Road Accident Analysis

Project Overview:

The"Road Accident Analysis" project is a data validation and cross-checking project in Python, previously implemented using tools like Tableau, Power BI, Excel, and SQL. The primary objective of this project is to analyze and compare the number of road accidents and casualties that occurred between the years 2021 and 2022 in England. The analysis will focus on key aspects such as the number of casualties by year, accident severity, road type, area, light condition, and vehicle type involved in the accidents.

Goals :

Analyze road accidents and casualties between 2021 and 2022.

Investigate accident severity, road type, area, light condition, and vehicle type's impact on casualties.

In [1]:
```
#importing libraries
import pandas as pd
import plotly.graph_objects as go
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
# Load the data into a DataFrame named 'road_accident'
road_accident = pd.read_csv("road_accident.csv")
road_accident.head(5)
```

Out[2]:

	accident_index	accident_date	day_of_week	junction_control	junction_detail	accident_severity	light_conditions	local_authority	carriageway_hazards	number_of_casualties	number_of_vehicles	police_force	road_surface_cond
0	BS0000001	01-01-2021	Thursday	Give way or uncontrolled	T or staggered junction	Serious	Daylight	Kensington and Chelsea	None	1	2	Metropolitan Police	
1	BS0000002	05-01-2021	Monday	Give way or uncontrolled	Crossroads	Serious	Daylight	Kensington and Chelsea	None	11	2	Metropolitan Police	Wet or
2	BS0000003	04-01-2021	Sunday	Give way or uncontrolled	T or staggered junction	Slight	Daylight	Kensington and Chelsea	None	1	2	Metropolitan Police	
3	BS0000004	05-01-2021	Monday	Auto traffic signal	T or staggered junction	Serious	Daylight	Kensington and Chelsea	None	1	2	Metropolitan Police	Frost
4	BS0000005	06-01-2021	Tuesday	Auto traffic signal	Crossroads	Serious	Darkness - lights lit	Kensington and Chelsea	None	1	2	Metropolitan Police	

Data Preprocessing

In [3]:
```
#number of rows and columns
road_accident.shape
```

Out[3]:
```
(307973, 19)
```

In [4]:
```
#checking for null values
road_accident.isnull().sum()
```

Out[4]:
```
accident_index              0
accident_date               0
day_of_week                 0
junction_control            0
junction_detail             0
accident_severity           0
light_conditions            0
local_authority             0
carriageway_hazards         3
number_of_casualties        0
number_of_vehicles          0
police_force                0
road_surface_conditions     0
road_type                   0
speed_limit                 0
time                       17
urban_or_rural_area         0
weather_conditions          0
vehicle_type                0
dtype: int64
```

In [5]:
```
#checking for duplicate values
road_accident.duplicated().sum()
```

Out[5]:
```
0
```

In [6]:
```
#checking data types
road_accident.dtypes
```

Out[6]:
```
accident_index             object
accident_date              object
day_of_week                object
junction_control           object
junction_detail            object
accident_severity          object
light_conditions           object
local_authority            object
carriageway_hazards        object
number_of_casualties        int64
number_of_vehicles          int64
police_force               object
road_surface_conditions    object
road_type                  object
speed_limit                 int64
time                       object
urban_or_rural_area        object
weather_conditions         object
vehicle_type               object
dtype: object
```

In [7]:
```
# changing data type of column [accident_date] to datetime for further analysis
road_accident['accident_date'] = pd.to_datetime(road_accident['accident_date'], dayfirst=True)
```

In [8]:
```
#checking data types
road_accident.dtypes
```

Out[8]:
```
accident_index                     object
accident_date              datetime64[ns]
day_of_week                        object
junction_control                   object
junction_detail                    object
accident_severity                  object
light_conditions                   object
local_authority                    object
carriageway_hazards                object
number_of_casualties                int64
number_of_vehicles                  int64
police_force                       object
road_surface_conditions            object
road_type                          object
speed_limit                         int64
time                               object
urban_or_rural_area                object
weather_conditions                 object
vehicle_type                       object
dtype: object
```

