In [24]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [26]:
```
data = pd.read_csv("Road Accident Data.csv")

print(data.head())
print(data.info())
```

```
  Accident_Index Accident Date Day_of_Week          Junction_Control  \
0  200901BS70001      1/1/2021    Thursday  Give way or uncontrolled
1  200901BS70002      1/5/2021      Monday  Give way or uncontrolled
2  200901BS70003      1/4/2021      Sunday  Give way or uncontrolled
3  200901BS70004      1/5/2021      Monday       Auto traffic signal
4  200901BS70005      1/6/2021     Tuesday       Auto traffic signal

          Junction_Detail Accident_Severity   Latitude  \
0  T or staggered junction           Serious  51.512273
1              Crossroads           Serious  51.514399
2  T or staggered junction            Slight  51.486668
3  T or staggered junction           Serious  51.507804
4              Crossroads           Serious  51.482076

        Light_Conditions Local_Authority_(District) Carriageway_Hazards  ...  \
0               Daylight     Kensington and Chelsea                 NaN  ...
1               Daylight     Kensington and Chelsea                 NaN  ...
2               Daylight     Kensington and Chelsea                 NaN  ...
3               Daylight     Kensington and Chelsea                 NaN  ...
4  Darkness - lights lit     Kensington and Chelsea                 NaN  ...

   Number_of_Casualties  Number_of_Vehicles         Police_Force  \
0                     1                   2  Metropolitan Police
1                    11                   2  Metropolitan Police
2                     1                   2  Metropolitan Police
3                     1                   2  Metropolitan Police
4                     1                   2  Metropolitan Police

  Road_Surface_Conditions           Road_Type Speed_limit   Time  \
0                     Dry      One way street          30  15:11
1             Wet or damp  Single carriageway          30  10:59
2                     Dry  Single carriageway          30  14:19
3            Frost or ice  Single carriageway          30   8:10
4                     Dry  Single carriageway          30  17:25

  Urban_or_Rural_Area  Weather_Conditions          Vehicle_Type
0               Urban  Fine no high winds                   Car
1               Urban  Fine no high winds  Taxi/Private hire car
2               Urban  Fine no high winds  Taxi/Private hire car
3               Urban               Other  Motorcycle over 500cc
4               Urban  Fine no high winds                   Car

[5 rows x 21 columns]
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 307973 entries, 0 to 307972
Data columns (total 21 columns):
 #   Column                      Non-Null Count   Dtype
---  ------                      --------------   -----
 0   Accident_Index              307973 non-null  object
 1   Accident Date               307973 non-null  object
 2   Day_of_Week                 307973 non-null  object
 3   Junction_Control            307973 non-null  object
 4   Junction_Detail             307973 non-null  object
 5   Accident_Severity           307973 non-null  object
 6   Latitude                    307973 non-null  float64
 7   Light_Conditions            307973 non-null  object
```

```
 8   Local_Authority_(District)  307973 non-null  object
 9   Carriageway_Hazards         5424 non-null    object
 10  Longitude                   307973 non-null  float64
 11  Number_of_Casualties        307973 non-null  int64
 12  Number_of_Vehicles          307973 non-null  int64
 13  Police_Force                307973 non-null  object
 14  Road_Surface_Conditions     307656 non-null  object
 15  Road_Type                   306439 non-null  object
 16  Speed_limit                 307973 non-null  int64
 17  Time                        307956 non-null  object
 18  Urban_or_Rural_Area         307973 non-null  object
 19  Weather_Conditions          301916 non-null  object
 20  Vehicle_Type                307973 non-null  object
dtypes: float64(2), int64(3), object(16)
memory usage: 49.3+ MB
None
```

In [36]:
```
print(data.columns)
```

```
Index(['Accident_Index', 'Accident Date', 'Day_of_Week', 'Junction_Control',
       'Junction_Detail', 'Accident_Severity', 'Latitude', 'Light_Conditions',
       'Local_Authority_(District)', 'Carriageway_Hazards', 'Longitude',
       'Number_of_Casualties', 'Number_of_Vehicles', 'Police_Force',
       'Road_Surface_Conditions', 'Road_Type', 'Speed_limit', 'Time',
       'Urban_or_Rural_Area', 'Weather_Conditions', 'Vehicle_Type'],
      dtype='object')
```

In [40]:
```
data['Weather_Conditions'] = data['Weather_Conditions'].fillna(data['Weather_Condit
```

In [44]:
```
data.columns = data.columns.str.strip()  # Removes leading spaces
```

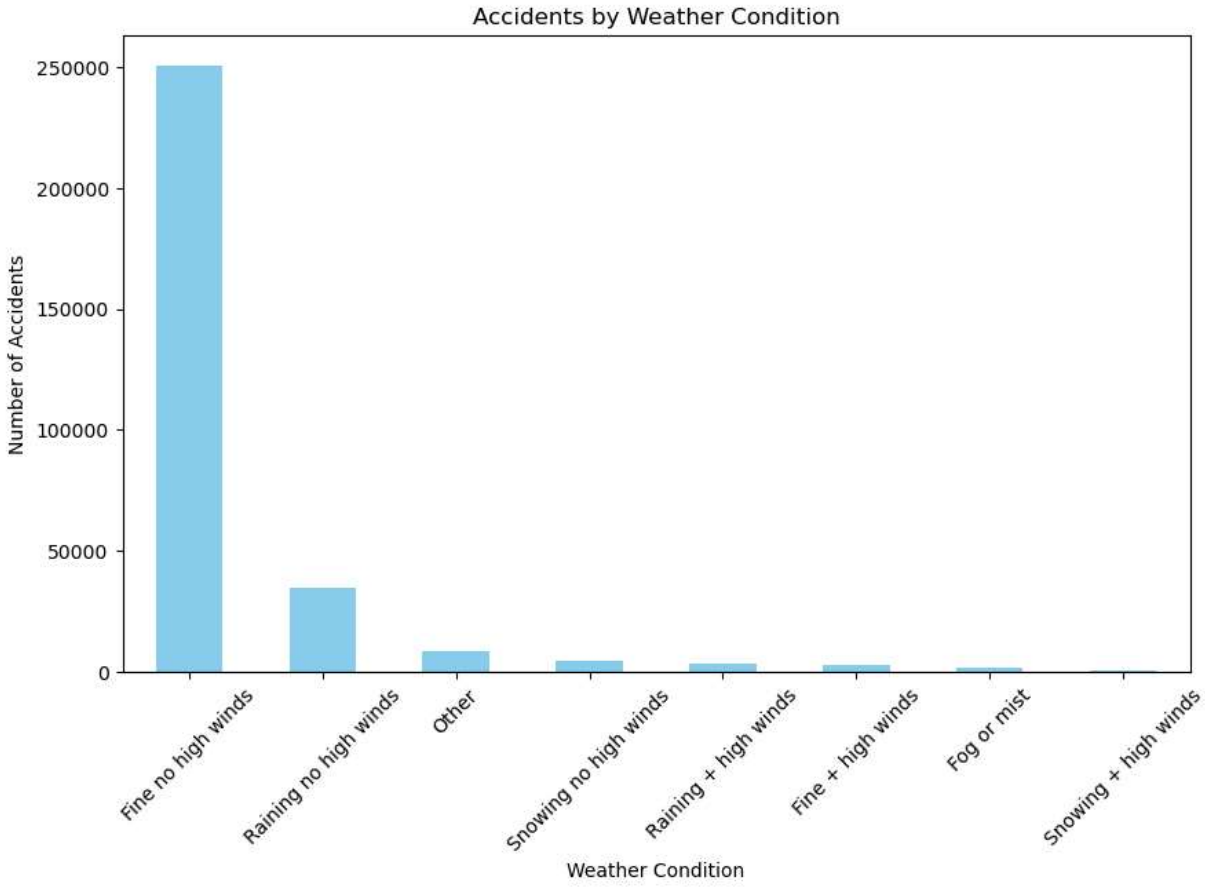
In [46]:
```
if 'Weather_Conditions' in data.columns:
    data['Weather_Conditions'] = data['Weather_Conditions'].fillna(data['Weather_Co
else:
    print("Column 'Weather_Conditions' not found in the dataset.")
```

In [48]:
```
data['Weather_Conditions'] = data['Weather_Conditions'].fillna(data['Weather_Condit
```

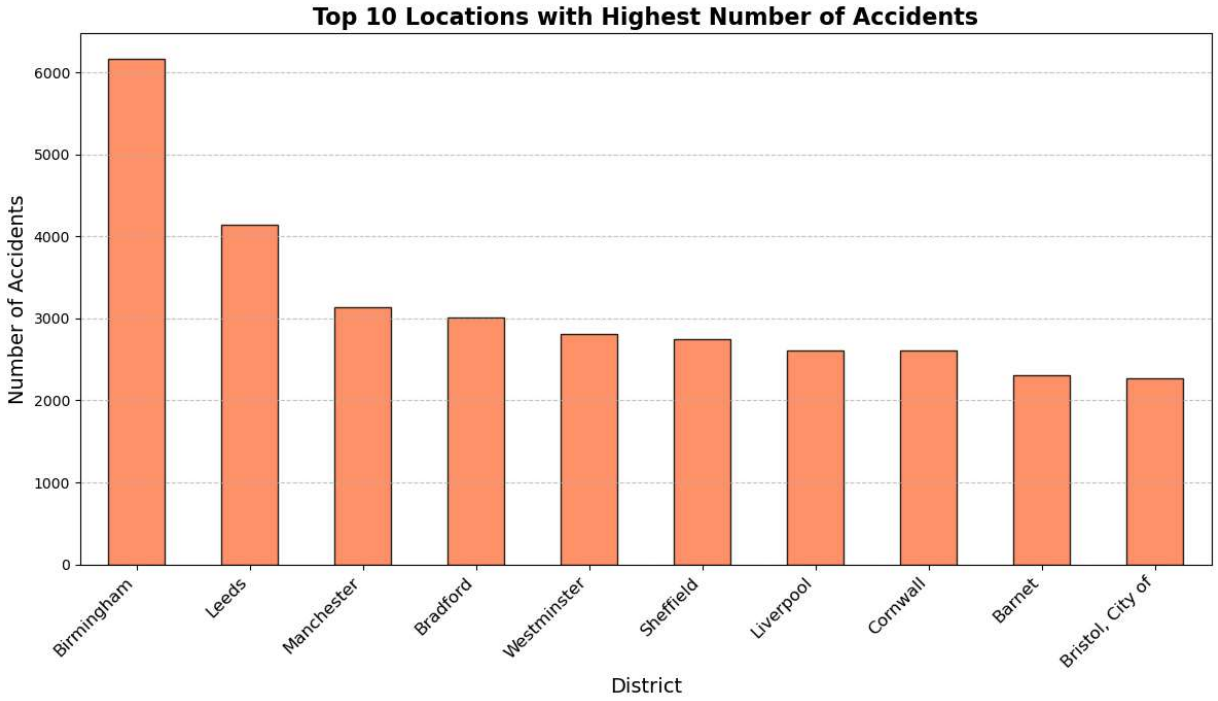
In [52]:
```
print(data.isnull().sum())

# Fill missing values in the 'Weather_Conditions' column
data['Weather_Conditions'] = data['Weather_Conditions'].fillna(data['Weather_Condit

# Drop irrelevant or excessive missing columns (if applicable)
data = data.drop(columns=['Irrelevant_Column1', 'Irrelevant_Column2'], errors='igno

# Confirm the changes
print(data.isnull().sum())
```

```
Accident_Index                    0
Accident Date                     0
Day_of_Week                       0
Junction_Control                  0
Junction_Detail                   0
Accident_Severity                 0
Latitude                          0
Light_Conditions                  0
Local_Authority_(District)        0
Carriageway_Hazards          302549
Longitude                         0
Number_of_Casualties              0
Number_of_Vehicles                0
Police_Force                      0
Road_Surface_Conditions         317
Road_Type                      1534
Speed_limit                       0
Time                             17
Urban_or_Rural_Area               0
Weather_Conditions                0
Vehicle_Type                      0
dtype: int64
Accident_Index                    0
Accident Date                     0
Day_of_Week                       0
Junction_Control                  0
Junction_Detail                   0
Accident_Severity                 0
Latitude                          0
Light_Conditions                  0
Local_Authority_(District)        0
Carriageway_Hazards          302549
Longitude                         0
Number_of_Casualties              0
Number_of_Vehicles                0
Police_Force                      0
Road_Surface_Conditions         317
Road_Type                      1534
Speed_limit                       0
Time                             17
Urban_or_Rural_Area               0
Weather_Conditions                0
Vehicle_Type                      0
dtype: int64
```

