

```
In [2]: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import folium
from folium.plugins import HeatMap
```

```
In [3]: plt.style.use('seaborn-v0_8')
sns.set_palette("Set2")
pd.set_option('display.max_columns', None)
```

```
In [4]: df = pd.read_csv("US_Accidents_March23.csv")
df.head()
```

```
Out[4]:
```

	ID	Source	Severity	Start_Time	End_Time	Start_Lat	Start_Lng	End_Lat	End_Lng
0	A-1	Source2	3	2016-02-08 05:46:00	2016-02-08 11:00:00	39.865147	-84.058723	NaN	NaN
1	A-2	Source2	2	2016-02-08 06:07:59	2016-02-08 06:37:59	39.928059	-82.831184	NaN	NaN
2	A-3	Source2	2	2016-02-08 06:49:27	2016-02-08 07:19:27	39.063148	-84.032608	NaN	NaN
3	A-4	Source2	3	2016-02-08 07:23:34	2016-02-08 07:53:34	39.747753	-84.205582	NaN	NaN
4	A-5	Source2	2	2016-02-08 07:39:07	2016-02-08 08:09:07	39.627781	-84.188354	NaN	NaN



```
In [5]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7728394 entries, 0 to 7728393
Data columns (total 46 columns):
 #   Column            Dtype  
 --- 
 0   ID                object  
 1   Source             object  
 2   Severity           int64   
 3   Start_Time         object  
 4   End_Time           object  
 5   Start_Lat          float64 
 6   Start_Lng          float64 
 7   End_Lat            float64 
 8   End_Lng            float64 
 9   Distance(mi)      float64 
 10  Description        object  
 11  Street             object  
 12  City               object  
 13  County              object  
 14  State               object  
 15  Zipcode             object  
 16  Country              object  
 17  Timezone            object  
 18  Airport_Code        object  
 19  Weather_Timestamp   object  
 20  Temperature(F)     float64 
 21  Wind_Chill(F)      float64 
 22  Humidity(%)        float64 
 23  Pressure(in)       float64 
 24  Visibility(mi)     float64 
 25  Wind_Direction      object  
 26  Wind_Speed(mph)    float64 
 27  Precipitation(in)  float64 
 28  Weather_Condition  object  
 29  Amenity             bool    
 30  Bump                bool    
 31  Crossing            bool    
 32  Give_Way            bool    
 33  Junction            bool    
 34  No_Exit             bool    
 35  Railway              bool    
 36  Roundabout           bool    
 37  Station              bool    
 38  Stop                bool    
 39  Traffic_Calming     bool    
 40  Traffic_Signal       bool    
 41  Turning_Loop          bool    
 42  Sunrise_Sunset        object  
 43  Civil_Twilight        object  
 44  Nautical_Twilight    object  
 45  Astronomical_Twilight object  
dtypes: bool(13), float64(12), int64(1), object(20)
memory usage: 2.0+ GB
```

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

Out[6]:

	Severity	Start_Lat	Start_Lng	End_Lat	End_Lng	Distance (
count	7.728394e+06	7.728394e+06	7.728394e+06	4.325632e+06	4.325632e+06	7.728394e
mean	2.212384e+00	3.620119e+01	-9.470255e+01	3.626183e+01	-9.572557e+01	5.618423e
std	4.875313e-01	5.076079e+00	1.739176e+01	5.272905e+00	1.810793e+01	1.776811e
min	1.000000e+00	2.455480e+01	-1.246238e+02	2.456601e+01	-1.245457e+02	0.000000e
25%	2.000000e+00	3.339963e+01	-1.172194e+02	3.346207e+01	-1.177543e+02	0.000000e
50%	2.000000e+00	3.582397e+01	-8.776662e+01	3.618349e+01	-8.802789e+01	3.000000e
75%	2.000000e+00	4.008496e+01	-8.035368e+01	4.017892e+01	-8.024709e+01	4.640000e
max	4.000000e+00	4.900220e+01	-6.711317e+01	4.907500e+01	-6.710924e+01	4.417500e

◀ ▶

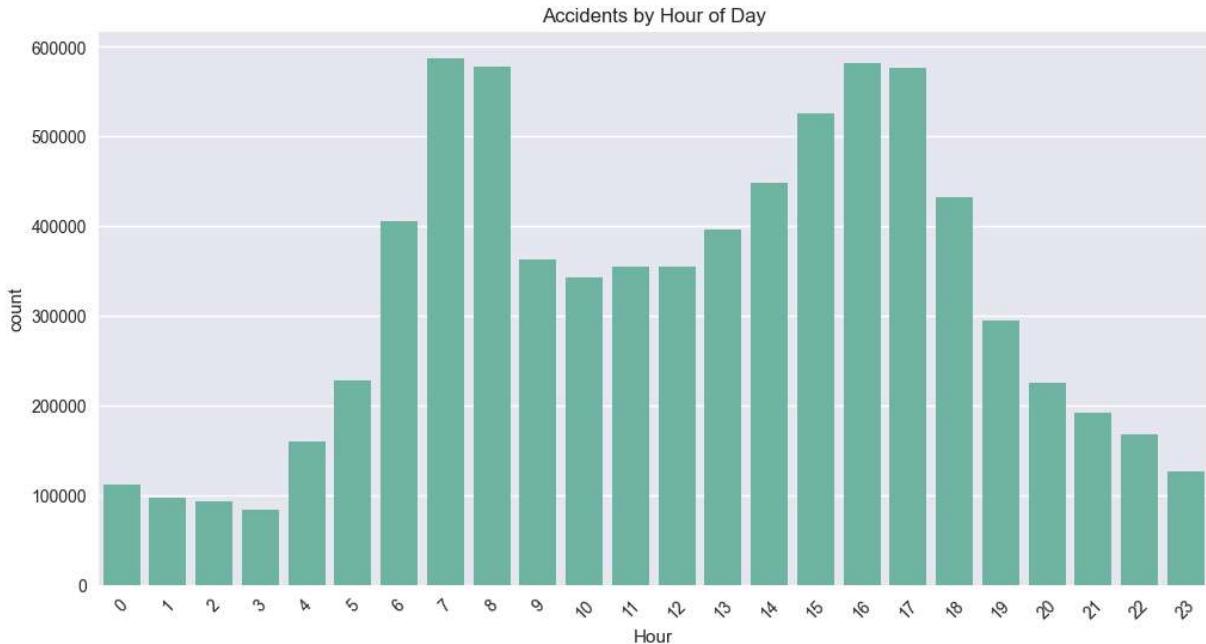
In [8]: `df['Start_Time'] = pd.to_datetime(df['Start_Time'], format='mixed')
df['End_Time'] = pd.to_datetime(df['End_Time'], format='mixed')`

In [9]: `df['Year'] = df['Start_Time'].dt.year
df['Month'] = df['Start_Time'].dt.month
df['Day'] = df['Start_Time'].dt.day
df['Hour'] = df['Start_Time'].dt.hour
df['DayOfWeek'] = df['Start_Time'].dt.day_name()`

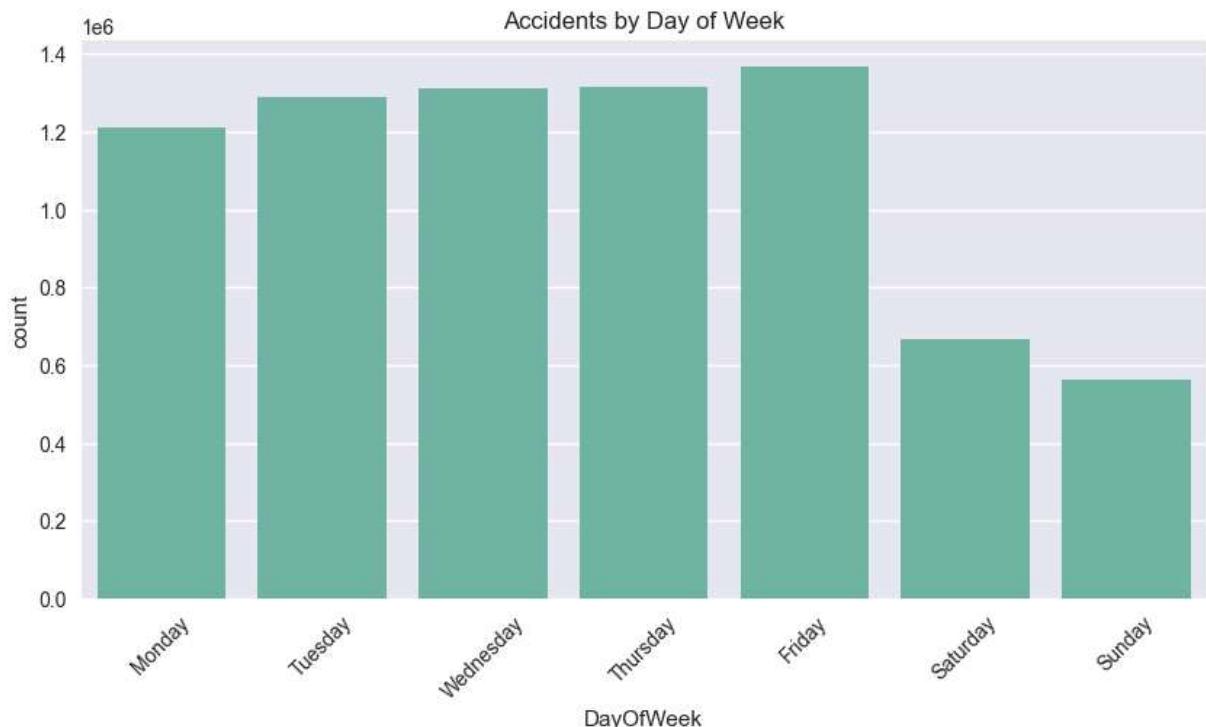
In [11]: `def time_of_day(hour):
 if 5 <= hour < 12:
 return 'Morning'
 elif 12 <= hour < 17:
 return 'Afternoon'
 elif 17 <= hour < 21:
 return 'Evening'
 else:
 return 'Night'

df['Time_of_Day'] = df['Hour'].apply(time_of_day)`

In [12]: `plt.figure(figsize=(12,6))
sns.countplot(x='Hour', data=df)
plt.title("Accidents by Hour of Day")
plt.xticks(rotation=45)
plt.show()`



```
In [13]: plt.figure(figsize=(10,5))
sns.countplot(x='DayOfWeek', data=df, order=['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday'])
plt.title("Accidents by Day of Week")
plt.xticks(rotation=45)
plt.show()
```

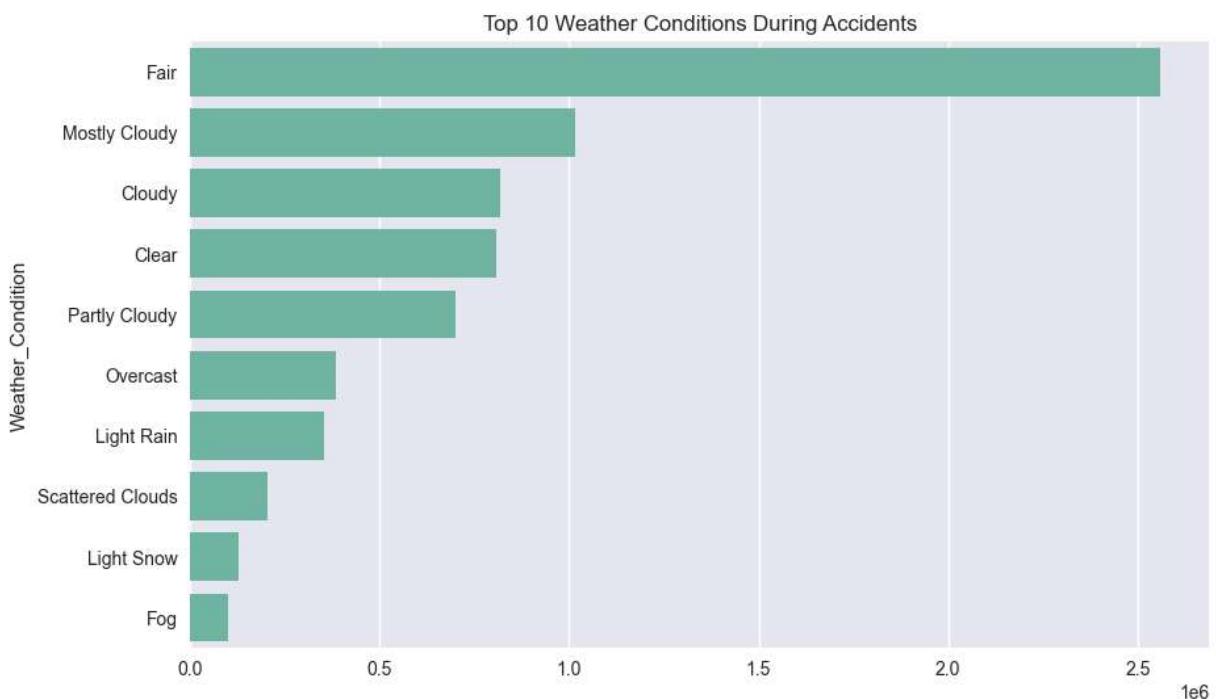


```
In [14]: plt.figure(figsize=(8,5))
sns.countplot(x='Time_of_Day', data=df)
plt.title("Accidents by Time of Day")
plt.show()
```



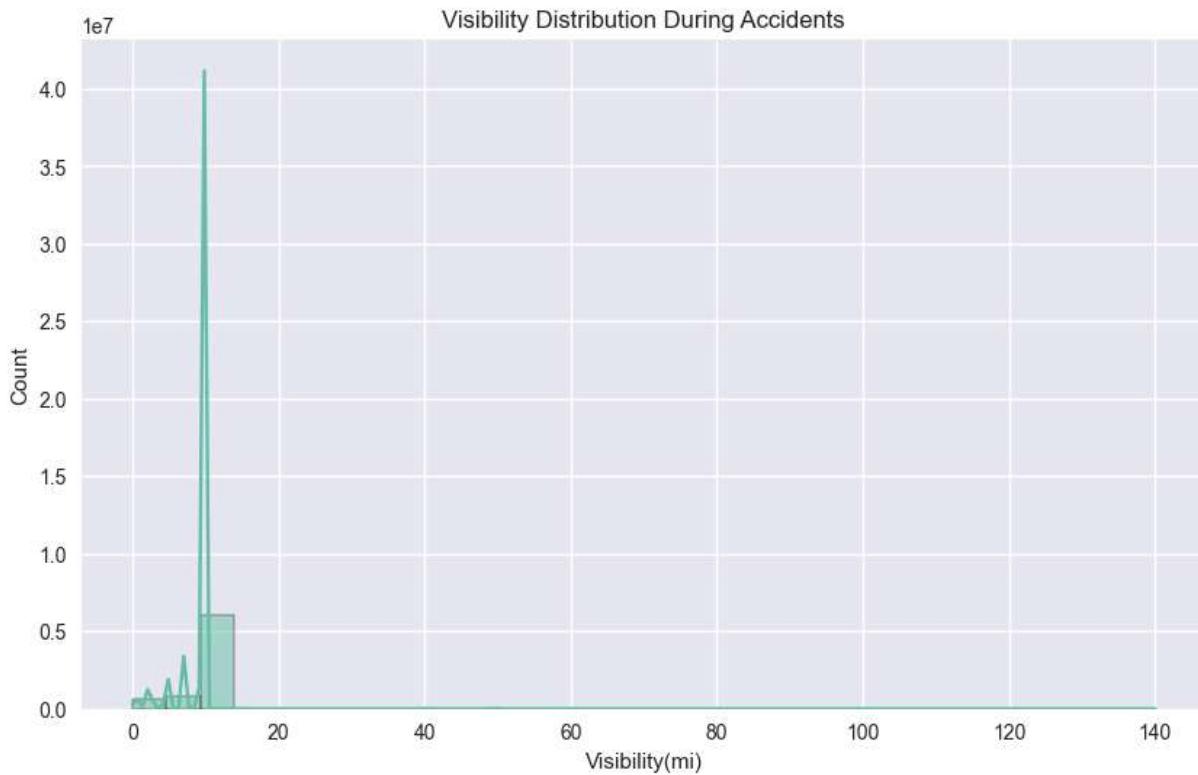
```
In [15]: weather_counts = df['Weather_Condition'].value_counts().head(10)

plt.figure(figsize=(10,6))
sns.barplot(x=weather_counts.values, y=weather_counts.index)
plt.title("Top 10 Weather Conditions During Accidents")
plt.show()
```

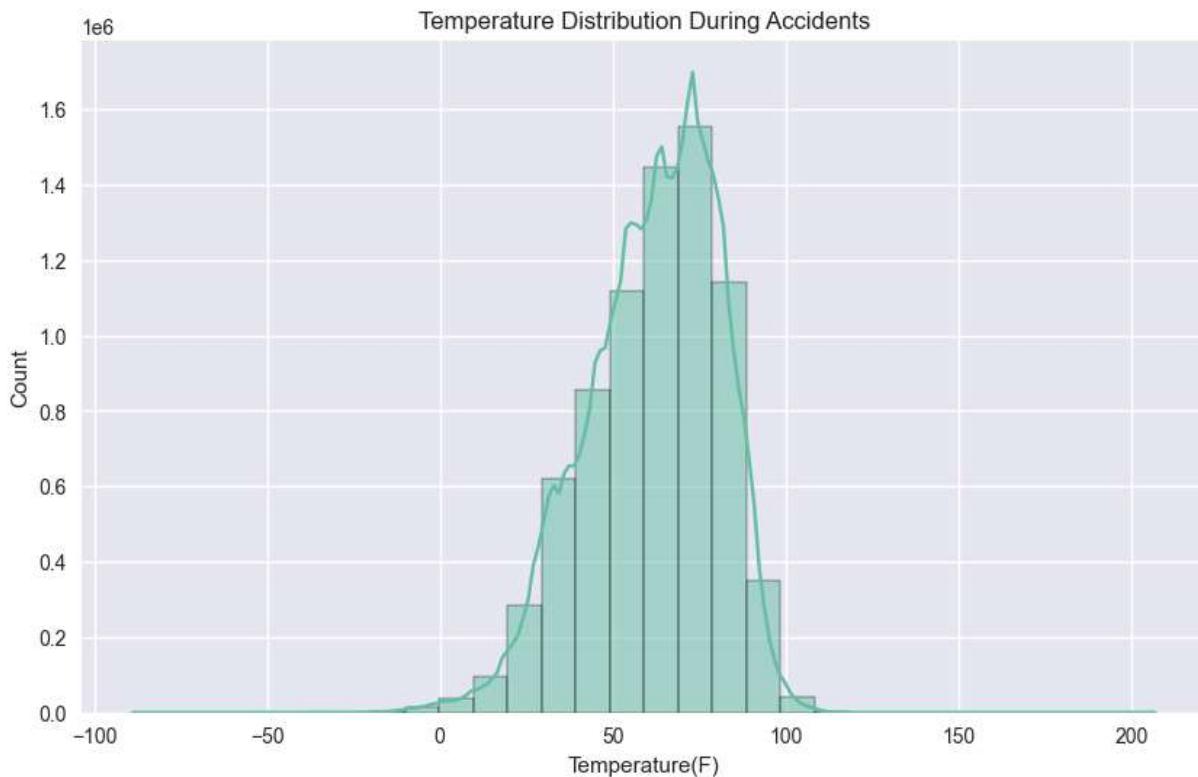


```
In [17]: plt.figure(figsize=(10,6))
sns.histplot(df['Visibility(mi)'].dropna(), bins=30, kde=True)
```

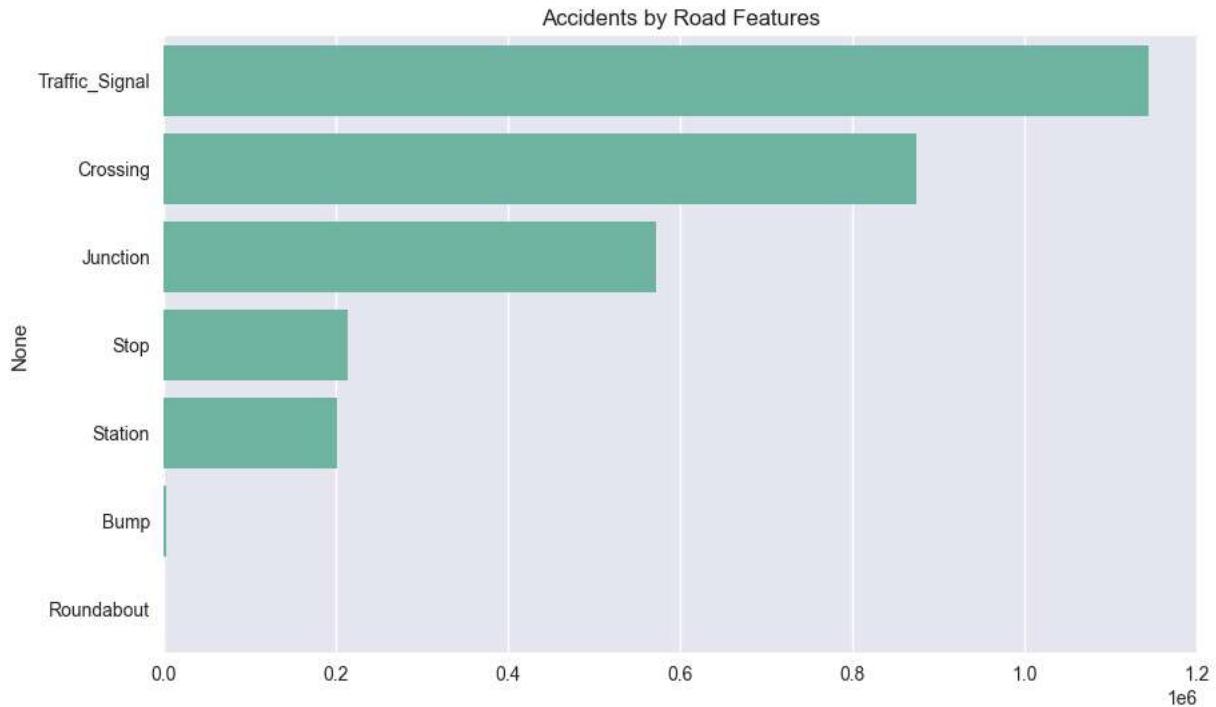
```
plt.title("Visibility Distribution During Accidents")
plt.show()
```



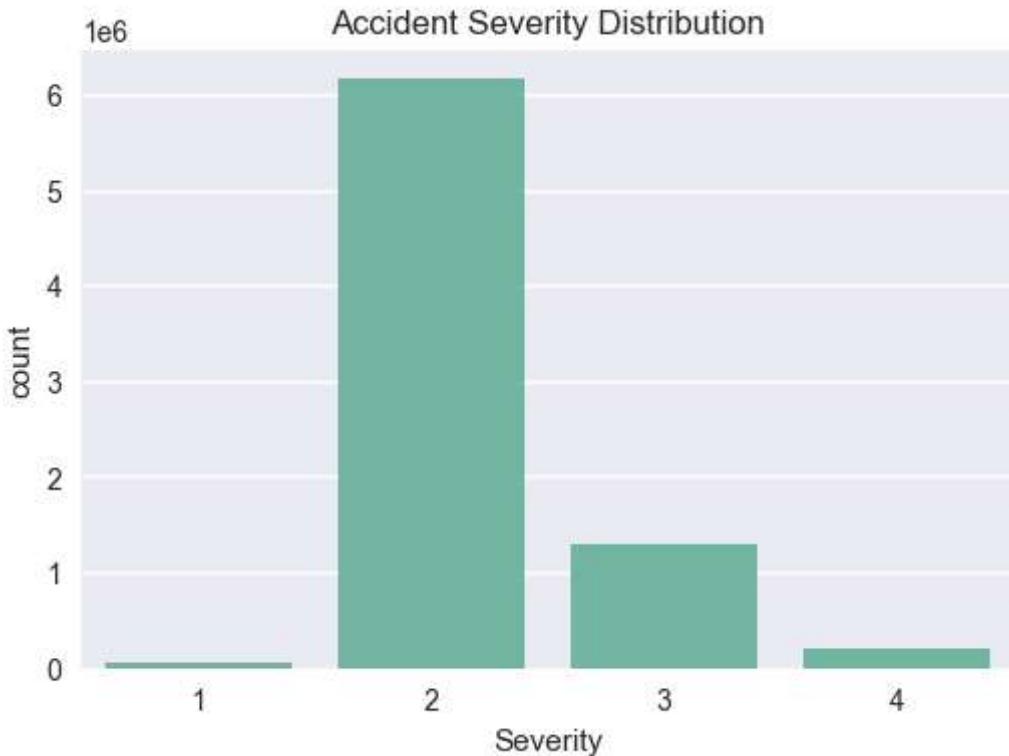
```
In [21]: plt.figure(figsize=(10,6))
sns.histplot(df['Temperature(F)'].dropna(), bins=30, kde=True)
plt.title("Temperature Distribution During Accidents")
plt.show()
```



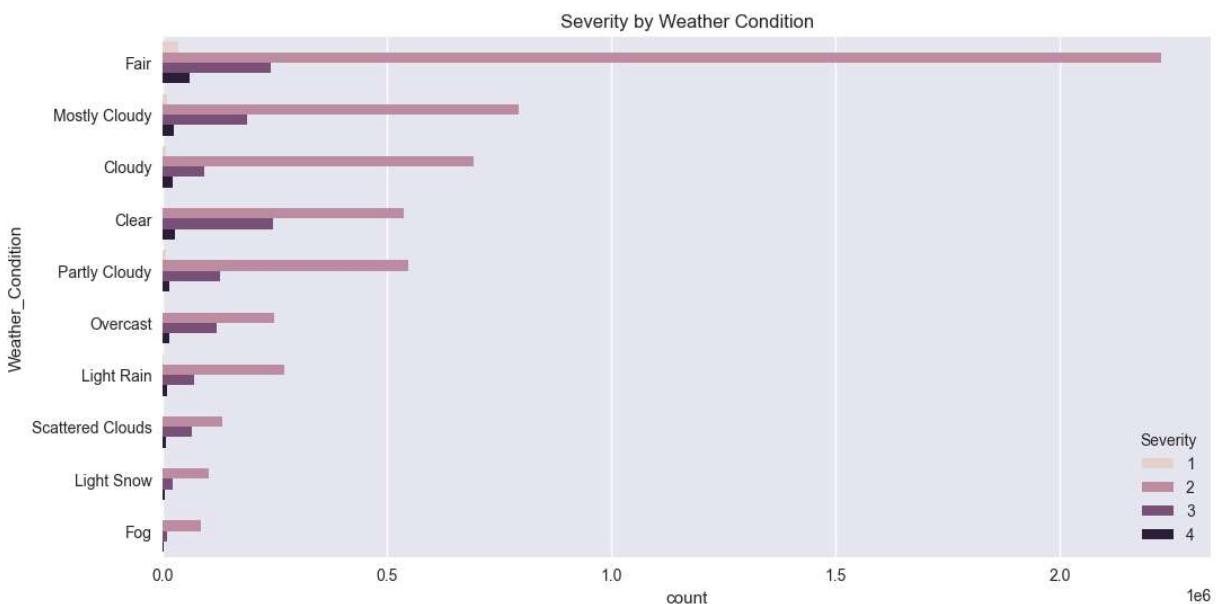
```
In [22]: road_features = ['Bump', 'Junction', 'Traffic_Signal', 'Roundabout', 'Station', 'Cr\n\nfeature_counts = df[road_features].sum().sort_values(ascending=False)\n\nplt.figure(figsize=(10,6))\nsns.barplot(x=feature_counts.values, y=feature_counts.index)\nplt.title("Accidents by Road Features")\nplt.show()
```



```
In [23]: plt.figure(figsize=(6,4))\nsns.countplot(x='Severity', data=df)\nplt.title("Accident Severity Distribution")\nplt.show()
```



```
In [25]: plt.figure(figsize=(12,6))
sns.countplot(y='Weather_Condition', data=df,
               order=df['Weather_Condition'].value_counts().iloc[:10].index,
               hue='Severity')
plt.title("Severity by Weather Condition")
plt.show()
```



```
In [28]: sample_df = df.sample(10000)

fig = px.scatter_map(
    sample_df,
    lat="Start_Lat",
    lon="Start_Lng",
```

```
        color="Severity",
        zoom=3,
        title="Accidents Hotspots Map"
    )

fig.show()
```

Accidents Hotspots Map

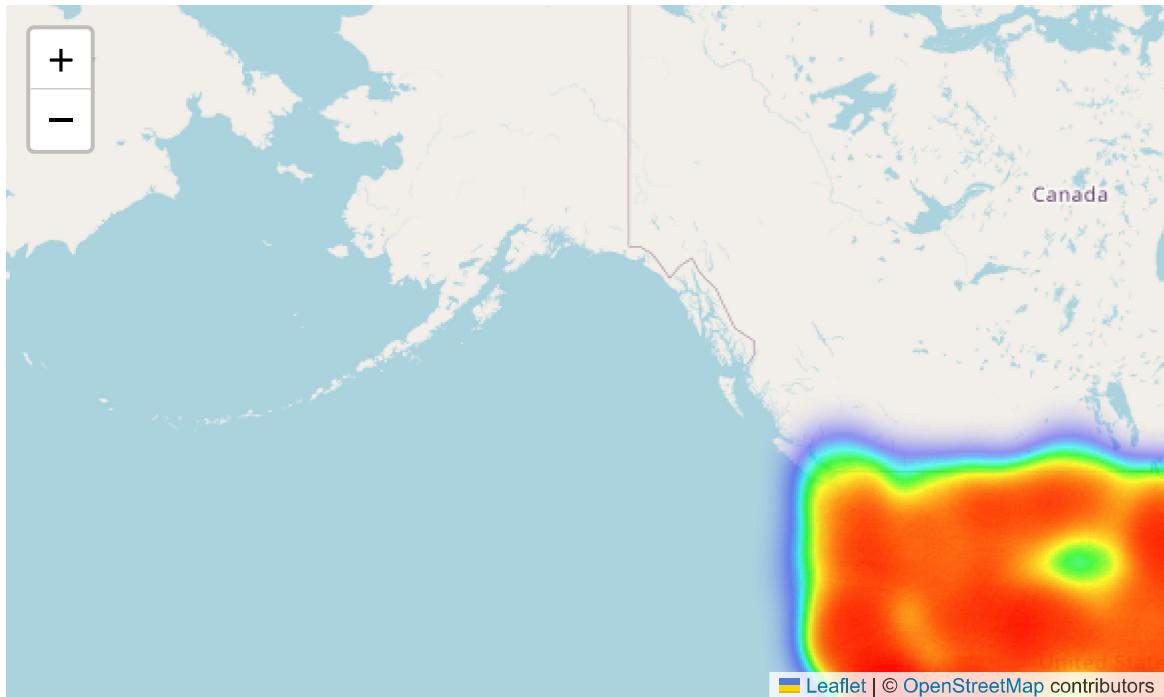


```
In [29]: map_centre = [df['Start_Lat'].mean(), df['Start_Lng'].mean()]
m = folium.Map(location=map_centre, zoom_start=4)

heat_data = df[['Start_Lat', 'Start_Lng']].dropna().sample(20000).values.tolist()
HeatMap(heat_data).add_to(m)

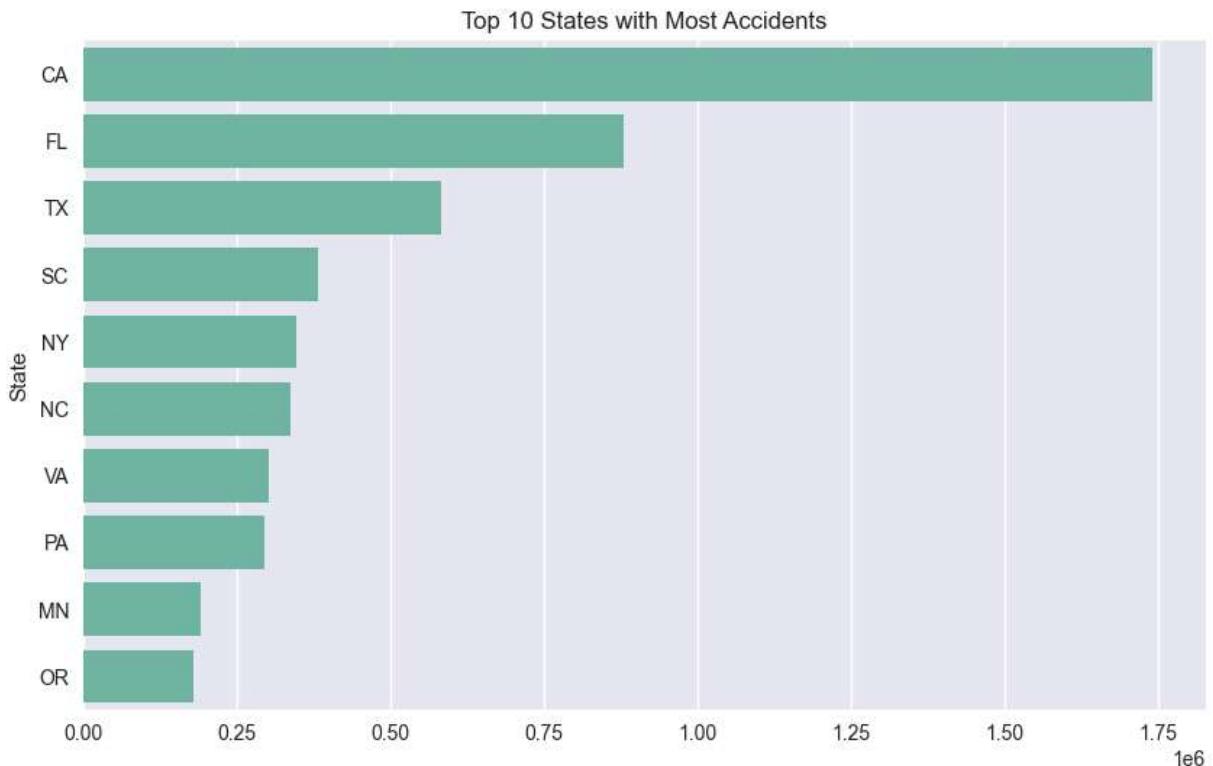
m
```

Out[29]:



```
In [30]: state_counts = df['State'].value_counts().head(10)

plt.figure(figsize=(10,6))
sns.barplot(x=state_counts.values, y=state_counts.index)
plt.title("Top 10 States with Most Accidents")
plt.show()
```



```
In [31]: numeric_cols = ['Temperature(F)', 'Humidity(%)', 'Pressure(in)',  
                   'Visibility(mi)', 'Wind_Speed(mph)', 'Severity']  
  
plt.figure(figsize=(10,6))  
sns.heatmap(df[numeric_cols].corr(), annot=True, cmap='coolwarm')  
plt.title("Correlation Heatmap")  
plt.show()
```

