In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
df = pd.read_csv('accident data.csv')
```

In [3]:
```
df.head()
```

Out[3]:

	Index	Accident_Severity	Accident Date	Latitude	Light_Conditions	District Area	Longitude
0	200701BS64157	Serious	05-06-2019	51.506187	Darkness - lights lit	Kensington and Chelsea	-0.209082
1	200701BS65737	Serious	02-07-2019	51.495029	Daylight	Kensington and Chelsea	-0.173647
2	200701BS66127	Serious	26-08-2019	51.517715	Darkness - lighting unknown	Kensington and Chelsea	-0.210215
3	200701BS66128	Serious	16-08-2019	51.495478	Daylight	Kensington and Chelsea	-0.202731
4	200701BS66837	Slight	03-09-2019	51.488576	Darkness - lights lit	Kensington and Chelsea	-0.192487

In [4]:
```
df.tail()
```

Out[4]:

	Index	Accident_Severity	Accident Date	Latitude	Light_Conditions	District Area	Longit
660674	201091NM01760	Slight	18-02-2022	57.374005	Daylight	Highland	-3.467
660675	201091NM01881	Slight	21-02-2022	57.232273	Darkness - no lighting	Highland	-3.809
660676	201091NM01935	Slight	23-02-2022	57.585044	Daylight	Highland	-3.862
660677	201091NM01964	Serious	23-02-2022	57.214898	Darkness - no lighting	Highland	-3.823
660678	201091NM02142	Serious	28-02-2022	57.575210	Daylight	Highland	-3.895

In [5]:
```
df.shape
```

Out[5]:
```
(660679, 14)
```

In [6]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 660679 entries, 0 to 660678
Data columns (total 14 columns):
 #   Column                       Non-Null Count   Dtype
```

```
---  ------                   --------------   -----
 0   Index                    660679 non-null  object
 1   Accident_Severity        660679 non-null  object
 2   Accident Date            660679 non-null  object
 3   Latitude                 660654 non-null  float64
 4   Light_Conditions         660679 non-null  object
 5   District Area            660679 non-null  object
 6   Longitude                660653 non-null  float64
 7   Number_of_Casualties     660679 non-null  int64
 8   Number_of_Vehicles       660679 non-null  int64
 9   Road_Surface_Conditions  659953 non-null  object
 10  Road_Type                656159 non-null  object
 11  Urban_or_Rural_Area      660664 non-null  object
 12  Weather_Conditions       646551 non-null  object
 13  Vehicle_Type             660679 non-null  object
dtypes: float64(2), int64(2), object(10)
memory usage: 70.6+ MB
```

In [7]:
```
df.columns
```

Out[7]:
```
Index(['Index', 'Accident_Severity', 'Accident Date', 'Latitude',
       'Light_Conditions', 'District Area', 'Longitude',
       'Number_of_Casualties', 'Number_of_Vehicles', 'Road_Surface_Condition
s',
       'Road_Type', 'Urban_or_Rural_Area', 'Weather_Conditions',
       'Vehicle_Type'],
      dtype='object')
```

In [8]:
```
df.isnull().sum()
```

Out[8]:
```
Index                        0
Accident_Severity            0
Accident Date                0
Latitude                    25
Light_Conditions             0
District Area                0
Longitude                   26
Number_of_Casualties         0
Number_of_Vehicles           0
Road_Surface_Conditions    726
Road_Type                 4520
Urban_or_Rural_Area         15
Weather_Conditions       14128
Vehicle_Type                 0
dtype: int64
```

In [9]:
```
 df.duplicated().sum()
```

Out[9]:
```
19
```

In [10]:
```
df=df.drop_duplicates()
```

In [11]:
```
df.isnull().sum()
```

Out[11]:
```
Index                        0
Accident_Severity            0
Accident Date                0
Latitude                    25
Light_Conditions             0
District Area                0
Longitude                   26
Number_of_Casualties         0
```

```
Number_of_Vehicles             0
Road_Surface_Conditions      726
Road_Type                   4520
Urban_or_Rural_Area           15
Weather_Conditions         14127
Vehicle_Type                   0
dtype: int64
```

In [12]:
```
df.describe()
```

Out[12]:

	Latitude	Longitude	Number_of_Casualties	Number_of_Vehicles
count	660635.000000	660634.000000	660660.000000	660660.000000
mean	52.553857	-1.431216	1.357047	1.831255
std	1.406909	1.383322	0.824856	0.715272
min	49.914430	-7.516225	1.000000	1.000000
25%	51.490692	-2.332278	1.000000	1.000000
50%	52.315628	-1.411666	1.000000	2.000000
75%	53.453451	-0.232870	1.000000	2.000000
max	60.757544	1.762010	68.000000	32.000000

In [13]:
```
df.nunique()
```

Out[13]:
```
Index                      421020
Accident_Severity               3
Accident Date                1461
Latitude                   511618
Light_Conditions                5
District Area                 422
Longitude                  529766
Number_of_Casualties           36
Number_of_Vehicles             19
Road_Surface_Conditions         5
Road_Type                       5
Urban_or_Rural_Area             3
Weather_Conditions              8
Vehicle_Type                   16
dtype: int64
```

In [14]:
```
object_columns=df.select_dtypes(include=['object','bool']).columns
print("Object type columns:")
print(object_columns)
numerical_columns=df.select_dtypes(include=['int64','float64']).columns
print("\nNumerical type columns:")
print(numerical_columns)
```

```
Object type columns:
Index(['Index', 'Accident_Severity', 'Accident Date', 'Light_Conditions',
       'District Area', 'Road_Surface_Conditions', 'Road_Type',
       'Urban_or_Rural_Area', 'Weather_Conditions', 'Vehicle_Type'],
      dtype='object')

Numerical type columns:
Index(['Latitude', 'Longitude', 'Number_of_Casualties', 'Number_of_Vehicles'],
dtype='object')
```

In [15]:
```
def classify_features(df):
    categorical_features = []
    non_categorical_features = []
    discrete_features = []
    continuous_features = []
    for column in df.columns:
        if df[column].dtype in ['object', 'bool']:
            if df[column].nunique() < 15:
                categorical_features.append(column)
            else:
                non_categorical_features.append(column)
        elif df[column].dtype in ['int64', 'float64']:
            if df[column].nunique() < 10:
                discrete_features.append(column)
            else:
                continuous_features.append(column)
    return categorical_features, non_categorical_features, discrete_features,
```

In [16]:
```
categorical,non_categorical,discrete,continuous=classify_features(df)
```

In [17]:
```
print("Categorical Features:",categorical)
print("Non-Categorical Features:",non_categorical)
print("Discrete Features:",discrete)
print("Continuous Features:",continuous)
```

```
Categorical Features: ['Accident_Severity', 'Light_Conditions', 'Road_Surface_
Conditions', 'Road_Type', 'Urban_or_Rural_Area', 'Weather_Conditions']
Non-Categorical Features: ['Index', 'Accident Date', 'District Area', 'Vehicle
_Type']
Discrete Features: []
Continuous Features: ['Latitude', 'Longitude', 'Number_of_Casualties', 'Number
_of_Vehicles']
```

In [18]:
```
null_counts = df.isnull().sum()
null_columns = null_counts[null_counts > 0].index.tolist()
```

In [19]:
```
total_rows = len(df)
null_percentage = (null_counts / total_rows) * 100
```

In [20]:
```
null_df = pd.DataFrame({
'Column': null_counts.index,
'Null Count': null_counts.values,
'Null Percentage': null_percentage.values
})
```

In [21]:
```
null_df = null_df.sort_values(by='Null Count', ascending=False)
```

In [22]:
```
null_df
```

Out[22]:

	Column	Null Count	Null Percentage
12	Weather_Conditions	14127	2.138316
10	Road_Type	4520	0.684164
9	Road_Surface_Conditions	726	0.109890
6	Longitude	26	0.003935
3	Latitude	25	0.003784

	Column	Null Count	Null Percentage
11	Urban_or_Rural_Area	15	0.002270
0	Index	0	0.000000
1	Accident_Severity	0	0.000000
2	Accident Date	0	0.000000
4	Light_Conditions	0	0.000000
5	District Area	0	0.000000
7	Number_of_Casualties	0	0.000000
8	Number_of_Vehicles	0	0.000000
13	Vehicle_Type	0	0.000000

In [23]:
```
mode_value = df['Weather_Conditions'].mode()[0]
df['Weather_Conditions'].fillna(mode_value, inplace=True)
```

In [24]:
```
mode_value = df['Road_Type'].mode()[0]
df['Road_Type'].fillna(mode_value, inplace=True)
```

In [25]:
```
mode_value = df['Road_Surface_Conditions'].mode()[0]
df['Road_Surface_Conditions'].fillna(mode_value, inplace=True)
```

In [26]:
```
mode_value = df['Urban_or_Rural_Area'].mode()[0]
df['Urban_or_Rural_Area'].fillna(mode_value, inplace=True)
```

In [27]:
```
mean_value = df['Latitude'].mean()
df['Latitude'].fillna(mean_value, inplace=True)
```

In [28]:
```
mean_value = df['Longitude'].mean()
df['Longitude'].fillna(mean_value, inplace=True)
```

In [29]:
```
df.isnull().sum()
```

Out[29]:
```
Index                       0
Accident_Severity           0
Accident Date               0
Latitude                    0
Light_Conditions            0
District Area               0
Longitude                   0
Number_of_Casualties        0
Number_of_Vehicles          0
Road_Surface_Conditions     0
Road_Type                   0
Urban_or_Rural_Area         0
Weather_Conditions          0
Vehicle_Type                0
dtype: int64
```

In [30]:
```
for i in categorical:
    print(i, ':')
    print(df[i].unique())
    print()
```

```
Accident_Severity :
['Serious' 'Slight' 'Fatal']

Light_Conditions :
['Darkness - lights lit' 'Daylight' 'Darkness - lighting unknown'
 'Darkness - lights unlit' 'Darkness - no lighting']

Road_Surface_Conditions :
['Dry' 'Wet or damp' 'Snow' 'Frost or ice' 'Flood over 3cm. deep']

Road_Type :
['Single carriageway' 'Dual carriageway' 'One way street' 'Roundabout'
 'Slip road']

Urban_or_Rural_Area :
['Urban' 'Rural' 'Unallocated']

Weather_Conditions :
['Fine no high winds' 'Raining no high winds' 'Other' 'Fine + high winds'
 'Raining + high winds' 'Snowing no high winds' 'Fog or mist'
 'Snowing + high winds']
```

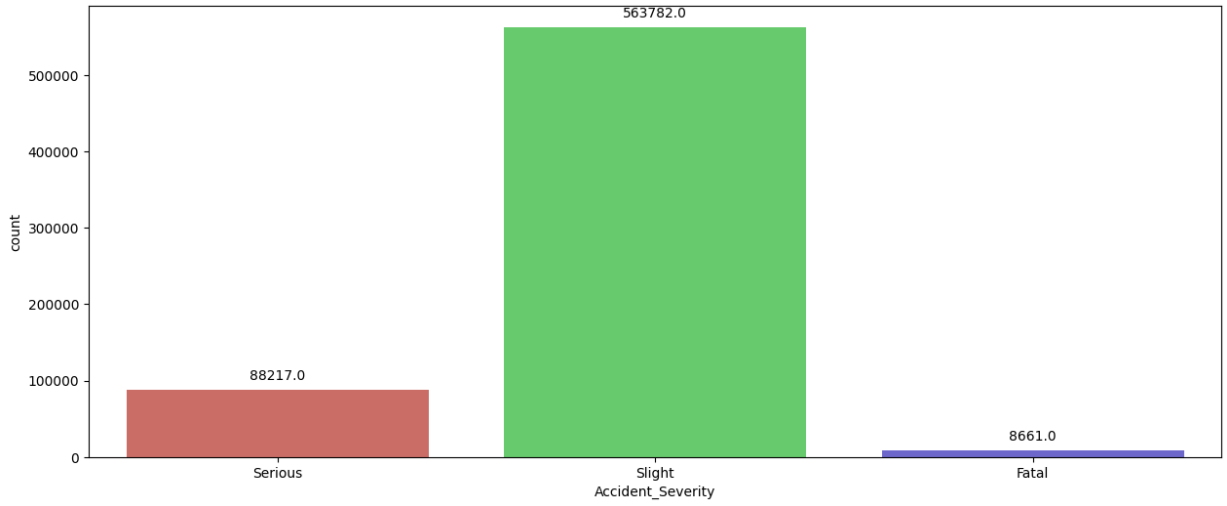
In [31]:
```
for i in categorical:
    print(i, ':')
    print(df[i].value_counts())
    print()
```

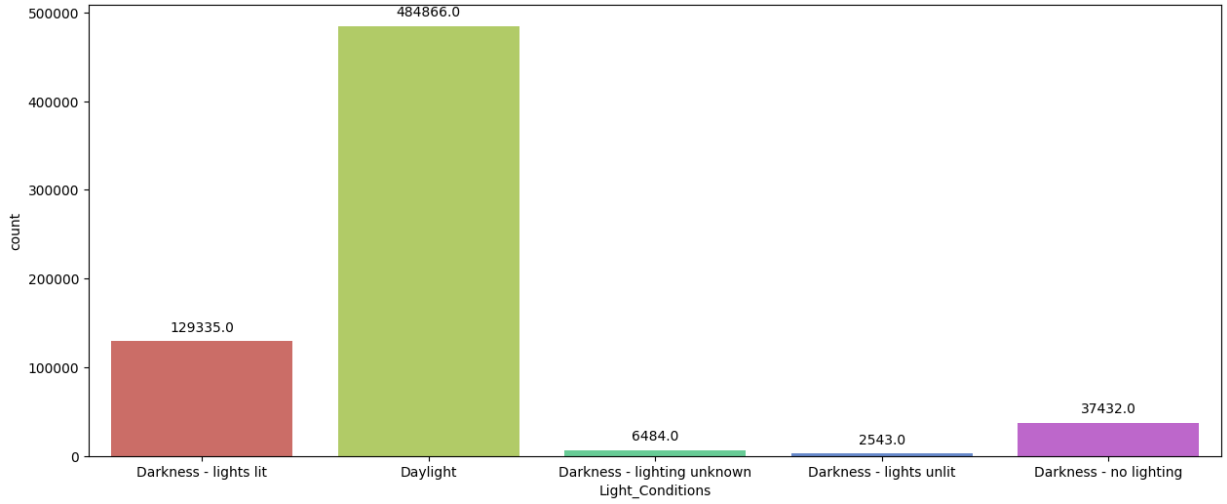
```
Accident_Severity :
Slight     563782
Serious     88217
Fatal        8661
Name: Accident_Severity, dtype: int64

Light_Conditions :
Daylight                       484866
Darkness - lights lit          129335
Darkness - no lighting          37432
Darkness - lighting unknown      6484
Darkness - lights unlit          2543
Name: Light_Conditions, dtype: int64

Road_Surface_Conditions :
Dry                     448536
Wet or damp             186705
Frost or ice             18514
Snow                      5888
Flood over 3cm. deep      1017
Name: Road_Surface_Conditions, dtype: int64

Road_Type :
Single carriageway    496655
Dual carriageway       99416
Roundabout             43989
One way street         13559
Slip road               7041
Name: Road_Type, dtype: int64

Urban_or_Rural_Area :
Urban          421675
Rural          238974
Unallocated        11
```

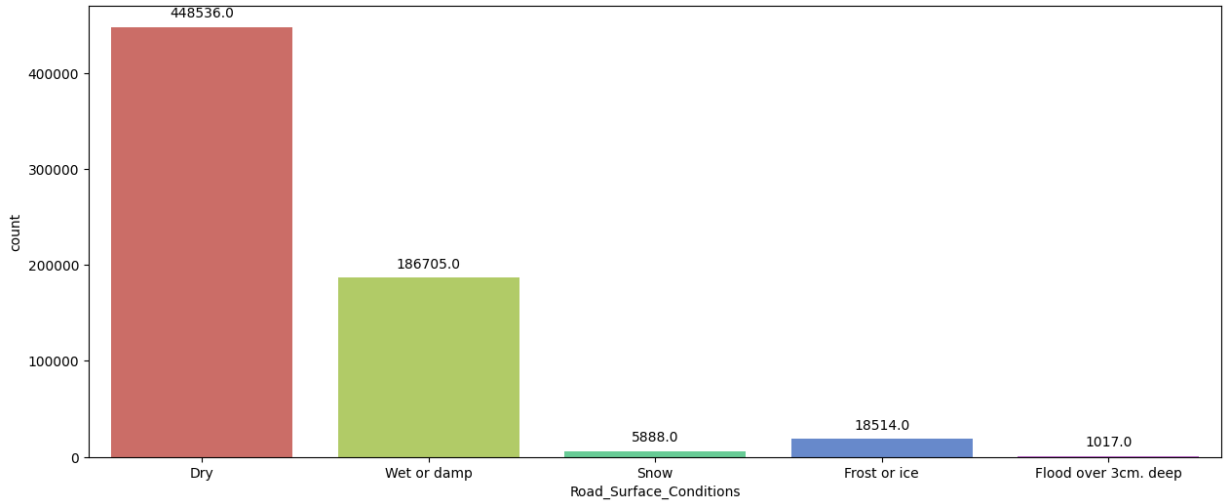
```
Name: Urban_or_Rural_Area, dtype: int64

Weather_Conditions :
Fine no high winds       534999
Raining no high winds     79696
Other                     17148
Raining + high winds       9615
Fine + high winds          8553
Snowing no high winds      6237
Fog or mist                3528
Snowing + high winds        884
Name: Weather_Conditions, dtype: int64
```

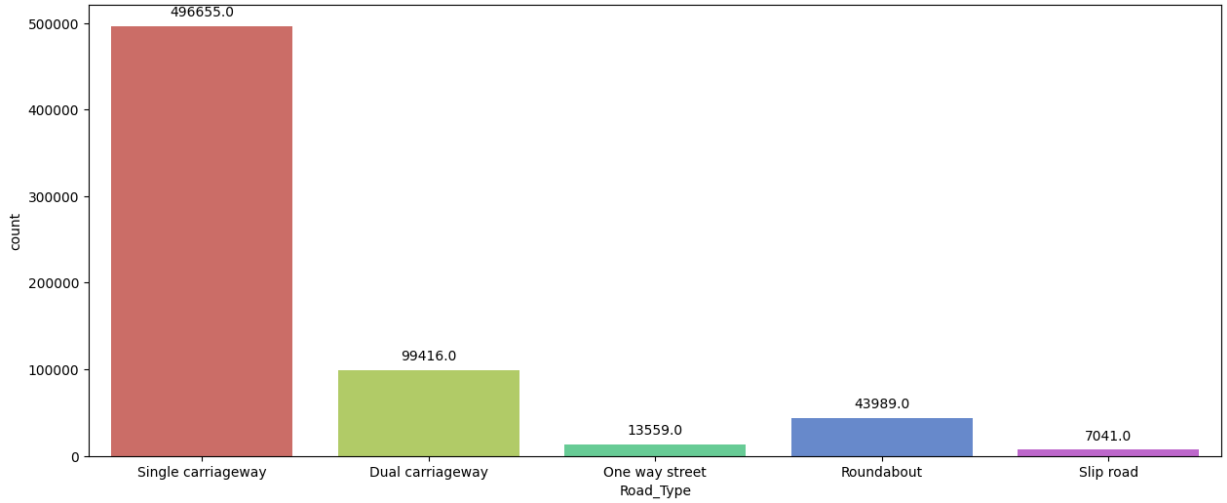
In [32]:

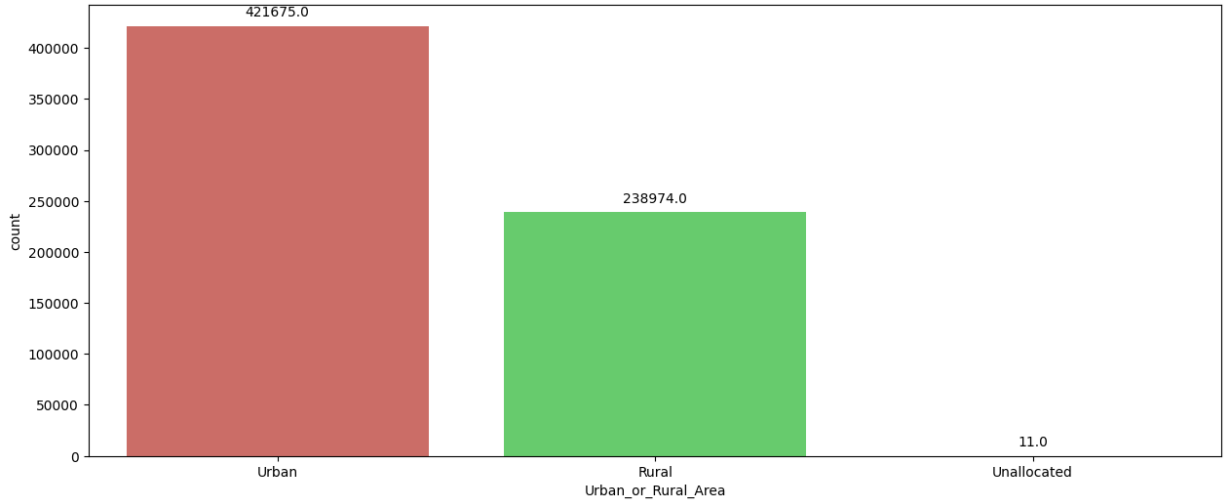
```
for i in categorical:
    plt.figure(figsize=(15, 6))
    ax = sns.countplot(x=i, data=df, palette='hls')
    for p in ax.patches:
        ax.annotate(f'{p.get_height()}', (p.get_x() + p.get_width() / 2., p.ge
                    ha='center', va='center', xytext=(0, 10), textcoords='offs
    plt.show()
```

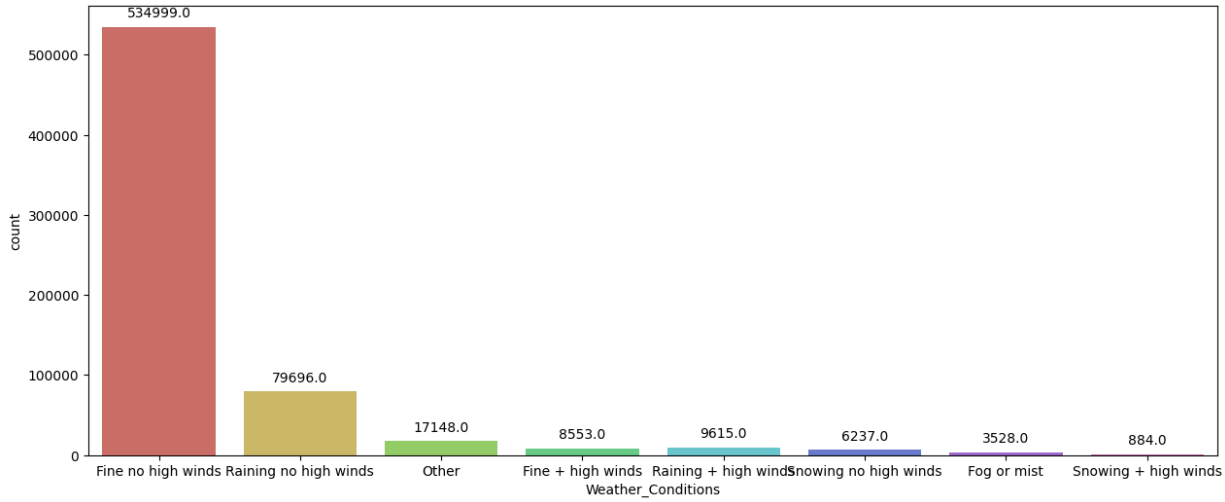




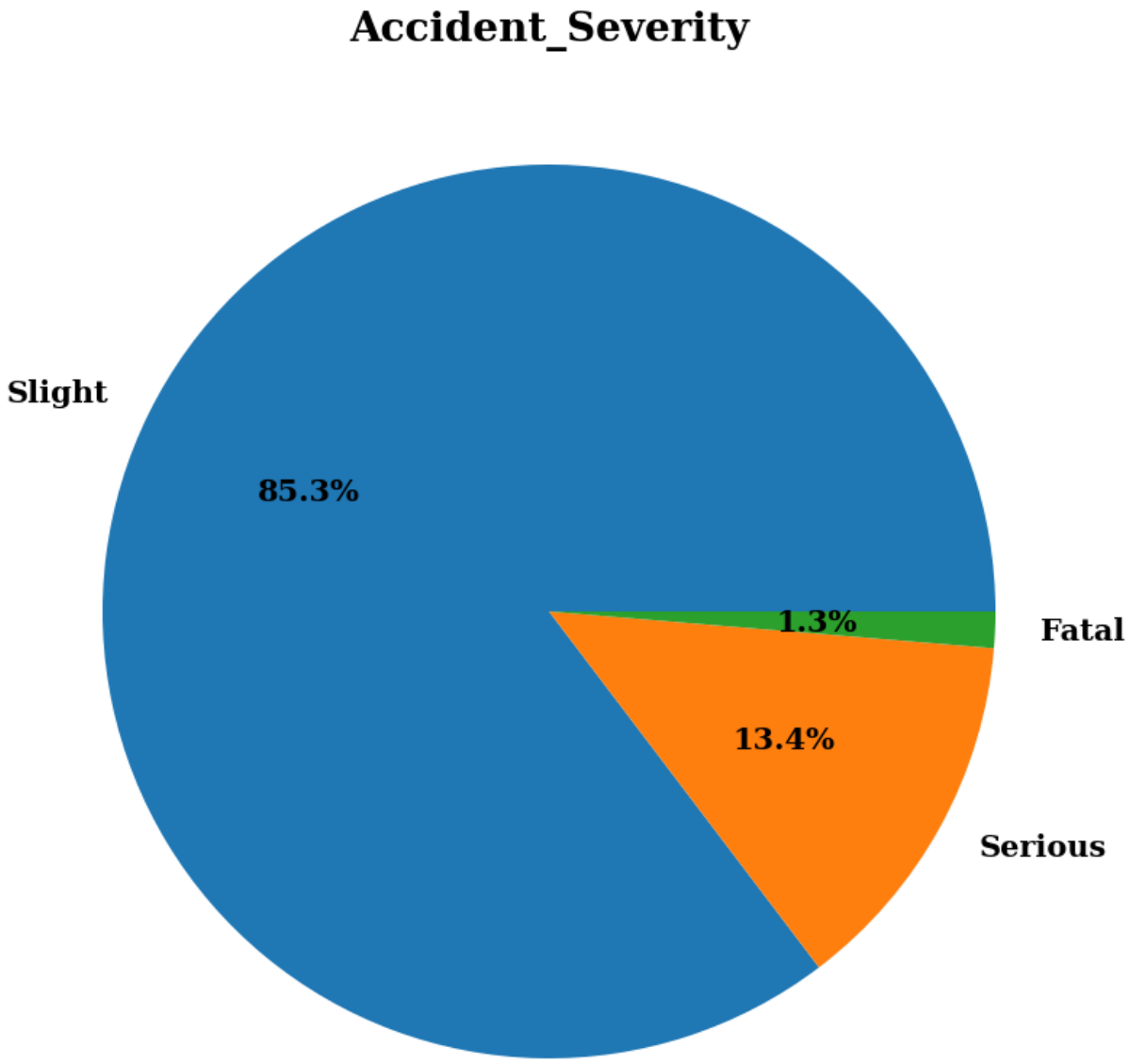




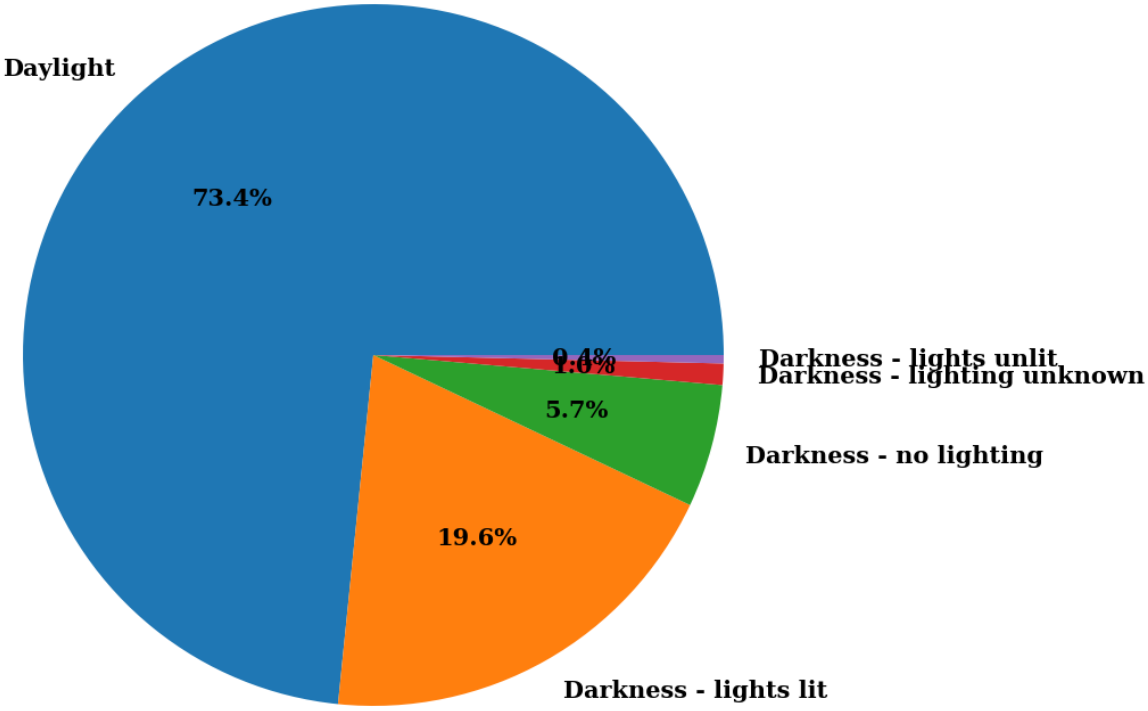




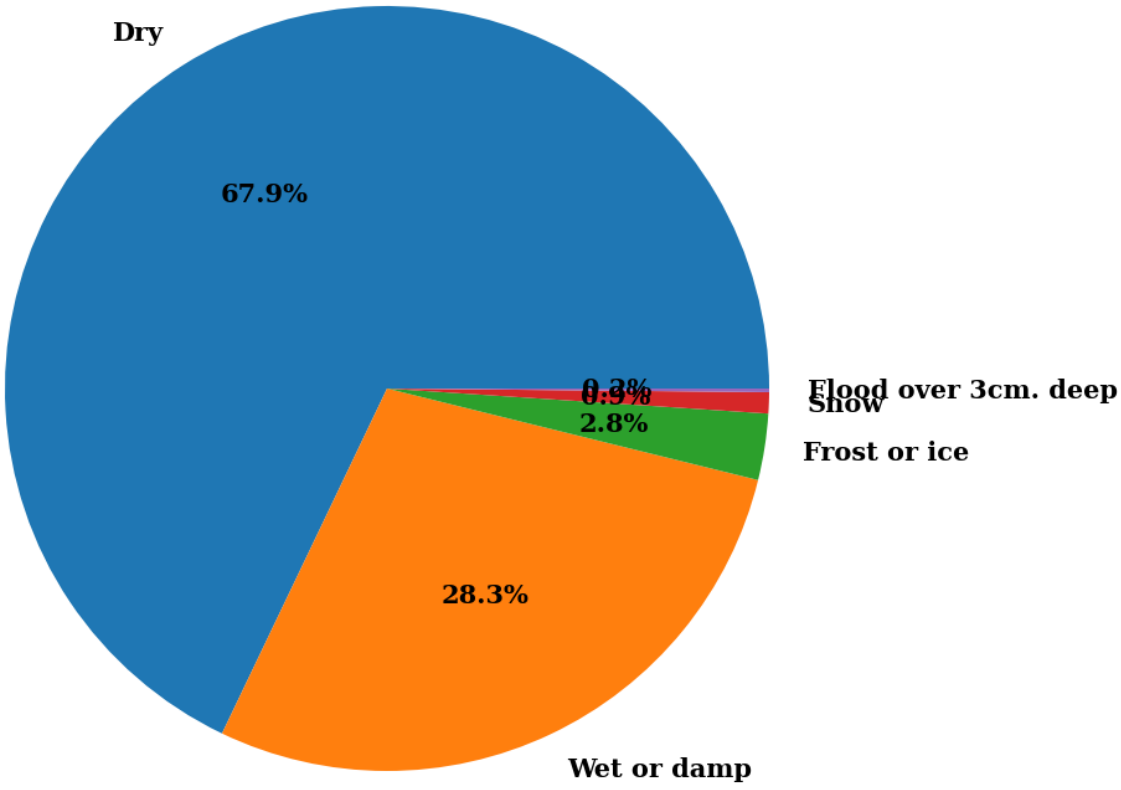
In [33]:
```
for i in categorical:
    plt.figure(figsize=(20,10))
    plt.pie(df[i].value_counts(),labels=df[i].value_counts().index,autopct='%1
       'color':'black',
       'weight':'bold',
       'family':'serif'})
    hfont = {'fontname':'serif','weight':'bold'}
    plt.title(i,size=20,**hfont)
    plt.show()
```

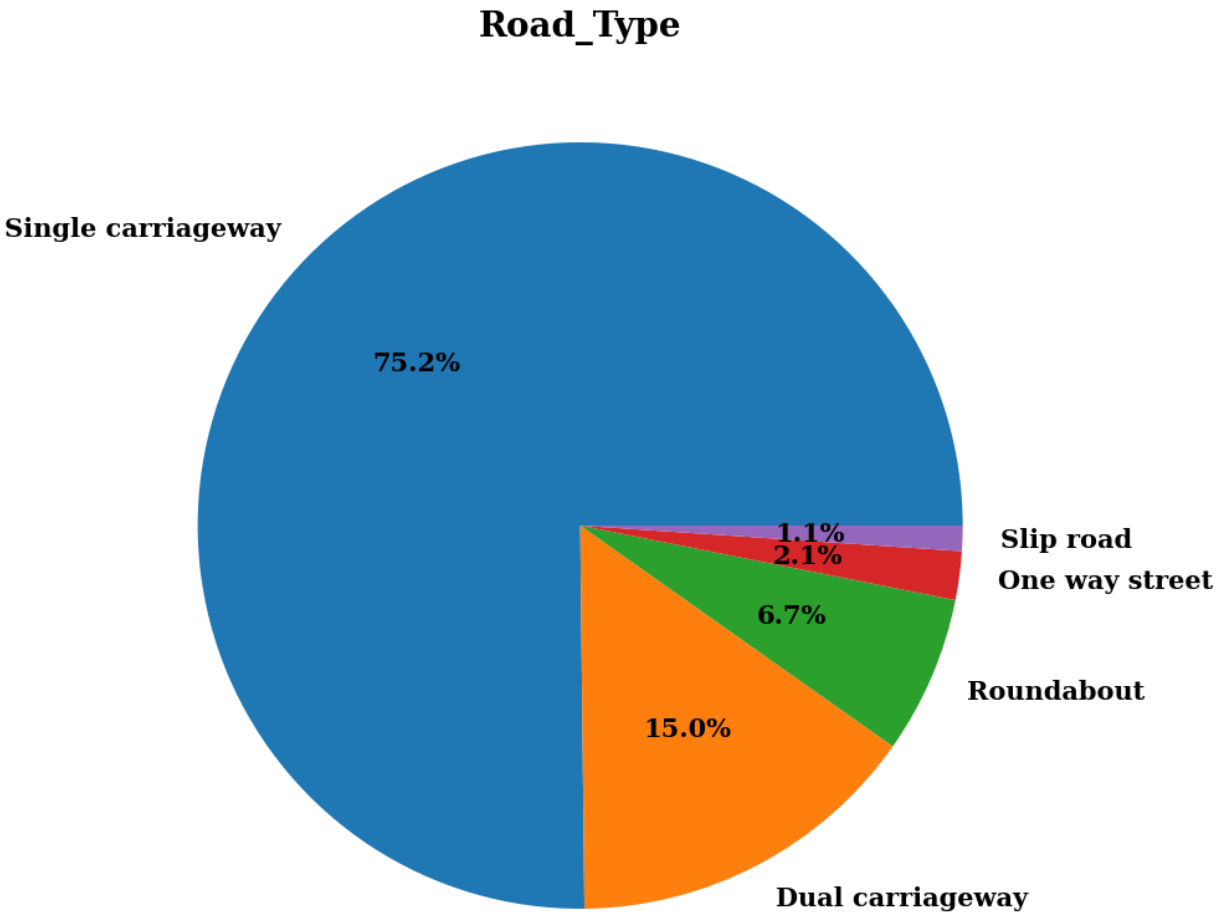


Light_Conditions

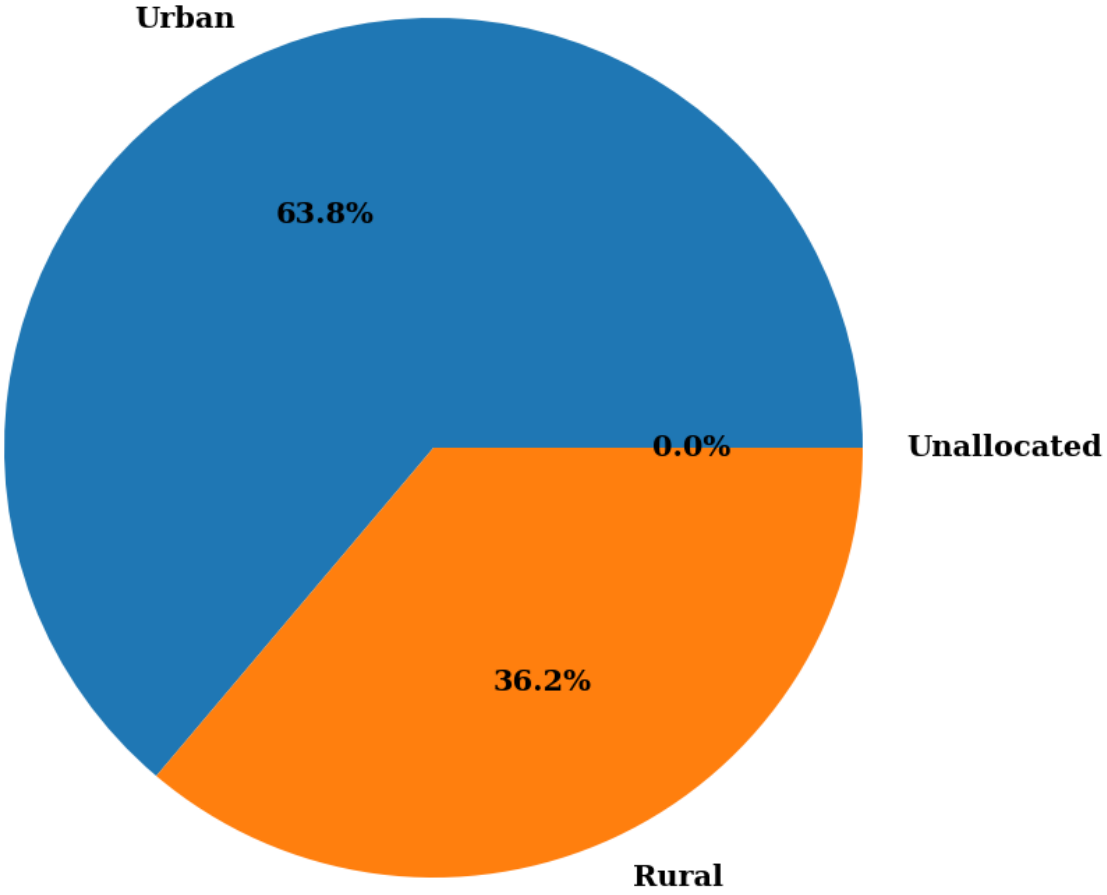


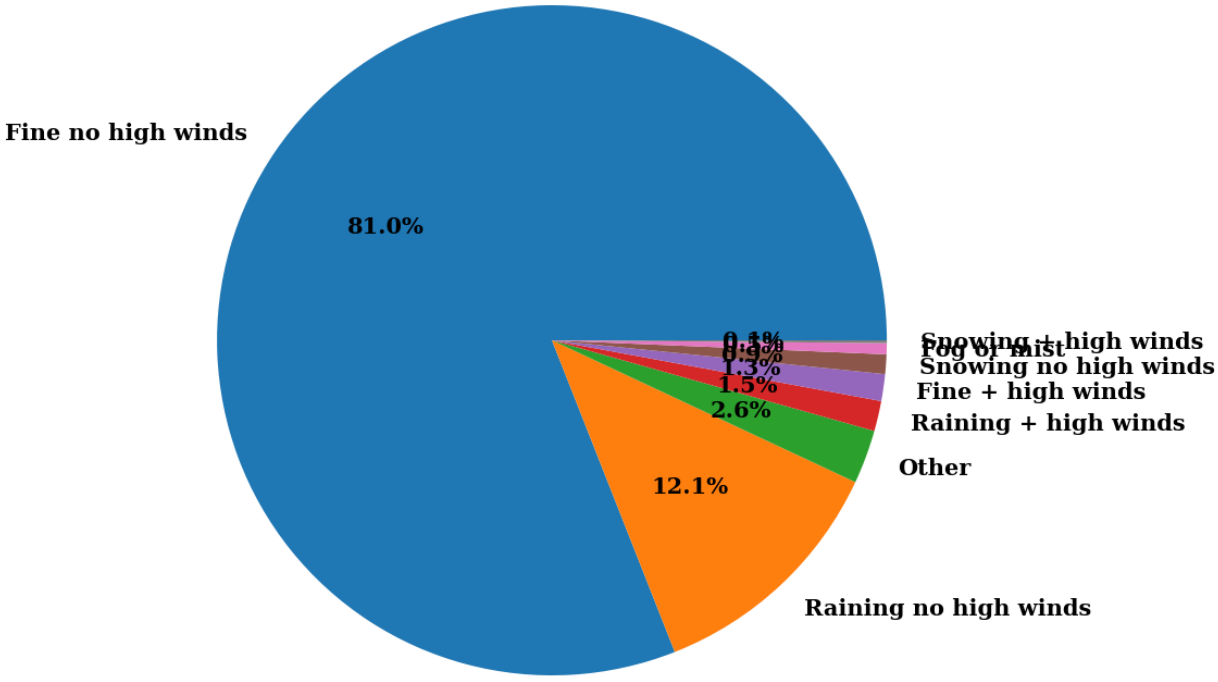
Road_Surface_Conditions





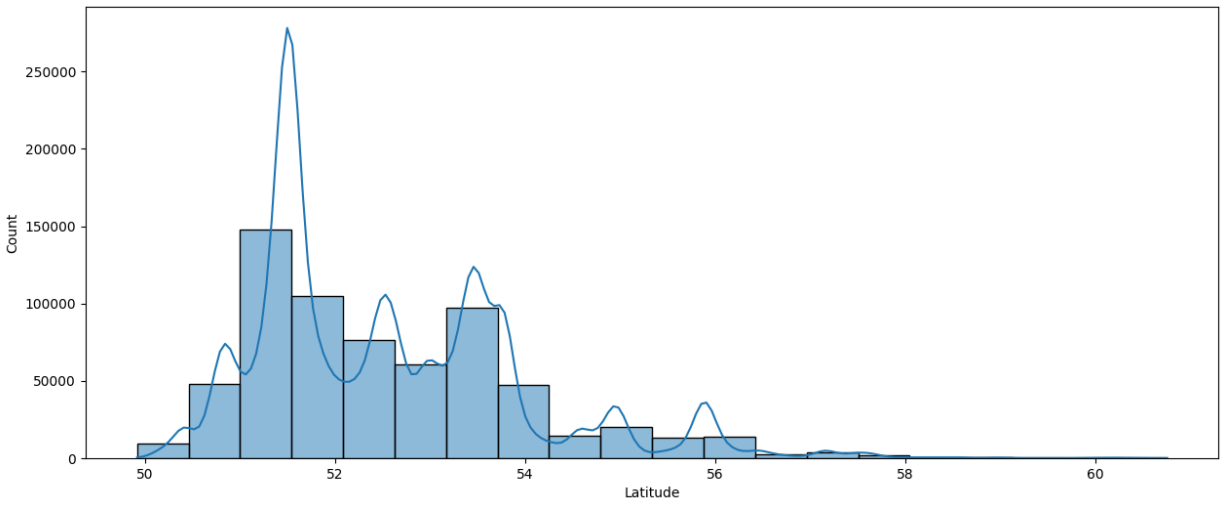
Urban_or_Rural_Area



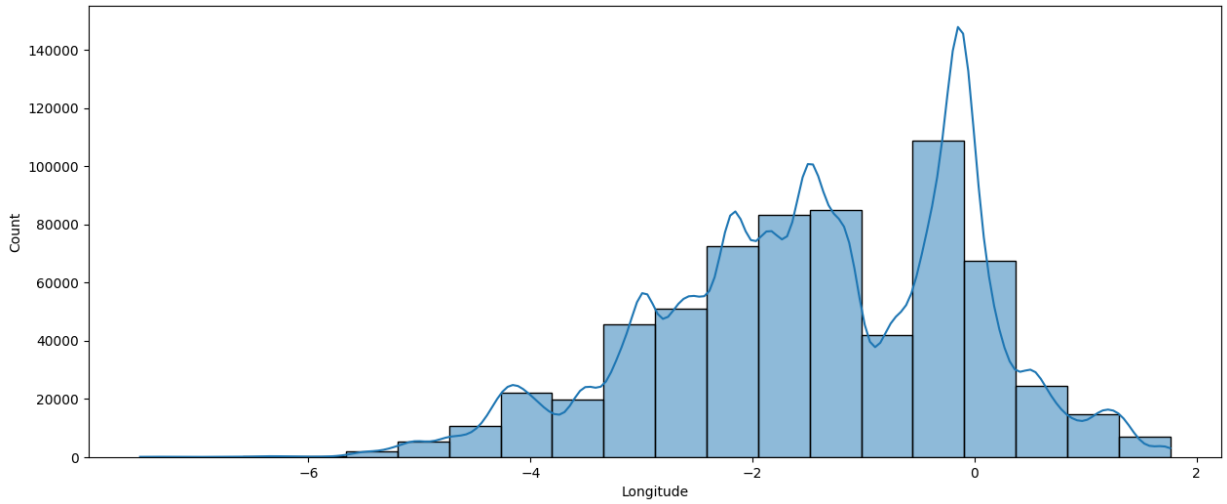


In [34]:
```
for i in continuous:
    plt.figure(figsize=(15,6))
    sns.histplot(df[i], bins = 20,kde = True,palette='hls')
    plt.show()
```

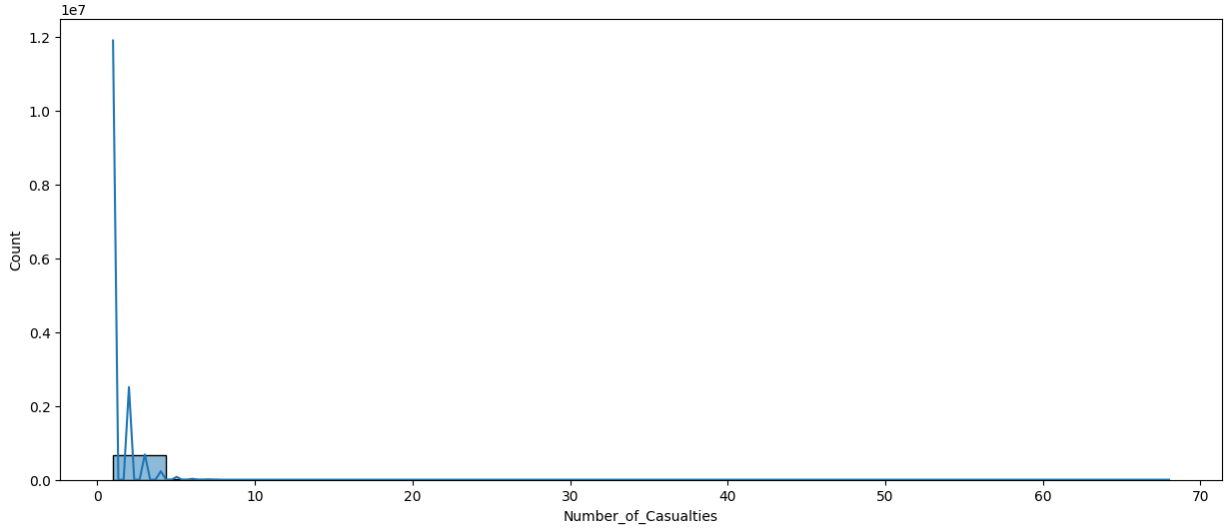
```
C:\Users\tamil\AppData\Local\Temp\ipykernel_3900\2283040530.py:3: UserWarning:
Ignoring `palette` because no `hue` variable has been assigned.
  sns.histplot(df[i], bins = 20,kde = True,palette='hls')
```



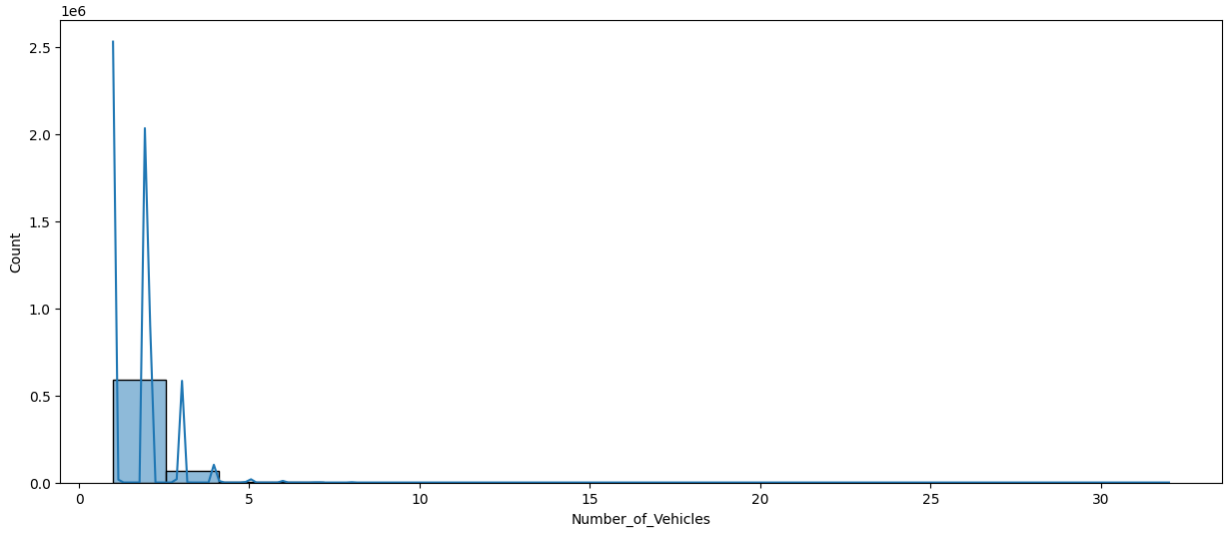

```
C:\Users\tamil\AppData\Local\Temp\ipykernel_3900\2283040530.py:3: UserWarning:
Ignoring `palette` because no `hue` variable has been assigned.
  sns.histplot(df[i], bins = 20,kde = True,palette='hls')
```




```
C:\Users\tamil\AppData\Local\Temp\ipykernel_3900\2283040530.py:3: UserWarning:
Ignoring `palette` because no `hue` variable has been assigned.
  sns.histplot(df[i], bins = 20,kde = True,palette='hls')
```

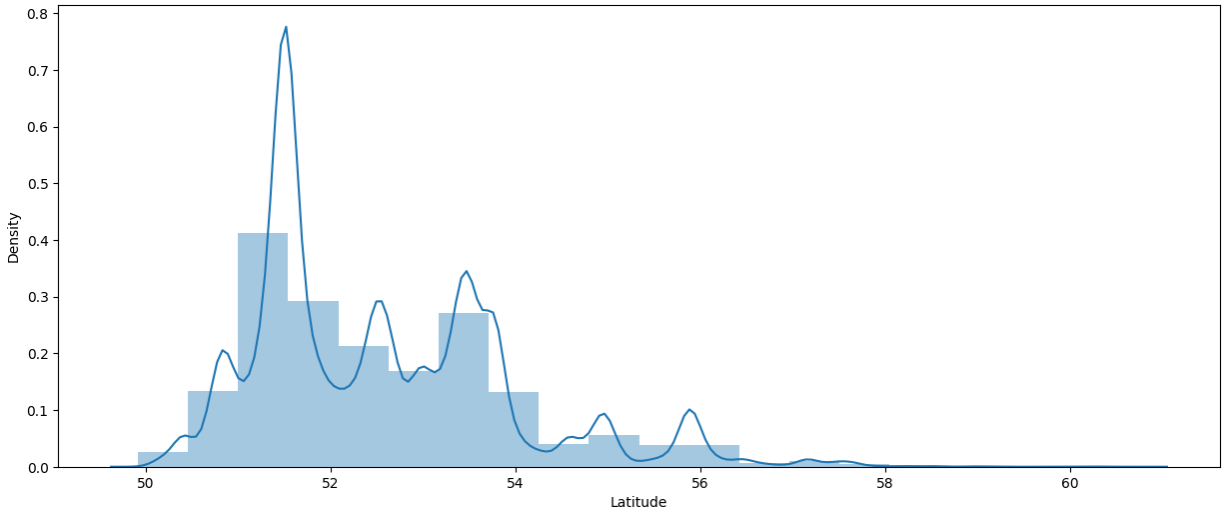



```
C:\Users\tamil\AppData\Local\Temp\ipykernel_3900\2283040530.py:3: UserWarning:
Ignoring `palette` because no `hue` variable has been assigned.
  sns.histplot(df[i], bins = 20,kde = True,palette='hls')
```

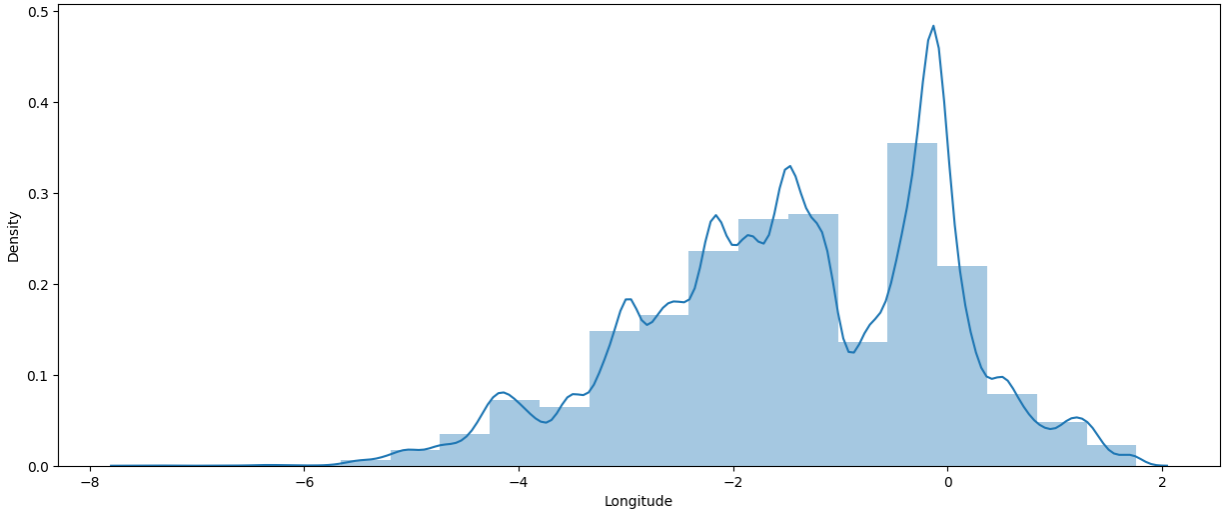


In [35]:
```
for i in continuous:
    plt.figure(figsize=(15,6))
    sns.distplot(df[i], bins = 20, kde = True)
    plt.show()
```

```
C:\Users\tamil\AppData\Local\Temp\ipykernel_3900\3589386306.py:3: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df[i], bins = 20, kde = True)
```

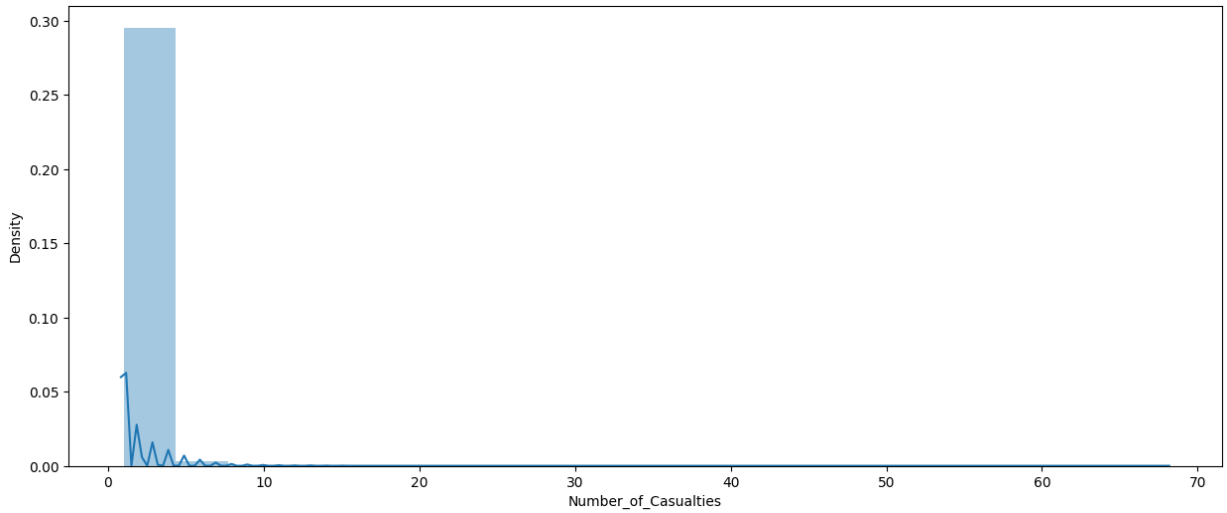



```
C:\Users\tamil\AppData\Local\Temp\ipykernel_3900\3589386306.py:3: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df[i], bins = 20, kde = True)
```

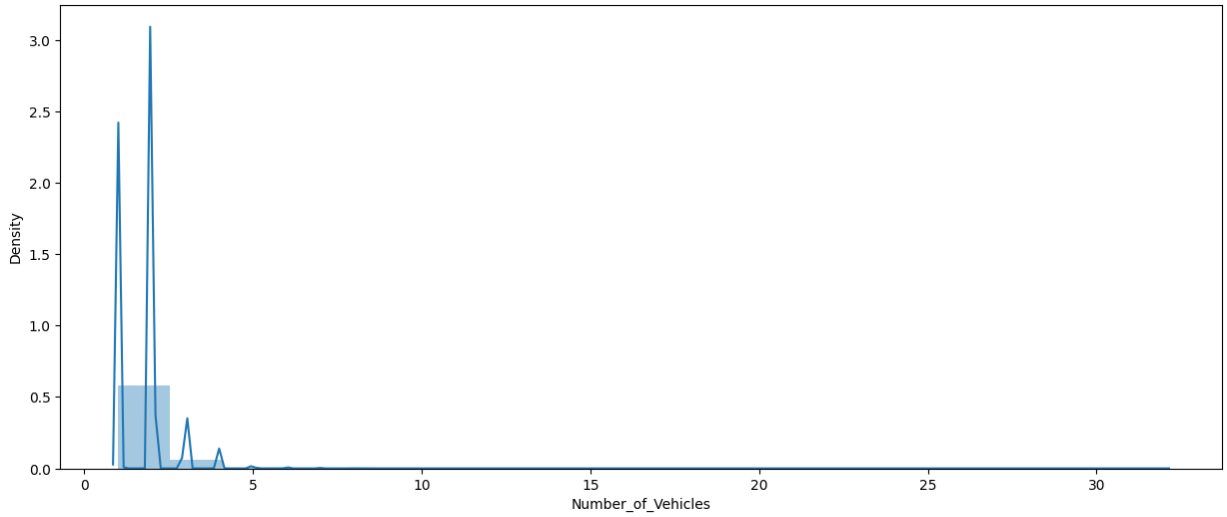



```
C:\Users\tamil\AppData\Local\Temp\ipykernel_3900\3589386306.py:3: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).
```

```
For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df[i], bins = 20, kde = True)
```

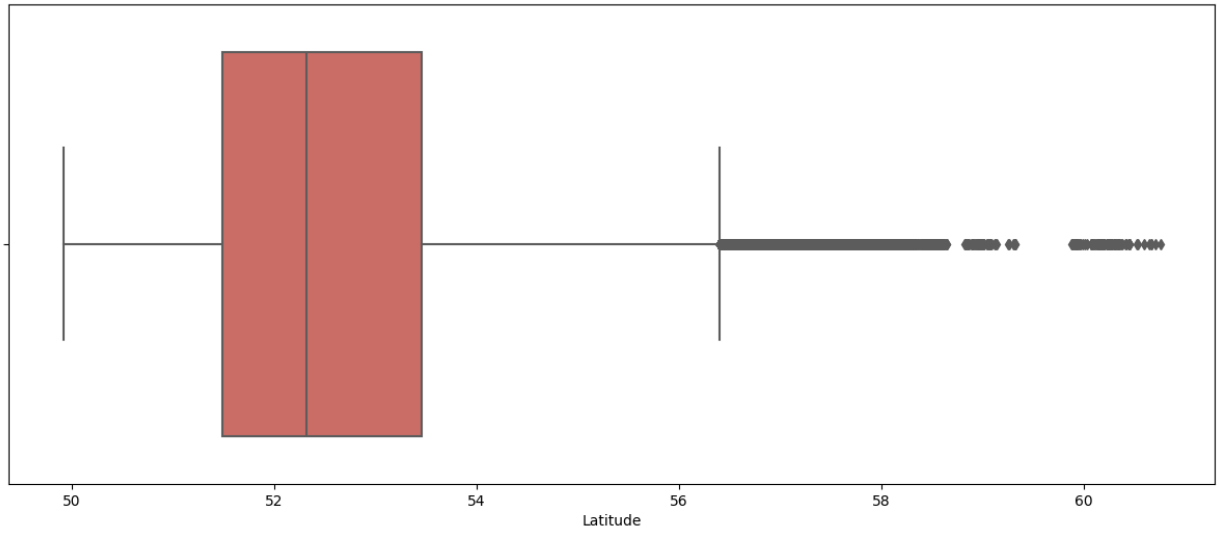


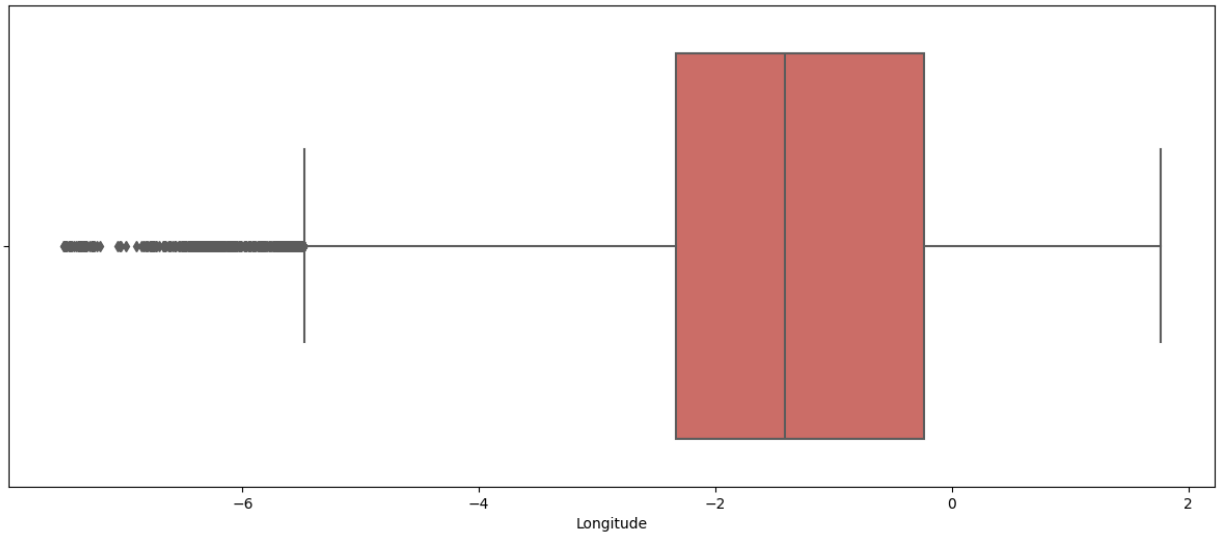

```
C:\Users\tamil\AppData\Local\Temp\ipykernel_3900\3589386306.py:3: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df[i], bins = 20, kde = True)
```

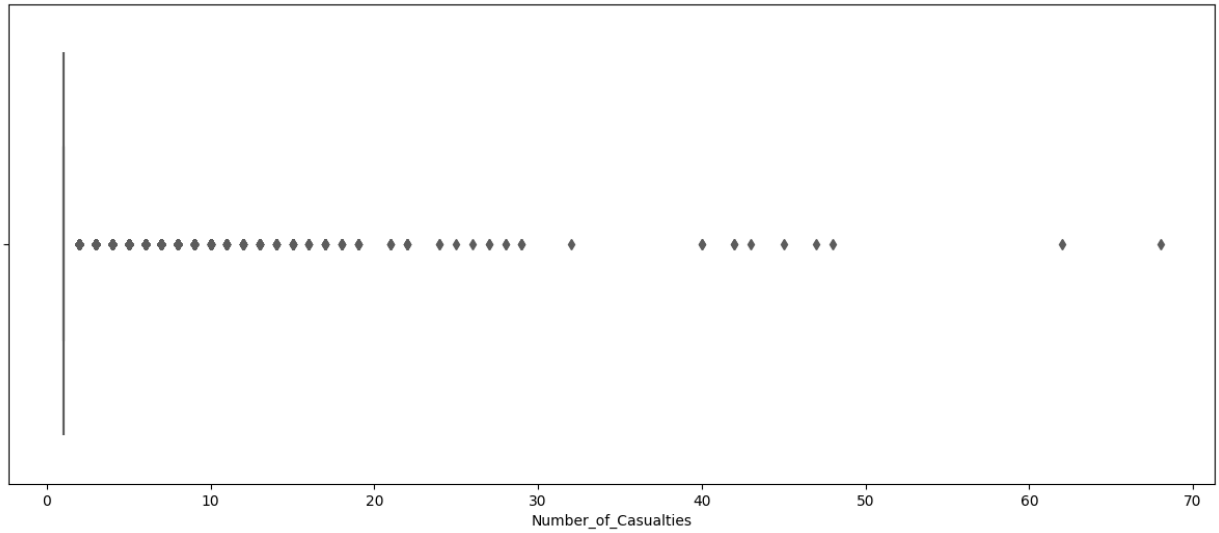


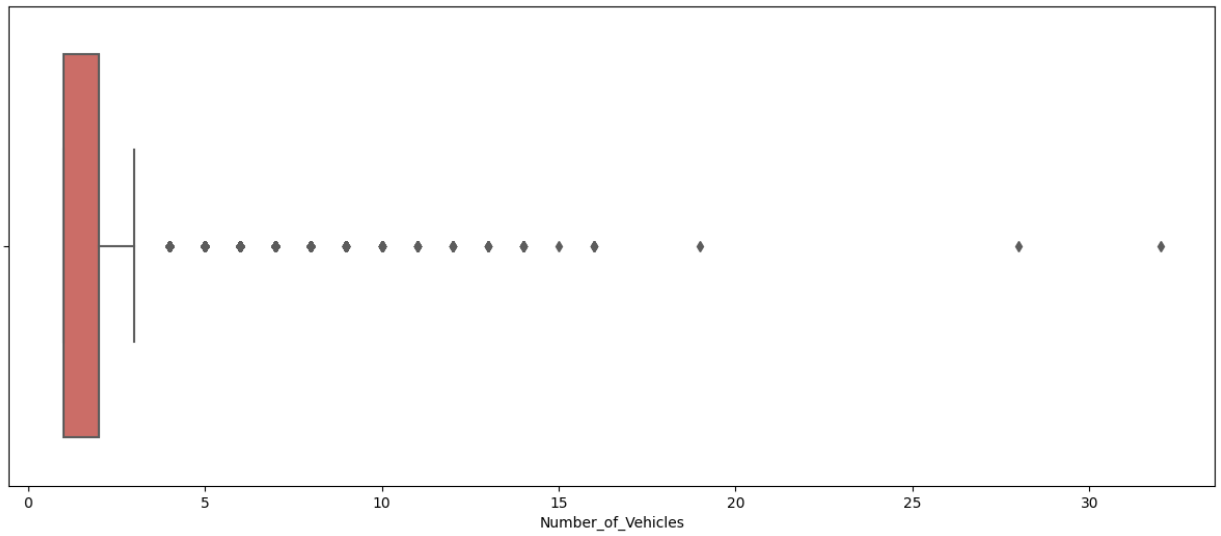
In [36]:
```
import warnings
warnings.filterwarnings('ignore')
```

In [37]:
```
for i in continuous:
    plt.figure(figsize=(15, 6))
    sns.boxplot(x=i, data=df, palette='hls')  # Pass `i` as `x=i`
    plt.show()
```

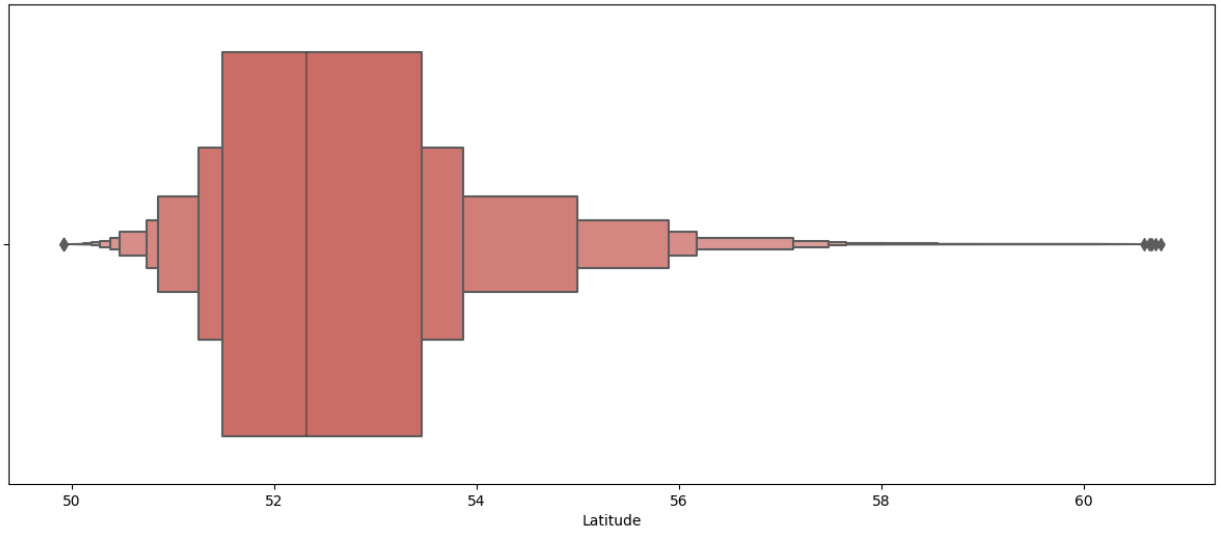


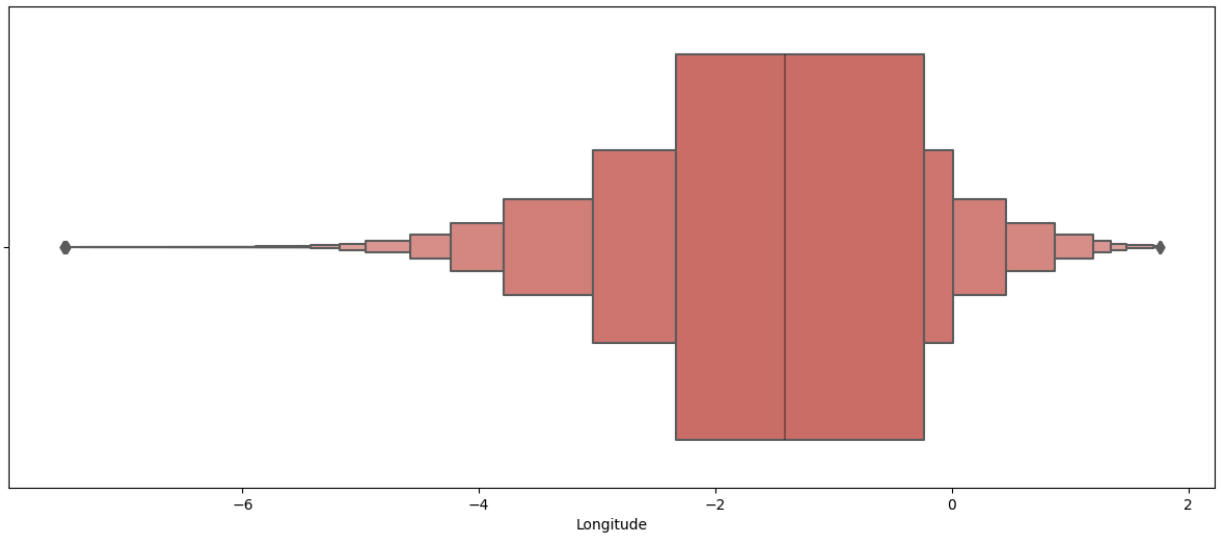


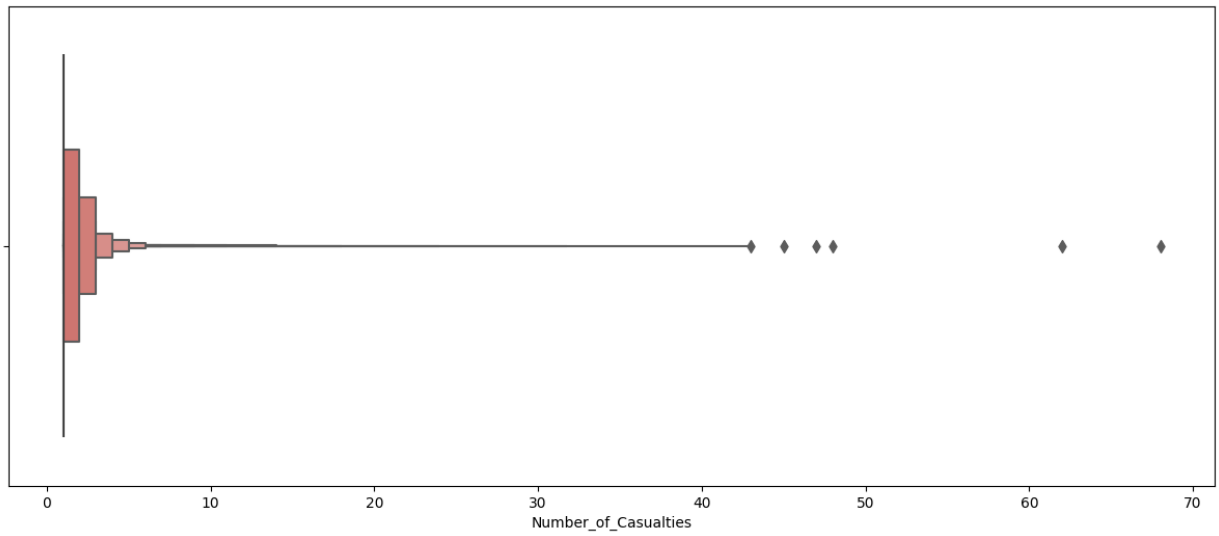


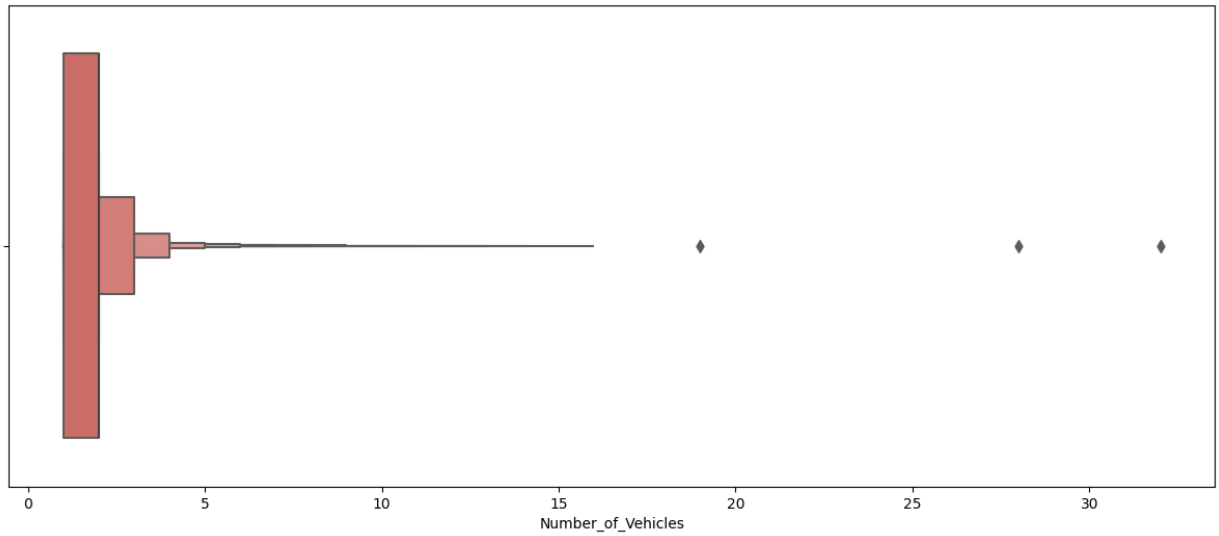


In [38]:
```
for i in continuous:
    plt.figure(figsize=(15,6))
    sns.boxenplot(x=i, data = df, palette='hls')
    plt.show()
```

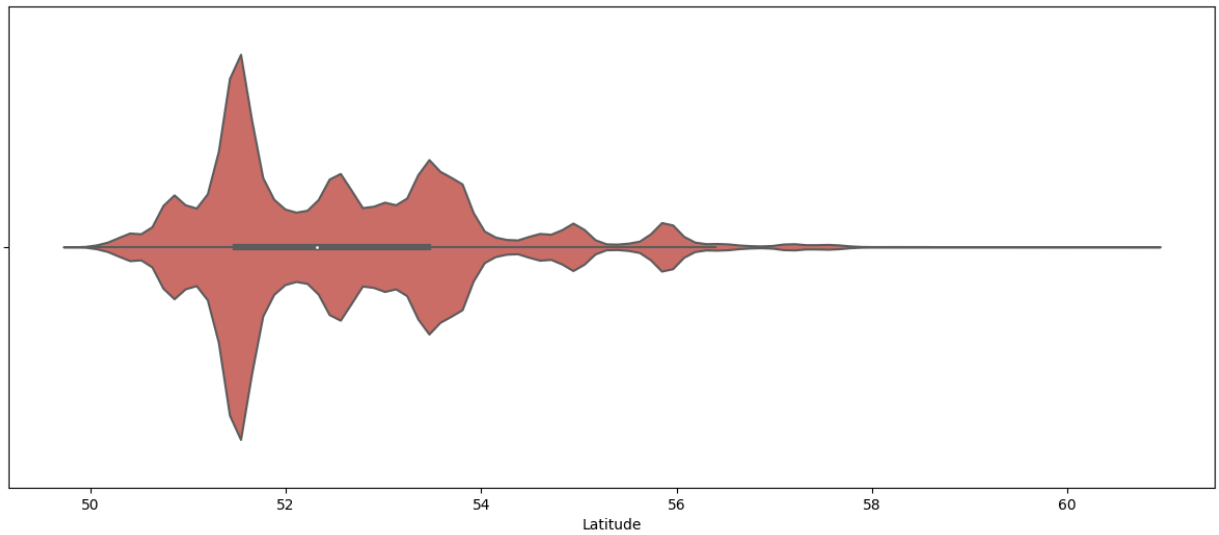


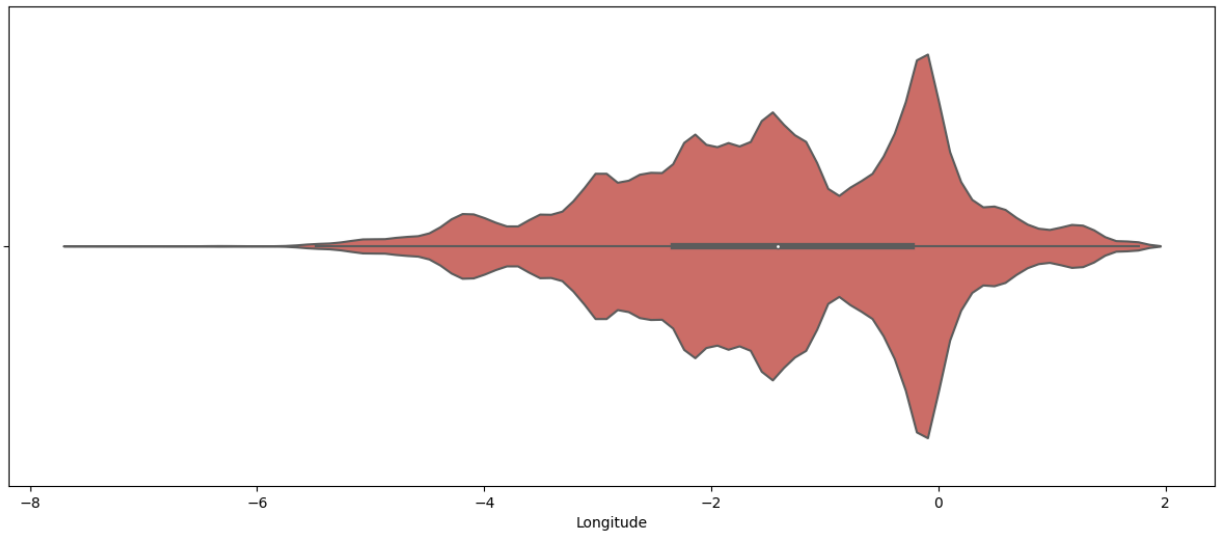


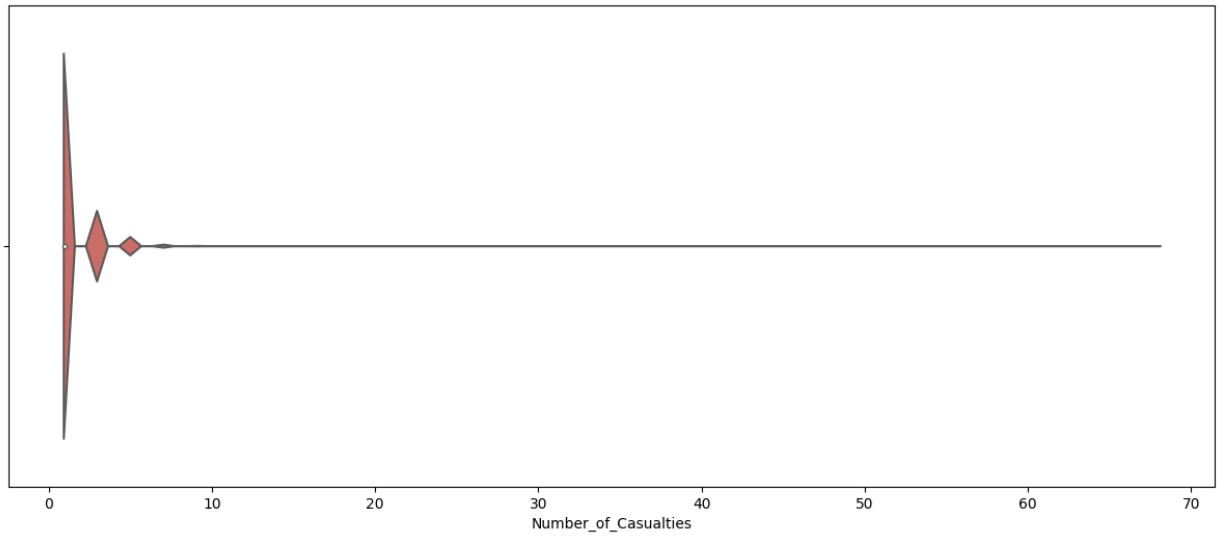


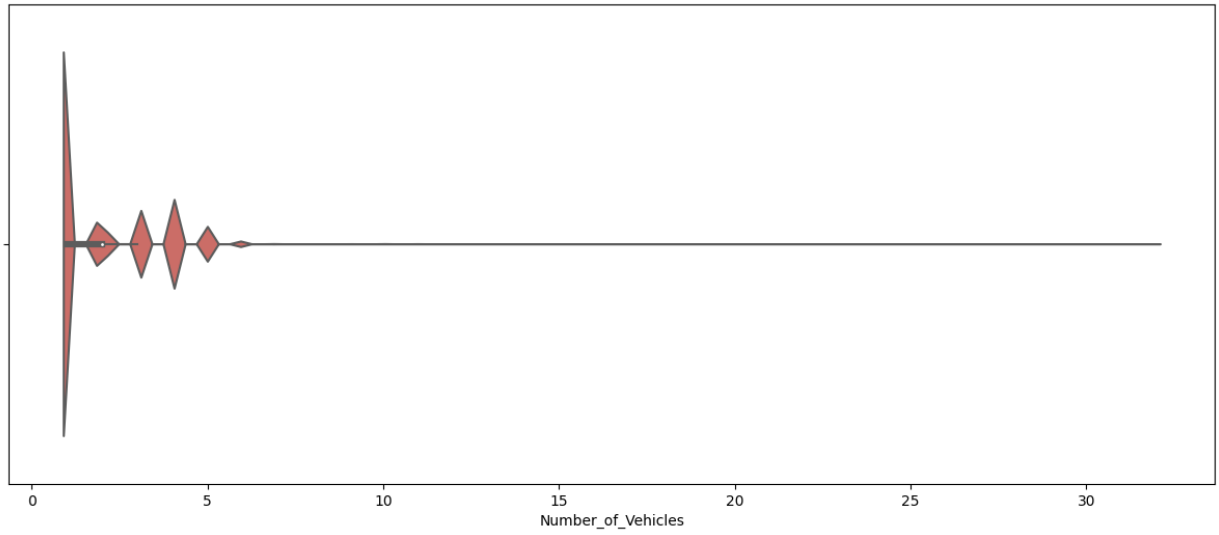


In [39]:
```
for i in continuous:
    plt.figure(figsize=(15,6))
    sns.violinplot(x = i, data = df, palette='hls')
    plt.show()
```

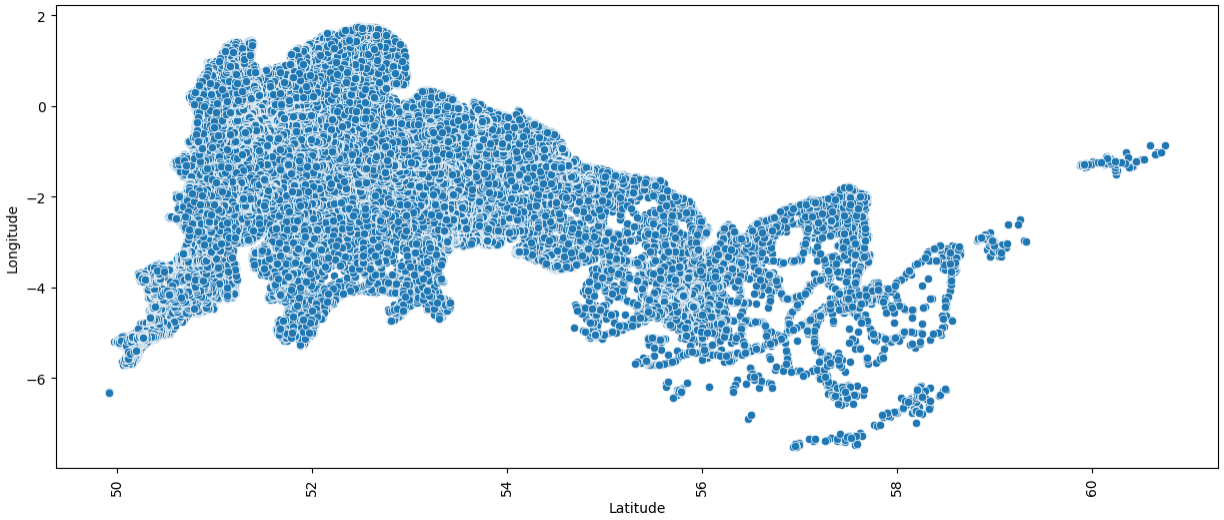


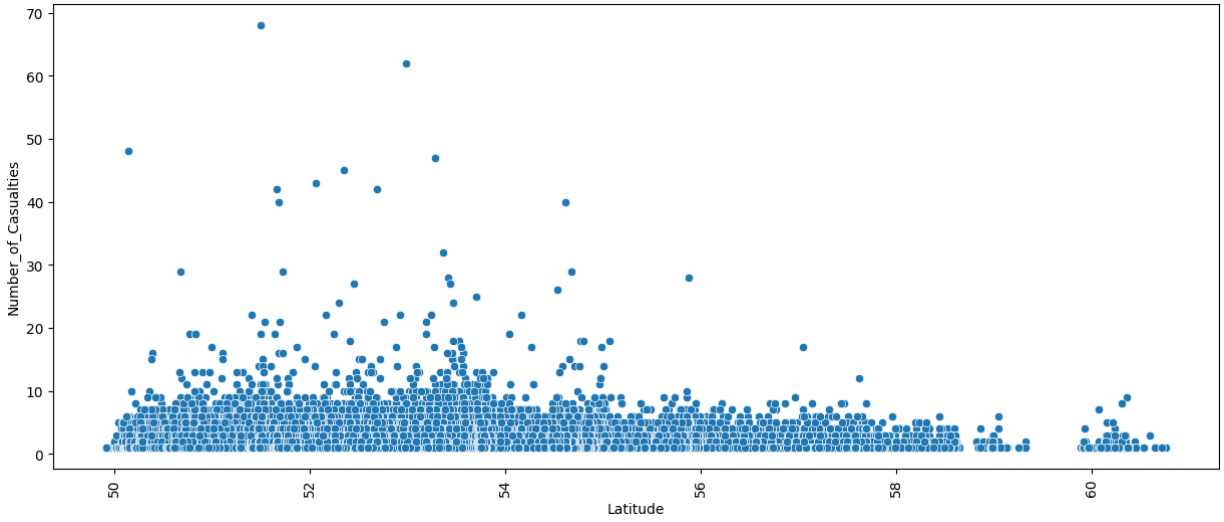


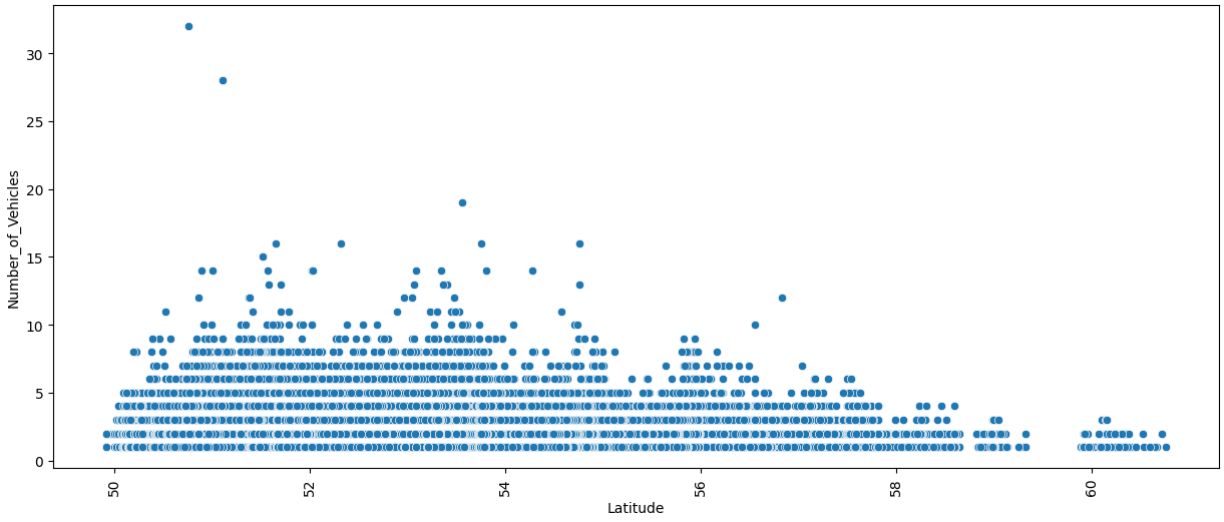


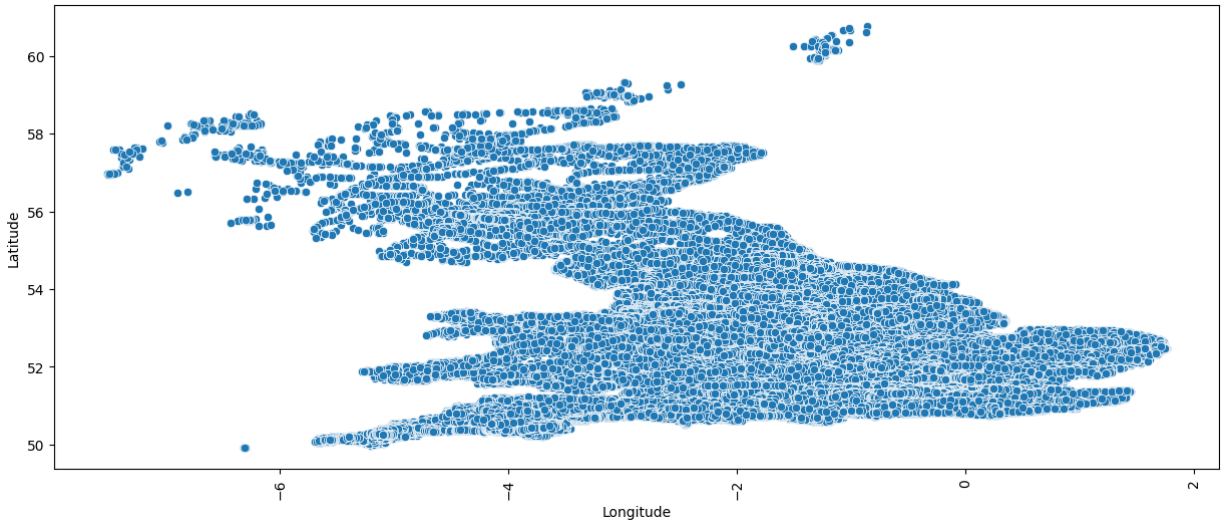


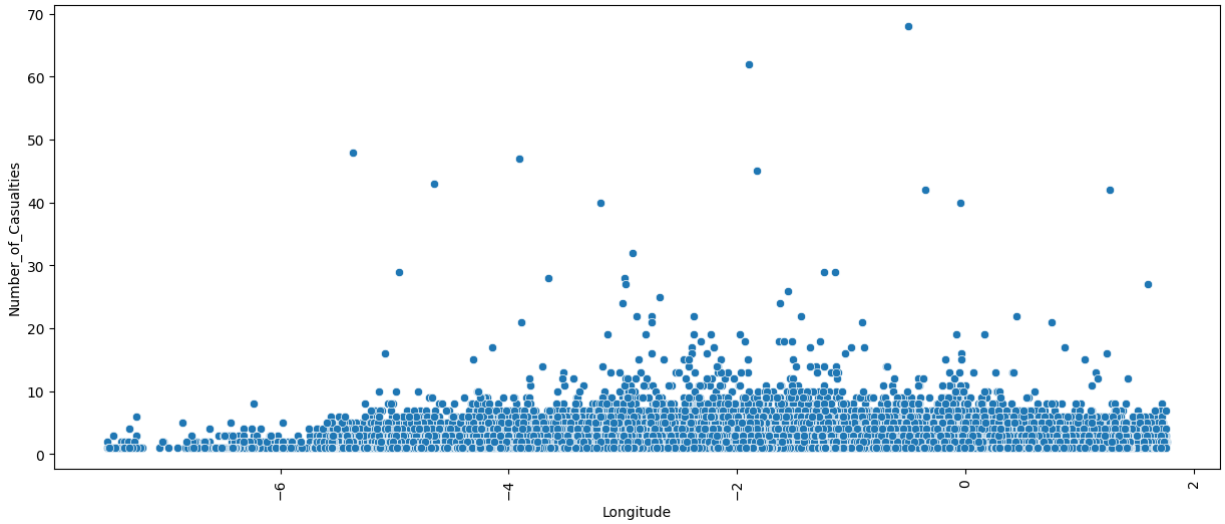
In [40]:
```
for i in continuous:
    for j in continuous:
        if i != j:
            plt.figure(figsize=(15, 6))
            sns.scatterplot(x=i, y=j, data=df, palette='hls')
            plt.xticks(rotation=90)
            plt.show()
```