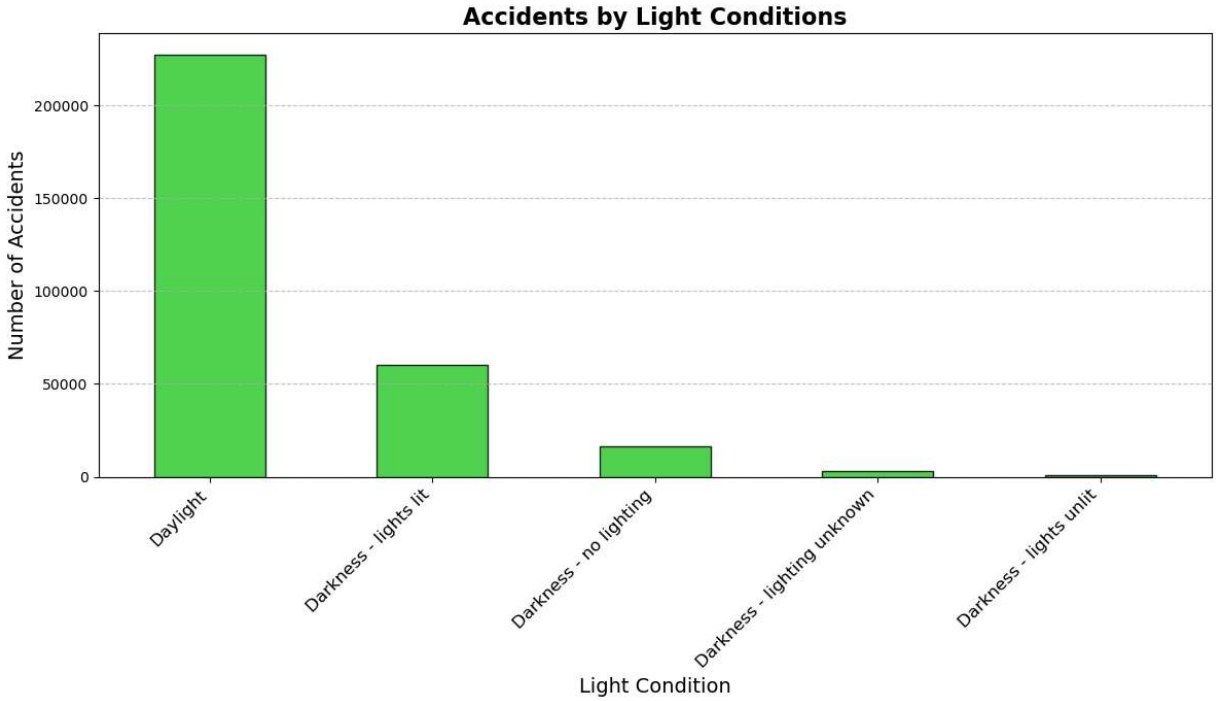
In [56]:
```
# Count accidents by weather condition
weather_counts = data['Weather_Conditions'].value_counts()

# Plot
plt.figure(figsize=(10, 6))
weather_counts.plot(kind='bar', color='skyblue')
plt.title('Accidents by Weather Condition')
plt.xlabel('Weather Condition')
plt.ylabel('Number of Accidents')
plt.xticks(rotation=45)
plt.show()
```



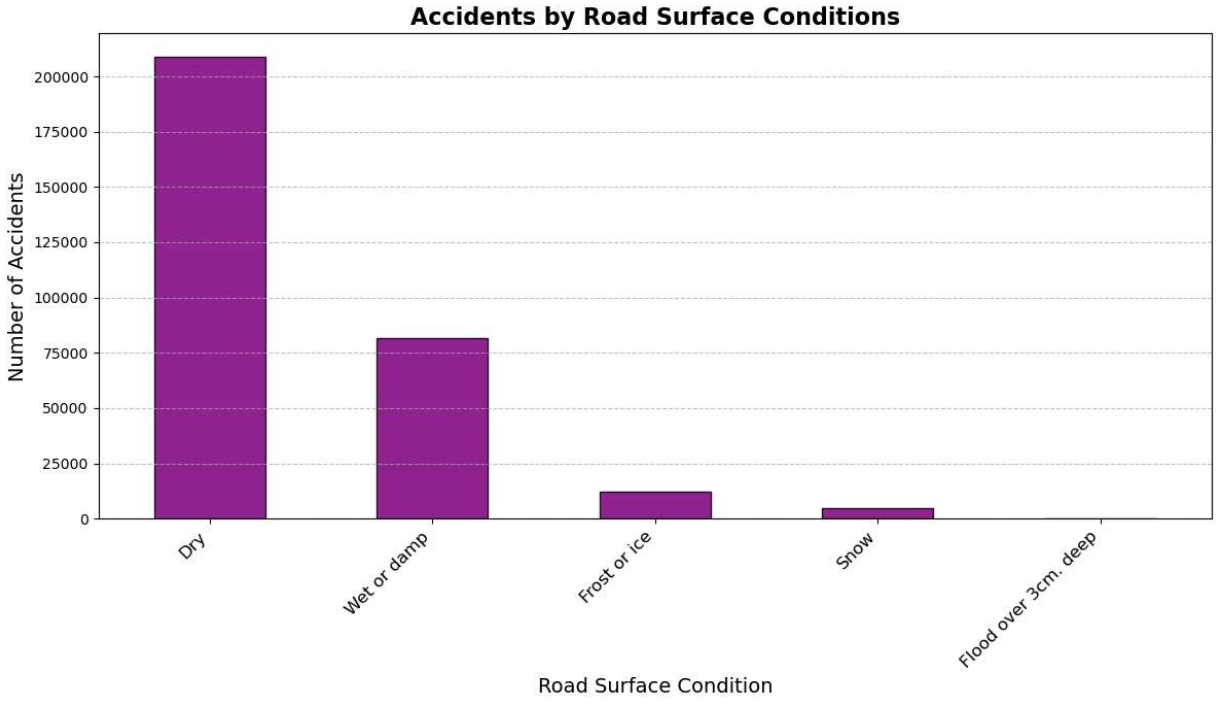
In [58]:
```
# Count accidents by location
location_counts = data['Local_Authority_(District)'].value_counts().head(10)

# Plotting
plt.figure(figsize=(12, 7))
location_counts.plot(
    kind='bar',
    color='coral',
    edgecolor='black',
    alpha=0.85
)

# Adding titles and labels
plt.title('Top 10 Locations with Highest Number of Accidents', fontsize=16, fontwei
plt.xlabel('District', fontsize=14)
plt.ylabel('Number of Accidents', fontsize=14)

# Rotating x-axis labels
plt.xticks(rotation=45, ha='right', fontsize=12)

# Adding grid
plt.grid(axis='y', linestyle='--', alpha=0.7)

# Display the plot
plt.tight_layout()
plt.show()
```



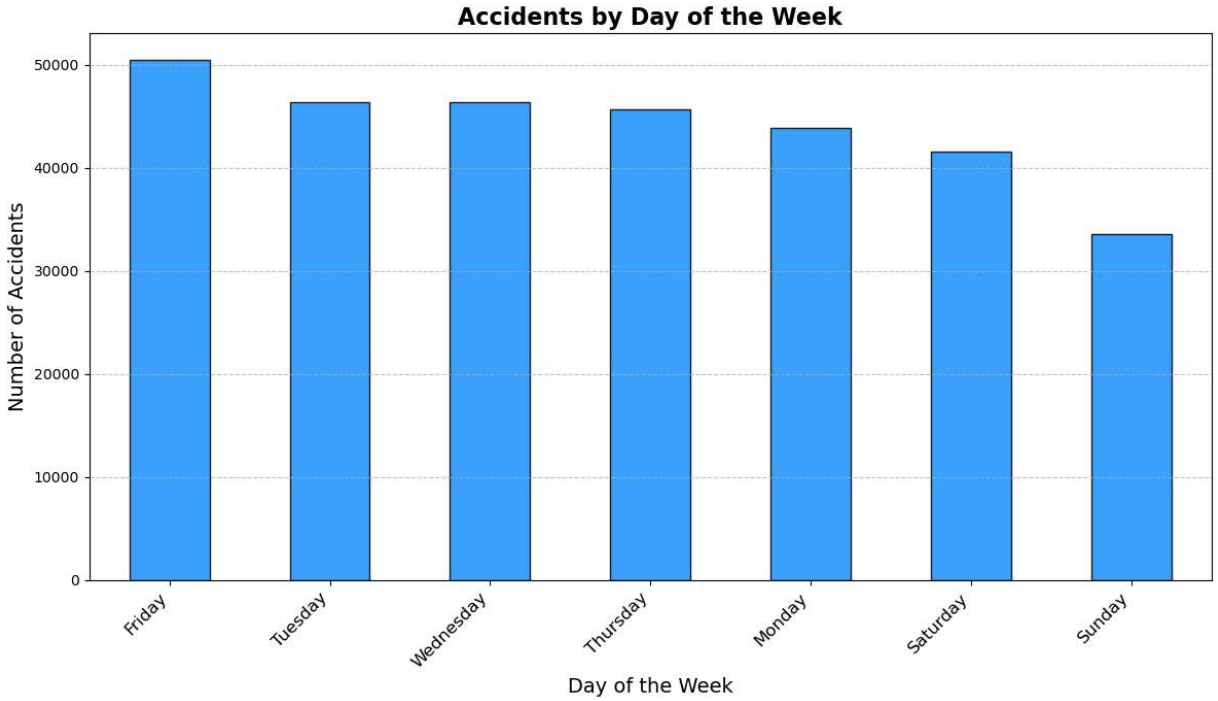
In [60]:
```
# Count accidents by light conditions
light_condition_counts = data['Light_Conditions'].value_counts()

# Plotting
plt.figure(figsize=(12, 7))
light_condition_counts.plot(
    kind='bar',
    color='limegreen',
    edgecolor='black',
    alpha=0.85
)

# Adding titles and labels
plt.title('Accidents by Light Conditions', fontsize=16, fontweight='bold')
plt.xlabel('Light Condition', fontsize=14)
plt.ylabel('Number of Accidents', fontsize=14)

# Rotating x-axis labels
plt.xticks(rotation=45, ha='right', fontsize=12)

# Adding grid
plt.grid(axis='y', linestyle='--', alpha=0.7)

# Display the plot
plt.tight_layout()
plt.show()
```



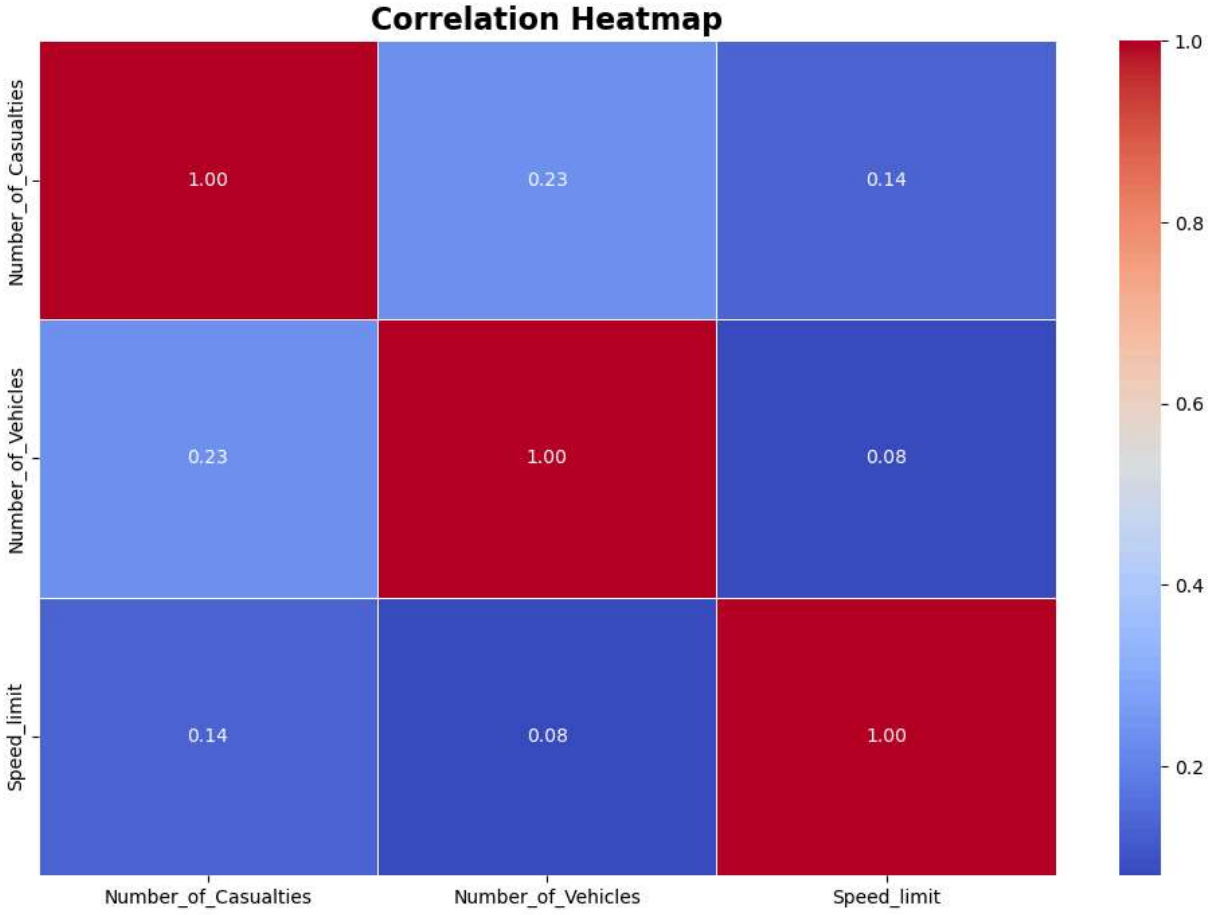
In [62]:
```
# Count accidents by road surface conditions
road_surface_counts = data['Road_Surface_Conditions'].value_counts()

# Plotting
plt.figure(figsize=(12, 7))
road_surface_counts.plot(
    kind='bar',
    color='purple',
    edgecolor='black',
    alpha=0.85
)

# Adding titles and labels
plt.title('Accidents by Road Surface Conditions', fontsize=16, fontweight='bold')
plt.xlabel('Road Surface Condition', fontsize=14)
plt.ylabel('Number of Accidents', fontsize=14)

# Rotating x-axis labels
plt.xticks(rotation=45, ha='right', fontsize=12)

# Adding grid
plt.grid(axis='y', linestyle='--', alpha=0.7)

# Display the plot
plt.tight_layout()
plt.show()
```



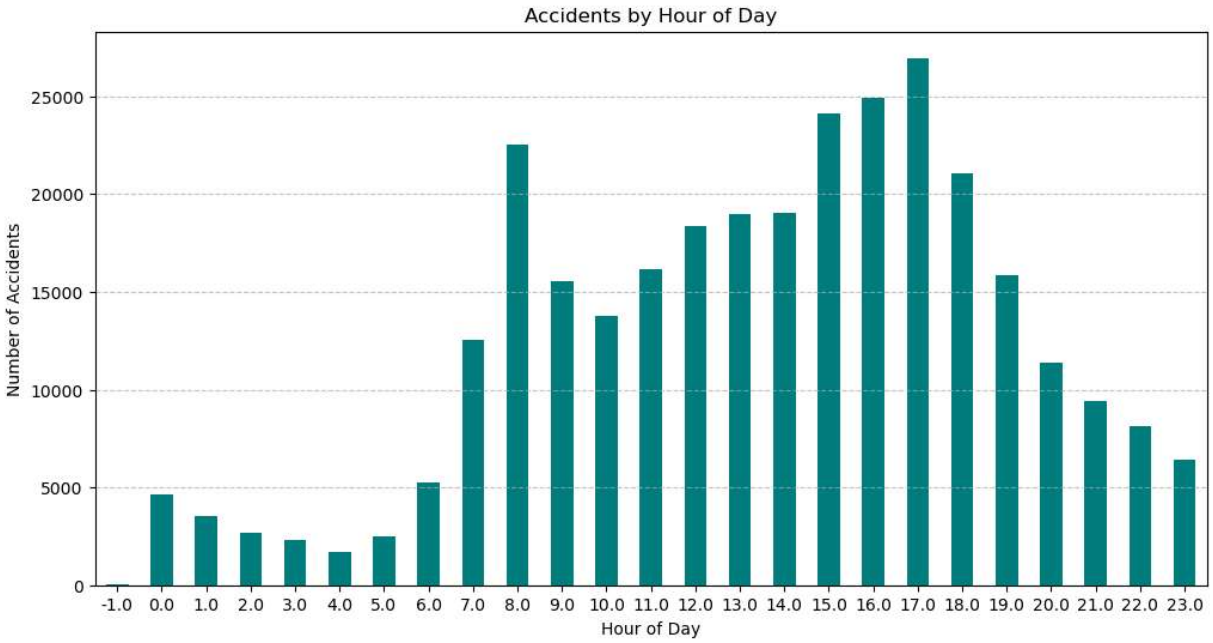
In [64]:
```
# Count accidents by day of the week
day_counts = data['Day_of_Week'].value_counts()

# Plotting
plt.figure(figsize=(12, 7))
day_counts.plot(
    kind='bar',
    color='dodgerblue',
    edgecolor='black',
    alpha=0.85
)

# Adding titles and labels
plt.title('Accidents by Day of the Week', fontsize=16, fontweight='bold')
plt.xlabel('Day of the Week', fontsize=14)
plt.ylabel('Number of Accidents', fontsize=14)

# Rotating x-axis labels
plt.xticks(rotation=45, ha='right', fontsize=12)

# Adding grid
plt.grid(axis='y', linestyle='--', alpha=0.7)

# Display the plot
plt.tight_layout()
plt.show()
```



In [66]:
```
import seaborn as sns

# Selecting numerical columns for correlation
numerical_data = data[['Number_of_Casualties', 'Number_of_Vehicles', 'Speed_limit']]

# Plotting the heatmap
plt.figure(figsize=(10, 7))
sns.heatmap(
    numerical_data.corr(),
    annot=True,
    cmap='coolwarm',
    fmt='.2f',
    linewidths=0.5
)

# Adding title
plt.title('Correlation Heatmap', fontsize=16, fontweight='bold')

# Display the plot
plt.tight_layout()
plt.show()
```

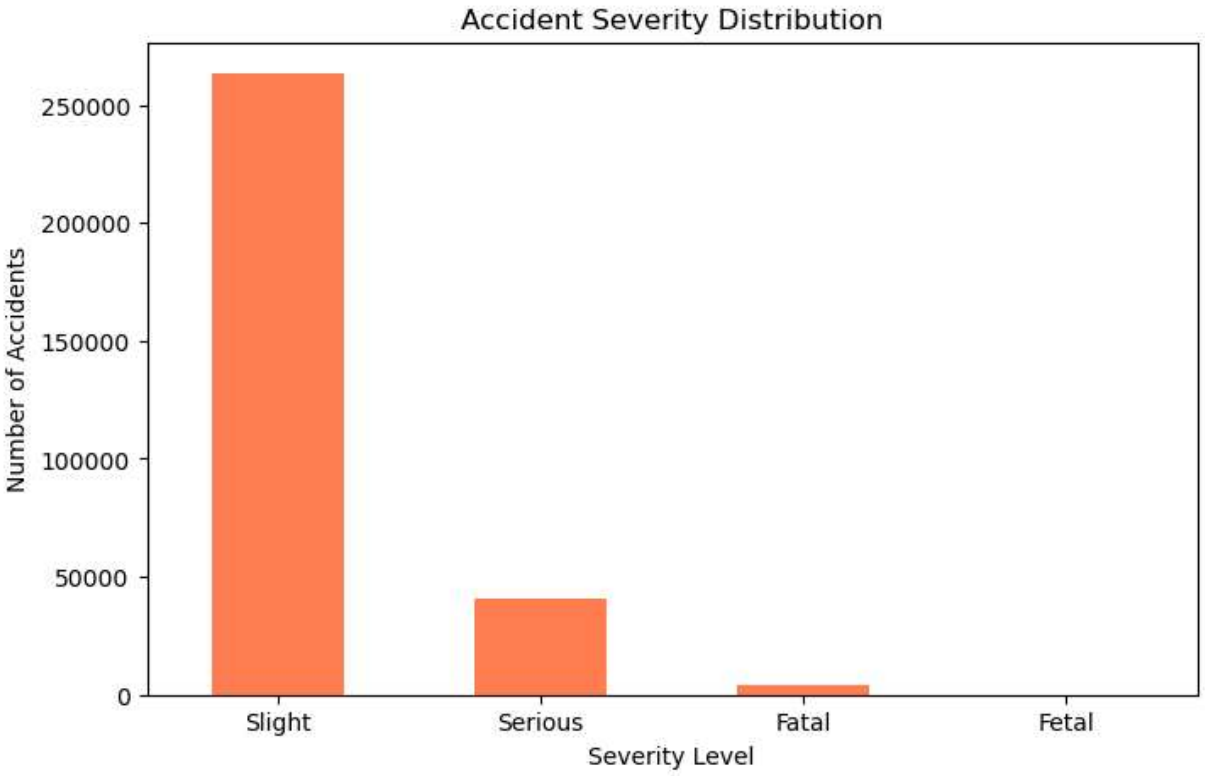


In [76]:
```
data['Hour'] = pd.to_datetime(data['Time'], format='%H:%M', errors='coerce').dt.hou

# Handle missing or invalid hours without using inplace
data['Hour'] = data['Hour'].fillna(-1)

# Group accidents by time of day
hourly_accidents = data['Hour'].value_counts().sort_index()

# Plot accidents by time of day
plt.figure(figsize=(12, 6))
hourly_accidents.plot(kind='bar', color='teal')
plt.title('Accidents by Hour of Day')
plt.xlabel('Hour of Day')
plt.ylabel('Number of Accidents')
plt.xticks(rotation=0)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()

# Additional analysis for peak hours
peak_hour = hourly_accidents.idxmax()
print(f"The peak hour for accidents is: {peak_hour}:00")
```



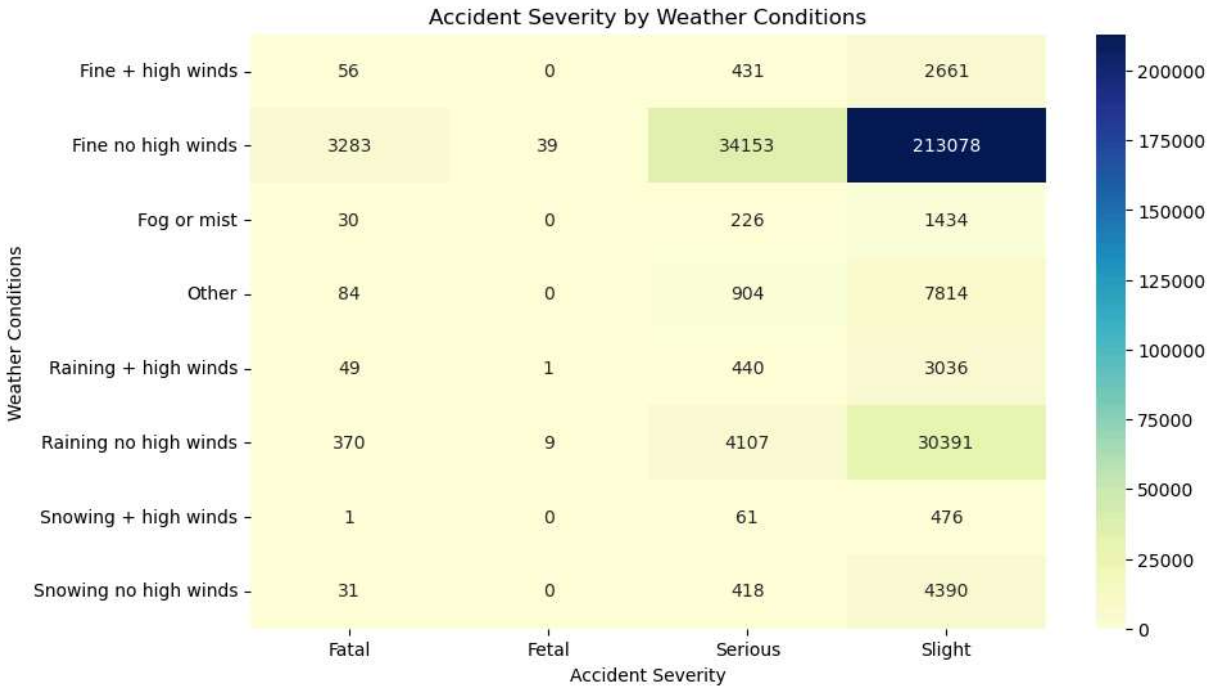

```
The peak hour for accidents is: 17.0:00
```

In [78]:
```python
# Count accidents by severity
severity_counts = data['Accident_Severity'].value_counts()

# Plot the distribution
plt.figure(figsize=(8, 5))
severity_counts.plot(kind='bar', color='coral')
plt.title('Accident Severity Distribution')
plt.xlabel('Severity Level')
plt.ylabel('Number of Accidents')
plt.xticks(rotation=0)
plt.show()

# Print insights
print("Accident severity counts:\n", severity_counts)
```

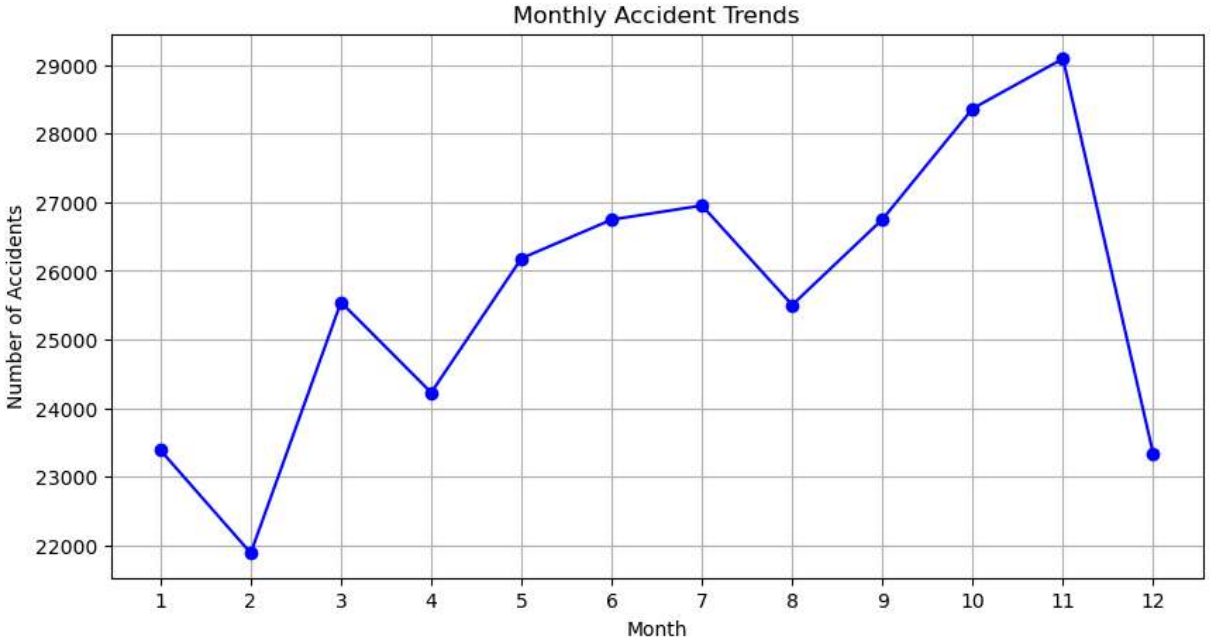



```
Accident severity counts:
 Accident_Severity
Slight     263280
Serious     40740
Fatal        3904
Fetal          49
Name: count, dtype: int64
```

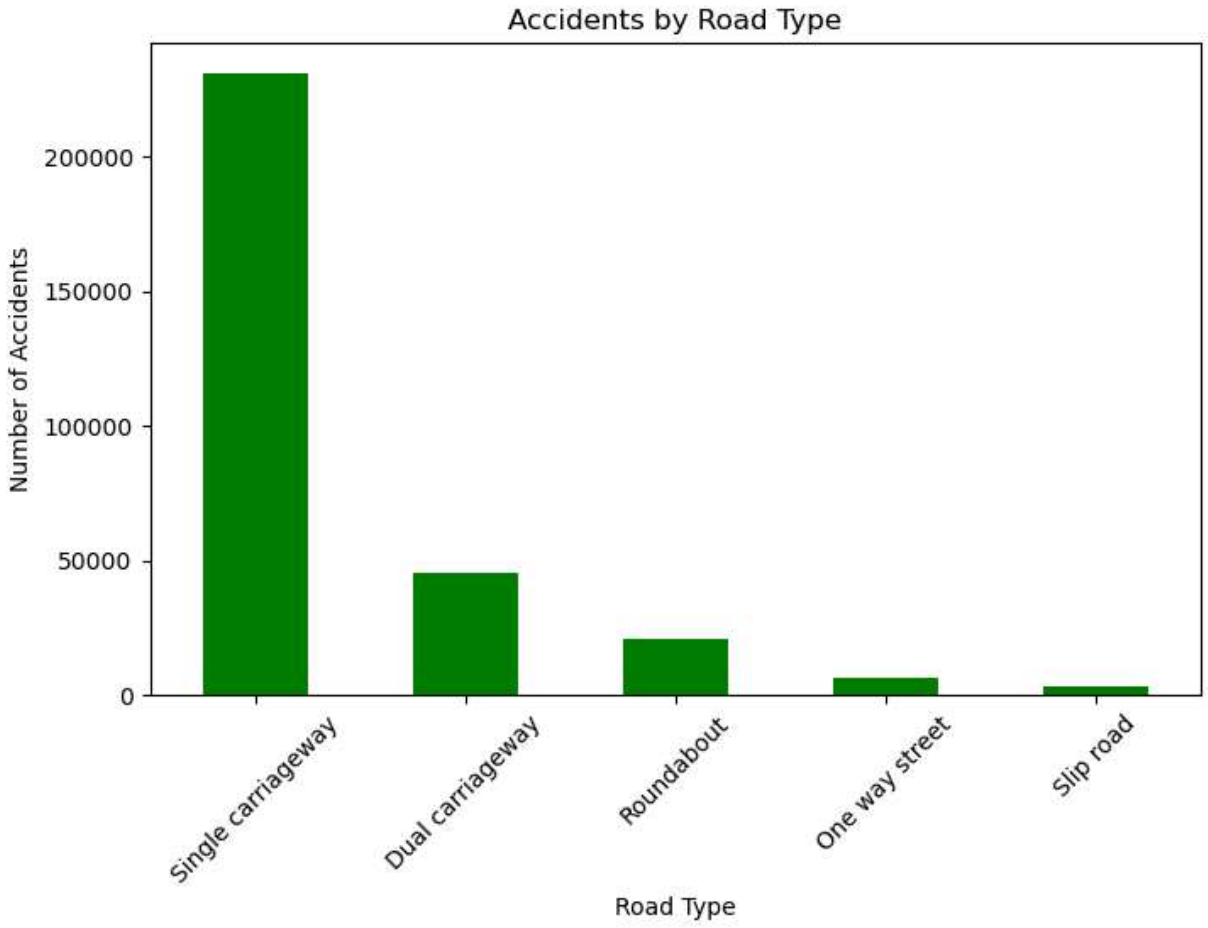
In [82]:
```
# Crosstab for weather vs severity
weather_severity = pd.crosstab(data['Weather_Conditions'], data['Accident_Severity'

# Plot heatmap
plt.figure(figsize=(10, 6))
sns.heatmap(weather_severity, annot=True, cmap='YlGnBu', fmt="d")
plt.title('Accident Severity by Weather Conditions')
plt.xlabel('Accident Severity')
plt.ylabel('Weather Conditions')
plt.show()

# Insights: Which weather condition is most associated with severe accidents?
```



In [89]:
```
# Extract month from 'Accident Date'
data['Month'] = pd.to_datetime(data['Accident Date']).dt.month

# Monthly accident trends
monthly_accidents = data['Month'].value_counts().sort_index()

# Plot
plt.figure(figsize=(10, 5))
monthly_accidents.plot(kind='line', marker='o', color='blue')
plt.title('Monthly Accident Trends')
plt.xlabel('Month')
plt.ylabel('Number of Accidents')
plt.xticks(range(1, 13))
plt.grid()
plt.show()
```



In [91]:
```
road_accidents = data['Road_Type'].value_counts()

# Plot
plt.figure(figsize=(8, 5))
road_accidents.plot(kind='bar', color='green')
plt.title('Accidents by Road Type')
plt.xlabel('Road Type')
plt.ylabel('Number of Accidents')
plt.xticks(rotation=45)
plt.show()

# Insights: Which road type has the highest accident frequency?
```


Accidents by Road Type
Number of Accidents
200000
150000
100000
50000
0
Single carriageway
Dual carriageway
Roundabout
One way street
Slip road
Road Type

In [ ]: