualties_severity'},
                {'label': 'Vehicles involved in Accidents', 'value':
'vehicles'},
                {'label': 'Accidents based on Settlement', 'value':
```

```
'settlement'},
                {'label': 'Accidents based on Settlement and Severity',
'value': 'settlement_severity'},
                {'label': 'Accidents based on Police Attending Scene',
'value': 'police'}
            ],
            style={'width': '50%'},
            multi=False,
            value='accidents',
        ),
        dcc.Graph(id='display-bar-graph'),
        html.Br(),
    ]
)


@uk_accidents_app.callback(
    Output('display-bar-graph', 'figure'),
    [Input('graph-bar-dropdown', 'value')])

def update_graph(graph_type):

    if graph_type == 'accidents':

        fig1 = px.bar(year_accidents, x='Year', y='Accident_Index',
color='Accident_Index', height=500,
                            title='Number of accidents per year',
template='plotly_dark').update_layout(
                    {'plot_bgcolor': 'rgba(0, 0, 0, 0)', 'paper_bgcolor':
'rgba(0, 0, 0, 0)'})

        return fig1

    elif graph_type == 'accidents_severity':

        fig2 = px.bar(accidents_by_year_severity, x='Year',
y='Accident_Index', color='Accident_Severity', height=500,
                            title='Number of accidents per year based on
Severity', template='plotly_dark').update_layout(
                    {'plot_bgcolor': 'rgba(0, 0, 0, 0)', 'paper_bgcolor':
'rgba(0, 0, 0, 0)'})

        return fig2

    elif graph_type == 'casualties':

        fig3 = px.bar(casualties_by_year, x='Year', y='Number_of_Casualties',
color='Number_of_Casualties', height=500,
                            title='Number of casulties per year',
template='plotly_dark').update_layout(
                    {'plot_bgcolor': 'rgba(0, 0, 0, 0)', 'paper_bgcolor':
'rgba(0, 0, 0, 0)'})

        return fig3

    elif graph_type == 'casualties_severity':
```

```
        fig4 = px.bar(casualties_by_year_severity, x='Year',
y='Number_of_Casualties', color='Accident_Severity', height=500,
                              title='Number of casulties per year based on
Severity', template='plotly_dark').update_layout(
                    {'plot_bgcolor': 'rgba(0, 0, 0, 0)', 'paper_bgcolor':
'rgba(0, 0, 0, 0)'})

        return fig4

    elif graph_type == 'vehicles':

        fig5 = px.bar(accidents_by_vehicles, x='Number_of_Vehicles',
y='Accident_Index', color='Accident_Index', height=500,
                              title='Number of vehicles involved in accidents',
template='plotly_dark').update_layout(
                    {'plot_bgcolor': 'rgba(0, 0, 0, 0)', 'paper_bgcolor':
'rgba(0, 0, 0, 0)'})

        return fig5

    elif graph_type == 'settlement':

        fig6 = px.bar(accidents_by_settlement, x='Urban_or_Rural_Area',
y='Accident_Index', color='Accident_Index', height=500,
                              title='Number of accidents based on Settlement',
template='plotly_dark').update_layout(
                    {'plot_bgcolor': 'rgba(0, 0, 0, 0)', 'paper_bgcolor':
'rgba(0, 0, 0, 0)'})

        return fig6

    elif graph_type == 'settlement_severity':

        fig7 = px.bar(accidents_by_settlement_severity,
x='Urban_or_Rural_Area', y='Accident_Index', color='Accident_Severity',
height=500,
                              title='Number of accidents based on Settlement
and Severity', template='plotly_dark').update_layout(
                    {'plot_bgcolor': 'rgba(0, 0, 0, 0)', 'paper_bgcolor':
'rgba(0, 0, 0, 0)'})

        return fig7

    elif graph_type == 'police':

        fig8 = px.bar(accidents_by_police,
x='Did_Police_Officer_Attend_Scene_of_Accident', y='Accident_Index',
color='Accident_Index', height=500,
                              title='Number of accidents based on Police
Attending Scene', template='plotly_dark').update_layout(
                    {'plot_bgcolor': 'rgba(0, 0, 0, 0)', 'paper_bgcolor':
'rgba(0, 0, 0, 0)'})

        return fig8

#------------------------------
# Pie Plots
```

```
#--------------------------------

# Create pie graphs such that there should be a dropdown:
# 1. Number of Accidents by Road Type
# 2. Number of Accidents by Speed Limit
# 3. Number of Accidents by Light Conditions
# 4. Number of Accidents by Weather Conditions
# 5. Number of Accidents by Road Surface Conditions
# 6. Number of Accidents by Special Conditions
# 7. Number of Accidents by Carriageway Hazards

accidents_by_road_type =
accidents_vehicles_casualties.groupby('Road_Type').count()['Accident_Index'].
reset_index()

accidents_by_speed_limit =
accidents_vehicles_casualties.groupby('Speed_limit').count()['Accident_Index'
].reset_index()

accidents_by_light_conditions =
accidents_vehicles_casualties.groupby('Light_Conditions').count()['Accident_I
ndex'].reset_index()

accidents_by_weather_conditions =
accidents_vehicles_casualties.groupby('Weather_Conditions').count()['Accident
_Index'].reset_index()

accidents_by_road_surface =
accidents_vehicles_casualties.groupby('Road_Surface_Conditions').count()['Acc
ident_Index'].reset_index()

accidents_by_special_conditions =
accidents_vehicles_casualties.groupby('Special_Conditions_at_Site').count()['
Accident_Index'].reset_index()

accidents_by_carriageway_hazards =
accidents_vehicles_casualties.groupby('Carriageway_Hazards').count()['Acciden
t_Index'].reset_index()

pie_plots = html.Div(
    children=[

    html.Header('The Pie Plots for the given Data', style={'color':
'#F57241'}),

    html.Label('Select a Pie Plot', style={'color': '#F57241'}),

    html.H4('UK Accidents Pie Plots', style={'textAlign': 'center', 'color':
'#503D36', 'font-size': 40}),

    dcc.Dropdown(
        id='graph-pie-dropdown',
        options=[
            {'label': 'Accidents by Road Type', 'value': 'road_type'},
            {'label': 'Accidents by Speed Limit', 'value': 'speed_limit'},
            {'label': 'Accidents by Light Conditions', 'value':
'light_conditions'},
```

```
            {'label': 'Accidents by Weather Conditions', 'value':
'weather_conditions'},
            {'label': 'Accidents by Road Surface Conditions', 'value':
'road_surface'},
            {'label': 'Accidents by Special Conditions', 'value':
'special_conditions'},
            {'label': 'Accidents by Carriageway Hazards', 'value':
'carriageway_hazards'}
        ],
        style={'width': '50%'},
        multi=False,
        value='road_type',
    ),

    dcc.Graph(id='display-pie-graph'),

    html.Br(),
    ]
)

@uk_accidents_app.callback(
    Output('display-pie-graph', 'figure'),
    [Input('graph-pie-dropdown', 'value')])

def update_graph(graph_type):

    if graph_type == 'road_type':

        fig1 = px.pie(accidents_by_road_type, values='Accident_Index',
names='Road_Type',
                      title='Pie Chart of Road Type',
                      template='plotly_dark').update_layout(
                  {'plot_bgcolor': 'rgba(0, 0, 0, 0)', 'paper_bgcolor':
'rgba(0, 0, 0, 0)', 'showlegend': False})

        return fig1

    elif graph_type == 'speed_limit':

        fig2 = px.pie(accidents_by_speed_limit, values='Accident_Index',
names='Speed_limit',
                      title='Pie Chart of Speed Limit',
                      template='plotly_dark').update_layout(
                  {'plot_bgcolor': 'rgba(0, 0, 0, 0)', 'paper_bgcolor':
'rgba(0, 0, 0, 0)', 'showlegend': False})

        return fig2

    elif graph_type == 'light_conditions':

        fig3 = px.pie(accidents_by_light_conditions, values='Accident_Index',
names='Light_Conditions',
                      title='Pie Chart of Light Conditions',
                      template='plotly_dark').update_layout(
                  {'plot_bgcolor': 'rgba(0, 0, 0, 0)', 'paper_bgcolor':
'rgba(0, 0, 0, 0)', 'showlegend': False})
```

```
        return fig3

    elif graph_type == 'weather_conditions':

        fig4 = px.pie(accidents_by_weather_conditions,
values='Accident_Index', names='Weather_Conditions',
                      title='Pie Chart of Weather Conditions',
                      template='plotly_dark').update_layout(
                  {'plot_bgcolor': 'rgba(0, 0, 0, 0)', 'paper_bgcolor':
'rgba(0, 0, 0, 0)', 'showlegend': False})

        return fig4

    elif graph_type == 'road_surface':

        fig5 = px.pie(accidents_by_road_surface, values='Accident_Index',
names='Road_Surface_Conditions',
                      title='Pie Chart of Road Surface Conditions',
                      template='plotly_dark').update_layout(
                  {'plot_bgcolor': 'rgba(0, 0, 0, 0)', 'paper_bgcolor':
'rgba(0, 0, 0, 0)', 'showlegend': False})

        return fig5

    elif graph_type == 'special_conditions':

        fig6 = px.pie(accidents_by_special_conditions,
values='Accident_Index', names='Special_Conditions_at_Site',
                      title='Pie Chart of Special Conditions at Site',
                      template='plotly_dark').update_layout(
                  {'plot_bgcolor': 'rgba(0, 0, 0, 0)', 'paper_bgcolor':
'rgba(0, 0, 0, 0)', 'showlegend': False})

        return fig6

    elif graph_type == 'carriageway_hazards':

        fig7 = px.pie(accidents_by_carriageway_hazards,
values='Accident_Index', names='Carriageway_Hazards',
                      title='Pie Chart of Carriageway Hazards',
                      template='plotly_dark').update_layout(
                  {'plot_bgcolor': 'rgba(0, 0, 0, 0)', 'paper_bgcolor':
'rgba(0, 0, 0, 0)', 'showlegend': False})

        return fig7


#--------------------------------
# Count Plots
#--------------------------------

# Create count plots such that there should be a dropdown:
# 1. Number of Accidents by Road Type
# 2. Number of Accidents by Speed Limit
# 3. Number of Accidents by Light Conditions
# 4. Number of Accidents by Weather Conditions
# 5. Number of Accidents by Road Surface Conditions
```

```
# 6. Number of Accidents by Special Conditions
# 7. Number of Accidents by Carriageway Hazards

count_plots = html.Div(
    children=[

    html.Header('The Count Plots for the given Data', style={'color':
'#F57241'}),

    html.Label('Select a Count Plot', style={'color': '#F57241'}),

    html.H4('UK Accidents Count Plots', style={'textAlign': 'center',
'color': '#503D36', 'font-size': 40}),

    dcc.Dropdown(
        id='graph-count-dropdown',
        options=[
            {'label': 'Accidents by Road Type', 'value': 'road_type'},
            {'label': 'Accidents by Speed Limit', 'value': 'speed_limit'},
            {'label': 'Accidents by Light Conditions', 'value':
'light_conditions'},
            {'label': 'Accidents by Weather Conditions', 'value':
'weather_conditions'},
            {'label': 'Accidents by Road Surface Conditions', 'value':
'road_surface'},
            {'label': 'Accidents by Special Conditions', 'value':
'special_conditions'},
            {'label': 'Accidents by Carriageway Hazards', 'value':
'carriageway_hazards'}
        ],
        style={'width': '50%'},
        multi=False,
        value='road_type',
    ),

    dcc.Graph(id='display-count-graph'),

    html.Br(),

    ]
)

@uk_accidents_app.callback(
    Output('display-count-graph', 'figure'),
    [Input('graph-count-dropdown', 'value')])

def update_graph(graph_type):

    if graph_type == 'road_type':

        fig1 = px.bar(accidents_by_road_type, x='Road_Type',
y='Accident_Index', color='Accident_Index', height=500,
                              title='Number of accidents based on Road Type',
template='plotly_dark').update_layout(
                    {'plot_bgcolor': 'rgba(0, 0, 0, 0)', 'paper_bgcolor':
'rgba(0,0,0,0)'})
```

```
        return fig1

    elif graph_type == 'speed_limit':

        fig2 = px.bar(accidents_by_speed_limit, x='Speed_limit',
y='Accident_Index', color='Accident_Index', height=500,
                              title='Number of accidents based on Speed Limit',
template='plotly_dark').update_layout(
                      {'plot_bgcolor': 'rgba(0, 0, 0, 0)', 'paper_bgcolor':
'rgba(0,0,0,0)'})

        return fig2

    elif graph_type == 'light_conditions':

        fig3 = px.bar(accidents_by_light_conditions, x='Light_Conditions',
y='Accident_Index', color='Accident_Index', height=500,
                              title='Number of accidents based on Light
Conditions', template='plotly_dark').update_layout(
                      {'plot_bgcolor': 'rgba(0, 0, 0, 0)', 'paper_bgcolor':
'rgba(0,0,0,0)'})

        return fig3

    elif graph_type == 'weather_conditions':

        fig4 = px.bar(accidents_by_weather_conditions,
x='Weather_Conditions', y='Accident_Index', color='Accident_Index',
height=500,
                              title='Number of accidents based on Weather
Conditions', template='plotly_dark').update_layout(
                      {'plot_bgcolor': 'rgba(0, 0, 0, 0)', 'paper_bgcolor':
'rgba(0,0,0,0)'})

        return fig4

    elif graph_type == 'road_surface':

        fig5 = px.bar(accidents_by_road_surface, x='Road_Surface_Conditions',
y='Accident_Index', color='Accident_Index', height=500,
                              title='Number of accidents based on Road Surface
Conditions', template='plotly_dark').update_layout(
                      {'plot_bgcolor': 'rgba(0, 0, 0, 0)', 'paper_bgcolor':
'rgba(0,0,0,0)'})

        return fig5

    elif graph_type == 'special_conditions':

        fig6 = px.bar(accidents_by_special_conditions,
x='Special_Conditions_at_Site', y='Accident_Index', color='Accident_Index',
height=500,
                              title='Number of accidents based on Special
Conditions at Site', template='plotly_dark').update_layout(
                      {'plot_bgcolor': 'rgba(0, 0, 0, 0)', 'paper_bgcolor':
'rgba(0,0,0,0)'})
```

```
        return fig6

    elif graph_type == 'carriageway_hazards':

        fig7 = px.bar(accidents_by_carriageway_hazards,
x='Carriageway_Hazards', y='Accident_Index', color='Accident_Index',
height=500,
                          title='Number of accidents based on Carriageway
Hazards', template='plotly_dark').update_layout(
                    {'plot_bgcolor': 'rgba(0, 0, 0, 0)', 'paper_bgcolor':
'rgba(0,0,0,0)'})

        return fig7


#--------------------------------
# Histogram Plots
#--------------------------------

# Create histogram plots such that there should be a dropdown:
# 1. Number of Vehicles
# 2. Number of Casualties
# 3. Speed Limit
# 4. Age of Vehicle
# 5. Engine Capacity
# 6. Age of Driver
# 7. Number of Casualties by Age of Driver

histogram_plots = html.Div(
    children=[

    html.Header('The Histogram Plots for the given Data', style={'color':
'#F57241'}),

    html.Label('Select a Histogram Plot', style={'color': '#F57241'}),

    html.H4('UK Accidents Histogram Plots', style={'textAlign': 'center',
'color': '#503D36', 'font-size': 40}),

    dcc.Dropdown(
        id='graph-hist-dropdown',
        options=[
            {'label': 'Number of Vehicles', 'value': 'vehicles'},
            {'label': 'Number of Casualties', 'value': 'casualties'},
            {'label': 'Speed Limit', 'value': 'speed_limit'},
            {'label': 'Age of Vehicle', 'value': 'age_vehicle'},
            {'label': 'Engine Capacity', 'value': 'engine_capacity'},
            {'label': 'Age of Driver', 'value': 'age_driver'},
            {'label': 'Number of Casualties by Age of Driver', 'value':
'casualties_age_driver'}
        ],
        style={'width': '50%'},
        multi=False,
        value='vehicles',
    ),

    dcc.Graph(id='display-hist-graph'),
```

```
    html.Br(),

    ]
)

@uk_accidents_app.callback(
    Output('display-hist-graph', 'figure'),
    [Input('graph-hist-dropdown', 'value')])

def update_graph(graph_type):

    if graph_type == 'vehicles':

        fig1 = px.histogram(accidents_vehicles_casualties,
x='Number_of_Vehicles', color='Number_of_Vehicles',
                            title='Histogram of Number of Vehicles',
                            template='plotly_dark', nbins=20).update_layout(
                    {'plot_bgcolor': 'rgba(0, 0, 0, 0)', 'paper_bgcolor':
'rgba(0, 0, 0, 0)'})

        return fig1

    elif graph_type == 'casualties':

        fig2 = px.histogram(accidents_vehicles_casualties,
x='Number_of_Casualties', color='Number_of_Casualties',
                            title='Histogram of Number of Casualties',
                            template='plotly_dark', nbins=20).update_layout(
                    {'plot_bgcolor': 'rgba(0, 0, 0, 0)', 'paper_bgcolor':
'rgba(0, 0, 0, 0)'})

        return fig2

    elif graph_type == 'speed_limit':

        fig3 = px.histogram(accidents_vehicles_casualties, x='Speed_limit',
color='Speed_limit',
                            title='Histogram of Speed Limit',
                            template='plotly_dark', nbins=20).update_layout(
                    {'plot_bgcolor': 'rgba(0, 0, 0, 0)', 'paper_bgcolor':
'rgba(0, 0, 0, 0)'})

        return fig3

    elif graph_type == 'age_vehicle':

        fig4 = px.histogram(accidents_vehicles_casualties,
x='Age_of_Vehicle', color='Age_of_Vehicle',
                            title='Histogram of Age of Vehicle',
                            template='plotly_dark', nbins=20).update_layout(
                    {'plot_bgcolor': 'rgba(0, 0, 0, 0)', 'paper_bgcolor':
'rgba(0, 0, 0, 0)'})

        return fig4

    elif graph_type == 'engine_capacity':
```

```
        fig5 = px.histogram(accidents_vehicles_casualties,
x='Engine_Capacity_(CC)', color='Engine_Capacity_(CC)',
                            title='Histogram of Engine Capacity',
                            template='plotly_dark', nbins=20).update_layout(
                    {'plot_bgcolor': 'rgba(0, 0, 0, 0)', 'paper_bgcolor':
'rgba(0, 0, 0, 0)', 'showlegend': False})

        return fig5

    elif graph_type == 'age_driver':

        fig6 = px.histogram(accidents_vehicles_casualties, x='Age_of_Driver',
color='Age_of_Driver',
                            title='Histogram of Age of Driver',
                            template='plotly_dark', nbins=20).update_layout(
                    {'plot_bgcolor': 'rgba(0, 0, 0, 0)', 'paper_bgcolor':
'rgba(0, 0, 0, 0)', 'showlegend': False})

        return fig6

    elif graph_type == 'casualties_age_driver':

        fig7 = px.histogram(accidents_vehicles_casualties, x='Age_of_Driver',
color='Number_of_Casualties',
                            title='Histogram of Number of Casualties by Age
of Driver',
                            template='plotly_dark', nbins=20).update_layout(
                    {'plot_bgcolor': 'rgba(0, 0, 0, 0)', 'paper_bgcolor':
'rgba(0, 0, 0, 0)', 'showlegend': False})

        return fig7


#-------------------------------
# Miscellaneous Plots
#-------------------------------

# Create a check list to select the type of violin plot to be displayed:
# 1. Area plot of Number of Accidents by year
# 1. Area plot of Number of Casualties by year
# 2. Area plot of Number of Vehicles affected by year

accidents_by_year_severity = accidents_vehicles_casualties.groupby(['Year',
'Accident_Severity']).count()['Accident_Index'].reset_index()
casualties_by_year_severity = accidents_vehicles_casualties.groupby(['Year',
'Accident_Severity']).sum()['Number_of_Casualties'].reset_index()
accidents_by_vehicles_severity =
accidents_vehicles_casualties.groupby(['Number_of_Vehicles',
'Accident_Severity']).count()['Accident_Index'].reset_index()

misc_plots = html.Div(
    children=[

    html.Header('The respective Plots for the given Data', style={'color':
'#F57241'}),
```

```
    html.Label('Select the type of Miscellaneous Plots to be displayed:'),

    dcc.Checklist(
        id='misc_plots',
        options=[
            {'label': 'Area Plot - Number of Accidents per Year', 'value':
'area_accidents_by_year'},
            {'label': 'Area Plot - Number of Casualties per Year', 'value':
'area_casualties_by_year'},
            {'label': 'Area Plot - Number of Vehicles affected per Year',
'value': 'area_vehicles_by_year'},
        ],
        value=[],
        labelStyle={'display': 'inline-block', 'margin-right': '20px'}
    ),

    html.Div(id='misc_plots_output'),

    ]
)

@uk_accidents_app.callback(
    Output(component_id='misc_plots_output', component_property='children'),
    [Input(component_id='misc_plots', component_property='value')])

def update_misc_plots(plot_types):
    graphs = []

    if 'area_accidents_by_year' in plot_types:
        graphs.append(dcc.Graph(
            id='area_accidents_by_year',
            figure=px.area(accidents_by_year_severity, x='Year',
y='Accident_Index', color='Accident_Severity', height=500,
                           title='Area plot of Number of Accidents by year',
template='plotly_dark').update_layout(
                {'plot_bgcolor': 'rgba(0, 0, 0, 0)', 'paper_bgcolor':
'rgba(0, 0, 0, 0)'})
            ))

    if 'area_casualties_by_year' in plot_types:
        graphs.append(dcc.Graph(
            id='area_casualties_by_year',
            figure=px.area(casualties_by_year_severity, x='Year',
y='Number_of_Casualties', color='Accident_Severity', height=500,
                           title='Area plot of Number of Casualties by year',
template='plotly_dark').update_layout(
                {'plot_bgcolor': 'rgba(0, 0, 0, 0)', 'paper_bgcolor':
'rgba(0, 0, 0, 0)'})
            ))

    if 'area_vehicles_by_year' in plot_types:
        graphs.append(dcc.Graph(
            id='area_vehicles_by_year',
            figure=px.area(accidents_by_vehicles_severity,
x='Number_of_Vehicles', y='Accident_Index', color='Accident_Severity',
height=500,
                           title='Area plot of Number of Vehicles affected by
```

```
year', template='plotly_dark').update_layout(
                {'plot_bgcolor': 'rgba(0, 0, 0, 0)', 'paper_bgcolor':
'rgba(0, 0, 0, 0)'})
            ))

    return graphs



#--------------------------------
# Geospatial Plots
#--------------------------------

# Create a radio button to select the type of geospatial plot to be
displayed, but not multiple plots at the same time:
# 1. Density Heatmap of accidents in the UK in year slider
# 2. Bubble Map of accidents in the UK in year slider
# 3. 3D Scatter Plot of accidents in the UK in year slider
# 4. Cluster Map of accidents in the UK in year slider
# 5. Line Map of accidents where accidents occurred in the UK in year slider


geo_plots = html.Div(
    children=[

    html.Header('The Geospatial Plots for the given Data', style={'color':
'#F57241'}),

    html.Label('Select the type of Geospatial Plot to be displayed:'),

    html.H4('UK Accidents Geospatial Plots', style={'textAlign': 'center',
'color': '#503D36', 'font-size': 40}),

    dcc.RadioItems(
        id='geospatial_plots',
        options=[
            {'label': 'Density Heatmap', 'value': 'density_heatmap'},
            {'label': 'Bubble Map', 'value': 'bubble_map'},
            {'label': '3D Scatter Plot', 'value': '3d_scatter_plot'},
            {'label': 'Cluster Map', 'value': 'cluster_map'},
            {'label': 'Line Map', 'value': 'line_map'},
        ],
        value=[],
        labelStyle={'display': 'inline-block', 'margin-right': '20px'}
    ),

    html.Br(),

    dcc.Slider(
        id='year_slider',
        min=2005,
        max=2014,
        value=2005,
        marks={str(year): str(year) for year in range(2005, 2015)},
        step=None
    ),
```

```
    html.Div(id='geospatial_plots_output'),


    dcc.Textarea(
        id='geospatial_comments',
        placeholder='Enter your comments or descriptions here...',
        style={'width': '100%', 'height': 100},
    ),

    html.Button('Submit Comment', id='submit_comment_button', n_clicks=0),

    html.Div(id='comments_output'),
    ]
)