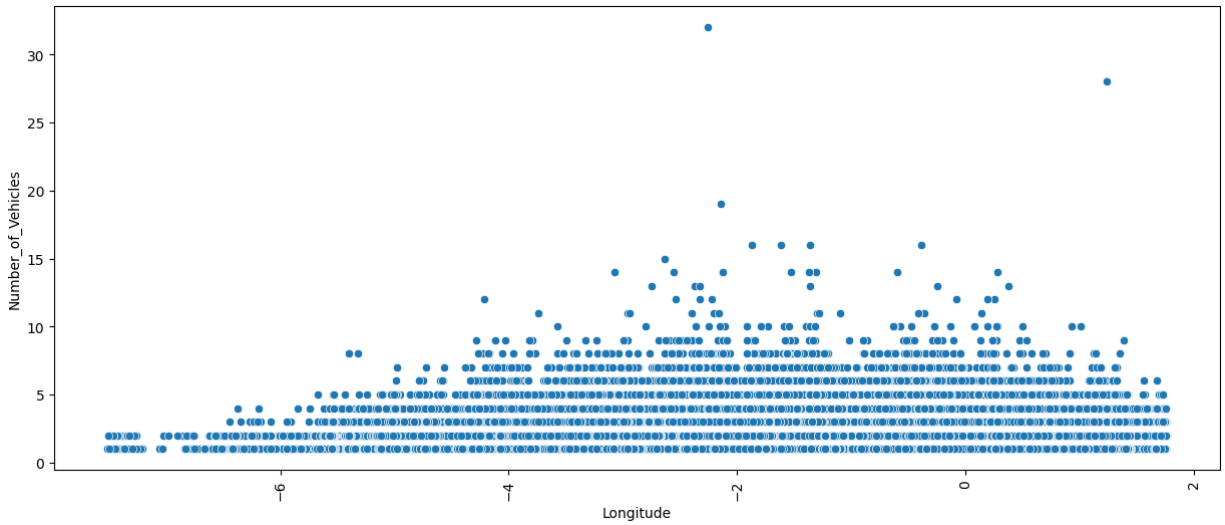


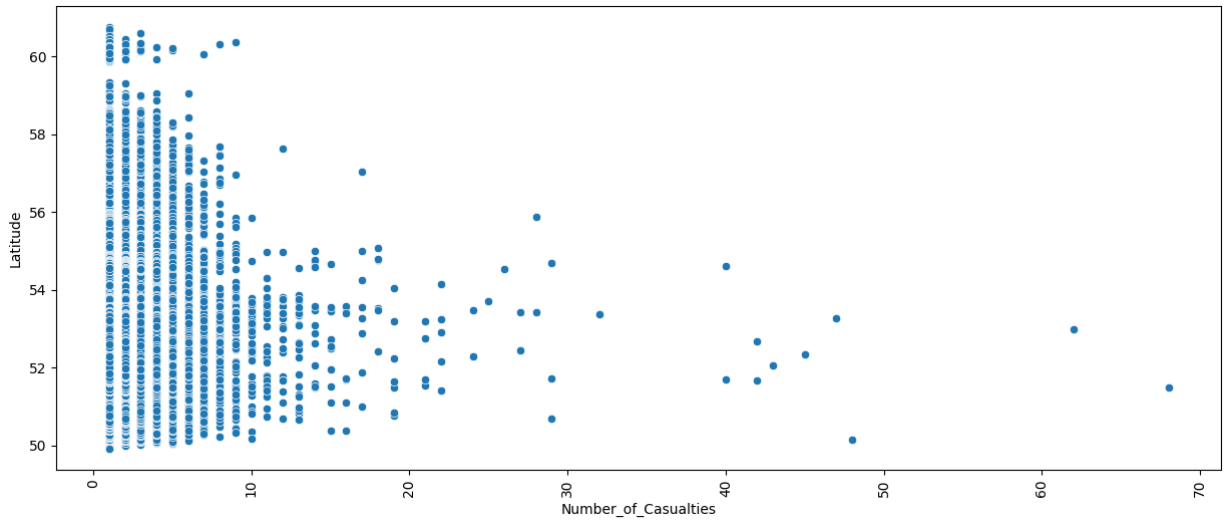


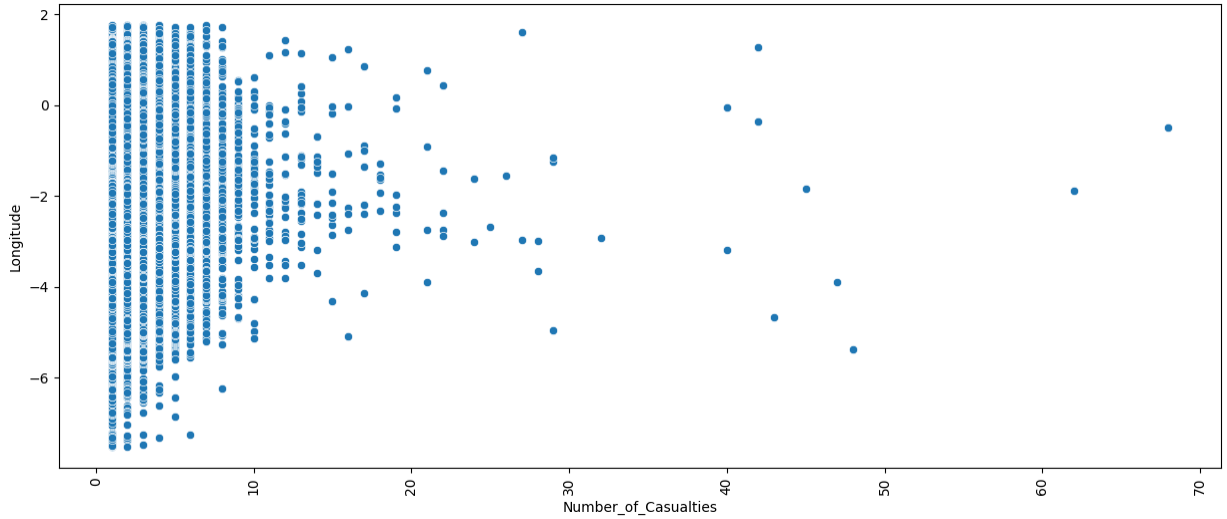


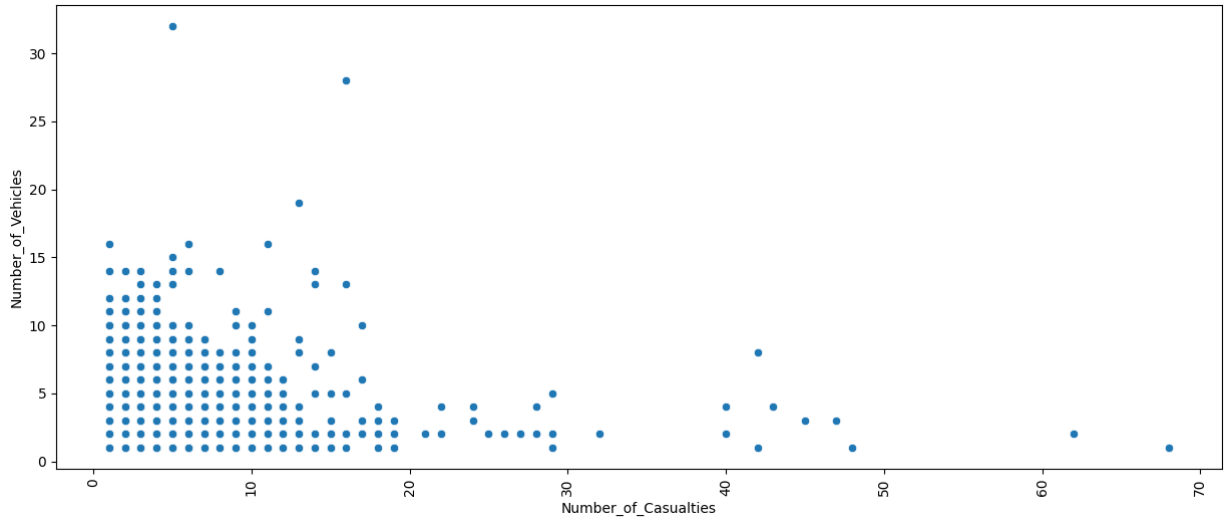


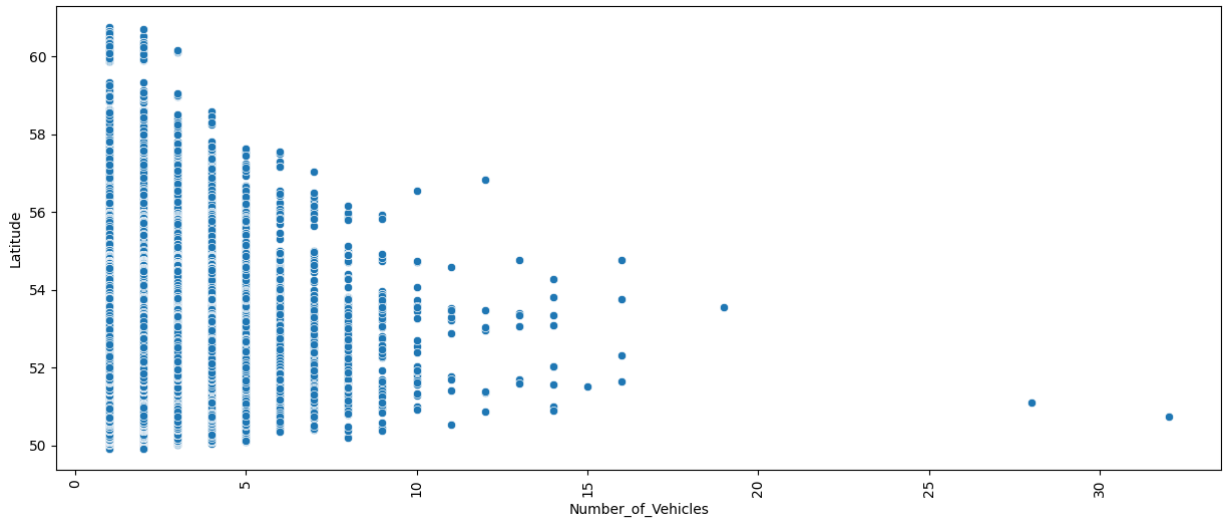


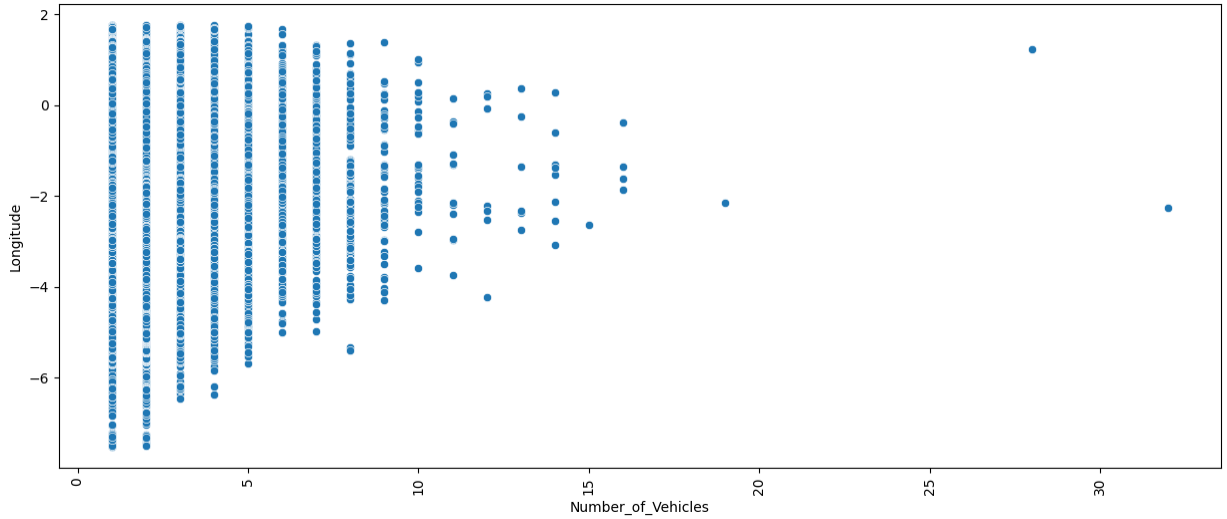


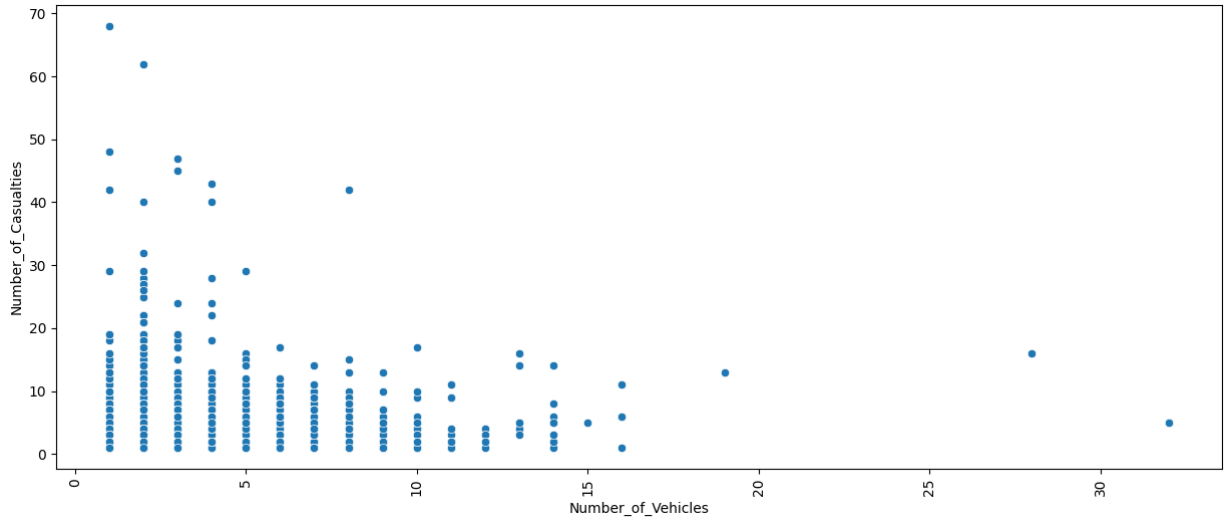




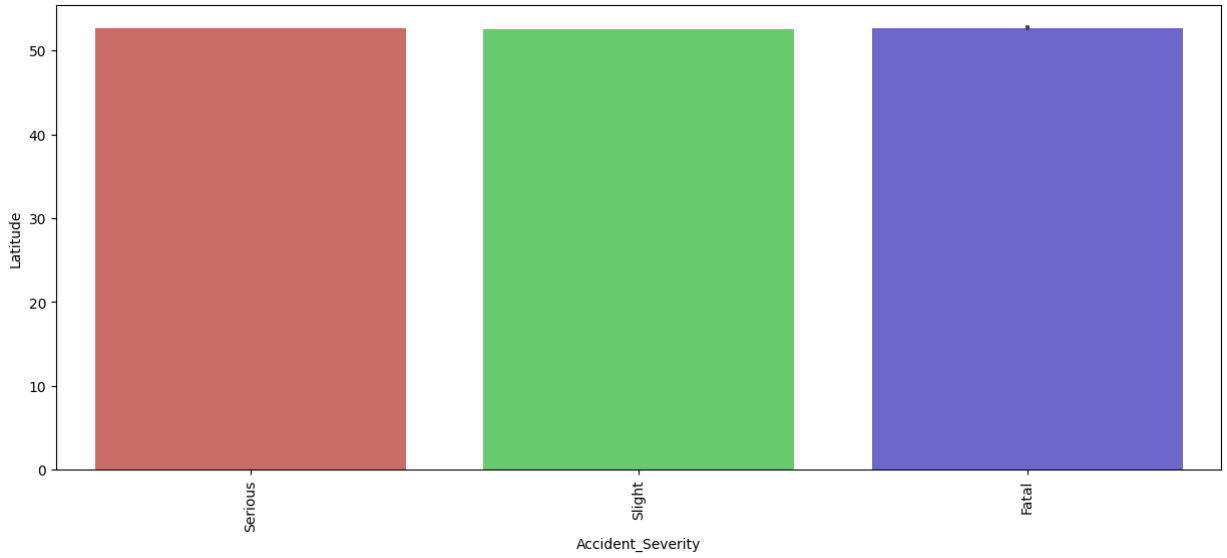


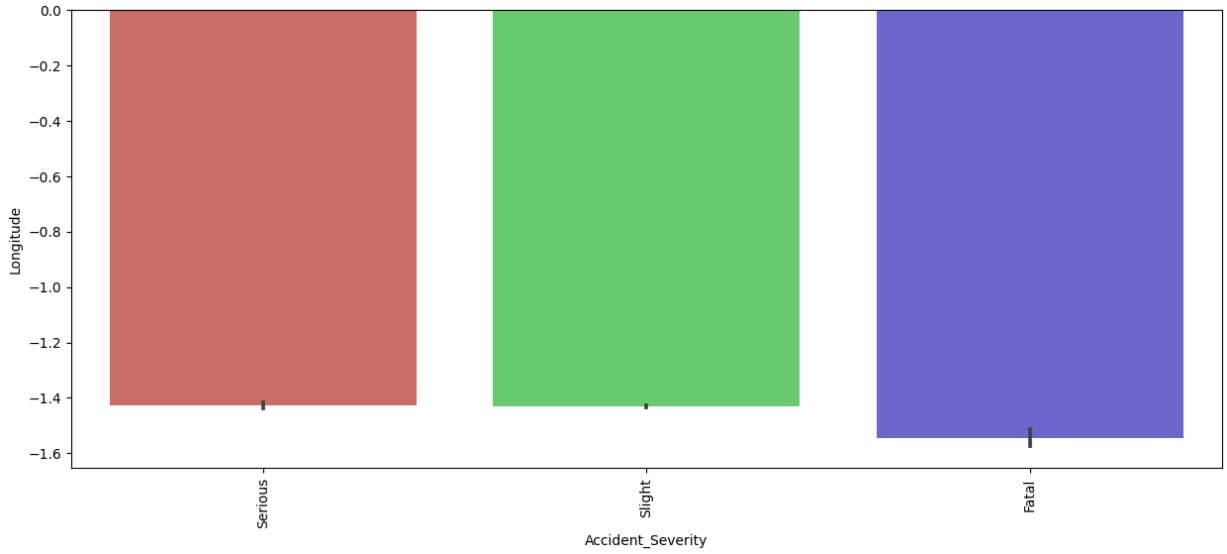


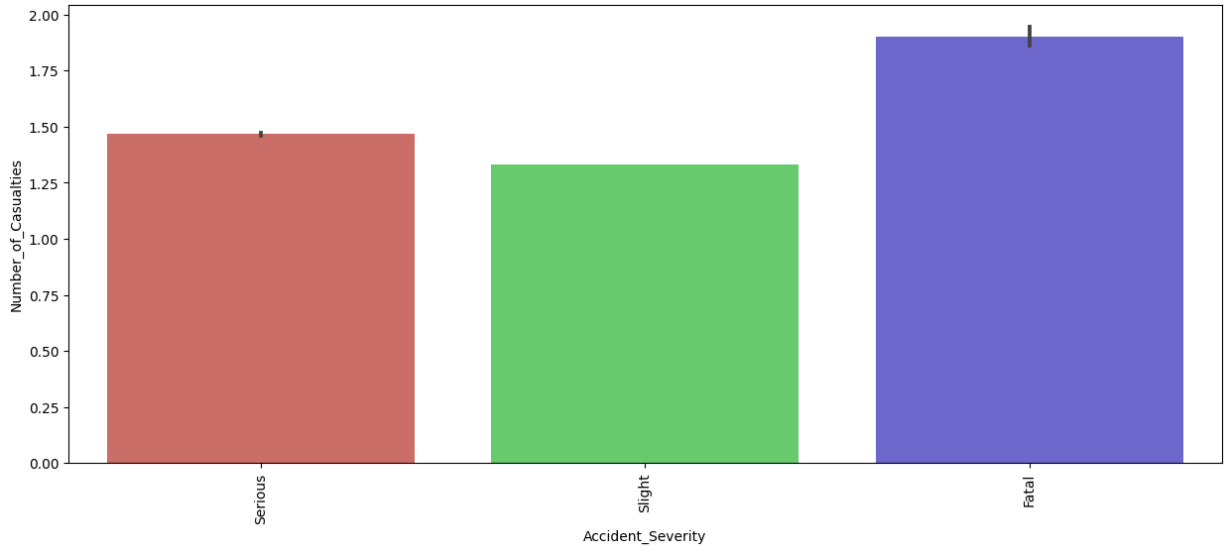


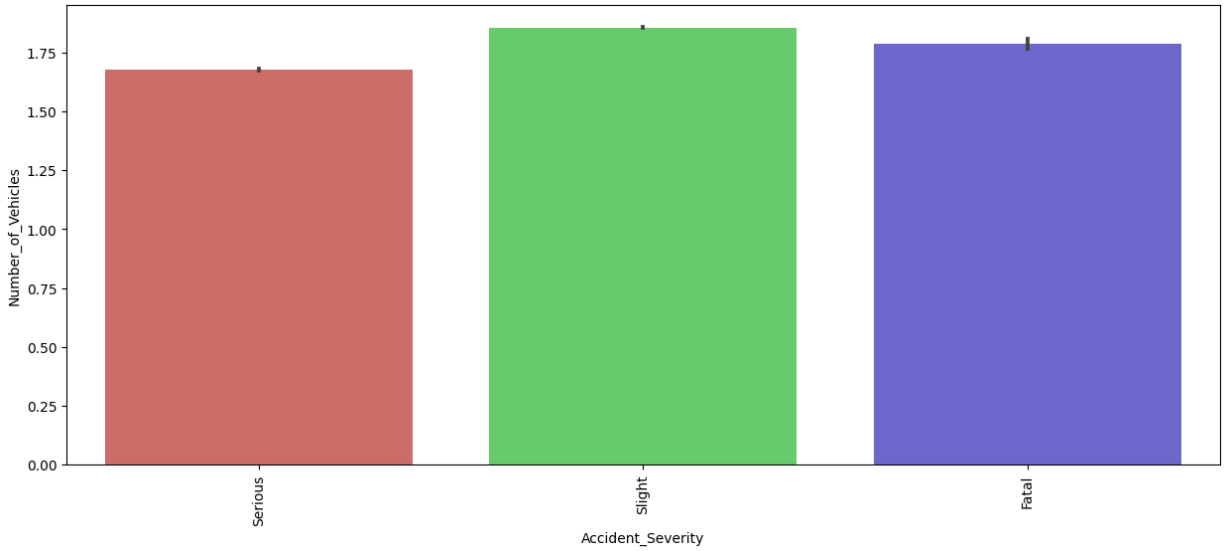


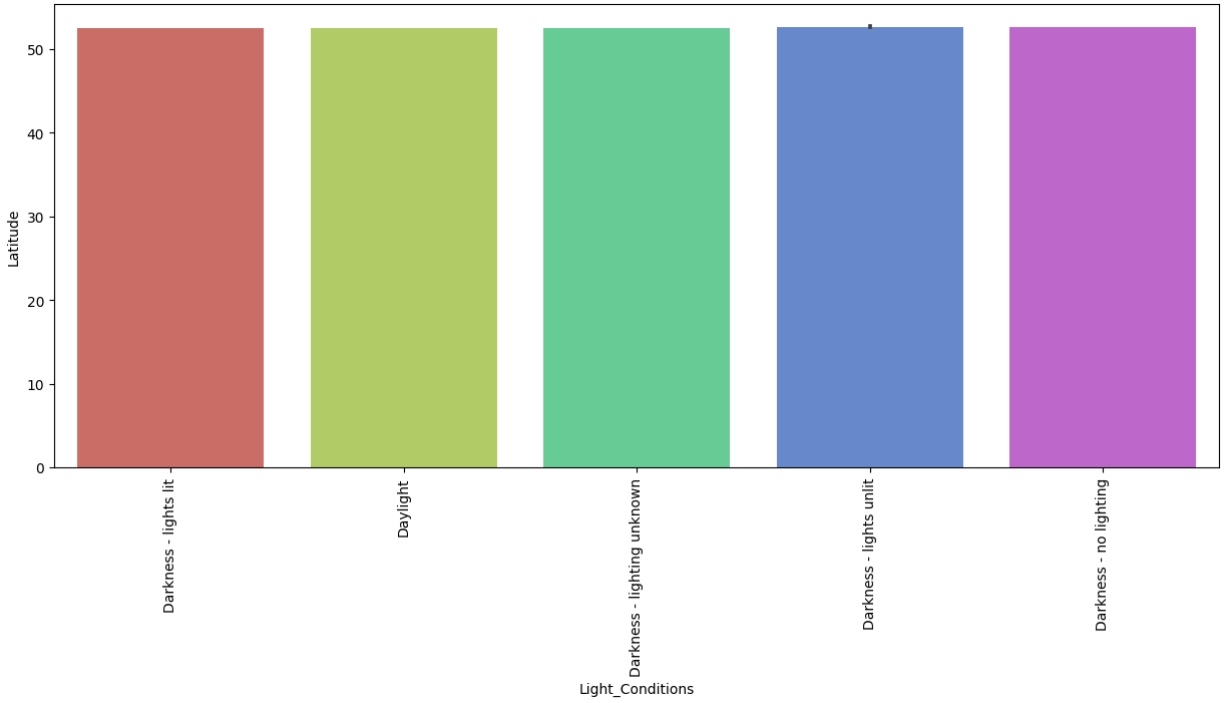
In [41]:
```
for i in categorical:
    for j in continuous:
        plt.figure(figsize=(15,6))
        sns.barplot(x = df[i], y = df[j], data = df,palette = 'hls')
        plt.xticks(rotation = 90)
        plt.show()
```

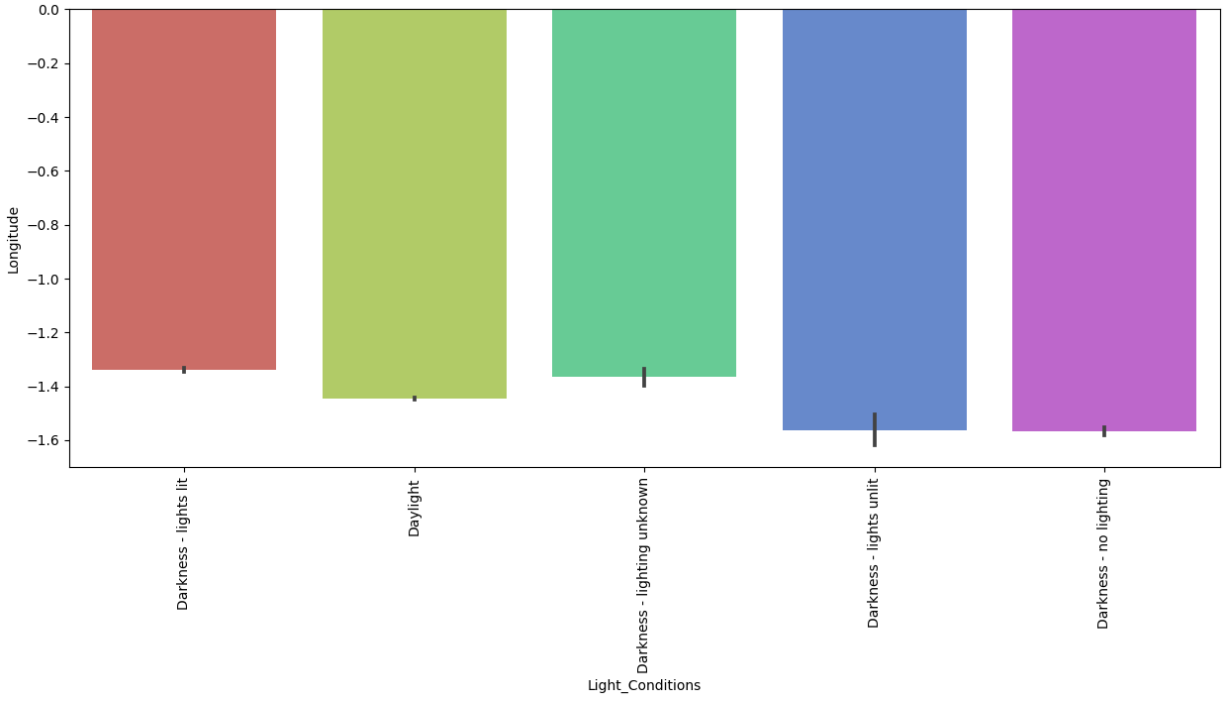


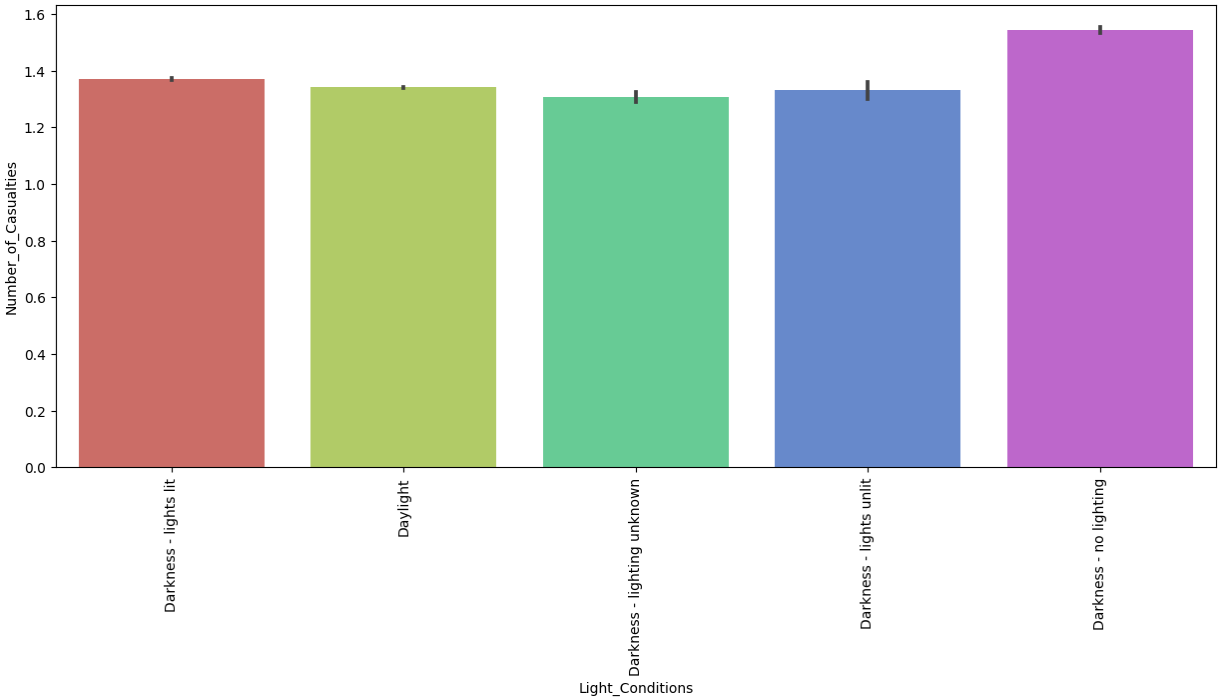


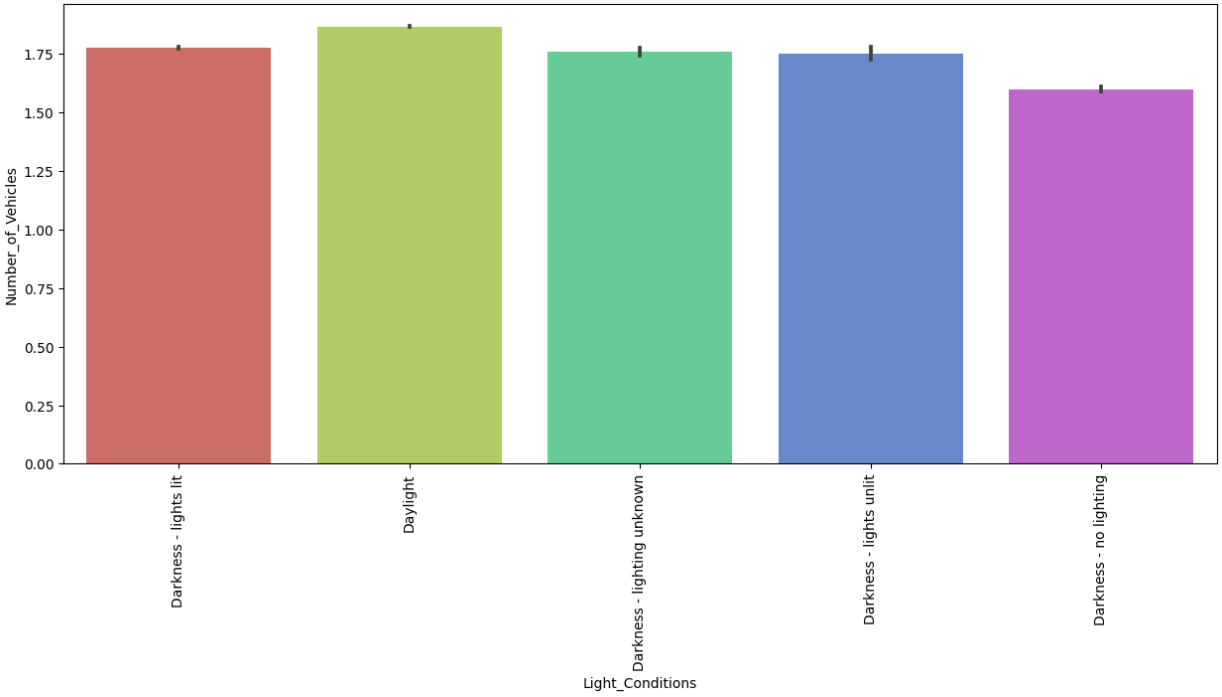


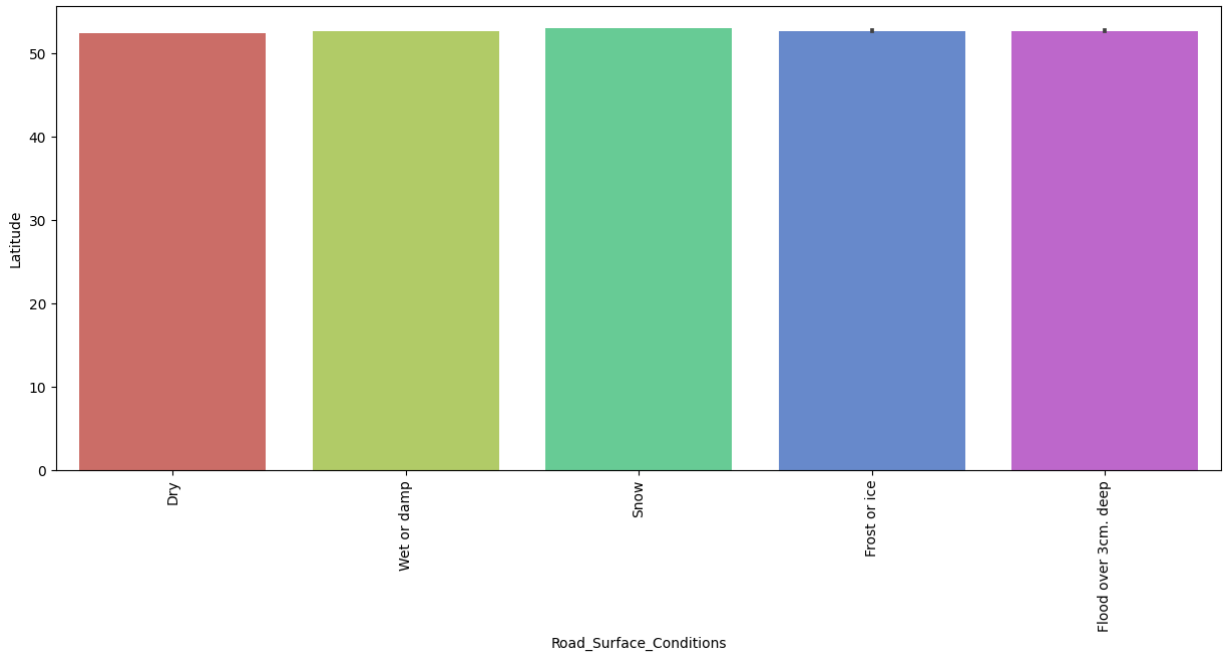


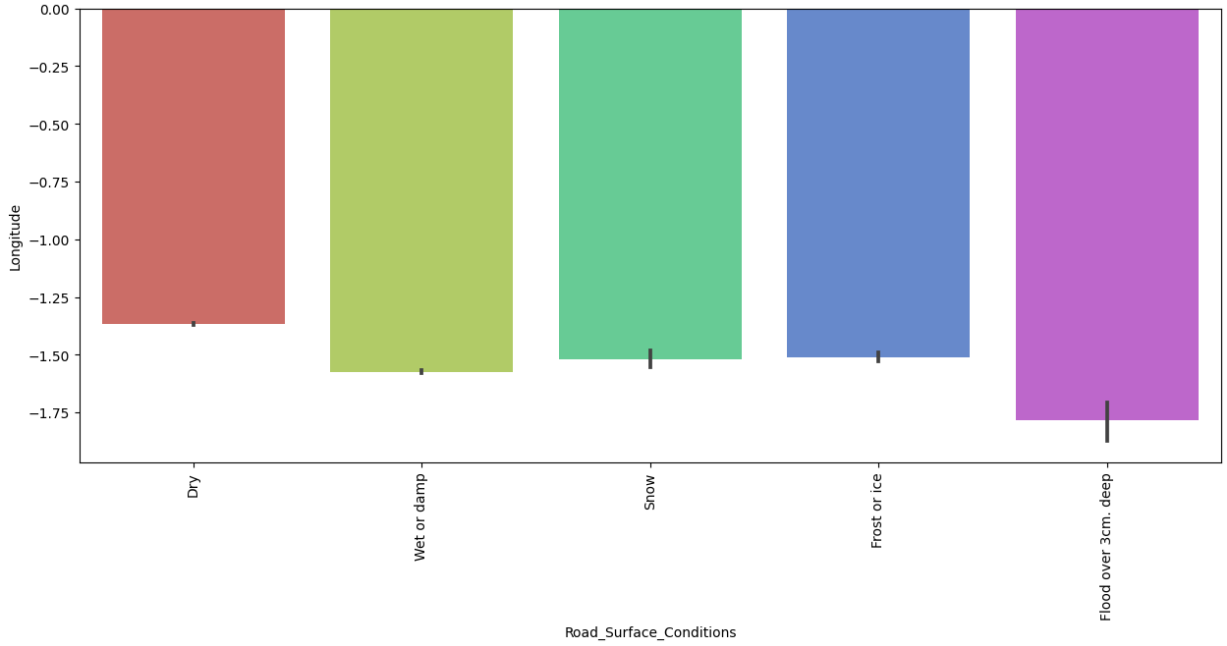


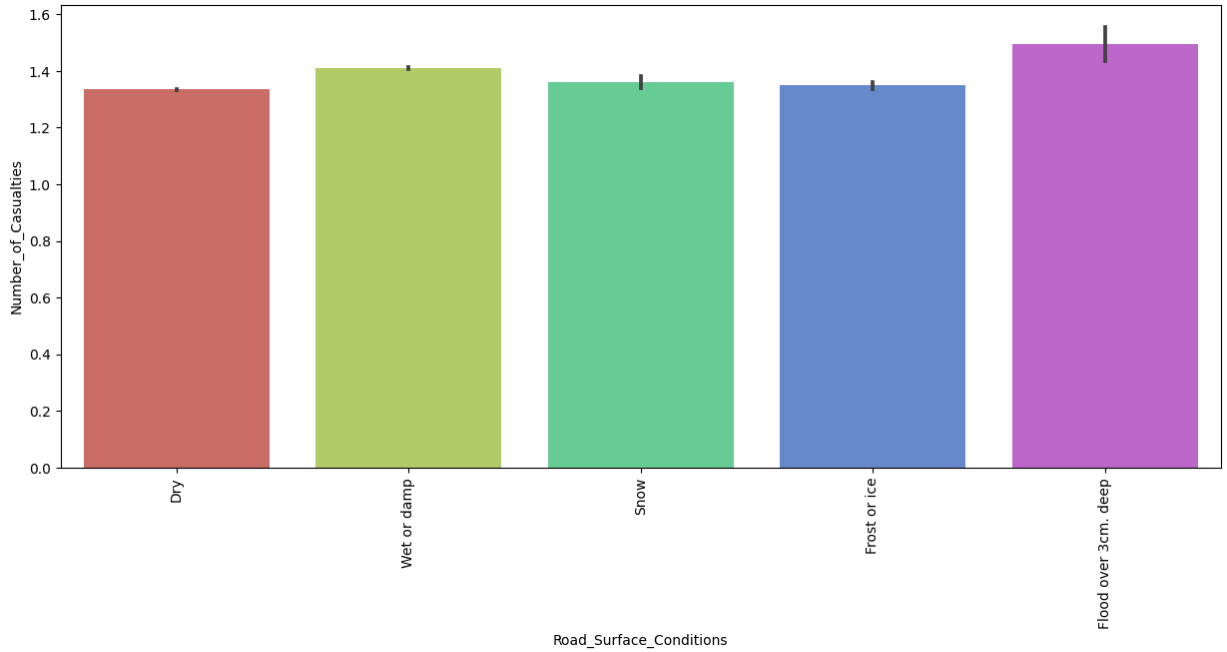


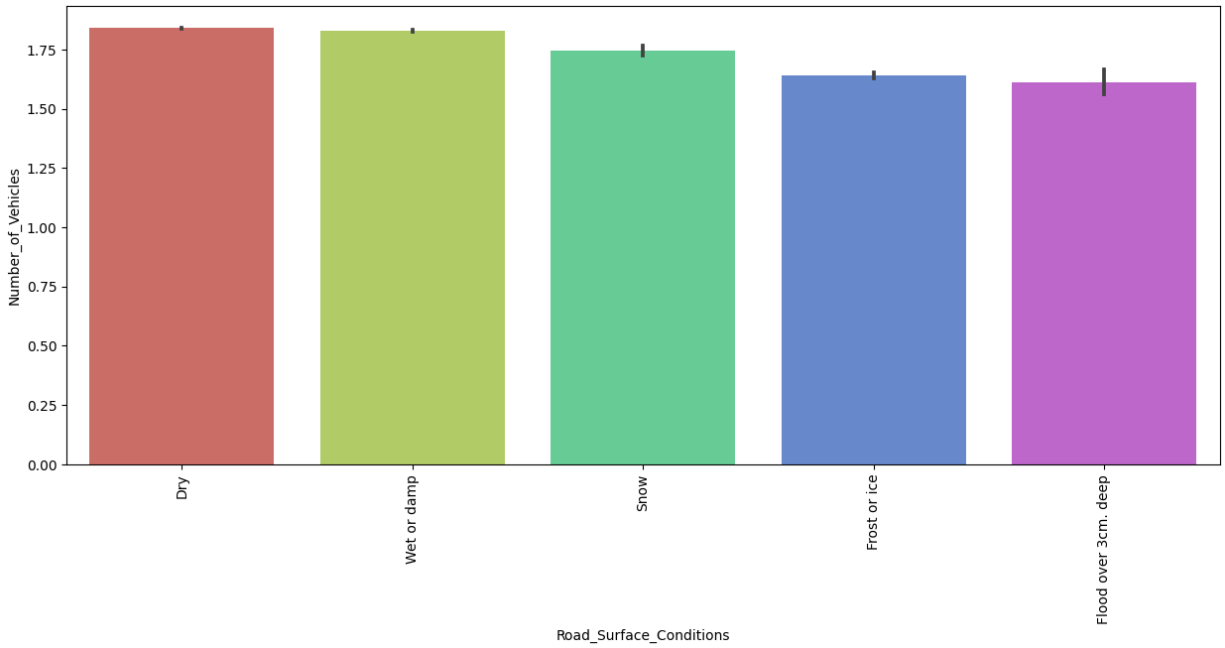


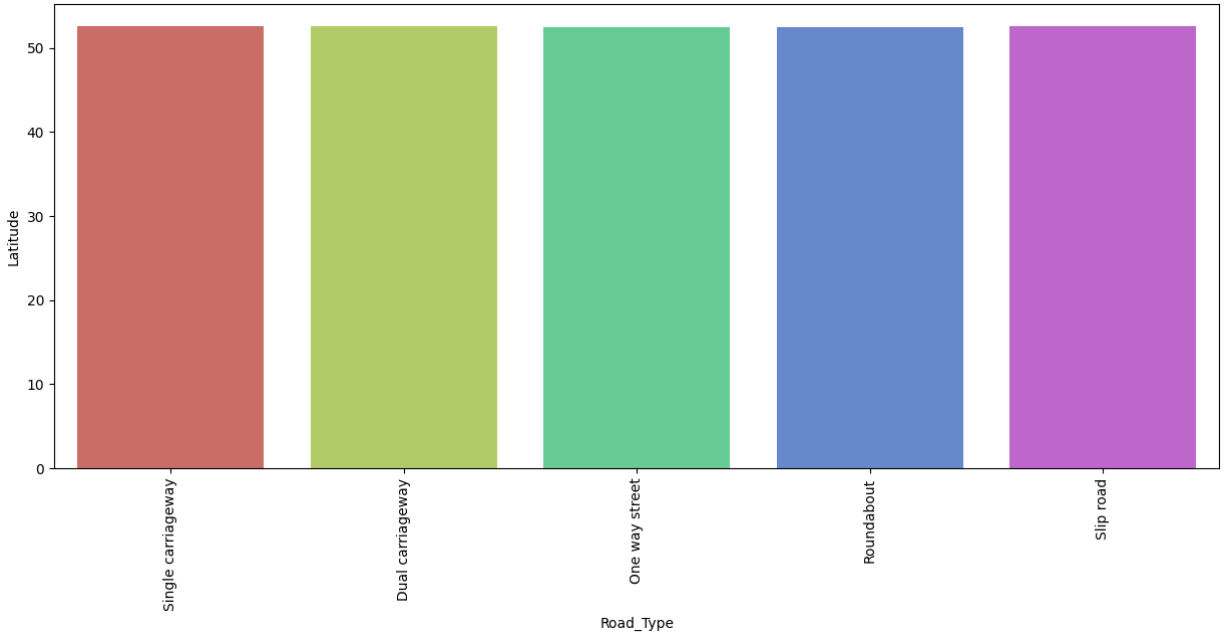


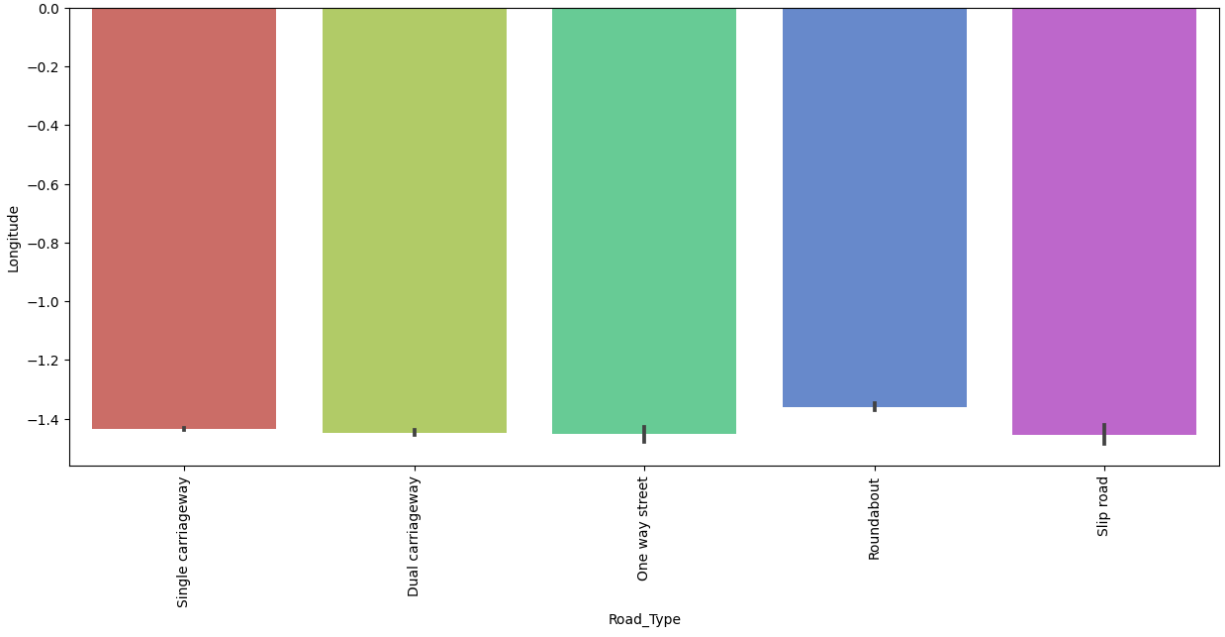


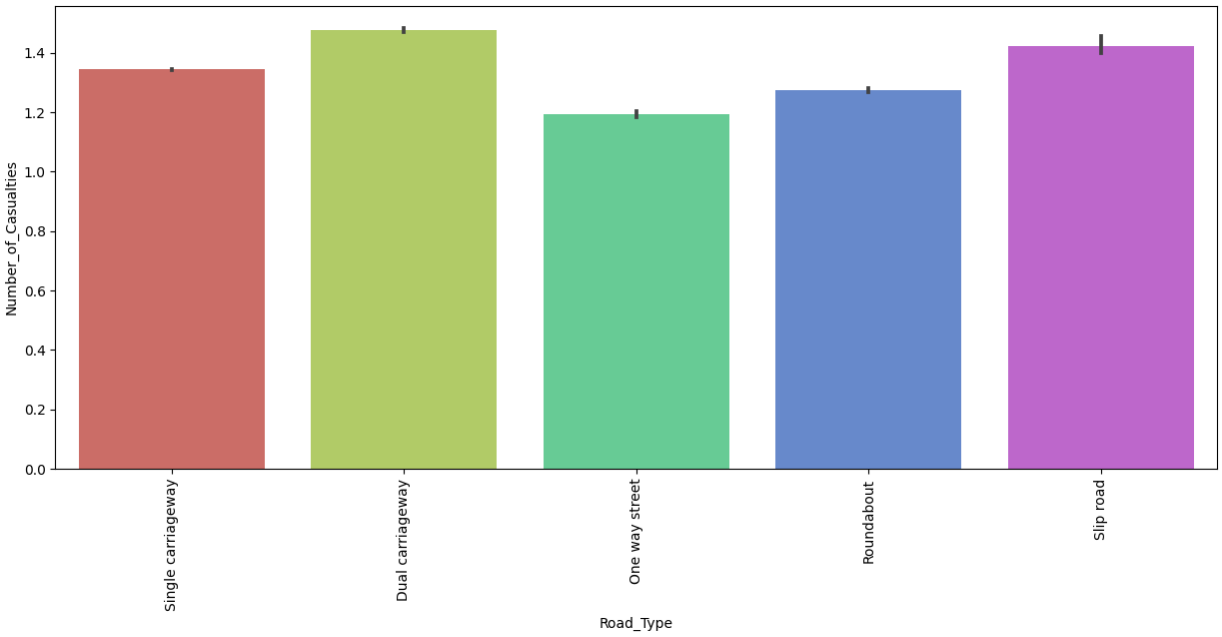


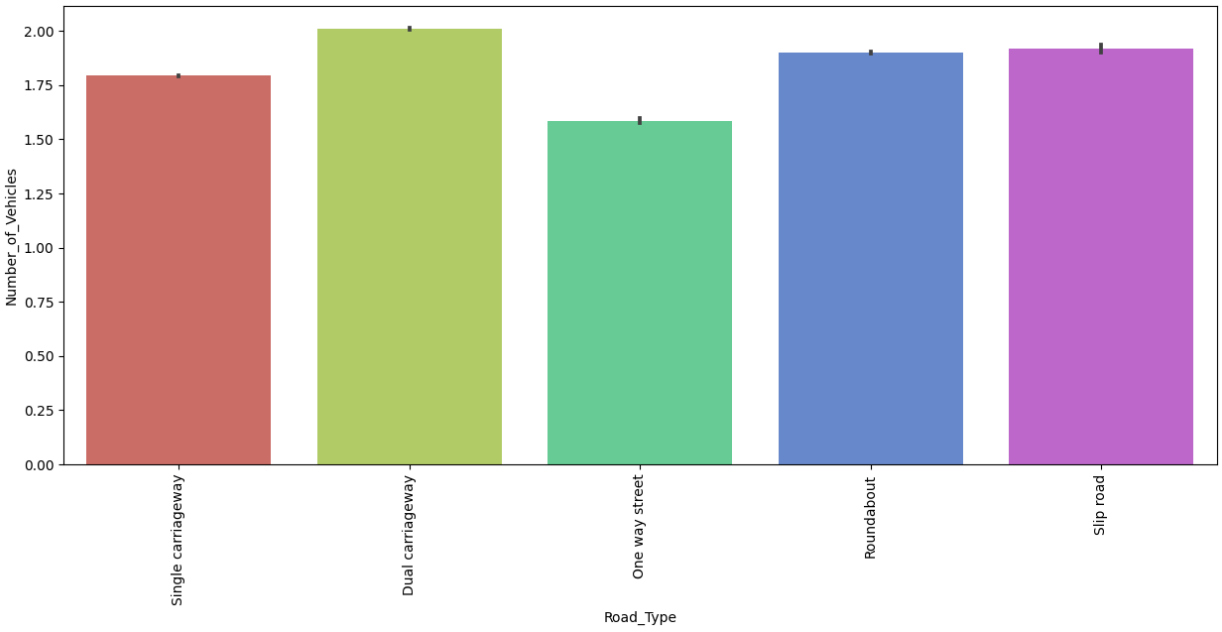


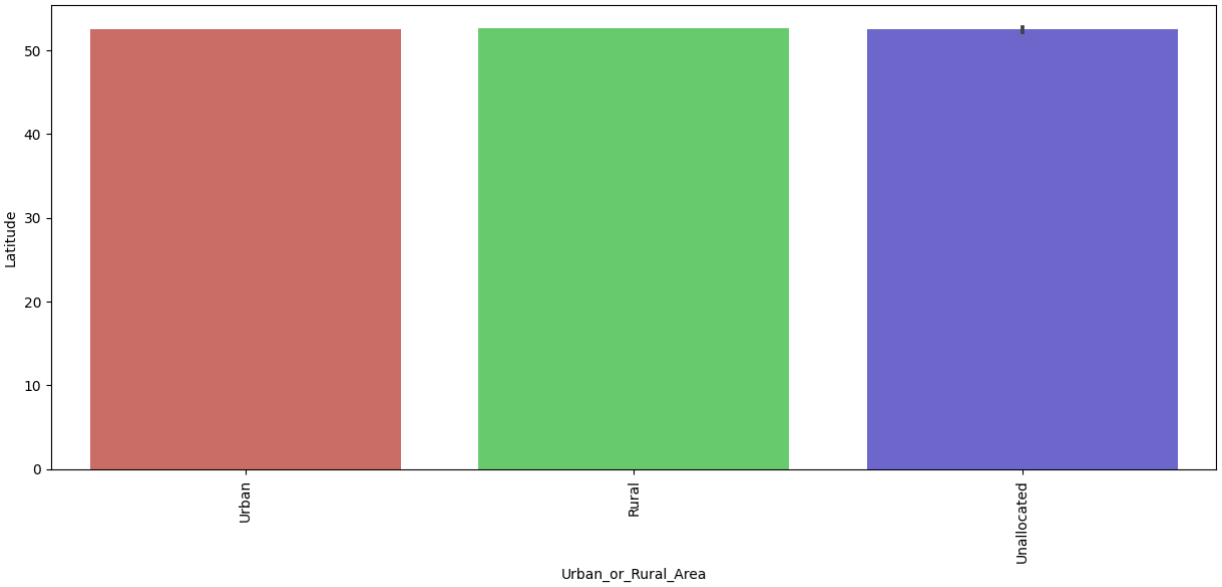


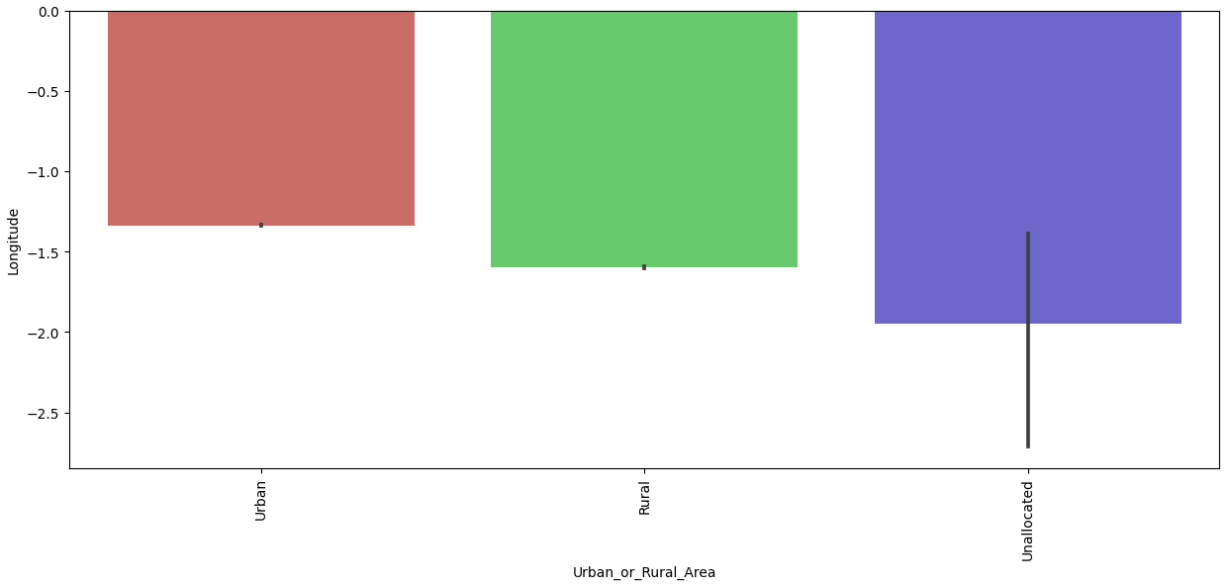


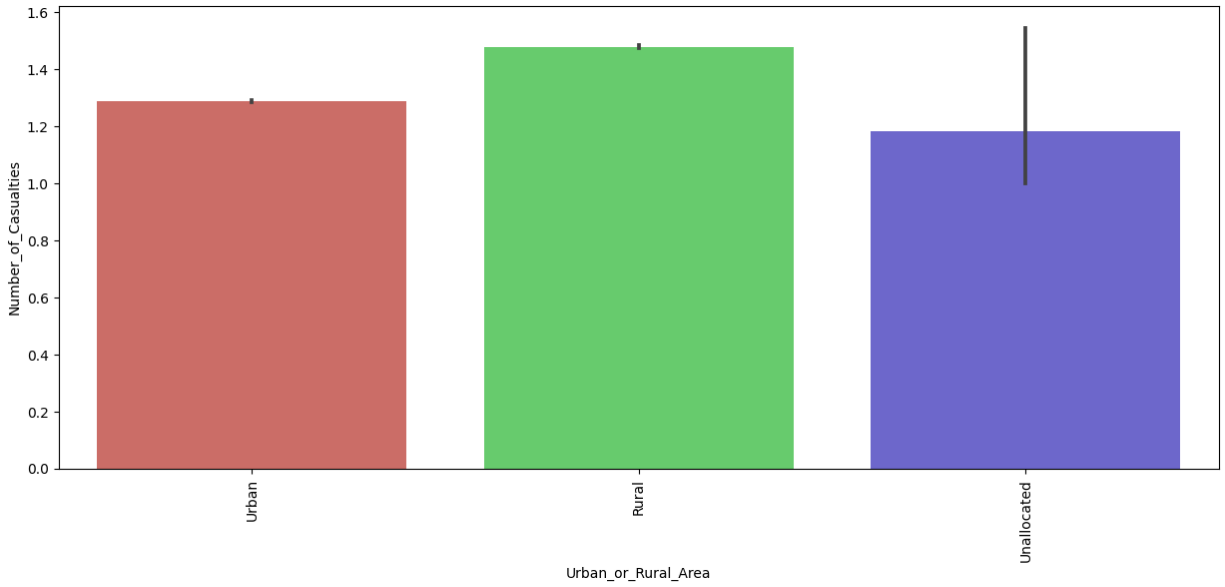


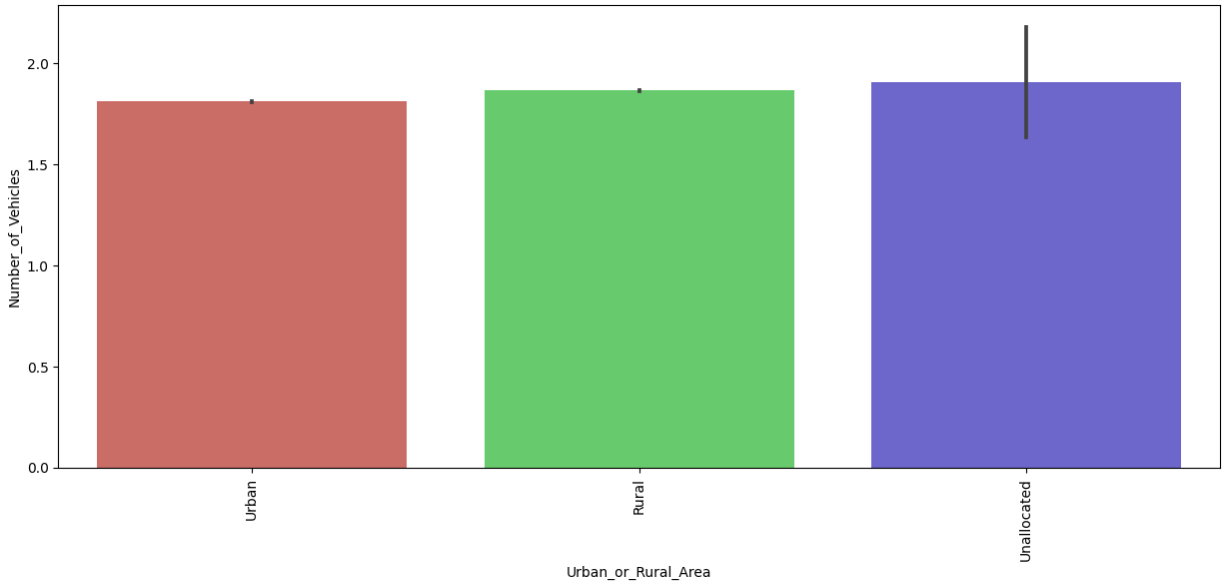


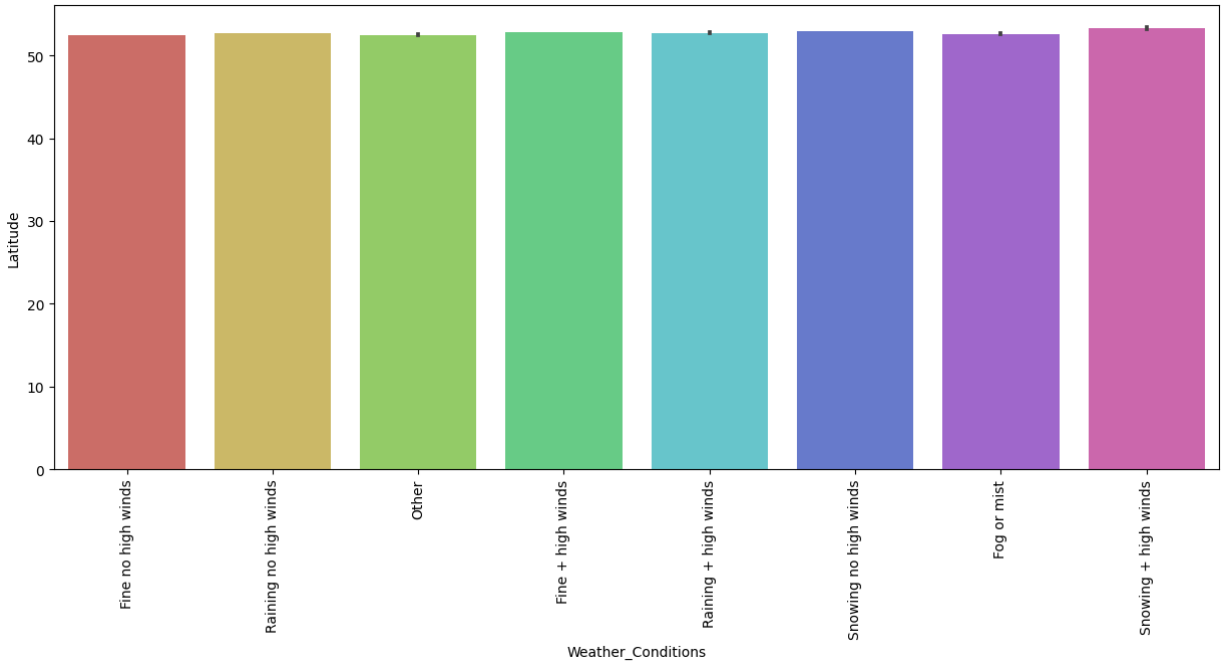


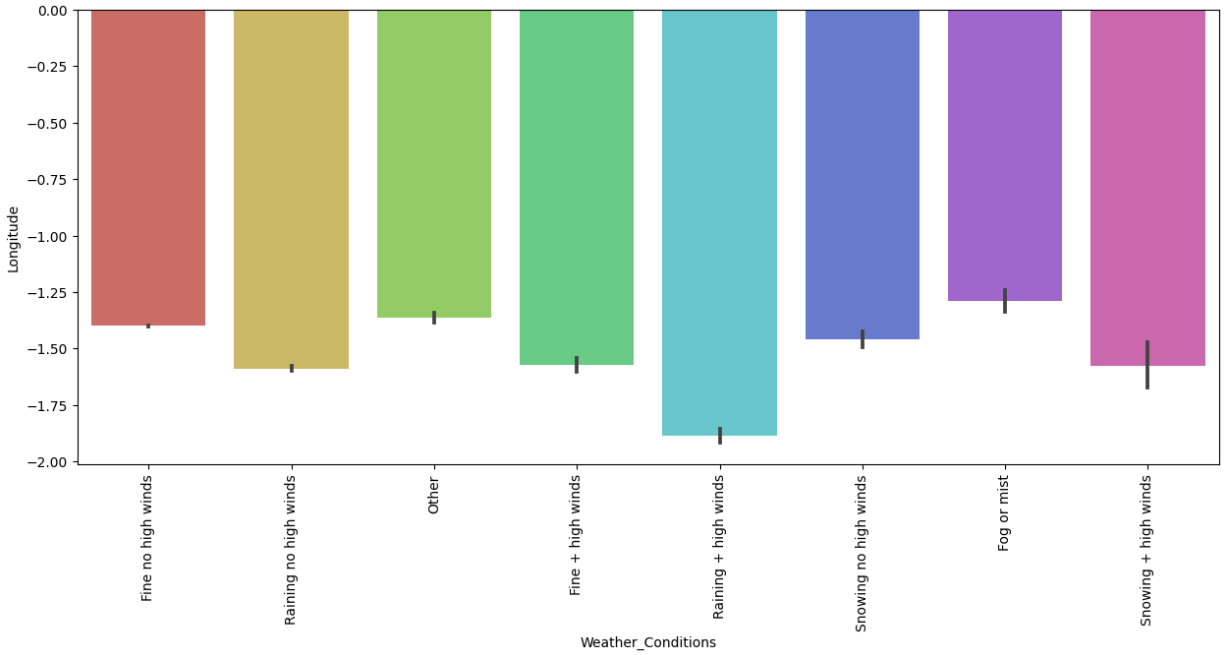


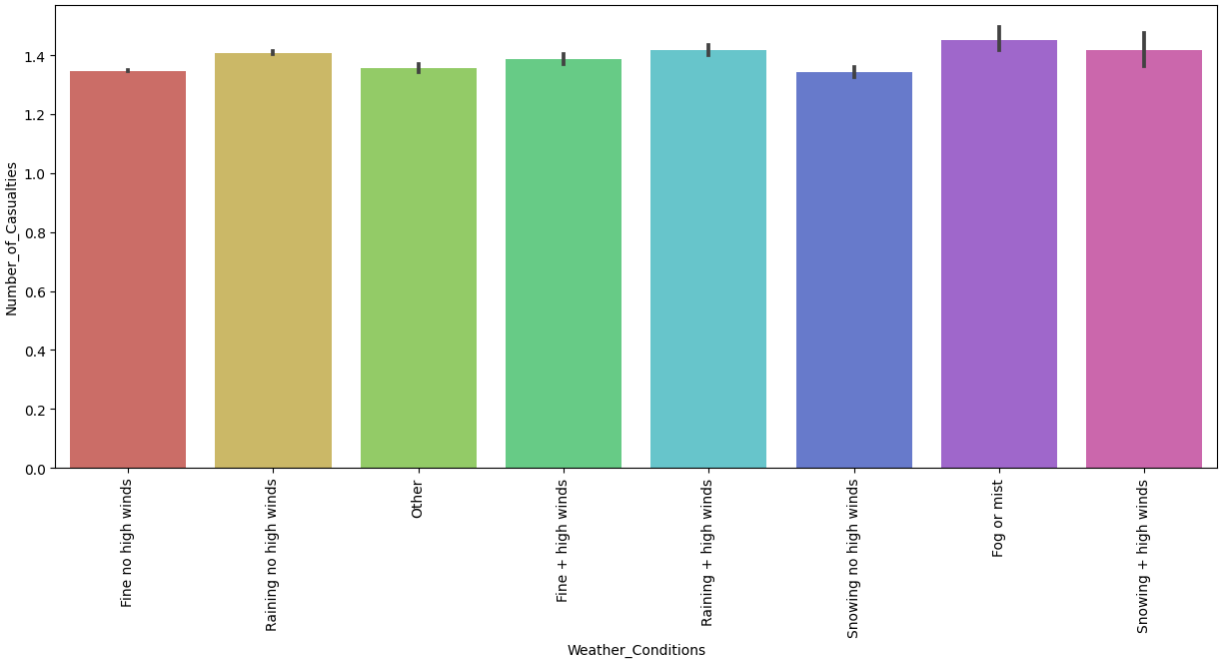


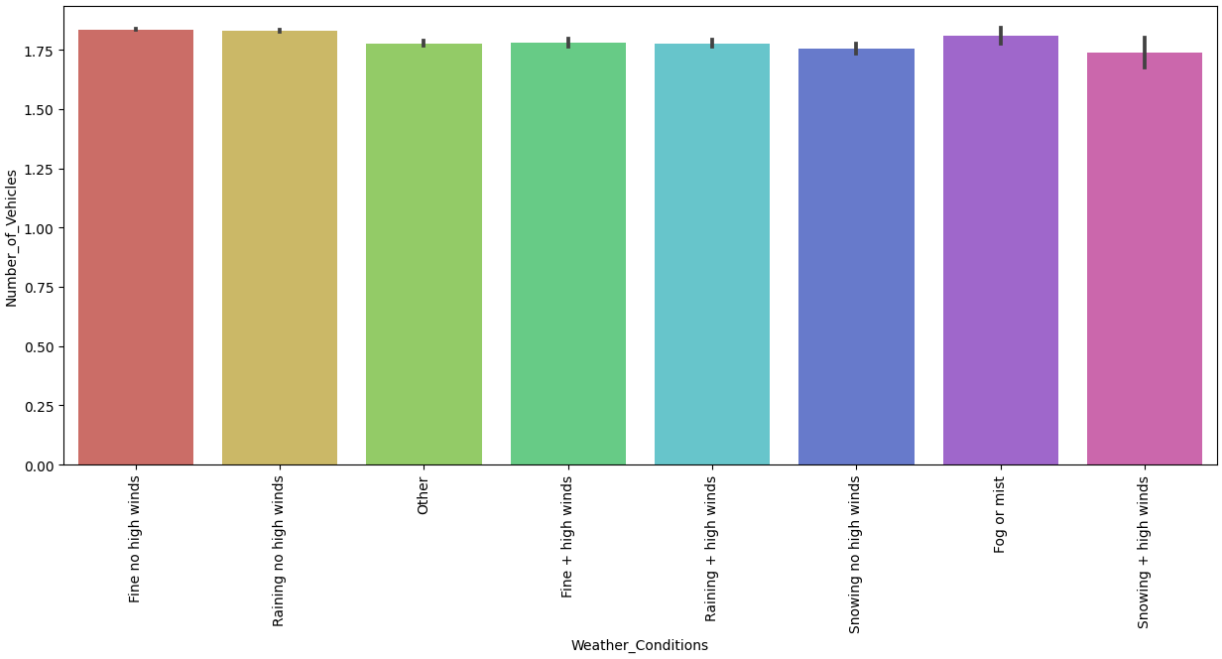




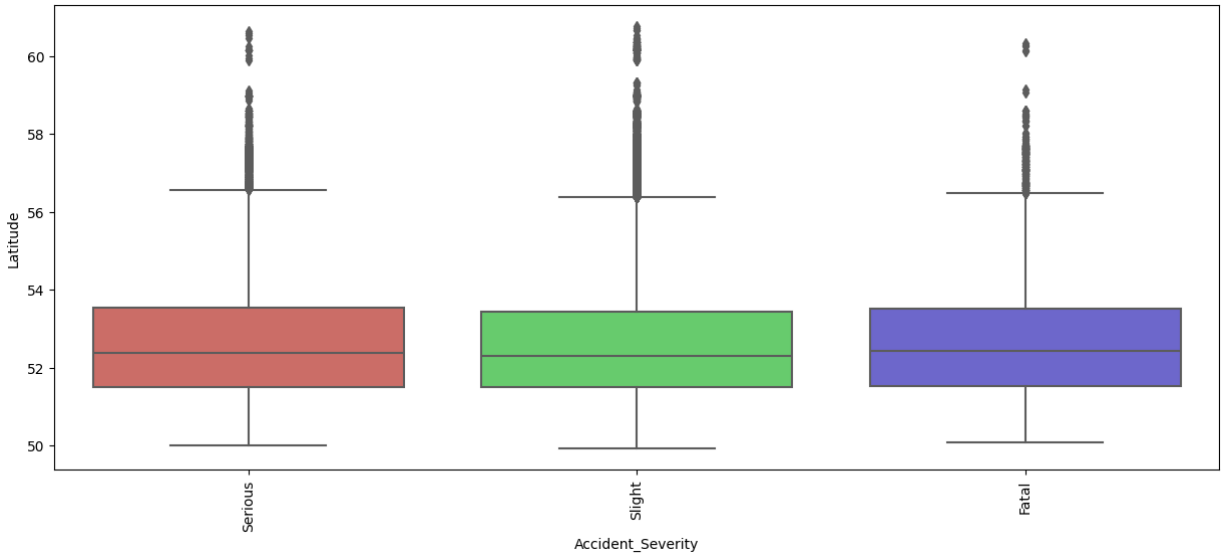


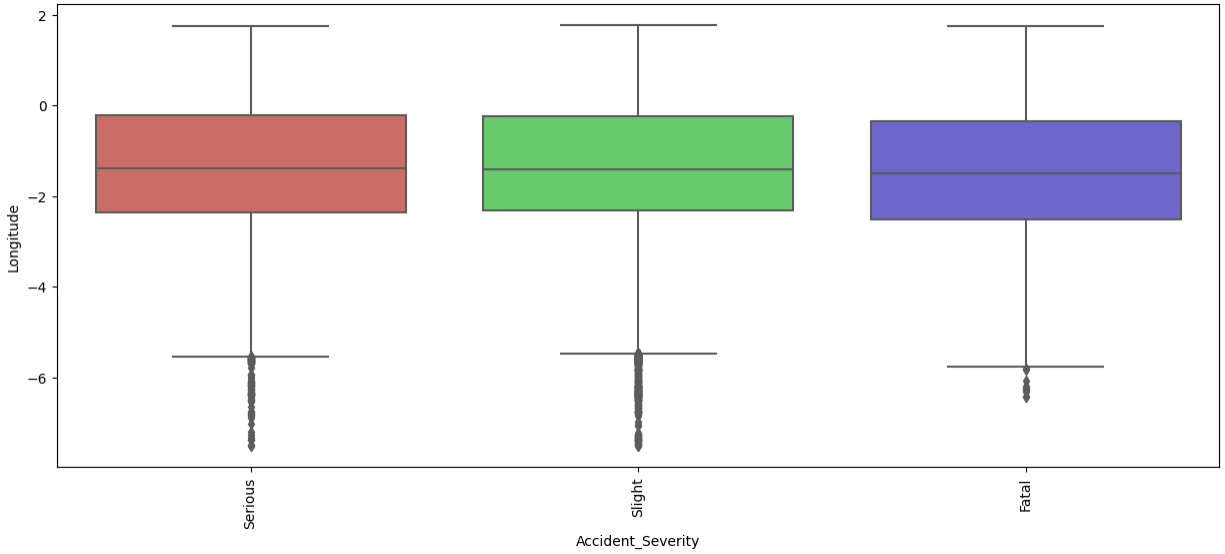


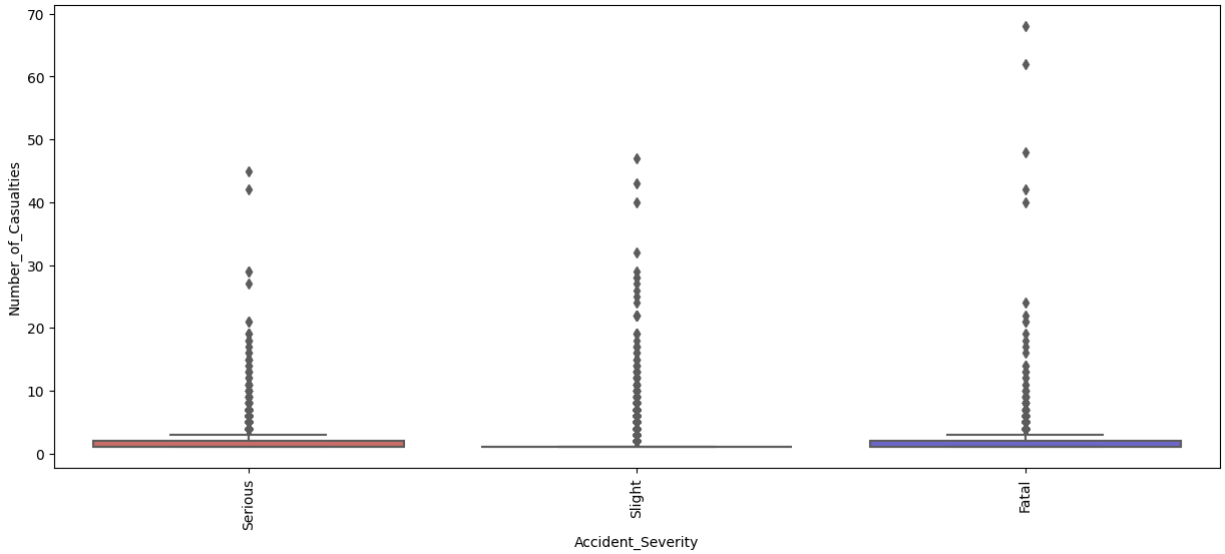


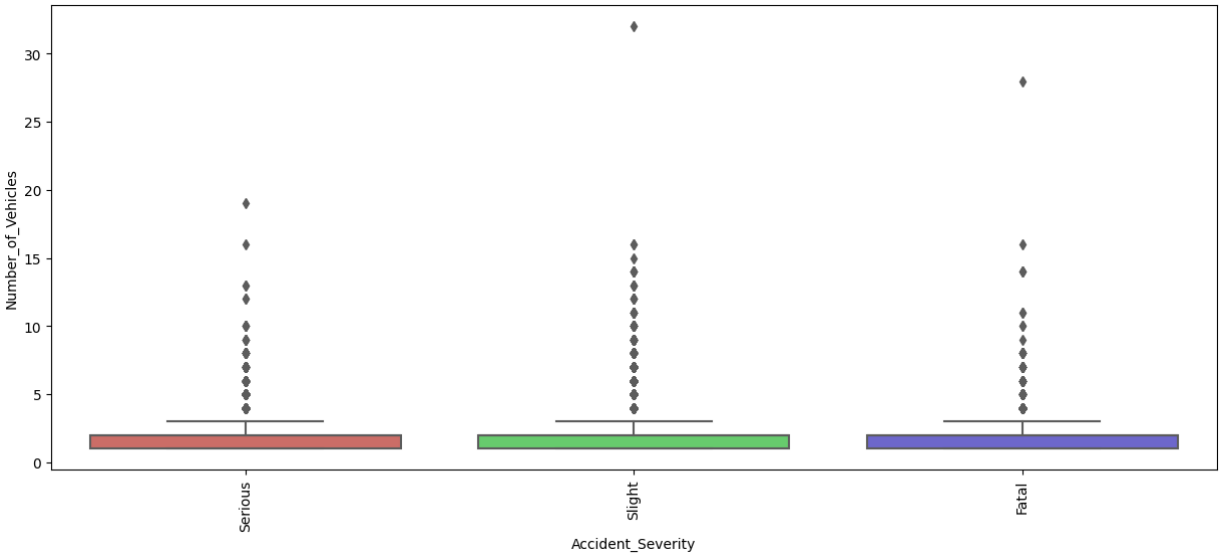


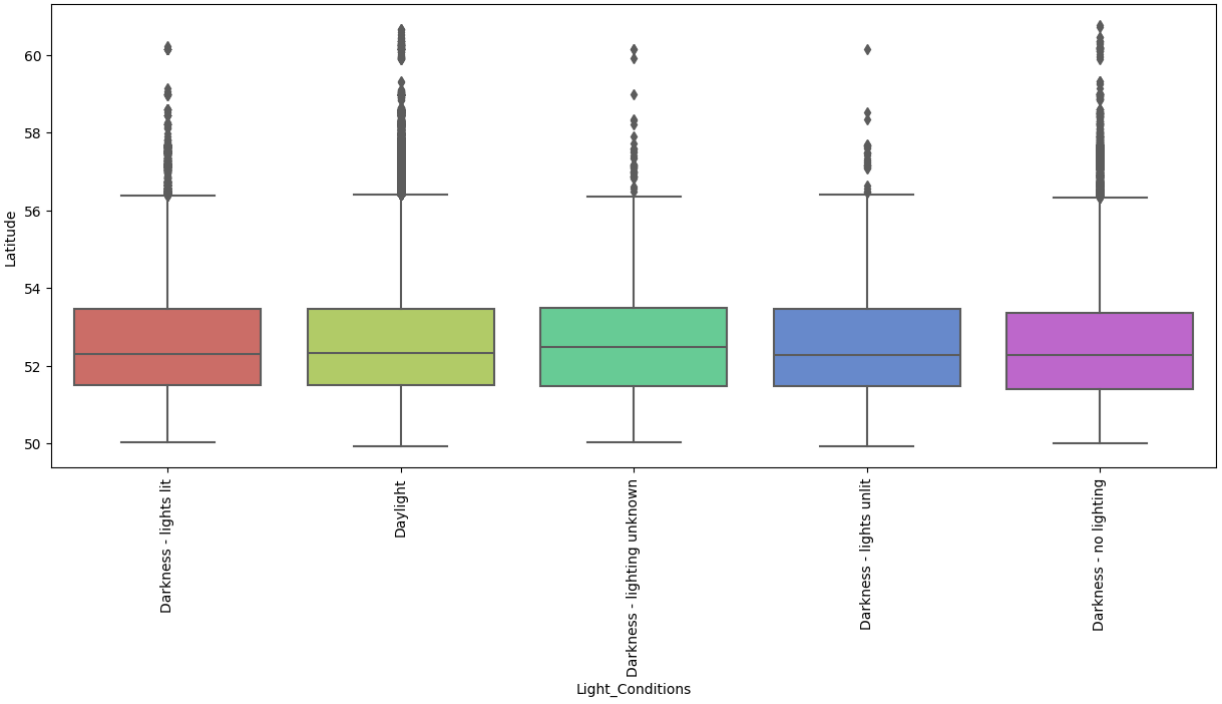
In [42]:
```
for i in categorical:
    for j in continuous:
        plt.figure(figsize=(15,6))
        sns.boxplot(x = df[i], y = df[j], data = df, palette = 'hls')
        plt.xticks(rotation = 90)
        plt.show()
```

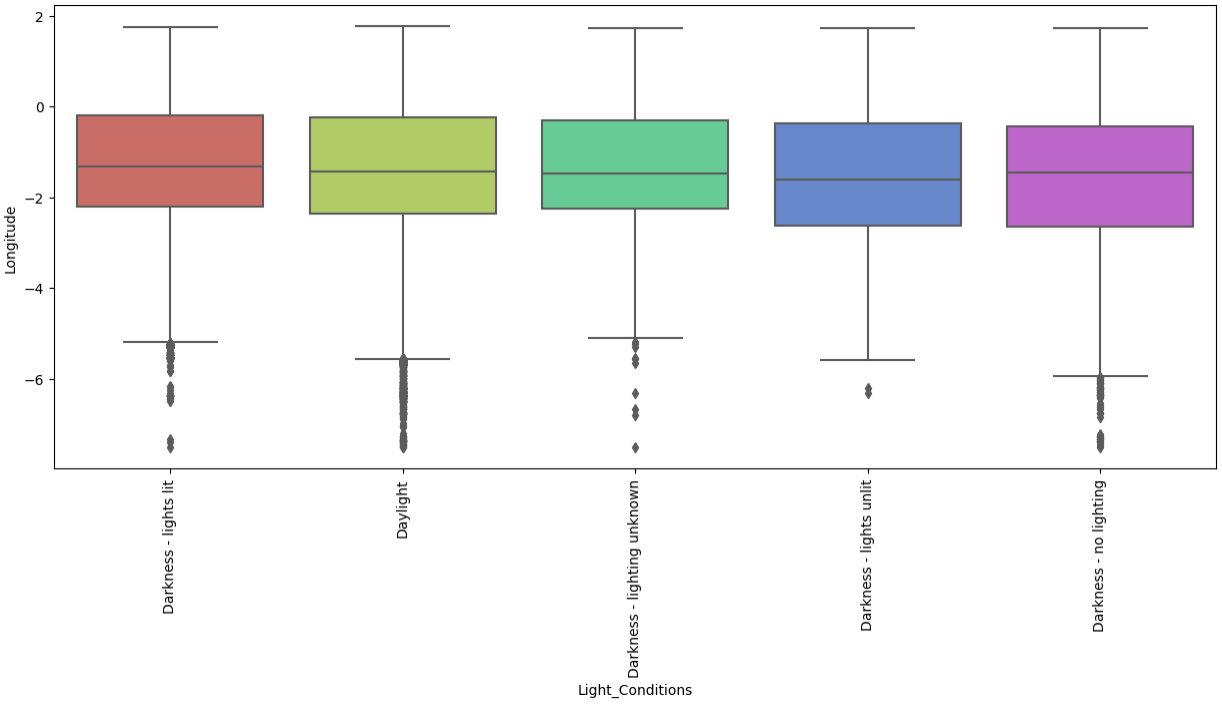


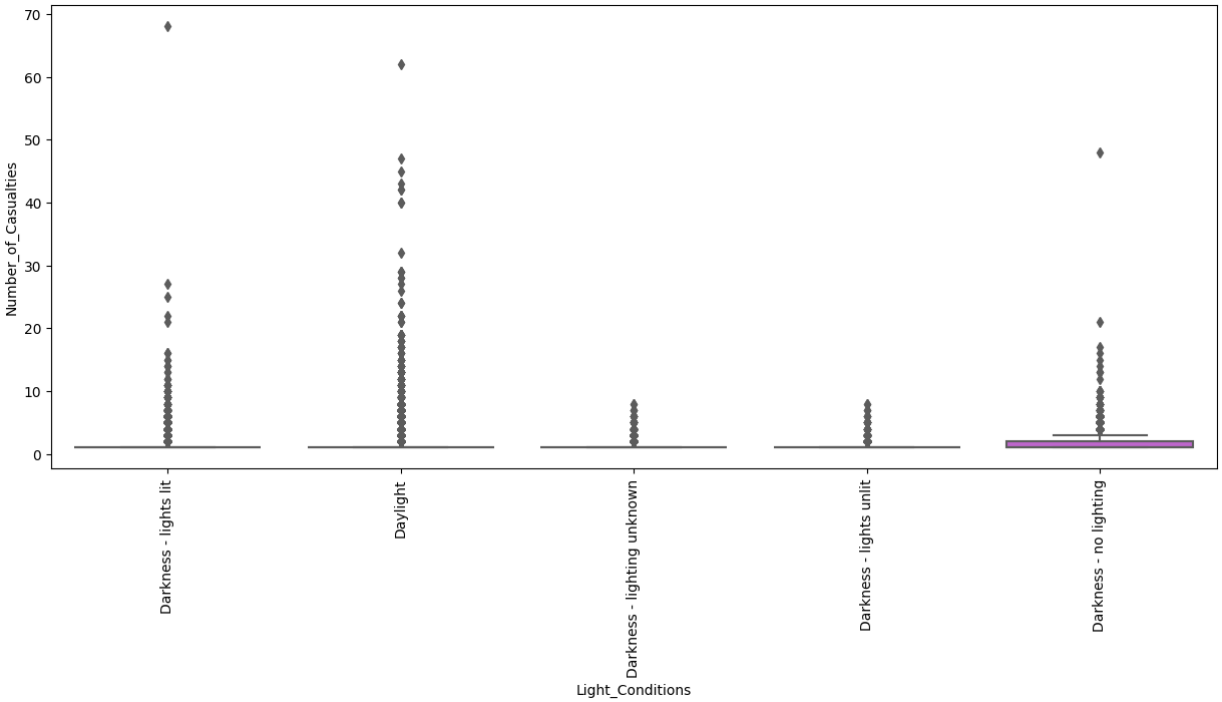


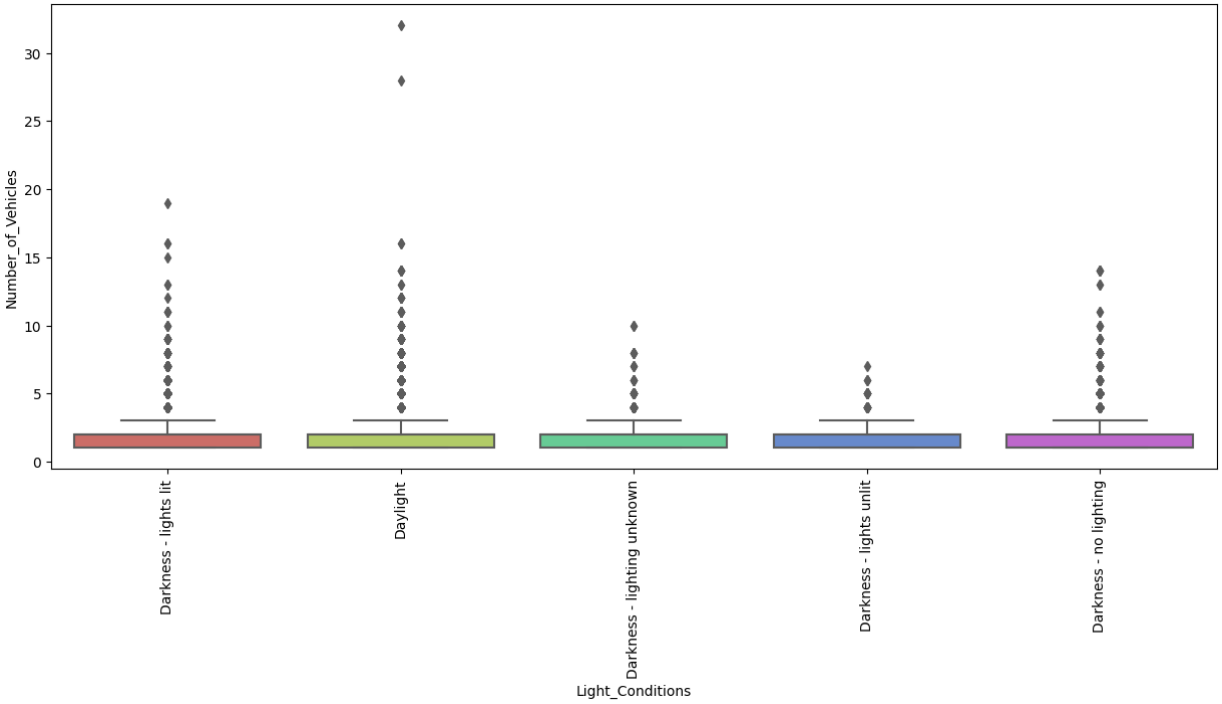


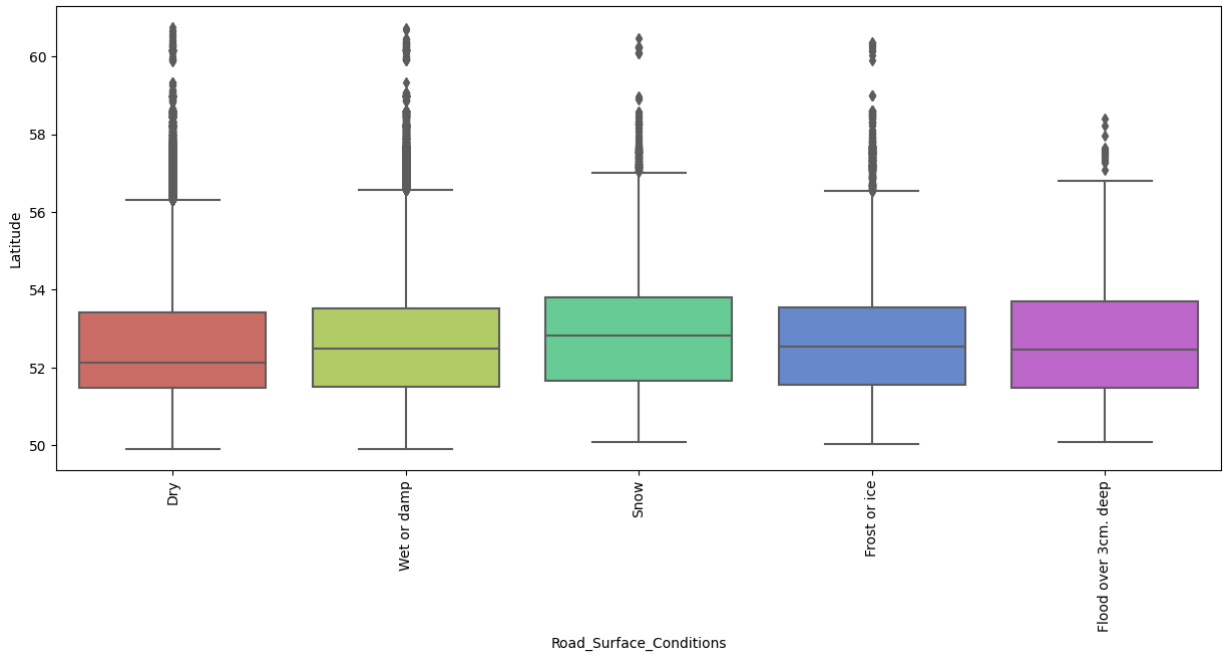


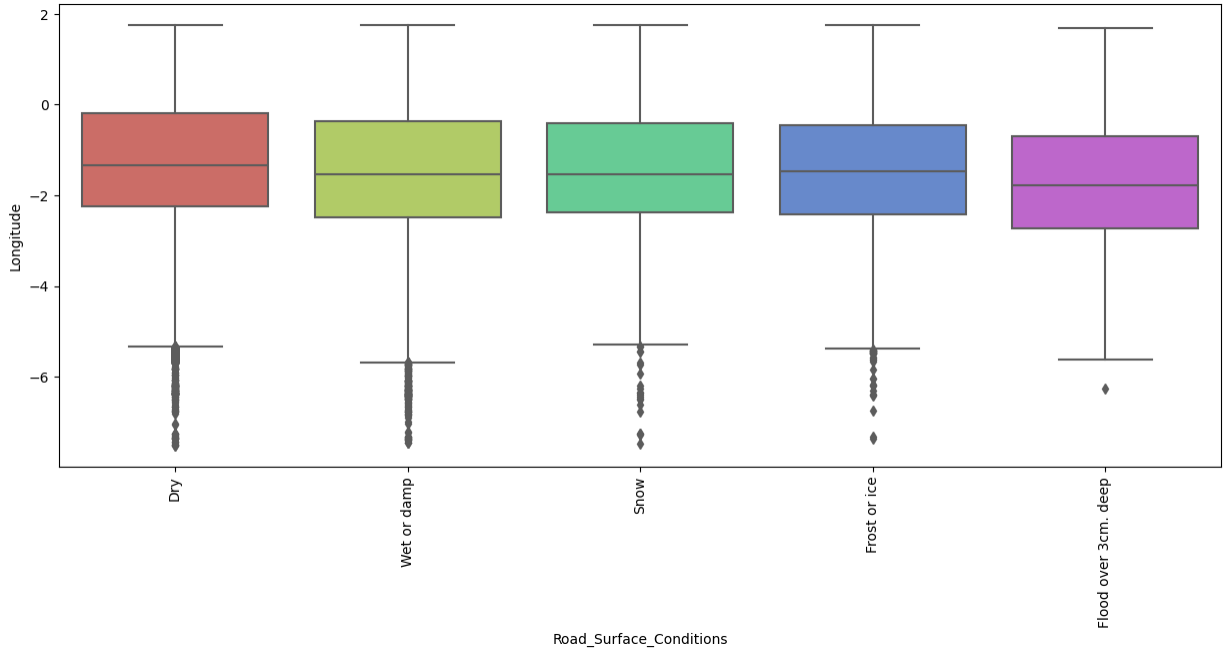


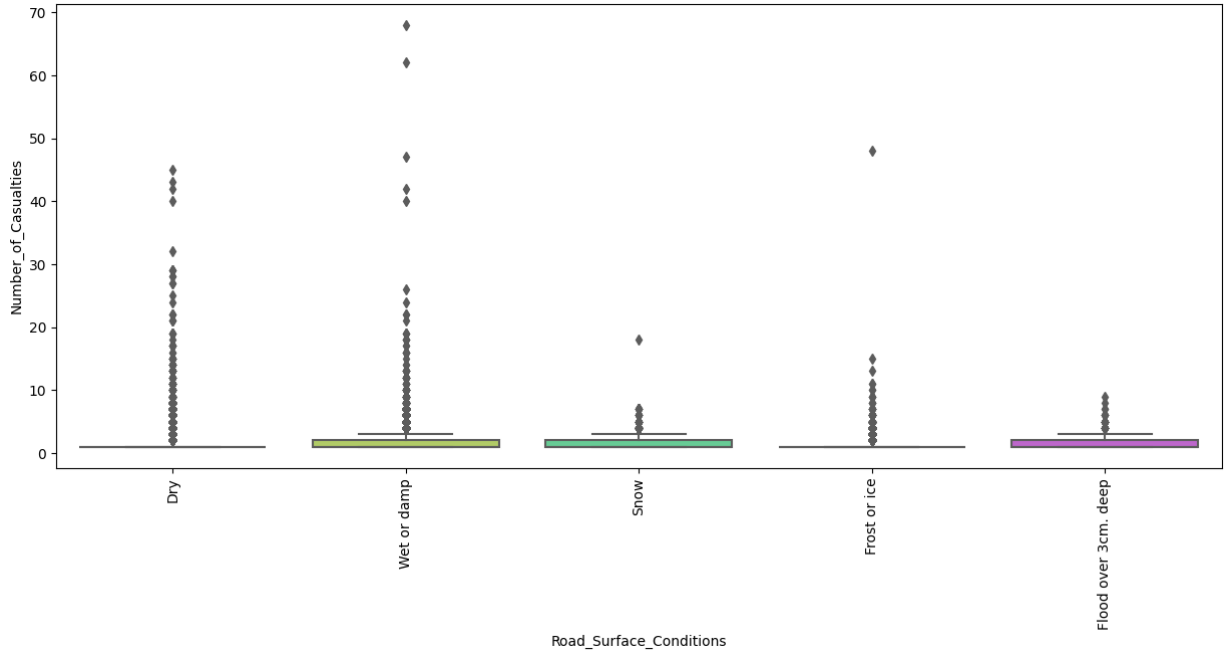


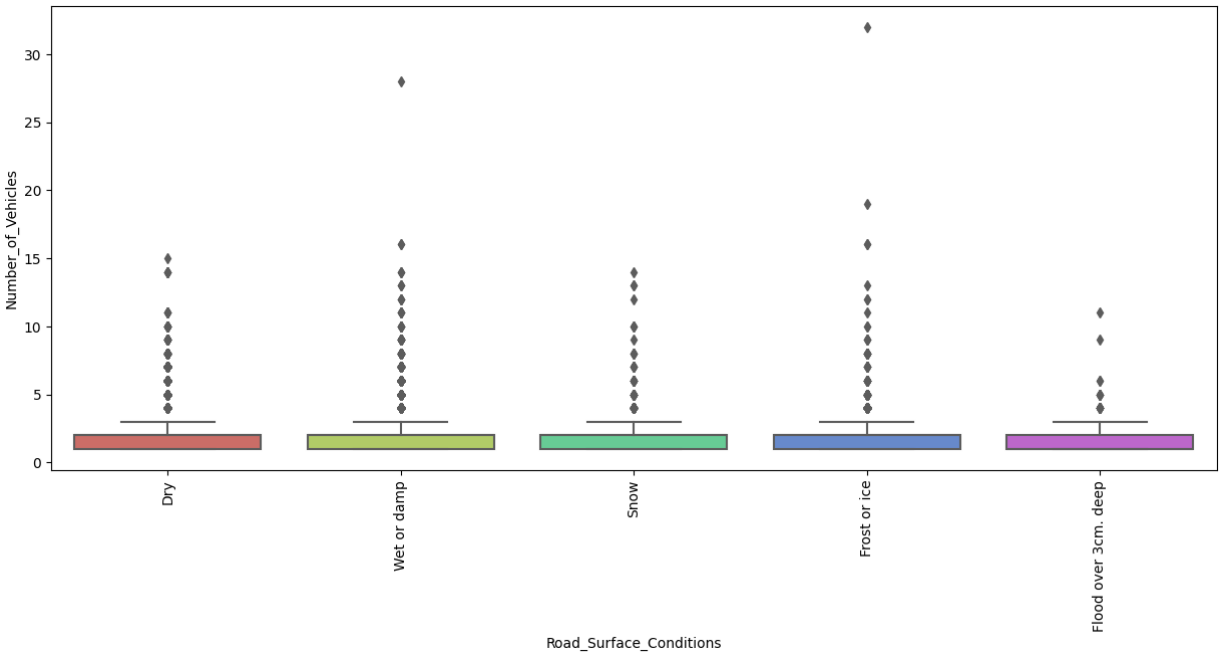


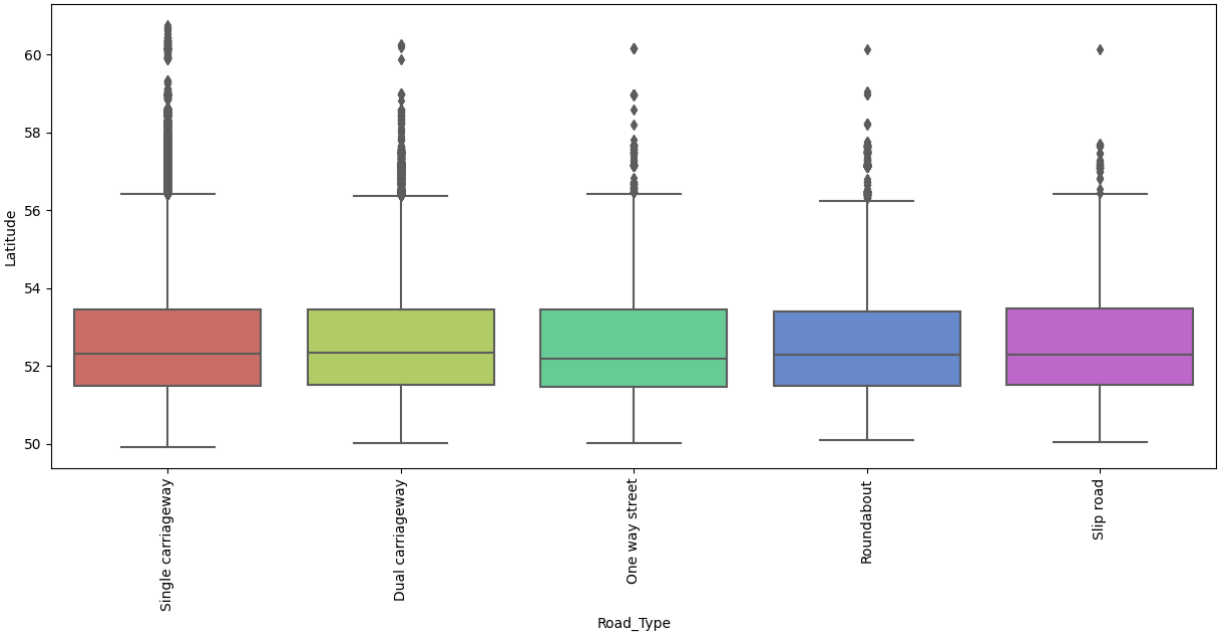


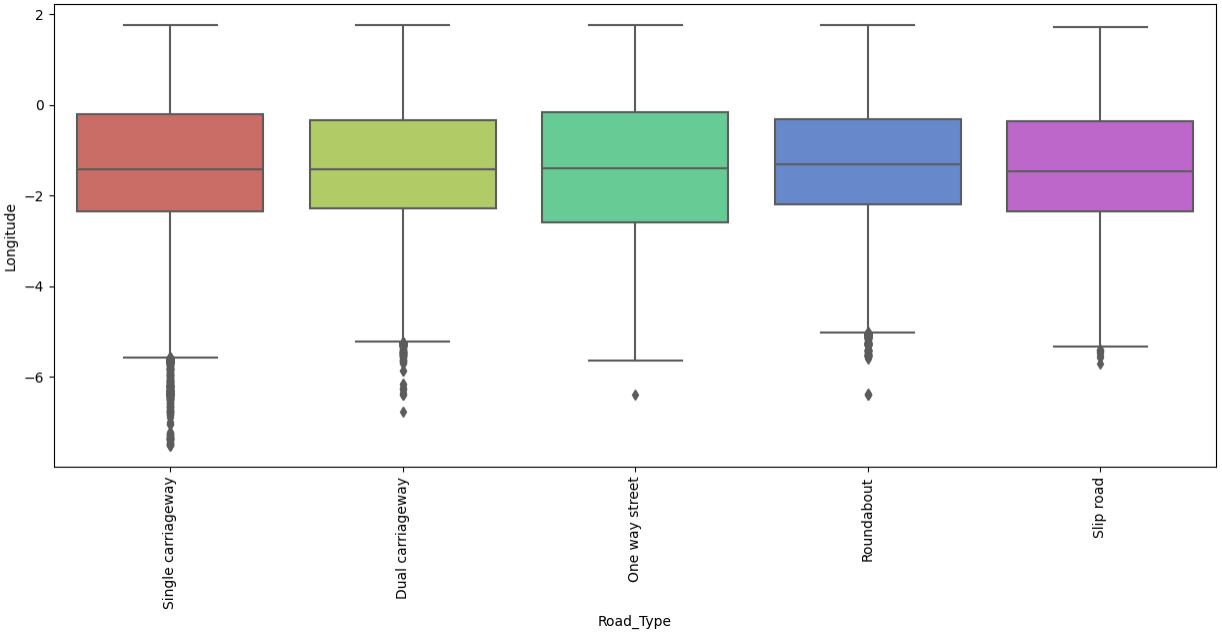


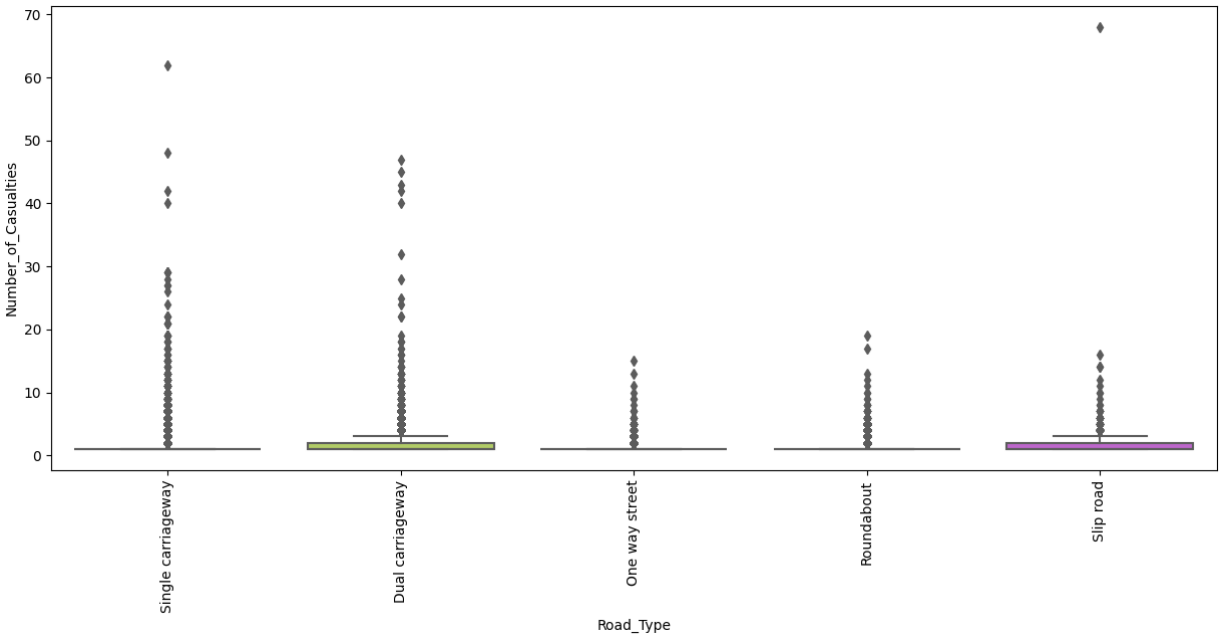


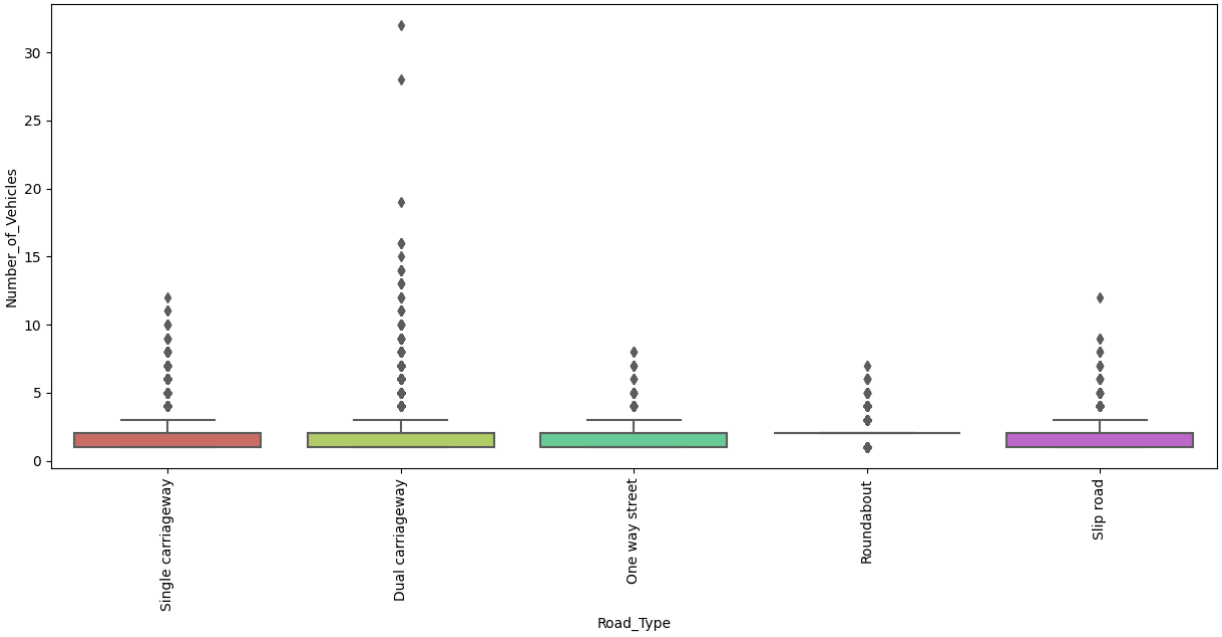


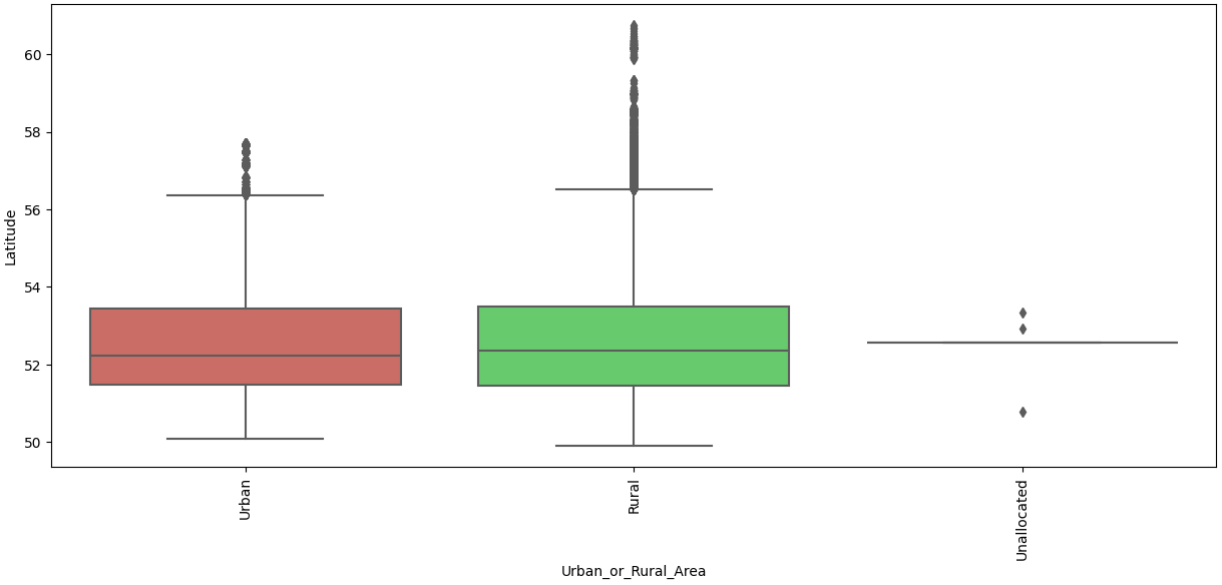


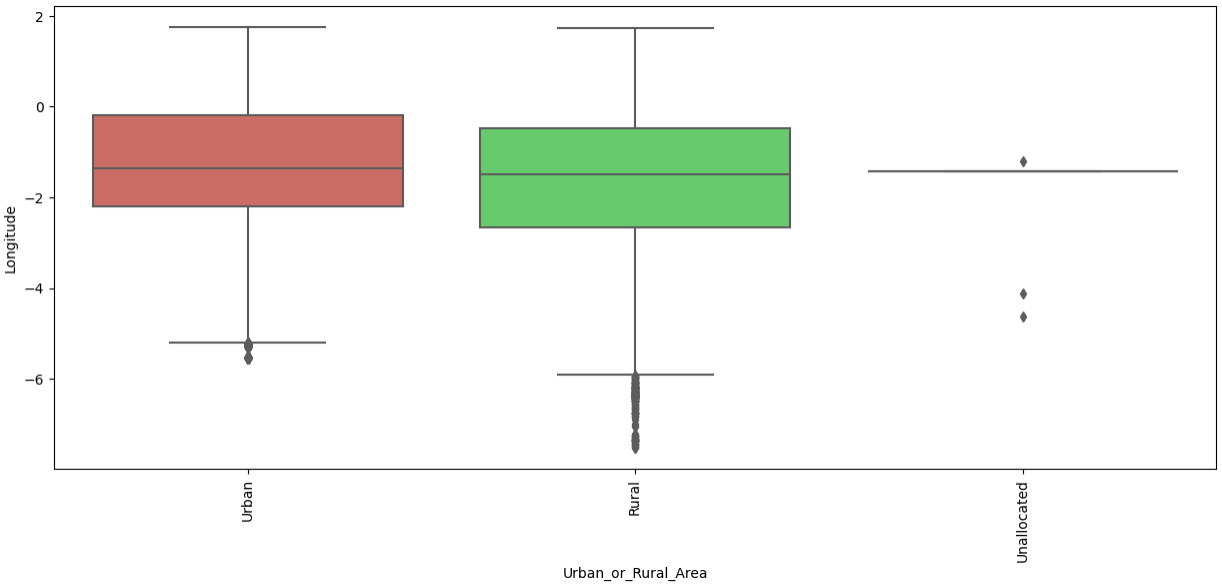


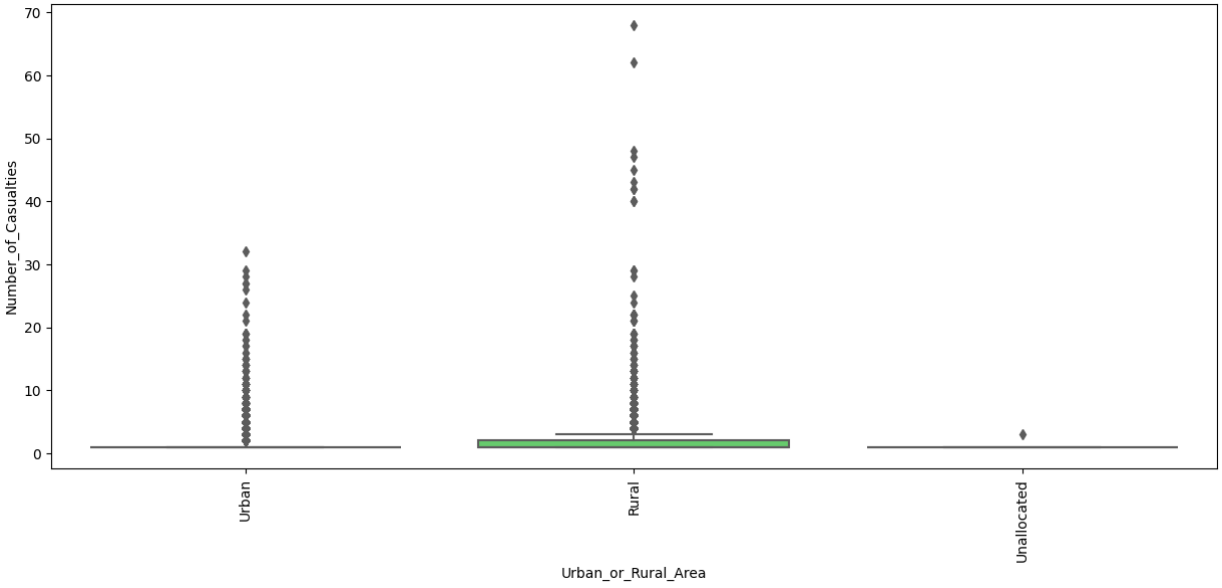


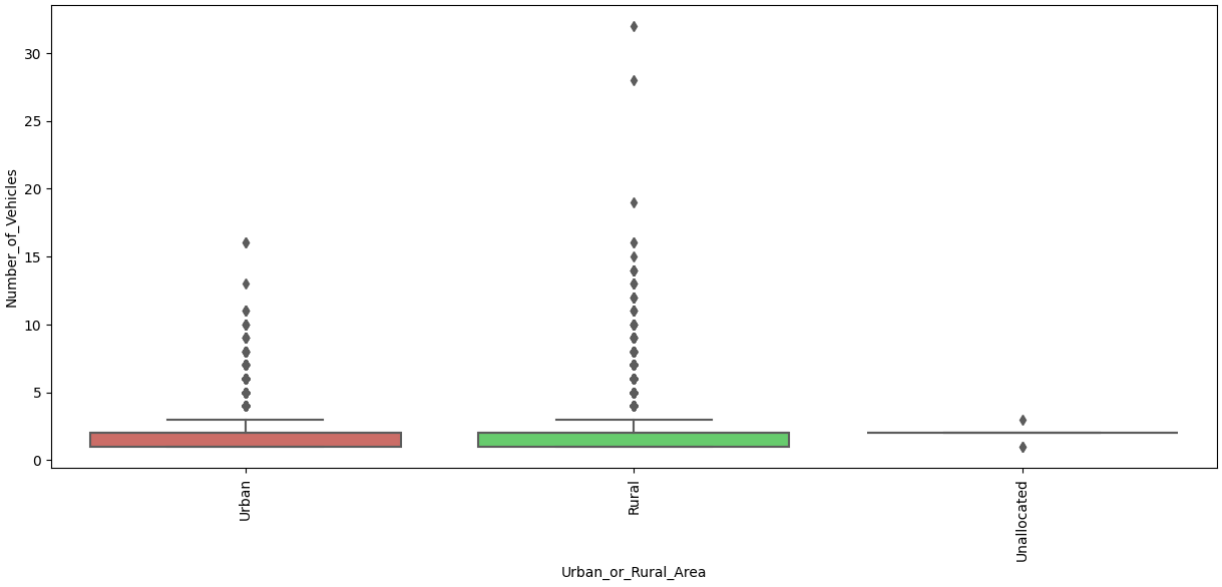


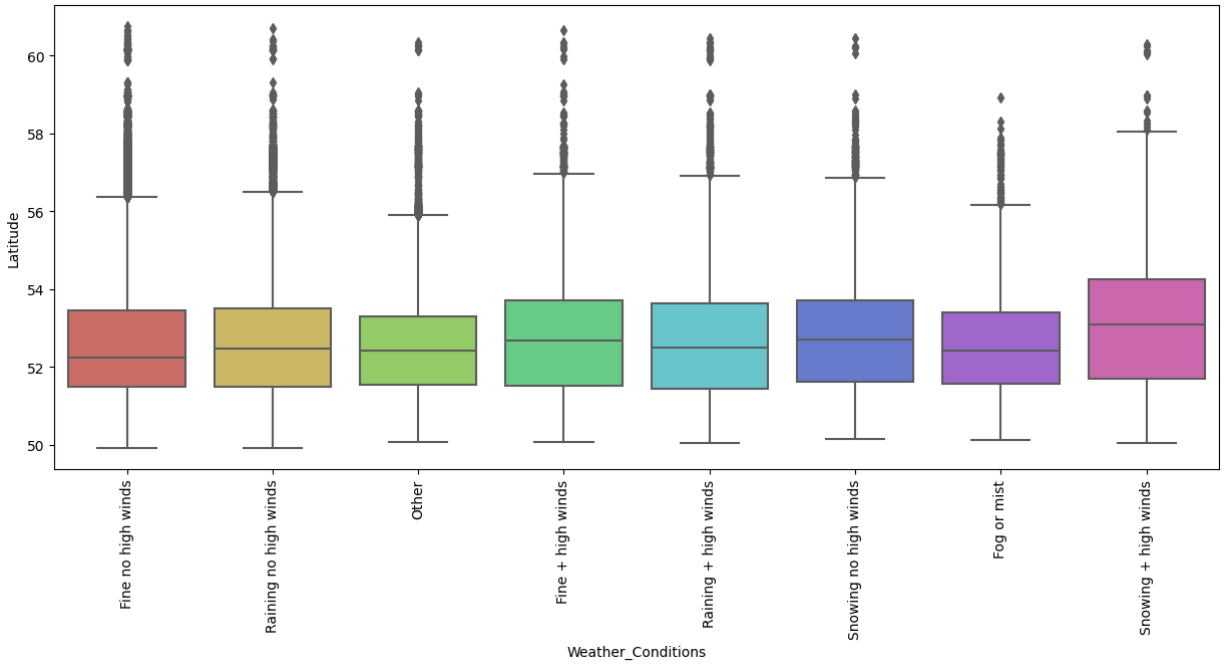


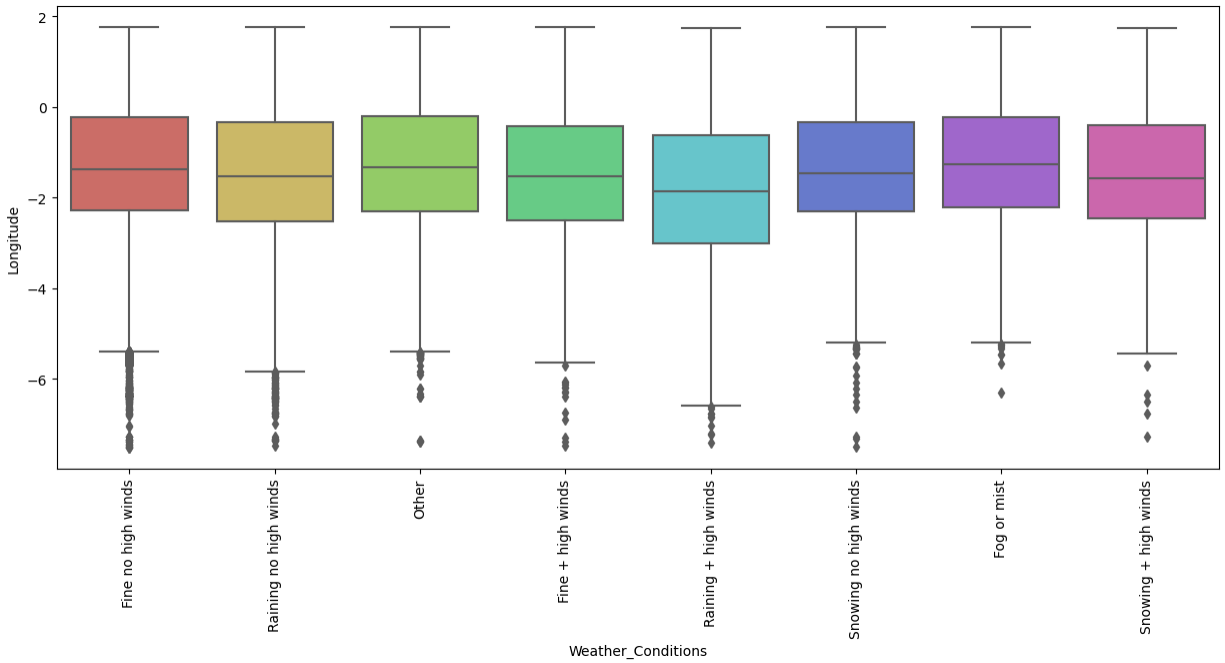


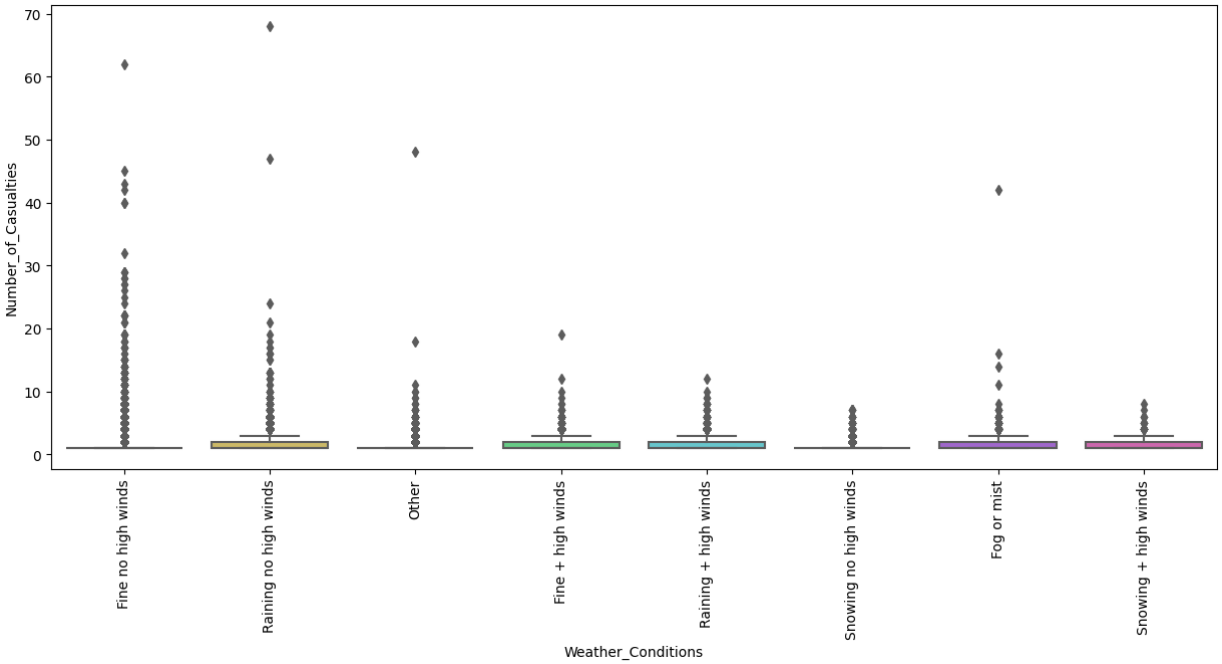


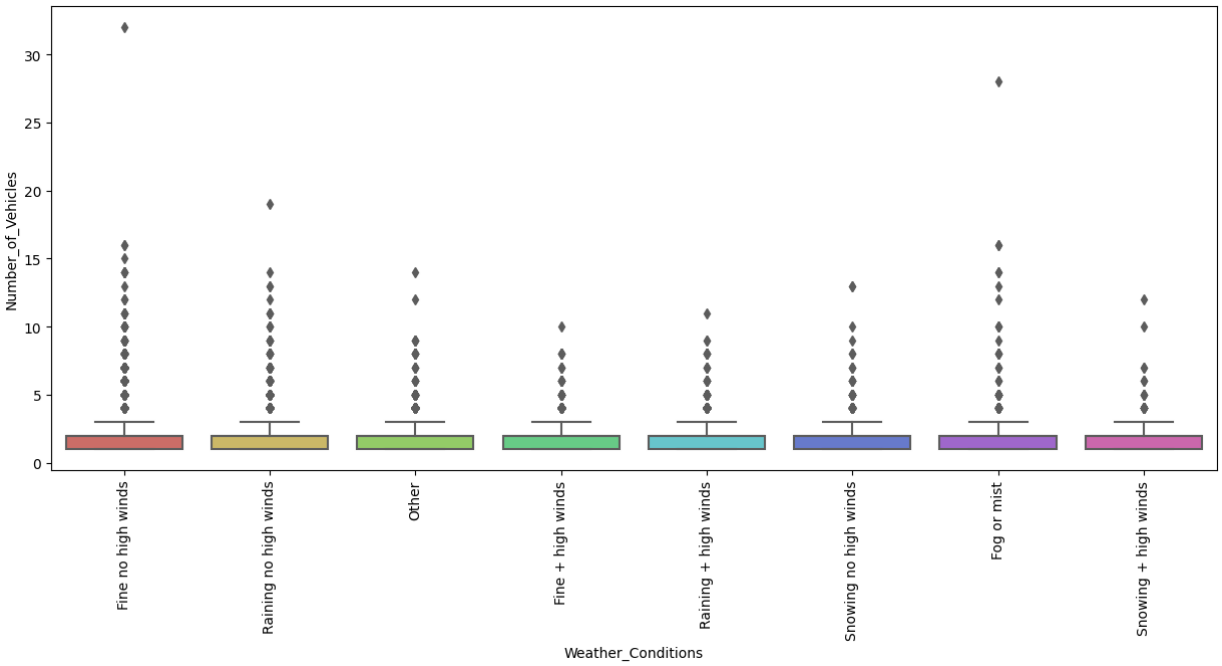




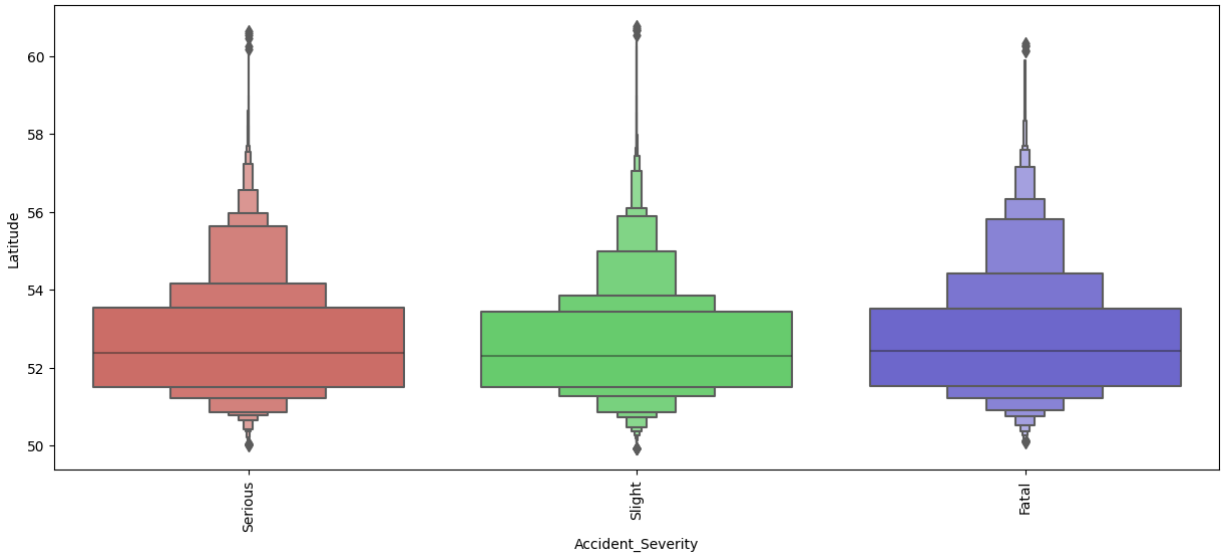


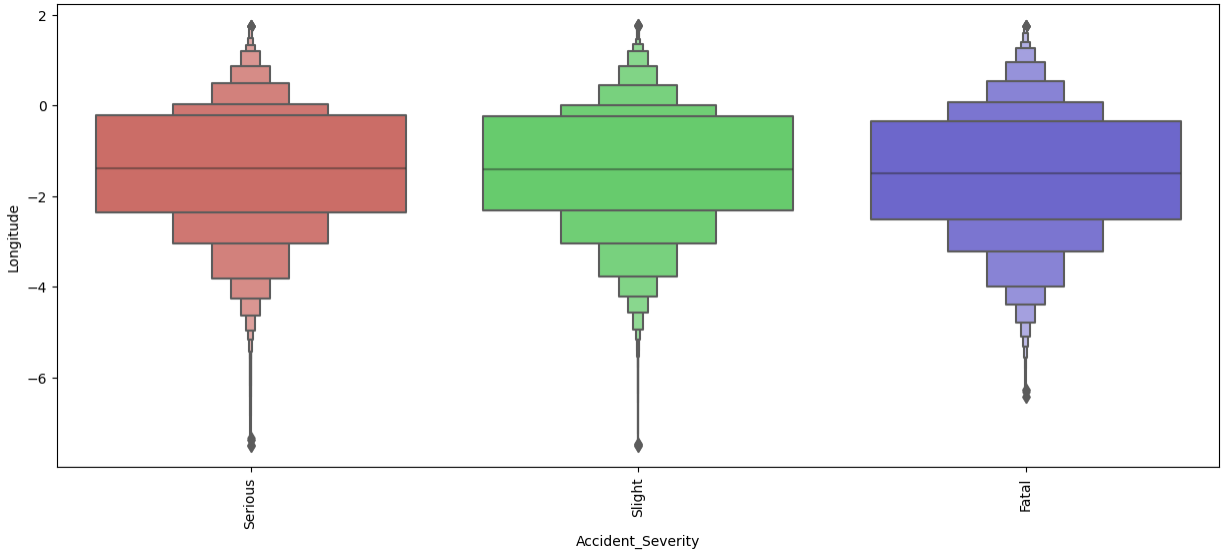


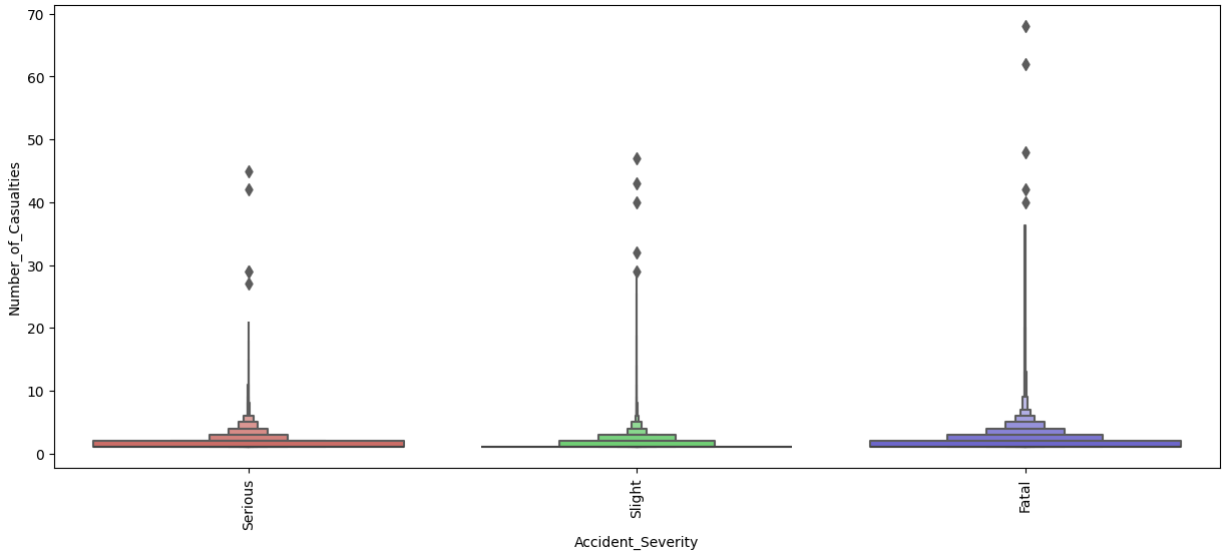


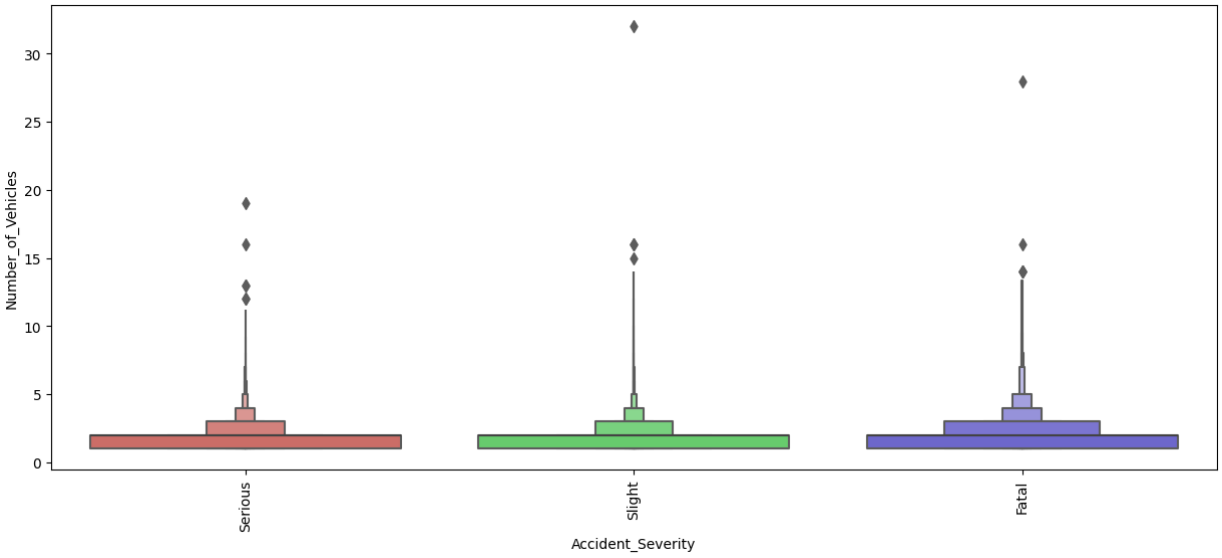


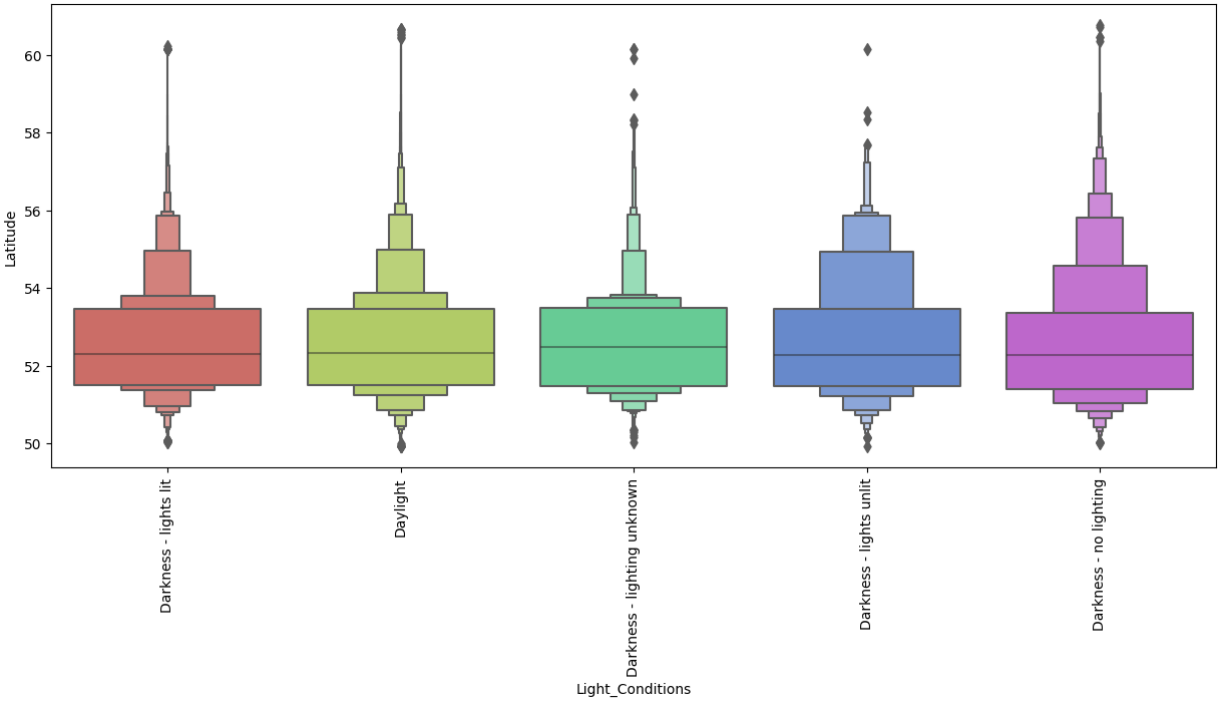
In [43]:
```
for i in categorical:
    for j in continuous:
        plt.figure(figsize=(15,6))
        sns.boxenplot(x = df[i], y = df[j], data = df, palette = 'hls')
        plt.xticks(rotation = 90)
        plt.show()
```

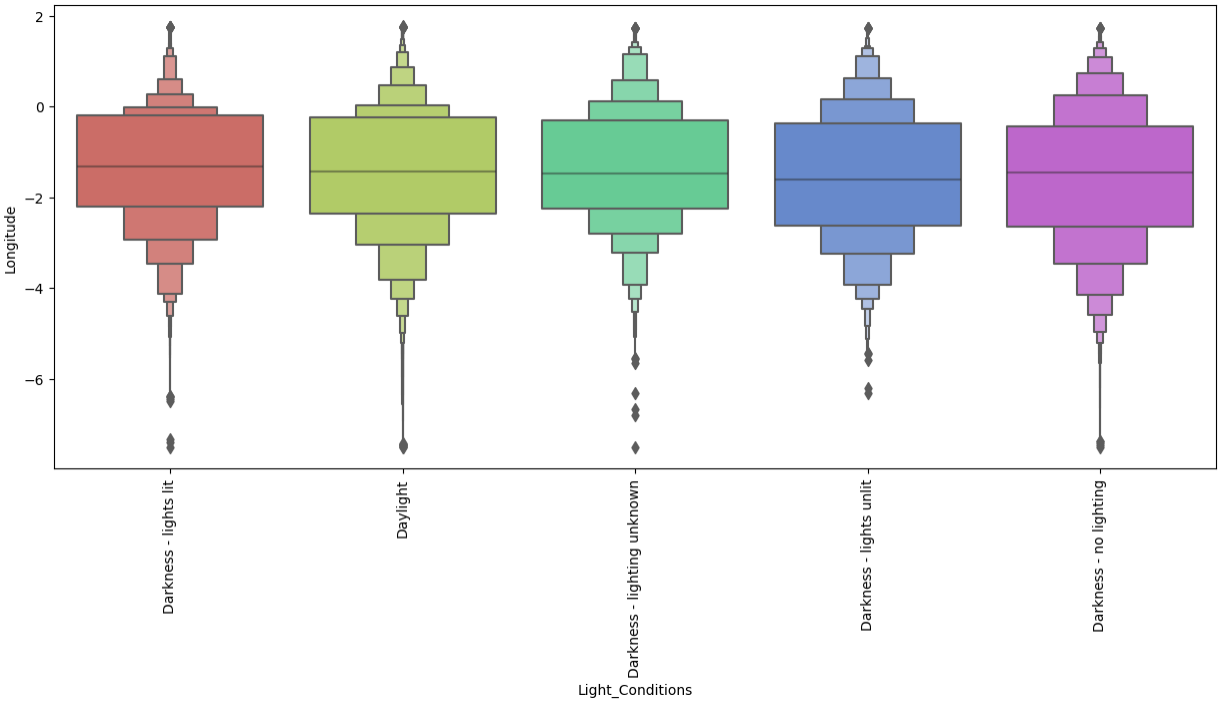


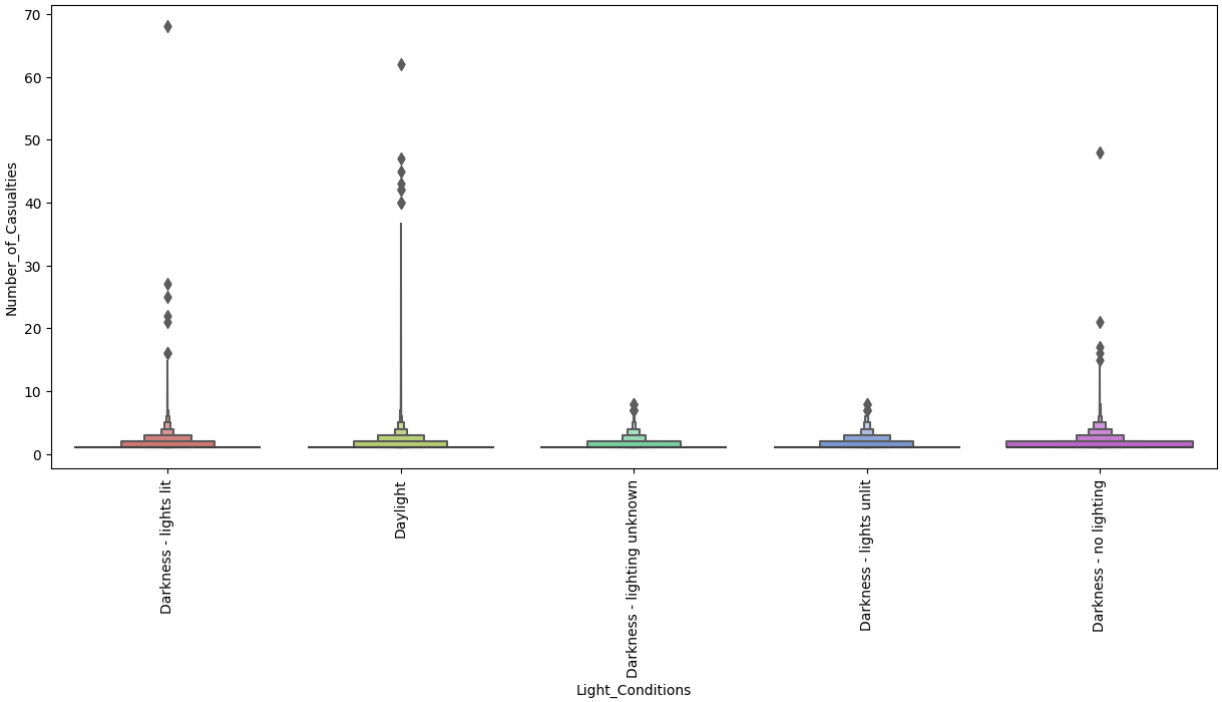


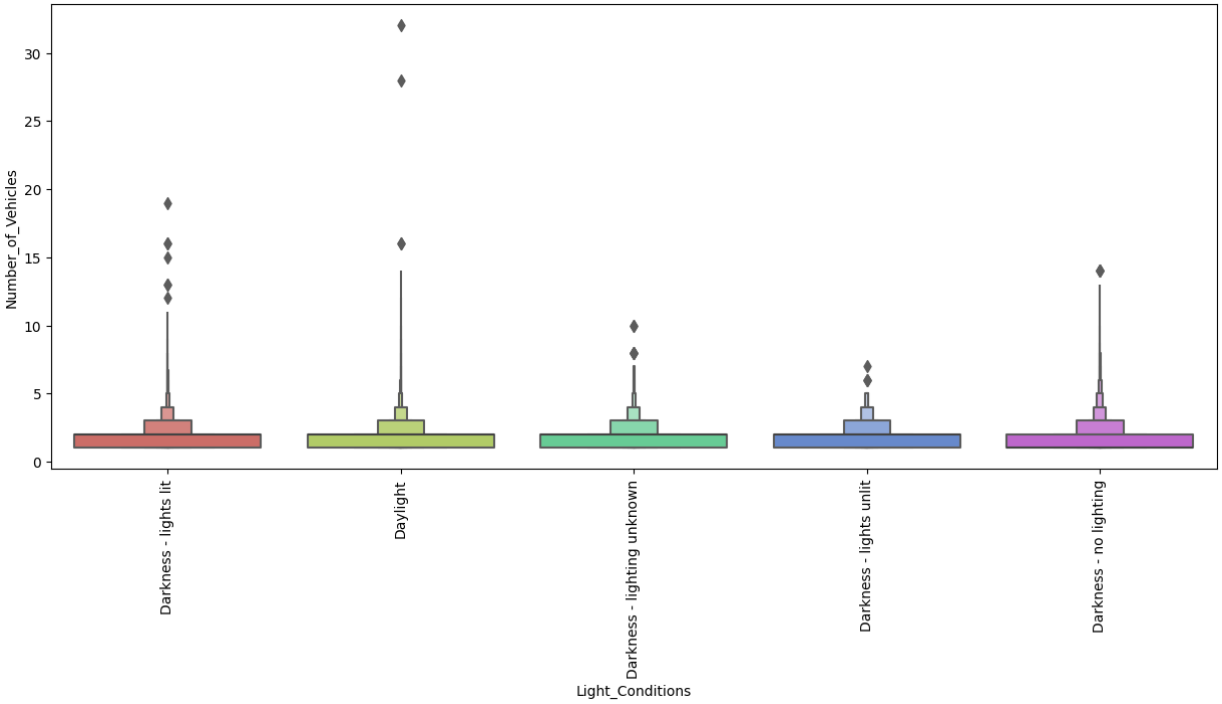


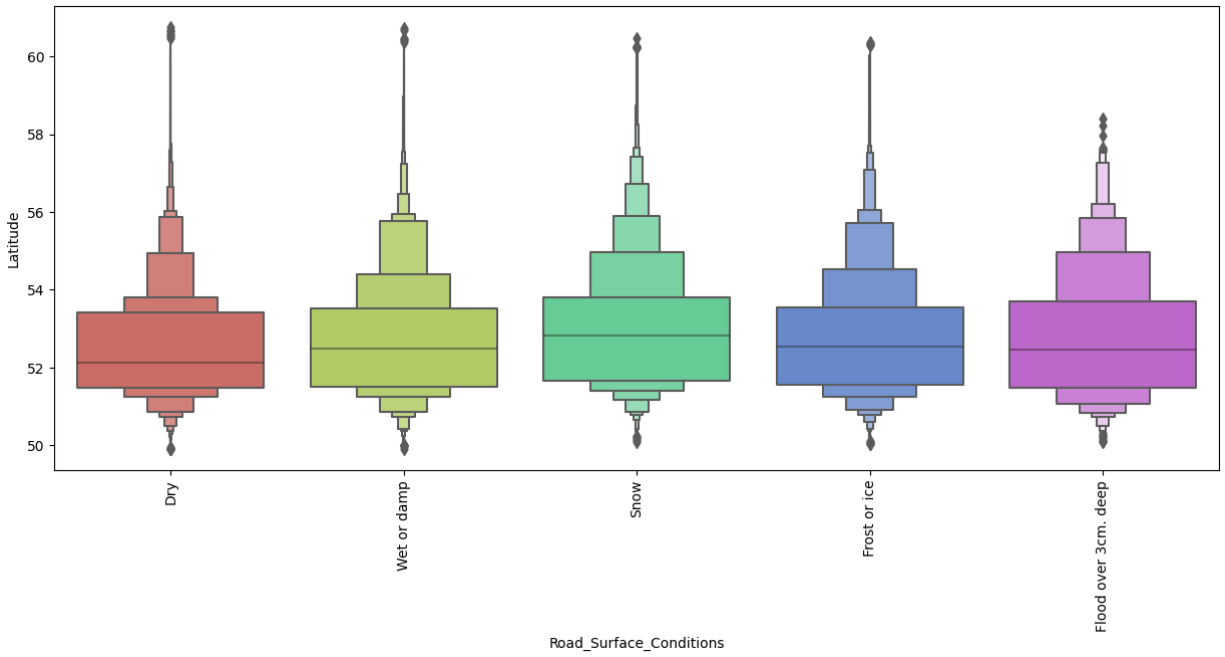


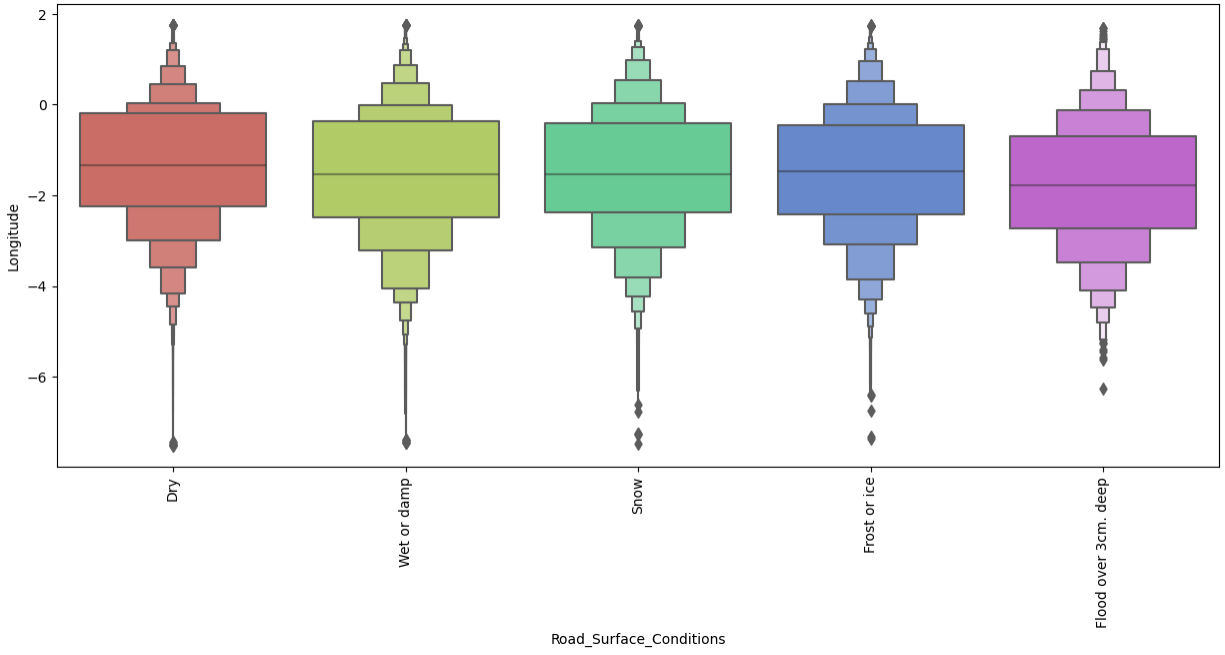


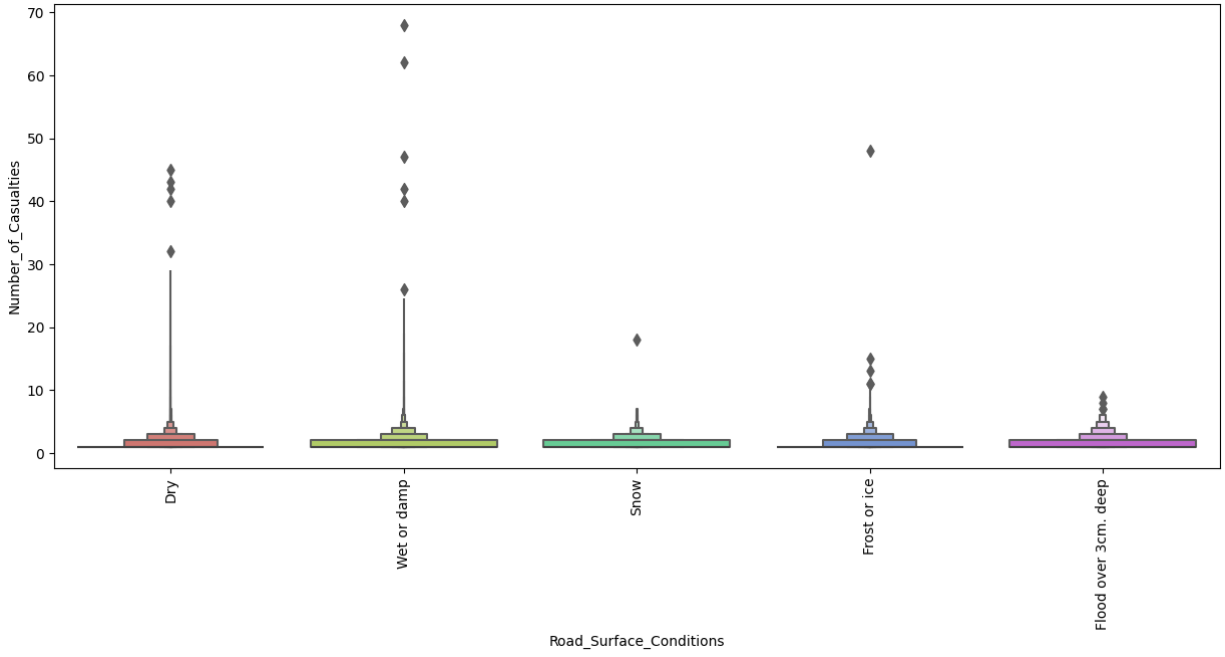


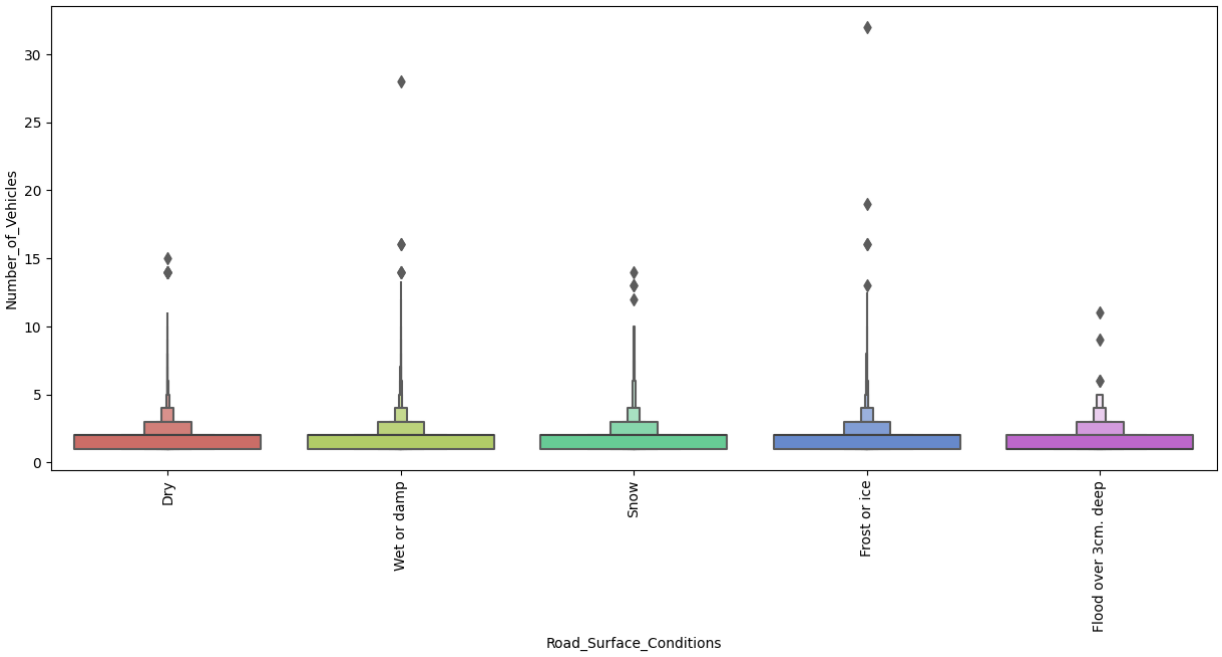


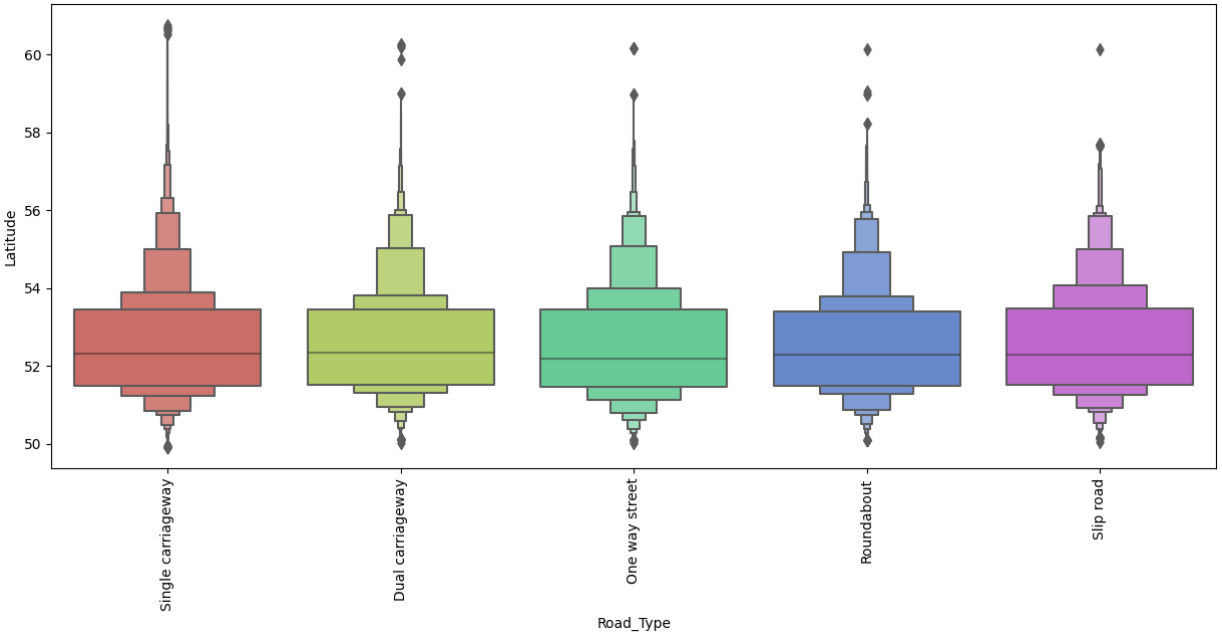


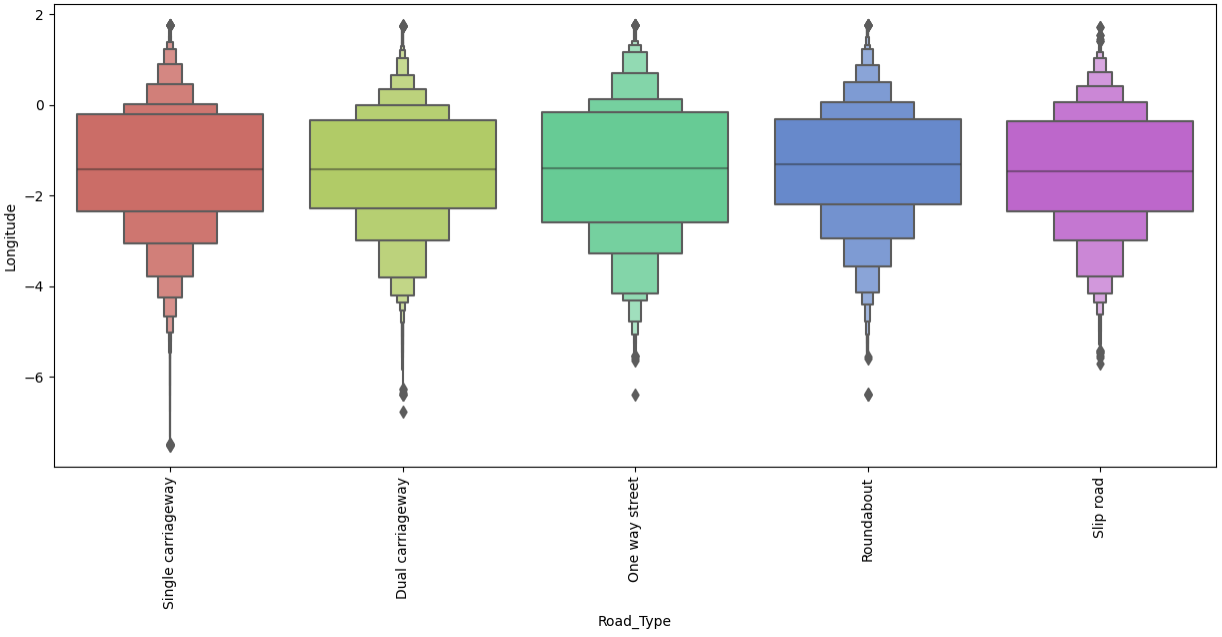


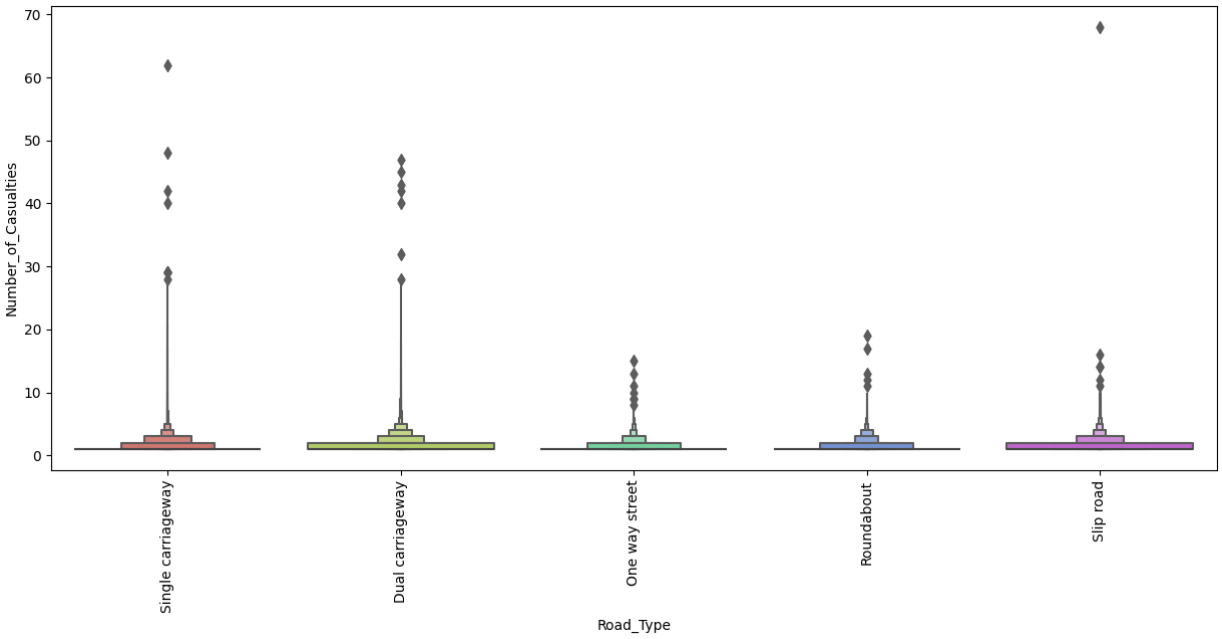


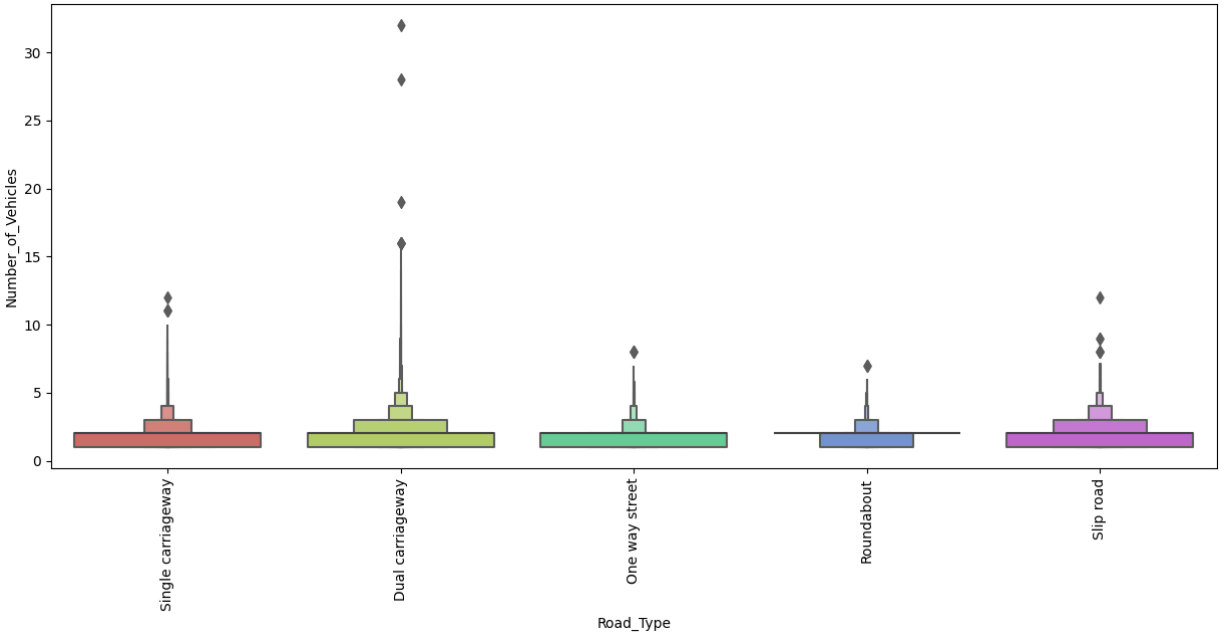


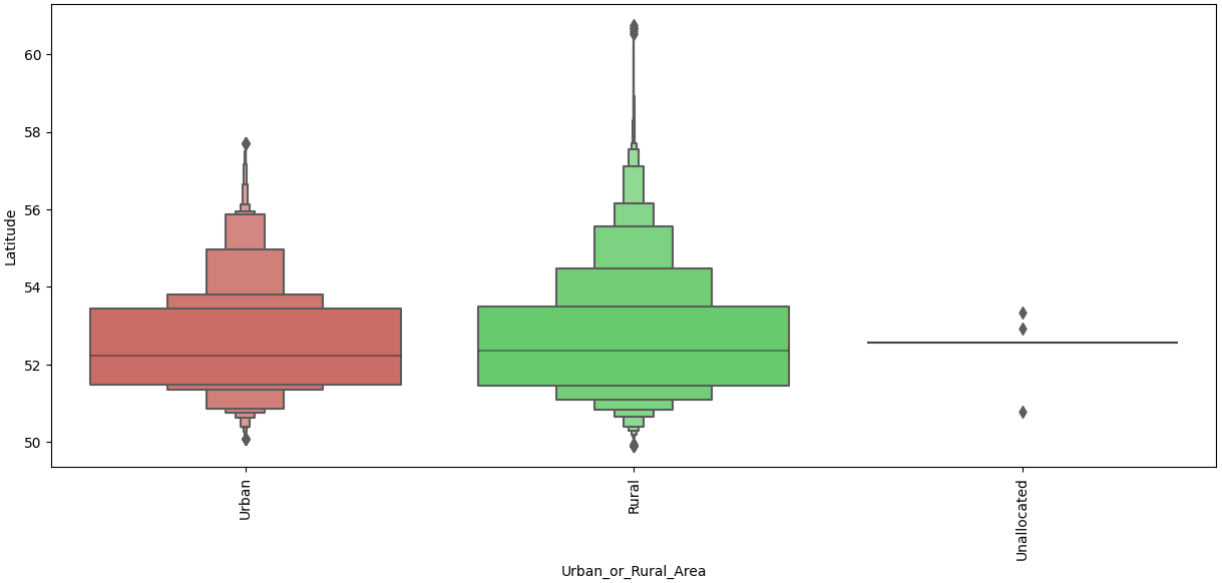


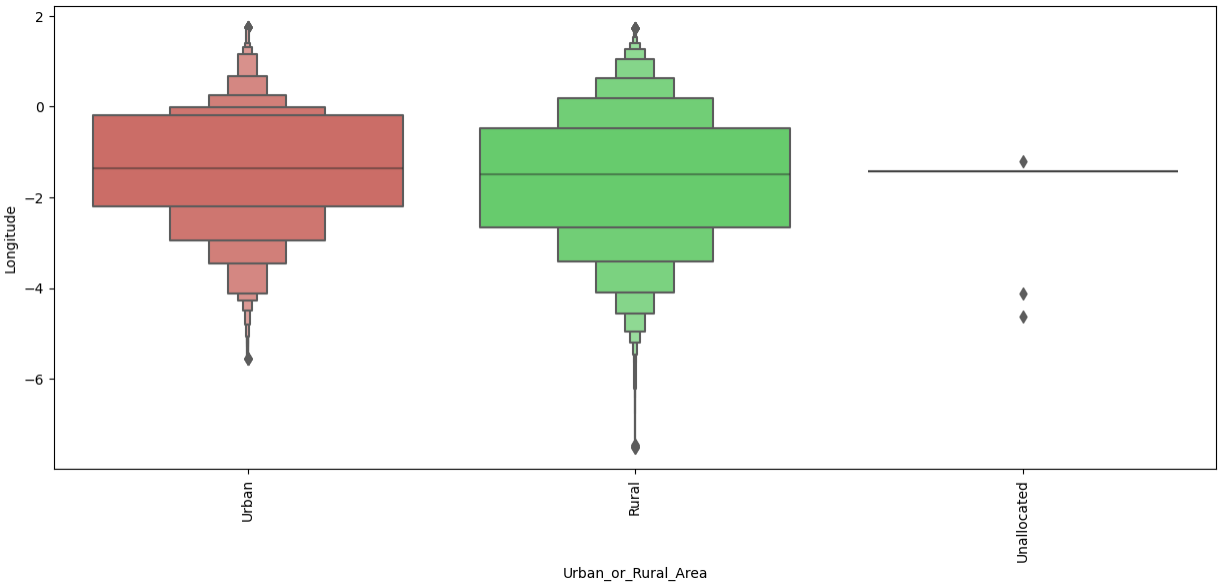


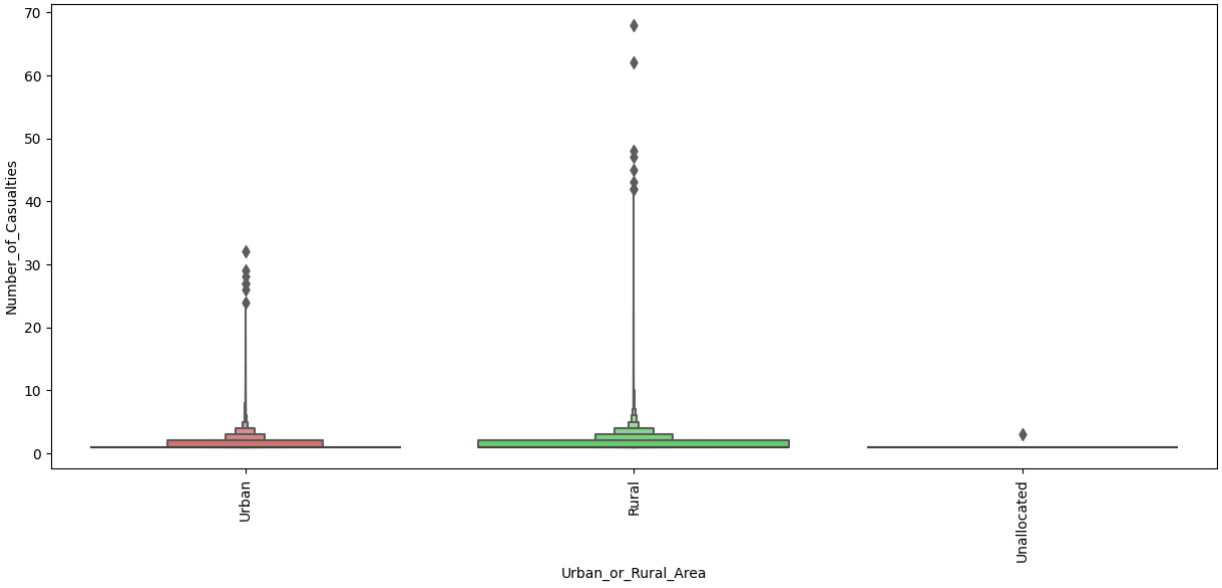


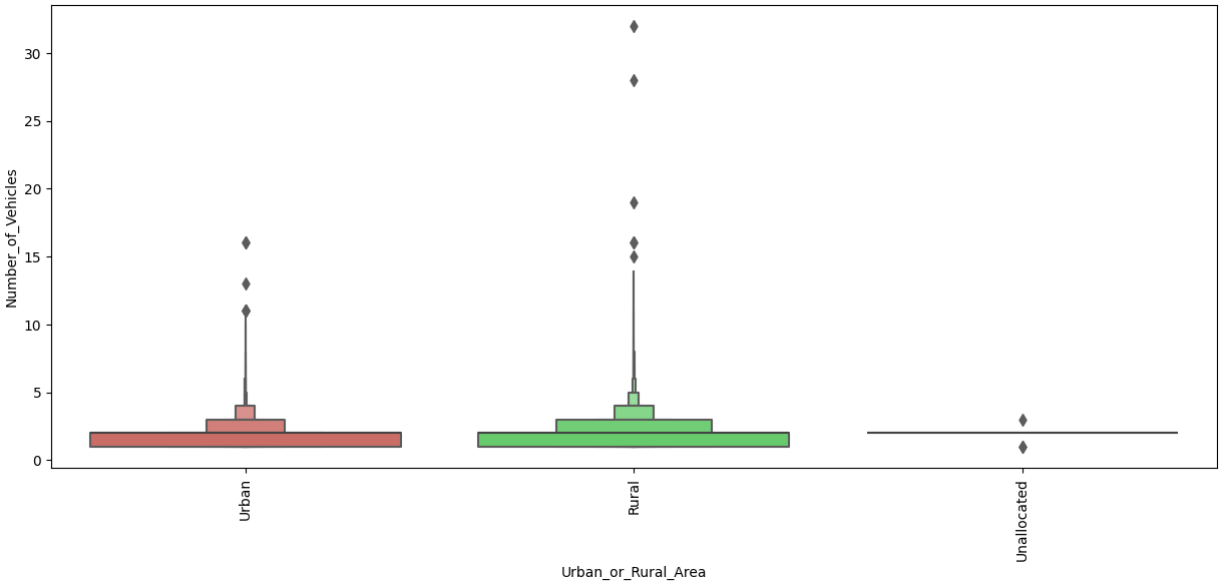


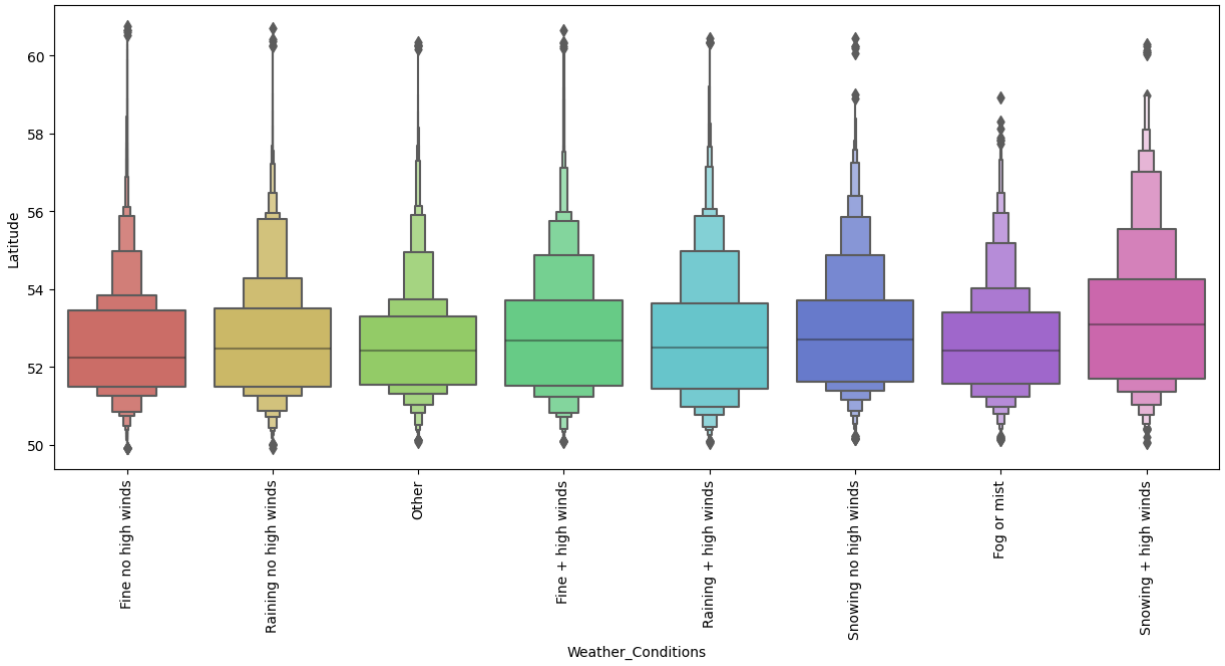


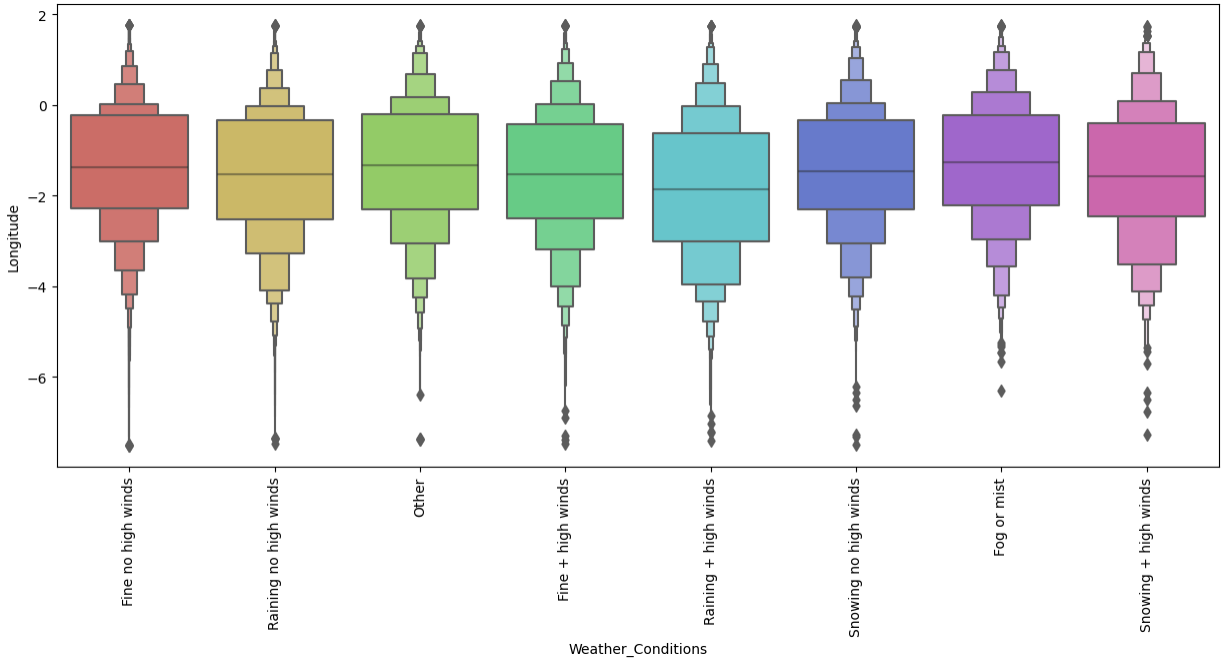


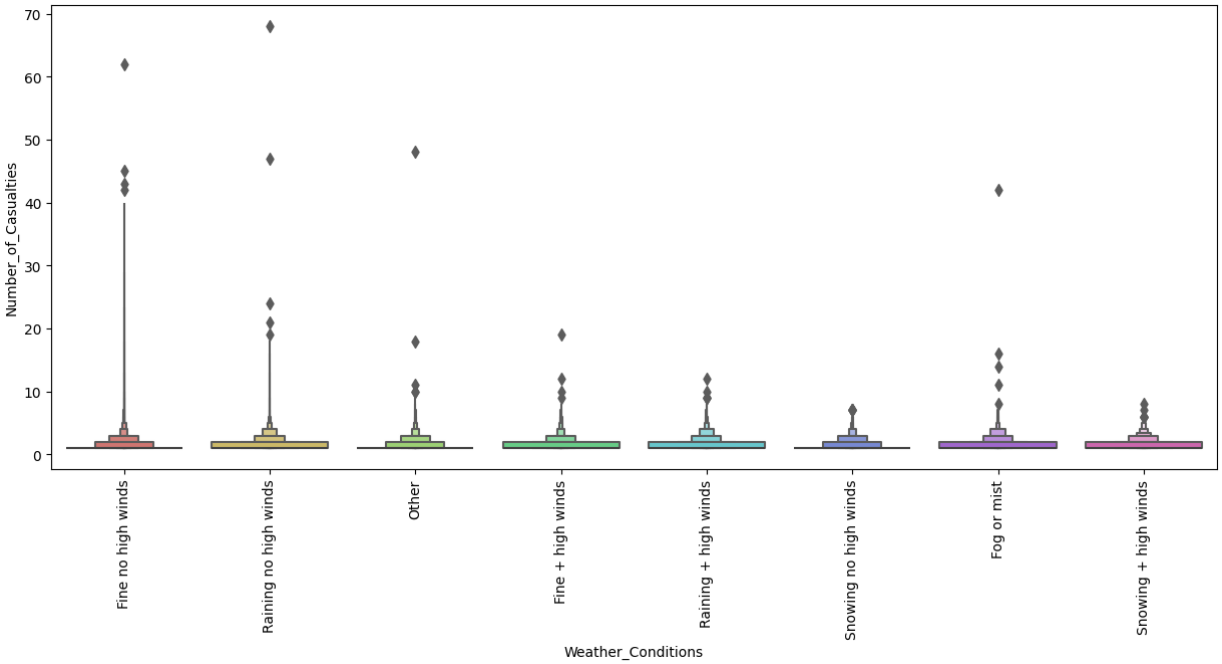


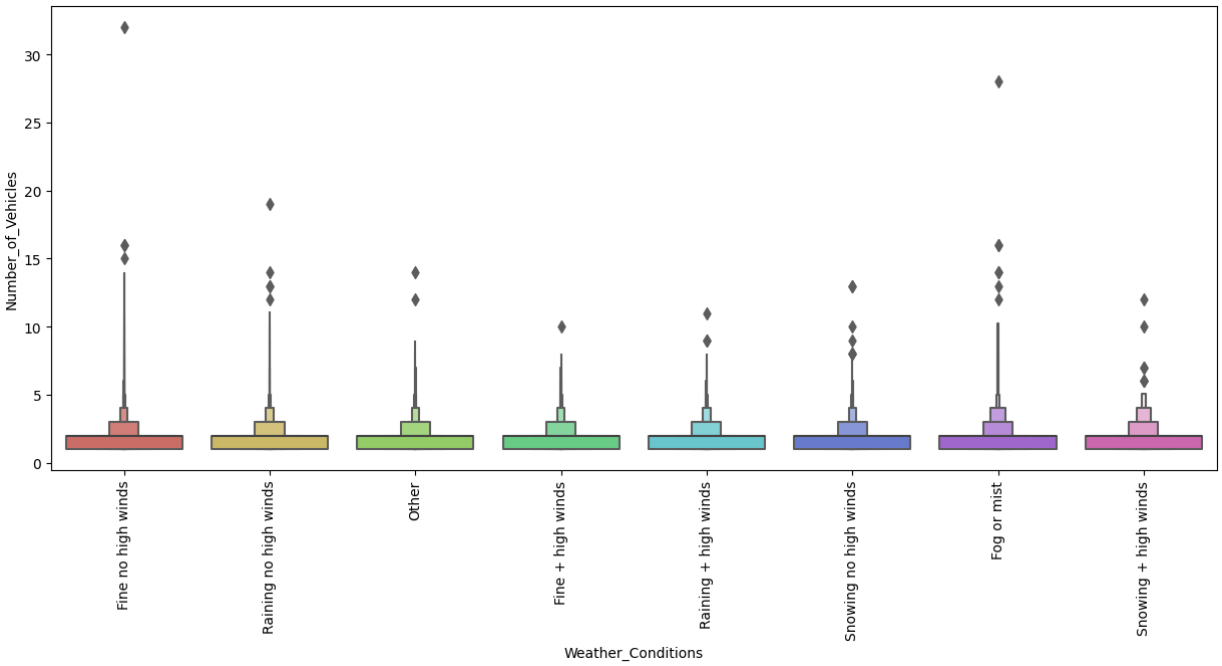




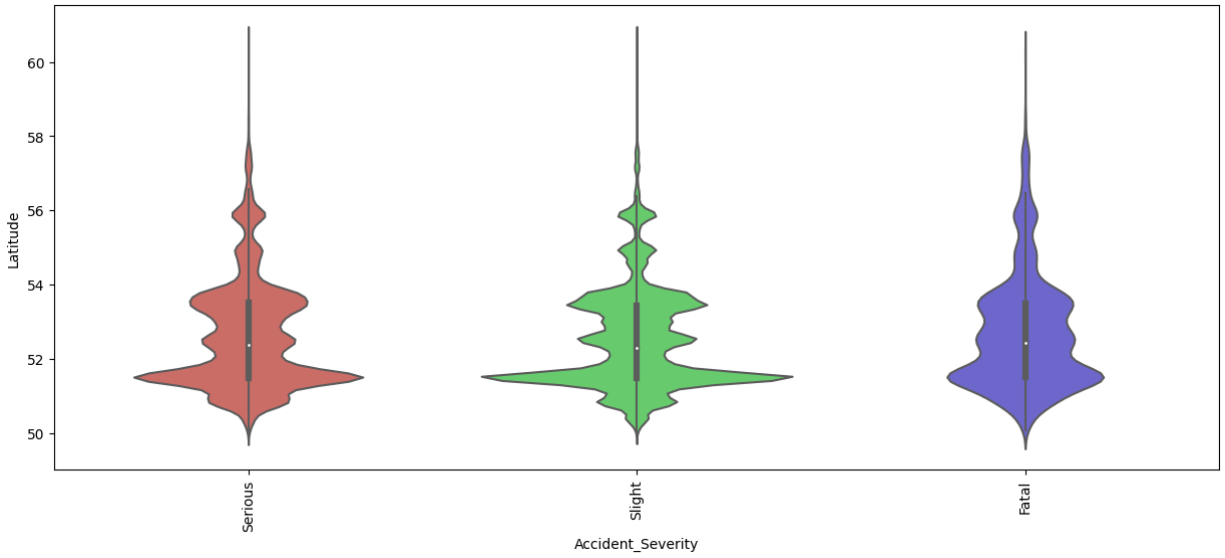


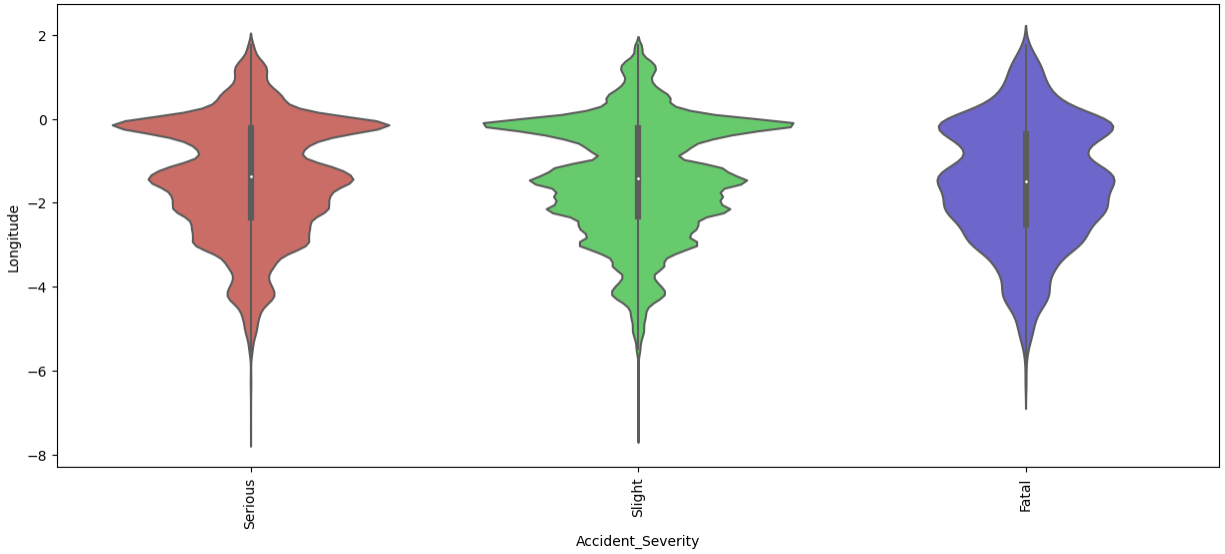


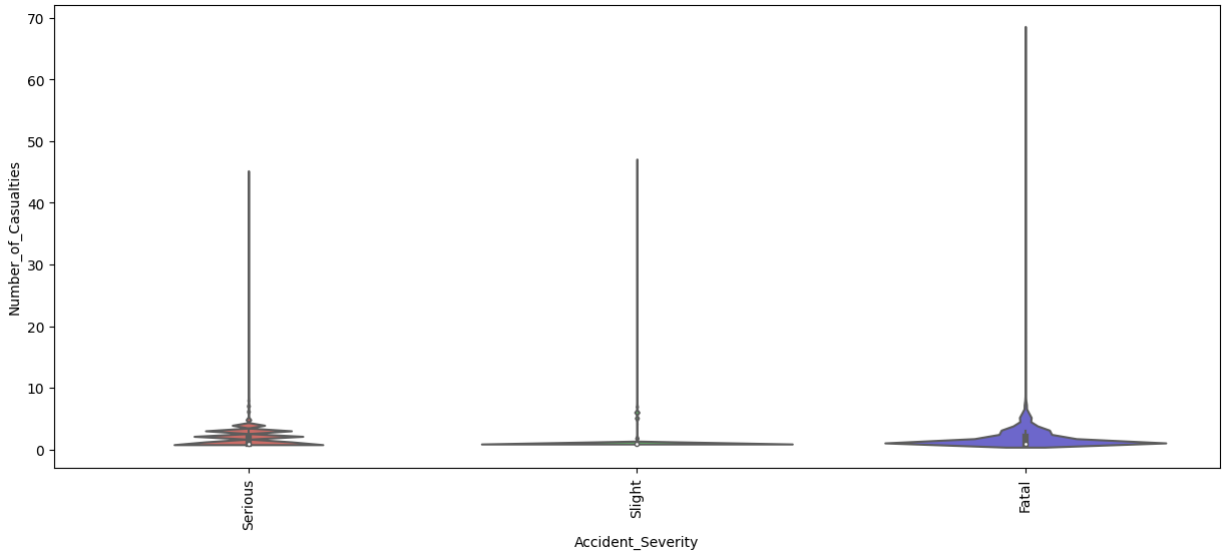


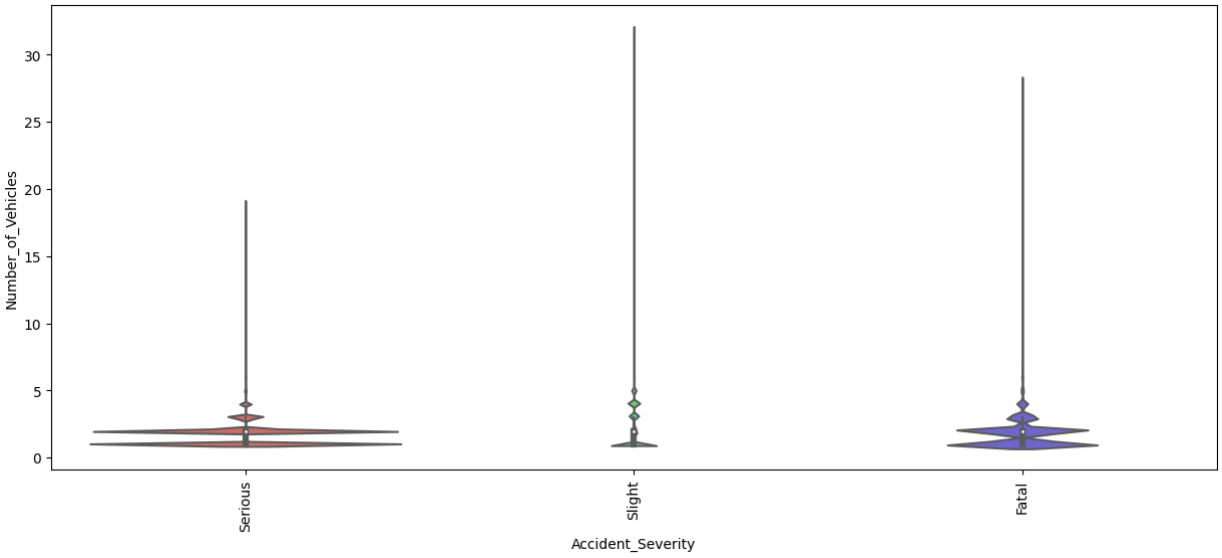


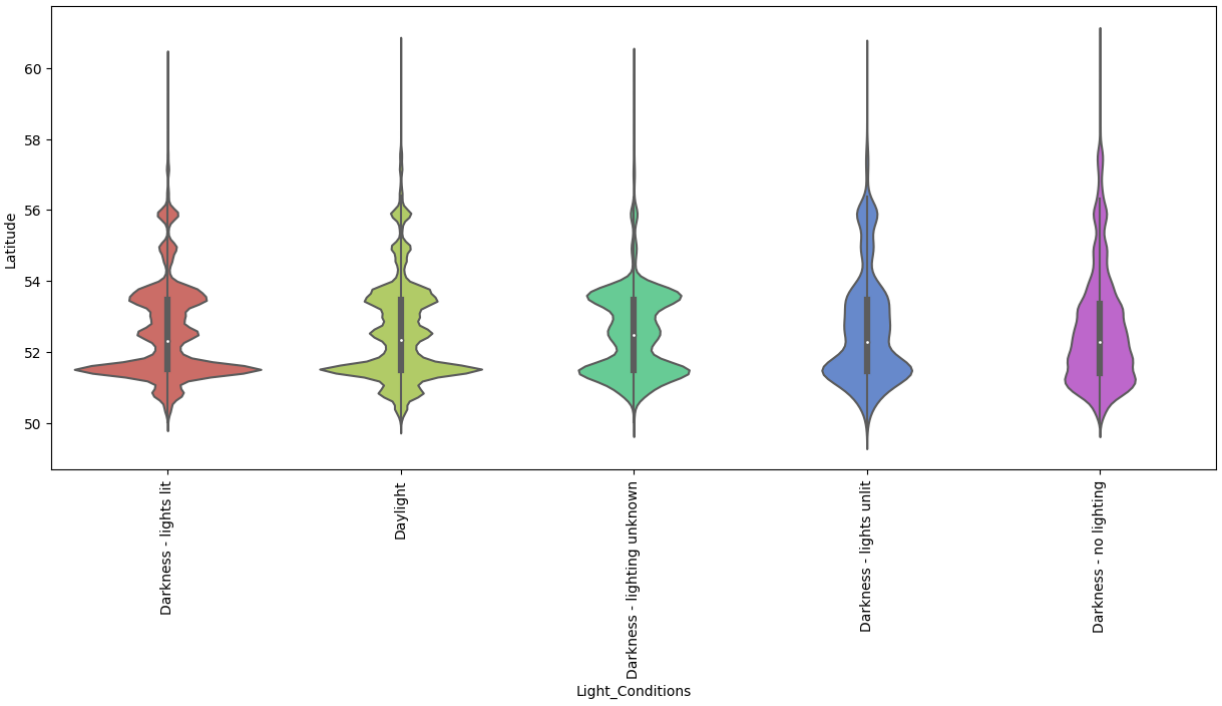
In [44]:
```
for i in categorical:
    for j in continuous:
        plt.figure(figsize=(15,6))
        sns.violinplot(x = df[i], y = df[j], data = df, palette = 'hls')
        plt.xticks(rotation = 90)
        plt.show()
```

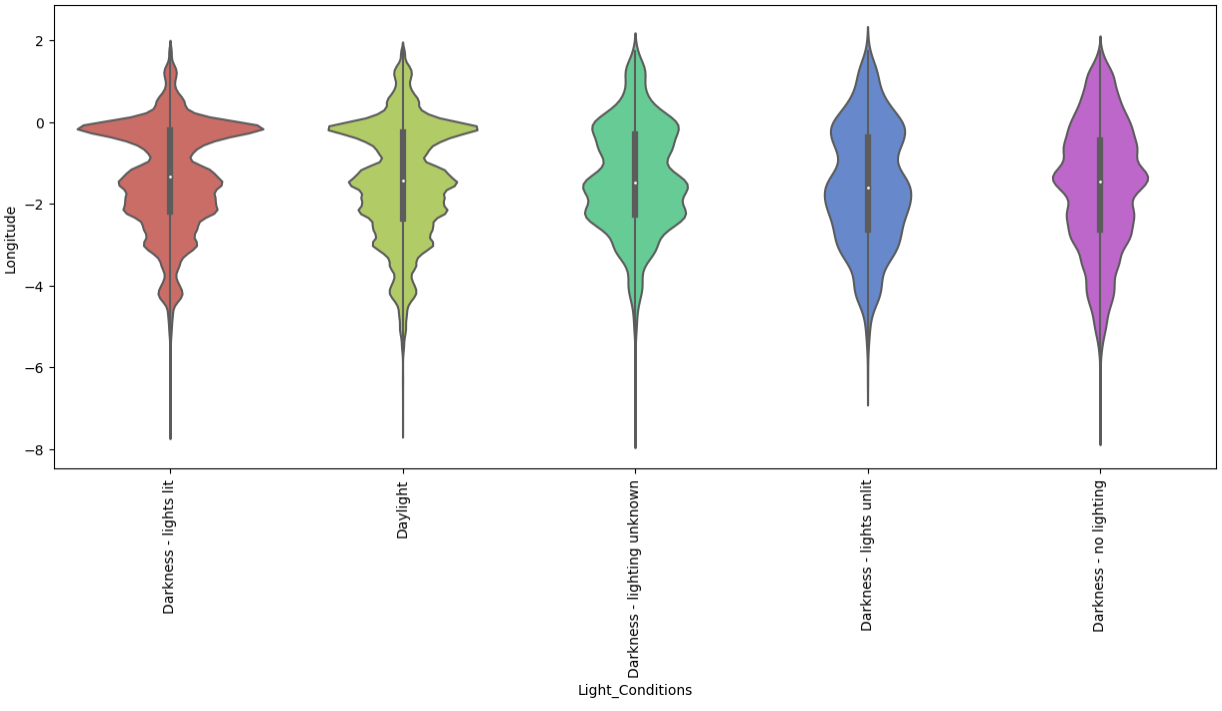


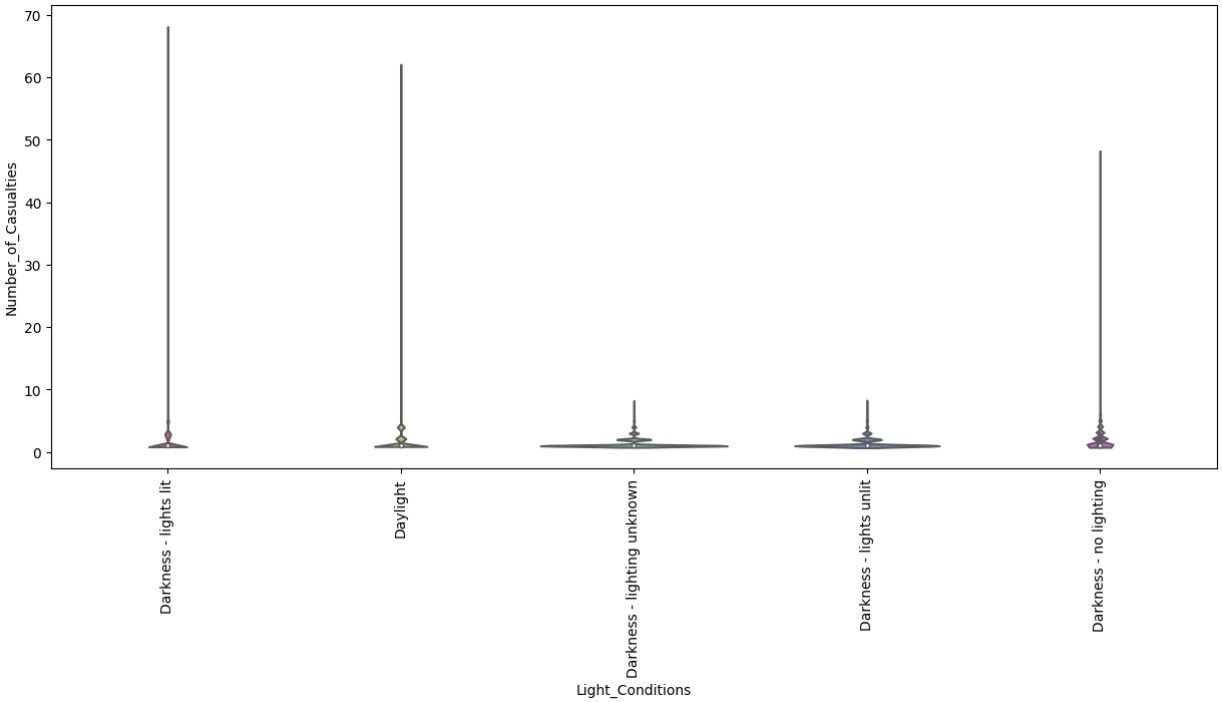


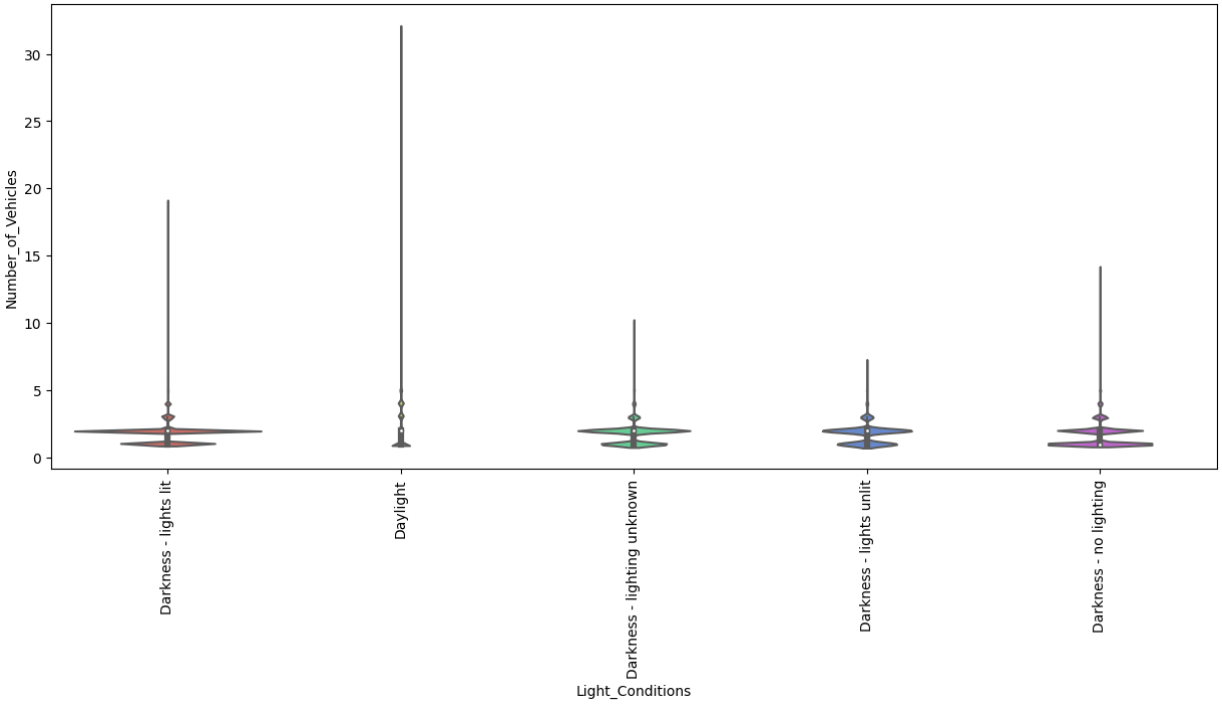


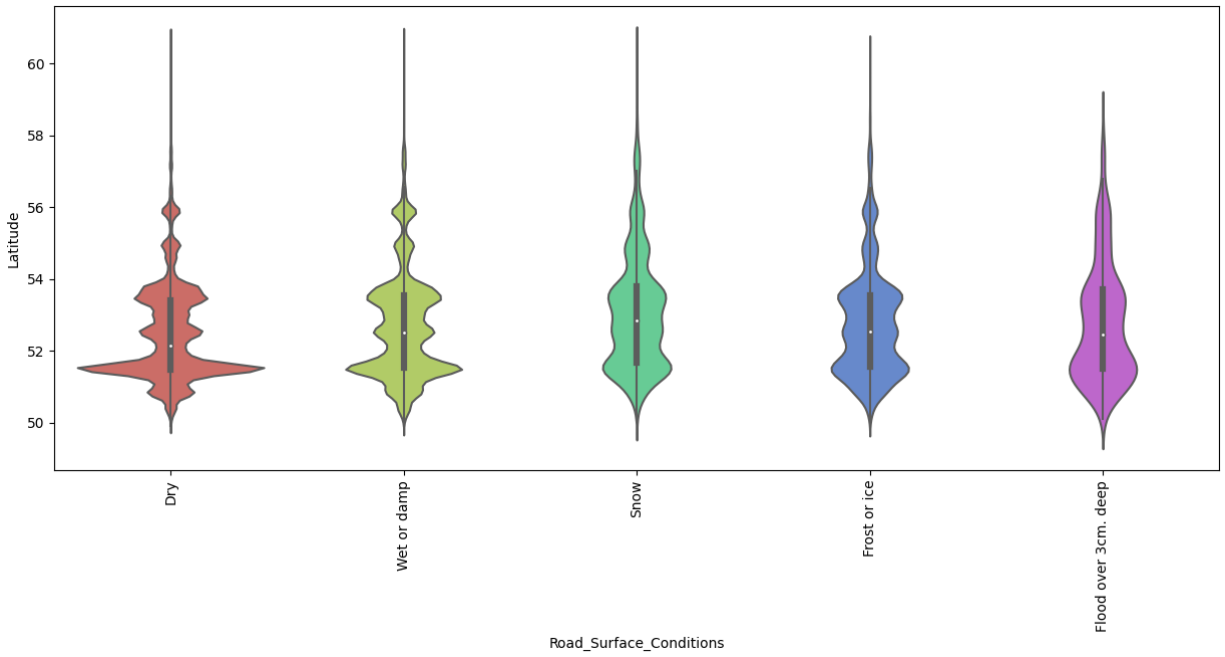


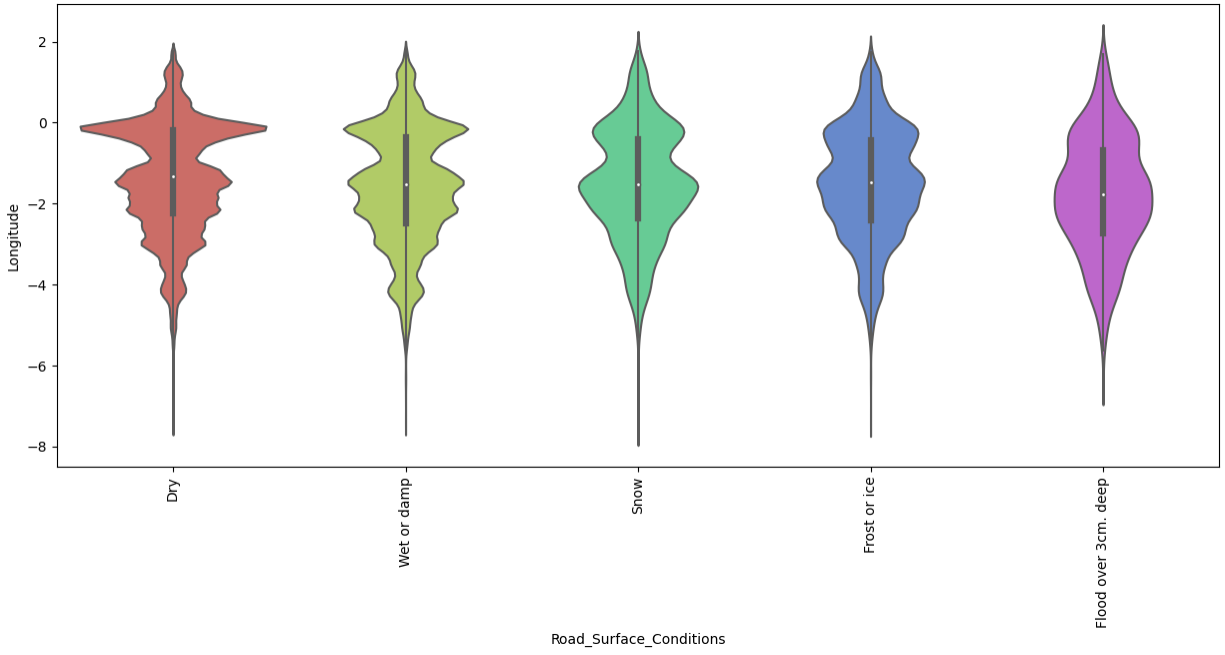


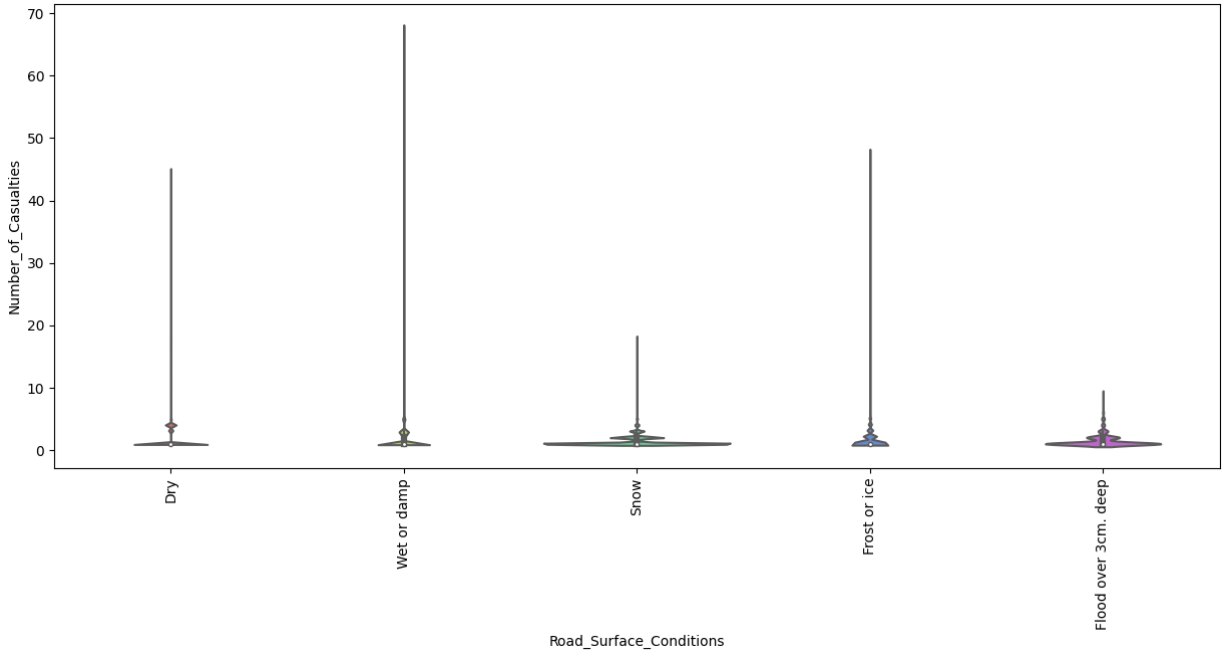


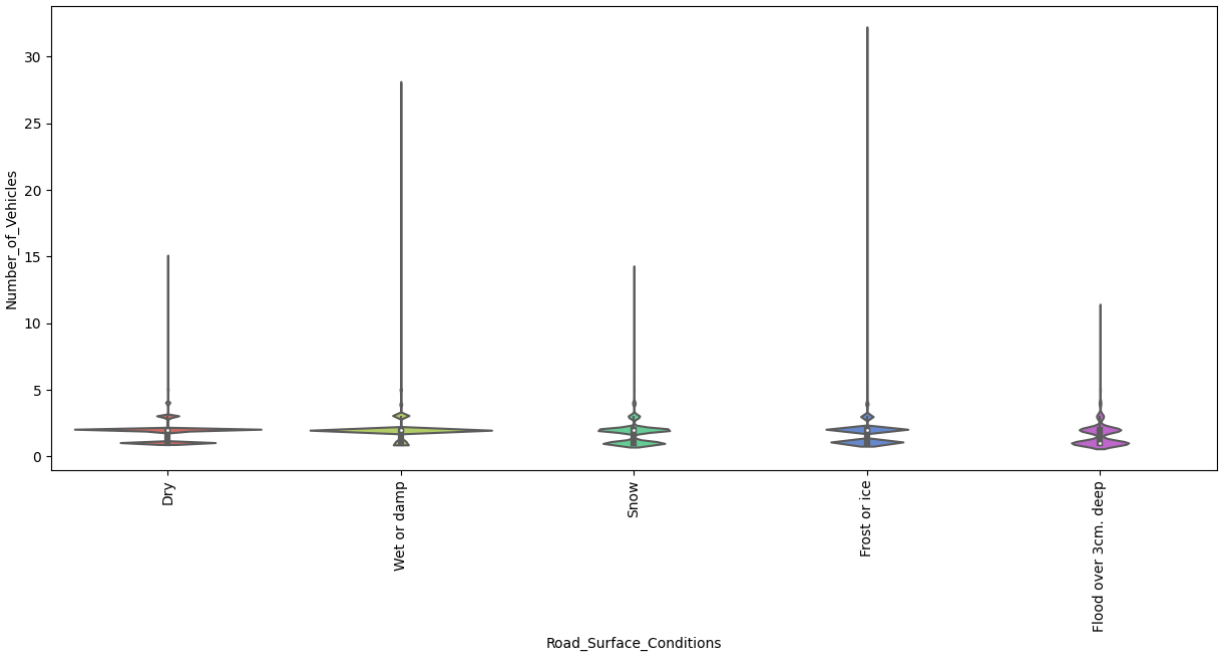


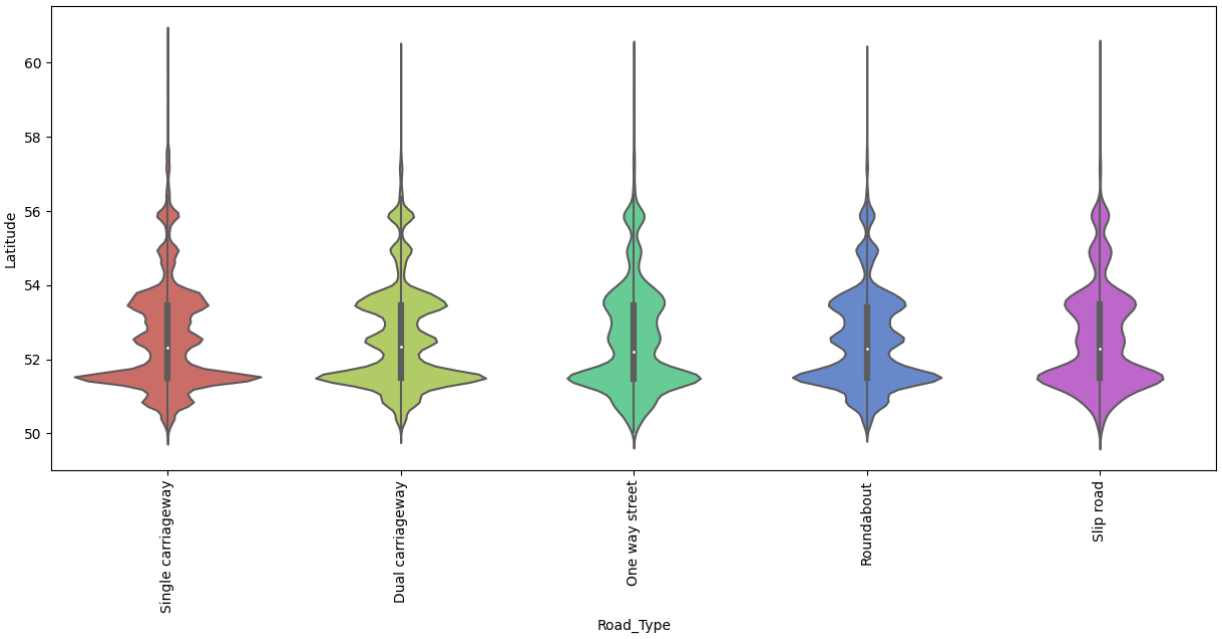


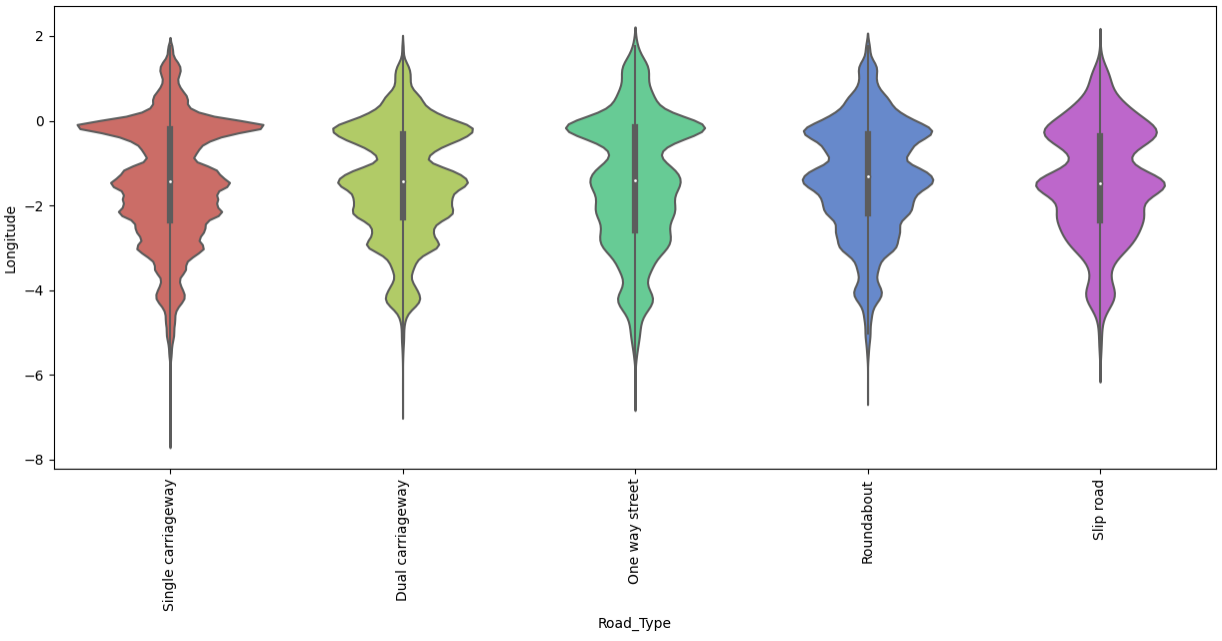


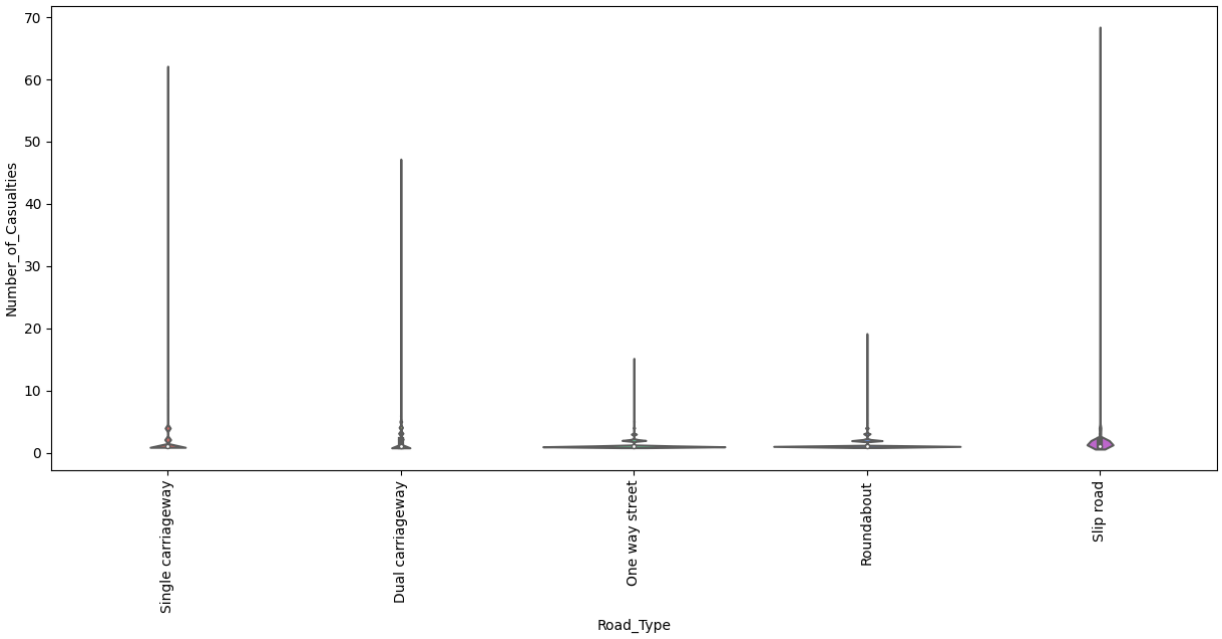


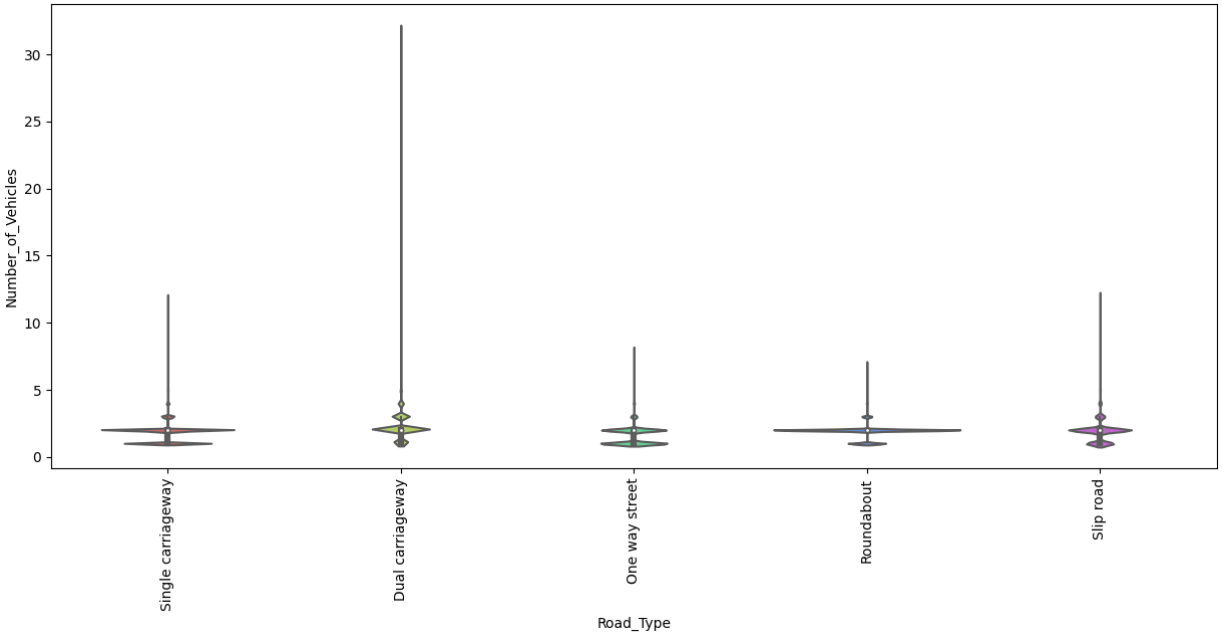


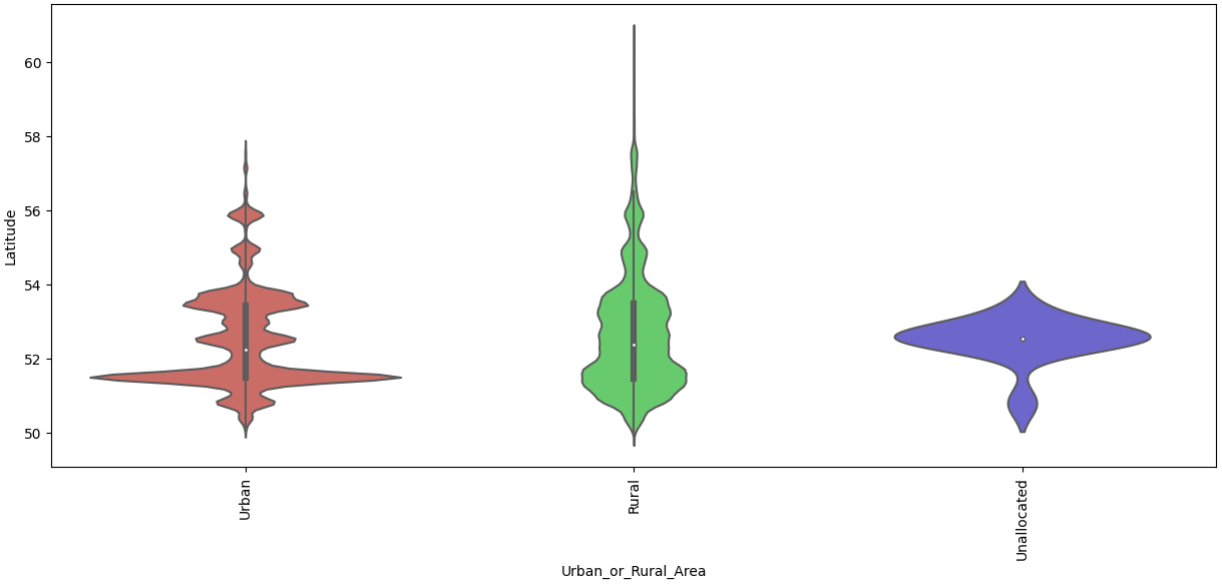


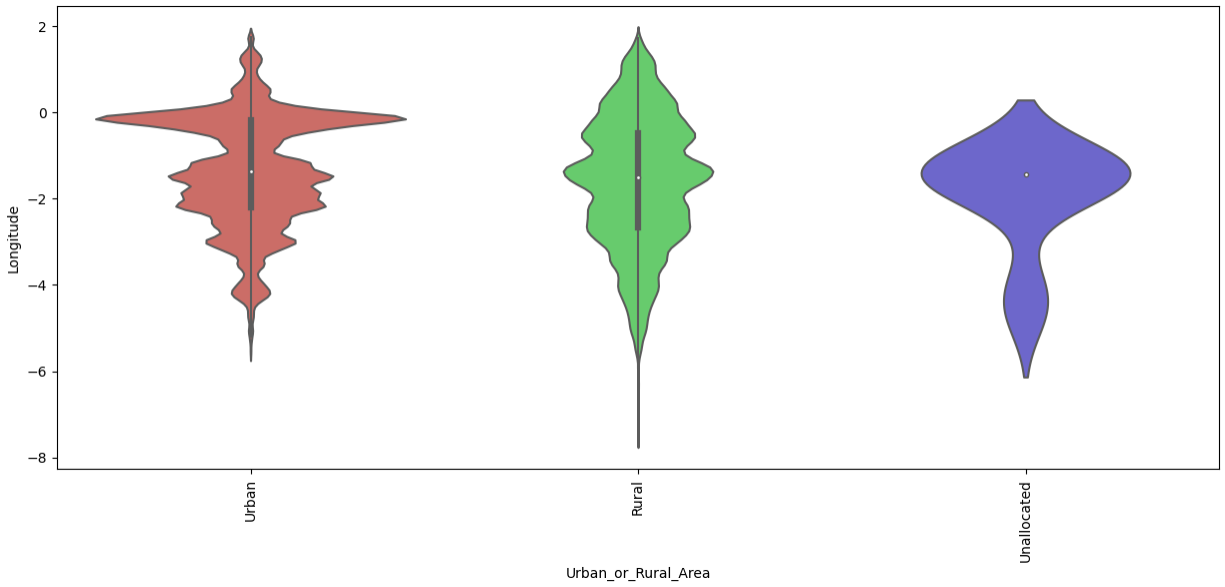


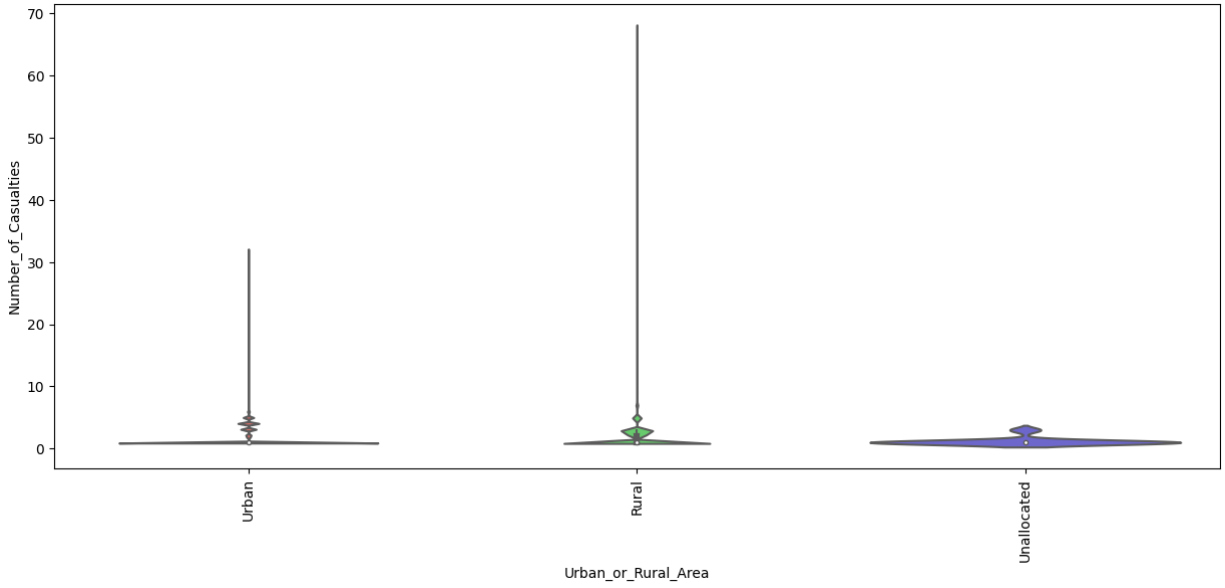


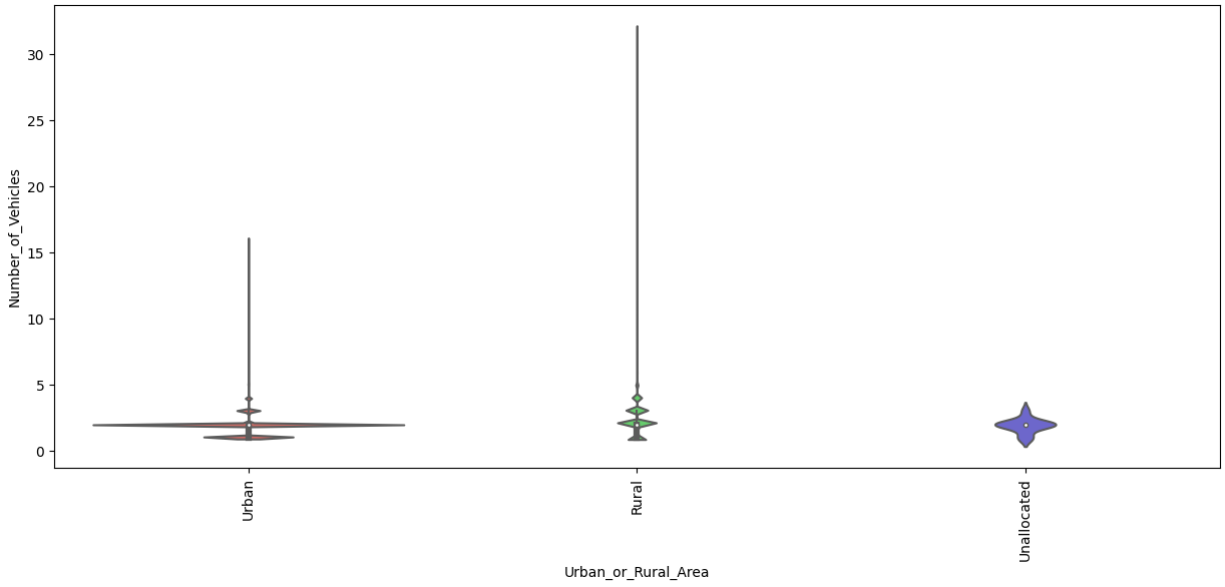


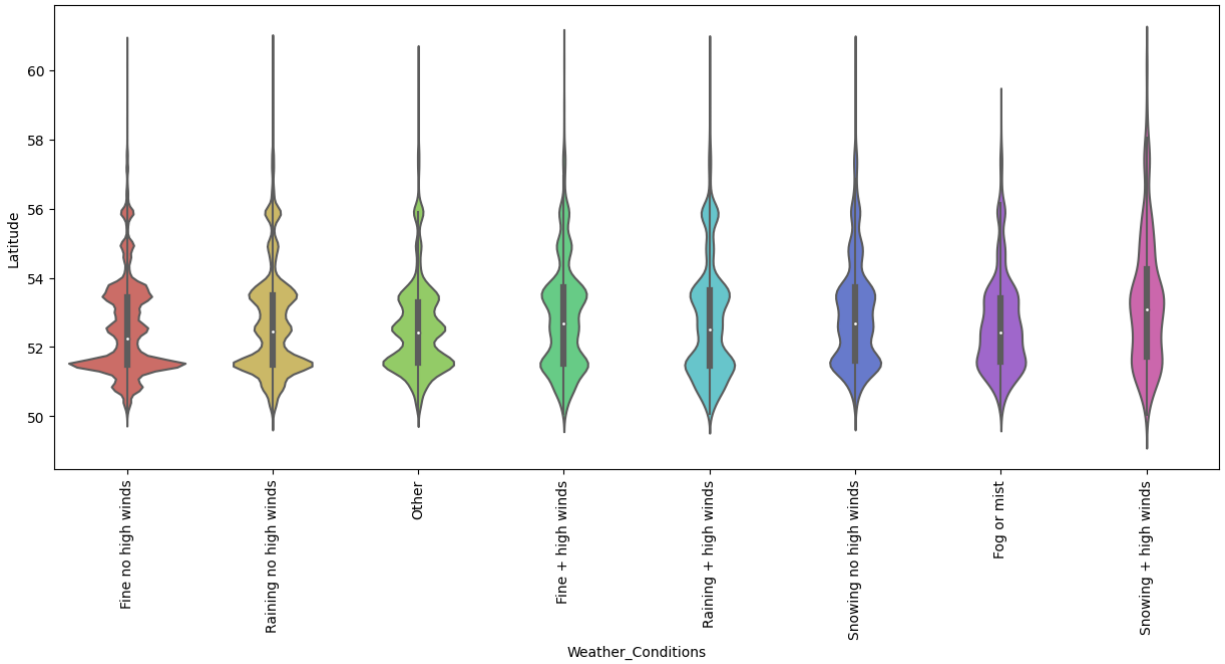


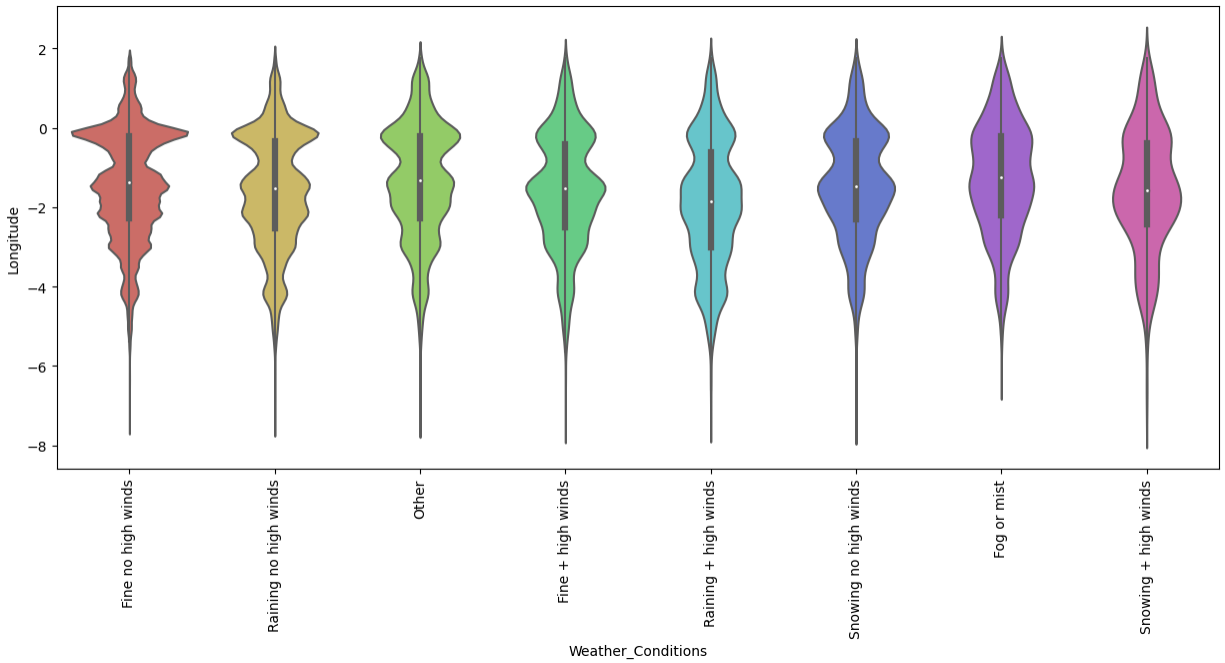


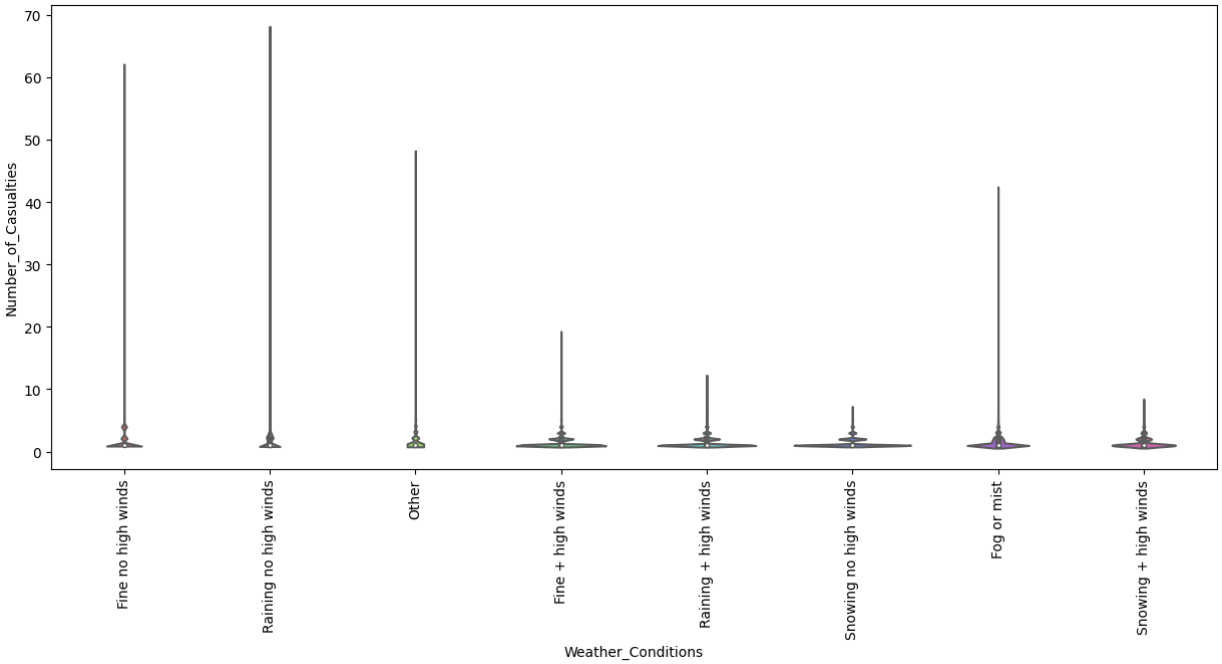


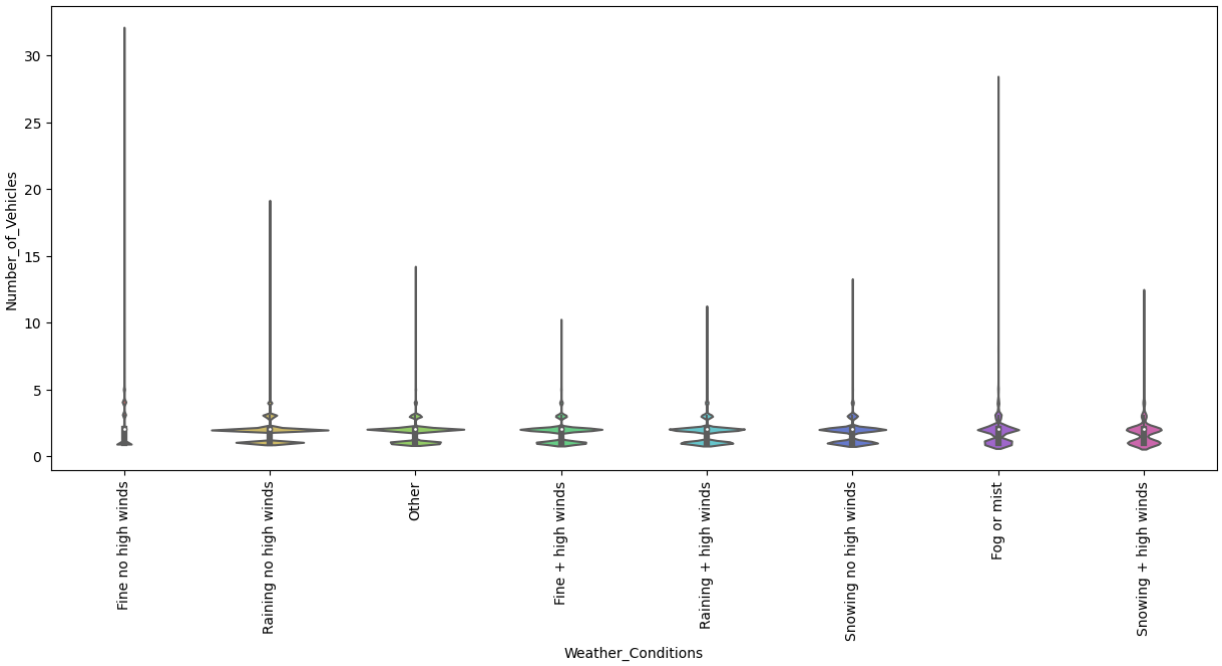












In [32]:
```
correlation_matrix = df.corr()
```

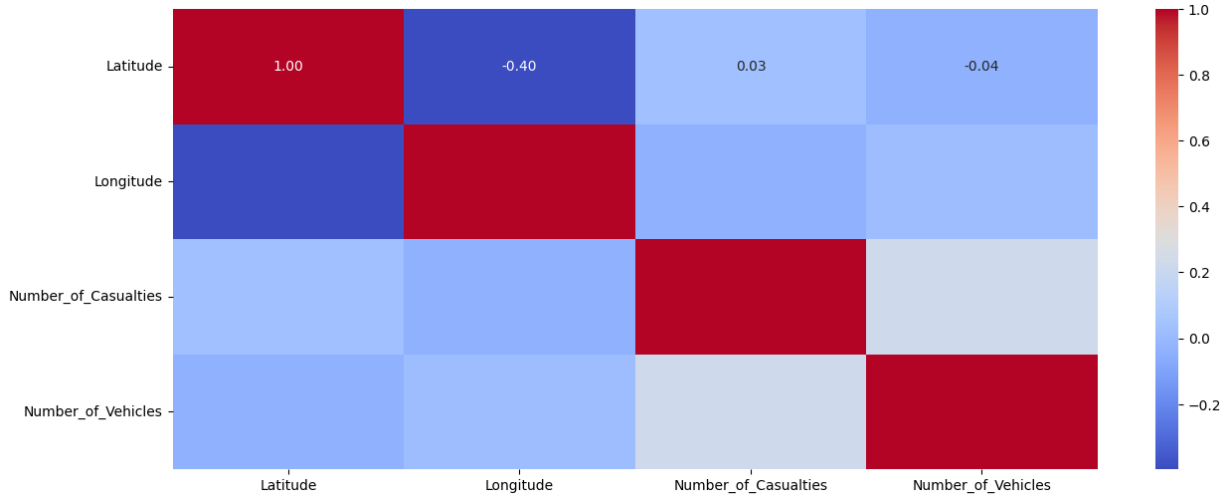
```
C:\Users\tamil\AppData\Local\Temp\ipykernel_28612\4214245630.py:1: FutureWarni
ng: The default value of numeric_only in DataFrame.corr is deprecated. In a fu
ture version, it will default to False. Select only valid columns or specify t
he value of numeric_only to silence this warning.
  correlation_matrix = df.corr()
```

In [33]:
```
correlation_matrix
```

Out[33]:

	Latitude	Longitude	Number_of_Casualties	Number_of_Vehicles
Latitude	1.000000	-0.398116	0.032198	-0.040028
Longitude	-0.398116	1.000000	-0.040404	0.014716
Number_of_Casualties	0.032198	-0.040404	1.000000	0.228889
Number_of_Vehicles	-0.040028	0.014716	0.228889	1.000000

In [34]:
```
plt.figure(figsize=(15, 6))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.show()
```



In [35]:
```
df_new = df.copy()
```

In [36]:
```
df_new
```

Out[36]:

	Index	Accident_Severity	Accident Date	Latitude	Light_Conditions	District Area	Long
0	200701BS64157	Serious	05-06-2019	51.506187	Darkness - lights lit	Kensington and Chelsea	-0.2
1	200701BS65737	Serious	02-07-2019	51.495029	Daylight	Kensington and Chelsea	-0.1
2	200701BS66127	Serious	26-08-2019	51.517715	Darkness - lighting unknown	Kensington and Chelsea	-0.2
3	200701BS66128	Serious	16-08-2019	51.495478	Daylight	Kensington and Chelsea	-0.2
4	200701BS66837	Slight	03-09-2019	51.488576	Darkness - lights lit	Kensington and Chelsea	-0.1
...	...	...	...	...	...	...	
660674	201091NM01760	Slight	18-02-2022	57.374005	Daylight	Highland	-3.4
660675	201091NM01881	Slight	21-02-2022	57.232273	Darkness - no lighting	Highland	-3.8
660676	201091NM01935	Slight	23-02-2022	57.585044	Daylight	Highland	-3.8
660677	201091NM01964	Serious	23-02-2022	57.214898	Darkness - no lighting	Highland	-3.8
660678	201091NM02142	Serious	28-02-2022	57.575210	Daylight	Highland	-3.8

660660 rows × 14 columns

In [37]:
```
df_new['Accident Date'] = pd.to_datetime(df_new['Accident Date'])
```

```
C:\Users\tamil\AppData\Local\Temp\ipykernel_28612\1543151786.py:1: UserWarnin
g: Parsing dates in DD/MM/YYYY format when dayfirst=False (the default) was sp
ecified. This may lead to inconsistently parsed dates! Specify a format to ens
ure consistent parsing.
  df_new['Accident Date'] = pd.to_datetime(df_new['Accident Date'])
```

In [38]:
```
df_new.set_index('Accident Date', inplace=True)
```

In [39]:
```
df_new
```

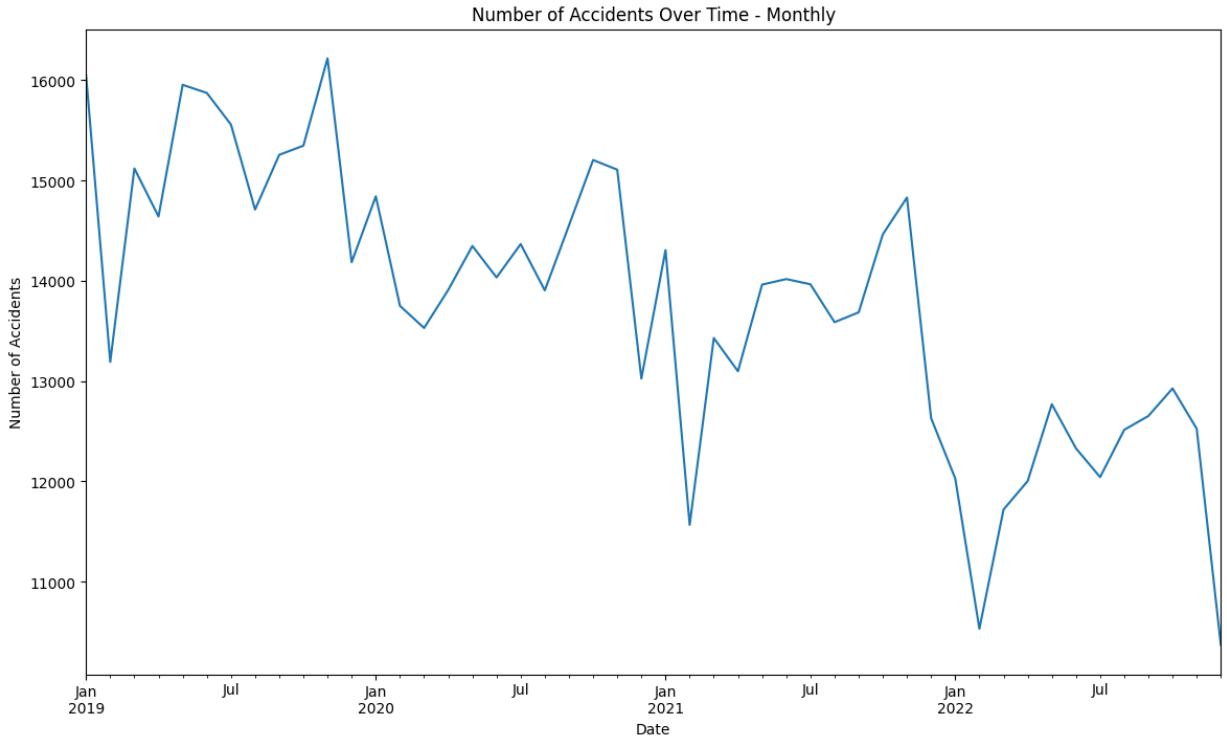
Out[39]:

Accident Date	Index	Accident_Severity	Latitude	Light_Conditions	District Area	Longitude	N
2019-05-06	200701BS64157	Serious	51.506187	Darkness - lights lit	Kensington and Chelsea	-0.209082	
2019-02-07	200701BS65737	Serious	51.495029	Daylight	Kensington and Chelsea	-0.173647	

Accident Date	Index	Accident_Severity	Latitude	Light_Conditions	District Area	Longitude	N
2019-08-26	200701BS66127	Serious	51.517715	Darkness - lighting unknown	Kensington and Chelsea	-0.210215	
2019-08-16	200701BS66128	Serious	51.495478	Daylight	Kensington and Chelsea	-0.202731	
2019-03-09	200701BS66837	Slight	51.488576	Darkness - lights lit	Kensington and Chelsea	-0.192487	
...	...	...	...	...	...	...	
2022-02-18	201091NM01760	Slight	57.374005	Daylight	Highland	-3.467828	
2022-02-21	201091NM01881	Slight	57.232273	Darkness - no lighting	Highland	-3.809281	
2022-02-23	201091NM01935	Slight	57.585044	Daylight	Highland	-3.862727	
2022-02-23	201091NM01964	Serious	57.214898	Darkness - no lighting	Highland	-3.823997	
2022-02-28	201091NM02142	Serious	57.575210	Daylight	Highland	-3.895673	

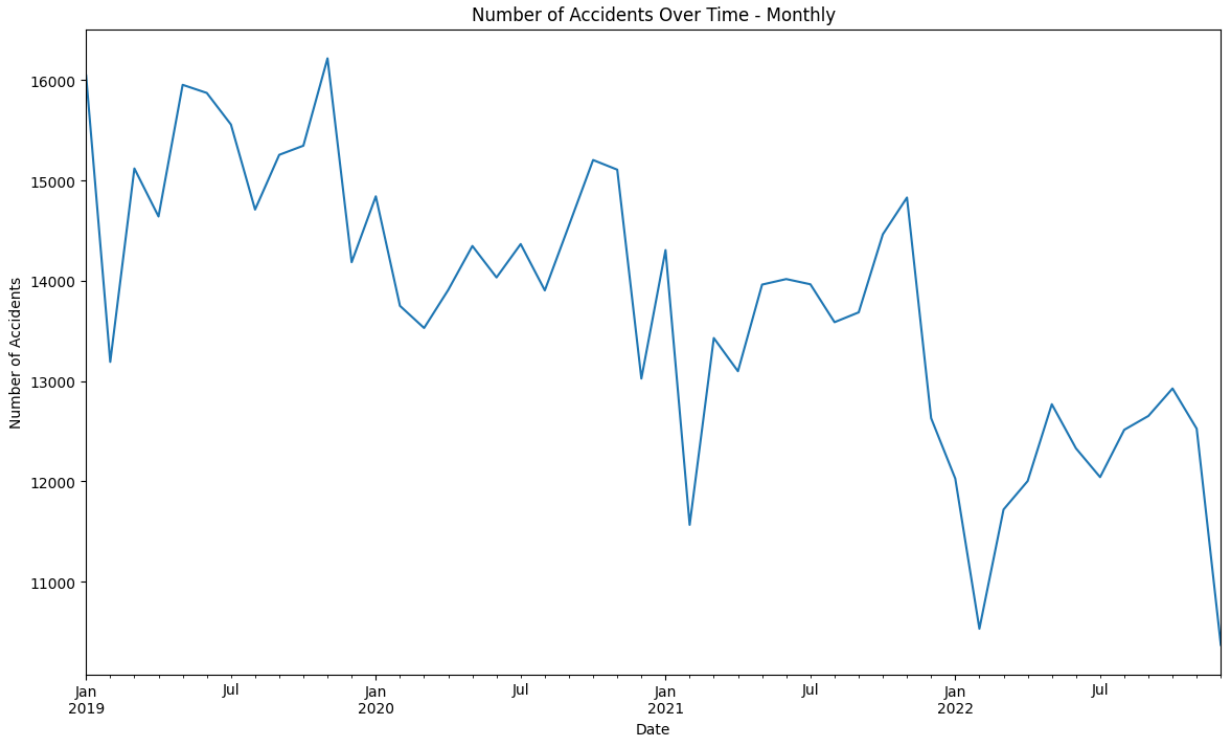
660660 rows × 13 columns

In [40]:
```
plt.figure(figsize=(14, 8))
df_new.resample('M').size().plot(legend=False)
plt.title('Number of Accidents Over Time - Monthly')
plt.xlabel('Date')
plt.ylabel('Number of Accidents')
plt.show()
```

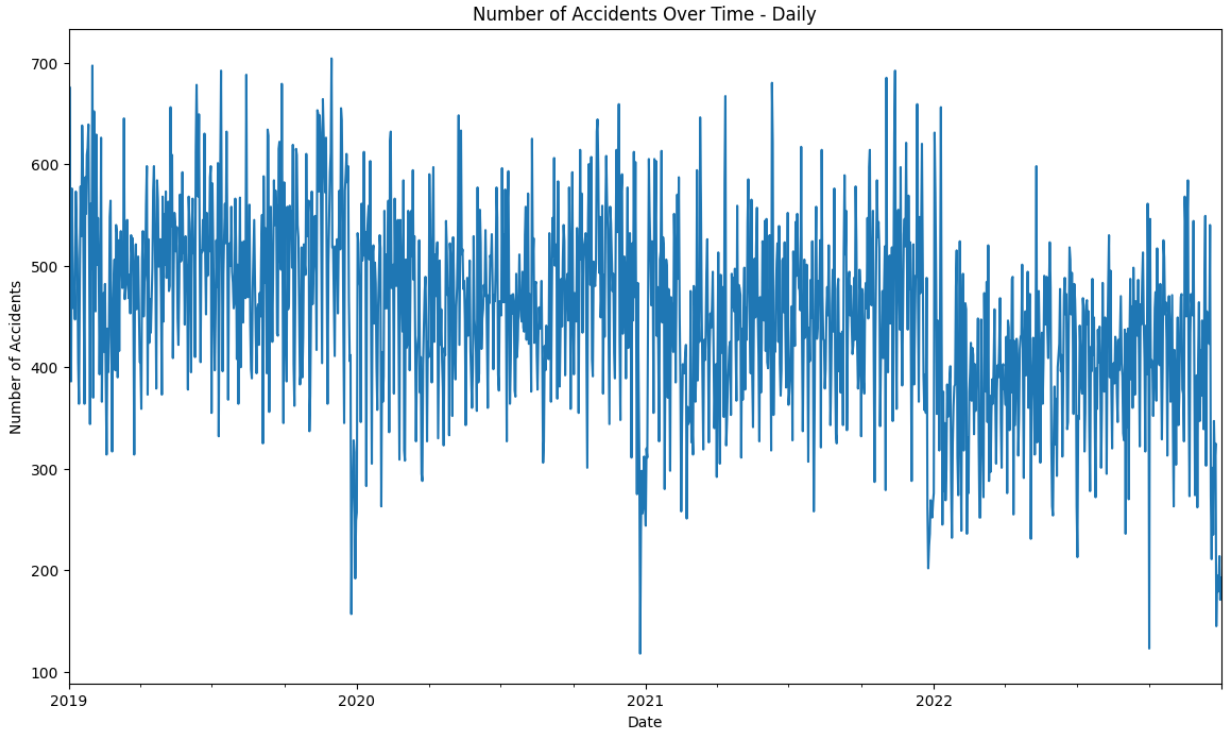


In [41]:
```
df['Accident Date'] = pd.to_datetime(df['Accident Date'])
df_new = df.copy()
df_new.set_index('Accident Date', inplace=True)

# Now plot
plt.figure(figsize=(14, 8))
df_new.resample('M').size().plot(legend=False)
plt.title('Number of Accidents Over Time - Monthly')
plt.xlabel('Date')
plt.ylabel('Number of Accidents')
plt.show()
```

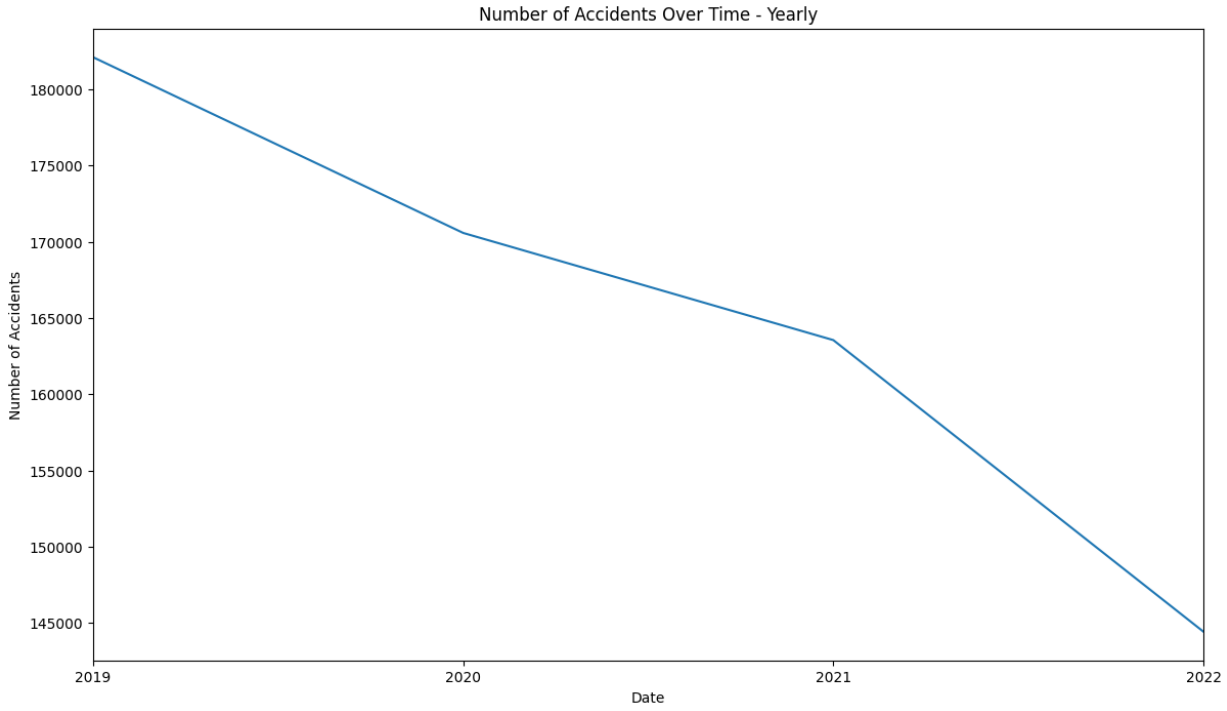
```
C:\Users\tamil\AppData\Local\Temp\ipykernel_28612\950211844.py:1: UserWarning:
Parsing dates in DD/MM/YYYY format when dayfirst=False (the default) was speci
fied. This may lead to inconsistently parsed dates! Specify a format to ensure
consistent parsing.
  df['Accident Date'] = pd.to_datetime(df['Accident Date'])
```



In [42]:
```
plt.figure(figsize=(14, 8))
df_new.resample('D').size().plot(legend=False)
plt.title('Number of Accidents Over Time - Daily')
plt.xlabel('Date')
plt.ylabel('Number of Accidents')
plt.show()
```



In [43]:
```
plt.figure(figsize=(14, 8))
df_new.resample('Y').size().plot(legend=False)
plt.title('Number of Accidents Over Time - Yearly')
plt.xlabel('Date')
plt.ylabel('Number of Accidents')
plt.show()
```



In [44]:
```
pivot_table_result = pd.pivot_table(df, values='Number_of_Casualties', index='
pivot_table_result
```

Out[44]:

Accident_Severity	Fatal	Serious	Slight
Accident Date			
2019-01-01	19	74	401
2019-01-02	9	134	774
2019-01-03	17	89	557
2019-01-04	16	107	437
2019-01-05	20	93	618
...	...	...	...
2022-12-27	8	34	224
2022-12-28	12	39	223
2022-12-29	2	46	255
2022-12-30	0	33	209
2022-12-31	8	43	211

1461 rows × 3 columns

In [45]:
```
pivot_table_result = pd.pivot_table(df, values='Number_of_Casualties', index='
pivot_table_result
```

Out[45]:

Accident_Severity	Fatal	Serious	Slight
Accident Date			
2019-01-01	19	74	401
2019-01-02	9	134	774
2019-01-03	17	89	557

Accident_Severity	Fatal	Serious	Slight
Accident Date			
2019-01-04	16	107	437
2019-01-05	20	93	618
...	...	...	...
2022-12-27	8	34	224
2022-12-28	12	39	223
2022-12-29	2	46	255
2022-12-30	0	33	209
2022-12-31	8	43	211

1461 rows × 3 columns

In [46]:
```
pivot_table_result = pd.pivot_table(df, values='Number_of_Vehicles', index='Ac
pivot_table_result
```

Out[46]:

Accident_Severity	Fatal	Serious	Slight
Accident Date			
2019-01-01	11	53	271
2019-01-02	7	79	590
2019-01-03	11	64	439
2019-01-04	10	70	306
2019-01-05	9	67	500
...	...	...	...
2022-12-27	3	18	158
2022-12-28	6	27	148
2022-12-29	1	25	188
2022-12-30	0	21	150
2022-12-31	4	24	165

1461 rows × 3 columns

In [47]:
```
pivot_table_result = pd.pivot_table(df, values='Number_of_Vehicles', index='Ac
pivot_table_result
```

Out[47]:

Accident_Severity	Fatal	Serious	Slight
Accident Date			
2019-01-01	12	76	463
2019-01-02	10	142	1076
2019-01-03	15	106	801
2019-01-04	14	122	557
2019-01-05	17	125	968

Accident_Severity	Fatal	Serious	Slight
Accident Date			
...	...	...	...
2022-12-27	6	34	268
2022-12-28	9	39	257
2022-12-29	2	44	341
2022-12-30	0	32	270
2022-12-31	5	38	293

1461 rows × 3 columns

In [48]:
```
crosstab_result = pd.crosstab(index=df['Accident_Severity'], columns=df['Weath
crosstab_result
```

Out[48]:

Weather_Conditions	Fine + high winds	Fine no high winds	Fog or mist	Other	Raining + high winds	Raining no high winds	Snowing + high winds	Snowing no high winds
Accident_Severity								
Fatal	175	7207	82	165	145	848	3	36
Serious	1245	73285	483	1801	1261	9468	109	565
Slight	7133	454507	2963	15182	8209	69380	772	5636

In [49]:
```
crosstab_result = pd.crosstab(index=df['Accident_Severity'], columns=df['Light
crosstab_result
```

Out[49]:

Light_Conditions	Darkness - lighting unknown	Darkness - lights lit	Darkness - lights unlit	Darkness - no lighting	Daylight
Accident_Severity					
Fatal	68	1860	45	1612	5076
Serious	794	19130	360	7174	60759
Slight	5622	108345	2138	28646	419031

In [50]:
```
crosstab_result = pd.crosstab(index=df['Accident_Severity'], columns=df['Road_
crosstab_result
```

Out[50]:

Road_Surface_Conditions	Dry	Flood over 3cm. deep	Frost or ice	Snow	Wet or damp
Accident_Severity					
Fatal	5790	23	193	35	2620
Serious	61708	152	2007	565	23785
Slight	381038	842	16314	5288	160300

In [51]:
```
crosstab_result = pd.crosstab(index=df['Accident_Severity'], columns=df['Road_
crosstab_result
```

Out[51]:

Road_Type	Dual carriageway	One way street	Roundabout	Single carriageway	Slip road
Accident_Severity					
Fatal	1815	95	142	6560	49
Serious	11746	1655	3665	70540	611
Slight	85855	11809	40182	419555	6381

In [52]:
```
crosstab_result = pd.crosstab(index=df['Accident_Severity'], columns=df['Urban
crosstab_result
```

Out[52]:

Urban_or_Rural_Area	Rural	Unallocated	Urban
Accident_Severity			
Fatal	5601	0	3060
Serious	37312	1	50904
Slight	196061	10	367711

In [53]:
```
groupby_result = df.groupby(['Accident_Severity', 'Weather_Conditions'])['Numb
groupby_result
```

Out[53]:

Weather_Conditions	Fine + high winds	Fine no high winds	Fog or mist	Other	Raining + high winds	Raining no high winds	Snowing + high winds	Snowing no high winds
Accident_Severity								
Fatal	9	62	42	48	8	68	3	7
Serious	10	45	11	18	9	21	6	7
Slight	19	43	14	9	12	47	8	7

In [54]:
```
groupby_result = df.groupby(['Accident_Severity', 'Weather_Conditions'])['Numb
groupby_result
```

Out[54]:

Weather_Conditions	Fine + high winds	Fine no high winds	Fog or mist	Other	Raining + high winds	Raining no high winds	Snowing + high winds	Snowing no high winds
Accident_Severity								
Fatal	1	1	1	1	1	1	2	1
Serious	1	1	1	1	1	1	1	1
Slight	1	1	1	1	1	1	1	1

In [55]:
```
groupby_result = df.groupby(['Accident_Severity', 'Light_Conditions'])['Number
groupby_result
```

Out[55]:

Light_Conditions	Darkness - lighting unknown	Darkness - lights lit	Darkness - lights unlit	Darkness - no lighting	Daylight
Accident_Severity					
Fatal	6	68	6	48	62
Serious	8	15	7	21	45

Light_Conditions	Darkness - lighting unknown	Darkness - lights lit	Darkness - lights unlit	Darkness - no lighting	Daylight
Accident_Severity					
Slight	8	27	8	13	47

In [56]:
```
groupby_result = df.groupby(['Accident_Severity', 'Light_Conditions'])['Number
groupby_result
```

Out[56]:

Light_Conditions	Darkness - lighting unknown	Darkness - lights lit	Darkness - lights unlit	Darkness - no lighting	Daylight
Accident_Severity					
Fatal	1	1	1	1	1
Serious	1	1	1	1	1
Slight	1	1	1	1	1

In [57]:
```
groupby_result = df.groupby(['Accident_Severity', 'Road_Surface_Conditions'])[
groupby_result
```

Out[57]:

Road_Surface_Conditions	Dry	Flood over 3cm. deep	Frost or ice	Snow	Wet or damp
Accident_Severity					
Fatal	40	7	48	7	68
Serious	45	7	13	18	21
Slight	43	9	15	7	47

In [58]:
```
groupby_result = df.groupby(['Accident_Severity', 'Road_Surface_Conditions'])[
groupby_result
```

Out[58]:

Road_Surface_Conditions	Dry	Flood over 3cm. deep	Frost or ice	Snow	Wet or damp
Accident_Severity					
Fatal	1	1	1	1	1
Serious	1	1	1	1	1
Slight	1	1	1	1	1

In [59]:
```
groupby_result = df.groupby(['Accident_Severity', 'Road_Type'])['Number_of_Cas
groupby_result
```

Out[59]:

Road_Type	Dual carriageway	One way street	Roundabout	Single carriageway	Slip road
Accident_Severity					
Fatal	42	8	4	62	68
Serious	45	15	19	42	16
Slight	47	13	17	29	14

In [60]:
```
groupby_result = df.groupby(['Accident_Severity', 'Road_Type'])['Number_of_Cas
groupby_result
```

Out[60]:

Road_Type	Dual carriageway	One way street	Roundabout	Single carriageway	Slip road
Accident_Severity					
Fatal	1	1	1	1	1
Serious	1	1	1	1	1
Slight	1	1	1	1	1

In [61]:
```
groupby_result = df.groupby(['Accident_Severity', 'Urban_or_Rural_Area'])['Num
groupby_result
```

Out[61]:

Urban_or_Rural_Area	Rural	Unallocated	Urban
Accident_Severity			
Fatal	68.0	NaN	21.0
Serious	45.0	1.0	29.0
Slight	47.0	3.0	32.0

In [62]:
```
groupby_result = df.groupby(['Accident_Severity', 'Urban_or_Rural_Area'])['Num
groupby_result
```

Out[62]:

Urban_or_Rural_Area	Rural	Unallocated	Urban
Accident_Severity			
Fatal	1.0	NaN	1.0
Serious	1.0	1.0	1.0
Slight	1.0	1.0	1.0

In [63]:
```
df
```

Out[63]:

	Index	Accident_Severity	Accident Date	Latitude	Light_Conditions	District Area	Long
0	200701BS64157	Serious	2019-05-06	51.506187	Darkness - lights lit	Kensington and Chelsea	-0.2
1	200701BS65737	Serious	2019-02-07	51.495029	Daylight	Kensington and Chelsea	-0.1
2	200701BS66127	Serious	2019-08-26	51.517715	Darkness - lighting unknown	Kensington and Chelsea	-0.2
3	200701BS66128	Serious	2019-08-16	51.495478	Daylight	Kensington and Chelsea	-0.2
4	200701BS66837	Slight	2019-03-09	51.488576	Darkness - lights lit	Kensington and Chelsea	-0.1
...	...	...	...	...	...	...	
660674	201091NM01760	Slight	2022-02-18	57.374005	Daylight	Highland	-3.4
660675	201091NM01881	Slight	2022-02-	57.232273	Darkness - no	Highland	-3.8

	Index	Accident_Severity	Accident Date	Latitude	Light_Conditions	District Area	Long
			21		lighting		
660676	201091NM01935	Slight	2022-02-23	57.585044	Daylight	Highland	-3.8
660677	201091NM01964	Serious	2022-02-23	57.214898	Darkness - no lighting	Highland	-3.8
660678	201091NM02142	Serious	2022-02-28	57.575210	Daylight	Highland	-3.8

660660 rows × 14 columns

In [67]:
```
df = df.drop(['Index','Accident Date','District Area'],axis=1)
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
Cell In[67], line 1
----> 1 df = df.drop(['Index','Accident Date','District Area'],axis=1)

File D:\Users\tamil\anaconda3\Lib\site-packages\pandas\util\_decorators.py:331, in deprecate_nonkeyword_arguments.<locals>.decorate.<locals>.wrapper(*args, **kwargs)
    325 if len(args) > num_allow_args:
    326     warnings.warn(
    327         msg.format(arguments=_format_argument_list(allow_args)),
    328         FutureWarning,
    329         stacklevel=find_stack_level(),
    330     )
--> 331 return func(*args, **kwargs)

File D:\Users\tamil\anaconda3\Lib\site-packages\pandas\core\frame.py:5399, in DataFrame.drop(self, labels, axis, index, columns, level, inplace, errors)
   5251 @deprecate_nonkeyword_arguments(version=None, allowed_args=["self", "labels"])
   5252 def drop(  # type: ignore[override]
   5253     self,
   (...)
   5260     errors: IgnoreRaise = "raise",
   5261 ) -> DataFrame | None:
   5262     """
   5263     Drop specified labels from rows or columns.
   5264
   (...)
   5397             weight  1.0     0.8
   5398     """
-> 5399     return super().drop(
   5400         labels=labels,
   5401         axis=axis,
   5402         index=index,
   5403         columns=columns,
   5404         level=level,
   5405         inplace=inplace,
   5406         errors=errors,
   5407     )

File D:\Users\tamil\anaconda3\Lib\site-packages\pandas\util\_decorators.py:331, in deprecate_nonkeyword_arguments.<locals>.decorate.<locals>.wrapper(*args, **kwargs)
```

```
    325 if len(args) > num_allow_args:
    326     warnings.warn(
    327         msg.format(arguments=_format_argument_list(allow_args)),
    328         FutureWarning,
    329         stacklevel=find_stack_level(),
    330     )
--> 331 return func(*args, **kwargs)

File D:\Users\tamil\anaconda3\Lib\site-packages\pandas\core\generic.py:4505, i
n NDFrame.drop(self, labels, axis, index, columns, level, inplace, errors)
   4503 for axis, labels in axes.items():
   4504     if labels is not None:
-> 4505         obj = obj._drop_axis(labels, axis, level=level, errors=errors)
   4507 if inplace:
   4508     self._update_inplace(obj)

File D:\Users\tamil\anaconda3\Lib\site-packages\pandas\core\generic.py:4546, i
n NDFrame._drop_axis(self, labels, axis, level, errors, only_slice)
   4544         new_axis = axis.drop(labels, level=level, errors=errors)
   4545     else:
-> 4546         new_axis = axis.drop(labels, errors=errors)
   4547     indexer = axis.get_indexer(new_axis)
   4549 # Case for non-unique axis
   4550 else:

File D:\Users\tamil\anaconda3\Lib\site-packages\pandas\core\indexes\base.py:69
34, in Index.drop(self, labels, errors)
   6932 if mask.any():
   6933     if errors != "ignore":
-> 6934         raise KeyError(f"{list(labels[mask])} not found in axis")
   6935     indexer = indexer[~mask]
   6936 return self.delete(indexer)

KeyError: "['Index', 'Accident Date', 'District Area'] not found in axis"
```

In [68]:
```
df = df.drop(['Index', 'Accident Date', 'District Area'], axis=1, errors='igno
```

In [77]:
```
print(df.columns)
```

```
Index(['Accident_Severity', 'Number_of_Casualties', 'Number_of_Vehicles',
       'Light_Conditions_Darkness - lighting unknown',
       'Light_Conditions_Darkness - lights lit',
       'Light_Conditions_Darkness - lights unlit',
       'Light_Conditions_Darkness - no lighting', 'Light_Conditions_Daylight',
       'Road_Surface_Conditions_Dry',
       'Road_Surface_Conditions_Flood over 3cm. deep',
       'Road_Surface_Conditions_Frost or ice', 'Road_Surface_Conditions_Snow',
       'Road_Surface_Conditions_Wet or damp', 'Road_Type_Dual carriageway',
       'Road_Type_One way street', 'Road_Type_Roundabout',
       'Road_Type_Single carriageway', 'Road_Type_Slip road',
       'Urban_or_Rural_Area_Rural', 'Urban_or_Rural_Area_Unallocated',
       'Urban_or_Rural_Area_Urban', 'Vehicle_Type_Agricultural vehicle',
       'Vehicle_Type_Bus or coach (17 or more pass seats)', 'Vehicle_Type_Ca
r',
       'Vehicle_Type_Data missing or out of range',
       'Vehicle_Type_Goods 7.5 tonnes mgw and over',
       'Vehicle_Type_Goods over 3.5t. and under 7.5t',
       'Vehicle_Type_Minibus (8 - 16 passenger seats)',
       'Vehicle_Type_Motorcycle 125cc and under',
       'Vehicle_Type_Motorcycle 50cc and under',
       'Vehicle_Type_Motorcycle over 125cc and up to 500cc',
```

```
       'Vehicle_Type_Motorcycle over 500cc', 'Vehicle_Type_Other vehicle',
       'Vehicle_Type_Pedal cycle', 'Vehicle_Type_Ridden horse',
       'Vehicle_Type_Taxi/Private hire car',
       'Vehicle_Type_Van / Goods 3.5 tonnes mgw or under',
       'Weather_Conditions_Fine + high winds',
       'Weather_Conditions_Fine no high winds',
       'Weather_Conditions_Fog or mist', 'Weather_Conditions_Other',
       'Weather_Conditions_Raining + high winds',
       'Weather_Conditions_Raining no high winds',
       'Weather_Conditions_Snowing + high winds',
       'Weather_Conditions_Snowing no high winds'],
      dtype='object')
```

In [78]:
```
label_mapping = {'Slight': 0, 'Serious': 1, 'Fatal': 2}
df['Accident_Severity'] = df['Accident_Severity'].map(label_mapping)
```

In [79]:
```
df = pd.get_dummies(df,columns = ['Light_Conditions','Road_Surface_Conditions'
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
Cell In[79], line 1
----> 1 df = pd.get_dummies(df,columns = ['Light_Conditions','Road_Surface_Con
ditions','Road_Type','Urban_or_Rural_Area','Vehicle_Type','Weather_Condition
s'])

File D:\Users\tamil\anaconda3\Lib\site-packages\pandas\core\reshape\encoding.p
y:146, in get_dummies(data, prefix, prefix_sep, dummy_na, columns, sparse, dro
p_first, dtype)
    144     raise TypeError("Input must be a list-like for parameter `columns
`")
    145 else:
--> 146     data_to_encode = data[columns]
    148 # validate prefixes and separator to avoid silently dropping cols
    149 def check_len(item, name):

File D:\Users\tamil\anaconda3\Lib\site-packages\pandas\core\frame.py:3813, in
DataFrame.__getitem__(self, key)
   3811     if is_iterator(key):
   3812         key = list(key)
-> 3813     indexer = self.columns._get_indexer_strict(key, "columns")[1]
   3815 # take() does not accept boolean indexers
   3816 if getattr(indexer, "dtype", None) == bool:

File D:\Users\tamil\anaconda3\Lib\site-packages\pandas\core\indexes\base.py:60
70, in Index._get_indexer_strict(self, key, axis_name)
   6067 else:
   6068     keyarr, indexer, new_indexer = self._reindex_non_unique(keyarr)
-> 6070 self._raise_if_missing(keyarr, indexer, axis_name)
   6072 keyarr = self.take(indexer)
   6073 if isinstance(key, Index):
   6074     # GH 42790 - Preserve name from an Index

File D:\Users\tamil\anaconda3\Lib\site-packages\pandas\core\indexes\base.py:61
30, in Index._raise_if_missing(self, key, indexer, axis_name)
   6128     if use_interval_msg:
   6129         key = list(key)
-> 6130     raise KeyError(f"None of [{key}] are in the [{axis_name}]")
   6132 not_found = list(ensure_index(key)[missing_mask.nonzero()[0]].unique
())
   6133 raise KeyError(f"{not_found} not in index")
```

```
KeyError: "None of [Index(['Light_Conditions', 'Road_Surface_Conditions', 'Roa
d_Type',\n       'Urban_or_Rural_Area', 'Vehicle_Type', 'Weather_Condition
s'],\n      dtype='object')] are in the [columns]"
```

In [75]:

```
# Check if the columns are present
print(df.columns)

# Strip extra whitespaces if any
df.columns = df.columns.str.strip()

# Check for missing columns
required_columns = ['Light_Conditions', 'Road_Surface_Conditions', 'Road_Type'
                    'Urban_or_Rural_Area', 'Vehicle_Type', 'Weather_Conditions

missing_columns = [col for col in required_columns if col not in df.columns]

if missing_columns:
    print(f"Missing columns: {missing_columns}")
else:
    # Apply pd.get_dummies if all columns are present
    df = pd.get_dummies(df, columns=required_columns)

# Print the updated DataFrame columns after applying get_dummies
print(df.columns)
```

```
Index(['Accident_Severity', 'Number_of_Casualties', 'Number_of_Vehicles',
       'Light_Conditions_Darkness - lighting unknown',
       'Light_Conditions_Darkness - lights lit',
       'Light_Conditions_Darkness - lights unlit',
       'Light_Conditions_Darkness - no lighting', 'Light_Conditions_Daylight',
       'Road_Surface_Conditions_Dry',
       'Road_Surface_Conditions_Flood over 3cm. deep',
       'Road_Surface_Conditions_Frost or ice', 'Road_Surface_Conditions_Snow',
       'Road_Surface_Conditions_Wet or damp', 'Road_Type_Dual carriageway',
       'Road_Type_One way street', 'Road_Type_Roundabout',
       'Road_Type_Single carriageway', 'Road_Type_Slip road',
       'Urban_or_Rural_Area_Rural', 'Urban_or_Rural_Area_Unallocated',
       'Urban_or_Rural_Area_Urban', 'Vehicle_Type_Agricultural vehicle',
       'Vehicle_Type_Bus or coach (17 or more pass seats)', 'Vehicle_Type_Ca
r',
       'Vehicle_Type_Data missing or out of range',
       'Vehicle_Type_Goods 7.5 tonnes mgw and over',
       'Vehicle_Type_Goods over 3.5t. and under 7.5t',
       'Vehicle_Type_Minibus (8 - 16 passenger seats)',
       'Vehicle_Type_Motorcycle 125cc and under',
       'Vehicle_Type_Motorcycle 50cc and under',
       'Vehicle_Type_Motorcycle over 125cc and up to 500cc',
       'Vehicle_Type_Motorcycle over 500cc', 'Vehicle_Type_Other vehicle',
       'Vehicle_Type_Pedal cycle', 'Vehicle_Type_Ridden horse',
       'Vehicle_Type_Taxi/Private hire car',
       'Vehicle_Type_Van / Goods 3.5 tonnes mgw or under',
       'Weather_Conditions_Fine + high winds',
       'Weather_Conditions_Fine no high winds',
       'Weather_Conditions_Fog or mist', 'Weather_Conditions_Other',
       'Weather_Conditions_Raining + high winds',
       'Weather_Conditions_Raining no high winds',
       'Weather_Conditions_Snowing + high winds',
       'Weather_Conditions_Snowing no high winds'],
      dtype='object')
Missing columns: ['Light_Conditions', 'Road_Surface_Conditions', 'Road_Type',
'Urban_or_Rural_Area', 'Vehicle_Type', 'Weather_Conditions']
Index(['Accident_Severity', 'Number_of_Casualties', 'Number_of_Vehicles',
```

```
       'Light_Conditions_Darkness - lighting unknown',
       'Light_Conditions_Darkness - lights lit',
       'Light_Conditions_Darkness - lights unlit',
       'Light_Conditions_Darkness - no lighting', 'Light_Conditions_Daylight',
       'Road_Surface_Conditions_Dry',
       'Road_Surface_Conditions_Flood over 3cm. deep',
       'Road_Surface_Conditions_Frost or ice', 'Road_Surface_Conditions_Snow',
       'Road_Surface_Conditions_Wet or damp', 'Road_Type_Dual carriageway',
       'Road_Type_One way street', 'Road_Type_Roundabout',
       'Road_Type_Single carriageway', 'Road_Type_Slip road',
       'Urban_or_Rural_Area_Rural', 'Urban_or_Rural_Area_Unallocated',
       'Urban_or_Rural_Area_Urban', 'Vehicle_Type_Agricultural vehicle',
       'Vehicle_Type_Bus or coach (17 or more pass seats)', 'Vehicle_Type_Ca
r',
       'Vehicle_Type_Data missing or out of range',
       'Vehicle_Type_Goods 7.5 tonnes mgw and over',
       'Vehicle_Type_Goods over 3.5t. and under 7.5t',
       'Vehicle_Type_Minibus (8 - 16 passenger seats)',
       'Vehicle_Type_Motorcycle 125cc and under',
       'Vehicle_Type_Motorcycle 50cc and under',
       'Vehicle_Type_Motorcycle over 125cc and up to 500cc',
       'Vehicle_Type_Motorcycle over 500cc', 'Vehicle_Type_Other vehicle',
       'Vehicle_Type_Pedal cycle', 'Vehicle_Type_Ridden horse',
       'Vehicle_Type_Taxi/Private hire car',
       'Vehicle_Type_Van / Goods 3.5 tonnes mgw or under',
       'Weather_Conditions_Fine + high winds',
       'Weather_Conditions_Fine no high winds',
       'Weather_Conditions_Fog or mist', 'Weather_Conditions_Other',
       'Weather_Conditions_Raining + high winds',
       'Weather_Conditions_Raining no high winds',
       'Weather_Conditions_Snowing + high winds',
       'Weather_Conditions_Snowing no high winds'],
      dtype='object')
```

In [76]:
```
df
```

Out[76]:

	Accident_Severity	Number_of_Casualties	Number_of_Vehicles	Light_Conditions_Darkness - lighting unknown
0	NaN	1	2	0
1	NaN	1	2	0
2	NaN	1	3	1
3	NaN	1	4	0
4	NaN	1	2	0
...	...	...	...	...
660674	NaN	2	1	0
660675	NaN	1	1	0
660676	NaN	1	3	0
660677	NaN	1	2	0
660678	NaN	1	1	0

660660 rows × 45 columns

In [89]:
```
from sklearn.model_selection import train_test_split
X = df.drop('Accident_Severity', axis=1)
y = df['Accident_Severity']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, strat
```

In [90]:
```
from sklearn.linear_model import LogisticRegression
```

In [91]:
```
model = LogisticRegression(random_state=42)
```

In [92]:
```
model.fit(X_train,y_train)
```

Out[92]:
```
LogisticRegression(random_state=42)
```

In [93]:
```
y_pred = model.predict(X_test)
```

In [94]:
```
from sklearn.metrics import accuracy_score, classification_report, confusion_m
```

In [95]:
```
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
classification_rep = classification_report(y_test, y_pred)
print(f'Accuracy: {accuracy:.2f}')
print(f'Confusion Matrix:\n{conf_matrix}')
print(f'Classification Report:\n{classification_rep}')
```

```
Accuracy: 0.85
Confusion Matrix:
[[169083     43      9]
 [ 26430     28      7]
 [  2580     11      7]]
Classification Report:
              precision    recall  f1-score   support

           0       0.85      1.00      0.92    169135
           1       0.34      0.00      0.00     26465
           2       0.30      0.00      0.01      2598

    accuracy                           0.85    198198
   macro avg       0.50      0.33      0.31    198198
weighted avg       0.78      0.85      0.79    198198
```

In [75]:
```
from sklearn.ensemble import RandomForestClassifier

# Initialize Random Forest model
forest = RandomForestClassifier(n_estimators=100, max_depth=5, random_state=42
```

In [76]:
```
# Fit model
forest.fit(X_train, y_train)
```

Out[76]:
```
RandomForestClassifier(max_depth=5, random_state=42)
```

In [77]:
```
# Predict and calculate accuracy
y_pred = forest.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"Random Forest Accuracy: {accuracy}")
```

```
Random Forest Accuracy: 0.8533663306390579
```

In [78]:
```
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
classification_rep = classification_report(y_test, y_pred)
print(f'Accuracy: {accuracy:.2f}')
print(f'Confusion Matrix:\n{conf_matrix}')
print(f'Classification Report:\n{classification_rep}')
```

```
Accuracy: 0.85
Confusion Matrix:
[[112757      0      0]
 [ 17643      0      0]
 [  1732      0      0]]
Classification Report:
              precision    recall  f1-score   support

           0       0.85      1.00      0.92    112757
           1       0.00      0.00      0.00     17643
           2       0.00      0.00      0.00      1732

    accuracy                           0.85    132132
   macro avg       0.28      0.33      0.31    132132
weighted avg       0.73      0.85      0.79    132132
```

```
D:\Users\tamil\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:
1497: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in
labels with no predicted samples. Use `zero_division` parameter to control thi
s behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
D:\Users\tamil\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:
1497: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in
labels with no predicted samples. Use `zero_division` parameter to control thi
s behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
D:\Users\tamil\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:
1497: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in
labels with no predicted samples. Use `zero_division` parameter to control thi
s behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```

In [79]:
```
from sklearn.tree import DecisionTreeClassifier

# Initialize Decision Tree model
tree = DecisionTreeClassifier(max_depth=5)
```

In [80]:
```
# Fit model
tree.fit(X_train, y_train)

# Predict and calculate accuracy
y_pred = tree.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"Decision Tree Accuracy: {accuracy}")
```

```
Decision Tree Accuracy: 0.853328489692126
```

In [81]:
```
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
classification_rep = classification_report(y_test, y_pred)
print(f'Accuracy: {accuracy:.2f}')
print(f'Confusion Matrix:\n{conf_matrix}')
print(f'Classification Report:\n{classification_rep}')
```

```
Accuracy: 0.85
Confusion Matrix:
[[112744     13      0]
 [ 17635      8      0]
 [  1728      4      0]]
Classification Report:
              precision    recall  f1-score   support

           0       0.85      1.00      0.92    112757
           1       0.32      0.00      0.00     17643
           2       0.00      0.00      0.00      1732

    accuracy                           0.85    132132
   macro avg       0.39      0.33      0.31    132132
weighted avg       0.77      0.85      0.79    132132
```

```
D:\Users\tamil\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:
1497: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in
labels with no predicted samples. Use `zero_division` parameter to control thi
s behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
D:\Users\tamil\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:
1497: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in
labels with no predicted samples. Use `zero_division` parameter to control thi
s behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
D:\Users\tamil\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:
1497: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in
labels with no predicted samples. Use `zero_division` parameter to control thi
s behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```

In [ ]:

In [76]:
```
from sklearn.neighbors import KNeighborsClassifier

# Initialize the model
model = KNeighborsClassifier(n_neighbors=5)  # k=5
```

In [77]:
```
# Train the model
model.fit(X_train, y_train)
```

Out[77]:
```
▾ KNeighborsClassifier
KNeighborsClassifier()
```

In [78]:
```
# Make predictions
y_pred = model.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
print(f"KNN Accuracy: {accuracy}")
```

```
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))
```

```
KNN Accuracy: 0.8425589561953198
Confusion Matrix:
 [[110738   2014      5]
 [ 17050    590      3]
 [  1633     98      1]]
Classification Report:
              precision    recall  f1-score   support

           0       0.86      0.98      0.91    112757
           1       0.22      0.03      0.06     17643
           2       0.11      0.00      0.00      1732

    accuracy                           0.84    132132
   macro avg       0.40      0.34      0.32    132132
weighted avg       0.76      0.84      0.79    132132
```

In [80]:
```
from sklearn.ensemble import GradientBoostingClassifier

# Initialize the model
model = GradientBoostingClassifier(n_estimators=100, learning_rate=0.1, max_de
```

In [81]:
```
# Train the model
model.fit(X_train, y_train)
```

Out[81]:
```
GradientBoostingClassifier
GradientBoostingClassifier()
```

In [82]:
```
# Make predictions
y_pred = model.predict(X_test)
```

In [83]:
```
# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
print(f"Gradient Boosting Accuracy: {accuracy}")
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))
```

```
Gradient Boosting Accuracy: 0.8533057851239669
Confusion Matrix:
 [[112729     27      1]
 [ 17623     20      0]
 [  1716     16      0]]
Classification Report:
              precision    recall  f1-score   support

           0       0.85      1.00      0.92    112757
           1       0.32      0.00      0.00     17643
           2       0.00      0.00      0.00      1732

    accuracy                           0.85    132132
   macro avg       0.39      0.33      0.31    132132
weighted avg       0.77      0.85      0.79    132132
```

In [84]:
```
from sklearn.neural_network import MLPClassifier
```

```
# Initialize the model
model = MLPClassifier(hidden_layer_sizes=(50, 30), max_iter=500, random_state=
```

In [85]:
```
# Train the model
model.fit(X_train, y_train)
```

Out[85]:
```
MLPClassifier(hidden_layer_sizes=(50, 30), max_iter=500, random_state
=42)
```

In [86]:
```
# Make predictions
y_pred = model.predict(X_test)
```

In [87]:
```
# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
print(f"Neural Network (MLP) Accuracy: {accuracy}")
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))
```

```
Neural Network (MLP) Accuracy: 0.853245239608876
Confusion Matrix:
 [[112720     37      0]
 [ 17621     21      1]
 [  1719     13      0]]
Classification Report:
               precision    recall  f1-score   support

           0       0.85      1.00      0.92    112757
           1       0.30      0.00      0.00     17643
           2       0.00      0.00      0.00      1732

    accuracy                           0.85    132132
   macro avg       0.38      0.33      0.31    132132
weighted avg       0.77      0.85      0.79    132132
```

In [86]:
```
!pip install -U scikit-learn
!pip install -U imbalanced-learn
```

```
Requirement already satisfied: scikit-learn in d:\users\tamil\anaconda3\lib\si
te-packages (1.5.2)
Requirement already satisfied: numpy>=1.19.5 in d:\users\tamil\anaconda3\lib\s
ite-packages (from scikit-learn) (1.24.3)
Requirement already satisfied: scipy>=1.6.0 in d:\users\tamil\anaconda3\lib\si
te-packages (from scikit-learn) (1.12.0)
Requirement already satisfied: joblib>=1.2.0 in d:\users\tamil\anaconda3\lib\s
ite-packages (from scikit-learn) (1.2.0)
Requirement already satisfied: threadpoolctl>=3.1.0 in d:\users\tamil\anaconda
3\lib\site-packages (from scikit-learn) (3.5.0)
[notice] A new release of pip is available: 24.2 -> 24.3.1
[notice] To update, run: python.exe -m pip install --upgrade pip
Requirement already satisfied: imbalanced-learn in d:\users\tamil\anaconda3\li
b\site-packages (0.12.4)
Requirement already satisfied: numpy>=1.17.3 in d:\users\tamil\anaconda3\lib\s
ite-packages (from imbalanced-learn) (1.24.3)
Requirement already satisfied: scipy>=1.5.0 in d:\users\tamil\anaconda3\lib\si
te-packages (from imbalanced-learn) (1.12.0)
Requirement already satisfied: scikit-learn>=1.0.2 in d:\users\tamil\anaconda3
\lib\site-packages (from imbalanced-learn) (1.5.2)
```

```
Requirement already satisfied: joblib>=1.1.1 in d:\users\tamil\anaconda3\lib\s
ite-packages (from imbalanced-learn) (1.2.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in d:\users\tamil\anaconda
3\lib\site-packages (from imbalanced-learn) (3.5.0)
[notice] A new release of pip is available: 24.2 -> 24.3.1
[notice] To update, run: python.exe -m pip install --upgrade pip
```

In [82]:
```
import numpy as np
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
from tensorflow.keras.utils import to_categorical
from sklearn.metrics import classification_report, confusion_matrix
```

In [83]:
```
# One-hot encode labels if needed (useful for multi-class classification)
y_train_oh = to_categorical(y_train)
y_test_oh = to_categorical(y_test)
```

In [84]:
```
# Define the model structure
model = Sequential()
model.add(Dense(128, input_shape=(X_train.shape[1],), activation='relu'))
model.add(Dropout(0.3))  # Dropout layer to reduce overfitting
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.3))
model.add(Dense(32, activation='relu'))
model.add(Dense(y_train_oh.shape[1], activation='softmax'))  # Output layer wi
```

```
D:\Users\tamil\anaconda3\Lib\site-packages\keras\src\layers\core\dense.py:87:
UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. Whe
n using Sequential models, prefer using an `Input(shape)` object as the first
layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

In [85]:
```
# Compile the model
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['acc
```

In [86]:
```
# Train the model
model.fit(X_train, y_train_oh, epochs=50, batch_size=32, validation_split=0.2,
```

```
Epoch 1/50
13214/13214 ━━━━━━━━━━━━━━━━━━━━ 33s 2ms/step - accuracy: 0.8520 - loss: 0.455
7 - val_accuracy: 0.8536 - val_loss: 0.4411
Epoch 2/50
13214/13214 ━━━━━━━━━━━━━━━━━━━━ 31s 2ms/step - accuracy: 0.8524 - loss: 0.444
8 - val_accuracy: 0.8536 - val_loss: 0.4407
Epoch 3/50
13214/13214 ━━━━━━━━━━━━━━━━━━━━ 33s 2ms/step - accuracy: 0.8532 - loss: 0.441
8 - val_accuracy: 0.8536 - val_loss: 0.4420
Epoch 4/50
13214/13214 ━━━━━━━━━━━━━━━━━━━━ 33s 2ms/step - accuracy: 0.8536 - loss: 0.441
7 - val_accuracy: 0.8536 - val_loss: 0.4406
Epoch 5/50
13214/13214 ━━━━━━━━━━━━━━━━━━━━ 31s 2ms/step - accuracy: 0.8538 - loss: 0.441
1 - val_accuracy: 0.8535 - val_loss: 0.4409
Epoch 6/50
13214/13214 ━━━━━━━━━━━━━━━━━━━━ 31s 2ms/step - accuracy: 0.8524 - loss: 0.443
9 - val_accuracy: 0.8536 - val_loss: 0.4408
Epoch 7/50
13214/13214 ━━━━━━━━━━━━━━━━━━━━ 33s 2ms/step - accuracy: 0.8533 - loss: 0.443
0 - val_accuracy: 0.8536 - val_loss: 0.4412
Epoch 8/50
```

```
13214/13214 ━━━━━━━━━━━━━━━━━━━━ 32s 2ms/step - accuracy: 0.8528 - loss: 0.443
4 - val_accuracy: 0.8536 - val_loss: 0.4406
Epoch 9/50
13214/13214 ━━━━━━━━━━━━━━━━━━━━ 31s 2ms/step - accuracy: 0.8539 - loss: 0.440
6 - val_accuracy: 0.8536 - val_loss: 0.4403
Epoch 10/50
13214/13214 ━━━━━━━━━━━━━━━━━━━━ 31s 2ms/step - accuracy: 0.8530 - loss: 0.442
1 - val_accuracy: 0.8536 - val_loss: 0.4409
Epoch 11/50
13214/13214 ━━━━━━━━━━━━━━━━━━━━ 31s 2ms/step - accuracy: 0.8540 - loss: 0.440
2 - val_accuracy: 0.8536 - val_loss: 0.4407
Epoch 12/50
13214/13214 ━━━━━━━━━━━━━━━━━━━━ 32s 2ms/step - accuracy: 0.8527 - loss: 0.442
6 - val_accuracy: 0.8535 - val_loss: 0.4406
Epoch 13/50
13214/13214 ━━━━━━━━━━━━━━━━━━━━ 31s 2ms/step - accuracy: 0.8530 - loss: 0.441
3 - val_accuracy: 0.8535 - val_loss: 0.4410
Epoch 14/50
13214/13214 ━━━━━━━━━━━━━━━━━━━━ 31s 2ms/step - accuracy: 0.8528 - loss: 0.442
3 - val_accuracy: 0.8536 - val_loss: 0.4405
Epoch 15/50
13214/13214 ━━━━━━━━━━━━━━━━━━━━ 32s 2ms/step - accuracy: 0.8529 - loss: 0.441
5 - val_accuracy: 0.8536 - val_loss: 0.4412
Epoch 16/50
13214/13214 ━━━━━━━━━━━━━━━━━━━━ 31s 2ms/step - accuracy: 0.8534 - loss: 0.441
3 - val_accuracy: 0.8535 - val_loss: 0.4404
Epoch 17/50
13214/13214 ━━━━━━━━━━━━━━━━━━━━ 32s 2ms/step - accuracy: 0.8532 - loss: 0.441
9 - val_accuracy: 0.8536 - val_loss: 0.4404
Epoch 18/50
13214/13214 ━━━━━━━━━━━━━━━━━━━━ 31s 2ms/step - accuracy: 0.8519 - loss: 0.443
8 - val_accuracy: 0.8536 - val_loss: 0.4419
Epoch 19/50
13214/13214 ━━━━━━━━━━━━━━━━━━━━ 31s 2ms/step - accuracy: 0.8535 - loss: 0.441
1 - val_accuracy: 0.8536 - val_loss: 0.4408
Epoch 20/50
13214/13214 ━━━━━━━━━━━━━━━━━━━━ 31s 2ms/step - accuracy: 0.8537 - loss: 0.441
0 - val_accuracy: 0.8535 - val_loss: 0.4410
Epoch 21/50
13214/13214 ━━━━━━━━━━━━━━━━━━━━ 31s 2ms/step - accuracy: 0.8530 - loss: 0.443
2 - val_accuracy: 0.8536 - val_loss: 0.4407
Epoch 22/50
13214/13214 ━━━━━━━━━━━━━━━━━━━━ 32s 2ms/step - accuracy: 0.8545 - loss: 0.438
8 - val_accuracy: 0.8536 - val_loss: 0.4413
Epoch 23/50
13214/13214 ━━━━━━━━━━━━━━━━━━━━ 101s 8ms/step - accuracy: 0.8541 - loss: 0.43
89 - val_accuracy: 0.8535 - val_loss: 0.4404
Epoch 24/50
13214/13214 ━━━━━━━━━━━━━━━━━━━━ 31s 2ms/step - accuracy: 0.8535 - loss: 0.441
0 - val_accuracy: 0.8536 - val_loss: 0.4407
Epoch 25/50
13214/13214 ━━━━━━━━━━━━━━━━━━━━ 31s 2ms/step - accuracy: 0.8537 - loss: 0.440
6 - val_accuracy: 0.8535 - val_loss: 0.4410
Epoch 26/50
13214/13214 ━━━━━━━━━━━━━━━━━━━━ 32s 2ms/step - accuracy: 0.8537 - loss: 0.440
6 - val_accuracy: 0.8534 - val_loss: 0.4415
Epoch 27/50
13214/13214 ━━━━━━━━━━━━━━━━━━━━ 31s 2ms/step - accuracy: 0.8522 - loss: 0.443
9 - val_accuracy: 0.8536 - val_loss: 0.4414
Epoch 28/50
13214/13214 ━━━━━━━━━━━━━━━━━━━━ 32s 2ms/step - accuracy: 0.8535 - loss: 0.441
5 - val_accuracy: 0.8535 - val_loss: 0.4408
```

```
Epoch 29/50
13214/13214 ━━━━━━━━━━━━━━━━━━━━ 31s 2ms/step - accuracy: 0.8535 - loss: 0.441
4 - val_accuracy: 0.8535 - val_loss: 0.4412
Epoch 30/50
13214/13214 ━━━━━━━━━━━━━━━━━━━━ 32s 2ms/step - accuracy: 0.8532 - loss: 0.441
6 - val_accuracy: 0.8536 - val_loss: 0.4414
Epoch 31/50
13214/13214 ━━━━━━━━━━━━━━━━━━━━ 32s 2ms/step - accuracy: 0.8526 - loss: 0.443
1 - val_accuracy: 0.8536 - val_loss: 0.4414
Epoch 32/50
13214/13214 ━━━━━━━━━━━━━━━━━━━━ 33s 2ms/step - accuracy: 0.8533 - loss: 0.441
5 - val_accuracy: 0.8536 - val_loss: 0.4415
Epoch 33/50
13214/13214 ━━━━━━━━━━━━━━━━━━━━ 32s 2ms/step - accuracy: 0.8530 - loss: 0.441
5 - val_accuracy: 0.8536 - val_loss: 0.4412
Epoch 34/50
13214/13214 ━━━━━━━━━━━━━━━━━━━━ 31s 2ms/step - accuracy: 0.8536 - loss: 0.441
5 - val_accuracy: 0.8536 - val_loss: 0.4414
Epoch 35/50
13214/13214 ━━━━━━━━━━━━━━━━━━━━ 32s 2ms/step - accuracy: 0.8543 - loss: 0.440
6 - val_accuracy: 0.8536 - val_loss: 0.4406
Epoch 36/50
13214/13214 ━━━━━━━━━━━━━━━━━━━━ 33s 2ms/step - accuracy: 0.8533 - loss: 0.441
1 - val_accuracy: 0.8536 - val_loss: 0.4407
Epoch 37/50
13214/13214 ━━━━━━━━━━━━━━━━━━━━ 31s 2ms/step - accuracy: 0.8529 - loss: 0.442
8 - val_accuracy: 0.8535 - val_loss: 0.4405
Epoch 38/50
13214/13214 ━━━━━━━━━━━━━━━━━━━━ 31s 2ms/step - accuracy: 0.8537 - loss: 0.439
5 - val_accuracy: 0.8536 - val_loss: 0.4416
Epoch 39/50
13214/13214 ━━━━━━━━━━━━━━━━━━━━ 31s 2ms/step - accuracy: 0.8524 - loss: 0.443
0 - val_accuracy: 0.8536 - val_loss: 0.4411
Epoch 40/50
13214/13214 ━━━━━━━━━━━━━━━━━━━━ 31s 2ms/step - accuracy: 0.8532 - loss: 0.442
4 - val_accuracy: 0.8536 - val_loss: 0.4416
Epoch 41/50
13214/13214 ━━━━━━━━━━━━━━━━━━━━ 33s 2ms/step - accuracy: 0.8534 - loss: 0.441
5 - val_accuracy: 0.8536 - val_loss: 0.4406
Epoch 42/50
13214/13214 ━━━━━━━━━━━━━━━━━━━━ 33s 2ms/step - accuracy: 0.8532 - loss: 0.441
8 - val_accuracy: 0.8535 - val_loss: 0.4411
Epoch 43/50
13214/13214 ━━━━━━━━━━━━━━━━━━━━ 33s 2ms/step - accuracy: 0.8530 - loss: 0.442
0 - val_accuracy: 0.8536 - val_loss: 0.4410
Epoch 44/50
13214/13214 ━━━━━━━━━━━━━━━━━━━━ 34s 3ms/step - accuracy: 0.8528 - loss: 0.443
1 - val_accuracy: 0.8536 - val_loss: 0.4410
Epoch 45/50
13214/13214 ━━━━━━━━━━━━━━━━━━━━ 34s 3ms/step - accuracy: 0.8522 - loss: 0.443
1 - val_accuracy: 0.8535 - val_loss: 0.4413
Epoch 46/50
13214/13214 ━━━━━━━━━━━━━━━━━━━━ 31s 2ms/step - accuracy: 0.8529 - loss: 0.442
5 - val_accuracy: 0.8535 - val_loss: 0.4413
Epoch 47/50
13214/13214 ━━━━━━━━━━━━━━━━━━━━ 31s 2ms/step - accuracy: 0.8532 - loss: 0.442
1 - val_accuracy: 0.8536 - val_loss: 0.4413
Epoch 48/50
13214/13214 ━━━━━━━━━━━━━━━━━━━━ 31s 2ms/step - accuracy: 0.8529 - loss: 0.442
1 - val_accuracy: 0.8535 - val_loss: 0.4414
Epoch 49/50
13214/13214 ━━━━━━━━━━━━━━━━━━━━ 31s 2ms/step - accuracy: 0.8536 - loss: 0.440
```

```
1 - val_accuracy: 0.8536 - val_loss: 0.4415
Epoch 50/50
13214/13214 ━━━━━━━━━━━━━━━━━━━━ 30s 2ms/step - accuracy: 0.8530 - loss: 0.442
2 - val_accuracy: 0.8536 - val_loss: 0.4408
```

Out[86]:
```
<keras.src.callbacks.history.History at 0x188cf23ed10>
```

In [87]:
```
# Evaluate on the test data
y_pred = model.predict(X_test)
y_pred_classes = np.argmax(y_pred, axis=1)  # Convert predictions to class lab
```

```
4130/4130 ━━━━━━━━━━━━━━━━━━━━ 7s 2ms/step
```

In [88]:
```
# Display metrics
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred_classes))
print("Classification Report:\n", classification_report(y_test, y_pred_classes
```

```
Confusion Matrix:
 [[112750      6      1]
 [ 17637      6      0]
 [  1726      3      3]]
Classification Report:
              precision    recall  f1-score   support

           0       0.85      1.00      0.92    112757
           1       0.40      0.00      0.00     17643
           2       0.75      0.00      0.00      1732

    accuracy                           0.85    132132
   macro avg       0.67      0.33      0.31    132132
weighted avg       0.79      0.85      0.79    132132
```

In [ ]:

In [ ]:

In [ ]: