Task5

July 13, 2024

0.1 PRODIGY INFO TECH DATA SCIENCE INTERN

0.1.1 PRAGADEESH G

0.1.2 TASK5:Analyze traffic accident data to identify patterns related to road conditions, weather, and time of day. Visualize accident hotspots and contributing factors.

ctors.

[3]:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import folium
from folium.plugins import HeatMap
```

[6]:

```
task5 = pd.read_csv("D:/prodigy info tech/US_Accidents.csv")
task5
```

[6]:

```
                ID  Severity        Start_Time          End_Time  Start_Lat  \
0        A-2716600         3  08-02-2016 00:37  08-02-2016 06:37  40.108910
1        A-2716601         2  08-02-2016 05:56  08-02-2016 11:56  39.865420
2        A-2716602         2  08-02-2016 06:15  08-02-2016 12:15  39.102660
3        A-2716603         2  08-02-2016 06:15  08-02-2016 12:15  39.101480
4        A-2716604         2  08-02-2016 06:51  08-02-2016 12:51  41.062130
…            …       …               …               …        …
1048570  A-3771908         2  10-12-2019 17:05  10-12-2019 17:58  33.195825
1048571  A-3771909         2  10-12-2019 17:02  10-12-2019 18:33  33.901813
1048572  A-3771910         2  10-12-2019 17:11  10-12-2019 18:24  33.651594
1048573  A-3771911         2  10-12-2019 17:04  10-12-2019 20:19  35.419703
1048574  A-3771912         2  10-12-2019 17:02  10-12-2019 18:04  33.806080

          Start_Lng    End_Lat     End_Lng  Distance(mi)  \
0        -83.092860  40.112060  -83.031870         3.230
1        -84.062800  39.865010  -84.048730         0.747
2        -84.524680  39.102090  -84.523960         0.055
3        -84.523410  39.098410  -84.522410         0.219
4        -81.537840  41.062170  -81.535470         0.123
…             …        …         …           …
```

```
1048570 -117.367005  33.195825 -117.367005         0.000
1048571 -117.466712  33.901813 -117.466712         0.000
1048572 -117.761153  33.651594 -117.761153         0.000
1048573 -119.012848  35.419703 -119.012848         0.000
1048574 -117.880100  33.806080 -117.880100         0.000

                                               Description  …  Roundabout  \
0        Between Sawmill Rd/Exit 20 and OH-315/Olentang… …       False
1                       At OH-4/OH-235/Exit 41 - Accident.  …       False
2                         At I-71/US-50/Exit 1 - Accident.  …       False
3                         At I-71/US-50/Exit 1 - Accident.  …       False
4                          At Dart Ave/Exit 21 - Accident.  …       False
…                                                    …  …         …
1048570              At Oceanside Blvd/Exit 52 - Accident.  …       False
1048571                       At La Sierra Ave - Accident.  …       False
1048572            At CA-133/Laguna Fwy/Exit 2 - Accident.  …       False
1048573                        At E Roberts Ln - Accident.  …       False
1048574                    At CA-57/Orange Fwy - Accident.  …       False

        Station   Stop Traffic_Calming Traffic_Signal Turning_Loop  \
0         False  False           False          False        False
1         False  False           False          False        False
2         False  False           False          False        False
3         False  False           False          False        False
4         False  False           False          False        False
…         …    …             …            …          …
1048570   False  False           False          False        False
1048571   False  False           False          False        False
1048572   False  False           False          False        False
1048573   False  False           False           True        False
1048574   False  False           False          False        False

        Sunrise_Sunset Civil_Twilight Nautical_Twilight Astronomical_Twilight
0                Night          Night             Night                 Night
1                Night          Night             Night                 Night
2                Night          Night             Night                   Day
3                Night          Night             Night                   Day
4                Night          Night               Day                   Day
…                …            …               …                   …
1048570          Night            Day               Day                   Day
1048571          Night            Day               Day                   Day
1048572          Night          Night               Day                   Day
1048573          Night            Day               Day                   Day
1048574          Night            Day               Day                   Day

[1048575 rows x 47 columns]
```

[7]:

```
task5.head()
```

[7]:

```
          ID  Severity        Start_Time          End_Time  Start_Lat  \
0  A-2716600         3  08-02-2016 00:37  08-02-2016 06:37   40.10891
1  A-2716601         2  08-02-2016 05:56  08-02-2016 11:56   39.86542
2  A-2716602         2  08-02-2016 06:15  08-02-2016 12:15   39.10266
3  A-2716603         2  08-02-2016 06:15  08-02-2016 12:15   39.10148
4  A-2716604         2  08-02-2016 06:51  08-02-2016 12:51   41.06213

   Start_Lng   End_Lat   End_Lng  Distance(mi)  \
0  -83.09286  40.11206 -83.03187         3.230
1  -84.06280  39.86501 -84.04873         0.747
2  -84.52468  39.10209 -84.52396         0.055
3  -84.52341  39.09841 -84.52241         0.219
4  -81.53784  41.06217 -81.53547         0.123

                                         Description  …  Roundabout Station  \
0  Between Sawmill Rd/Exit 20 and OH-315/Olentang…  …       False  False
1                 At OH-4/OH-235/Exit 41 - Accident.  …       False   False
2                   At I-71/US-50/Exit 1 - Accident.  …       False   False
3                   At I-71/US-50/Exit 1 - Accident.  …       False   False
4                    At Dart Ave/Exit 21 - Accident.  …       False   False

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

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

[5 rows x 47 columns]
```

[8]:

```
task5.tail()
```

[8]:

```
                ID  Severity        Start_Time          End_Time  Start_Lat  \
1048570  A-3771908         2  10-12-2019 17:05  10-12-2019 17:58  33.195825
1048571  A-3771909         2  10-12-2019 17:02  10-12-2019 18:33  33.901813
1048572  A-3771910         2  10-12-2019 17:11  10-12-2019 18:24  33.651594
1048573  A-3771911         2  10-12-2019 17:04  10-12-2019 20:19  35.419703
1048574  A-3771912         2  10-12-2019 17:02  10-12-2019 18:04  33.806080
```

```
          Start_Lng    End_Lat     End_Lng  Distance(mi)  \
1048570 -117.367005  33.195825 -117.367005           0.0
1048571 -117.466712  33.901813 -117.466712           0.0
1048572 -117.761153  33.651594 -117.761153           0.0
1048573 -119.012848  35.419703 -119.012848           0.0
1048574 -117.880100  33.806080 -117.880100           0.0

                                     Description  …  Roundabout Station  \
1048570    At Oceanside Blvd/Exit 52 - Accident.  …       False   False
1048571             At La Sierra Ave - Accident.  …       False   False
1048572  At CA-133/Laguna Fwy/Exit 2 - Accident.  …       False   False
1048573              At E Roberts Ln - Accident.  …       False   False
1048574          At CA-57/Orange Fwy - Accident.  …       False   False

          Stop Traffic_Calming Traffic_Signal Turning_Loop Sunrise_Sunset  \
1048570  False           False          False        False          Night
1048571  False           False          False        False          Night
1048572  False           False          False        False          Night
1048573  False           False           True        False          Night
1048574  False           False          False        False          Night

        Civil_Twilight Nautical_Twilight Astronomical_Twilight
1048570            Day               Day                   Day
1048571            Day               Day                   Day
1048572          Night               Day                   Day
1048573            Day               Day                   Day
1048574            Day               Day                   Day

[5 rows x 47 columns]
```

```
[10]: task5.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1048575 entries, 0 to 1048574
Data columns (total 47 columns):
 #   Column                 Non-Null Count    Dtype
---  ------                 --------------    -----
 0   ID                     1048575 non-null  object
 1   Severity               1048575 non-null  int64
 2   Start_Time             1048575 non-null  object
 3   End_Time               1048575 non-null  object
 4   Start_Lat              1048575 non-null  float64
 5   Start_Lng              1048575 non-null  float64
 6   End_Lat                1048575 non-null  float64
 7   End_Lng                1048575 non-null  float64
 8   Distance(mi)           1048575 non-null  float64
 9   Description            1048575 non-null  object
 10  Number                 360065 non-null   float64
 11  Street                 1048575 non-null  object
 12  Side                   1048575 non-null  object
 13  City                   1048506 non-null  object
 14  County                 1048575 non-null  object
 15  State                  1048575 non-null  object
 16  Zipcode                1048086 non-null  object
 17  Country                1048575 non-null  object
 18  Timezone               1047421 non-null  object
 19  Airport_Code           1045917 non-null  object
 20  Weather_Timestamp      1026460 non-null  object
 21  Temperature(F)         1017832 non-null  float64
 22  Wind_Chill(F)          813974 non-null   float64
 23  Humidity(%)            1015803 non-null  float64
 24  Pressure(in)           1023115 non-null  float64
 25  Visibility(mi)         1017760 non-null  float64
 26  Wind_Direction         1017532 non-null  object
 27  Wind_Speed(mph)        980665 non-null   float64
 28  Precipitation(in)      789334 non-null   float64
 29  Weather_Condition      1018315 non-null  object
 30  Amenity                1048575 non-null  bool
 31  Bump                   1048575 non-null  bool
 32  Crossing               1048575 non-null  bool
 33  Give_Way               1048575 non-null  bool
 34  Junction               1048575 non-null  bool
 35  No_Exit                1048575 non-null  bool
 36  Railway                1048575 non-null  bool
 37  Roundabout             1048575 non-null  bool
 38  Station                1048575 non-null  bool
 39  Stop                   1048575 non-null  bool
 40  Traffic_Calming        1048575 non-null  bool
 41  Traffic_Signal         1048575 non-null  bool
 42  Turning_Loop           1048575 non-null  bool
 43  Sunrise_Sunset         1048506 non-null  object
 44  Civil_Twilight         1048506 non-null  object
 45  Nautical_Twilight      1048506 non-null  object
 46  Astronomical_Twilight  1048506 non-null  object
dtypes: bool(13), float64(13), int64(1), object(20)
memory usage: 285.0+ MB
```

[11]:

```
task5.describe()
```

[11]:

```
           Severity     Start_Lat     Start_Lng       End_Lat       End_Lng  \
count  1.048575e+06  1.048575e+06  1.048575e+06  1.048575e+06  1.048575e+06
mean   2.164623e+00  3.646292e+01 -9.720202e+01  3.646305e+01 -9.720181e+01
std    5.460908e-01  5.165882e+00  1.831984e+01  5.165957e+00  1.831963e+01
min    1.000000e+00  2.457058e+01 -1.244975e+02  2.457433e+01 -1.244975e+02
```

```
25%    2.000000e+00  3.371034e+01 -1.180358e+02  3.371188e+01 -1.180361e+02
50%    2.000000e+00  3.635720e+01 -9.292586e+01  3.635665e+01 -9.292828e+01
75%    2.000000e+00  4.023489e+01 -8.038426e+01  4.023471e+01 -8.038469e+01
max    4.000000e+00  4.900058e+01 -6.711317e+01  4.907500e+01 -6.710924e+01

       Distance(mi)         Number  Temperature(F)  Wind_Chill(F)  \
count  1.048575e+06  360065.000000    1.017832e+06  813974.000000
mean   5.867531e-01    7869.769653    5.915797e+01      55.283185
std    1.601684e+00   15619.751306    1.778244e+01      20.048839
min    0.000000e+00       1.000000   -8.900000e+01     -89.000000
25%    0.000000e+00    1175.000000    4.700000e+01      42.000000
50%    1.380000e-01    3771.000000    6.000000e+01      57.000000
75%    5.910000e-01    9229.000000    7.300000e+01      70.000000
max    1.551860e+02  961005.000000    1.292000e+02     113.000000

        Humidity(%)  Pressure(in)  Visibility(mi)  Wind_Speed(mph)  \
count  1.015803e+06  1.023115e+06    1.017760e+06    980665.000000
mean   6.553987e+01  2.951557e+01    9.098803e+00         7.464395
std    2.298438e+01  9.907375e-01    2.718757e+00         5.810570
min    2.000000e+00  2.000000e-02    0.000000e+00         0.000000
25%    4.900000e+01  2.934000e+01    1.000000e+01         3.500000
50%    6.900000e+01  2.985000e+01    1.000000e+01         7.000000
75%    8.500000e+01  3.002000e+01    1.000000e+01        10.400000
max    1.000000e+02  5.804000e+01    1.200000e+02       984.000000

       Precipitation(in)
count      789334.000000
mean            0.007445
std             0.114932
min             0.000000
25%             0.000000
50%             0.000000
75%             0.000000
max            24.000000
```

[12]:

```
# Data Preprocessing
# Handling missing values
df = task.dropna(subset=['Start_Time', 'Severity', 'Weather_Condition',␣
 ↪'Start_Lat', 'Start_Lng'])
```

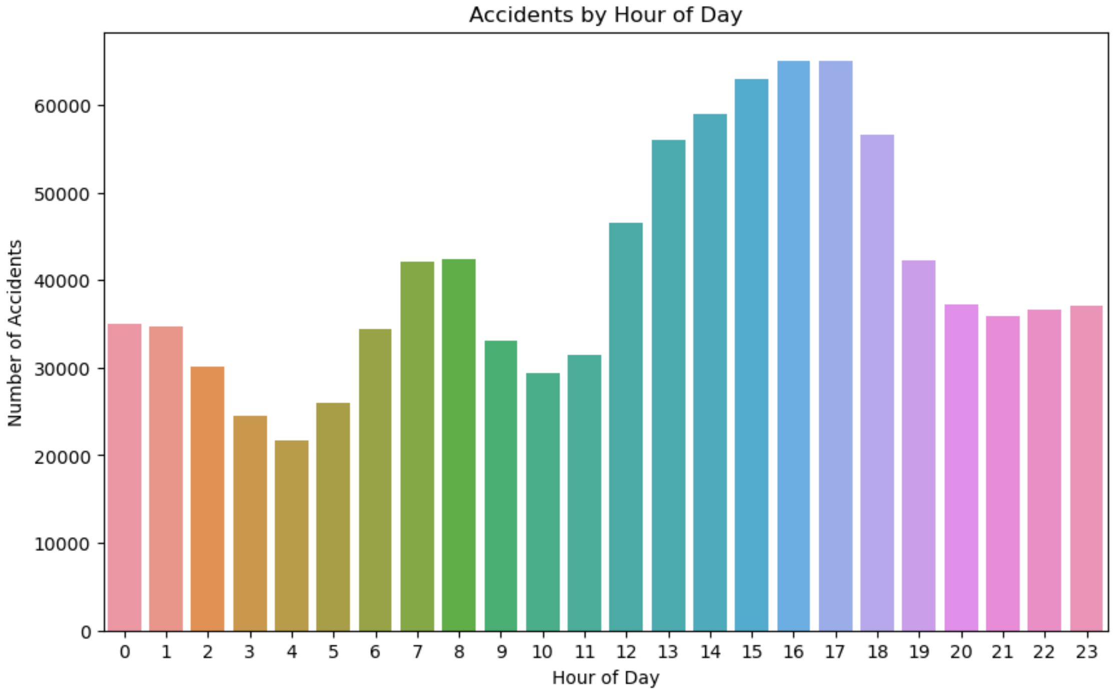
[16]:

```
# Extract relevant features
df.loc[:, 'Start_Time'] = pd.to_datetime(df['Start_Time'], format='%d-%m-%Y %H:
 ↪%M', errors='coerce')
df = df.dropna(subset=['Start_Time'])  # Drop rows where the date conversion␣
 ↪failed
df.loc[:, 'Hour'] = df['Start_Time'].dt.hour
df.loc[:, 'DayOfWeek'] = df['Start_Time'].dt.dayofweek
df.loc[:, 'Date'] = df['Start_Time'].dt.date
```

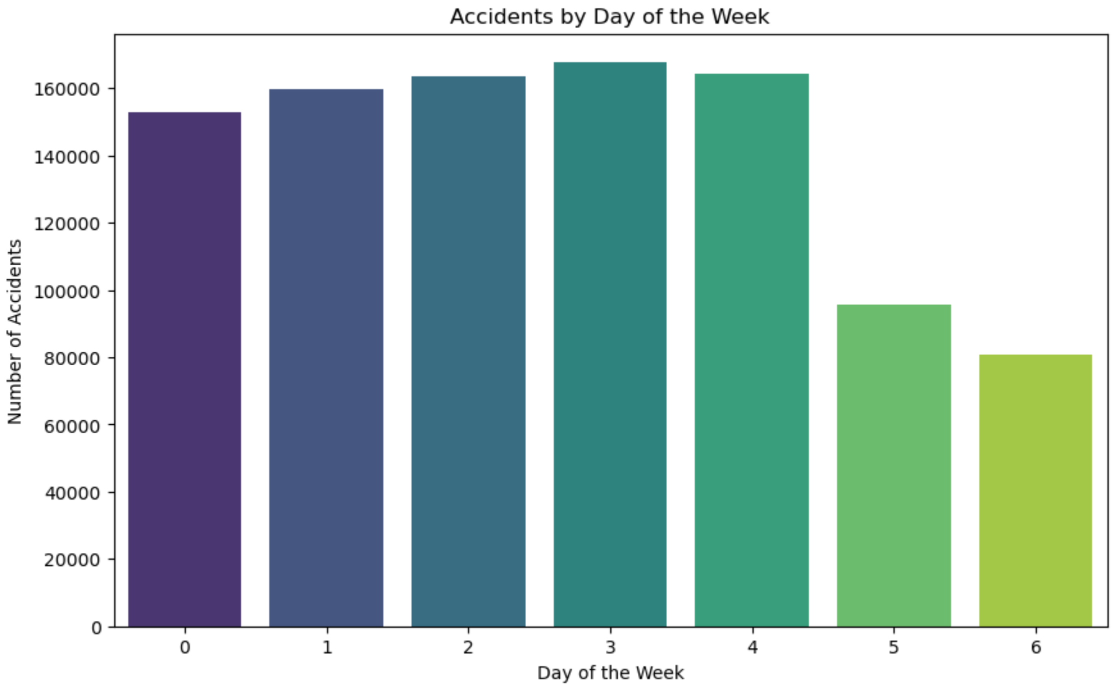
[17]:
```
# Verify the DataFrame to check if 'Hour' column exists
print(df[['Start_Time', 'Hour', 'DayOfWeek', 'Date']].head())
```

```
           Start_Time  Hour  DayOfWeek        Date
0 2016-02-08 00:37:00     0          0  2016-02-08
1 2016-02-08 05:56:00     5          0  2016-02-08
2 2016-02-08 06:15:00     6          0  2016-02-08
3 2016-02-08 06:15:00     6          0  2016-02-08
4 2016-02-08 06:51:00     6          0  2016-02-08
```

[18]:
```
# EDA
# Accidents by time of day
plt.figure(figsize=(10, 6))
sns.countplot(x='Hour', data=df)
plt.title('Accidents by Hour of Day')
plt.xlabel('Hour of Day')
plt.ylabel('Number of Accidents')
plt.show()
```

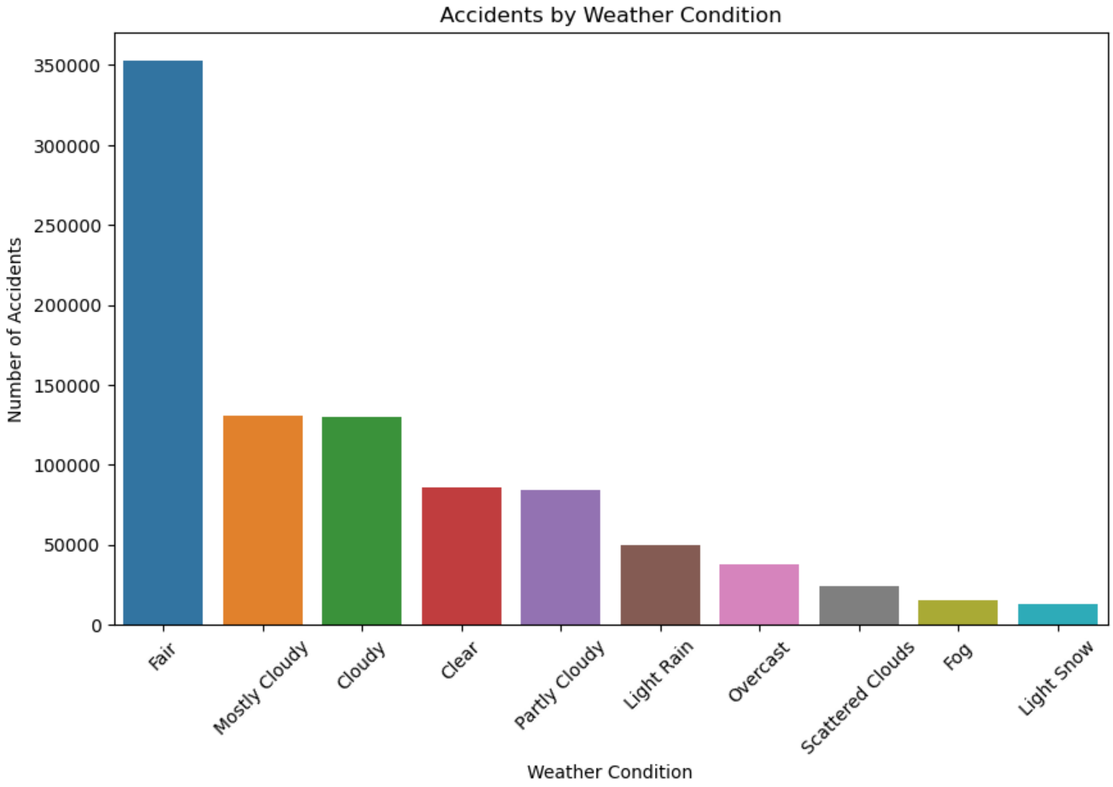


[25]:
```
# Accidents by day of the week
plt.figure(figsize=(10, 6))
sns.countplot(x='DayOfWeek', data=df, palette='viridis')
plt.title('Accidents by Day of the Week')
plt.xlabel('Day of the Week')
plt.ylabel('Number of Accidents')
plt.show()
```

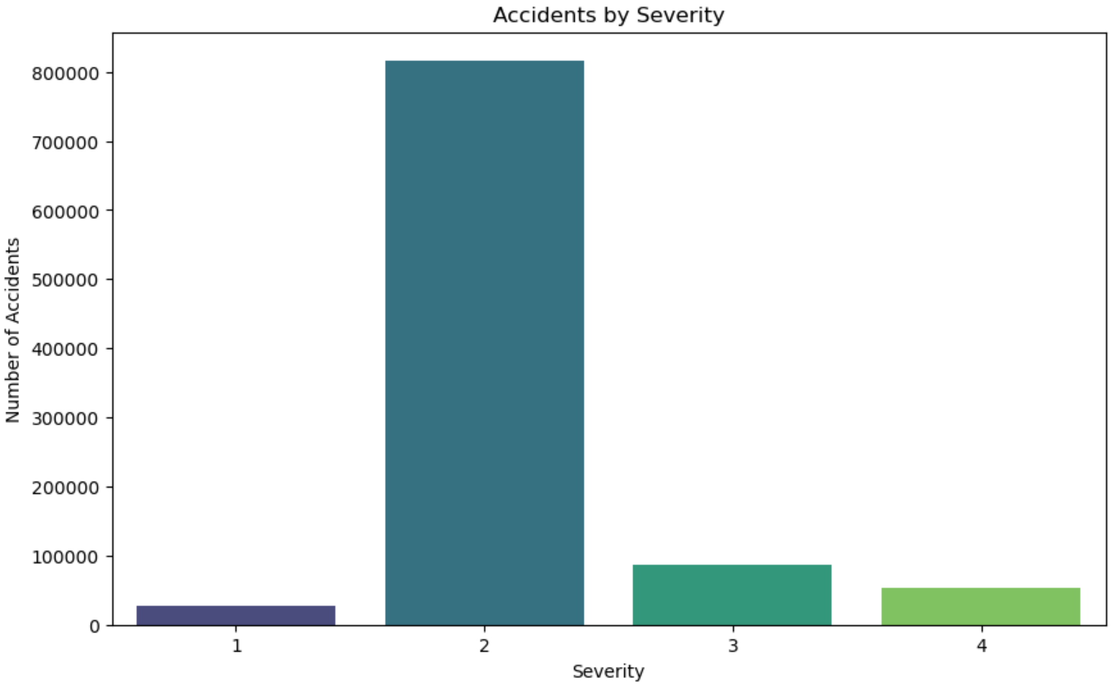


[19]:

```
# Accidents by weather conditions
plt.figure(figsize=(10, 6))
sns.countplot(x='Weather_Condition', data=df, order=df['Weather_Condition'].
 ↪value_counts().iloc[:10].index)
plt.title('Accidents by Weather Condition')
plt.xlabel('Weather Condition')
plt.ylabel('Number of Accidents')
plt.xticks(rotation=45)
plt.show()
```

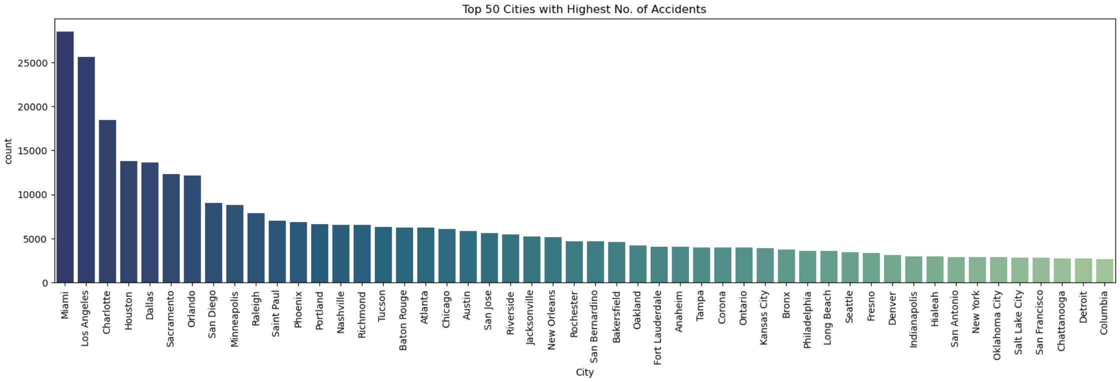


[26]:
```
# Accidents by severity
plt.figure(figsize=(10, 6))
sns.countplot(x='Severity', data=df, palette='viridis')
plt.title('Accidents by Severity')
plt.xlabel('Severity')
plt.ylabel('Number of Accidents')
plt.show()
```



[35]:

```
# Plot for top 50 cities with the highest number of accidents
fig, ax = plt.subplots(figsize=(20, 5))
c = sns.countplot(x="City", data=df, order=df.City.value_counts().iloc[:50].
  ↪index, orient='v', palette="crest_r")
c.set_title("Top 50 Cities with Highest No. of Accidents")
c.set_xticklabels(c.get_xticklabels(), rotation=90)
plt.show()
```

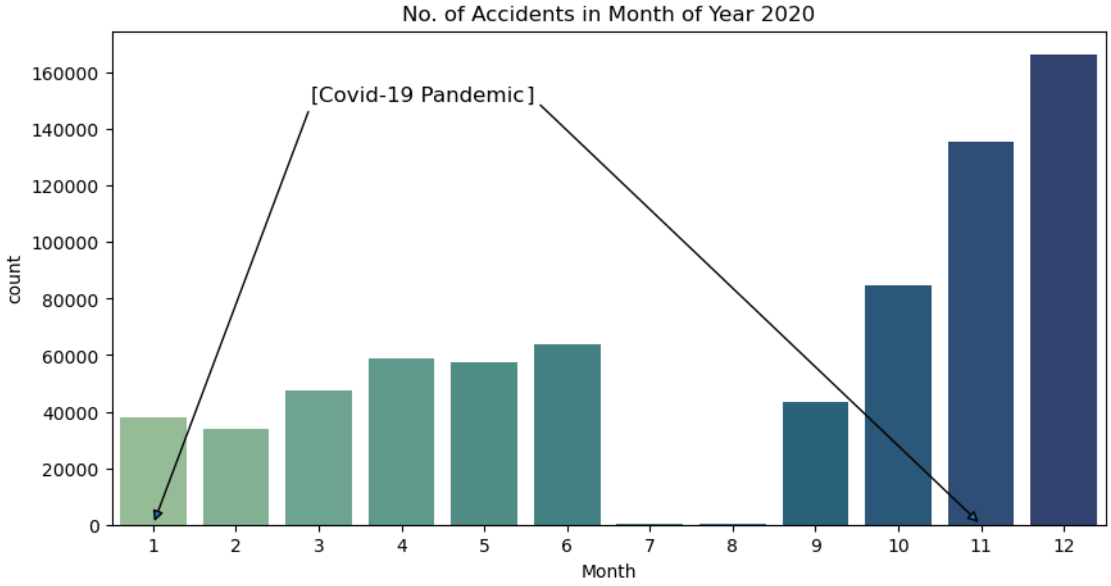


[36]:

```
# Extracting data for the year 2020
df['Start_Time'] = pd.to_datetime(df['Start_Time'])
```

```
data_2020 = df[df['Start_Time'].dt.year == 2020]
data_2020['Month'] = data_2020['Start_Time'].dt.month

# Plot for the number of accidents in the month of year 2020 with annotations
fig, ax = plt.subplots(figsize=(10, 5))
c = sns.countplot(x="Month", data=data_2020, orient='v', palette="crest")
plt.annotate('Covid-19 Pandemic', xy=(2, 150000), fontsize=12)
plt.annotate("[", xy=(0, 0), xytext=(1.9, 150000), arrowprops={'arrowstyle':␣
  ↪'-|>'}, fontsize=12)
plt.annotate("]", xy=(10, 0), xytext=(4.5, 150000), arrowprops={'arrowstyle':␣
  ↪'-|>'}, fontsize=12)
c.set_title("No. of Accidents in Month of Year 2020")
plt.show()
```

```
C:\Users\praga\AppData\Local\Temp\ipykernel_3796\3917644056.py:4:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  data_2020['Month'] = data_2020['Start_Time'].dt.month
```

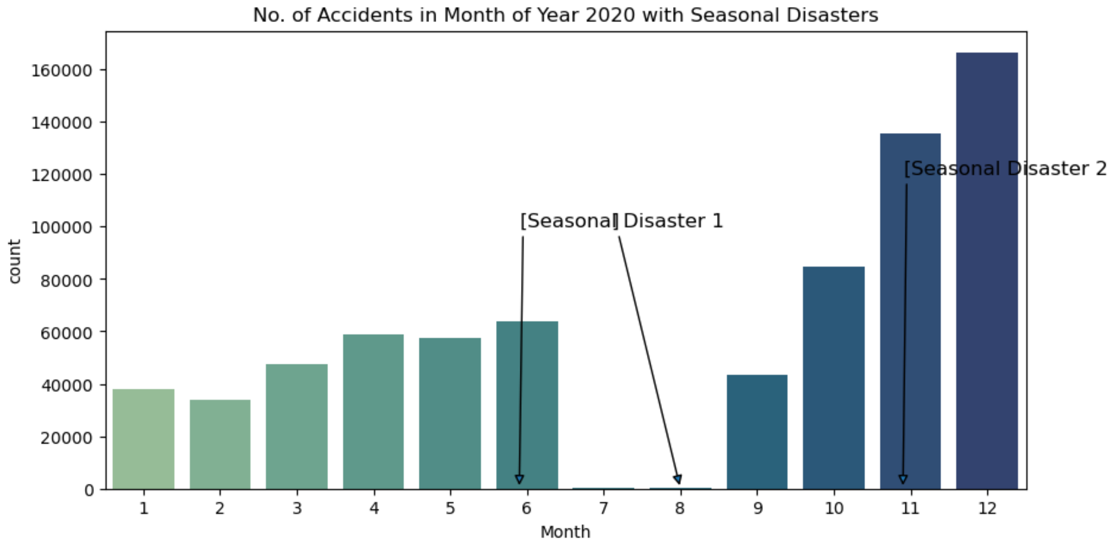


[41]:

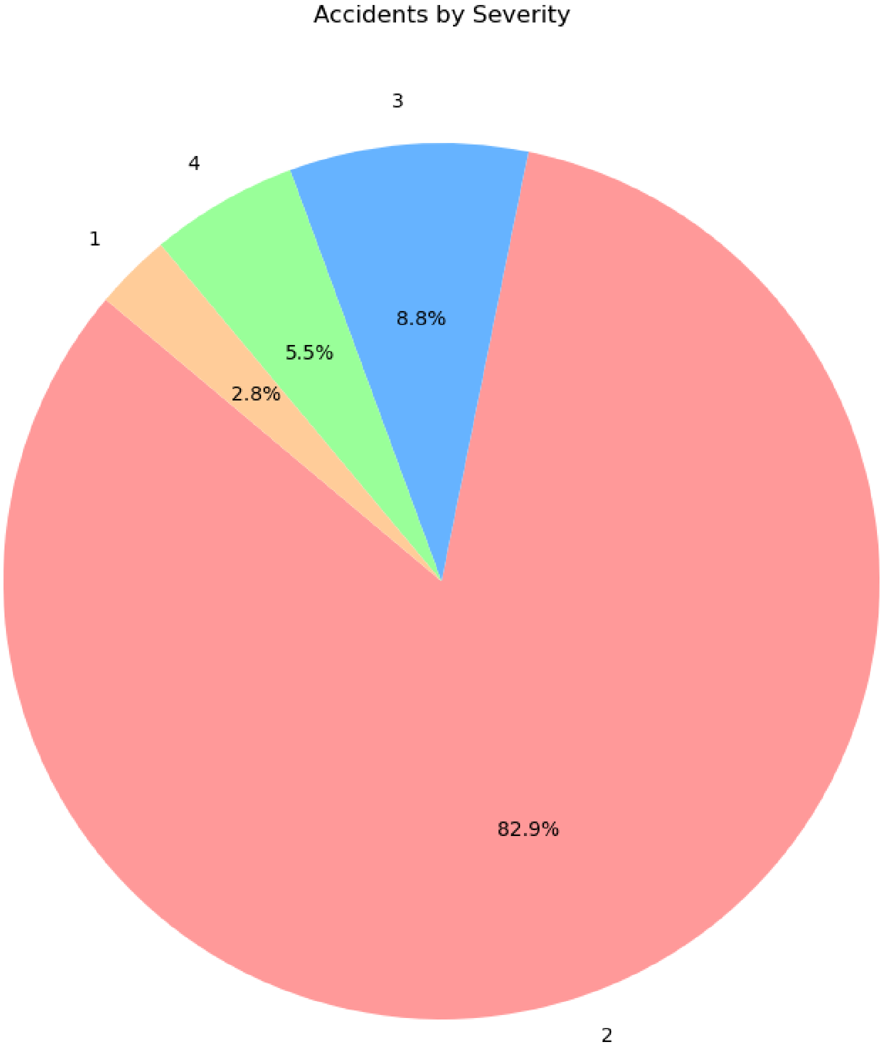
```
fig, ax = plt.subplots(figsize=(10, 5))
c = sns.countplot(x="Month", data=data_2020, orient='v', palette="crest")

# Annotate the seasonal disaster events
```

```
plt.annotate('Seasonal Disaster 1', xy=(5, 100000), fontsize=12)
plt.annotate("[", xy=(4.9, 0), xytext=(4.9, 100000), arrowprops={'arrowstyle':␣
 ↪'-|>'}, fontsize=12)
plt.annotate("]", xy=(7, 0), xytext=(6.1, 100000), arrowprops={'arrowstyle':␣
 ↪'-|>'}, fontsize=12)

plt.annotate('Seasonal Disaster 2', xy=(10, 120000), fontsize=12)
plt.annotate("[", xy=(9.9, 0), xytext=(9.9, 120000), arrowprops={'arrowstyle':␣
 ↪'-|>'}, fontsize=12)
plt.annotate("]", xy=(12, 0), xytext=(11.1, 120000), arrowprops={'arrowstyle':␣
 ↪'-|>'}, fontsize=12)

plt.annotate('Seasonal Disaster 3', xy=(15, 130000), fontsize=12)
plt.annotate("[", xy=(14.9, 0), xytext=(14.9, 130000), arrowprops={'arrowstyle':
 ↪ '-|>'}, fontsize=12)
plt.annotate("]", xy=(17, 0), xytext=(16.1, 130000), arrowprops={'arrowstyle':␣
 ↪'-|>'}, fontsize=12)

c.set_title("No. of Accidents in Month of Year 2020 with Seasonal Disasters")
```

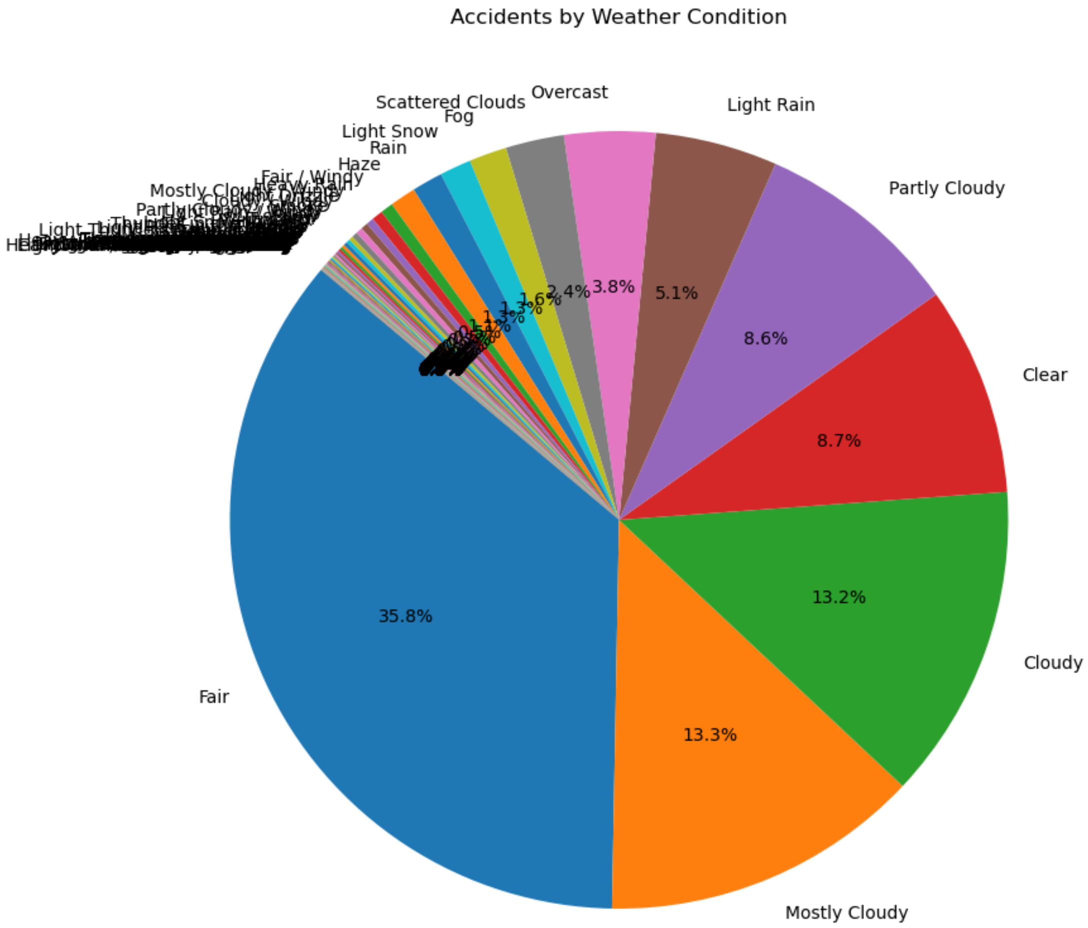
```
[41]: Text(0.5, 1.0, 'No. of Accidents in Month of Year 2020 with Seasonal Disasters')
```



```
[33]: df.dropna(subset=['Severity'], inplace=True)

      # Pie chart of accidents by severity
      severity_counts = df['Severity'].value_counts()

      plt.figure(figsize=(10, 10))
      plt.pie(severity_counts, labels=severity_counts.index, autopct='%1.1f%%',␣
       ↪startangle=140, colors=['#ff9999','#66b3ff','#99ff99','#ffcc99'])
      plt.title('Accidents by Severity')
      plt.show()
```



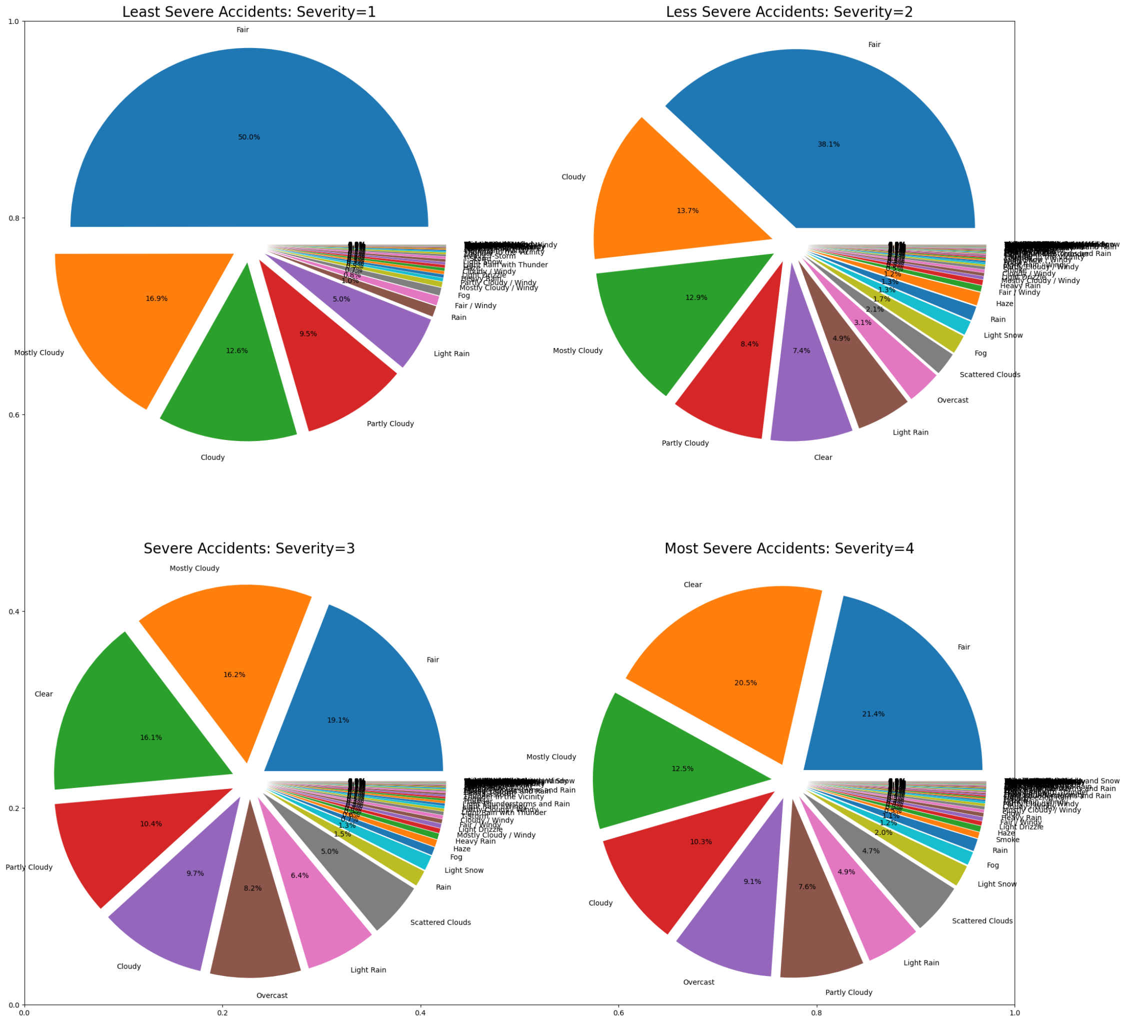
[34]:
```
# Pie chart of accidents by weather condition
weather_counts = df['Weather_Condition'].value_counts()

plt.figure(figsize=(10, 10))
plt.pie(weather_counts, labels=weather_counts.index, autopct='%1.1f%%',␣
 ↪startangle=140)
plt.title('Accidents by Weather Condition')
plt.show()
```



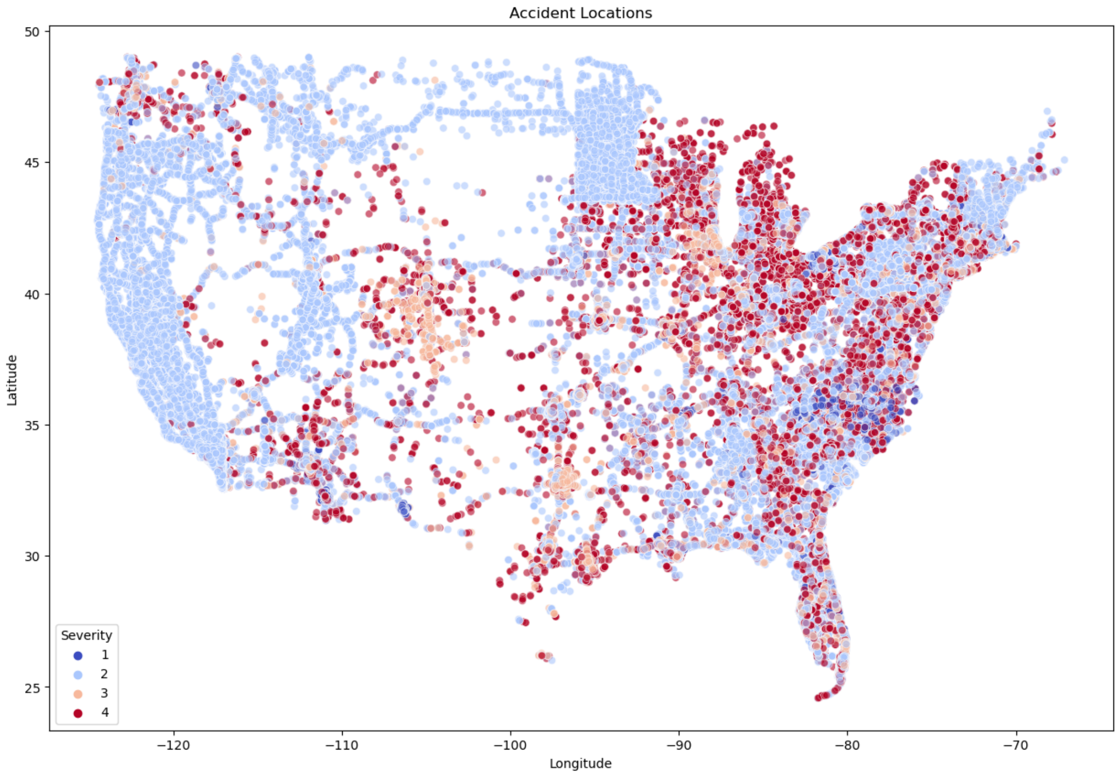
[37]:
```
# Count accidents by weather condition for each severity level
severe_accidents_1 = df[df['Severity'] == 1]['Weather_Condition'].
 ↪value_counts().to_dict()
severe_accidents_2 = df[df['Severity'] == 2]['Weather_Condition'].
 ↪value_counts().to_dict()
severe_accidents_3 = df[df['Severity'] == 3]['Weather_Condition'].
 ↪value_counts().to_dict()
severe_accidents_4 = df[df['Severity'] == 4]['Weather_Condition'].
 ↪value_counts().to_dict()
```

[39]:
```
fig, ax1 = plt.subplots(figsize=[25,25])

# Least Severe Accidents: Severity=1
```

```
ax1 = plt.subplot2grid((2,2),(0,0))
labels = severe_accidents_1.keys()
plt.pie(x=severe_accidents_1.values(), autopct="%.1f%%", explode=[0.
 ↪1]*len(severe_accidents_1), labels=labels, pctdistance=0.5)
plt.title("Least Severe Accidents: Severity=1", fontsize=20)

# Less Severe Accidents: Severity=2
ax1 = plt.subplot2grid((2,2),(0,1))
labels = severe_accidents_2.keys()
plt.pie(x=severe_accidents_2.values(), autopct="%.1f%%", explode=[0.
 ↪1]*len(severe_accidents_2), labels=labels, pctdistance=0.5)
plt.title("Less Severe Accidents: Severity=2", fontsize=20)

# Severe Accidents: Severity=3
ax1 = plt.subplot2grid((2,2),(1,0))
labels = severe_accidents_3.keys()
plt.pie(x=severe_accidents_3.values(), autopct="%.1f%%", explode=[0.
 ↪1]*len(severe_accidents_3), labels=labels, pctdistance=0.5)
plt.title("Severe Accidents: Severity=3", fontsize=20)

# Most Severe Accidents: Severity=4
ax1 = plt.subplot2grid((2,2),(1,1))
labels = severe_accidents_4.keys()
plt.pie(x=severe_accidents_4.values(), autopct="%.1f%%", explode=[0.
 ↪1]*len(severe_accidents_4), labels=labels, pctdistance=0.5)
plt.title("Most Severe Accidents: Severity=4", fontsize=20)
plt.show()
```



```
[31]: # Scatter plot of accident locations
plt.figure(figsize=(15, 10))
sns.scatterplot(y=df['Start_Lat'], x=df['Start_Lng'], hue=df['Severity'],␣
 ↪palette='coolwarm', alpha=0.6)
plt.title('Accident Locations')
plt.xlabel('Longitude')
plt.ylabel('Latitude')
plt.legend(title='Severity')
plt.show()
```



[21]:

```
# Visualize Accident Hotspots
map_data = df[['Start_Lat', 'Start_Lng']]
map_data = map_data.dropna()
```

[29]:

```
map_accidents = folium.Map(location=[map_data['Start_Lat'].mean(),␣
  ↪map_data['Start_Lng'].mean()], zoom_start=5)
HeatMap(data=map_data, radius=10).add_to(map_accidents)
map_accidents.save('accident_hotspots.html')
```

[27]:

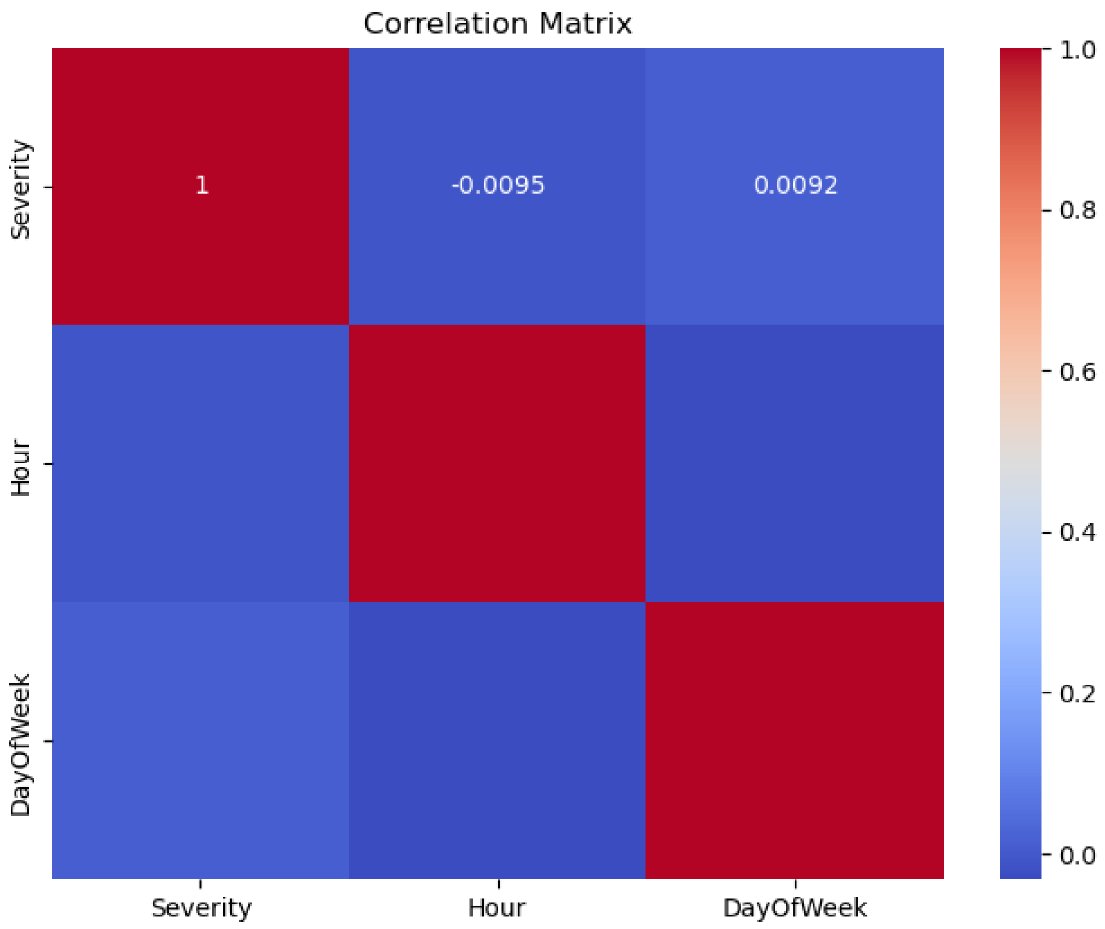
```
import folium
from folium.plugins import HeatMap

# Create a base map
base_map = folium.Map(location=[df['Start_Lat'].mean(), df['Start_Lng'].
  ↪mean()], zoom_start=5)

# Add heatmap layer
heat_data = [[row['Start_Lat'], row['Start_Lng']] for index, row in df.
  ↪iterrows()]
HeatMap(heat_data).add_to(base_map)

# Save the map as HTML file
```

```
base_map.save('accident_hotspots.html')

# Display the map
base_map
```

[27]:
```
<folium.folium.Map at 0x1690a8bc910>
```

[30]:
```
import folium
from folium.plugins import HeatMap

# Create a map centered around the average latitude and longitude
m = folium.Map(location=[df['Start_Lat'].mean(), df['Start_Lng'].mean()],␣
 ↪zoom_start=10)

# Prepare data for the heatmap
heat_data = [[row['Start_Lat'], row['Start_Lng']] for index, row in df.
 ↪iterrows()]

# Create a heatmap layer
HeatMap(heat_data).add_to(m)

# Save map to an HTML file
m.save('accident_hotspots.html')
```

[23]:
```
# Correlation analysis
correlation_matrix = df[['Severity', 'Hour', 'DayOfWeek']].corr()
plt.figure(figsize=(8, 6))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
plt.title('Correlation Matrix')
plt.show()
```


Correlation Matrix
Severity
Hour
DayOfWeek
1
-0.0095
0.0092
Severity
Hour
DayOfWeek
1.0
0.8
0.6
0.4
0.2
0.0

```
[ ]:
```