@uk_accidents_app.callback(
    Output(component_id='geospatial_plots_output',
component_property='children'),
    [Input(component_id='geospatial_plots', component_property='value'),
     Input(component_id='year_slider', component_property='value')])

def update_slider(plot_type, year):

    filtered_data =
accidents_vehicles_casualties[accidents_vehicles_casualties['Year'] == year]
    plot = render_plot(plot_type, filtered_data)

    if plot:
        return dcc.Graph(figure=plot)

    return "Select plot type and year to display."

def render_plot(plot_type, filtered_data):

    if plot_type == 'density_heatmap':
        return px.density_mapbox(filtered_data, lat='Latitude',
lon='Longitude', radius=10, zoom=5, height=800,
                                 hover_data=['Accident_Index'],
                                 hover_name='Accident_Index',
                                 title='Density Heatmap of Accidents in the
UK in {}'.format(filtered_data['Year'].unique()[0]),
                                 template='plotly_dark').update_layout(
            mapbox_style="open-street-map", mapbox_center_lon=0,
mapbox_center_lat=52)

    elif plot_type == 'bubble_map':
        return px.scatter_mapbox(filtered_data, lat='Latitude',
lon='Longitude', color='Accident_Severity', size='Number_of_Casualties',
                                 hover_data=['Accident_Index'],
                                 hover_name='Accident_Index',
                                 zoom=5, height=800, title='Bubble Map of
Accidents in the UK in {}'.format(filtered_data['Year'].unique()[0]),
                                 template='plotly_dark').update_layout(
            mapbox_style="open-street-map", mapbox_center_lon=0,
mapbox_center_lat=52)
```

```
    elif plot_type == '3d_scatter_plot':
        return px.scatter_3d(filtered_data, x='Longitude', y='Latitude',
z='Number_of_Casualties', color='Accident_Severity',
                             hover_data=['Accident_Index'],
                             hover_name='Accident_Index',
                             height=800, title='3D Scatter Plot of Accidents
in the UK in {}'.format(filtered_data['Year'].unique()[0]),
                             template='plotly_dark')

    elif plot_type == 'cluster_map':
        return px.scatter_mapbox(filtered_data, lat='Latitude',
lon='Longitude', color='Accident_Severity', zoom=5, height=800,
                                 hover_data=['Accident_Index'],
                                 hover_name='Accident_Index',
                                 title='Cluster Map of Accidents in the UK in
{}'.format(filtered_data['Year'].unique()[0]),
                                 template='plotly_dark').update_layout(
            mapbox_style="open-street-map", mapbox_center_lon=0,
mapbox_center_lat=52, mapbox_zoom=5)

    elif plot_type == 'line_map':
        return px.line_mapbox(filtered_data, lat='Latitude', lon='Longitude',
color='Accident_Severity', zoom=5, height=800,
                              hover_data=['Accident_Index'],
                              hover_name='Accident_Index',
                              title='Line Map of Accidents in the UK in
{}'.format(filtered_data['Year'].unique()[0]),
                              template='plotly_dark').update_layout(
                mapbox_style="open-street-map", mapbox_center_lon=0,
mapbox_center_lat=52, mapbox_zoom=5)

    else:
        return None

@uk_accidents_app.callback(
    Output(component_id='comments_output', component_property='children'),
    [Input(component_id='submit_comment_button',
component_property='n_clicks')],
    [State(component_id='geospatial_comments', component_property='value')])

def update_comments(n_clicks, comments):

    if n_clicks > 0:
        return html.Div([
            html.Hr(),
            html.H5('Comments:'),
            html.Br(),
            html.Div(comments)
        ])

    return None

uk_accidents_app.run_server(port = 8050, host='0.0.0.0')
```

References:

[1] https://www.kaggle.com/datasets/benoit72/uk-accidents-10-years-history-with-many-variables

[2] https://pandas.pydata.org/docs/user_guide/index.html

[2] https://matplotlib.org/stable/plot_types/index

[3] https://seaborn.pydata.org/api.html

[4] https://plotly.com/python/plotly-fundamentals/

[5] https://community.plotly.com/c/plotly-python/5

[6] https://plotly.com/examples/geospatial/

[7] https://plotly.com/python/maps/

[8] https://console.cloud.google.com/