In [9]:
```
#checking the descriptive statistics
road_accident.describe()
```

Out[9]:

	number_of_casualties	number_of_vehicles	speed_limit
count	307973.000000	307973.000000	307973.000000
mean	1.356882	1.829063	38.866037
std	0.815857	0.710477	14.032933
min	1.000000	1.000000	10.000000
25%	1.000000	1.000000	30.000000
50%	1.000000	2.000000	30.000000
75%	1.000000	2.000000	50.000000
max	48.000000	32.000000	70.000000

In [10]:
```
road_accident.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 307973 entries, 0 to 307972
Data columns (total 19 columns):
 #   Column                   Non-Null Count   Dtype
---  ------                   --------------   -----
 0   accident_index           307973 non-null  object
 1   accident_date            307973 non-null  datetime64[ns]
 2   day_of_week              307973 non-null  object
 3   junction_control         307973 non-null  object
 4   junction_detail          307973 non-null  object
 5   accident_severity        307973 non-null  object
 6   light_conditions         307973 non-null  object
 7   local_authority          307973 non-null  object
 8   carriageway_hazards      307970 non-null  object
 9   number_of_casualties     307973 non-null  int64
 10  number_of_vehicles       307973 non-null  int64
 11  police_force             307973 non-null  object
 12  road_surface_conditions  307973 non-null  object
 13  road_type                307973 non-null  object
 14  speed_limit              307973 non-null  int64
 15  time                     307956 non-null  object
 16  urban_or_rural_area      307973 non-null  object
 17  weather_conditions       307973 non-null  object
 18  vehicle_type             307973 non-null  object
dtypes: datetime64[ns](1), int64(3), object(15)
memory usage: 44.6+ MB
```

Data Analysis and Visualization

In [11]:
```
# Number of accident by years
def total_accident_by_year(year):
    filtered_data = road_accident[(road_accident['accident_date'].dt.year == year)]
    total_casualties = filtered_data['accident_index'].nunique()
    return f"{year} Accidents : {total_casualties}"
```

In [12]:
```
print(total_accident_by_year(year=2022))
```

```
2022 Accidents : 144419
```

In [13]:
```
# Total number of Accident
total_accident_count = road_accident['accident_index'].nunique()
print("Total number of Accidents :",total_accident_count)
```

```
Total number of Accidents : 307973
```

In [14]:
```
# Number of Casualties by years
def total_casualties_by_year(year):
    filtered_data = road_accident[(road_accident['accident_date'].dt.year == year)]
    total_casualties = filtered_data['number_of_casualties'].sum()
    return f"{year} Casualties : {total_casualties}"
```

In [15]:
```
print(total_casualties_by_year(year=2022))
```

```
2022 Casualties : 195737
```

In [16]:
```
# Total number of Casualties
total_casualties = road_accident['number_of_casualties'].sum()
print("Total number of Casualties :",total_casualties)
```

```
Total number of Casualties : 417883
```

In [17]:
```
# Casualties by year and accident severity
def total_casualties_by_year_and_severity(year, severity):
    filtered_data = road_accident[(road_accident['accident_date'].dt.year == year) & (road_accident['accident_severity'] == severity)]
    total_casualties = filtered_data['number_of_casualties'].sum()
    return f"{year} {severity} Casualties : {total_casualties}"
```

In [18]:
```
print(total_casualties_by_year_and_severity(year=2022, severity="Fatal"))
```

```
2022 Fatal Casualties : 2855
```

In [19]:
```
# Function to calculate the total casualties for a specific severity
def total_casualties_by_severity(severity):
    filtered_data = road_accident[road_accident['accident_severity'] == severity]
    return f"{severity} Casualties : {filtered_data['number_of_casualties'].sum()}"
```

In [20]:
```
print(total_casualties_by_severity(severity = "Fatal"))
```

```
Fatal Casualties : 7135
```

In [21]:
```
# Function to calculate the total casualties for a specific year
def total_casualties_by_year(year):
    filtered_data = road_accident[road_accident['accident_date'].dt.year == year]
    return f"{year} Casualties: {filtered_data['number_of_casualties'].sum()} "
```

In [22]:
```
print(total_casualties_by_year(year=2022))
```

```
2022 Casualties: 195737
```

In [23]:
```
# Casualties by month
# Extract month and month name from the 'accident_date' column
road_accident['month'] = road_accident['accident_date'].dt.month
road_accident['month_name'] = road_accident['accident_date'].dt.strftime('%B')

# Filter data for each year and calculate the casualties by each month
year_2021_data = road_accident[road_accident['accident_date'].dt.year == 2021].groupby(['month', 'month_name'])['number_of_casualties'].sum()
year_2022_data = road_accident[road_accident['accident_date'].dt.year == 2022].groupby(['month', 'month_name'])['number_of_casualties'].sum()
```

In [24]:
```
print("Casualties in 2022:", "\n",year_2022_data)
```

```
Casualties in 2022:
 month  month_name
1      January       13163
2      February      14804
3      March         16575
4      April         15767
5      May           16775
6      June          17230
7      July          17201
8      August        16796
9      September     17500
10     October       18287
11     November      18439
12     December      13200
Name: number_of_casualties, dtype: int64
```

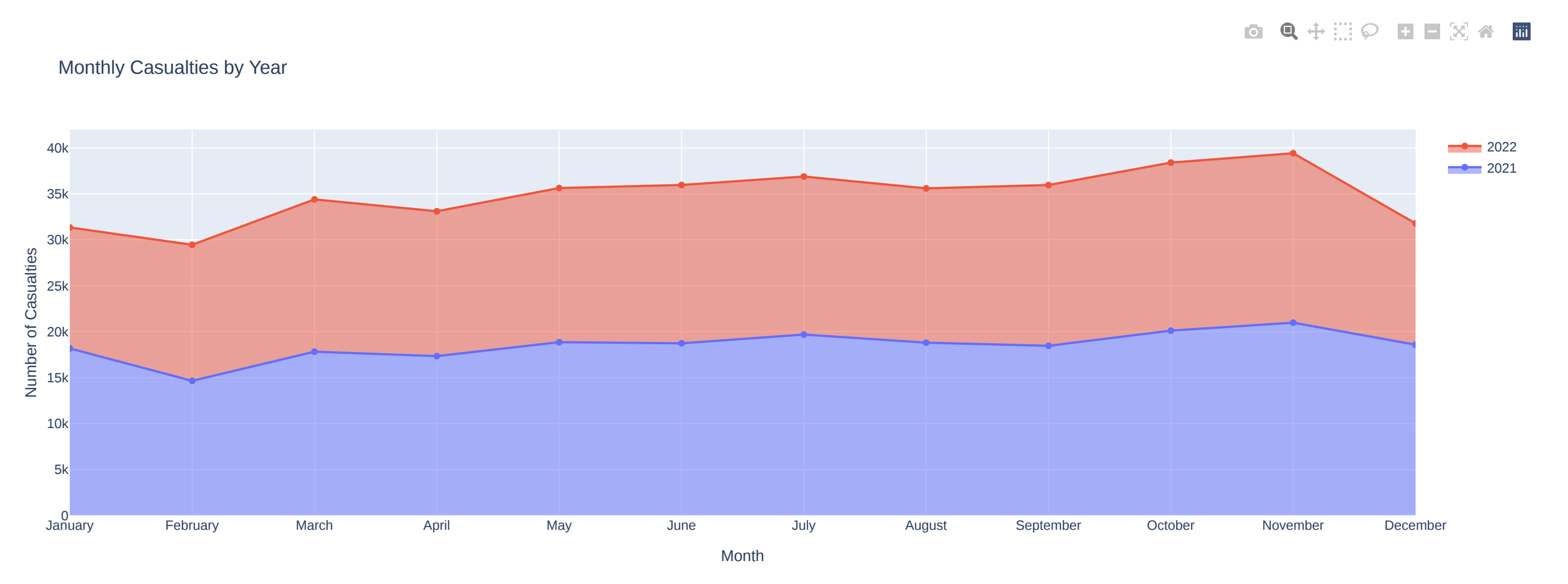
In [25]:
```
print("Casualties in 2021:", "\n",year_2021_data)
```

```
Casualties in 2021:
 month  month_name
1      January       18173
2      February      14648
3      March         17815
4      April         17335
5      May           18852
6      June          18728
7      July          19682
8      August        18797
9      September     18456
10     October       20109
11     November      20975
12     December      18576
Name: number_of_casualties, dtype: int64
```

In [26]:
```
#monthly casualties area chart
road_accident['accident_date'] = pd.to_datetime(road_accident['accident_date'], dayfirst=True)
road_accident['month_name'] = road_accident['accident_date'].dt.strftime('%B')
road_accident['year'] = road_accident['accident_date'].dt.year
casualties_by_month_year = road_accident.groupby(['year', 'month_name'])['number_of_casualties'].sum().reset_index()
fig = go.Figure()
years = casualties_by_month_year['year'].unique()
for year in years:
    data_by_year = casualties_by_month_year[casualties_by_month_year['year'] == year]
    fig.add_trace(go.Scatter(x=data_by_year['month_name'], y=data_by_year['number_of_casualties'],
                             mode='lines+markers', stackgroup='one', name=str(year)))


fig.update_layout(title='Monthly Casualties by Year',
                  xaxis_title='Month',
                  yaxis_title='Number of Casualties',
                  xaxis=dict(type='category', categoryorder='array', categoryarray=[
                      'January', 'February', 'March', 'April', 'May', 'June',
                      'July', 'August', 'September', 'October', 'November', 'December']),
                  yaxis=dict(title_standoff=0))


fig.show()
```



In [27]:
```
# casualties by road type for a specific year
def casualties_by_road_type(year):
    road_accident_year = road_accident[road_accident['accident_date'].dt.year == year]
    result = road_accident_year.groupby('road_type')['number_of_casualties'].sum().reset_index()
    result.rename(columns={'number_of_casualties': 'Casualties'}, inplace=True)
    result.sort_values(by='Casualties', ascending=True, inplace=True)
    return result
```

In [28]:
```
casualties_by_road_type(year=2022)
```

Out[28]:

	road_type	Casualties
4	Slip road	2990
1	One way street	3499
2	Roundabout	12683
0	Dual carriageway	31912
3	Single carriageway	144653

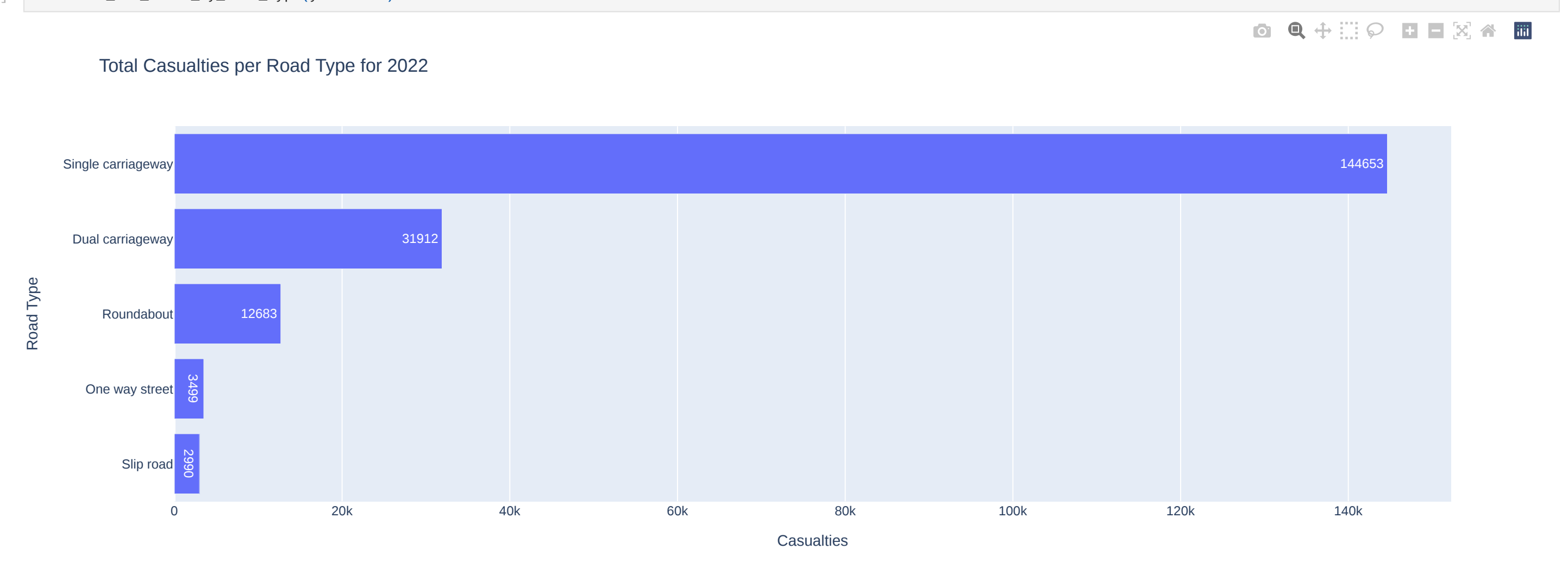
In [29]:
```
# clustered bar chart for casualties by road type for a specific year
def clustered_bar_chart_by_road_type(year):
    road_accident_year = road_accident[road_accident['accident_date'].dt.year == year]
    result = road_accident_year.groupby('road_type')['number_of_casualties'].sum().reset_index()
    result.rename(columns={'number_of_casualties': 'Casualties'}, inplace=True)
    result.sort_values(by='Casualties', ascending=True, inplace=True)

    fig = go.Figure()
    fig.add_trace(go.Bar(
        y=result['road_type'],
        x=result['Casualties'],
        orientation='h',
        text=result['Casualties'],
        textposition='inside',
        name=str(year),
    ))

    fig.update_layout(
        title=f'Total Casualties per Road Type for {year}',
        xaxis_title='Casualties',
        yaxis_title='Road Type',
    )

    fig.show()
```

In [30]:
```
clustered_bar_chart_by_road_type(year=2022)
```



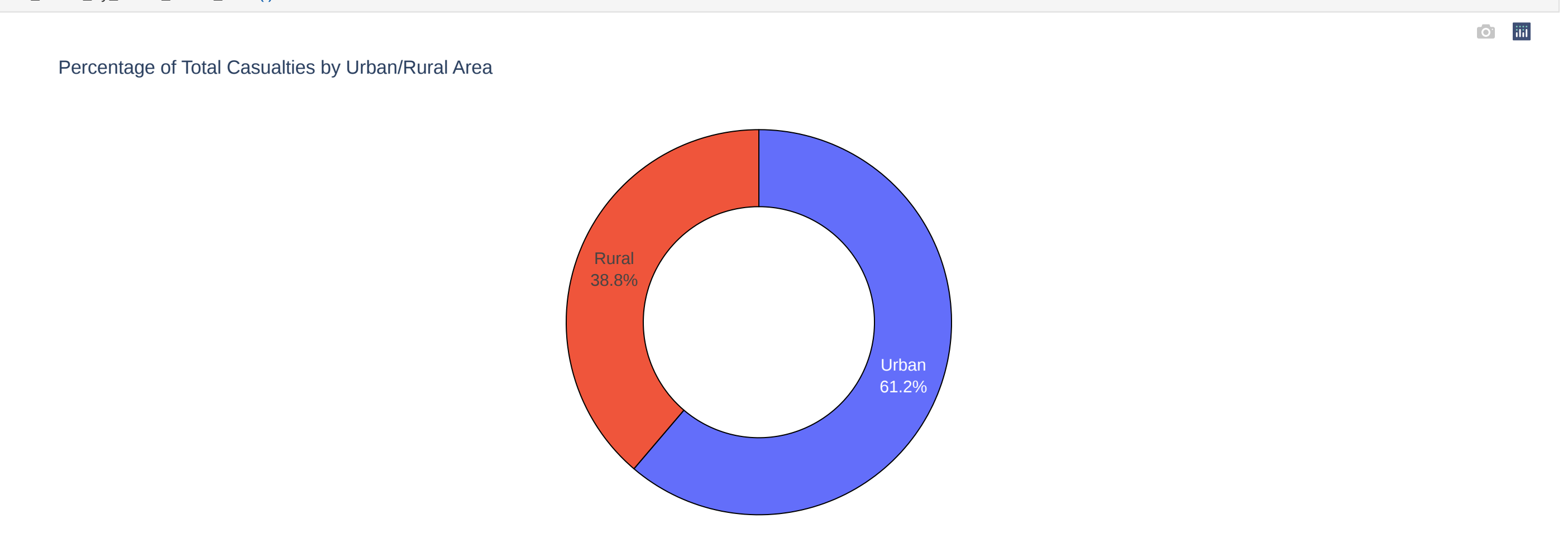
In [31]:
```
#calculate the total casualties by urban/rural area and plot a donut chart
def donut_chart_by_urban_rural_area():
    total_casualties = road_accident['number_of_casualties'].sum()
    result = road_accident.groupby('urban_or_rural_area')['number_of_casualties'].sum().reset_index()
    result['% Total'] = (result['number_of_casualties'] / total_casualties) * 100
    result['% Total'] = result['% Total'].round(2)
    result.drop(columns=['number_of_casualties'], inplace=True)

    fig = go.Figure()
    fig.add_trace(go.Pie(
        labels=result['urban_or_rural_area'],
        values=result['% Total'],
        hole=0.6,
        hoverinfo='label+percent',
        textinfo='label+percent',
        textfont_size=15,
        marker=dict(line=dict(color='#000000', width=1))
    ))

    fig.update_layout(
        title='Percentage of Total Casualties by Urban/Rural Area',
        showlegend=False,
    )

    fig.show()
```

In [32]:
```
donut_chart_by_urban_rural_area()
```



In [33]:
```
#calculate the total casualties by vehicle type
def casualties_by_vehicle_type():
    vehicle_type_mapping = {
        'Agricultural vehicle': 'Agriculture',
        'Car': 'Cars',
        'Taxi/Private hire car': 'Cars',
        'Motorcycle 125cc and under': 'Bike',
        'Motorcycle 50cc and under': 'Bike',
        'Motorcycle over 125cc and up to 500cc': 'Bike',
        'Motorcycle over 500cc': 'Bike',
        'Pedal cycle': 'Bike',
        'Bus or coach (17 or more pass seats)': 'Bus',
        'Minibus (8 - 16 passenger seats)': 'Bus',
        'Goods 7.5 tonnes mgw and over': 'Goods',
        'Goods over 3.5t. and under 7.5t': 'Goods',
        'Van / Goods 3.5 tonnes mgw or under': 'Goods',
    }

    road_accident['vehicle_group'] = road_accident['vehicle_type'].map(vehicle_type_mapping)
    result = road_accident.groupby('vehicle_group')['number_of_casualties'].sum().sort_values(ascending=False).reset_index()
    result.rename(columns={'number_of_casualties': 'Casualties'}, inplace=True)
    return result
```

In [34]:
```
casualties_by_vehicle_type()
```

Out[34]:

	vehicle_group	Casualties
0	Cars	333485
1	Bike	33764
2	Goods	33472
3	Bus	12798
4	Agriculture	1032

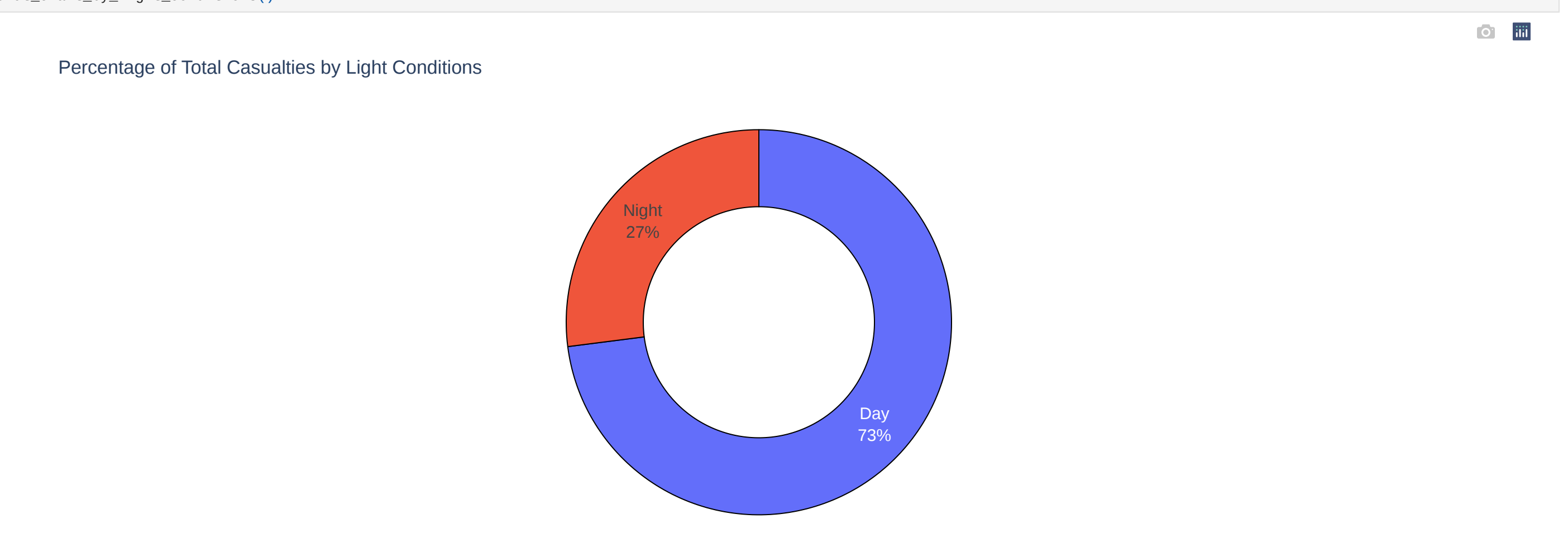
In [35]:
```
#calculate the total casualties by light conditions and plot a donut chart
def donut_chart_by_light_conditions():
    total_casualties = road_accident['number_of_casualties'].sum()
    light_conditions_mapping = {
        'Daylight': 'Day',
        'Darkness - lighting unknown': 'Night',
        'Darkness - lights lit': 'Night',
        'Darkness - lights unlit': 'Night',
        'Darkness - no lighting': 'Night',
    }
    road_accident['Light_Conditions'] = road_accident['light_conditions'].map(light_conditions_mapping)
    result = road_accident.groupby('Light_Conditions')['number_of_casualties'].sum().reset_index()
    result['Total % Casualties'] = (result['number_of_casualties'] / total_casualties) * 100
    result['Total % Casualties'] = result['Total % Casualties'].round(2)
    result.drop(columns=['number_of_casualties'], inplace=True)

    fig = go.Figure()
    fig.add_trace(go.Pie(
        labels=result['Light_Conditions'],
        values=result['Total % Casualties'],
        hole=0.6,
        hoverinfo='label+percent',
        textinfo='label+percent',
        textfont_size=15,
        marker=dict(line=dict(color='#000000', width=1))
    ))

    fig.update_layout(
        title='Percentage of Total Casualties by Light Conditions',
        showlegend=False,
    )

    fig.show()


donut_chart_by_light_conditions()
```



Insights from Analysis:

The dataset consists of 19 columns and 307,973 rows, with no duplicate values and very few null values.

Key findings are as follows:

1. Total accidents and casualties decreased in the year 2022 compared to 2021.
2. In terms of casualties by road type, the "Single Carriage Road" type reported the highest number of casualties.
3. Urban areas had the highest number of casualties compared to other areas.
4. The vehicle type "car" was involved in the most accidents, with a total of 333,485 occurrences.
5. Daytime had the highest number of casualties compared to other light conditions, accounting for 73% of the total casualties.