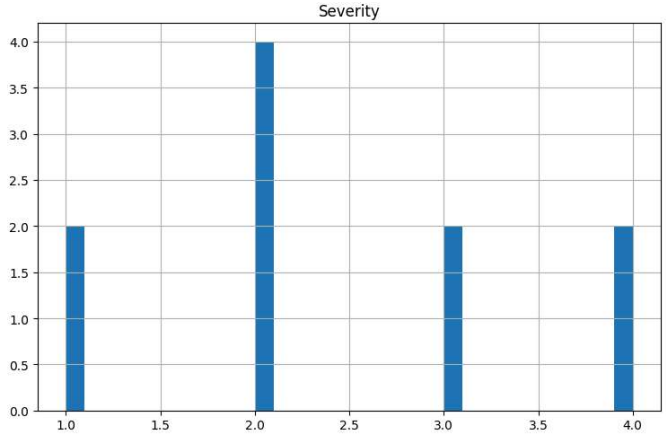
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
from sklearn.preprocessing import StandardScaler, MinMaxScaler
from sklearn.model_selection import train_test_split
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import OneHotEncoder, OrdinalEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.decomposition import PCA


import pandas as pd
df = pd.read_csv('/content/us_accidents_sample.csv')


df
```

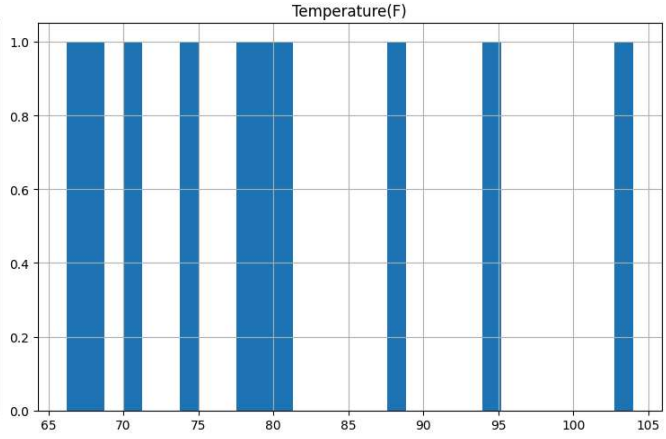
	ID	Severity	Start_Time	End_Time	City	State	Temperature(F)	Weather_Condition	Visibility(mi)	Description
0	A-001	2	2023-07-01 08:30:00	2023-07-01 09:15:00	Los Angeles	CA	78.0	Clear	10.0	Accident on I-405 N
1	A-002	3	2023-07-01 17:45:00	2023-07-01 18:30:00	Miami	FL	88.5	Rain	4.0	2-vehicle crash on US-1
2	A-003	1	2023-07-02 13:10:00	2023-07-02 13:40:00	Austin	TX	95.0	Clear	10.0	Minor accident at 5th St
3	A-004	2	2023-07-03 21:00:00	2023-07-03 21:45:00	Seattle	WA	66.2	Fog	2.0	Rear-end collision on I-5
4	A-005	4	2023-07-04 14:00:00	2023-07-04 16:00:00	Chicago	IL	79.0	Rain	5.0	Multi-car crash on I-90 E
5	A-006	2	2023-07-05 09:15:00	2023-07-05 10:00:00	New York	NY	74.3	Cloudy	9.0	3-car pileup on Brooklyn Bridge
	A-		2023-07-06	2023-07-06						Fender bender

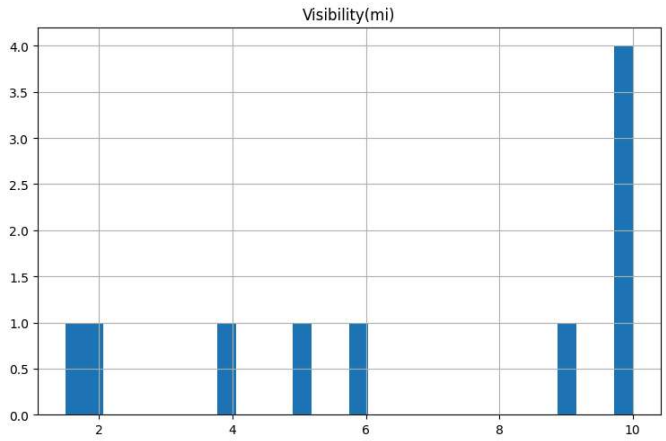
Next steps: Generate code with df | View recommended plots | New interactive sheet

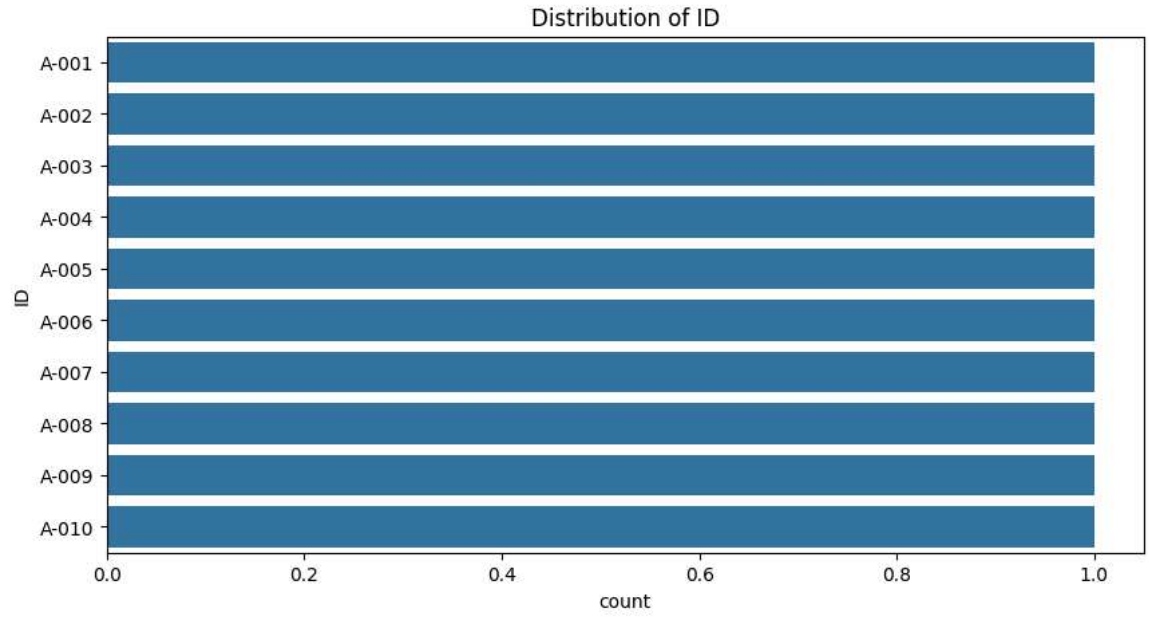
```
import matplotlib.pyplot as plt
import numpy as np

print(df.info())

print(df.describe(include='all'))

print(df.head())

print(df.isnull().sum())

print(df.duplicated().sum())

categorical_features = df.select_dtypes(include=['object', 'category']).columns
numerical_features = df.select_dtypes(include=np.number).columns

print(f"Categorical features: {list(categorical_features)}")
print(f"Numerical features: {list(numerical_features)}")

df[numerical_features[:5]].hist(bins=30, figsize=(15, 10))
plt.tight_layout()
plt.show()

for col in categorical_features[:3]:
  plt.figure(figsize=(10, 5))
  sns.countplot(data=df, y=col, order=df[col].value_counts().index[:10])
  plt.title(f'Distribution of {col}')
  plt.show()

plt.figure(figsize=(12, 10))
sns.heatmap(df[numerical_features].corr(), annot=False, cmap='coolwarm')
plt.title('Correlation Matrix of Numerical Features')
plt.show()
```

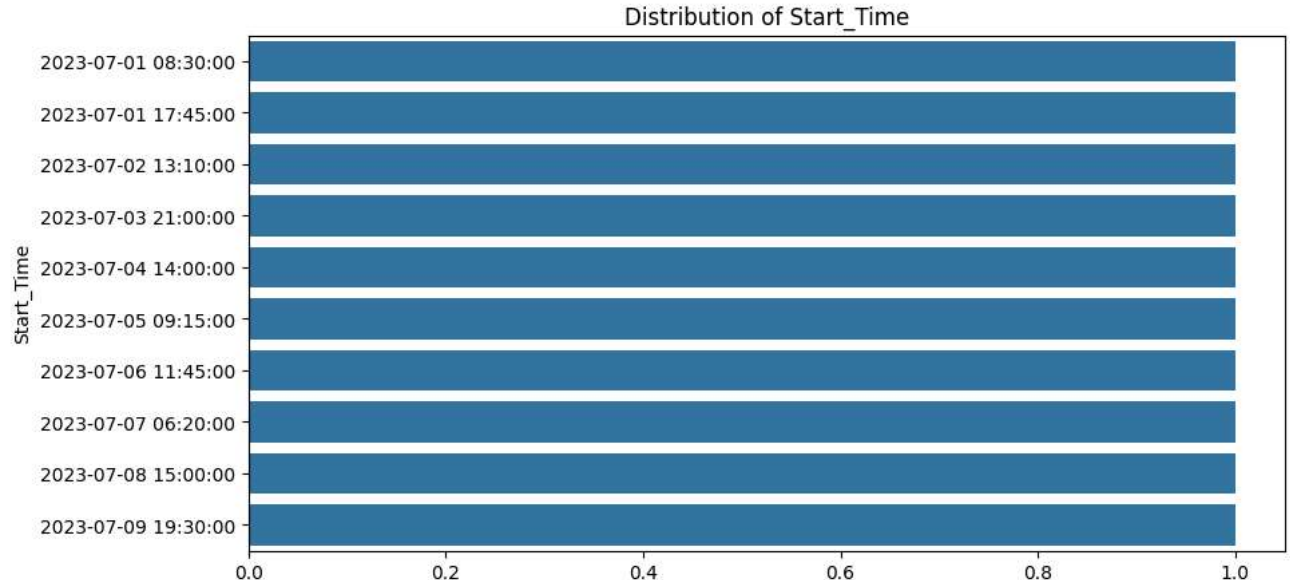
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 10 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   ID                 10 non-null     object 
 1   Severity           10 non-null     int64  
 2   Start_Time         10 non-null     object 
 3   End_Time           10 non-null     object 
 4   City               10 non-null     object 
 5   State              10 non-null     object 
 6   Temperature(F)     10 non-null     float64
 7   Weather_Condition  10 non-null     object 
 8   Visibility(mi)     10 non-null     float64
 9   Description        10 non-null     object 
dtypes: float64(2), int64(1), object(7)
memory usage: 932.0+ bytes
None
           ID   Severity           Start_Time             End_Time  \
count      10  10.000000                   10                   10   
unique     10        NaN                   10                   10   
top     A-001        NaN  2023-07-01 08:30:00  2023-07-01 09:15:00   
freq        1        NaN                    1                    1   
mean      NaN   2.400000                  NaN                  NaN   
std       NaN   1.074968                  NaN                  NaN   
min       NaN   1.000000                  NaN                  NaN   
25%       NaN   2.000000                  NaN                  NaN   
50%       NaN   2.000000                  NaN                  NaN   
75%       NaN   3.000000                  NaN                  NaN   
max       NaN   4.000000                  NaN                  NaN   

                City State  Temperature(F) Weather_Condition  Visibility(mi)  \
count             10    10       10.000000                10       10.000000   
unique            10    10             NaN                 5             NaN   
top     Los Angeles    CA             NaN             Clear             NaN   
freq               1     1             NaN                 4             NaN   
mean             NaN   NaN       80.460000               NaN        6.750000   
std              NaN   NaN       12.157869               NaN        3.474111   
min              NaN   NaN       66.200000               NaN        1.500000   
25%              NaN   NaN       71.150000               NaN        4.250000   
50%              NaN   NaN       78.500000               NaN        7.500000   
75%              NaN   NaN       86.625000               NaN       10.000000   
max              NaN   NaN      104.000000               NaN       10.000000   

                Description  
count                    10  
unique                   10  
top     Accident on I-405 N  
freq                      1  
mean                    NaN  
std                     NaN  
min                     NaN  
25%                     NaN  
50%                     NaN  
75%                     NaN  
max                     NaN  
      ID  Severity           Start_Time             End_Time         City  \
0  A-001         2  2023-07-01 08:30:00  2023-07-01 09:15:00  Los Angeles   
1  A-002         3  2023-07-01 17:45:00  2023-07-01 18:30:00        Miami   
2  A-003         1  2023-07-02 13:10:00  2023-07-02 13:40:00       Austin   
3  A-004         2  2023-07-03 21:00:00  2023-07-03 21:45:00      Seattle   
4  A-005         4  2023-07-04 14:00:00  2023-07-04 16:00:00      Chicago   

  State  Temperature(F) Weather_Condition  Visibility(mi)  \
0    CA           78.0             Clear            10.0   
1    FL           88.5              Rain             4.0   
2    TX           95.0             Clear            10.0   
3    WA           66.2               Fog             2.0   
4    IL           79.0              Rain             5.0   

                Description  
0       Accident on I-405 N  
1    2-vehicle crash on US-1  
2   Minor accident at 5th St  
3  Rear-end collision on I-5  
4  Multi-car crash on I-90 E  
ID                  0
Severity            0
Start_Time          0
End_Time            0
City                0
State               0
Temperature(F)      0
Weather_Condition   0
Visibility(mi)      0
Description         0
dtype: int64
0
Categorical features: ['ID', 'Start_Time', 'End_Time', 'City', 'State', 'Weather_Condition', 'Description']
Numerical features: ['Severity', 'Temperature(F)', 'Visibility(mi)']
```

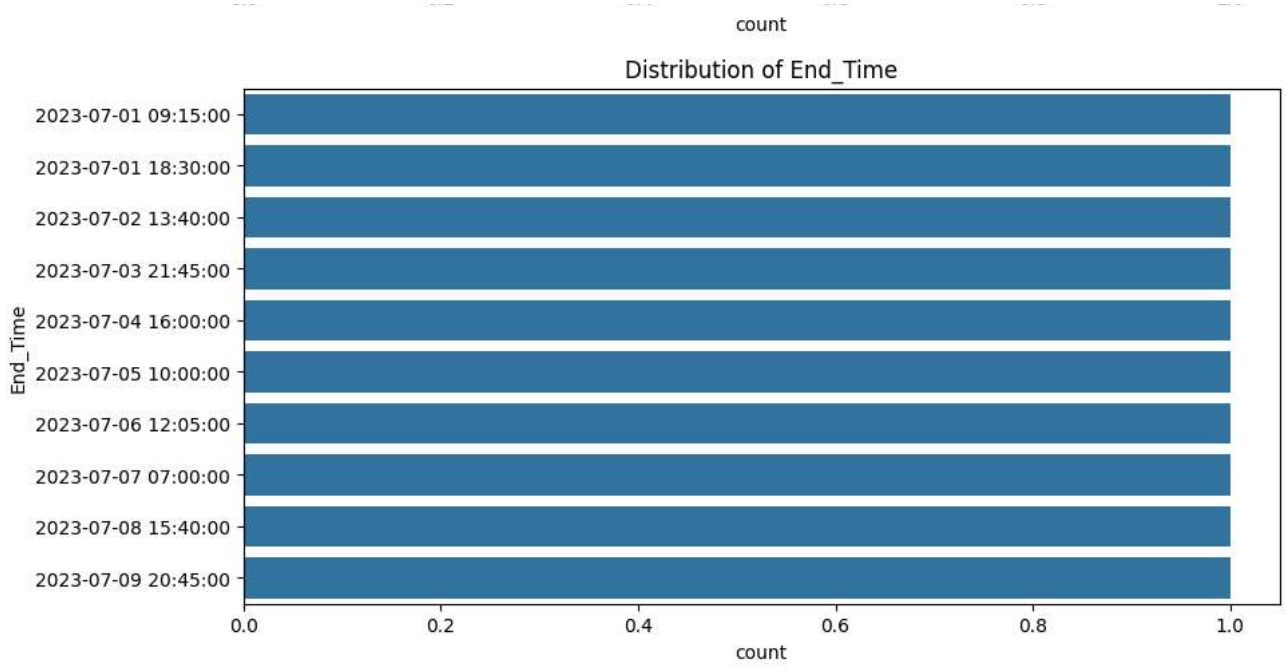

Severity
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
1.0
1.5
2.0
2.5
3.0
3.5
4.0

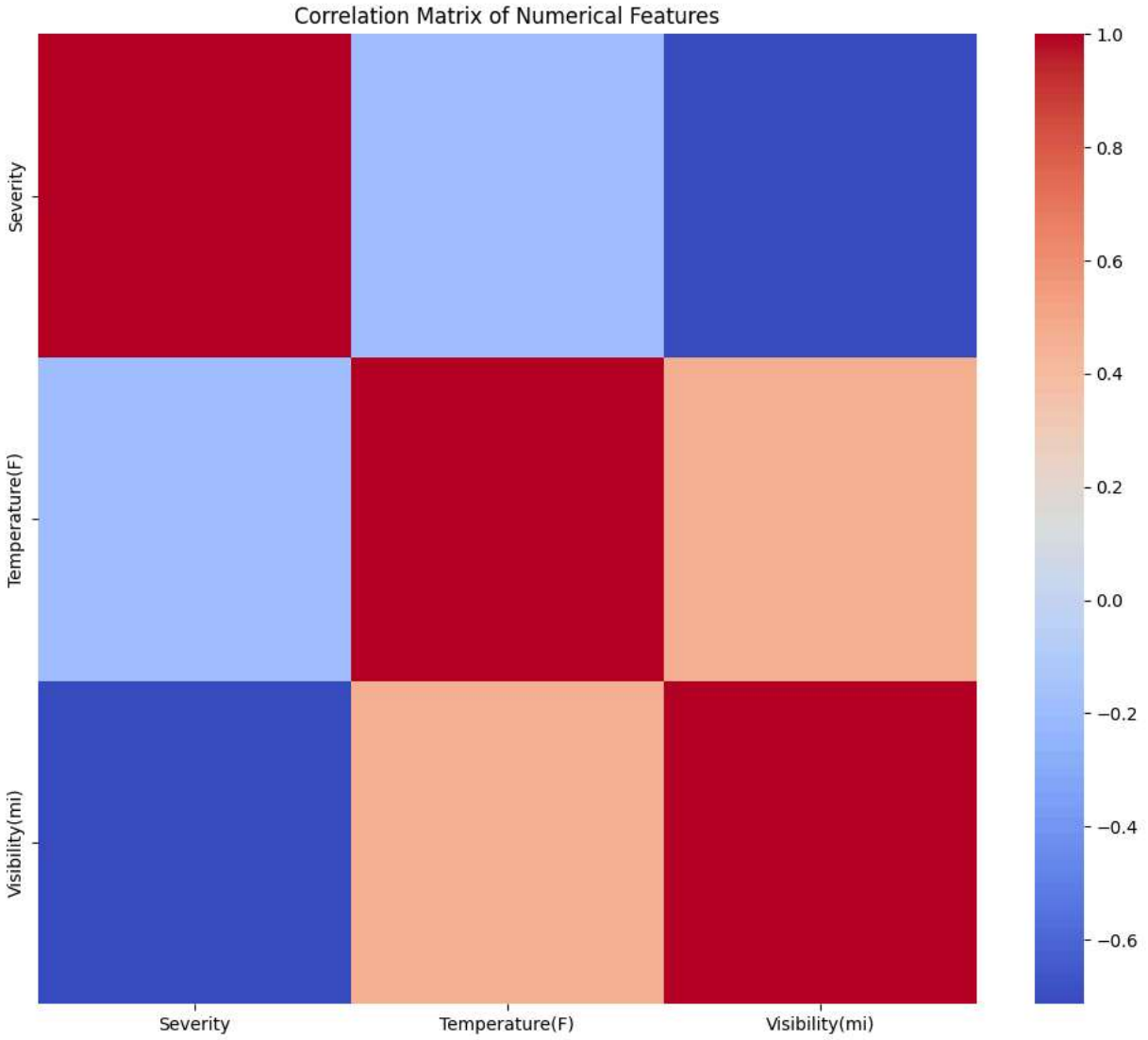

Temperature(F)
1.0
0.8
0.6
0.4
0.2
0.0
65
70
75
80
85
90
95
100
105


Visibility(mi)
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
2
4
6
8
10


Distribution of ID
A-001
A-002
A-003
A-004
A-005
A-006
A-007
A-008
A-009
A-010
ID
0.0
0.2
0.4
0.6
0.8
1.0
count


Distribution of Start_Time
2023-07-01 08:30:00
2023-07-01 17:45:00
2023-07-02 13:10:00
2023-07-03 21:00:00
2023-07-04 14:00:00
2023-07-05 09:15:00
2023-07-06 11:45:00
2023-07-07 06:20:00
2023-07-08 15:00:00
2023-07-09 19:30:00
Start_Time
0.0
0.2
0.4
0.6
0.8
1.0


count
Distribution of End_Time
2023-07-01 09:15:00
2023-07-01 18:30:00
2023-07-02 13:40:00
2023-07-03 21:45:00
2023-07-04 16:00:00
2023-07-05 10:00:00
2023-07-06 12:05:00
2023-07-07 07:00:00
2023-07-08 15:40:00
2023-07-09 20:45:00
End_Time
0.0
0.2
0.4
0.6
0.8
1.0
count

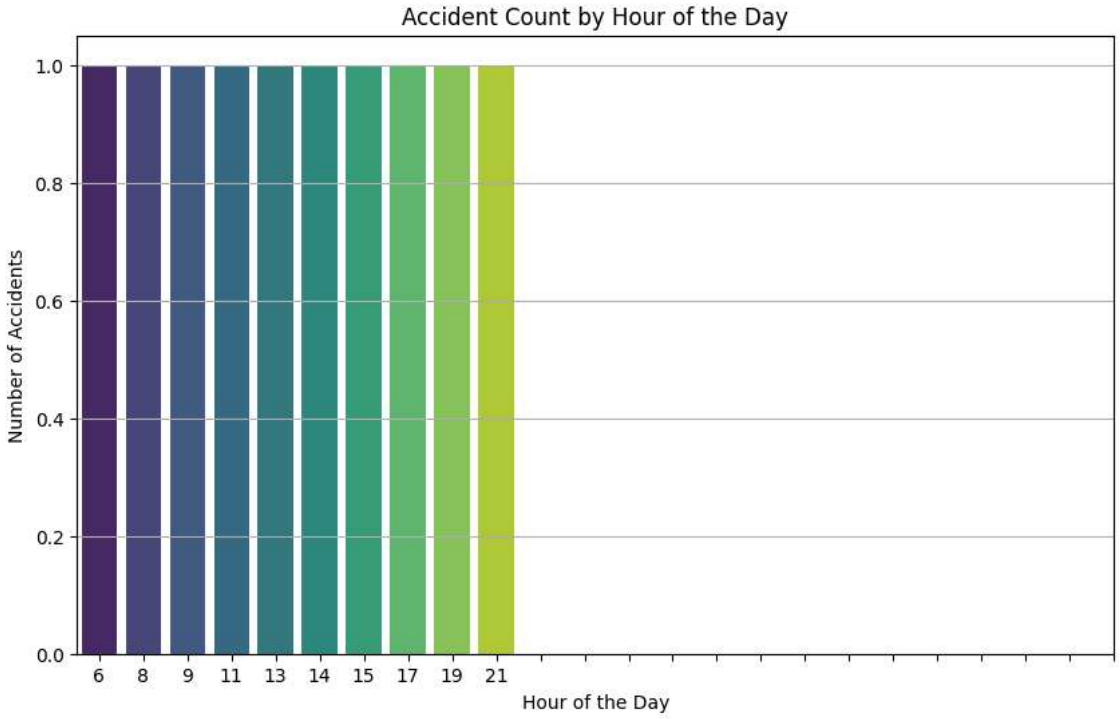

Correlation Matrix of Numerical Features
Severity
Temperature(F)
Visibility(mi)
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
Severity
Temperature(F)
Visibility(mi)

Generate | Time of Day Analysis \\ | Close

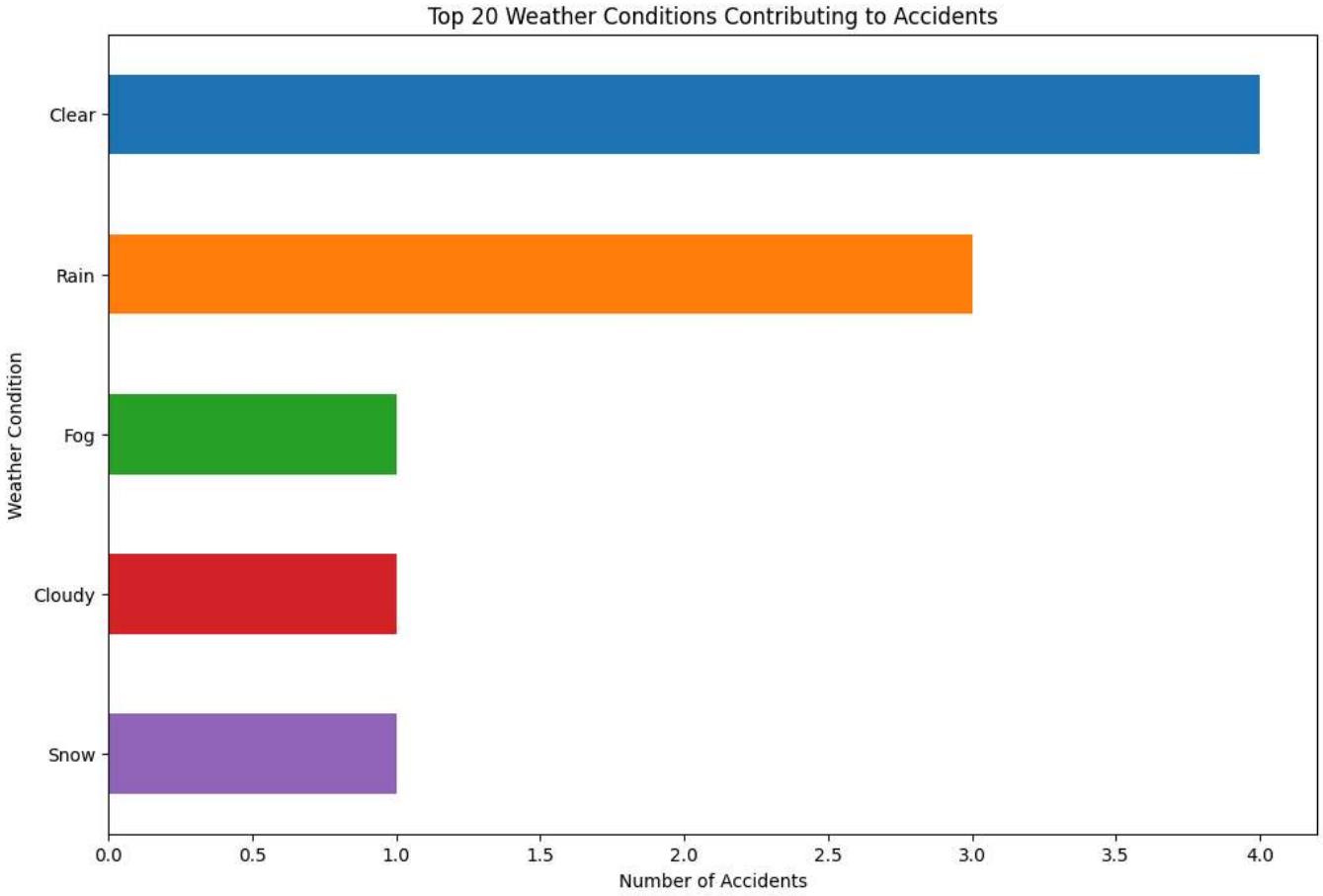
1 of 1 | Undo Changes | Use code with caution

```
import pandas as pd
import matplotlib.pyplot as plt

df['Start_Time'] = pd.to_datetime(df['Start_Time'])

df['Hour'] = df['Start_Time'].dt.hour

plt.figure(figsize=(10, 6))
sns.countplot(data=df, x='Hour', palette='viridis')
plt.title('Accident Count by Hour of the Day')
plt.xlabel('Hour of the Day')
plt.ylabel('Number of Accidents')
plt.xticks(range(0, 24))
plt.grid(axis='y')
plt.show()
```

```
/tmp/ipython-input-5-2134212272.py:9: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `le

  sns.countplot(data=df, x='Hour', palette='viridis')
```

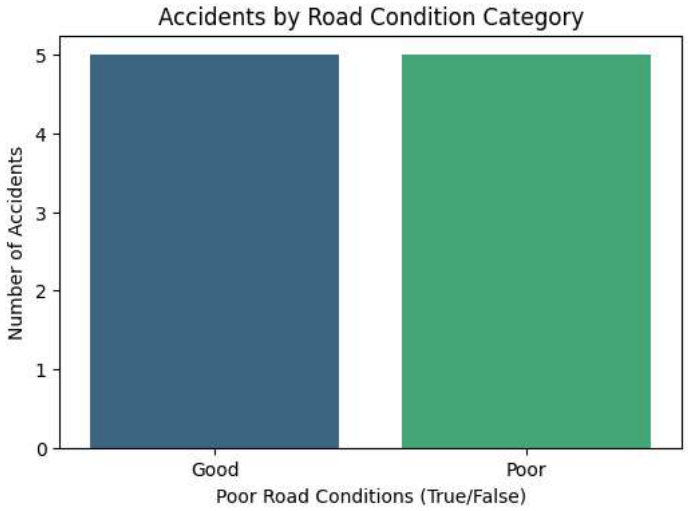


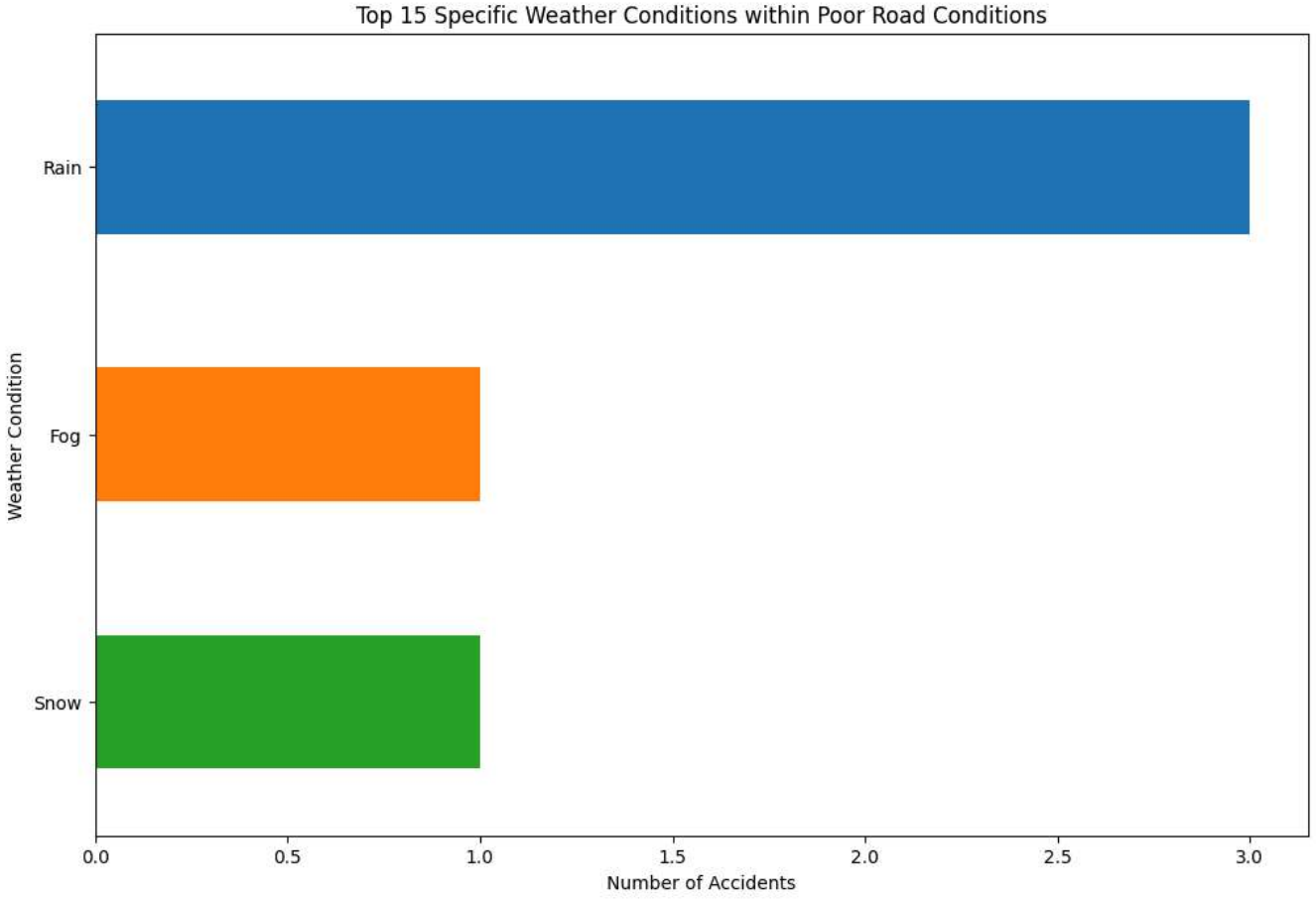

```
import matplotlib.pyplot as plt
weather_conditions = [col for col in df.columns if 'Weather_Condition' in col]
weather_counts = df[weather_conditions].stack().value_counts()

plt.figure(figsize=(12, 8))
weather_counts.head(20).plot(kind='barh', color=sns.color_palette('tab10', 20))
plt.title('Top 20 Weather Conditions Contributing to Accidents')
plt.xlabel('Number of Accidents')
plt.ylabel('Weather Condition')
plt.gca().invert_yaxis()
plt.show()
```




```
import matplotlib.pyplot as plt
poor_conditions_keywords = ['Rain', 'Snow', 'Ice', 'Fog', 'Sleet', 'Hail']

df['Poor_Road_Conditions'] = df['Weather_Condition'].apply(
    lambda x: any(keyword in str(x) for keyword in poor_conditions_keywords)
)

plt.figure(figsize=(6, 4))
sns.countplot(data=df, x='Poor_Road_Conditions', palette='viridis')
plt.title('Accidents by Road Condition Category')
plt.xlabel('Poor Road Conditions (True/False)')
plt.ylabel('Number of Accidents')
plt.xticks(ticks=[0, 1], labels=['Good', 'Poor'])
plt.show()

poor_condition_df = df[df['Poor_Road_Conditions'] == True]

if not poor_condition_df.empty:
    plt.figure(figsize=(12, 8))
    weather_counts_poor = poor_condition_df['Weather_Condition'].value_counts()
    weather_counts_poor.head(15).plot(kind='barh', color=sns.color_palette('tab10', 15))
    plt.title('Top 15 Specific Weather Conditions within Poor Road Conditions')
    plt.xlabel('Number of Accidents')
    plt.ylabel('Weather Condition')
    plt.gca().invert_yaxis()
    plt.show()
else:
    print("No accidents recorded under identified poor road conditions in this sample.")
```

```
/tmp/ipython-input-7-4230000265.py:9: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `

  sns.countplot(data=df, x='Poor_Road_Conditions', palette='viridis')
```





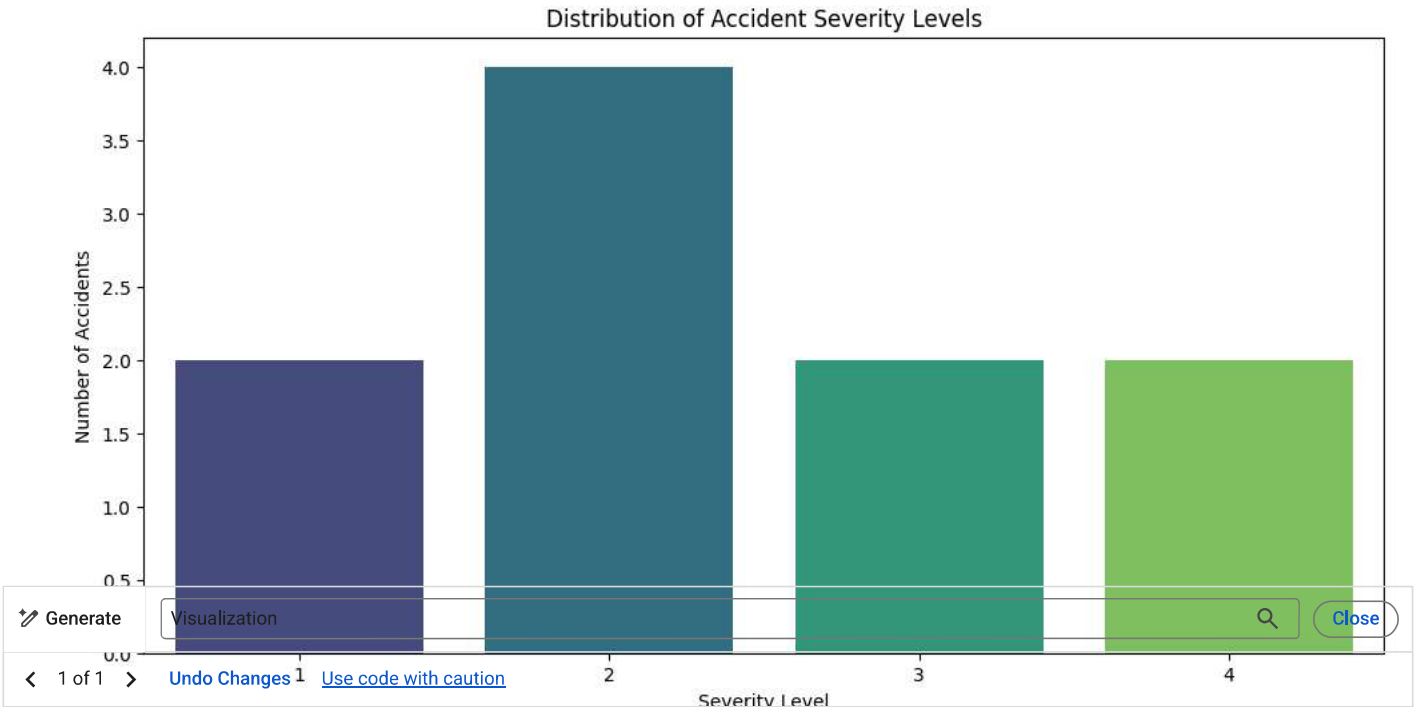
Generate | Accident Severity Factors | Close

1 of 1 | Undo Changes | Use code with caution

```
import matplotlib.pyplot as plt
plt.figure(figsize=(12, 6))
sns.countplot(data=df, x='Severity', palette='viridis')
plt.title('Distribution of Accident Severity Levels')
plt.xlabel('Severity Level')
plt.ylabel('Number of Accidents')
plt.show()

plt.figure(figsize=(12, 8))
sns.countplot(data=df, x='Hour', hue='Severity', palette='viridis')
plt.title('Accident Severity by Hour of the Day')
plt.xlabel('Hour of the Day')
plt.ylabel('Number of Accidents')
```

```
    plt.xticks(range(0, 24))
    plt.legend(title='Severity')
    plt.grid(axis='y')
    plt.show()

    plt.figure(figsize=(8, 5))
    sns.countplot(data=df, x='Poor_Road_Conditions', hue='Severity', palette='viridis')
    plt.title('Accident Severity by Road Condition Category')
    plt.xlabel('Poor Road Conditions (True/False)')
    plt.ylabel('Number of Accidents')
    plt.xticks(ticks=[0, 1], labels=['Good', 'Poor'])
    plt.legend(title='Severity')
    plt.show()
```

```
/tmp/ipython-input-8-1064763514.py:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `

  sns.countplot(data=df, x='Severity', palette='viridis')
```




```
import matplotlib.pyplot as plt
if 'Start_Lat' in df.columns and 'Start_Lng' in df.columns:
  plt.figure(figsize=(10, 8))
  sns.scatterplot(data=df.sample(n=10000, random_state=42),
                  x='Start_Lng', y='Start_Lat', hue='Severity',
                  palette='viridis', alpha=0.6, s=20)
  plt.title('Geographic Distribution of Accidents (Sample)')
  plt.xlabel('Longitude')
  plt.ylabel('Latitude')
  plt.show()
else:
  print("Geographic visualization requires 'Start_Lat' and 'Start_Lng' columns.")

df['Day_of_Week'] = df['Start_Time'].dt.day_name()

plt.figure(figsize=(10, 6))
sns.countplot(data=df, x='Day_of_Week', order=[
              'Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday'], palette='viridis')
plt.title('Accident Count by Day of the Week')
plt.xlabel('Day of the Week')
plt.ylabel('Number of Accidents')
plt.show()

df['Month'] = df['Start_Time'].dt.month_name()

plt.figure(figsize=(10, 6))
sns.countplot(data=df, x='Month', order=[
              'January', 'February', 'March', 'April', 'May', 'June', 'July', 'August', 'September', 'October', 'November', 'December'],
plt.title('Accident Count by Month')
plt.xlabel('Month')
plt.ylabel('Number of Accidents')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()

if 'Hour' in df.columns and 'Day_of_Week' in df.columns:
  accident_by_time_day = df.groupby(['Day_of_Week', 'Hour']).size().unstack(fill_value=0)

  accident_by_time_day = accident_by_time_day.reindex(
      columns=range(0, 24),
      index=['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday']
  )

  plt.figure(figsize=(14, 8))
  sns.heatmap(accident_by_time_day, cmap='viridis')
  plt.title('Accident Count Heatmap by Day of Week and Hour of Day')
  plt.xlabel('Hour of Day')
  plt.ylabel('Day of Week')
  plt.show()
```