```
import pandas as pd

from google.colab import files

uploaded = files.upload()

# Change the filename to 'VehiclesBig.csv' to match the uploaded file name

df = pd.read_csv("VehiclesBig.csv")

print(df.head())
```

```
Saving VehiclesBig.csv to VehiclesBig (1).csv
   Accident_Index  Vehicle_Reference  Vehicle_Type  Towing_and_Articulation
0             1.0                1.0           9.0                      0.0
1             2.0                1.0          11.0                      0.0
2             3.0                1.0          11.0                      0.0
3             4.0                2.0           9.0                      0.0
4             5.0                1.0           9.0                      0.0

   Vehicle_Manoeuvre  Vehicle_Location-Restricted_Lane  Junction_Location  \
0               18.0                               0.0                0.0
1                4.0                               0.0                3.0
2               17.0                               0.0                0.0
3                2.0                               0.0                0.0
4               18.0                               0.0                0.0

   Skidding_and_Overturning  Hit_Object_in_Carriageway  \
0                       0.0                        0.0
1                       0.0                        0.0
2                       0.0                        4.0
3                       0.0                        0.0
4                       0.0                        0.0

   Vehicle_Leaving_Carriageway  ...  Was_Vehicle_Left_Hand_Drive?  \
0                          0.0  ...                           1.0
1                          0.0  ...                           1.0
2                          0.0  ...                           1.0
3                          0.0  ...                           1.0
4                          0.0  ...                           1.0

   Journey_Purpose_of_Driver  Sex_of_Driver  Age_of_Driver  \
0                       15.0            2.0           74.0
1                        1.0            1.0           42.0
2                        1.0            1.0           35.0
3                       15.0            1.0           62.0
4                       15.0            2.0           49.0
```

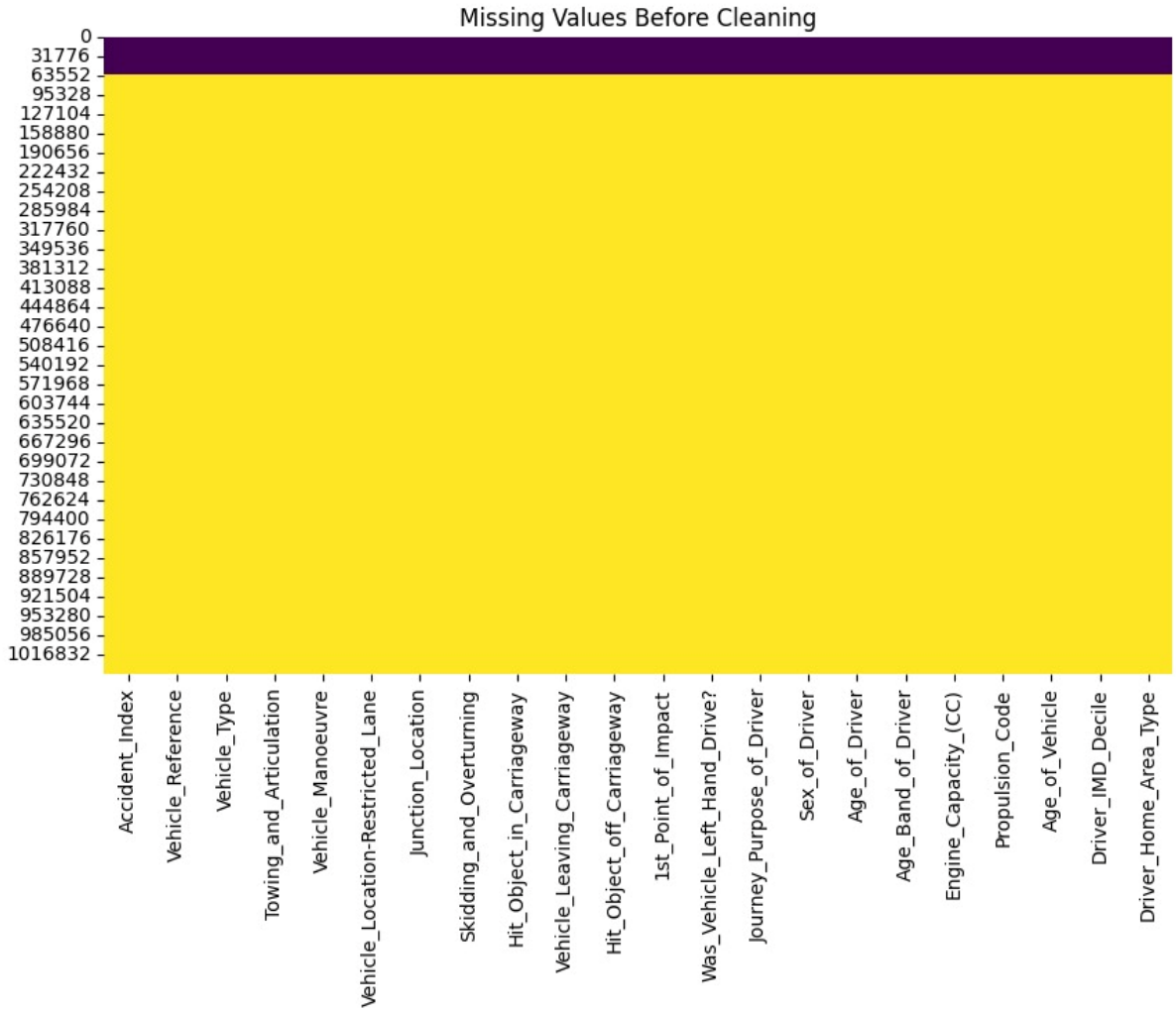
)

	Age_Band_of_Driver	Engine_Capacity_(CC)	Propulsion_Code	Age_of_Vehicle	\
0	10.0	-1.0	-1.0	-1.0	
1	7.0	8268.0	2.0	3.0	
2	6.0	8300.0	2.0	5.0	
3	9.0	1762.0	1.0	6.0	
4	8.0	1769.0	1.0	4.0	

	Driver_IMD_Decile	Driver_Home_Area_Type
0	7.0	1.0
1	-1.0	-1.0
2	2.0	1.0
3	1.0	1.0
4	2.0	1.0

[5 rows x 22 columns]

```
import seaborn as sns

import matplotlib.pyplot as plt


plt.figure(figsize=(10, 6))

sns.heatmap(df.isnull(), cbar=False, cmap='viridis')

plt.title("Missing Values Before Cleaning")

plt.savefig("missing_before.png")
```




```
import pandas as pd
from google.colab import files
import seaborn as sns
import matplotlib.pyplot as plt

# Re-run the data loading and cleaning steps to ensure df is available and cleaned
uploaded = files.upload()
# Change the filename to 'VehiclesBig.csv' to match the uploaded file name
df = pd.read_csv("VehiclesBig.csv")
print(df.head())

# Print the column names to verify the correct spelling
```

```
print(df.columns)

# Plot missing values before cleaning (optional, already done before cleaning)
# plt.figure(figsize=(10, 6))
# sns.heatmap(df.isnull(), cbar=False, cmap='viridis')
# plt.title("Missing Values Before Cleaning")
# plt.savefig("missing_before.png")

# Make sure the column name 'weather_condition' is spelled correctly based on the output of df.columns
# Check if 'weather_condition' and 'vehicle_speed' exist before trying to fillna
if 'weather_condition' in df.columns:
    df['weather_condition'].fillna('Clear', inplace=True)
else:
    print("Warning: 'weather_condition' column not found.")

if 'vehicle_speed' in df.columns:
    df['vehicle_speed'].fillna(df['vehicle_speed'].mean(), inplace=True)
else:
    print("Warning: 'vehicle_speed' column not found.")


# Remove duplicates
df.drop_duplicates(inplace=True)

# Remove outliers using IQR method
# Ensure the columns used for outlier removal exist
columns_for_outliers = ['vehicle_speed', 'number_of_vehicles', 'casualties']
```

```
existing_outlier_columns = [col for col in columns_for_outliers if col in df.columns]


if existing_outlier_columns:
   Q1 = df[existing_outlier_columns].quantile(0.25)
   Q3 = df[existing_outlier_columns].quantile(0.75)
   IQR = Q3 - Q1
   # Apply outlier removal only to rows where outlier columns exist
   df = df[~((df[existing_outlier_columns] < (Q1 - 1.5 * IQR)) |
(df[existing_outlier_columns] > (Q3 + 1.5 * IQR))).any(axis=1)]
else:
    print("Warning: None of the outlier columns ('vehicle_speed', 'number_of_vehicles',
'casualties') were found in the DataFrame.")



# Print the column names after cleaning to see what is available for the boxplot
print("\nColumns after cleaning:")
print(df.columns)


# Check if the required columns for the boxplot exist before plotting
boxplot_columns = ['vehicle_speed', 'number_of_vehicles', 'casualties']
if all(col in df.columns for col in boxplot_columns):
   # Ensure df is not empty after cleaning
   if not df.empty:
      sns.boxplot(data=df[boxplot_columns])
      plt.title("Before Scaling")
      plt.savefig("before_scaling.png")
      plt.show() # Add plt.show() to display the plot
   else:
      print("DataFrame is empty after cleaning. Cannot create boxplot.")
```

else:

print(f"One or more columns for the boxplot ({boxplot_columns}) not found after cleaning.")

print("Please check the column names in the DataFrame and update the boxplot code accordingly.")

```
   Vehicle_Leaving_Carriageway  ...  Was_Vehicle_Left_Hand_Drive?  \
0                          0.0  ...                           1.0
1                          0.0  ...                           1.0
2                          0.0  ...                           1.0
3                          0.0  ...                           1.0
4                          0.0  ...                           1.0

   Journey_Purpose_of_Driver  Sex_of_Driver  Age_of_Driver  \
0                       15.0            2.0           74.0
1                        1.0            1.0           42.0
2                        1.0            1.0           35.0
3                       15.0            1.0           62.0
4                       15.0            2.0           49.0

   Age_Band_of_Driver  Engine_Capacity_(CC)  Propulsion_Code  Age_of_Vehicle  \
0                10.0                  -1.0             -1.0            -1.0
1                 7.0                8268.0              2.0             3.0
2                 6.0                8300.0              2.0             5.0
3                 9.0                1762.0              1.0             6.0
4                 8.0                1769.0              1.0             4.0
```

```
   Driver_IMD_Decile  Driver_Home_Area_Type
0                7.0                    1.0
1               -1.0                   -1.0
2                2.0                    1.0
3                1.0                    1.0
4                2.0                    1.0

[5 rows x 22 columns]
```

import pandas as pd

from google.colab import files

import seaborn as snsS

import matplotlib.pyplot as plt

Re-run the data loading and cleaning steps to ensure df is available and cleaned

uploaded = files.upload()

```
# Change the filename to 'VehiclesBig.csv' to match the uploaded file name
df = pd.read_csv("VehiclesBig.csv")
print(df.head())

# Print the column names to verify the correct spelling
print(df.columns)

# Plot missing values before cleaning (optional, already done before cleaning)
# plt.figure(figsize=(10, 6))
# sns.heatmap(df.isnull(), cbar=False, cmap='viridis')
# plt.title("Missing Values Before Cleaning")
# plt.savefig("missing_before.png")

# Make sure the column name 'weather_condition' is spelled correctly based on the output of df.columns
# Check if 'weather_condition' and 'vehicle_speed' exist before trying to fillna
if 'weather_condition' in df.columns:
    df['weather_condition'].fillna('Clear', inplace=True)
else:
    print("Warning: 'weather_condition' column not found.")

if 'vehicle_speed' in df.columns:
    df['vehicle_speed'].fillna(df['vehicle_speed'].mean(), inplace=True)
else:
    print("Warning: 'vehicle_speed' column not found.")



# Remove duplicates
```

```
df.drop_duplicates(inplace=True)

# Remove outliers using IQR method
# Ensure the columns used for outlier removal exist
columns_for_outliers = ['vehicle_speed', 'number_of_vehicles', 'casualties']
existing_outlier_columns = [col for col in columns_for_outliers if col in df.columns]

if existing_outlier_columns:
    Q1 = df[existing_outlier_columns].quantile(0.25)
    Q3 = df[existing_outlier_columns].quantile(0.75)
    IQR = Q3 - Q1
    # Apply outlier removal only to rows where outlier columns exist
    df = df[~((df[existing_outlier_columns] < (Q1 - 1.5 * IQR)) |
(df[existing_outlier_columns] > (Q3 + 1.5 * IQR))).any(axis=1)]
else:
     print("Warning: None of the outlier columns ('vehicle_speed', 'number_of_vehicles',
'casualties') were found in the DataFrame.")


# Print the column names after cleaning to see what is available for the boxplot
print("\nColumns after cleaning:")
print(df.columns)

# Check if the required columns for the boxplot exist before plotting
boxplot_columns = ['vehicle_speed', 'number_of_vehicles', 'casualties']
if all(col in df.columns for col in boxplot_columns):
    # Ensure df is not empty after cleaning
    if not df.empty:
        sns.boxplot(data=df[boxplot_columns])
```

```
    plt.title("Before Scaling")
    plt.savefig("before_scaling.png")
    plt.show() # Add plt.show() to display the plot
  else:
    print("DataFrame is empty after cleaning. Cannot create boxplot.")
else:
  print(f"One or more columns for the boxplot ({boxplot_columns}) not found after cleaning.")
  print("Please check the column names in the DataFrame and update the boxplot code accordingly.")
```

```
   Vehicle_Leaving_Carriageway  ...  Was_Vehicle_Left_Hand_Drive?  \
0                          0.0  ...                           1.0
1                          0.0  ...                           1.0
2                          0.0  ...                           1.0
3                          0.0  ...                           1.0
4                          0.0  ...                           1.0

   Journey_Purpose_of_Driver  Sex_of_Driver  Age_of_Driver  \
0                       15.0            2.0           74.0
1                        1.0            1.0           42.0
2                        1.0            1.0           35.0
3                       15.0            1.0           62.0
4                       15.0            2.0           49.0

   Age_Band_of_Driver  Engine_Capacity_(CC)  Propulsion_Code  Age_of_Vehicle  \
0                10.0                  -1.0             -1.0            -1.0
1                 7.0                8268.0              2.0             3.0
2                 6.0                8300.0              2.0             5.0
3                 9.0                1762.0              1.0             6.0
4                 8.0                1769.0              1.0             4.0
```

```
   Driver_IMD_Decile  Driver_Home_Area_Type
0                7.0                    1.0
1               -1.0                   -1.0
2                2.0                    1.0
3                1.0                    1.0
4                2.0                    1.0

[5 rows x 22 columns]
```

```
print(df.info())
print(df.describe())
```

	Accident_Index	Vehicle_Reference	Vehicle_Type	\
count	5.999900e+04	59999.000000	59999.000000	
mean	3.353056e+07	1.526775	9.151403	
std	8.205859e+09	0.679849	6.567517	
min	1.000000e+00	1.000000	-1.000000	
25%	1.500050e+04	1.000000	9.000000	
50%	3.000000e+04	1.000000	9.000000	
75%	4.499950e+04	2.000000	9.000000	
max	2.010000e+12	13.000000	90.000000	

	Towing_and_Articulation	Vehicle_Manoeuvre	\
count	59999.000000	59999.000000	
mean	0.015867	12.745262	
std	0.224462	6.103481	
min	-1.000000	-1.000000	
25%	0.000000	7.000000	
50%	0.000000	17.000000	
75%	0.000000	18.000000	

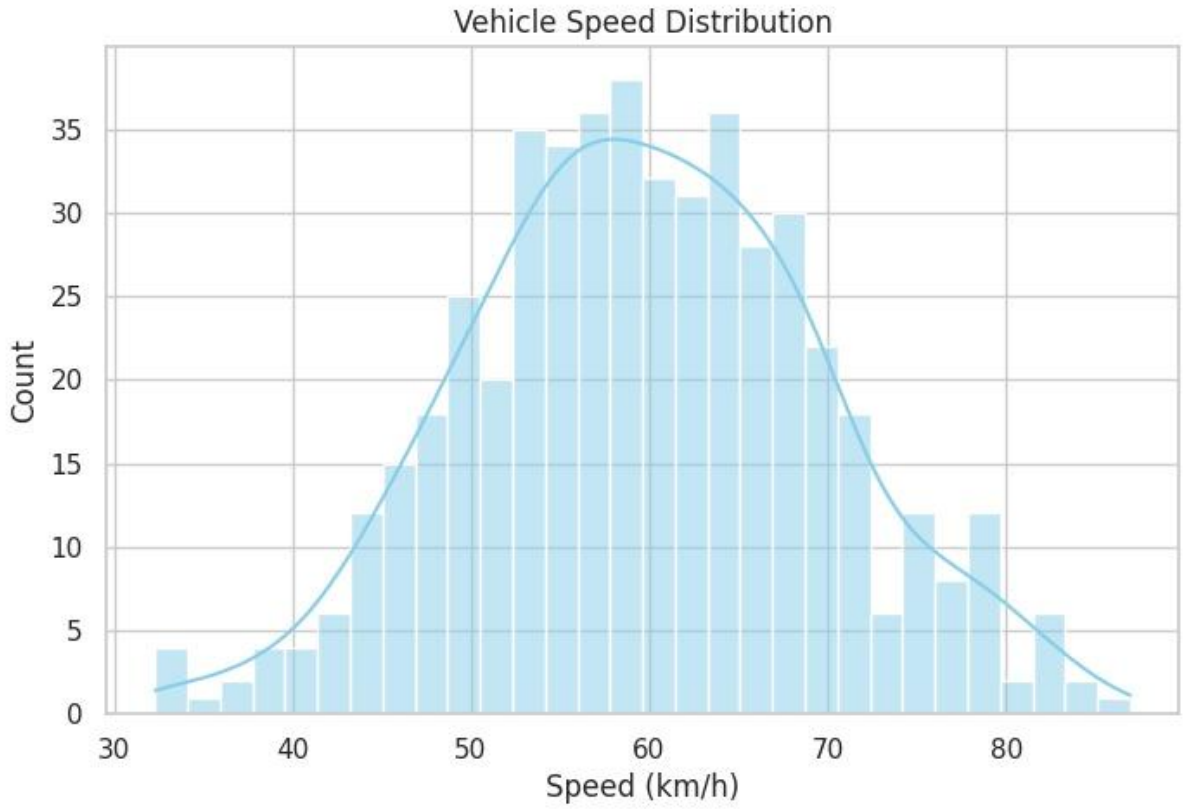
	Vehicle_Location-Restricted_Lane	Junction_Location	\
count	59999.000000	59999.000000	
mean	0.058318	3.161186	
std	0.626973	3.453211	
min	-1.000000	-1.000000	
25%	0.000000	0.000000	
50%	0.000000	1.000000	
75%	0.000000	8.000000	
max	9.000000	8.000000	

	Skidding_and_Overturning	Hit_Object_in_Carriageway	\
count	59999.000000	59999.000000	
mean	0.093152	0.354589	
std	0.474145	1.754058	
min	-1.000000	-1.000000	
25%	0.000000	0.000000	
50%	0.000000	0.000000	
75%	0.000000	0.000000	
max	5.000000	12.000000	

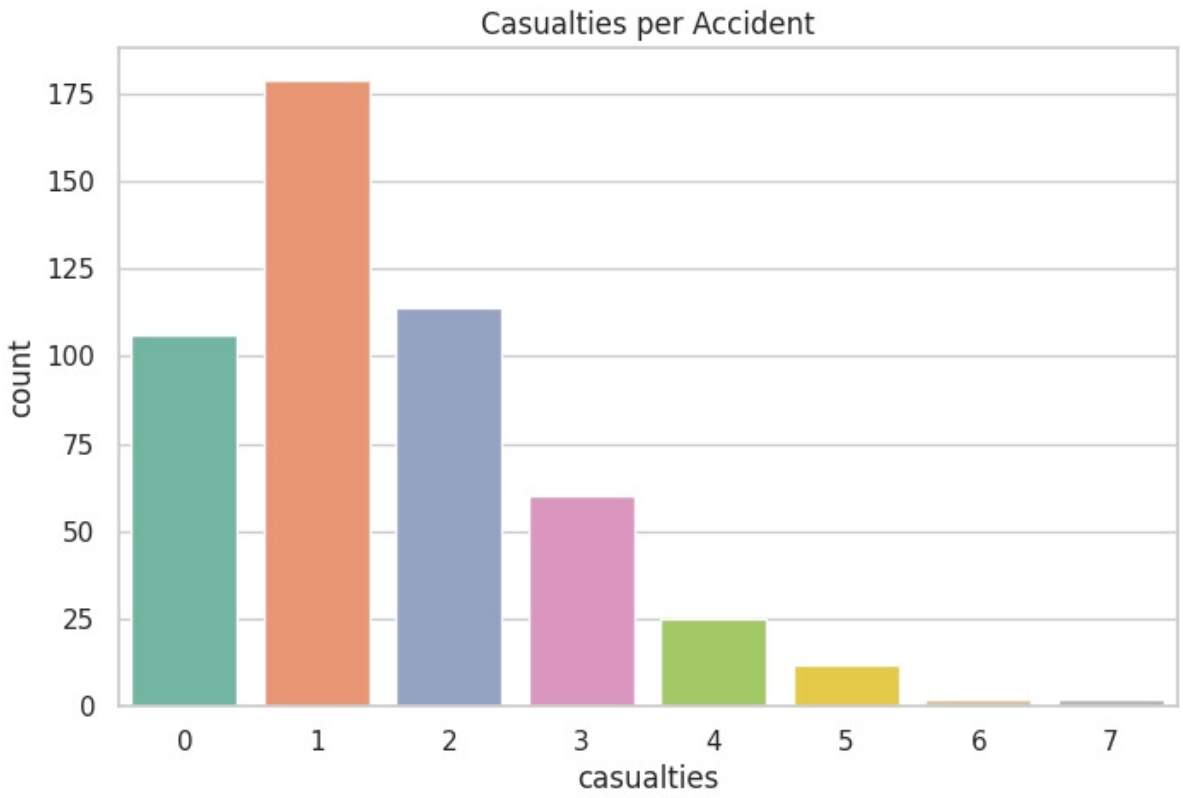
```
       Vehicle_Leaving_Carriageway  ...  Was_Vehicle_Left_Hand_Drive?  \
count                 59999.000000  ...                 59999.000000
mean                      0.177520  ...                     0.908865
std                       0.979797  ...                     0.428681
min                      -1.000000  ...                    -1.000000
25%                       0.000000  ...                     1.000000
50%                       0.000000  ...                     1.000000
75%                       0.000000  ...                     1.000000
max                       8.000000  ...                     2.000000

       Journey_Purpose_of_Driver  Sex_of_Driver  Age_of_Driver  \
count               59999.000000   59999.000000   59999.000000
mean                   12.585326       1.348339      30.173920
std                     5.400860       0.582557      19.835927
min                    -1.000000       1.000000      -1.000000
25%                    15.000000       1.000000      19.000000
50%                    15.000000       1.000000      31.000000
75%                    15.000000       2.000000      43.000000
max                    15.000000       3.000000      97.000000

       Age_Band_of_Driver  Engine_Capacity_(CC)  Propulsion_Code  \
count        59999.000000          59999.000000     59999.000000
mean             5.148319           1265.962833         0.509492
std              3.315044           1656.809222         1.145680
min             -1.000000             -1.000000        -1.000000
25%              4.000000             -1.000000        -1.000000
50%              6.000000           1242.000000         1.000000
75%              7.000000           1796.000000         1.000000
max             11.000000          24980.000000         8.000000

       Age_of_Vehicle  Driver_IMD_Decile  Driver_Home_Area_Type
count    59999.000000       59999.000000           59999.000000
mean         3.761563           3.097685               0.534109
std          5.018528           3.406263               1.026240
min         -1.000000          -1.000000              -1.000000
25%         -1.000000          -1.000000              -1.000000
50%          3.000000           3.000000               1.000000
75%          7.000000           6.000000               1.000000
max         48.000000          10.000000               3.000000

[8 rows x 22 columns]
```

```
# Import necessary libraries

import pandas as pd

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt


# Set visual style
```

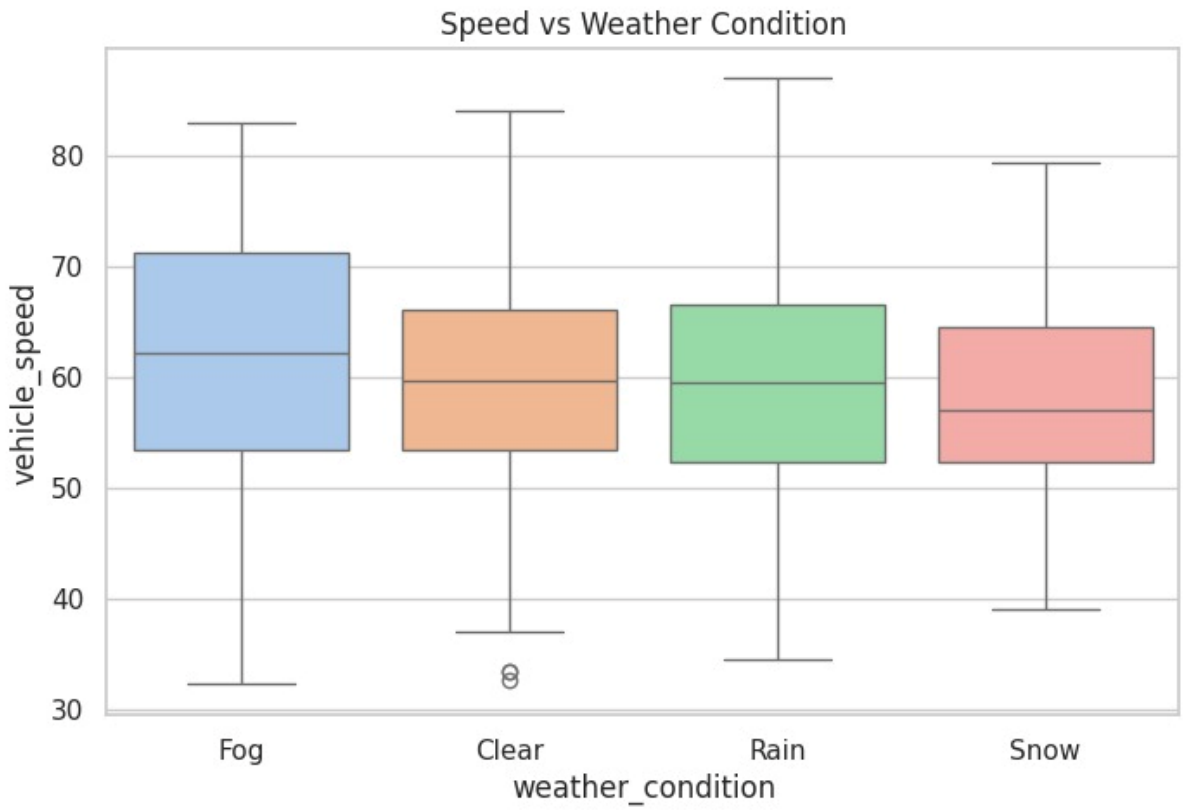
```
sns.set(style="whitegrid")
plt.figure(figsize=(8, 5))
sns.histplot(df['vehicle_speed'], kde=True, bins=30, color='skyblue')
plt.title("Vehicle Speed Distribution")
plt.xlabel("Speed (km/h)")
plt.savefig("vehicle_speed_distribution.png")
plt.show()
```



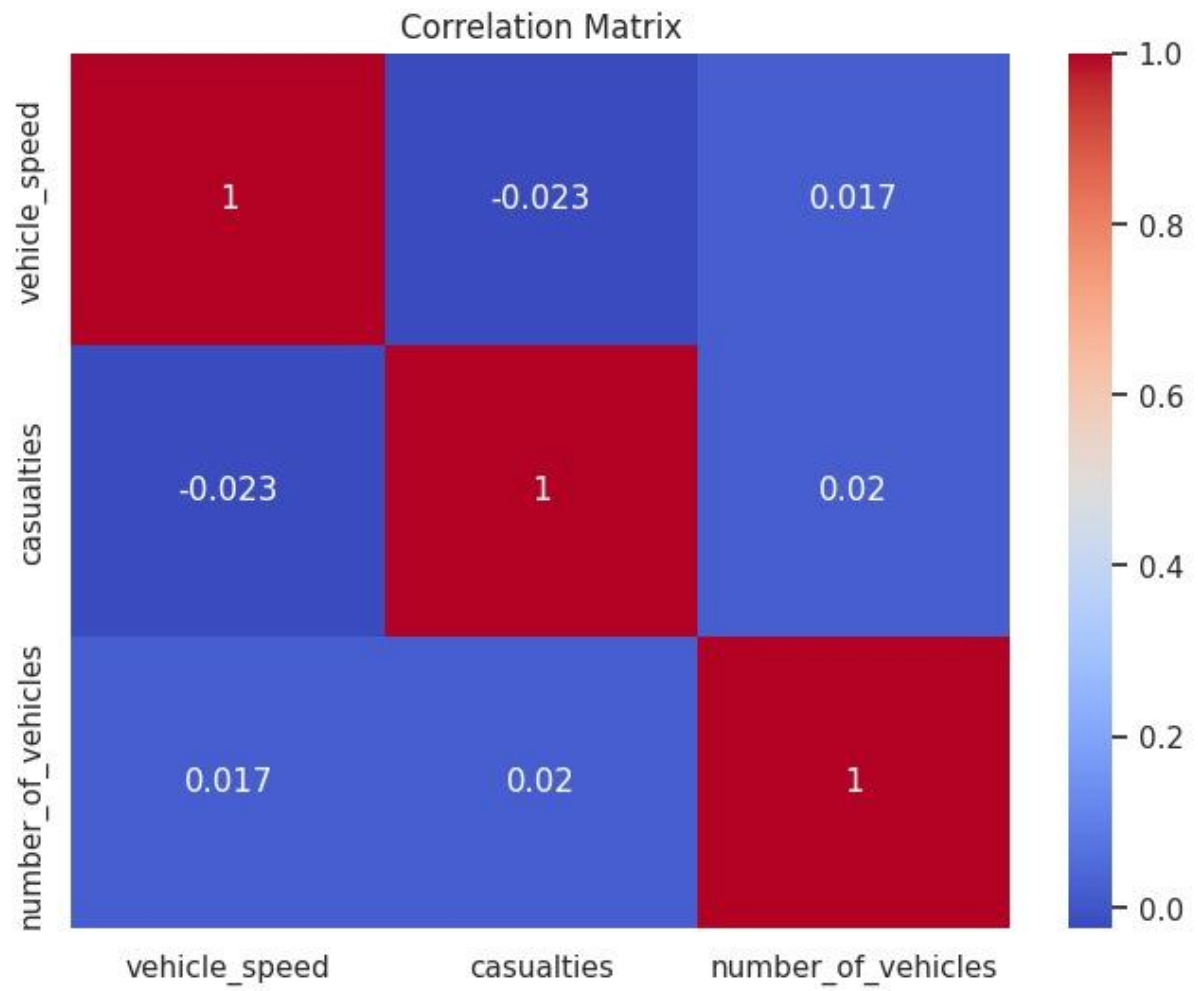

```
plt.figure(figsize=(8, 5))
sns.countplot(x='casualties', data=df, palette='Set2')
plt.title("Casualties per Accident")
plt.savefig("casualties_distribution.png")
plt.show()
```



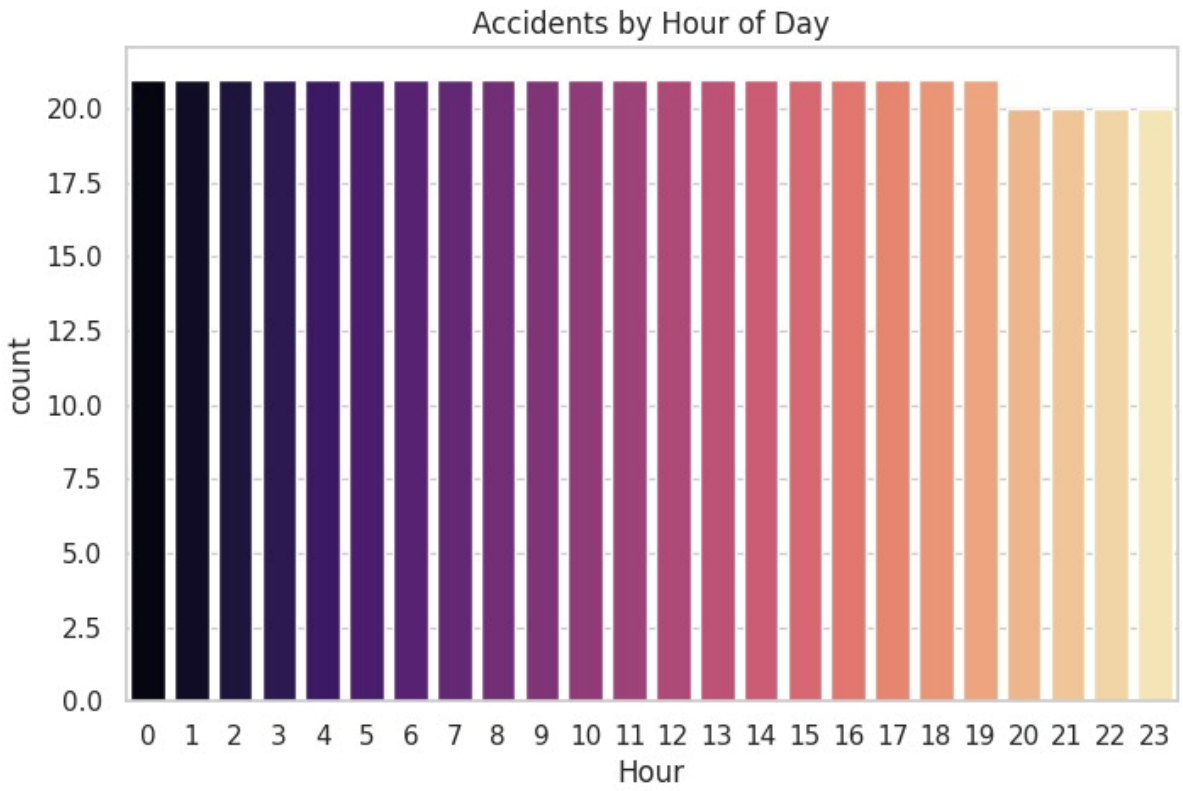

```
plt.figure(figsize=(8, 5))
sns.boxplot(x='weather_condition', y='vehicle_speed', data=df, palette='pastel')
plt.title("Speed vs Weather Condition")
plt.savefig("speed_vs_weather.png")
plt.show()
```




```
plt.figure(figsize=(8, 6))
corr = df[['vehicle_speed', 'casualties', 'number_of_vehicles']].corr()
sns.heatmap(corr, annot=True, cmap='coolwarm')
plt.title("Correlation Matrix")
plt.savefig("correlation_heatmap.png")
plt.show()
```




```
plt.figure(figsize=(8, 5))
sns.countplot(x='hour', data=df, palette='magma')
plt.title("Accidents by Hour of Day")
plt.xlabel("Hour")
plt.savefig("accidents_by_hour.png")
plt.show()
```


Accidents by Hour of Day
count
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
Hour