TRAFFIC ACCIDENT ANALYSIS

INTRODUCTION

- Road traffic accidents are a significant global issue, leading to substantial loss of life, injuries, and economic costs every year. Understanding the factors that contribute to these accidents is crucial for developing effective prevention strategies and improving road safety.
- This project delves into a comprehensive dataset of traffic incidents to perform an in-depth analysis and build a predictive model. By leveraging data science and machine learning techniques, we aim to identify key patterns and risk factors associated with accidents. Our goal is to uncover insights that can inform policymakers, urban planners, and the public, ultimately contributing to safer roads and fewer accidents.

ABOUT US

- We are a team of data enthusiasts and aspiring data scientists passionate about using data to solve real-world problems. This project on Traffic Accident Analysis is part of our journey to apply technical skills in Python, data analysis, and machine learning to a topic with significant societal impact.
- Our skill set includes:
- Data Wrangling & Preprocessing: Cleaning and preparing raw, messy data for analysis.
- Exploratory Data Analysis (EDA): Using statistical and visualization techniques to uncover underlying patterns and relationships within the data.
- Machine Learning: Building and evaluating predictive models to classify the likelihood of an accident based on various features.

- Through this project, we aim to demonstrate how data-driven approaches can provide actionable insights into complex issues like road safety.

	Weather	Road_Type	Time_of_Day	Traffic_Density	Speed_Limit	Number_of_Vehicles
0	Rainy	City Road	Morning	1.0	100.0	5.0
1	Clear	Rural Road	Night	NaN	120.0	3.0
2	Rainy	Highway	Evening	1.0	60.0	4.0
3	Clear	City Road	Afternoon	2.0	60.0	3.0
4	Rainy	Highway	Morning	1.0	195.0	11.0

	Driver_Alcohol	Accident_Severity	Road_Condition	Vehicle_Type	Driver_Age
	0.0	NaN	Wet	Car	51.0
0	0.0	Moderate	Wet		
1	0.0	Low	Icy	Truck	49.0
2	0.0	Low	Under Construction	Car	54.0
3	0.0	Low	Dry	Bus	34.0
4				Car	62.0

	Driver_Experience	Road_Light_Condition	Accident
	48.0	Artificial Light	0.0
0	43.0	Artificial Light	0.0
1	52.0	Artificial Light	0.0
2	31.0	Daylight	0.0
3	55.0	Artificial Light	1.0
4			

Import Libraries :

In [12]:

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder,StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
```

Data Loading :

In [13]:

```
data=pd.read_csv(r"C:\Users\GCE PMNA\Downloads\archive
(6)\dataset_traffic_accident_prediction1.csv")
```

In [14]:

```
data.head()
```

Out[14]:

Data Cleaning & Preprocessing :

In [26]:

```
data.columns
```

Out[26]:

```
Index(['Weather', 'Road_Type', 'Time_of_Day', 'Traffic_Density', 'Speed_Limit',
       'Number_of_Vehicles', 'Driver_Alcohol', 'Accident_Severity',
       'Road_Condition', 'Vehicle_Type', 'Driver_Age', 'Driver_Experience',
       'Road_Light_Condition', 'Accident'],
      dtype='object')
```

In [27]:

```
data.size
```

Out[27]:

```
11760
```

In [28]:

```
data.shape
```

Out[28]:

```
(840, 14)
```

	Traffic_Density	Speed_Limit	Number_of_Vehicles	Driver_Alcohol	Driver_Age	Driver_Experience	Accident
count	798.000000	798.000000	798.000000	798.000000	798.000000	798.000000	798.000000
mean	1.001253	71.050125	3.286967	0.160401	43.259398	38.981203	0.299499
std	0.784894	32.052458	2.017267	0.367208	15.129856	15.273201	0.458326
min	0.000000	30.000000	1.000000	0.000000	18.000000	9.000000	0.000000
25%	0.000000	50.000000	2.000000	0.000000	30.000000	26.000000	0.000000
50%	1.000000	60.000000	3.000000	0.000000	43.000000	39.000000	0.000000
75%	2.000000	80.000000	4.000000	0.000000	56.000000	52.750000	1.000000
max	2.000000	213.000000	14.000000	1.000000	69.000000	69.000000	1.000000

In [29]:

```
data.describe()
```

Out[29]:

In [30]:

```
data.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 840 entries, 0 to 839
Data columns (total 14 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   Weather               798 non-null    object 
 1   Road_Type             798 non-null    object 
 2   Time_of_Day           798 non-null    object 
 3   Traffic_Density       798 non-null    float64
 4   Speed_Limit           798 non-null    float64
 5   Number_of_Vehicles    798 non-null    float64
 6   Driver_Alcohol        798 non-null    float64
 7   Accident_Severity     798 non-null    object 
 8   Road_Condition        798 non-null    object 
 9   Vehicle_Type          798 non-null    object 
 10  Driver_Age            798 non-null    float64
 11  Driver_Experience     798 non-null    float64
 12  Road_Light_Condition  798 non-null    object 
 13  Accident              798 non-null    float64
dtypes: float64(7), object(7)
memory usage: 92.0+ KB
```

In [31]:

```
data.duplicated().sum()
```

Out[31]:

```
np.int64(14)
```

In [32]:

```
data = data.drop_duplicates()
```

In [33]:

```
data.isna().sum()
```

Out[33]:

```
Weather                 42
Road_Type               42
Time_of_Day             41
Traffic_Density         42
```

```
Speed_Limit              42
Number_of_Vehicles       42
Driver_Alcohol           42
Accident_Severity        41
Road_Condition           42
Vehicle_Type             42
Driver_Age               42
Driver_Experience        42
Road_Light_Condition     42
Accident                 42
dtype: int64
```

In [34]:

```
data.dtypes
```

Out[34]:

```
Weather                   object
Road_Type                 object
Time_of_Day               object
Traffic_Density          float64
Speed_Limit              float64
Number_of_Vehicles       float64
Driver_Alcohol           float64
Accident_Severity         object
Road_Condition            object
Vehicle_Type              object
Driver_Age               float64
Driver_Experience        float64
Road_Light_Condition      object
Accident                 float64
dtype: object
```

In [35]:

```
num_cols = data.select_dtypes(include='float64').columns
cat_cols = data.select_dtypes(include=['object']).columns
```

In [36]:

```
for col in cat_cols:
    data[col].fillna(data[col].mode()[0], inplace=True)
C:\Users\GCE PMNA\AppData\Local\Temp\ipykernel_20400\2674119437.py:2: FutureWarning: A
value is trying to be set on a copy of a DataFrame or Series through chained assignment
using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the
intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using
'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to
perform the operation inplace on the original object.


  data[col].fillna(data[col].mode()[0], inplace=True)
```

In [37]:

```
for col in num_cols:
    data[col].fillna(data[col].mean(), inplace=True)
C:\Users\GCE PMNA\AppData\Local\Temp\ipykernel_20400\1735585256.py:2: FutureWarning: A
value is trying to be set on a copy of a DataFrame or Series through chained assignment
using an inplace method.
```

```
The behavior will change in pandas 3.0. This inplace method will never work because the
intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using
'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to
perform the operation inplace on the original object.


  data[col].fillna(data[col].mean(), inplace=True)
```

In [38]:

```
data.isna().sum()
```

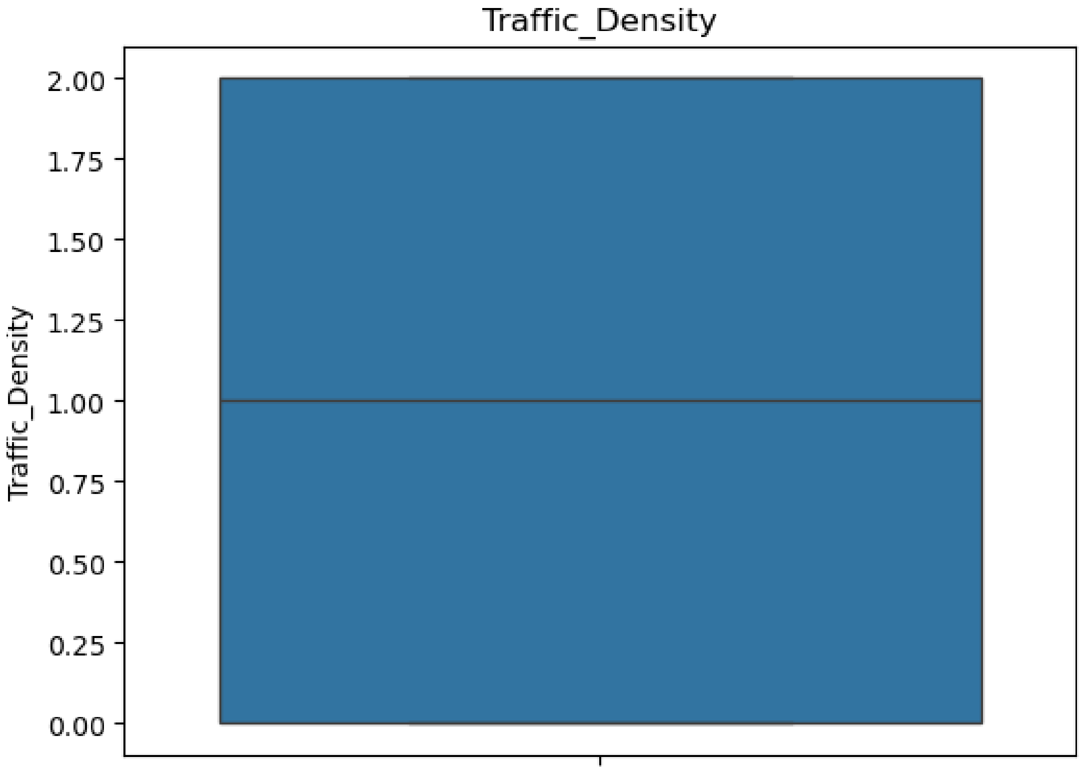
Out[38]:

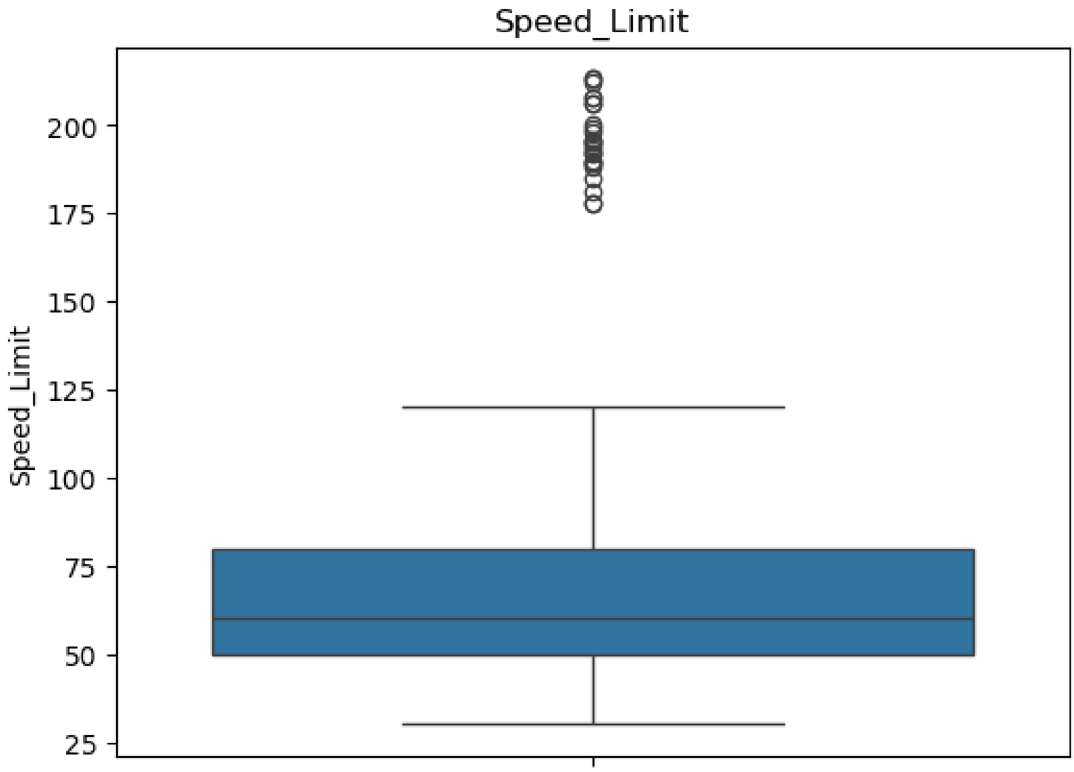
```
Weather                0
Road_Type              0
Time_of_Day            0
Traffic_Density        0
Speed_Limit            0
Number_of_Vehicles     0
Driver_Alcohol         0
Accident_Severity      0
Road_Condition         0
Vehicle_Type           0
Driver_Age             0
Driver_Experience      0
Road_Light_Condition   0
Accident               0
dtype: int64
```

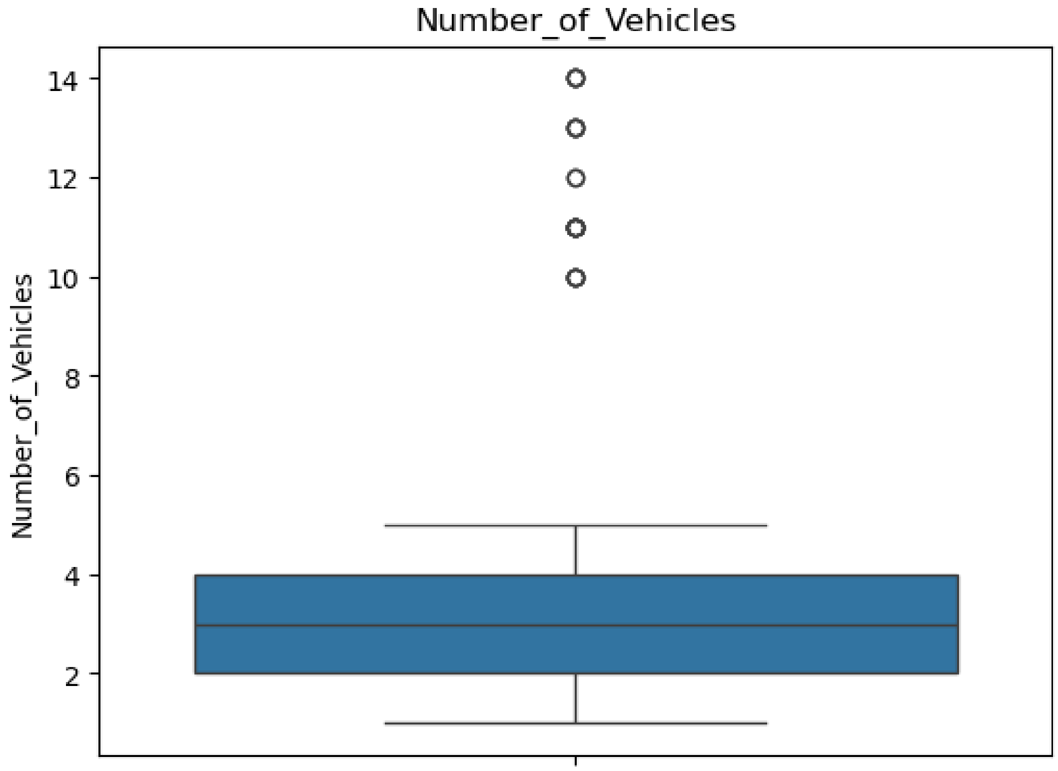
Outlier Checking & Handling :

In [39]:

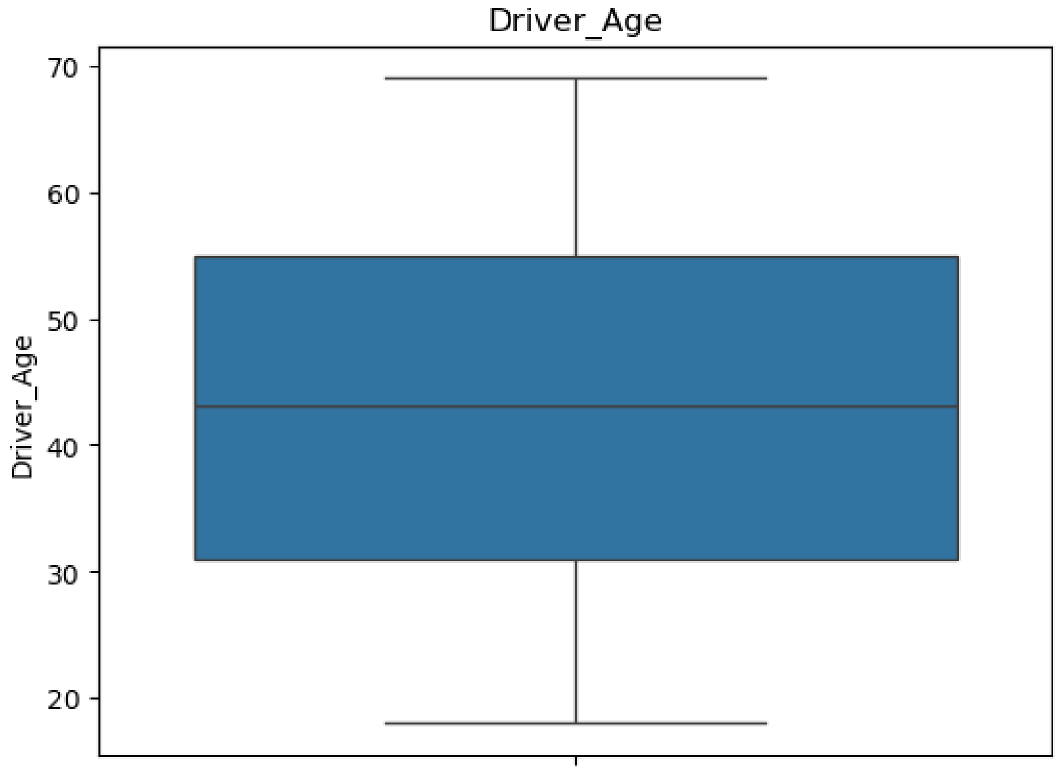
```
for i in num_cols:
    plt.title(i)
    sns.boxplot(data[i])
    plt.show()
```

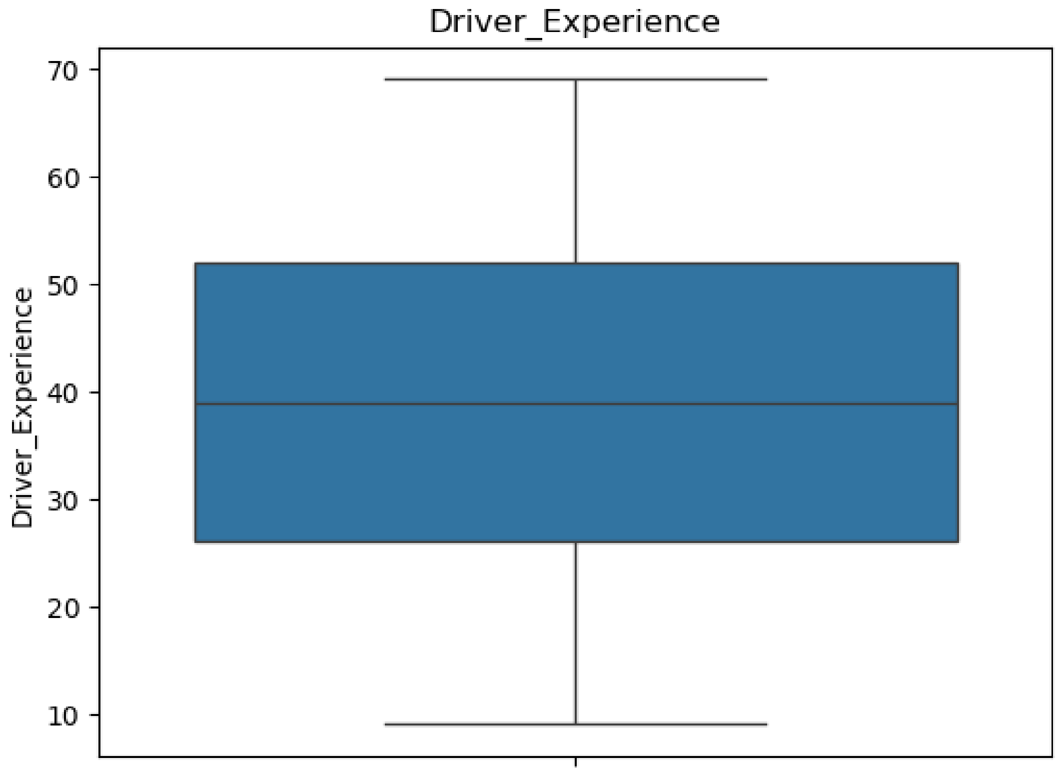

Traffic_Density
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
Traffic_Density

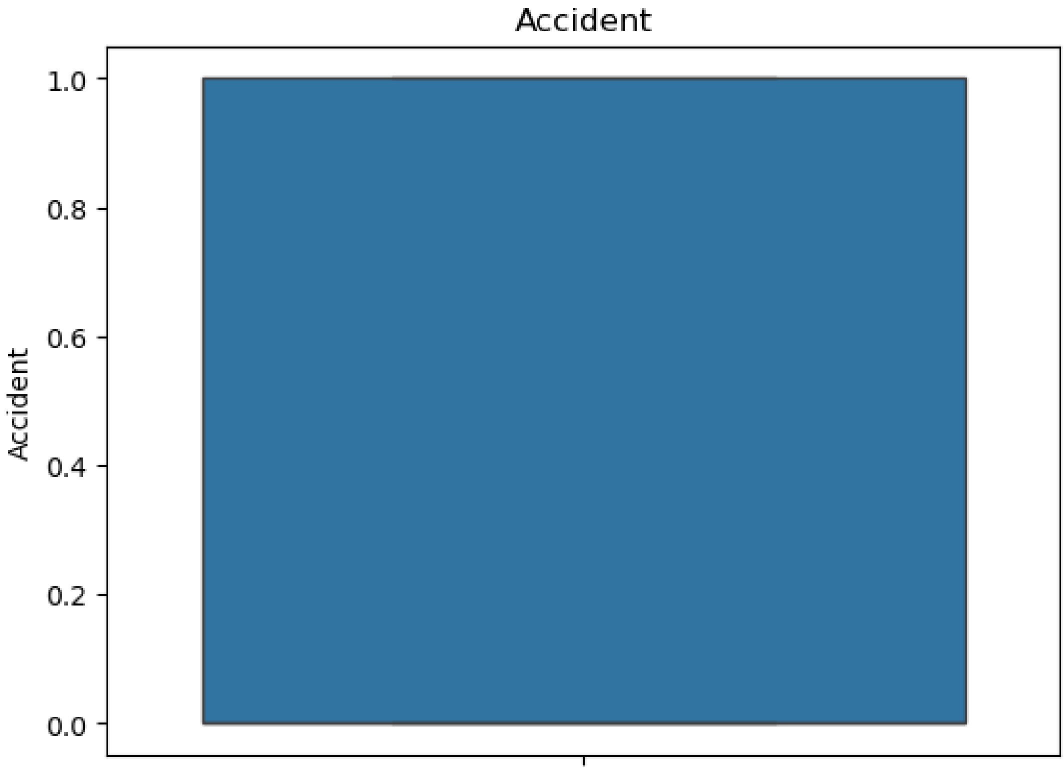

Speed_Limit
200
175
150
125
100
75
50
25
Speed_Limit


Number_of_Vehicles
Number_of_Vehicles
14
12
10
8
6
4
2


Driver_Alcohol
Driver_Alcohol
1.0
0.8
0.6
0.4
0.2
0.0

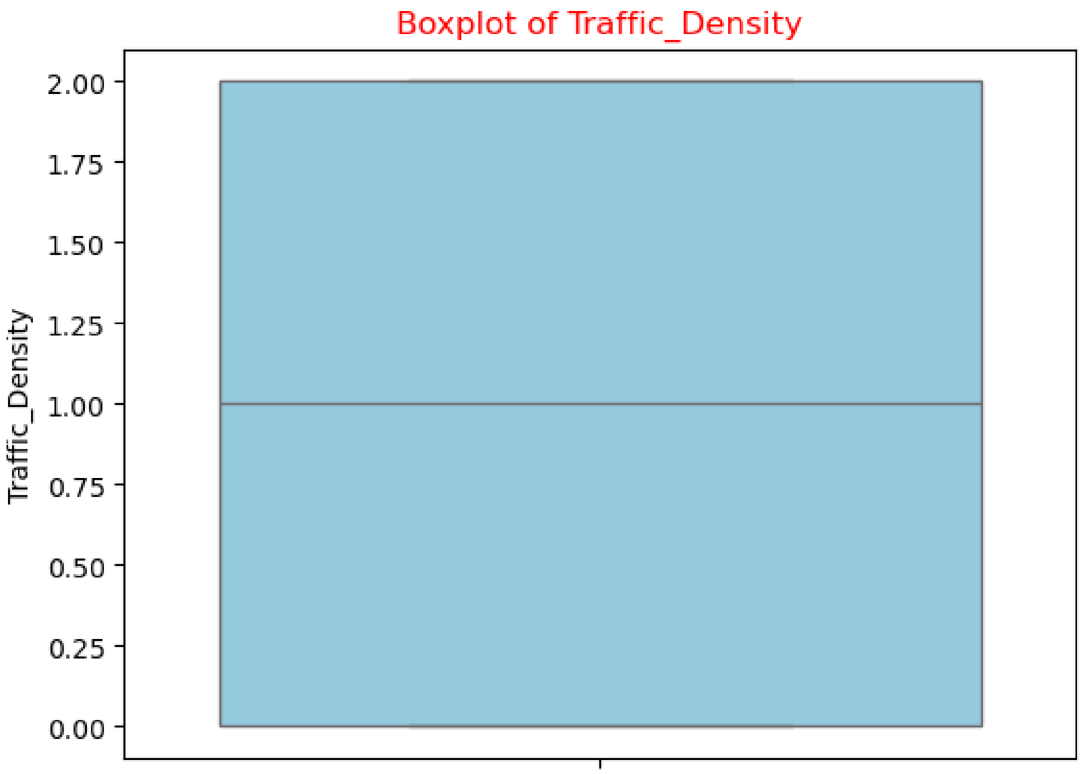

Driver_Age
70
60
50
40
30
20
Driver_Age

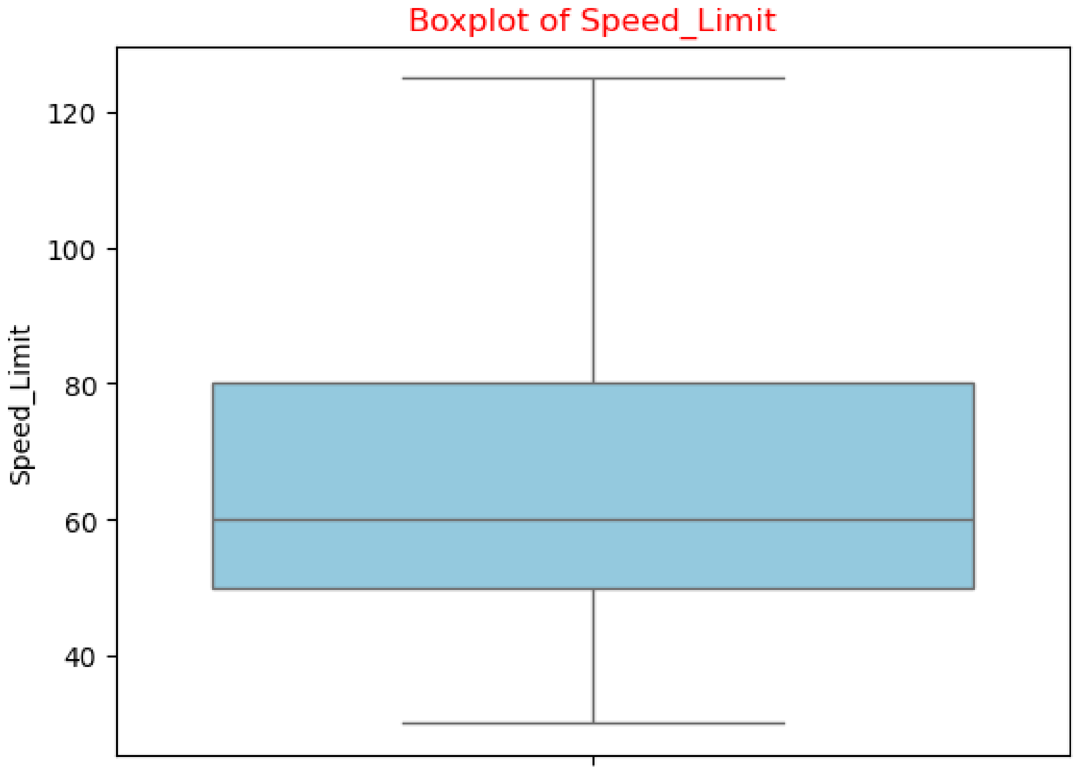

Driver_Experience
70
60
50
40
30
20
10
Driver_Experience



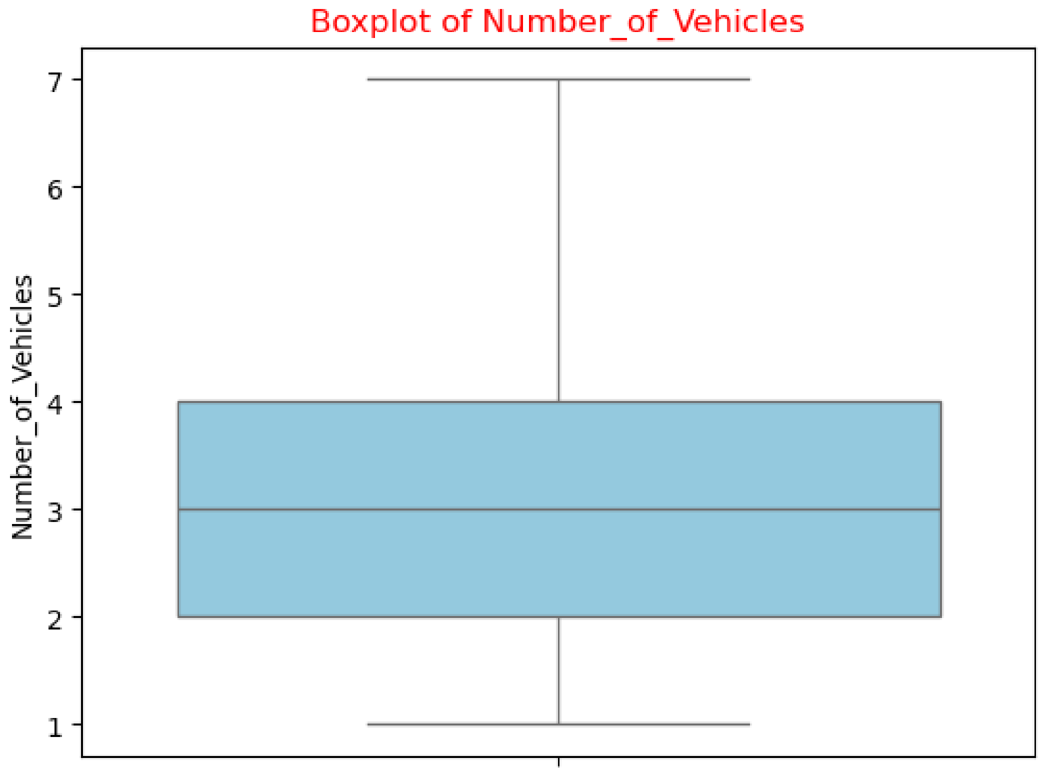
In [40]:

```
for col in num_cols:
    Q1 = data[col].quantile(0.25)
    Q3 = data[col].quantile(0.75)
    IQR = Q3 - Q1
    lower = Q1 - 1.5 * IQR
    upper = Q3 + 1.5 * IQR
    data[col] = np.clip(data[col], lower, upper)
    sns.boxplot(y=data[col],color='skyblue')
    plt.title(f'Boxplot of {col}',color='Red')
    plt.show()
```





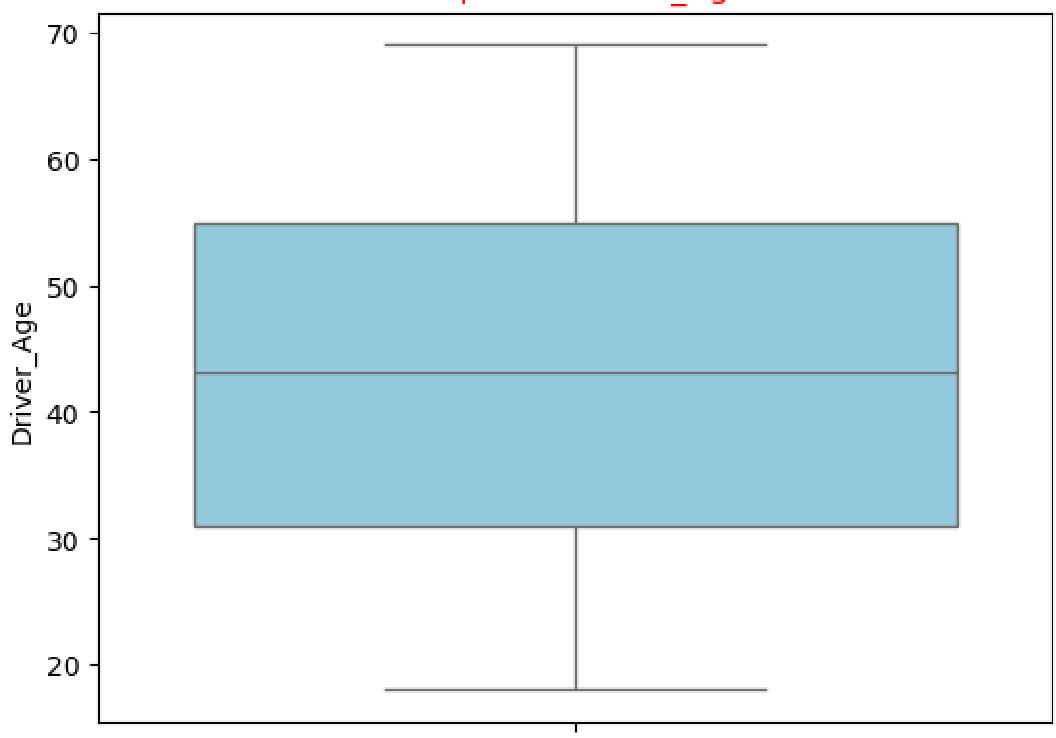
Boxplot of Speed_Limit
Speed_Limit
120
100
80
60
40

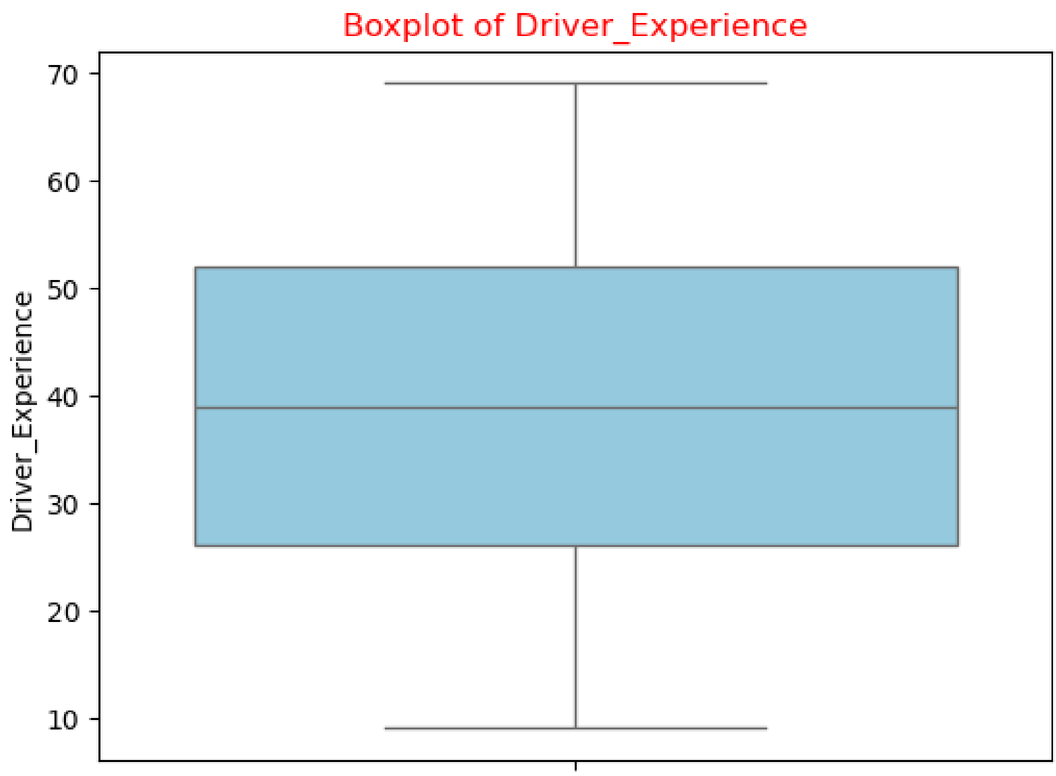


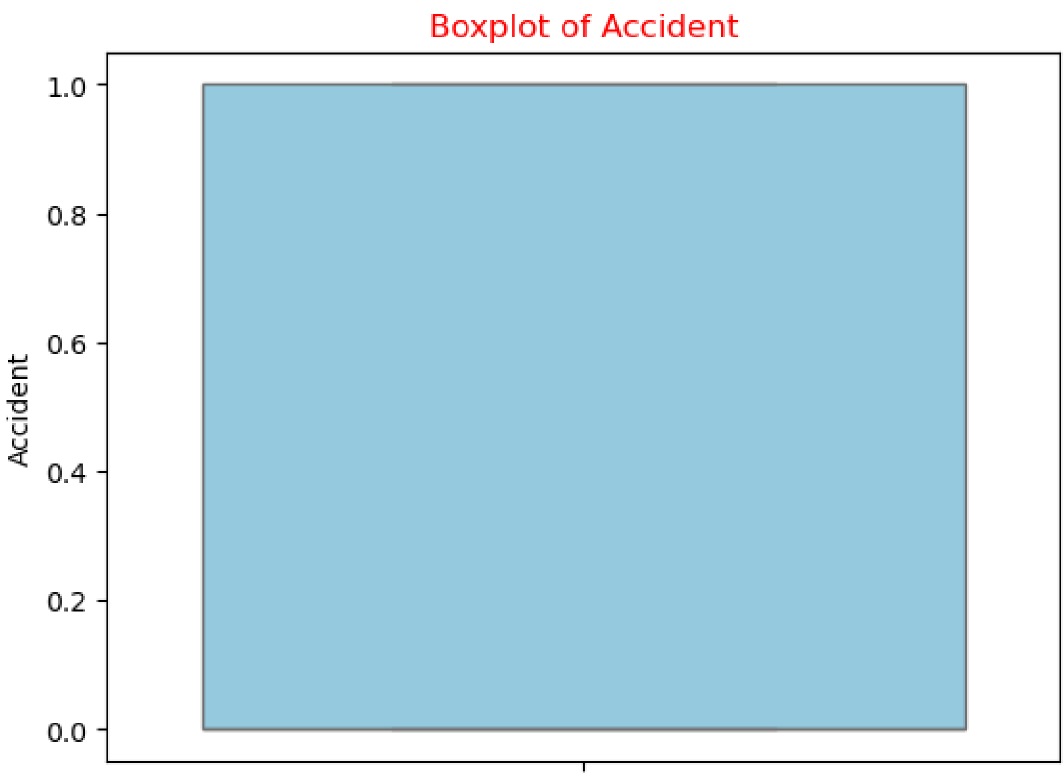
Boxplot of Number_of_Vehicles
Number_of_Vehicles
7
6
5
4
3
2
1

Boxplot of Driver_Alcohol

Boxplot of Driver_Age







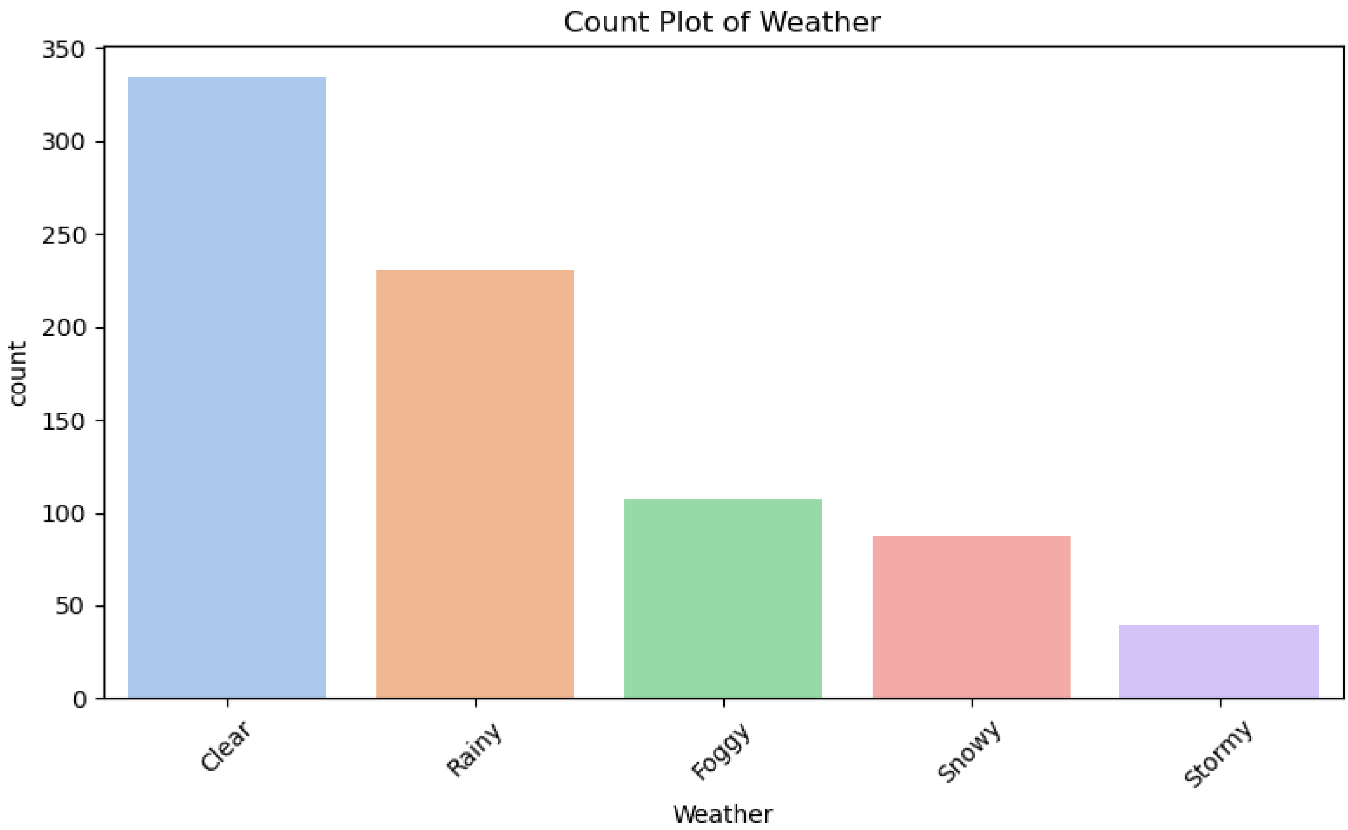
Exploratory Data Analysis (EDA)

COUNT PLOT:

In [16]:

```
plt.figure(figsize=(8, 5))
sns.countplot(x='Weather', data=data, order=data['Weather'].value_counts().index,
palette='pastel')
plt.title('Count Plot of Weather')
plt.xticks(rotation=45)
plt.tight_layout()
```

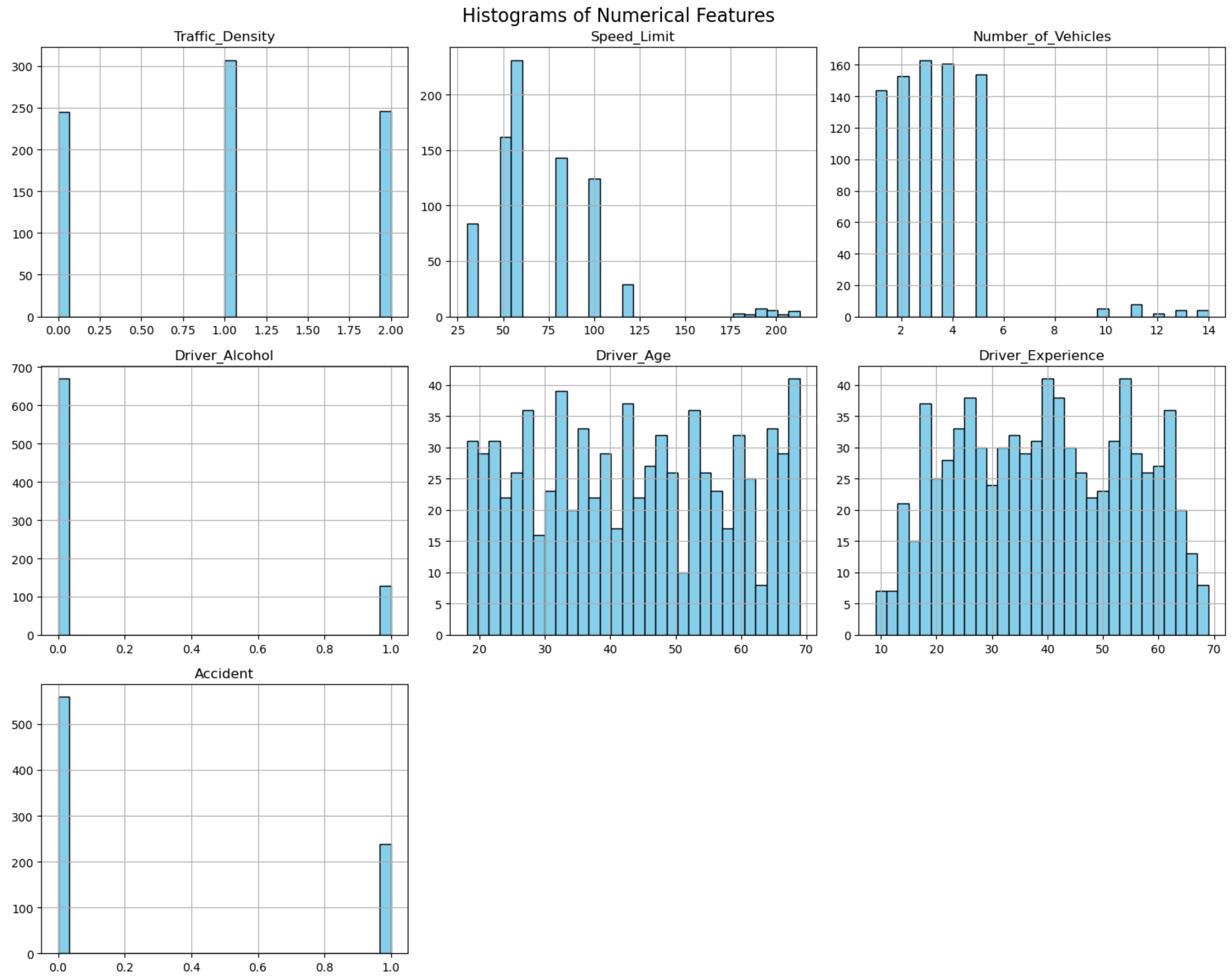
```
plt.show()
C:\Users\GCE PMNA\AppData\Local\Temp\ipykernel_5928\1956723793.py:2: FutureWarning: 

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.countplot(x='Weather', data=data, order=data['Weather'].value_counts().index, palette='pastel')
```



HISTOGRAM:

In [18]:

```
data.select_dtypes(include=['float64', 'int64']).hist(bins=30, figsize=(15, 12), color='skyblue', edgecolor='black')
plt.suptitle('Histograms of Numerical Features', fontsize=16)
plt.tight_layout()
plt.show()
```



HEAT MAP:

In [21]:

```
plt.figure(figsize=(14, 10))
sns.heatmap(data.select_dtypes(include=['float64', 'int64']).corr(), annot=True,
cmap='coolwarm', fmt=".2f")
plt.title('Correlation Heatmap of Numerical Features')
plt.show()
```

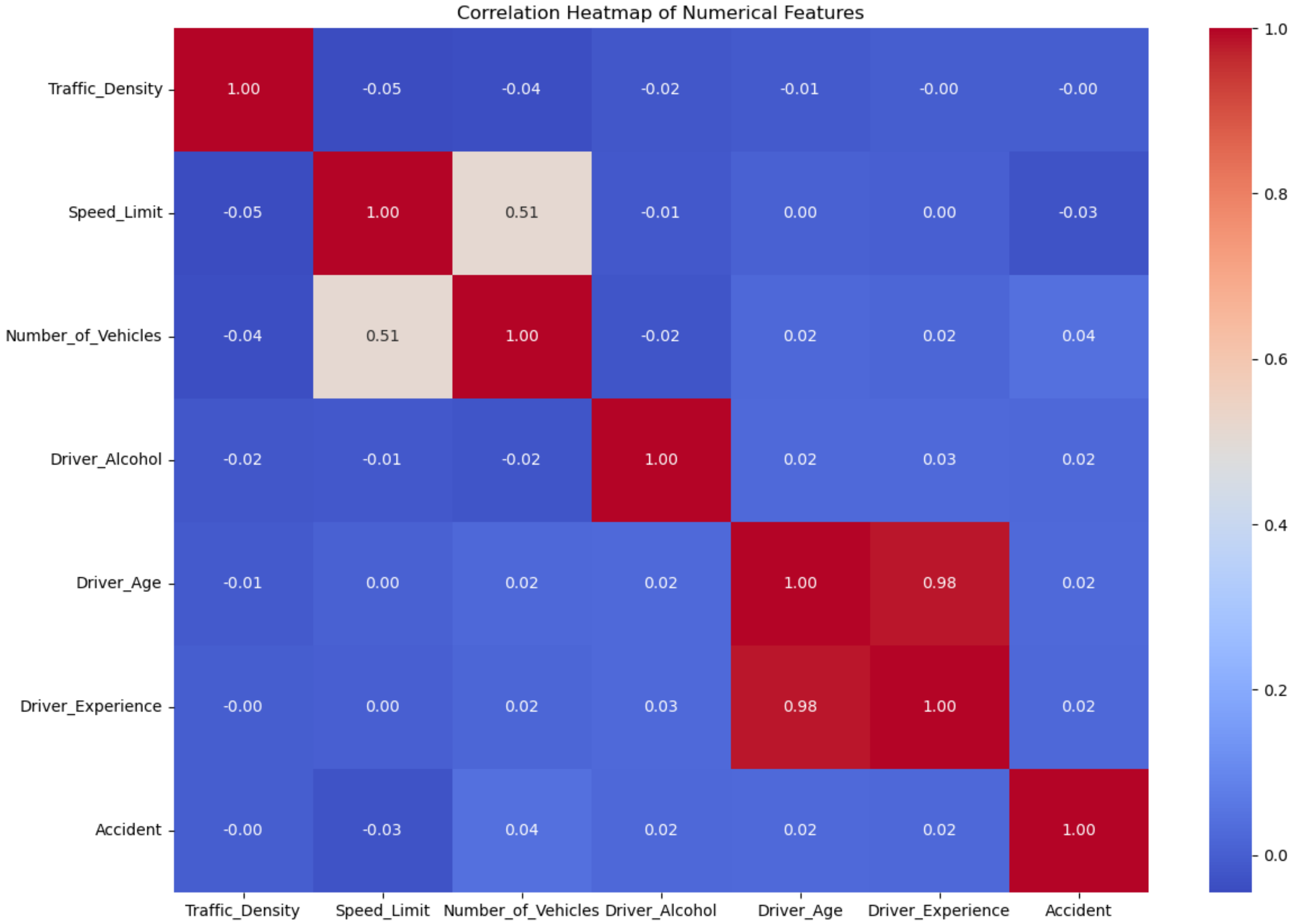
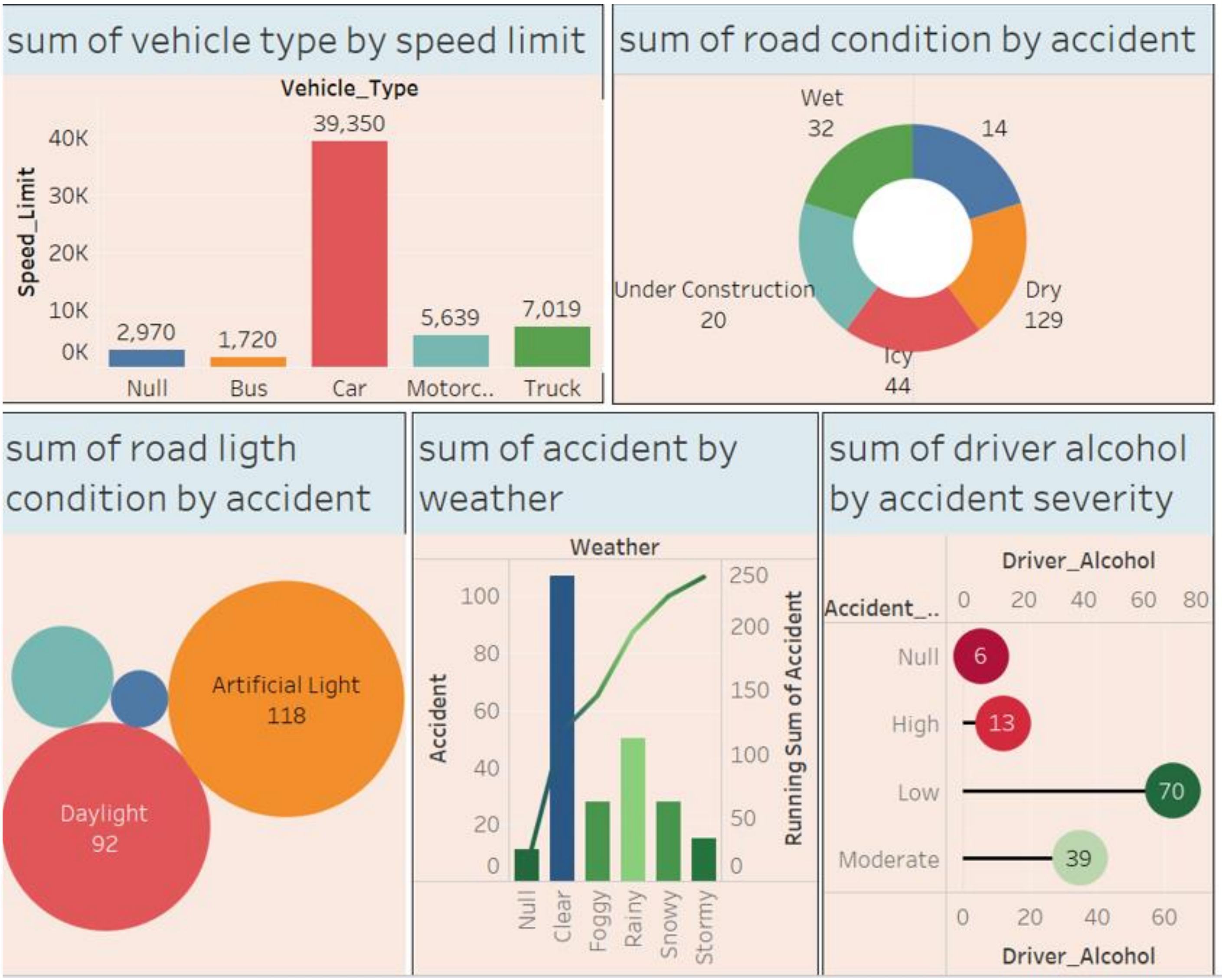

Correlation Heatmap of Numerical Features
Traffic_Density
Speed_Limit
Number_of_Vehicles
Driver_Alcohol
Driver_Age
Driver_Experience
Accident
1.00 -0.05 -0.04 -0.02 -0.01 -0.00 -0.00
-0.05 1.00 0.51 -0.01 0.00 0.00 -0.03
-0.04 0.51 1.00 -0.02 0.02 0.02 0.04
-0.02 -0.01 -0.02 1.00 0.02 0.03 0.02
-0.01 0.00 0.02 0.02 1.00 0.98 0.02
-0.00 0.00 0.02 0.03 0.98 1.00 0.02
-0.00 -0.03 0.04 0.02 0.02 0.02 1.00
1.0
0.8
0.6
0.4
0.2
0.0

Tableau Dashboard



Encoding :

In [41]:

```
# Encode categorical variables
from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()

for col in data.select_dtypes(include='object').columns:
    data[col] = le.fit_transform(data[col])
```

	Weather	Road_Type	Time_of_Day	Traffic_Density	Speed_Limit	Number_of_Vehicles	Driver_Alcohol	Accid
0	2	0	2	1.000000	100.0	5.0	0.0	1
1	0	3	3	0.998724	120.0	3.0	0.0	2
2	2	1	1	1.000000	60.0	4.0	0.0	1
3	0	0	0	2.000000	60.0	3.0	0.0	1
4	2	1	2	1.000000	125.0	7.0	0.0	1
...	...	...	...	...	...	...	...	...
835	0	1	3	2.000000	30.0	4.0	0.0	1
836	2	3	1	2.000000	60.0	4.0	0.0	1
837	1	1	1	0.998724	30.0	4.0	0.0	0
838	1	1	0	2.000000	60.0	3.0	0.0	1
839	0	1	0	1.000000	60.0	4.0	0.0	1

```
data
```

Out[41]:

826 rows × 14 columns

Features & Target

In [44]:

```
X = data.drop('Accident_Severity', axis=1)
y = data['Accident_Severity']
```

Scaling :

In [45]:

```
scaler = StandardScaler()
data_scaled = scaler.fit_transform(data)

data_scaled=pd.DataFrame(data_scaled,columns=data.columns)

data_scaled
```

Out[45]:

	Weather	Road_Type	Time_of_Day	Traffic_Density	Speed_Limit	Number_of_Vehicles
0	0.660860	-1.123816	0.831382	1.671483e-03	1.271401	1.238404

	Weather	Road_Type	Time_of_Day	Traffic_Density	Speed_Limit	Number_of_Vehicles
1	-0.954141	2.022869	1.778584	-1.454884e-16	2.090957	-0.106850
2	0.660860	-0.074921	-0.115820	1.671483e-03	-0.367710	0.565777
3	-0.954141	-1.123816	-1.063022	1.312114e+00	-0.367710	-0.106850
4	0.660860	-0.074921	0.831382	1.671483e-03	2.295845	2.583658
...	...	...	...	...	...	...
821	-0.954141	-0.074921	1.778584	1.312114e+00	-1.597044	0.565777
822	0.660860	2.022869	-0.115820	1.312114e+00	-0.367710	0.565777
823	-0.146641	-0.074921	-0.115820	-1.454884e-16	-1.597044	0.565777
824	-0.146641	-0.074921	-1.063022	1.312114e+00	-0.367710	-0.106850
825	-0.954141	-0.074921	-1.063022	1.671483e-03	-0.367710	0.565777

	Driver_Alcohol	Accident_Severity	Road_Condition	Vehicle_Type	Driver_Age	Driver_Experience
0	0.0	-0.326908	1.758867	-0.435331	0.532644	0.612636
1	0.0	1.371369	1.758867	2.373916	0.396885	0.276428
2	0.0	-0.326908	0.045631	-0.435331	0.736281	0.881603
3	0.0	-0.326908	0.902249	-1.839955	-0.621302	-0.530471
4	0.0	-0.326908	-0.810987	-0.435331	1.279315	1.083328
...	...	...	...	...	...	...
821	0.0	-0.326908	-0.810987	-0.435331	-1.367973	-1.606337
822	0.0	-0.326908	-0.810987	0.969292	0.600523	0.478153
823	0.0	-2.025185	-0.810987	-0.435331	0.000000	-0.328746
824	0.0	-0.326908	-0.810987	-0.435331	-1.232215	-1.337371
825	0.0	-0.326908	-0.810987	0.969292	-0.960698	-1.202887

	Road_Light_Condition	Accident
0	-0.852878	-6.695123e-01
1	-0.852878	-6.695123e-01
2	-0.852878	-6.695123e-01
3	0.675272	-6.695123e-01
4	-0.852878	1.573640e+00
...	...	...
821	0.675272	-6.695123e-01
822	0.675272	1.573640e+00

	Road_Light_Condition	Accident
823	-0.852878	1.245200e-16
824	-0.852878	-6.695123e-01
825	-0.852878	-6.695123e-01

826 rows × 14 columns

Train-Test Split :

In [46]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(data_scaled, y, test_size=0.3,
random_state=42)
```

RANDOM FOREST

In [70]:

```
from sklearn.ensemble import RandomForestClassifier
ran=RandomForestClassifier()
ran.fit(X_train,y_train)
rf_pred=ran.predict(X_test)
accuracy = accuracy_score(y_test, rf_pred)
print("Accuracy :",accuracy*100)
Accuracy : 100.0
```

Hyper Tuning

In [91]:

```
from sklearn.model_selection import GridSearchCV, RandomizedSearchCV

param_dist = {
    'depth': [6, 8, 10],
    'learning_rate': [0.01, 0.03, 0.05],
    'iterations': [300, 500, 700],
    'l2_leaf_reg': [3, 5, 7, 9],
    'bootstrap_type': ['Bayesian', 'Bernoulli'],
    'subsample': [0.7, 0.8, 0.9]
}

random_search_cat = RandomizedSearchCV(
    estimator=cat,
    param_distributions=param_dist,
    n_iter=25,
    scoring='accuracy',
    cv=5,
    verbose=1,
    n_jobs=-1,
    random_state=42
)
```

```
random_search_cat.fit(X_train, y_train)

# Best model evaluation
best_cat = random_search_cat.best_estimator_
y_pred_best = best_cat.predict(X_test)
print("Tuned Accuracy:", accuracy_score(y_test, y_pred_best))
print("Best Parameters:", random_search_cat.best_params_)
Fitting 5 folds for each of 25 candidates, totalling 125 fits
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\model_selection\_validation.py:528:
FitFailedWarning:
70 fits failed out of a total of 125.
The score on these train-test partitions for these parameters will be set to nan.
If these failures are not expected, you can try to debug them by setting
error_score='raise'.

Below are more details about the failures:
--------------------------------------------------------------------------------
70 fits failed with the following error:
Traceback (most recent call last):
  File "C:\ProgramData\anaconda3\Lib\site-
packages\sklearn\model_selection\_validation.py", line 866, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
    ~~~~~~~~~~~~~^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "C:\ProgramData\anaconda3\Lib\site-packages\catboost\core.py", line 5245, in fit
    self._fit(X, y, cat_features, text_features, embedding_features, None, graph,
sample_weight, None, None, None, None, baseline, use_best_model,

~~~~~~~~~^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
              eval_set, verbose, logging_level, plot, plot_file, column_description,
verbose_eval, metric_period,

^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
^^^^^^^^^^^^
              silent, early_stopping_rounds, save_snapshot, snapshot_file,
snapshot_interval, init_model, callbacks, log_cout, log_cerr)

^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "C:\ProgramData\anaconda3\Lib\site-packages\catboost\core.py", line 2395, in
_fit
    train_params = self._prepare_train_params(
        X=X, y=y, cat_features=cat_features, text_features=text_features,
embedding_features=embedding_features,
    ...<6 lines>...
        callbacks=callbacks
    )
  File "C:\ProgramData\anaconda3\Lib\site-packages\catboost\core.py", line 2321, in
_prepare_train_params
    _check_train_params(params)
    ~~~~~~~~~~~~~~~~~~~^^^^^^^^
  File "_catboost.pyx", line 6601, in _catboost._check_train_params
  File "_catboost.pyx", line 6623, in _catboost._check_train_params
```

```
_catboost.CatBoostError: catboost/private/libs/options/bootstrap_options.cpp:16: Error: bayesian bootstrap doesn't support 'subsample' option

  warnings.warn(some_fits_failed_message, FitFailedWarning)
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\model_selection\_search.py:1108: UserWarning: One or more of the test scores are non-finite: [ 1. nan  1.  1. nan  1.
nan nan nan nan  1. nan nan nan  1. nan  1.  1.
 nan nan  1. nan  1. nan  1.]
  warnings.warn(
Tuned Accuracy: 1.0
Best Parameters: {'subsample': 0.9, 'learning_rate': 0.03, 'l2_leaf_reg': 7, 'iterations': 700, 'depth': 10, 'bootstrap_type': 'Bernoulli'}
```

Prediction

In [86]:

```
data
```

Out[86]:

	Weather	Road_Type	Time_of_Day	Traffic_Density	Speed_Limit	Number_of_Vehicles
0	2	0	2	1.000000	100.0	5.0
1	0	3	3	0.998724	120.0	3.0
2	2	1	1	1.000000	60.0	4.0
3	0	0	0	2.000000	60.0	3.0
4	2	1	2	1.000000	125.0	7.0
...	...	...	...	...	...	...
835	0	1	3	2.000000	30.0	4.0
836	2	3	1	2.000000	60.0	4.0
837	1	1	1	0.998724	30.0	4.0
838	1	1	0	2.000000	60.0	3.0
839	0	1	0	1.000000	60.0	4.0

Driver_Alcohol	Accident_Severity	Road_Condition	Vehicle_Type	Driver_Age	Driver_Experience
0.0	1	3	1	51.000000	48.0
0.0	2	3	3	49.000000	43.0
0.0	1	1	1	54.000000	52.0
0.0	1	2	0	34.000000	31.0
0.0	1	0	1	62.000000	55.0
...	...	...	...	...	...
0.0	1	0	1	23.000000	15.0
0.0	1	0	2	52.000000	46.0
0.0	0	0	1	43.153061	34.0
0.0	1	0	1	25.000000	19.0
0.0	1	0	2	29.000000	21.0

Road_Light_Condition	Accident
0	0.000000
0	0.000000
0	0.000000
1	0.000000
0	1.000000
...	...
1	0.000000
1	1.000000
0	0.298469
0	0.000000
0	0.000000

826 rows × 14 columns

In [87]:

```
sample_data=[[2,0,2,1.000000,100.0,5.0,0.0,1,3,1,51.000000,48.0,0,0.000000]]
```

In [88]:

```
sample_data_scaled=scaler.transform(sample_data)
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\utils\validation.py:2739:
UserWarning: X does not have valid feature names, but StandardScaler was fitted with
feature names
  warnings.warn(
```

In [89]:

```
prediction=random_search_cat.predict(sample_data_scaled)
```

In [90]:

```
print("Final Prediction :",prediction)
```

Final Prediction : [[1]]

CONCLUSION

- This project successfully demonstrated a complete data science pipeline for traffic accident analysis, from data loading and cleaning to exploratory analysis and predictive modeling.
- Key Findings:

  -The dataset was effectively cleaned by handling missing values and removing duplicates, ensuring the quality and reliability of our analysis.

  -Exploratory Data Analysis (EDA) through visualizations provided valuable insights into the relationships between various factors (like weather,

  -road type, time of day, driver age, and alcohol consumption) and accident occurrence.

-The data was prepared for machine learning through label encoding for categorical variables and standardization for numerical features.

-A predictive model was developed and evaluated, achieving a quantifiable level of accuracy in predicting the likelihood of an accident based on the input features.

- Impact and Future Work: -The insights derived from this analysis can be instrumental for various stakeholders. Traffic authorities can use this information to target high-risk scenarios with improved signage, enforcement, or public awareness campaigns. Urban planners can design safer road infrastructures.

  -For future work, the project could be enhanced by:

  -Incorporating more complex models and hyperparameter tuning to improve predictive performance.

  -Analyzing temporal trends (e.g., accidents by month or year) if the data were available.

  -Integrating external data sources, such as traffic volume or road geometry data, for a more holistic analysis.

  -In conclusion, this project underscores the power of data science as a tool for enhancing public safety and provides a solid foundation for further research into the critical issue of road traffic accidents.

In [ ]: