In [12]:
```
import pandas as pd
data_filepath = ('D:/prodigy/task5 accient/US_Accidents_March23_sampled_500k.csv')
df = pd.read_csv(data_filepath)
df.head(10)
```

Out[12]:

	ID	Source	Severity	Start_Time	End_Time	Start_Lat	Start_Ln
0	A-2047758	Source2	2	2019-06-12 10:10:56	2019-06-12 10:55:58	30.641211	-91.15348
1	A-4694324	Source1	2	2022-12-03 23:37:14.000000000	2022-12-04 01:56:53.000000000	38.990562	-77.39907
2	A-5006183	Source1	2	2022-08-20 13:13:00.000000000	2022-08-20 15:22:45.000000000	34.661189	-120.49282
3	A-4237356	Source1	2	2022-02-21 17:43:04	2022-02-21 19:43:23	43.680592	-92.99331
4	A-6690583	Source1	2	2020-12-04 01:46:00	2020-12-04 04:13:09	35.395484	-118.98517
5	A-1101469	Source2	2	2021-03-29 07:03:58	2021-03-29 08:51:01	42.532082	-70.94426
6	A-7222249	Source1	2	2020-01-14 16:49:23	2020-01-14 20:49:23	42.421280	-123.11945
7	A-6198239	Source1	2	2021-08-13 16:48:00.000000000	2021-08-13 19:09:09.000000000	30.191010	-85.68250
8	A-4222549	Source1	2	2022-10-12 13:59:30	2022-10-12 15:33:53	32.868947	-96.80401
9	A-5924038	Source1	2	2021-10-21 07:39:30	2021-10-21 09:24:30	39.717218	-86.12469

10 rows × 46 columns

In [13]:
```
print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 500000 entries, 0 to 499999
Data columns (total 46 columns):
 #   Column                 Non-Null Count   Dtype  
---  ------                 --------------   -----  
 0   ID                     500000 non-null  object 
 1   Source                 500000 non-null  object 
 2   Severity               500000 non-null  int64  
 3   Start_Time             500000 non-null  object 
 4   End_Time               500000 non-null  object 
 5   Start_Lat              500000 non-null  float64
 6   Start_Lng              500000 non-null  float64
 7   End_Lat                279623 non-null  float64
 8   End_Lng                279623 non-null  float64
 9   Distance(mi)           500000 non-null  float64
 10  Description            499999 non-null  object 
 11  Street                 499309 non-null  object 
 12  City                   499981 non-null  object 
 13  County                 500000 non-null  object 
 14  State                  500000 non-null  object 
 15  Zipcode                499884 non-null  object 
 16  Country                500000 non-null  object 
 17  Timezone               499493 non-null  object 
 18  Airport_Code           498554 non-null  object 
 19  Weather_Timestamp      492326 non-null  object 
 20  Temperature(F)         489534 non-null  float64
 21  Wind_Chill(F)          370983 non-null  float64
 22  Humidity(%)            488870 non-null  float64
 23  Pressure(in)           491072 non-null  float64
 24  Visibility(mi)         488709 non-null  float64
 25  Wind_Direction         488803 non-null  object 
 26  Wind_Speed(mph)        463013 non-null  float64
 27  Precipitation(in)      357384 non-null  float64
 28  Weather_Condition      488899 non-null  object 
 29  Amenity                500000 non-null  bool   
 30  Bump                   500000 non-null  bool   
 31  Crossing               500000 non-null  bool   
 32  Give_Way               500000 non-null  bool   
 33  Junction               500000 non-null  bool   
 34  No_Exit                500000 non-null  bool   
 35  Railway                500000 non-null  bool   
 36  Roundabout             500000 non-null  bool   
 37  Station                500000 non-null  bool   
 38  Stop                   500000 non-null  bool   
 39  Traffic_Calming        500000 non-null  bool   
 40  Traffic_Signal         500000 non-null  bool   
 41  Turning_Loop           500000 non-null  bool   
 42  Sunrise_Sunset         498517 non-null  object 
 43  Civil_Twilight         498517 non-null  object 
 44  Nautical_Twilight      498517 non-null  object 
 45  Astronomical_Twilight  498517 non-null  object 
dtypes: bool(13), float64(12), int64(1), object(20)
memory usage: 132.1+ MB
None
```

In [14]:
```
import pandas as pd
file_path = 'D:/prodigy/task5 accient/US_Accidents_March23_sampled_500k.csv'
df = pd.read_csv(file_path)
df['Start_Time'] = pd.to_datetime(df['Start_Time'], errors='coerce')
df['End_Time'] = pd.to_datetime(df['End_Time'], errors='coerce')
print(df.head())
```

```
          ID   Source  Severity          Start_Time            End_Time  \
0  A-2047758  Source2         2 2019-06-12 10:10:56 2019-06-12 10:55:58
1  A-4694324  Source1         2                 NaT                 NaT
2  A-5006183  Source1         2                 NaT                 NaT
3  A-4237356  Source1         2 2022-02-21 17:43:04 2022-02-21 19:43:23
4  A-6690583  Source1         2 2020-12-04 01:46:00 2020-12-04 04:13:09

   Start_Lat   Start_Lng    End_Lat     End_Lng  Distance(mi)  ... Roundabout  \
0  30.641211  -91.153481        NaN         NaN         0.000  ...      False
1  38.990562  -77.399070  38.990037  -77.398282         0.056  ...      False
2  34.661189 -120.492822  34.661189 -120.492442         0.022  ...      False
3  43.680592  -92.993317  43.680574  -92.972223         1.054  ...      False
4  35.395484 -118.985176  35.395476 -118.985995         0.046  ...      False

  Station   Stop Traffic_Calming Traffic_Signal Turning_Loop Sunrise_Sunset  \
0   False  False           False           True        False            Day
1   False  False           False          False        False          Night
2   False  False           False           True        False            Day
3   False  False           False          False        False            Day
4   False  False           False          False        False          Night

  Civil_Twilight Nautical_Twilight Astronomical_Twilight
0            Day               Day                   Day
1          Night             Night                 Night
2            Day               Day                   Day
3            Day               Day                   Day
4          Night             Night                 Night

[5 rows x 46 columns]
```

In [15]:
```
import pandas as pd
file_path = 'D:/prodigy/task5 accient/US_Accidents_March23_sampled_500k.csv'
df = pd.read_csv(file_path)
df['Start_Time'] = pd.to_datetime(df['Start_Time'], errors='coerce')
df['End_Time'] = pd.to_datetime(df['End_Time'], errors='coerce')
print(df.head())
```

```
          ID   Source  Severity          Start_Time            End_Time  \
0  A-2047758  Source2         2 2019-06-12 10:10:56 2019-06-12 10:55:58
1  A-4694324  Source1         2                 NaT                 NaT
2  A-5006183  Source1         2                 NaT                 NaT
3  A-4237356  Source1         2 2022-02-21 17:43:04 2022-02-21 19:43:23
4  A-6690583  Source1         2 2020-12-04 01:46:00 2020-12-04 04:13:09

   Start_Lat   Start_Lng    End_Lat     End_Lng  Distance(mi)  ... Roundabout  \
0  30.641211  -91.153481        NaN         NaN         0.000  ...      False
1  38.990562  -77.399070  38.990037  -77.398282         0.056  ...      False
2  34.661189 -120.492822  34.661189 -120.492442         0.022  ...      False
3  43.680592  -92.993317  43.680574  -92.972223         1.054  ...      False
4  35.395484 -118.985176  35.395476 -118.985995         0.046  ...      False

  Station   Stop Traffic_Calming Traffic_Signal Turning_Loop Sunrise_Sunset  \
0   False  False           False           True        False            Day
1   False  False           False          False        False          Night
2   False  False           False           True        False            Day
3   False  False           False          False        False            Day
4   False  False           False          False        False          Night

  Civil_Twilight Nautical_Twilight Astronomical_Twilight
0            Day               Day                   Day
1          Night             Night                 Night
2            Day               Day                   Day
3            Day               Day                   Day
4          Night             Night                 Night

[5 rows x 46 columns]
```

In [16]:
```
missing_values = df.isnull().sum()
print("Columns with missing values:")
print(missing_values[missing_values > 0])
df['Weather_Condition'].fillna('Unknown', inplace=True)
df['Wind_Speed(mph)'].fillna(df['Wind_Speed(mph)'].mean(), inplace=True)
df.dropna(subset=['Start_Lat', 'Start_Lng'], inplace=True)
print(df.info())
```

```
Columns with missing values:
Start_Time                48163
End_Time                  48163
End_Lat                  220377
End_Lng                  220377
Description                   1
Street                      691
City                         19
Zipcode                     116
Timezone                    507
Airport_Code               1446
Weather_Timestamp          7674
Temperature(F)            10466
Wind_Chill(F)            129017
Humidity(%)               11130
Pressure(in)               8928
Visibility(mi)            11291
Wind_Direction            11197
Wind_Speed(mph)           36987
Precipitation(in)        142616
Weather_Condition         11101
Sunrise_Sunset             1483
Civil_Twilight             1483
Nautical_Twilight          1483
Astronomical_Twilight      1483
dtype: int64
```

```
C:\Users\chomo\AppData\Local\Temp\ipykernel_4256\4066144958.py:9: FutureWarning: A v
alue is trying to be set on a copy of a DataFrame or Series through chained assignme
nt 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.


  df['Weather_Condition'].fillna('Unknown', inplace=True)
C:\Users\chomo\AppData\Local\Temp\ipykernel_4256\4066144958.py:13: FutureWarning: A
value is trying to be set on a copy of a DataFrame or Series through chained assignm
ent 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.


  df['Wind_Speed(mph)'].fillna(df['Wind_Speed(mph)'].mean(), inplace=True)
```

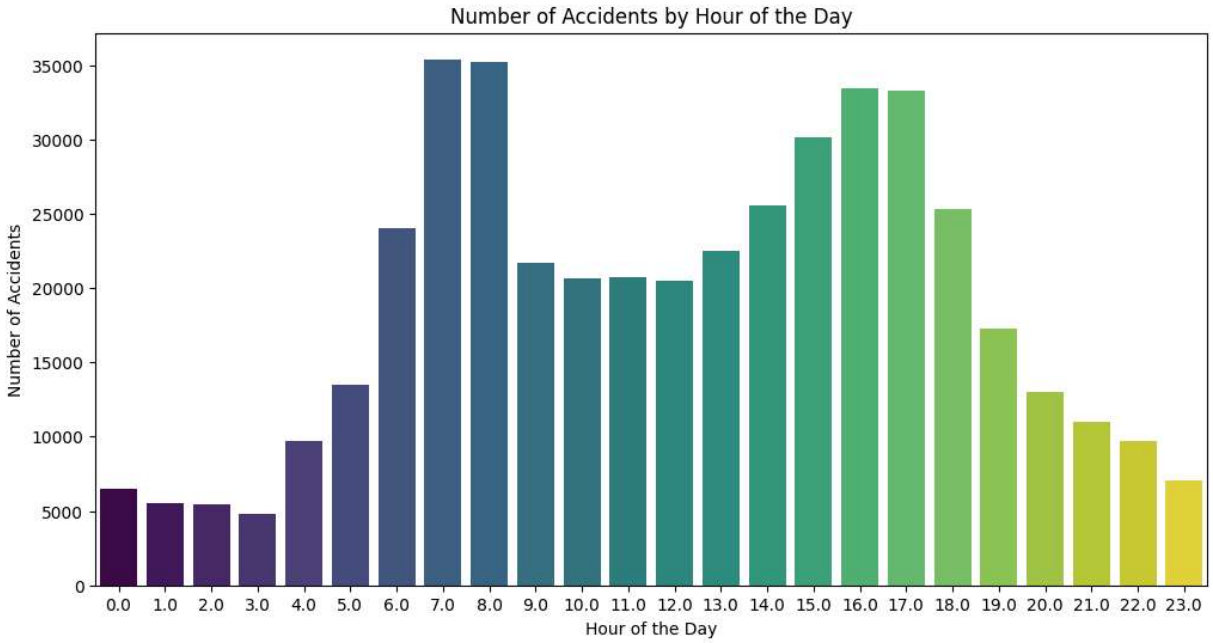
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 500000 entries, 0 to 499999
Data columns (total 46 columns):
 #   Column                 Non-Null Count   Dtype
---  ------                 --------------   -----
 0   ID                     500000 non-null  object
 1   Source                 500000 non-null  object
 2   Severity               500000 non-null  int64
 3   Start_Time             451837 non-null  datetime64[ns]
 4   End_Time               451837 non-null  datetime64[ns]
 5   Start_Lat              500000 non-null  float64
 6   Start_Lng              500000 non-null  float64
 7   End_Lat                279623 non-null  float64
 8   End_Lng                279623 non-null  float64
 9   Distance(mi)           500000 non-null  float64
 10  Description            499999 non-null  object
 11  Street                 499309 non-null  object
 12  City                   499981 non-null  object
 13  County                 500000 non-null  object
 14  State                  500000 non-null  object
 15  Zipcode                499884 non-null  object
 16  Country                500000 non-null  object
 17  Timezone               499493 non-null  object
 18  Airport_Code           498554 non-null  object
 19  Weather_Timestamp      492326 non-null  object
 20  Temperature(F)         489534 non-null  float64
 21  Wind_Chill(F)          370983 non-null  float64
 22  Humidity(%)            488870 non-null  float64
 23  Pressure(in)           491072 non-null  float64
 24  Visibility(mi)         488709 non-null  float64
 25  Wind_Direction         488803 non-null  object
 26  Wind_Speed(mph)        500000 non-null  float64
 27  Precipitation(in)      357384 non-null  float64
 28  Weather_Condition      500000 non-null  object
 29  Amenity                500000 non-null  bool
 30  Bump                   500000 non-null  bool
 31  Crossing               500000 non-null  bool
 32  Give_Way               500000 non-null  bool
 33  Junction               500000 non-null  bool
 34  No_Exit                500000 non-null  bool
 35  Railway                500000 non-null  bool
 36  Roundabout             500000 non-null  bool
 37  Station                500000 non-null  bool
 38  Stop                   500000 non-null  bool
 39  Traffic_Calming        500000 non-null  bool
 40  Traffic_Signal         500000 non-null  bool
 41  Turning_Loop           500000 non-null  bool
 42  Sunrise_Sunset         498517 non-null  object
 43  Civil_Twilight         498517 non-null  object
 44  Nautical_Twilight      498517 non-null  object
 45  Astronomical_Twilight  498517 non-null  object
dtypes: bool(13), datetime64[ns](2), float64(12), int64(1), object(18)
memory usage: 132.1+ MB
None
```

In [17]:
```
df['DayOfWeek'] = df['Start_Time'].dt.dayofweek
df['Hour'] = df['Start_Time'].dt.hour
accidents_by_day = df.groupby('DayOfWeek').size()
accidents_by_hour = df.groupby('Hour').size()
accidents_by_weather = df.groupby('Weather_Condition').size()
print("Accidents by day of the week:")
print(accidents_by_day)
print("\nAccidents by hour:")
print(accidents_by_hour)
print("\nAccidents by weather condition:")
print(accidents_by_weather)
```

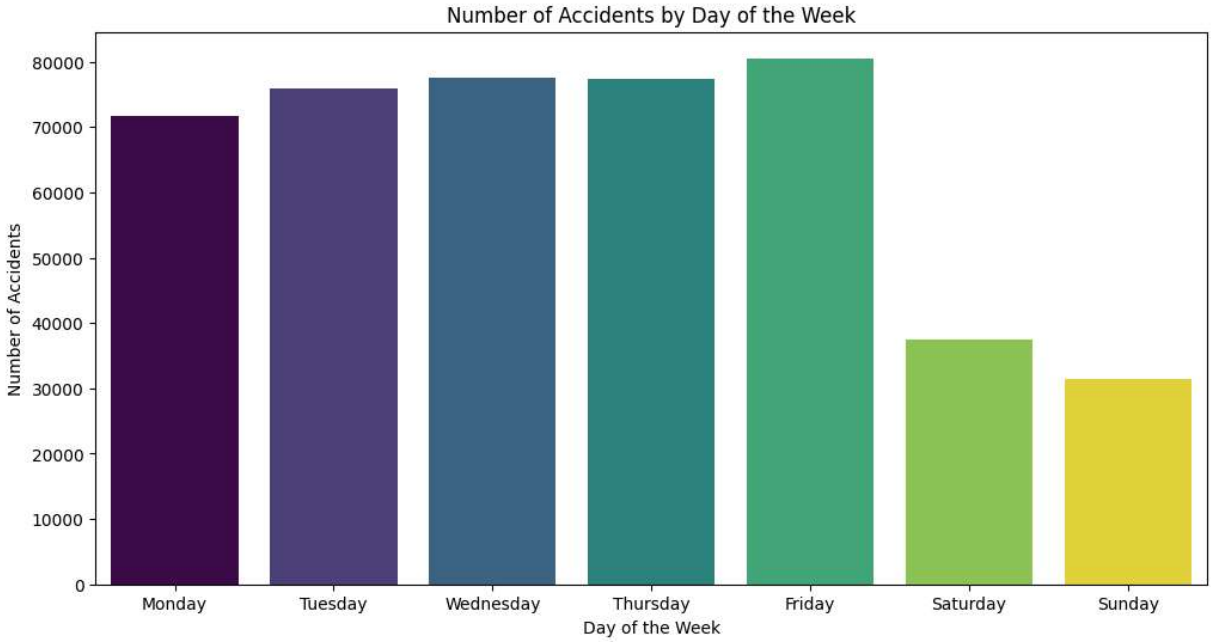
```
Accidents by day of the week:
DayOfWeek
0.0    71717
1.0    75862
2.0    77482
3.0    77404
4.0    80537
5.0    37447
6.0    31388
dtype: int64

Accidents by hour:
Hour
0.0      6502
1.0      5515
2.0      5422
3.0      4787
4.0      9672
5.0     13515
6.0     24038
7.0     35388
8.0     35228
9.0     21694
10.0    20675
11.0    20709
12.0    20513
13.0    22473
14.0    25526
15.0    30160
16.0    33437
17.0    33254
18.0    25307
19.0    17250
20.0    13005
21.0    10972
22.0     9707
23.0     7088
dtype: int64

Accidents by weather condition:
Weather_Condition
Blowing Dust                  16
Blowing Dust / Windy          17
Blowing Snow                  42
Blowing Snow / Windy          57
Clear                      52379
                           ...
Volcanic Ash                   1
Widespread Dust               11
Widespread Dust / Windy        3
Wintry Mix                   777
Wintry Mix / Windy            17
Length: 109, dtype: int64
```

In [57]:
```
import matplotlib.pyplot as plt
import seaborn as sns
```

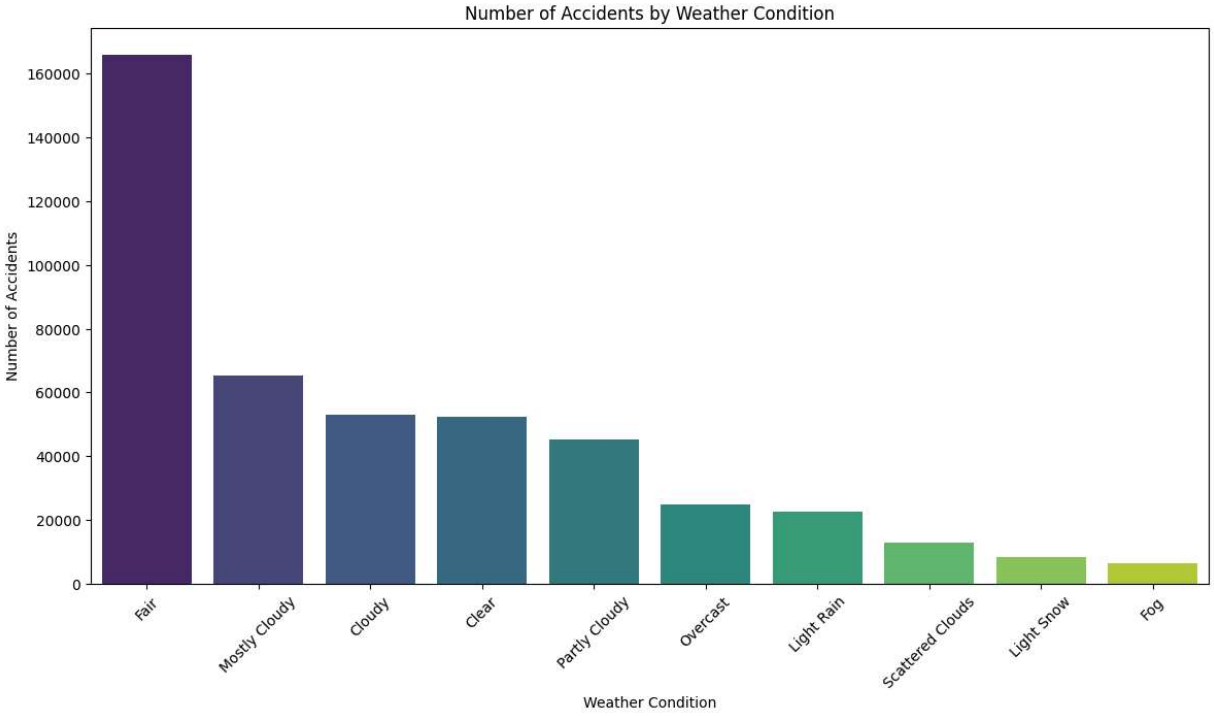
```
df['Hour'] = df['Start_Time'].dt.hour
plt.figure(figsize=(12, 6))
sns.countplot(x='Hour', data=df, hue='Hour', palette='viridis', legend=False)
plt.title('Number of Accidents by Hour of the Day')
plt.xlabel('Hour of the Day')
plt.ylabel('Number of Accidents')
plt.show()
```



In [58]:
```
import matplotlib.pyplot as plt
import seaborn as sns
df['DayOfWeek'] = df['Start_Time'].dt.dayofweek
plt.figure(figsize=(12, 6))
sns.countplot(x='DayOfWeek', data=df, hue='DayOfWeek', palette='viridis', legend=Fa
plt.title('Number of Accidents by Day of the Week')
plt.xlabel('Day of the Week')
plt.ylabel('Number of Accidents')
plt.xticks(ticks=range(7), labels=['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'F
plt.show()
```

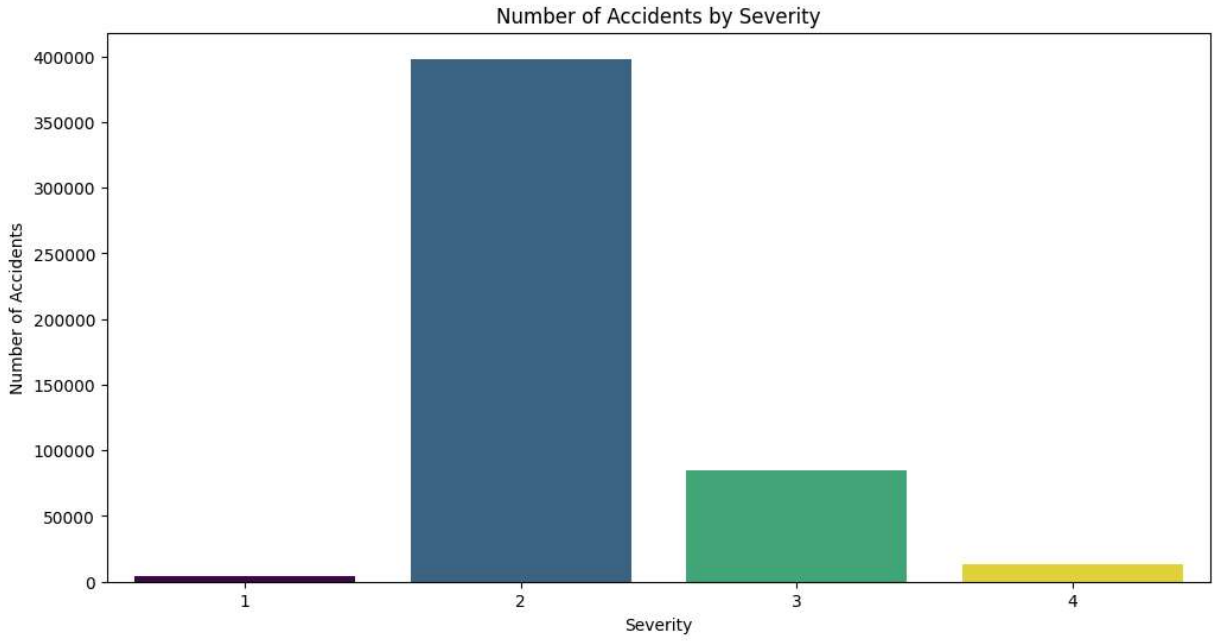


In [59]:
```
import matplotlib.pyplot as plt
import seaborn as sns
plt.figure(figsize=(14, 7))
weather_counts = df['Weather_Condition'].value_counts().head(10)  # Show top 10 wea
sns.barplot(x=weather_counts.index, y=weather_counts.values, hue=weather_counts.ind
plt.title('Number of Accidents by Weather Condition')
plt.xlabel('Weather Condition')
plt.ylabel('Number of Accidents')
plt.xticks(rotation=45)
plt.show()
```

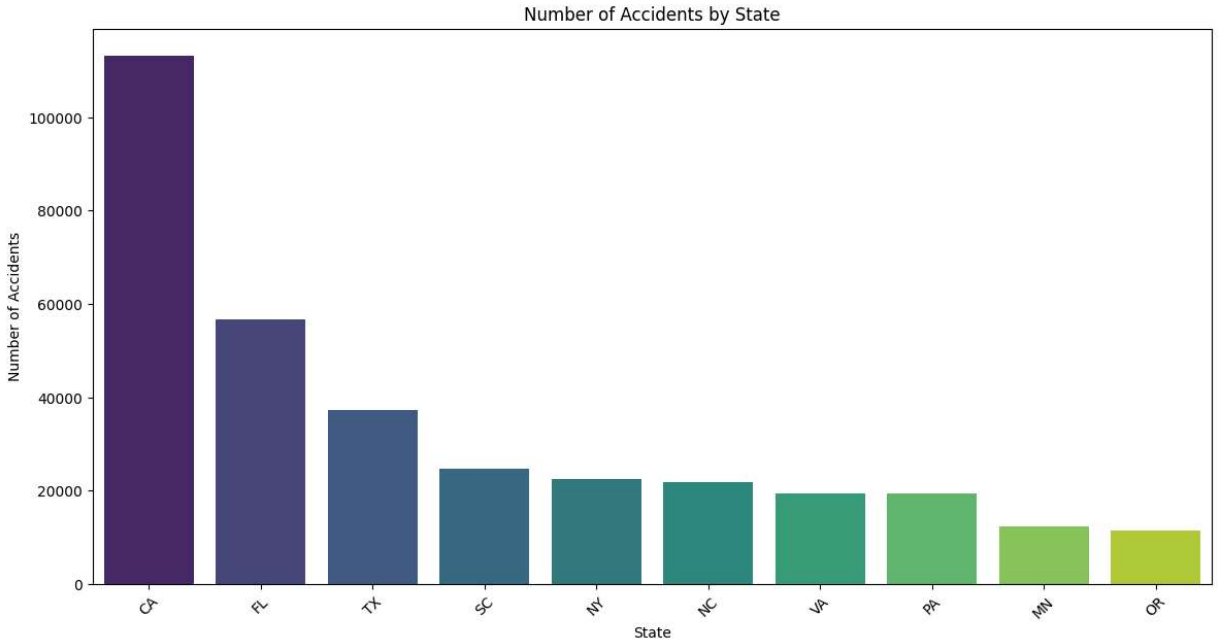


In [60]:
```
import matplotlib.pyplot as plt
import seaborn as sns
plt.figure(figsize=(12, 6))
```

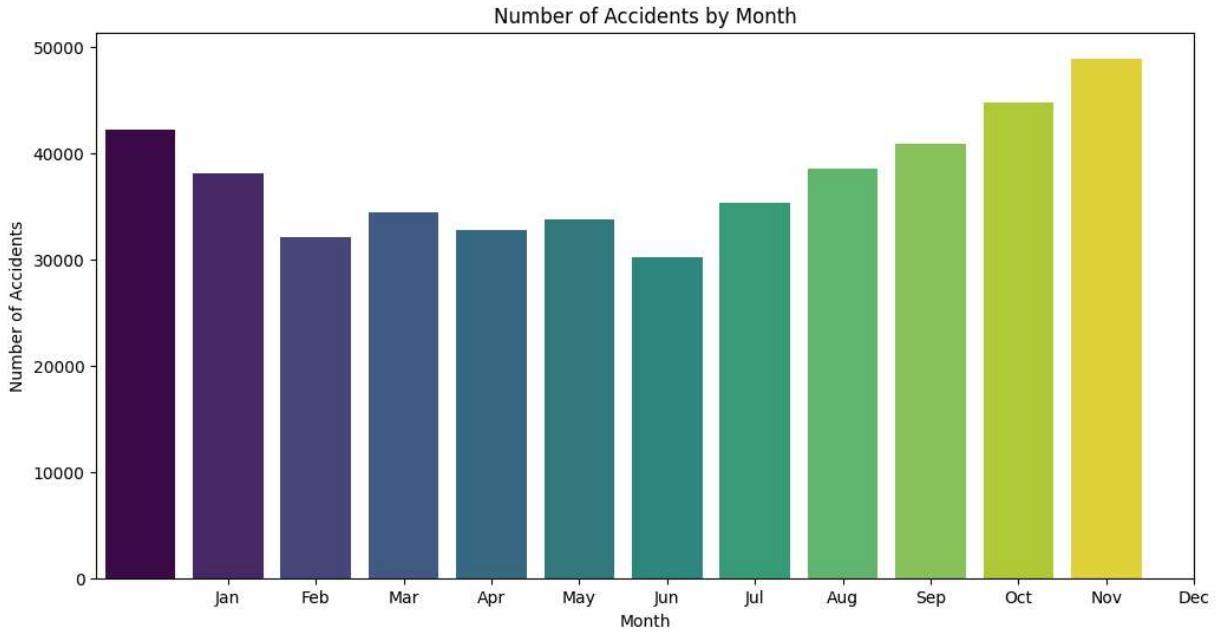
```
sns.countplot(x='Severity', data=df, hue='Severity', palette='viridis', legend=Fals
plt.title('Number of Accidents by Severity')
plt.xlabel('Severity')
plt.ylabel('Number of Accidents')
plt.show()
```



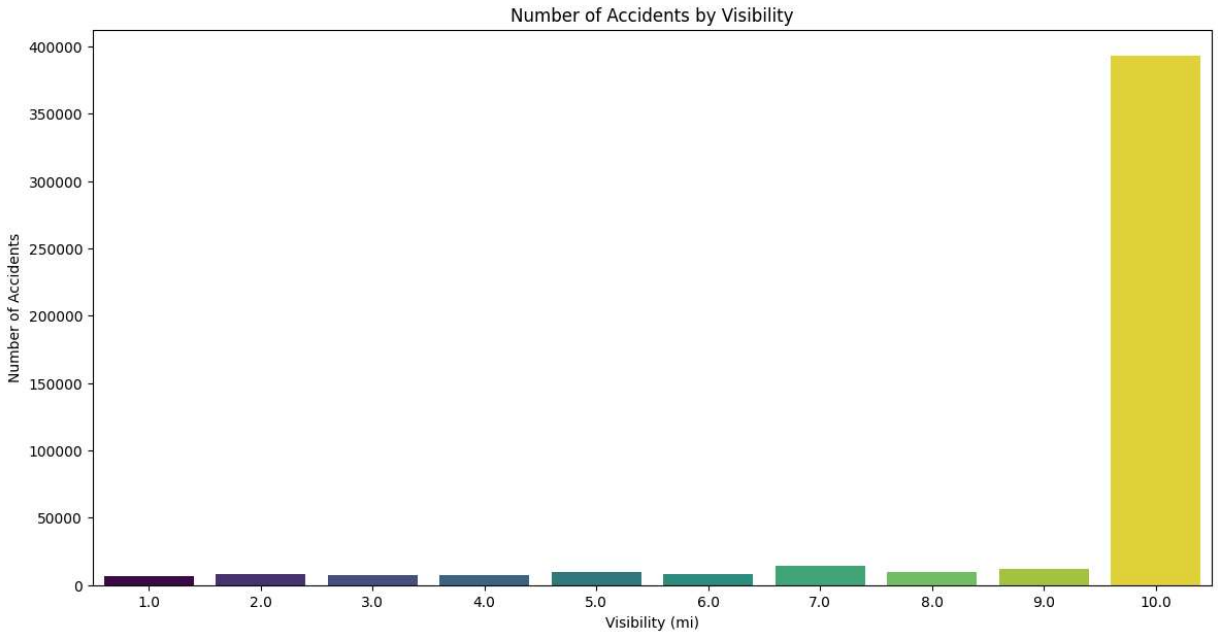
In [61]:
```
import matplotlib.pyplot as plt
import seaborn as sns
plt.figure(figsize=(14, 7))
state_counts = df['State'].value_counts().head(10)  # Show top 10 states
sns.barplot(x=state_counts.index, y=state_counts.values, hue=state_counts.index, pa
plt.title('Number of Accidents by State')
plt.xlabel('State')
plt.ylabel('Number of Accidents')
plt.xticks(rotation=45)
plt.show()
```



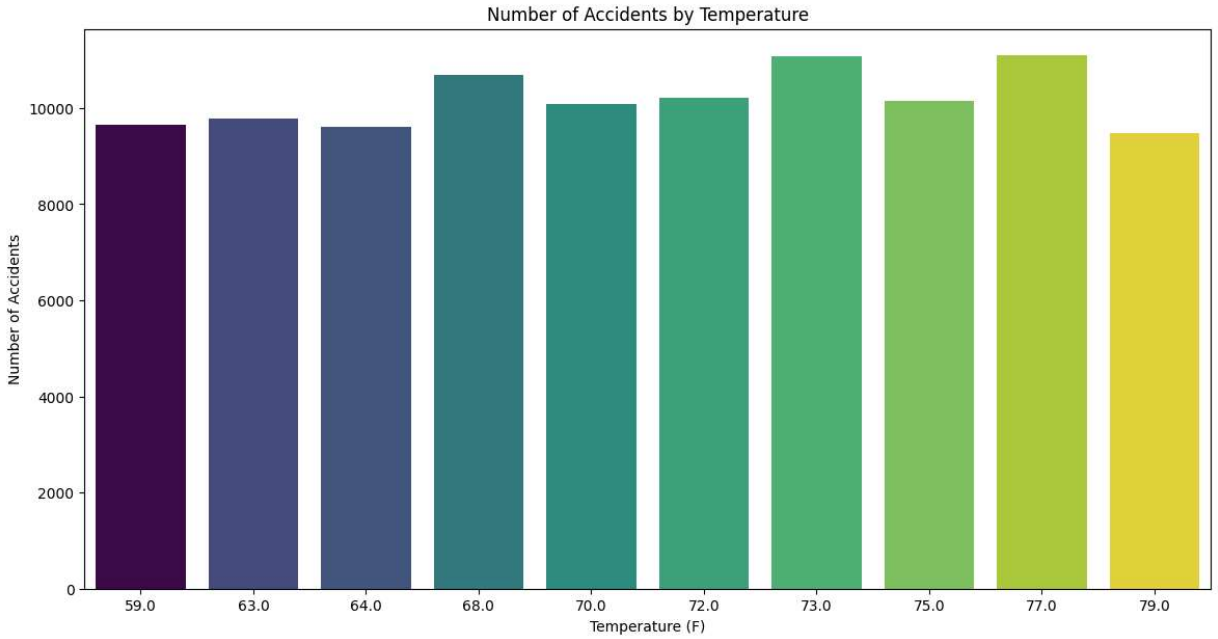
In [62]:
```
import matplotlib.pyplot as plt
import seaborn as sns
df['Month'] = df['Start_Time'].dt.month
plt.figure(figsize=(12, 6))
sns.countplot(x='Month', data=df, palette='viridis', hue='Month', legend=False)
plt.title('Number of Accidents by Month')
plt.xlabel('Month')
plt.ylabel('Number of Accidents')
plt.xticks(ticks=range(1, 13), labels=['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'J
plt.show()
```



In [63]:
```
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(14, 7))
visibility_counts = df['Visibility(mi)'].value_counts().head(10)
sns.barplot(x=visibility_counts.index, y=visibility_counts.values, palette='viridis
plt.title('Number of Accidents by Visibility')
plt.xlabel('Visibility (mi)')
plt.ylabel('Number of Accidents')
plt.show()
```

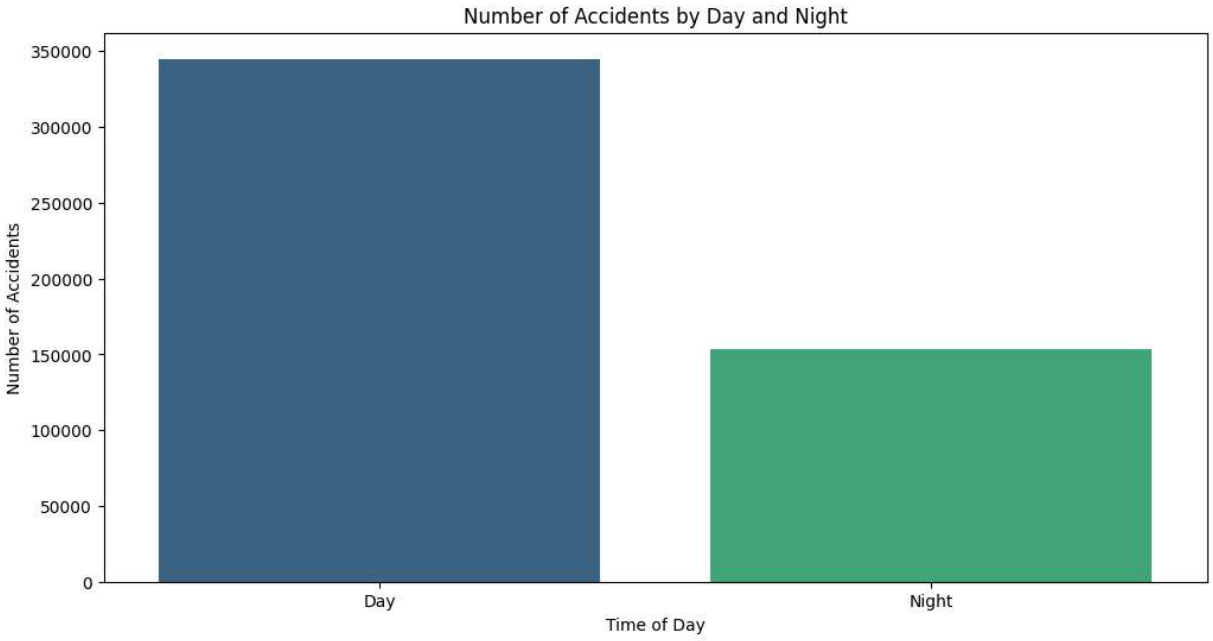


In [64]:
```
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(14, 7))
temperature_counts = df['Temperature(F)'].value_counts().head(10)
sns.barplot(x=temperature_counts.index, y=temperature_counts.values, palette='virid
plt.title('Number of Accidents by Temperature')
plt.xlabel('Temperature (F)')
plt.ylabel('Number of Accidents')
plt.show()
```



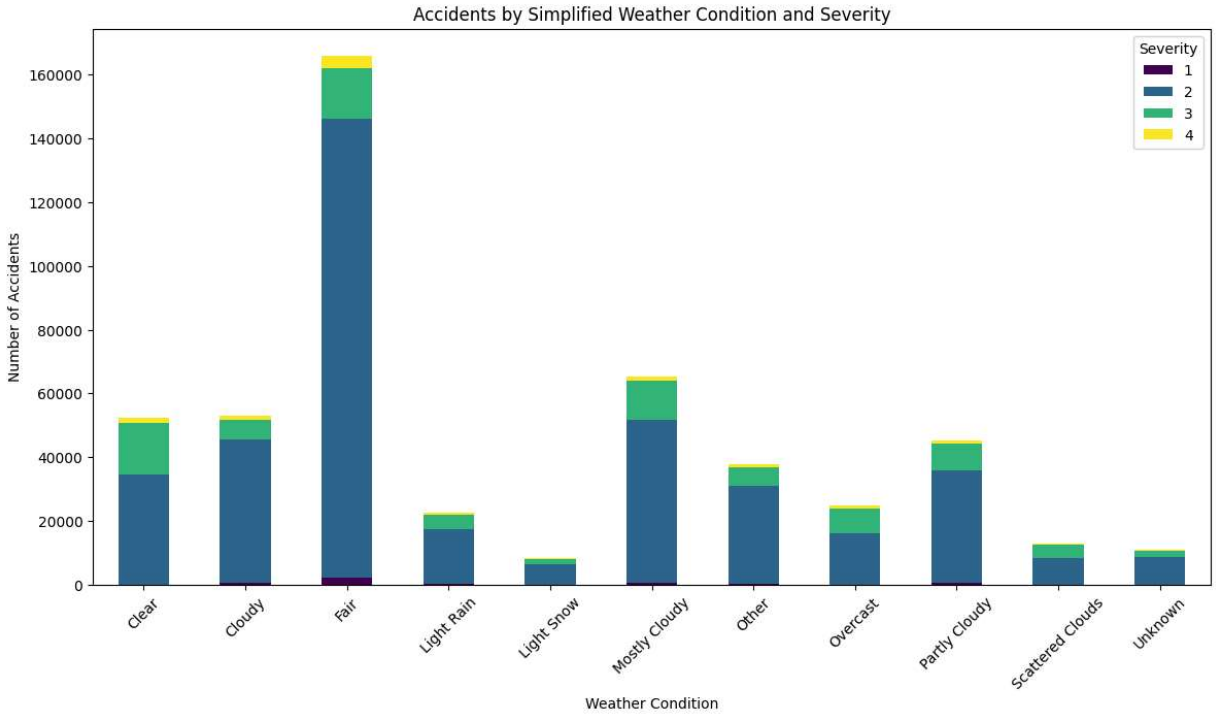
In [30]:
```
import matplotlib.pyplot as plt
import seaborn as sns
plt.figure(figsize=(12, 6))
sns.countplot(x='Sunrise_Sunset', hue='Sunrise_Sunset', data=df, palette='viridis',
plt.title('Number of Accidents by Day and Night')
```

```
plt.xlabel('Time of Day')
plt.ylabel('Number of Accidents')
plt.show()
```

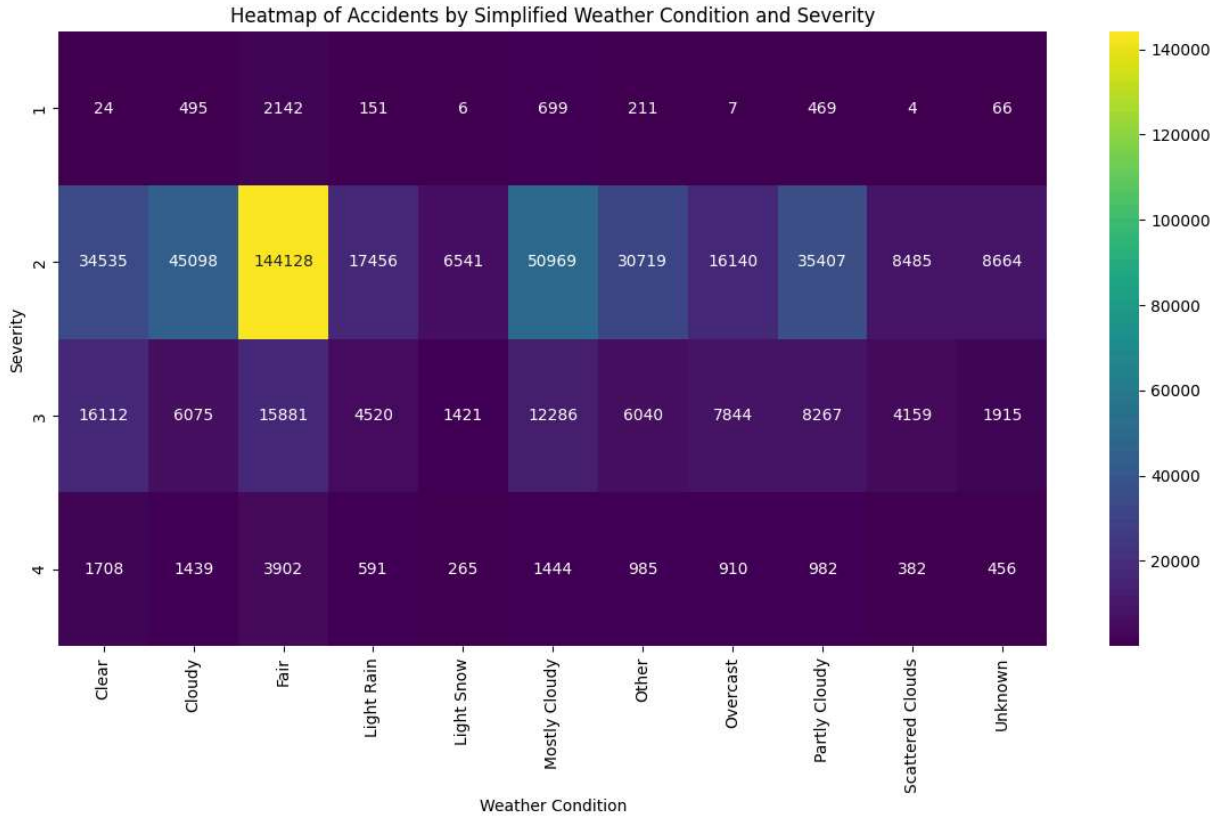


In [28]:
```
import matplotlib.pyplot as plt
import seaborn as sns
top_weather_conditions = df['Weather_Condition'].value_counts().nlargest(10).index
df['Weather_Condition_Simplified'] = df['Weather_Condition'].apply(lambda x: x if x
plt.figure(figsize=(14, 7))
weather_severity_simplified = df.groupby(['Weather_Condition_Simplified', 'Severity
weather_severity_simplified.plot(kind='bar', stacked=True, figsize=(14, 7), colorma
plt.title('Accidents by Simplified Weather Condition and Severity')
plt.xlabel('Weather Condition')
plt.ylabel('Number of Accidents')
plt.xticks(rotation=45)
plt.legend(title='Severity')
plt.show()
```

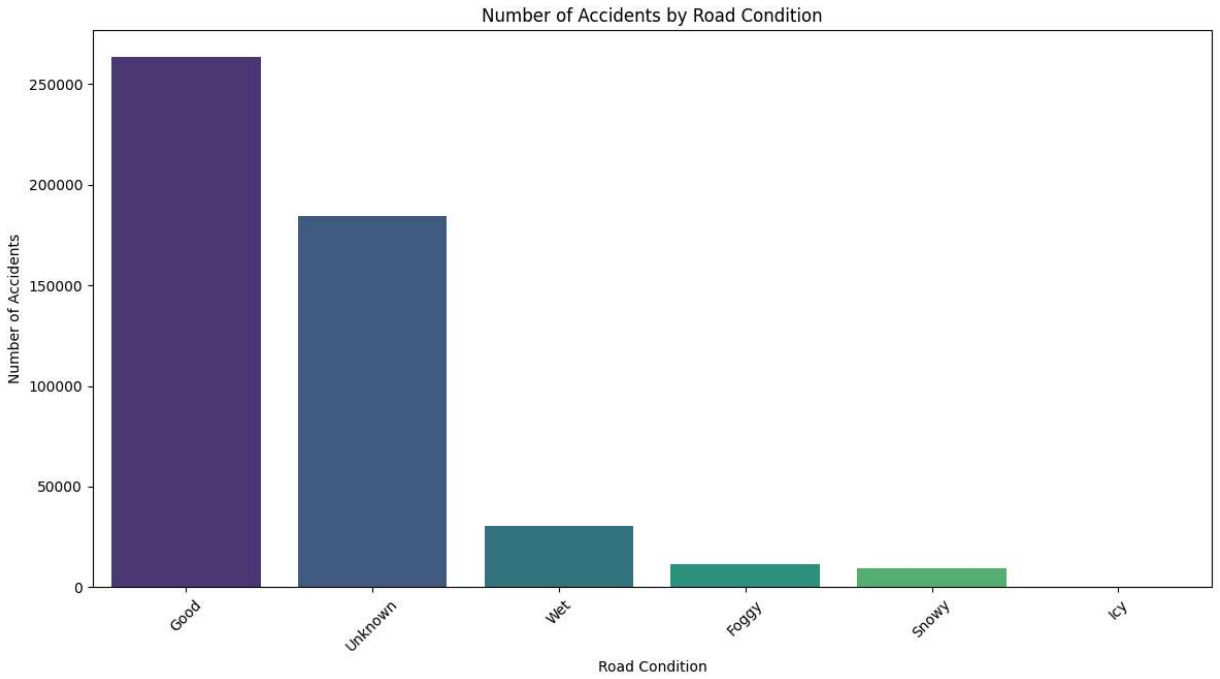
```
<Figure size 1400x700 with 0 Axes>
```



In [48]:
```
import seaborn as sns
plt.figure(figsize=(14, 7))
sns.heatmap(weather_severity_simplified.T, annot=True, cmap='viridis', fmt='g')
plt.title('Heatmap of Accidents by Simplified Weather Condition and Severity')
plt.xlabel('Weather Condition')
plt.ylabel('Severity')
plt.show()
```

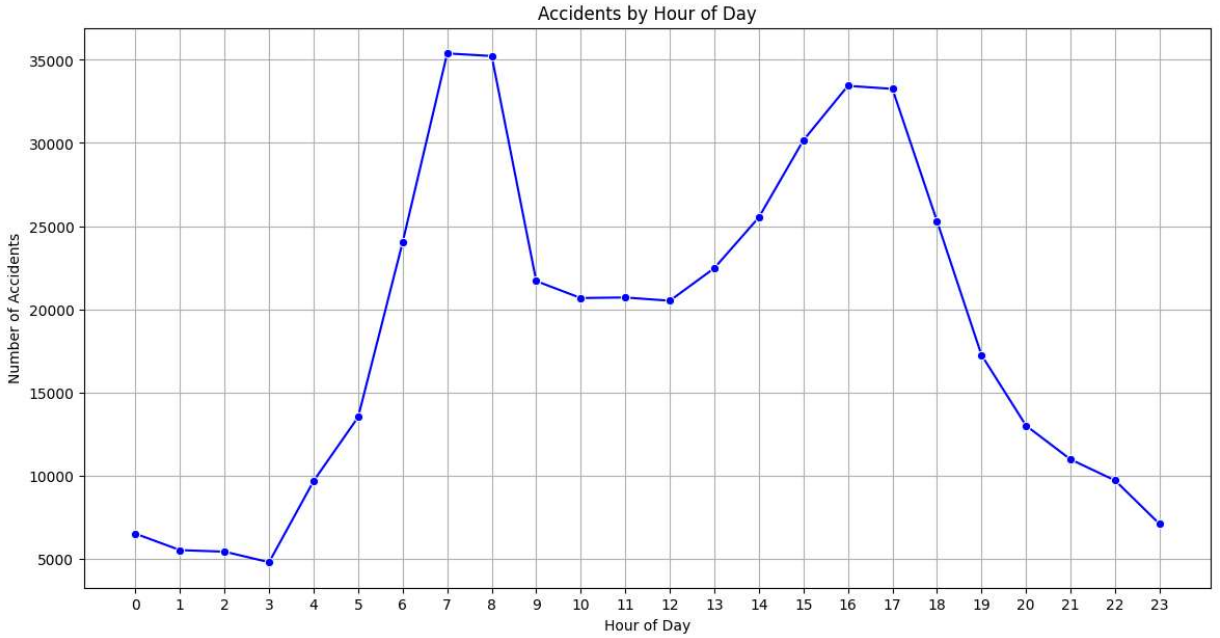


In [56]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
data_filepath = ('D:/prodigy/task5 accient/US_Accidents_March23_sampled_500k.csv')
df = pd.read_csv(data_filepath)
df['Start_Time'] = pd.to_datetime(df['Start_Time'], errors='coerce')
df['End_Time'] = pd.to_datetime(df['End_Time'], errors='coerce')
def categorize_road_condition(weather):
    if weather in ['Clear', 'Fair', 'Partly Cloudy']:
        return 'Good'
    elif weather in ['Rain', 'Heavy Rain', 'Light Rain', 'Showers', 'Thunderstorm']
        return 'Wet'
    elif weather in ['Snow', 'Heavy Snow', 'Light Snow', 'Sleet']:
        return 'Snowy'
    elif weather in ['Fog', 'Mist', 'Haze']:
        return 'Foggy'
    elif weather in ['Ice', 'Hail', 'Glaze']:
        return 'Icy'
    else:
        return 'Unknown'

df['Road_Condition'] = df['Weather_Condition'].apply(categorize_road_condition)
plt.figure(figsize=(14, 7))
road_conditions = df['Road_Condition'].value_counts()
sns.barplot(x=road_conditions.index, y=road_conditions.values, hue=road_conditions.
plt.title('Number of Accidents by Road Condition')
plt.xlabel('Road Condition')
plt.ylabel('Number of Accidents')
plt.xticks(rotation=45)
plt.show()
```

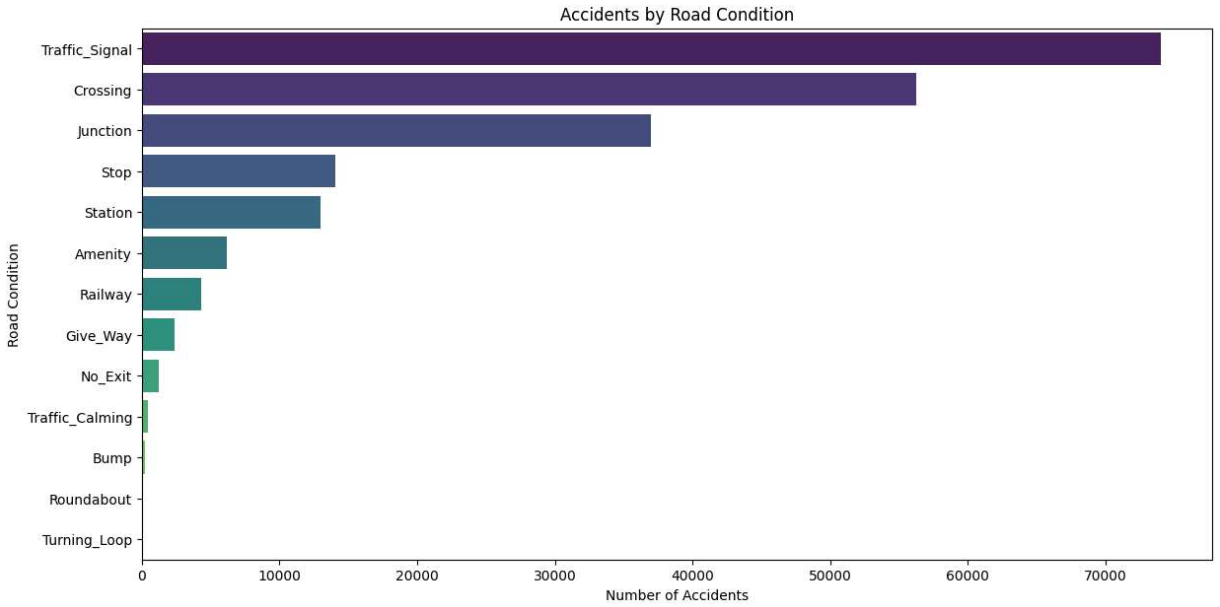


In [50]:
```
df['Hour'] = df['Start_Time'].dt.hour
plt.figure(figsize=(14, 7))
hourly_accidents = df.groupby('Hour').size()
```

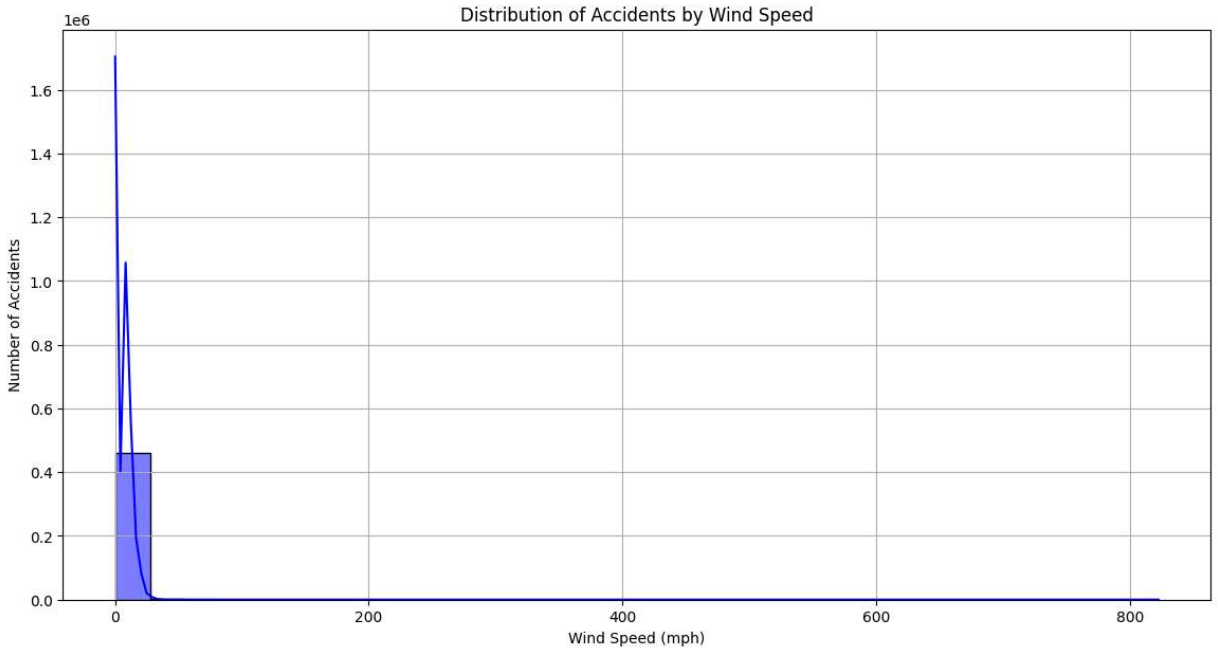
```
sns.lineplot(x=hourly_accidents.index, y=hourly_accidents.values, marker='o', color
plt.title('Accidents by Hour of Day')
plt.xlabel('Hour of Day')
plt.ylabel('Number of Accidents')
plt.grid(True)
plt.xticks(range(0, 24))
plt.show()
```



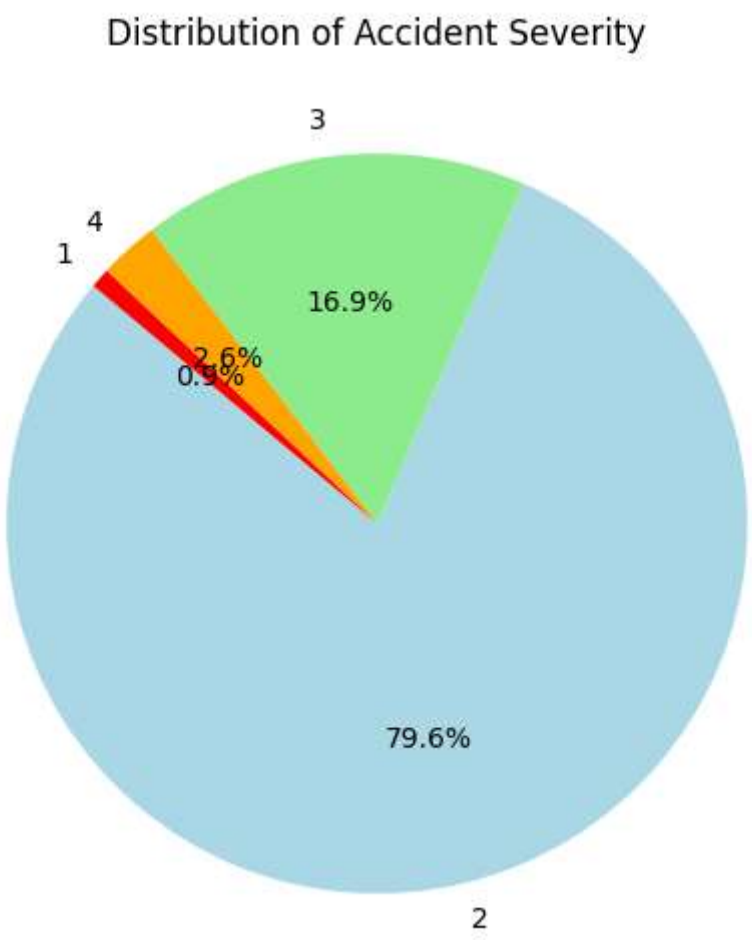
In [55]:
```
plt.figure(figsize=(14, 7))
road_conditions = ['Amenity', 'Bump', 'Crossing', 'Give_Way', 'Junction', 'No_Exit'
road_counts = df[road_conditions].sum().sort_values(ascending=False)

sns.barplot(x=road_counts.values, y=road_counts.index, hue=road_counts.index, palet
plt.title('Accidents by Road Condition')
plt.xlabel('Number of Accidents')
plt.ylabel('Road Condition')
plt.show()
```



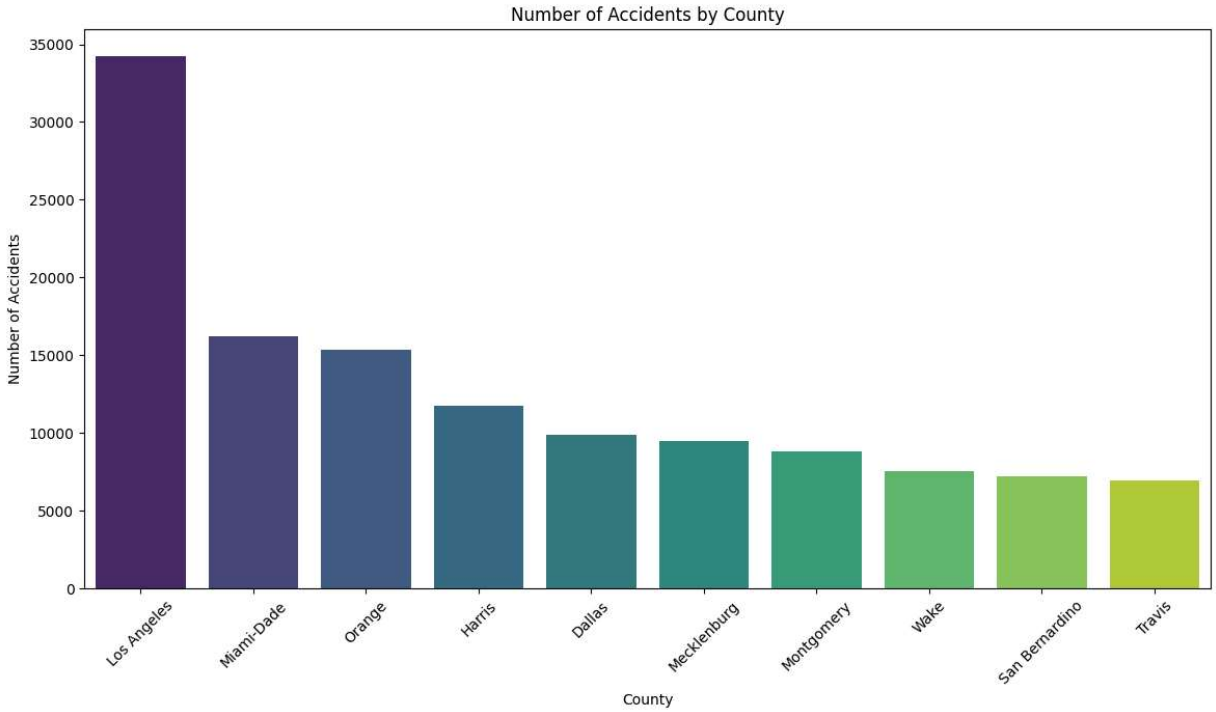
In [52]:
```
plt.figure(figsize=(14, 7))
sns.histplot(data=df, x='Wind_Speed(mph)', bins=30, kde=True, color='b', stat='coun
plt.title('Distribution of Accidents by Wind Speed')
plt.xlabel('Wind Speed (mph)')
plt.ylabel('Number of Accidents')
plt.grid(True)
plt.show()
```



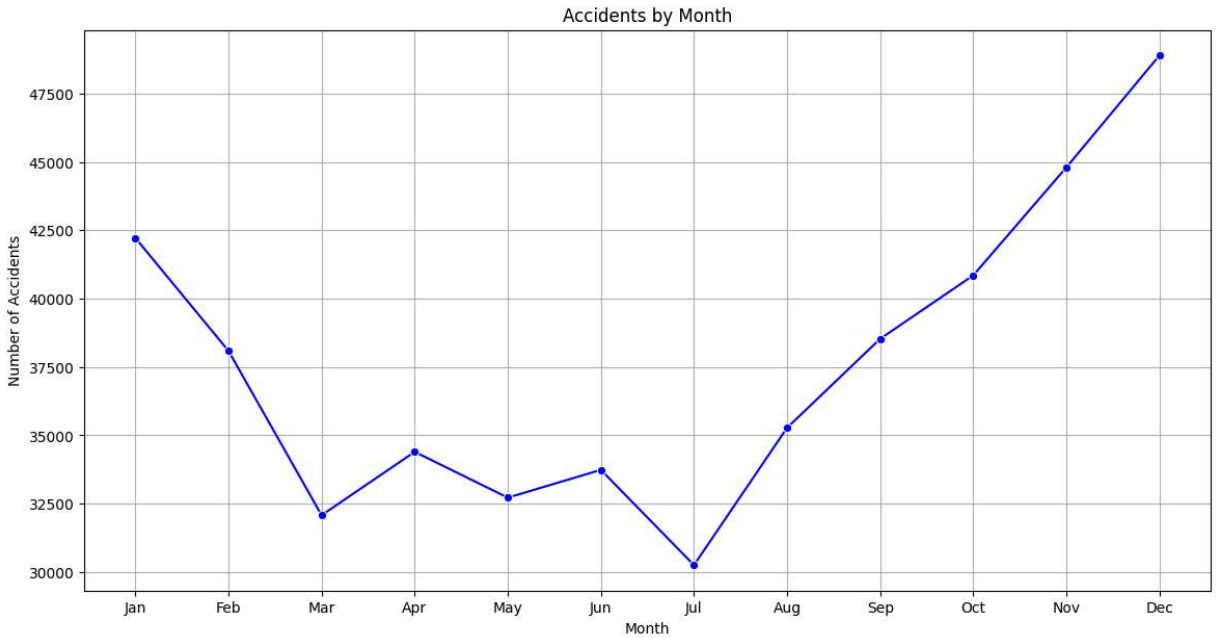
In [53]:
```
plt.figure(figsize=(8, 6))
severity_counts = df['Severity'].value_counts()
severity_counts.plot(kind='pie', autopct='%1.1f%%', startangle=140, colors=['lightb
plt.title('Distribution of Accident Severity')
plt.ylabel('')
plt.show()
```



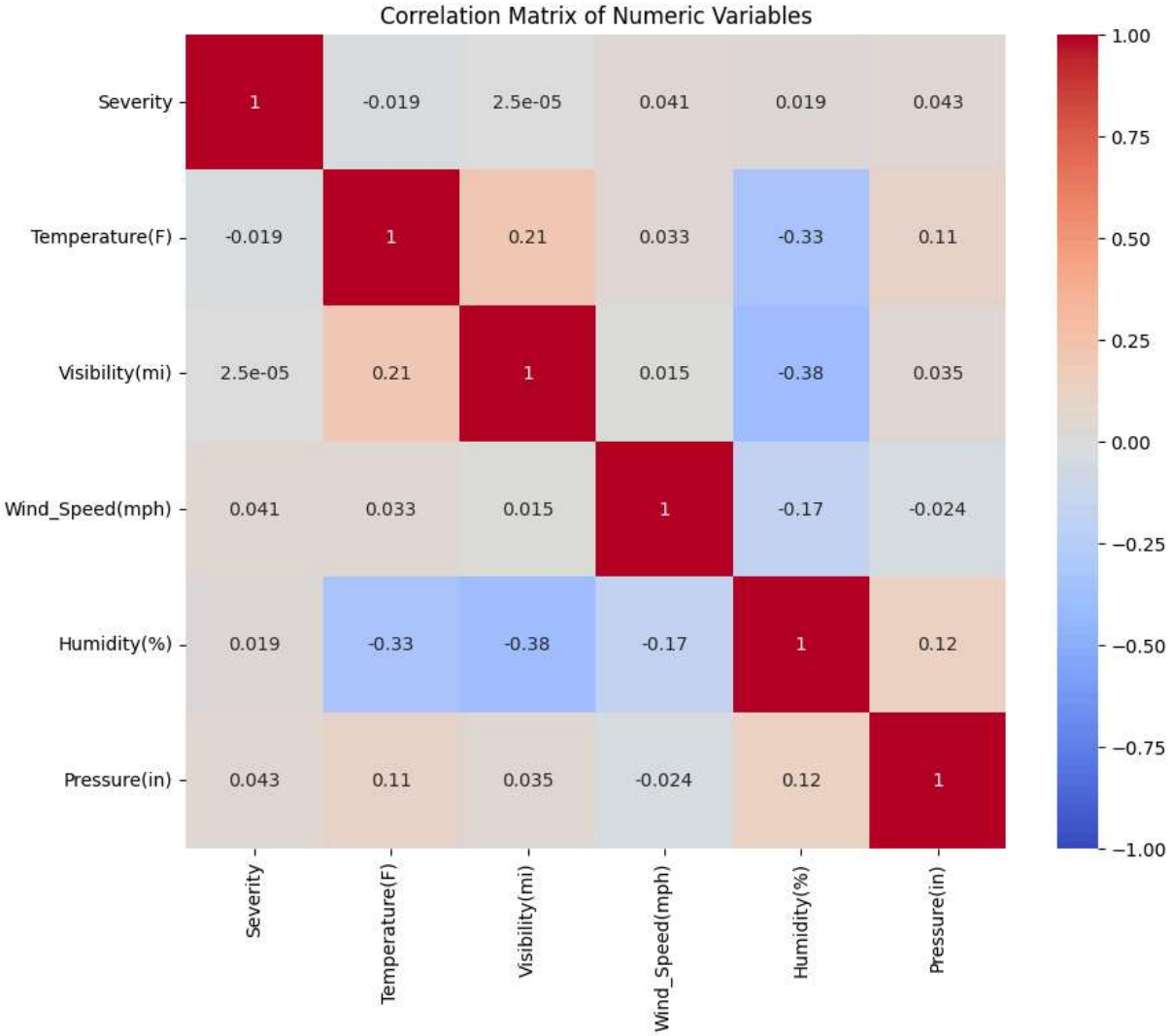
In [65]:
```
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(14, 7))
county_counts = df['County'].value_counts().head(10)
sns.barplot(x=county_counts.index, y=county_counts.values, palette='viridis', hue=c
plt.title('Number of Accidents by County')
plt.xlabel('County')
plt.ylabel('Number of Accidents')
plt.xticks(rotation=45)
plt.show()
```



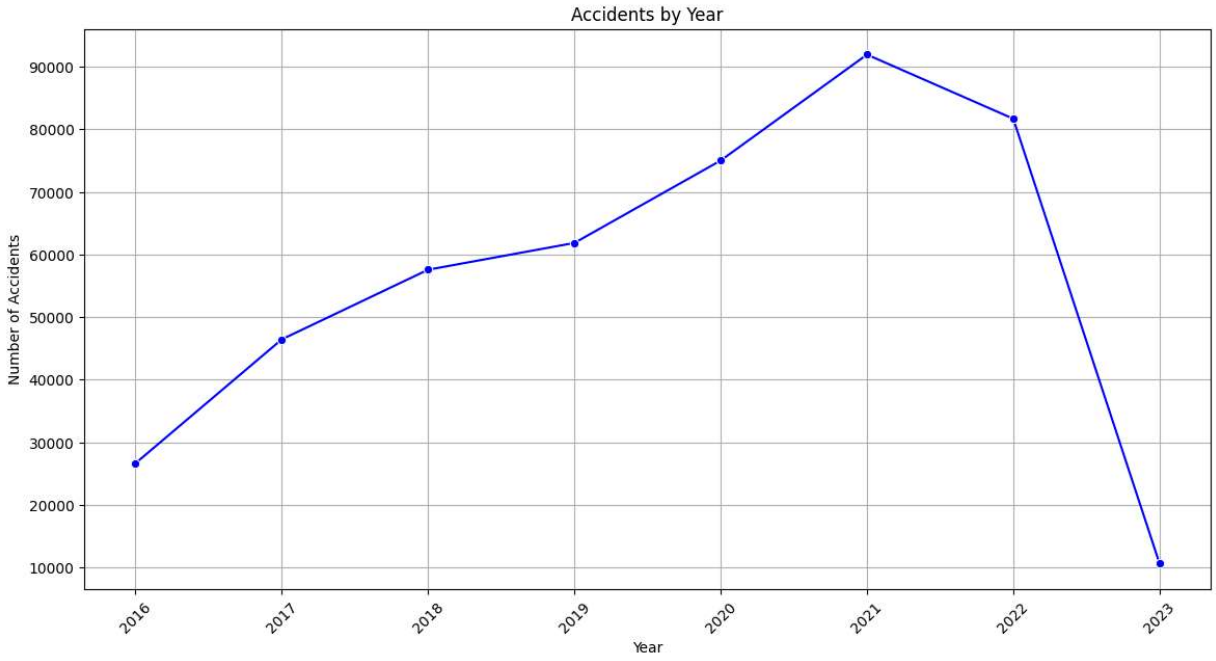
In [66]:
```
df['Month'] = df['Start_Time'].dt.month


plt.figure(figsize=(14, 7))
monthly_accidents = df.groupby('Month').size()
sns.lineplot(x=monthly_accidents.index, y=monthly_accidents.values, marker='o', col
plt.title('Accidents by Month')
plt.xlabel('Month')
plt.ylabel('Number of Accidents')
plt.grid(True)
plt.xticks(range(1, 13), ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', '
plt.show()
```



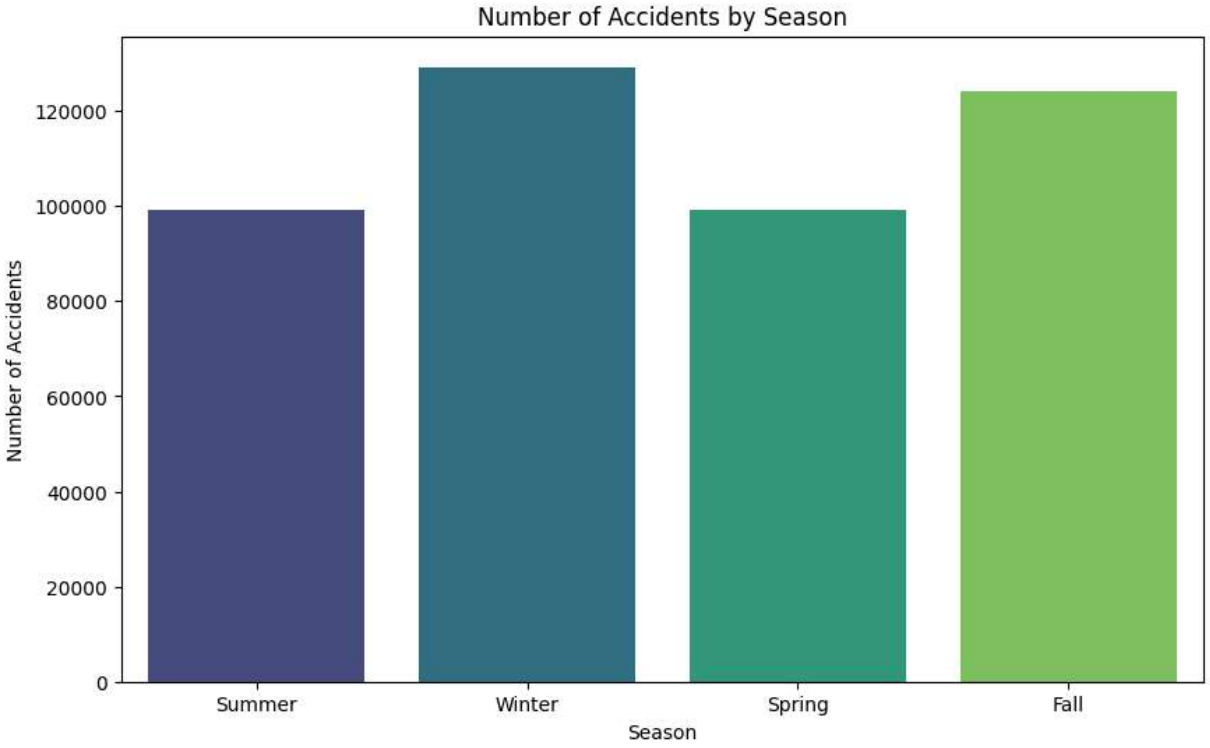
In [68]:
```
numeric_cols = ['Severity', 'Temperature(F)', 'Visibility(mi)', 'Wind_Speed(mph)',
corr_matrix = df[numeric_cols].corr()
plt.figure(figsize=(10, 8))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', vmin=-1, vmax=1)
plt.title('Correlation Matrix of Numeric Variables')
plt.show()
```



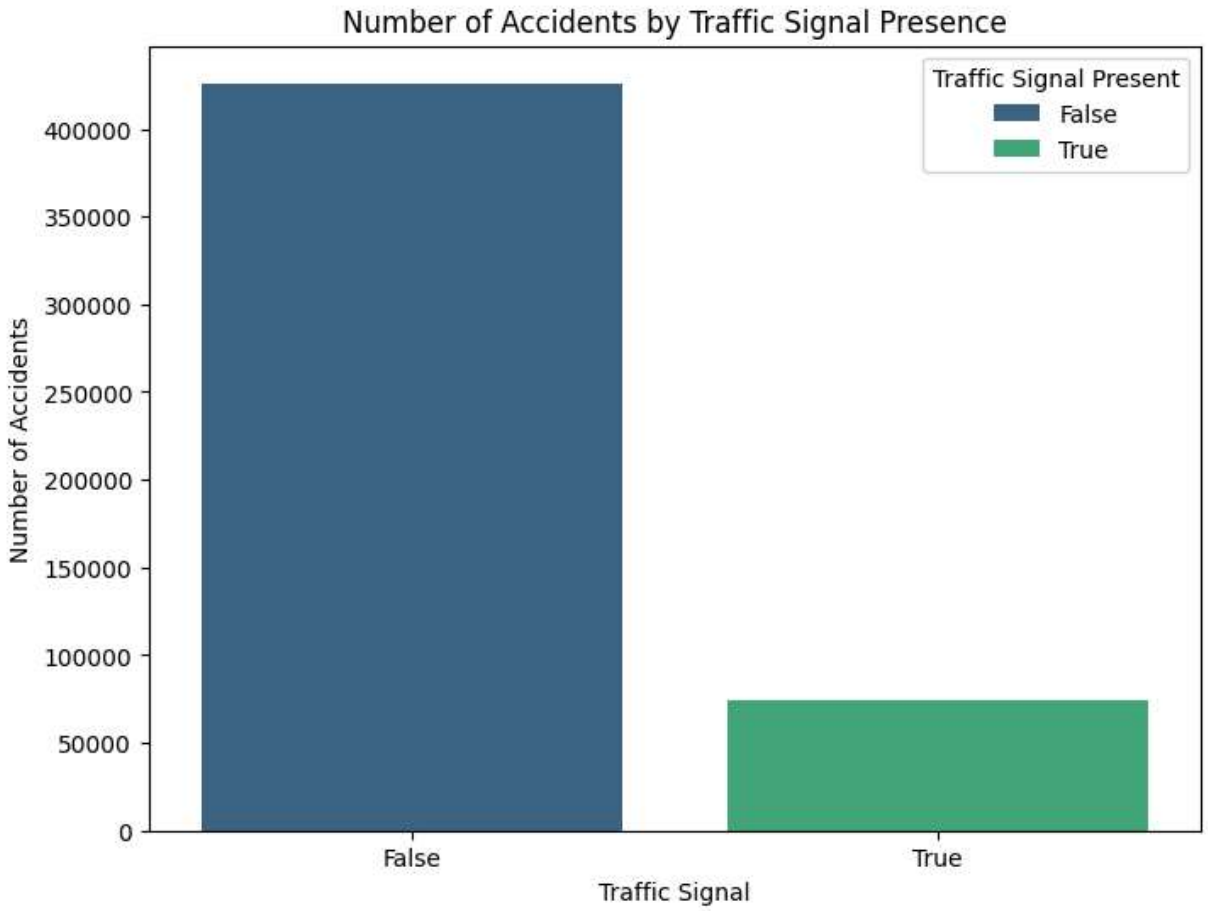
In [70]:
```
df['Year'] = df['Start_Time'].dt.year
plt.figure(figsize=(14, 7))
yearly_accidents = df.groupby('Year').size()
sns.lineplot(x=yearly_accidents.index, y=yearly_accidents.values, marker='o', color
plt.title('Accidents by Year')
plt.xlabel('Year')
plt.ylabel('Number of Accidents')
plt.grid(True)
plt.xticks(rotation=45)
plt.show()
```



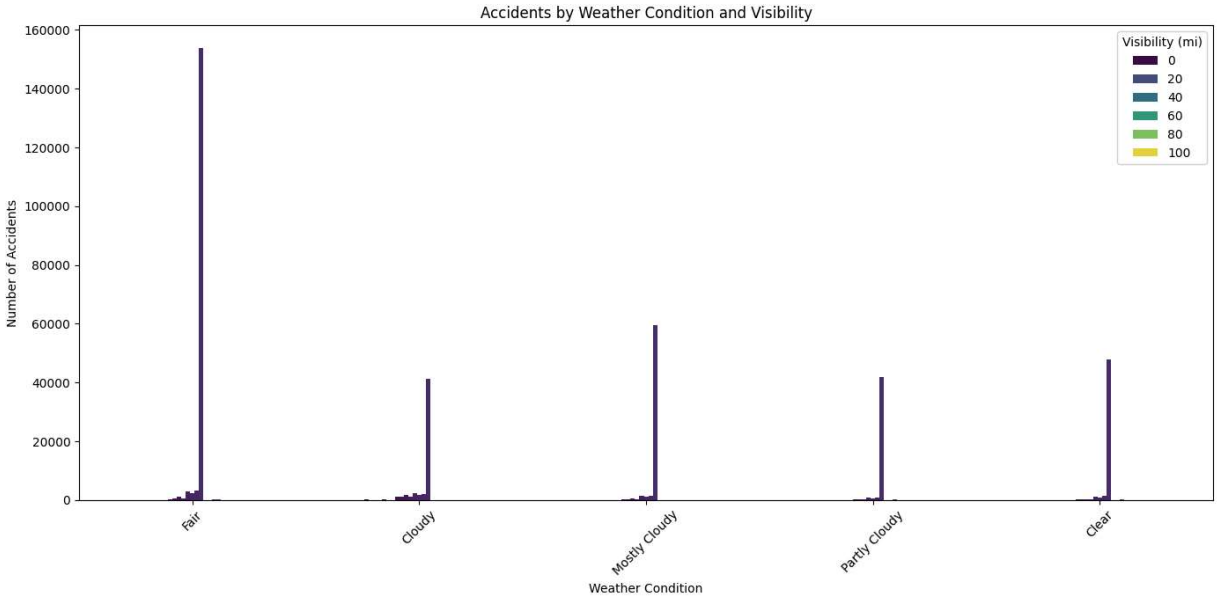
In [3]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
data_filepath = 'D:/prodigy/task5 accient/US_Accidents_March23_sampled_500k.csv'
df = pd.read_csv(data_filepath)
df['Start_Time'] = pd.to_datetime(df['Start_Time'], errors='coerce')
df['End_Time'] = pd.to_datetime(df['End_Time'], errors='coerce')
df['Season'] = df['Start_Time'].dt.month.map({1: 'Winter', 2: 'Winter', 3: 'Spring'
                                              6: 'Summer', 7: 'Summer', 8: 'Summer',
                                              11: 'Fall', 12: 'Winter'})
plt.figure(figsize=(10, 6))
sns.countplot(x='Season', data=df, palette='viridis', hue='Season', dodge=False, le
plt.title('Number of Accidents by Season')
plt.xlabel('Season')
plt.ylabel('Number of Accidents')
plt.show()
```



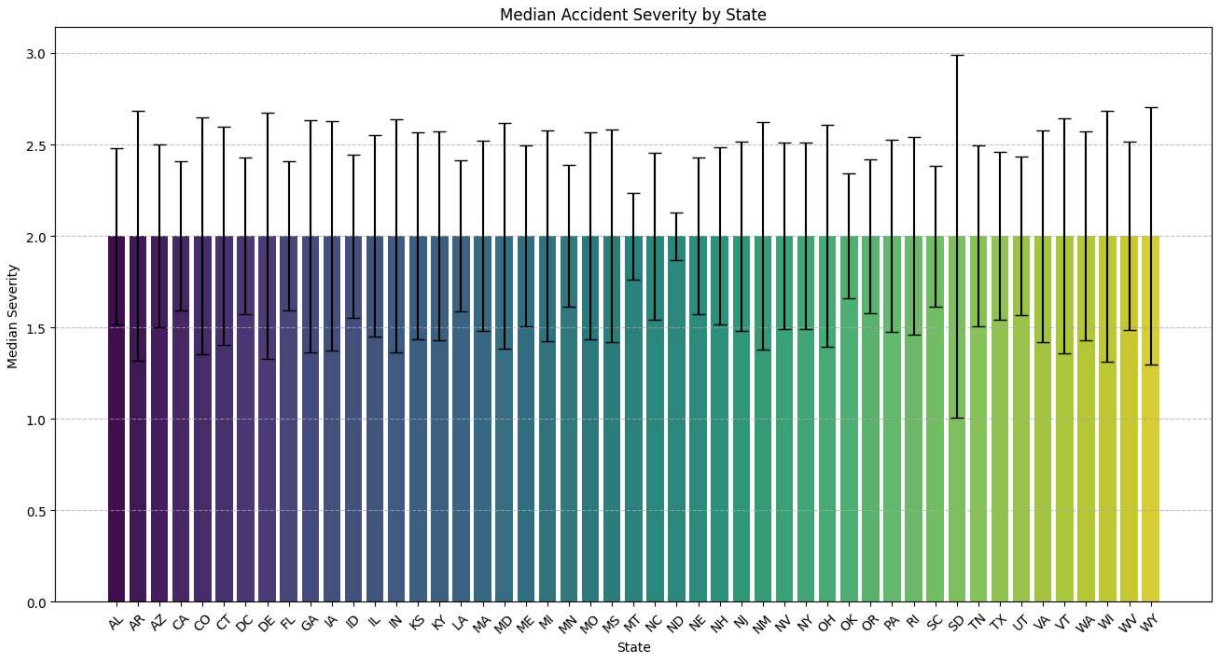
In [4]:
```
plt.figure(figsize=(8, 6))
traffic_signal_counts = df['Traffic_Signal'].value_counts()
sns.barplot(x=traffic_signal_counts.index, y=traffic_signal_counts.values, palette=
plt.title('Number of Accidents by Traffic Signal Presence')
plt.xlabel('Traffic Signal')
plt.ylabel('Number of Accidents')
plt.xticks(rotation=0)
plt.legend(title='Traffic Signal Present')
plt.show()
```



In [6]:
```
import matplotlib.pyplot as plt
import seaborn as sns
top_weather_conditions = df['Weather_Condition'].value_counts().nlargest(5).index
df_filtered = df[df['Weather_Condition'].isin(top_weather_conditions)]
plt.figure(figsize=(14, 7))
sns.countplot(data=df_filtered, x='Weather_Condition', hue='Visibility(mi)', palett
plt.title('Accidents by Weather Condition and Visibility')
plt.xlabel('Weather Condition')
plt.ylabel('Number of Accidents')
plt.xticks(rotation=45)
plt.legend(title='Visibility (mi)', loc='upper right')
plt.tight_layout()
plt.show()
```

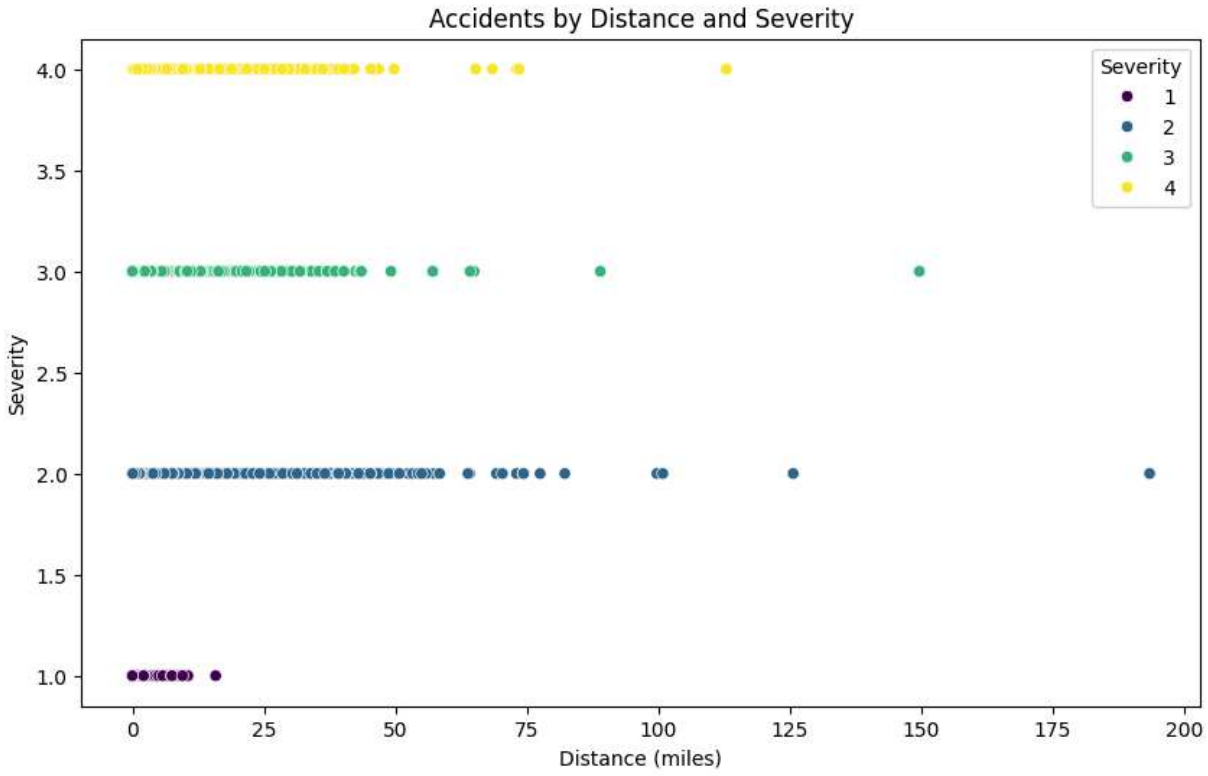


In [7]:
```
import matplotlib.pyplot as plt
import seaborn as sns
median_severity_by_state = df.groupby('State')['Severity'].median().sort_values(asc
plt.figure(figsize=(16, 8))
sns.barplot(x=median_severity_by_state.index, y=median_severity_by_state.values, hu
plt.errorbar(x=median_severity_by_state.index, y=median_severity_by_state.values,
             yerr=df.groupby('State')['Severity'].std().loc[median_severity_by_stat
             fmt='none', color='black', capsize=5)
plt.title('Median Accident Severity by State')
plt.xlabel('State')
plt.ylabel('Median Severity')
plt.xticks(rotation=45)
plt.grid(axis='y', linestyle='--', alpha=0.7)  # Add gridlines for better readabili
plt.show()
```

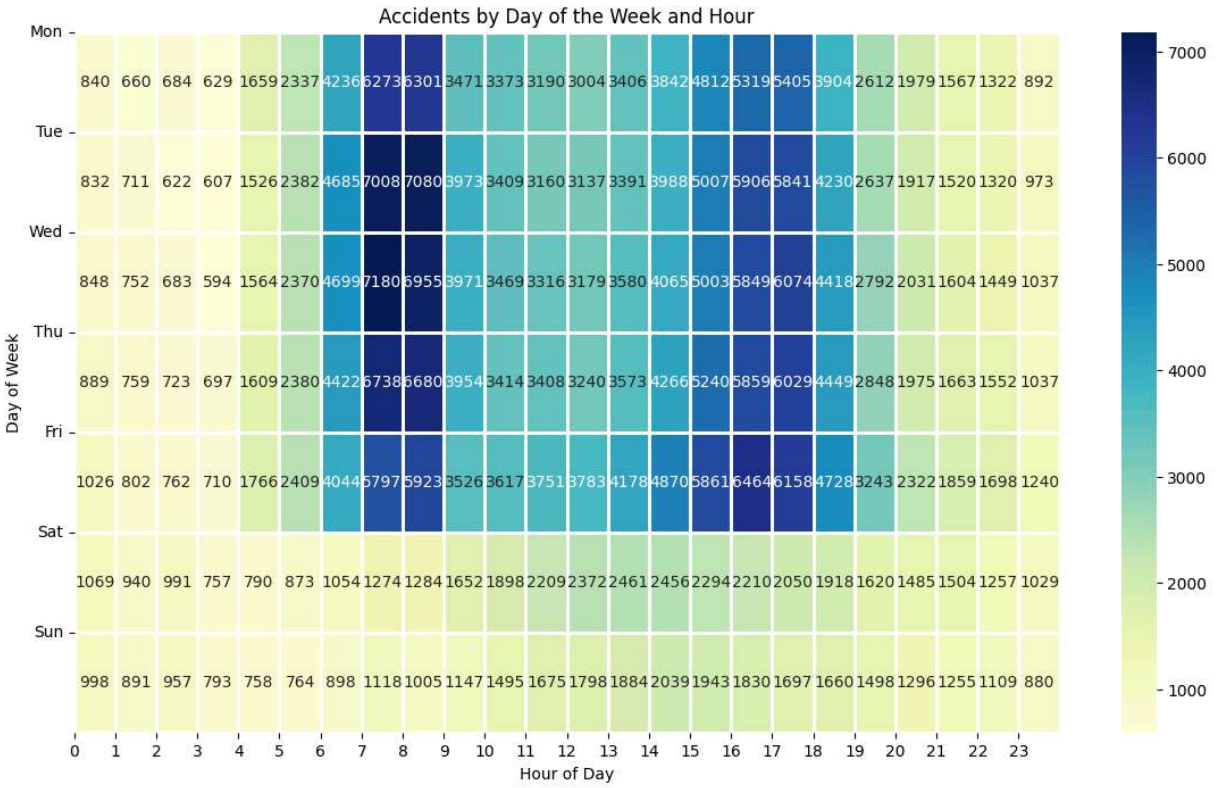


In [8]:
```
plt.figure(figsize=(10, 6))
sns.scatterplot(x='Distance(mi)', y='Severity', data=df, hue='Severity', palette='v
```

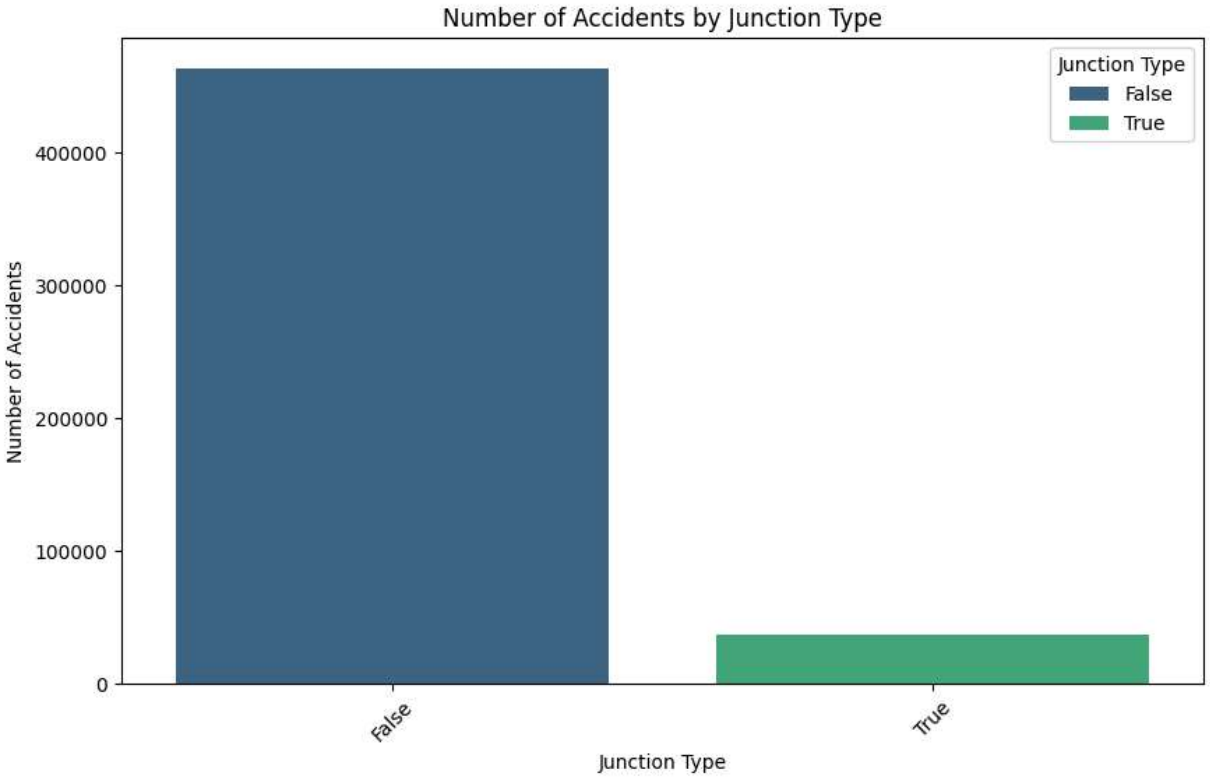
```
plt.title('Accidents by Distance and Severity')
plt.xlabel('Distance (miles)')
plt.ylabel('Severity')
plt.legend(title='Severity')
plt.show()
```



In [9]:
```
import matplotlib.pyplot as plt
import seaborn as sns
df['DayOfWeek'] = df['Start_Time'].dt.dayofweek
df['HourOfDay'] = df['Start_Time'].dt.hour
plt.figure(figsize=(14, 8))
sns.heatmap(df.groupby(['DayOfWeek', 'HourOfDay']).size().unstack(), cmap='YlGnBu',
plt.title('Accidents by Day of the Week and Hour')
plt.xlabel('Hour of Day')
plt.ylabel('Day of Week')
plt.xticks(range(24), labels=[str(i) for i in range(24)])
plt.yticks(range(7), labels=['Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat', 'Sun'], rota
plt.show()
```



In [10]:
```
plt.figure(figsize=(10, 6))
junction_counts = df['Junction'].value_counts()
sns.barplot(x=junction_counts.index, y=junction_counts.values, palette='viridis', h
plt.title('Number of Accidents by Junction Type')
plt.xlabel('Junction Type')
plt.ylabel('Number of Accidents')
plt.xticks(rotation=45)
plt.legend(title='Junction Type')
plt.show()
```


Number of Accidents by Junction Type
Junction Type
False
True
400000
300000
200000
100000
0
Number of Accidents
False
True
Junction Type

In [ ]: