In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import folium
from folium.plugins import HeatMap
```

In [2]:
```
# Load the dataset (Ensure the CSV file is in the same folder)
df = pd.read_csv("US_Accidents_March23.csv")  # Update the filename if needed
```

In [3]:
```
# Display basic info
print(df.info())
print(df.head())
```

```
<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
None
    ID   Source  Severity           Start_Time             End_Time  \
0  A-1  Source2         3  2016-02-08 05:46:00  2016-02-08 11:00:00
1  A-2  Source2         2  2016-02-08 06:07:59  2016-02-08 06:37:59
2  A-3  Source2         2  2016-02-08 06:49:27  2016-02-08 07:19:27
3  A-4  Source2         3  2016-02-08 07:23:34  2016-02-08 07:53:34
4  A-5  Source2         2  2016-02-08 07:39:07  2016-02-08 08:09:07

   Start_Lat  Start_Lng  End_Lat  End_Lng  Distance(mi)  ... Roundabout  \
0  39.865147 -84.058723      NaN      NaN          0.01  ...      False
1  39.928059 -82.831184      NaN      NaN          0.01  ...      False
2  39.063148 -84.032608      NaN      NaN          0.01  ...      False
3  39.747753 -84.205582      NaN      NaN          0.01  ...      False
4  39.627781 -84.188354      NaN      NaN          0.01  ...      False
```

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

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

[5 rows x 46 columns]
```

In [4]:
```
# Handle missing values (drop rows with missing values in critical columns)
df.dropna(subset=['Start_Time', 'End_Time', 'Start_Lat', 'Start_Lng', 'Weather_Condition'], inplace=True)
```

In [5]:
```
# Convert Start_Time to datetime format
# Convert Start_Time to datetime (handling errors)
df['Start_Time'] = pd.to_datetime(df['Start_Time'], errors='coerce', format='%Y-%m-%d %H:%M:%S', exact=False)
```

In [6]:
```
# Drop rows where conversion failed
df.dropna(subset=['Start_Time'], inplace=True)
```

In [7]:
```
df['Start_Time'] = pd.to_datetime(df['Start_Time'])
```

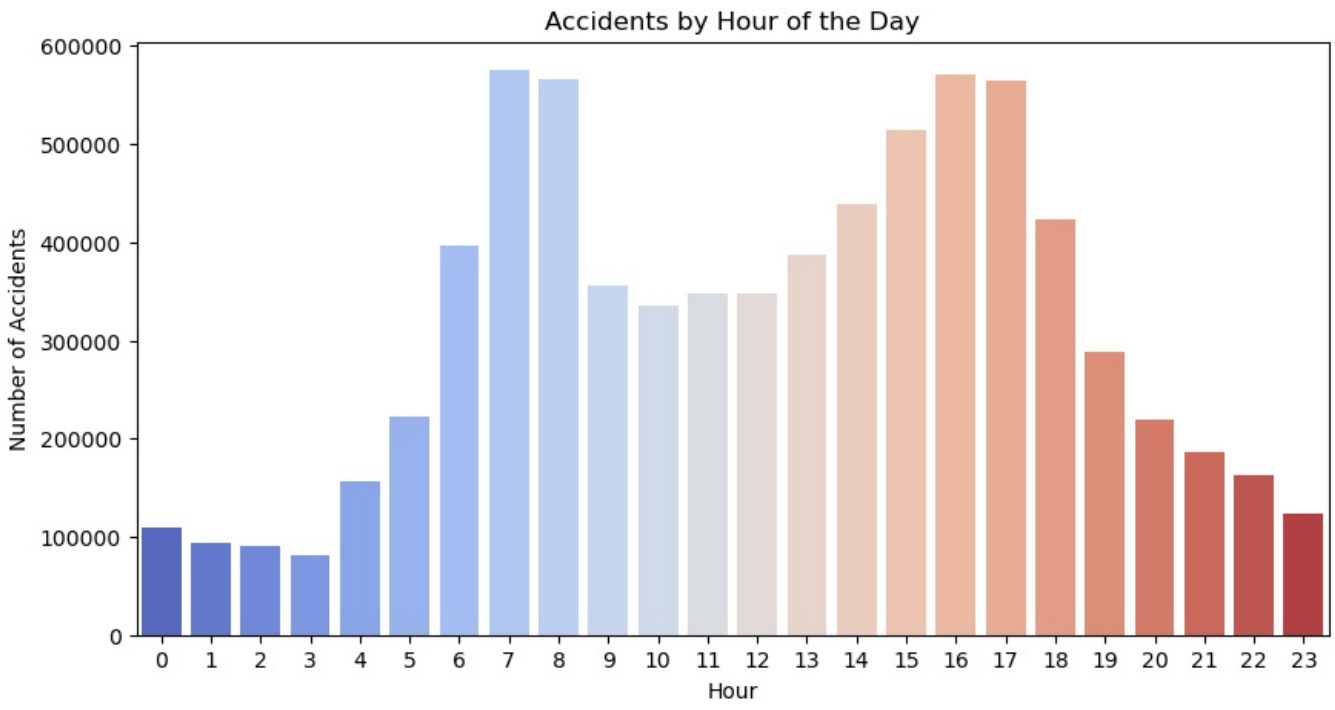
In [8]:
```
# Extract relevant time-based features
df['Hour'] = df['Start_Time'].dt.hour
df['DayOfWeek'] = df['Start_Time'].dt.dayofweek
df['Month'] = df['Start_Time'].dt.month
```

In [9]:
```
#  **Visualization 1: Accidents by Hour of Day**
plt.figure(figsize=(10, 5))
sns.countplot(x='Hour', data=df, palette='coolwarm')
plt.title('Accidents by Hour of the Day')
plt.xlabel('Hour')
plt.ylabel('Number of Accidents')
plt.show()
```

```
/var/folders/r3/trn6xwcx7js3_2j79fkcxz880000gn/T/ipykernel_34778/1268362251.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 `legend=False` for the same effect.

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



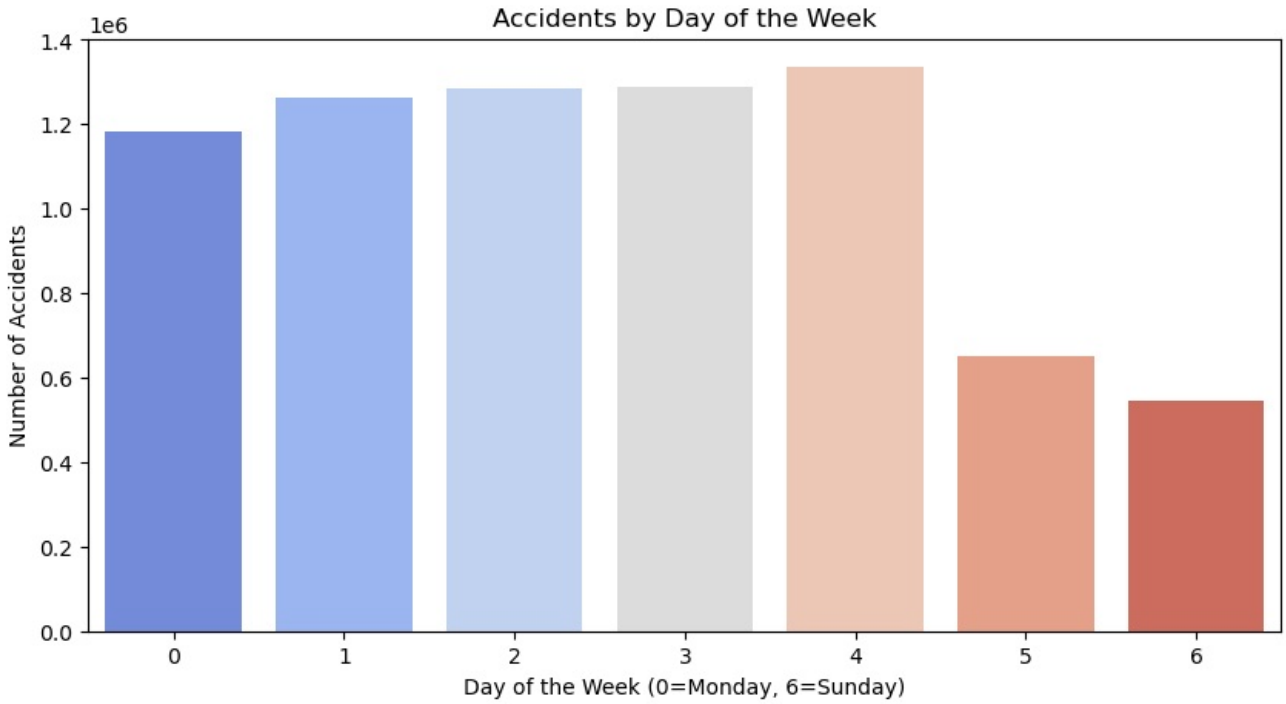
In [10]:
```
#  **Visualization 2: Accidents by Day of the Week**
plt.figure(figsize=(10, 5))
sns.countplot(x='DayOfWeek', data=df, palette='coolwarm')
plt.title('Accidents by Day of the Week')
plt.xlabel('Day of the Week (0=Monday, 6=Sunday)')
plt.ylabel('Number of Accidents')
```

```
plt.show()
```

```
/var/folders/r3/trn6xwcx7js3_2j79fkcxz880000gn/T/ipykernel_34778/2644590497.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 `legend=False` for the same effect.

  sns.countplot(x='DayOfWeek', data=df, palette='coolwarm')
```

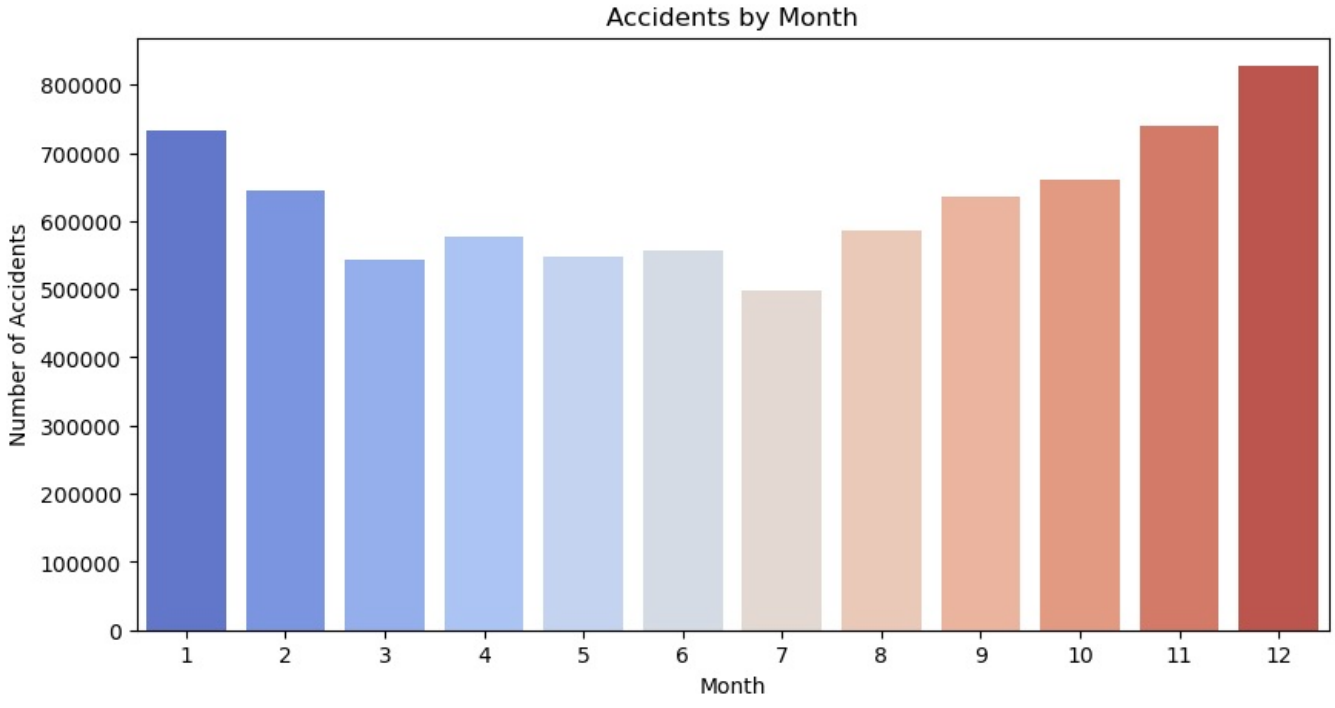


In [11]:
```
#  **Visualization 3: Accidents by Month**
plt.figure(figsize=(10, 5))
sns.countplot(x='Month', data=df, palette='coolwarm')
plt.title('Accidents by Month')
plt.xlabel('Month')
plt.ylabel('Number of Accidents')
plt.show()
```

```
/var/folders/r3/trn6xwcx7js3_2j79fkcxz880000gn/T/ipykernel_34778/3104947058.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 `legend=False` for the same effect.

  sns.countplot(x='Month', data=df, palette='coolwarm')
```



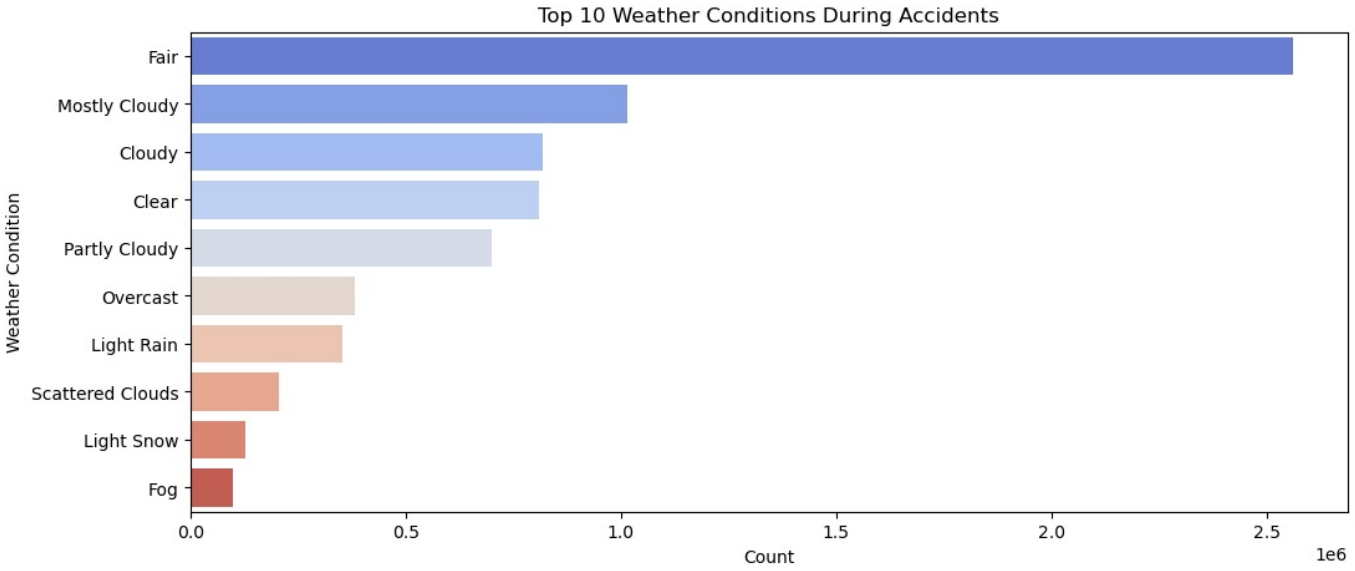
In [12]:
```
#  **Visualization 4: Impact of Weather Conditions on Accidents**
plt.figure(figsize=(12, 5))
sns.countplot(y='Weather_Condition', data=df, order=df['Weather_Condition'].value_counts().index[:10], palette=
plt.title('Top 10 Weather Conditions During Accidents')
```

```
plt.xlabel('Count')
plt.ylabel('Weather Condition')
plt.show()
```

```
/var/folders/r3/trn6xwcx7js3_2j79fkcxz880000gn/T/ipykernel_34778/2930784886.py:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable
to `hue` and set `legend=False` for the same effect.

  sns.countplot(y='Weather_Condition', data=df, order=df['Weather_Condition'].value_counts().index[:10], palette
='coolwarm')
```



In [13]:
```
#  **Visualization 5: Heatmap of Accident Locations**
m = folium.Map(location=[df['Start_Lat'].mean(), df['Start_Lng'].mean()], zoom_start=5)
```

In [14]:
```
# Prepare heatmap data
heat_data = [[row['Start_Lat'], row['Start_Lng']] for index, row in df.iterrows()]
```

In [15]:
```
# Add heatmap to the map
HeatMap(heat_data[:10000]).add_to(m)  # Using first 10,000 accidents for performance
```

Out[15]:
```
<folium.plugins.heat_map.HeatMap at 0x3aa44bb60>
```

In [16]:
```
# Save the map to an HTML file
m.save("accident_heatmap.html")
print("Heatmap saved as 'accident_heatmap.html'. Open it in a browser to view.")
```

```
Heatmap saved as 'accident_heatmap.html'. Open it in a browser to view.
```