In [1]:
```
import pandas as pd
import numpy as np
import re
import math
```

In [2]:
```
import seaborn as sns
import matplotlib.pyplot as plt
import plotly.express as px
import plotly.graph_objects as go
import plotly.io as pio
from plotly.subplots import make_subplots
%matplotlib inline
```

In [3]:
```
!pip install -U kaleido
```

```
Collecting kaleido
  Using cached kaleido-0.2.1-py2.py3-none-win_amd64.whl (65.9 MB)
Installing collected packages: kaleido
Successfully installed kaleido-0.2.1
```

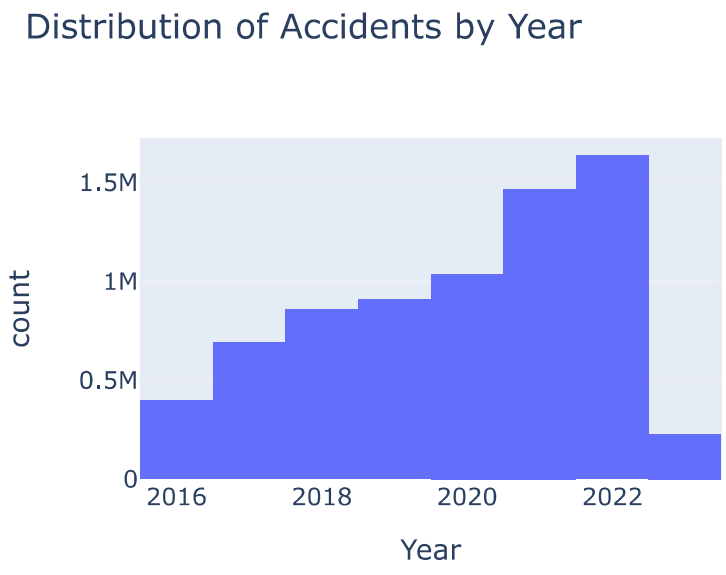
In [4]:
```
from scipy.stats import spearmanr, chi2_contingency, pointbiserialr
```

In this notebook, we analyse the US Traffic Accident dataset to derive insights and select features for predictive models.

In [5]:
```
%%time
data = pd.read_csv("data/US_Accidents_March23_Clean.csv")
data.shape
```

```
CPU times: total: 35.1 s
Wall time: 40.3 s
```
Out[5]:
```
(7223677, 42)
```

In [6]:
```
%%time
fig = px.histogram(data, x='Year', title='Distribution of Accidents by Year')
fig.update_layout(width=450, height=350)
fig.show(config={'staticPlot': True})
```



```
CPU times: total: 3.56 s
Wall time: 4.57 s
```

In [7]:
```
print(f'Portion of data for 2021 to 2022: {data[(data["Year"] == 2021) | (data["Yea
print(f'Portion of data for oher years: {data[(data["Year"] != 2021) & (data["Year'
```

```
Portion of data for 2021 to 2022: 0.43%
Portion of data for oher years: 0.57%
```

The dataset we have contains over 7 millions rows and has a size of 1.5Go (or more). In this analysis and beyond, we are focusing solely on the years 2021 and 2022 for our analysis to ensure that our insights and predictive models are based on the most recent and relevant data available.

- Recent and Relevant Data: The years 2021 and 2022 would be more relevant to leverage the most current insights into what influences impacting accident severity.
- Higher Data Volume: Despite comprising a smaller portion of the dataset (43%), data from the years 2021 and 2022 offer a substantial amount of recorded accidents, ensuring robust analysis and modeling.
- Accuracy in Predictions: By analyzing recent years, we aim to produce predictive models that accurately reflect present-day accident trends and conditions, enhancing the reliability of our forecasts.
- Resource Optimization: Prioritizing these years optimizes our resources (less data to process) by concentrating efforts on data that is more likely to yield actionable insights.

In [8]:
```
data = data[(data["Year"] == 2021) | (data["Year"] == 2022)].copy()
data.shape
```

Out[8]:
```
(3103674, 42)
```

The dataset we are dealing with is highly imbalanced. Severity of level 2 make up (89%) of the data. We understand that nearly all of the reported accidents in the US were of severity 2. However, for the purpose of analysis and modeling, we may downsample to make the dataset balanced as we are more interested in studying factors for accident severity.

In [9]:
```
data["Severity"].value_counts()
```

Out[9]:
```
2    2789545
3     213674
4      64398
1      36057
Name: Severity, dtype: int64
```

In [10]:
```
data["Severity"].value_counts() / len(data)
```

Out[10]:
```
2    0.898788
3    0.068846
4    0.020749
1    0.011618
Name: Severity, dtype: float64
```

In [11]:
```
downsampled_count = int(data["Severity"].value_counts().sort_values(ascending=False
downsampled_count
```

Out[11]:
```
331194
```

In [12]:
```
df_majority_downsampled = data[data["Severity"] == 2].sample(n=downsampled_count, r
df_rest = data[data["Severity"] != 2]
df_balanced = pd.concat([df_majority_downsampled, df_rest])
```

In [13]:
```
df_balanced["Severity"].value_counts() / len(df_balanced)
```

Out[13]:
```
2    0.513222
3    0.331112
4    0.099792
1    0.055874
Name: Severity, dtype: float64
```

In [14]:
```
df_balanced.shape
```

Out[14]:
```
(645323, 42)
```

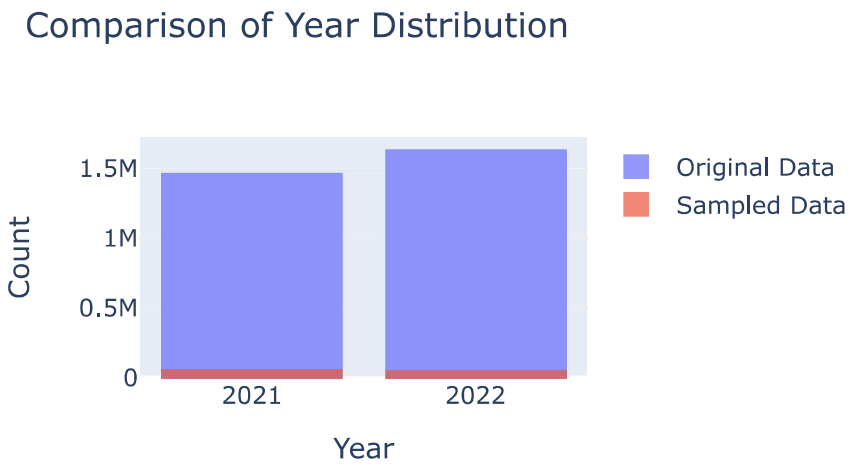
We are down to a bit more than half a million rows in the dataset which is still a bit much considering our resources. We may downsample again while maintain the distribution of keys variables.

In [15]:
```
%%time
category_year_proportions = df_balanced.groupby(['Severity', 'Year']).size() / len(
sample_size = 0.8
# Sample based on the proportions of each Severity and Year combination
df = df_balanced.groupby(['Severity', 'Year'], group_keys=False).apply(
    lambda x: x.sample(frac=sample_size * category_year_proportions[x.name], random
).reset_index(drop=True)

df.shape
```

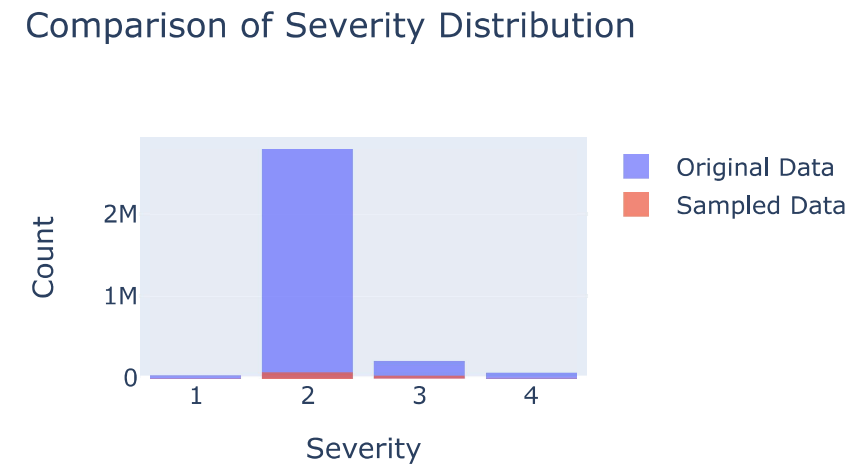
Out[15]:
```
CPU times: total: 953 ms
Wall time: 1.03 s
(103741, 42)
```

We make sure the distribution is the same

In [16]:
```
fig = go.Figure()
fig.add_trace(go.Histogram(x=data['Year'], opacity=0.7, name='Original Data'))
fig.add_trace(go.Histogram(x=df['Year'], opacity=0.7, name='Sampled Data'))
fig.update_layout(
    height=300, width=450,
    title='Comparison of Year Distribution',
    xaxis_title='Year',
    yaxis_title='Count',
    barmode='overlay',
    bargap=0.1,
    bargroupgap=0.1,
    xaxis=dict(
        tickmode='linear',  tick0=min(data['Year']), dtick=1
    )
)
# Show the plot
fig.show(config={'staticPlot': True})
```



In [17]:
```
fig = go.Figure()
fig.add_trace(go.Histogram(x=data['Severity'], opacity=0.7, name='Original Data'))
fig.add_trace(go.Histogram(x=df['Severity'], opacity=0.7, name='Sampled Data'))
fig.update_layout(
    height=300, width=450,
    title='Comparison of Severity Distribution',
    xaxis_title='Severity',
    yaxis_title='Count',
    barmode='overlay',
    bargap=0.1,
    bargroupgap=0.1,
    xaxis=dict(
        tickmode='linear',  tick0=min(data['Severity']), dtick=1
    )
)
# Show the plot
fig.show(config={'staticPlot': True})
```



In [18]:
```
del(data)
del(df_balanced)
```

In [19]:
```
%%time
df["Start_Time"] = pd.to_datetime(df["Start_Time"])
```

```
df["End_Time"] = pd.to_datetime(df["End_Time"])
df["Date"] = pd.to_datetime(df["Date"])
```

```
CPU times: total: 250 ms
Wall time: 334 ms
```

In [20]:
```
df = df.drop(columns=["Start_Time", "End_Time", "Date"])
```

Exploratory Data Analysis

In [21]:
```
numerical_cols = df.select_dtypes(include=['number']).columns.tolist()
boolean_cols = df.select_dtypes(include=['bool']).columns.tolist()
categorical_cols = df.select_dtypes(include=['object','category']).columns.tolist()
```

1. Univariate Analysis

Numerical and ordinal categorical variables

- Time of Day: The distribution of accidents by hour shows two peaks corresponding to morning and evening rush hours, with a significant drop during early morning hours (12-5 AM). These periods correlate with traffic density.
- Day of Week: Accidents are more frequent on weekdays, especially on Fridays. This could indicate a lower traffic density on the weekends as some people stay home and less people go out during the same hours.
- Monthly Trends: There is a clear seasonality effect with the highest number of accidents in winter months, suggesting that weather conditions might play a significant role.
- Yearly Trends: The number of accidents increases each year, potentially indicating either an increase in traffic volume or improved data collection over time.
- Clear Conditions: Accidents occur frequently in clear weather, likely due to the higher number of vehicles on the road under normal conditions (normal temperature, low humidity, normal but higher pressure, high visibility, lower wind speed, and low-to-no precipitation).
- Humidity levels: More accident occured at higher humidity levels. This might correlate with poor visibility and wet road conditions, which could increase accident risk.

In [22]:
```
len(numerical_cols)
```

Out[22]:
```
16
```

In [23]:
```
df[numerical_cols].info()
```

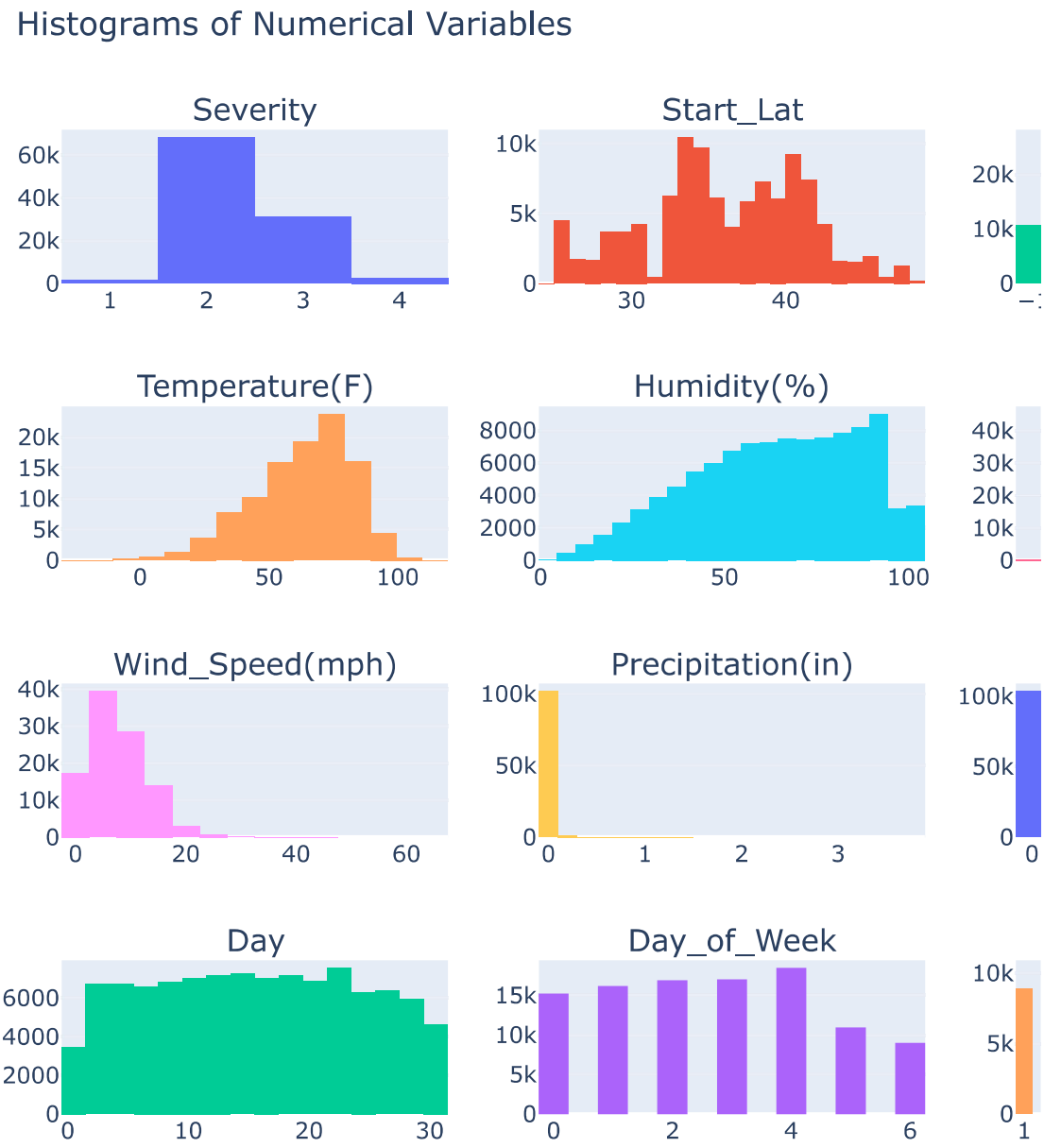
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 103741 entries, 0 to 103740
Data columns (total 16 columns):
 #   Column             Non-Null Count   Dtype  
---  ------             --------------   -----  
 0   Severity           103741 non-null  int64  
 1   Start_Lat          103741 non-null  float64
 2   Start_Lng          103741 non-null  float64
 3   Distance(mi)       103741 non-null  float64
 4   Temperature(F)     103741 non-null  float64
 5   Humidity(%)        103741 non-null  float64
 6   Pressure(in)       103741 non-null  float64
 7   Visibility(mi)     103741 non-null  float64
 8   Wind_Speed(mph)    103741 non-null  float64
 9   Precipitation(in)  103741 non-null  float64
 10  Duration(min)      103741 non-null  float64
 11  Hour               103741 non-null  int64  
 12  Day                103741 non-null  int64  
 13  Day_of_Week        103741 non-null  int64  
 14  Month              103741 non-null  int64  
 15  Year               103741 non-null  int64  
dtypes: float64(10), int64(6)
memory usage: 12.7 MB
```

In [24]:
```
df[numerical_cols].describe()
```

Out[24]:

	Severity	Start_Lat	Start_Lng	Distance(mi)	Temperature(F)	Humidity(%)
count	103741.000000	103741.000000	103741.000000	103741.000000	103741.000000	103741.000000
mean	2.336338	36.071940	-92.824166	0.716211	62.909139	64.115065
std	0.552022	5.169246	16.852608	2.112367	19.120403	22.600117
min	1.000000	24.603227	-124.482434	0.000000	-28.000000	1.000000
25%	2.000000	32.940380	-111.918968	0.000000	51.000000	48.000000
50%	2.000000	35.843700	-86.052238	0.095000	66.000000	66.000000
75%	3.000000	40.213383	-80.208626	0.659000	77.000000	83.000000
max	4.000000	48.968456	-68.741722	210.080002	115.000000	100.000000

In [25]:
```
%%time
fig = make_subplots(rows=4, cols=4, subplot_titles=numerical_cols)
for i, col in enumerate(numerical_cols):
    row = i // 4 + 1
    col_pos = i % 4 + 1
    fig.add_trace(
        go.Histogram(x=df[col], nbinsx=25 if col != 'Severity' else 4, showlegend=F
        row=row, col=col_pos
    )

# Update layout
fig.update_layout(height=700, width=1120, title_text="Histograms of Numerical Varia
fig.update_xaxes(tickvals=[1, 2, 3, 4], row=1, col=1)
fig.update_xaxes(tickvals=list(range(1, 13)), row=4, col=3)

# Show plot
fig.show(config={'staticPlot': True})
```




```
CPU times: total: 1.5 s
Wall time: 1.58 s
```

Boolean variables

- Most boolean variables (`Amenity` , `Bump` , `Give_Way` , `No_Exit` , `Railway` , `Roundabout` , `Station` , `Stop` , `Traffic_Calming`) are predominantly False (95% or more), indicating that these features are not present in the majority of accidents. Their abscence, however, could play a role in the severity of an accident.
- Hight-Risk Areas: Areas with crossing, junctions, and traffic signals see significant accident counts. As for the other boolean varibles, the values for these columns are also predominantly False (85% or more) tho more balanced.
- Night-time accidents: Most accidents (69.7%) occured during the day, thus matching traffic volume trends. In addition, about 31% of accidents occur at night, highlighting a significant proportion of accidents happening in low-light conditions.

- Highway Accidents: Although most accidents happened on local roads (72%), accidents on highways account for a notable portion (28%).

In [26]:
```
len(boolean_cols)
```

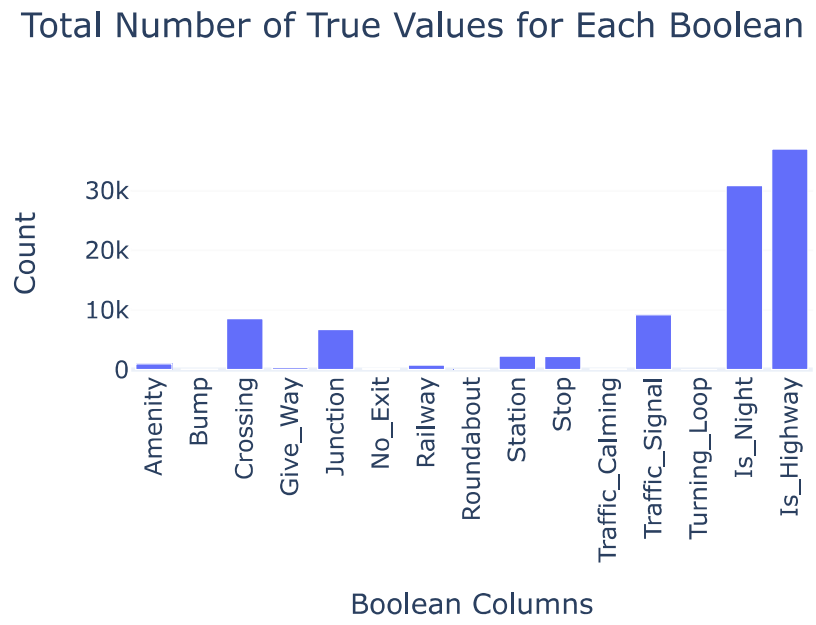
Out[26]:
```
15
```

In [27]:
```
df[boolean_cols].info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 103741 entries, 0 to 103740
Data columns (total 15 columns):
 #   Column           Non-Null Count   Dtype
---  ------           --------------   -----
 0   Amenity          103741 non-null  bool
 1   Bump             103741 non-null  bool
 2   Crossing         103741 non-null  bool
 3   Give_Way         103741 non-null  bool
 4   Junction         103741 non-null  bool
 5   No_Exit          103741 non-null  bool
 6   Railway          103741 non-null  bool
 7   Roundabout       103741 non-null  bool
 8   Station          103741 non-null  bool
 9   Stop             103741 non-null  bool
 10  Traffic_Calming  103741 non-null  bool
 11  Traffic_Signal   103741 non-null  bool
 12  Turning_Loop     103741 non-null  bool
 13  Is_Night         103741 non-null  bool
 14  Is_Highway       103741 non-null  bool
dtypes: bool(15)
memory usage: 1.5 MB
```

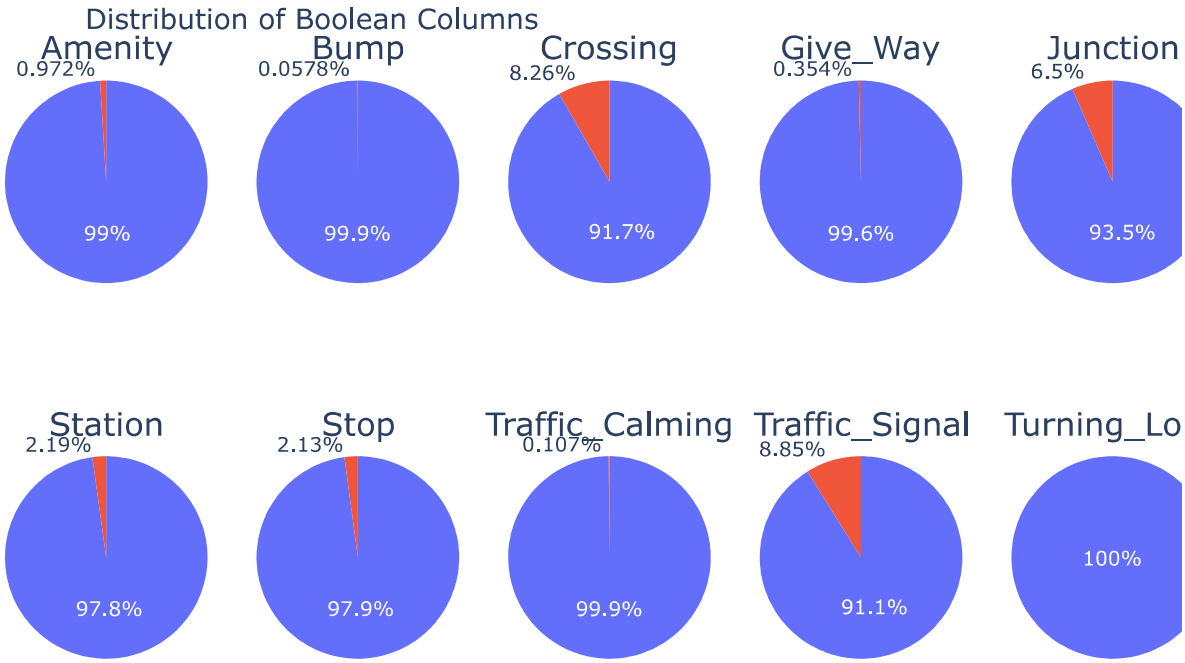
In [28]:
```
true_counts = df[boolean_cols].sum()
```

In [29]:
```
fig = go.Figure(data=go.Bar(x=true_counts.index, y=true_counts.values))
fig.update_layout(
    height=350, width=500,
    xaxis_tickangle=-90,
    xaxis_title='Boolean Columns',
    yaxis_title='Count',
    title='Total Number of True Values for Each Boolean Column',
    template='plotly_white',
)
fig.show(config={'staticPlot': True})
```



In [30]:

```
num_plots = len(boolean_cols)
num_cols = 8
num_rows = math.ceil(num_plots / num_cols)

# Create Plotly figure with subplots
fig = make_subplots(rows=num_rows, cols=num_cols, subplot_titles=boolean_cols,
                    specs=[[{'type':'pie'}]*num_cols]*num_rows)
# Populate subplots with pie charts
for i, column in enumerate(boolean_cols):
    row = i // num_cols + 1  # Plotly subplots start from row 1
    col = i % num_cols + 1   # Plotly subplots start from col 1
    counts = df[column].value_counts()
    fig.add_trace(
        go.Pie(labels=counts.index, values=counts, textinfo='percent', sort=False),
        row=row, col=col
    )
    fig.update_layout(
        title=f"{column}",
        font=dict(size=10),
        margin=dict(l=10, r=10, t=40, b=10),  # Adjust margins for better layout
        showlegend=False
    )
# Update layout and show figure
fig.update_layout(
    title='Distribution of Boolean Columns',
    height=350, width=1000,
    template='plotly_white',
)

fig.show(config={'staticPlot': True})
```



Categorical variables

- Weather Conditions: Accidents are most frequent under clear and cloudy conditions.
- Location Factor: Cities and States with higher accident counts often coincide with high population density, extensive road networks, and potentially varying driving conditions due to local climate and geography.

Some variables may not be very useful as they are so we would use their transformed version or new variables extracted from them.

In [31]:
```
categorical_cols.remove("Street")
categorical_cols.remove("Weather_Condition")
categorical_cols.remove("ID")
```

In [32]:
```
len(categorical_cols)
```

Out[32]:
```
5
```

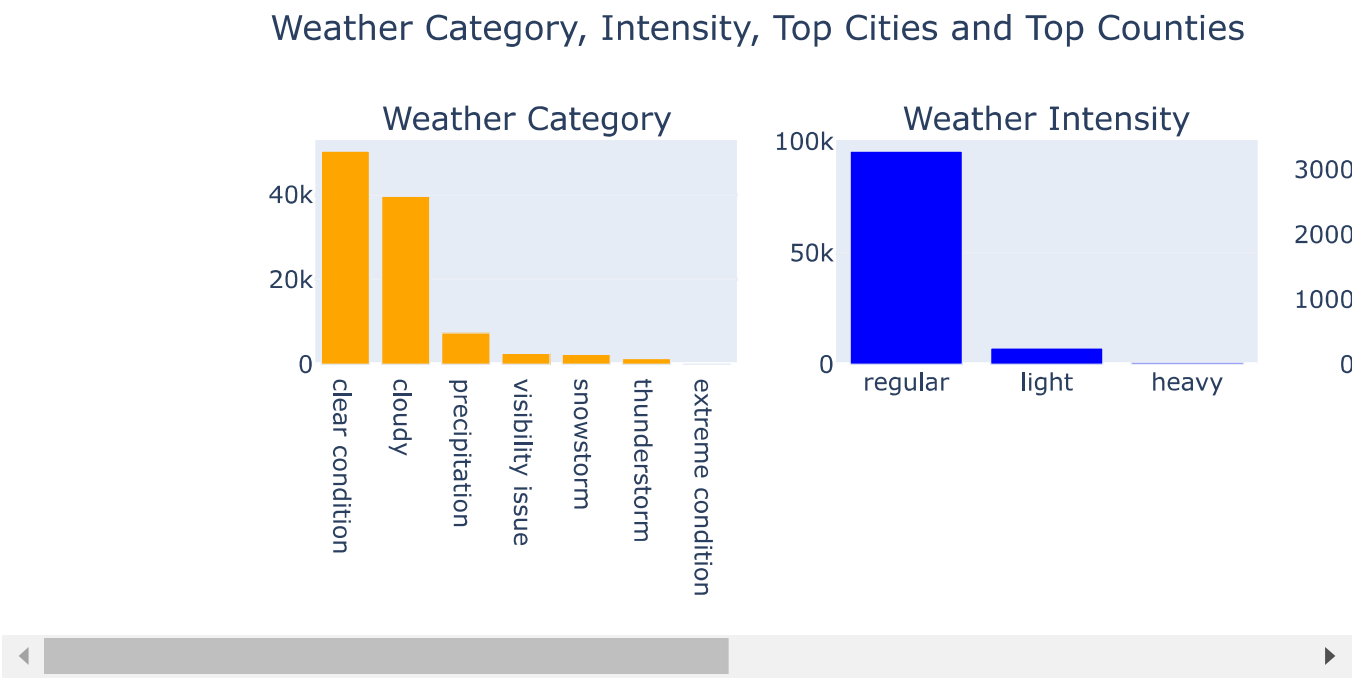
In [33]:
```
df[categorical_cols].info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 103741 entries, 0 to 103740
Data columns (total 5 columns):
 #   Column             Non-Null Count   Dtype 
---  ------             --------------   ----- 
 0   City               103741 non-null  object
 1   County             103741 non-null  object
 2   State              103741 non-null  object
 3   Weather_Category   103741 non-null  object
 4   Weather_Intensity  103741 non-null  object
dtypes: object(5)
memory usage: 4.0+ MB
```

In [34]:
```
print("Unique values counts")
for col in categorical_cols:
    print(f"{col}: {df[col].unique().shape[0]}")
```

```
Unique values counts
City: 6024
County: 1213
State: 48
Weather_Category: 7
Weather_Intensity: 3
```

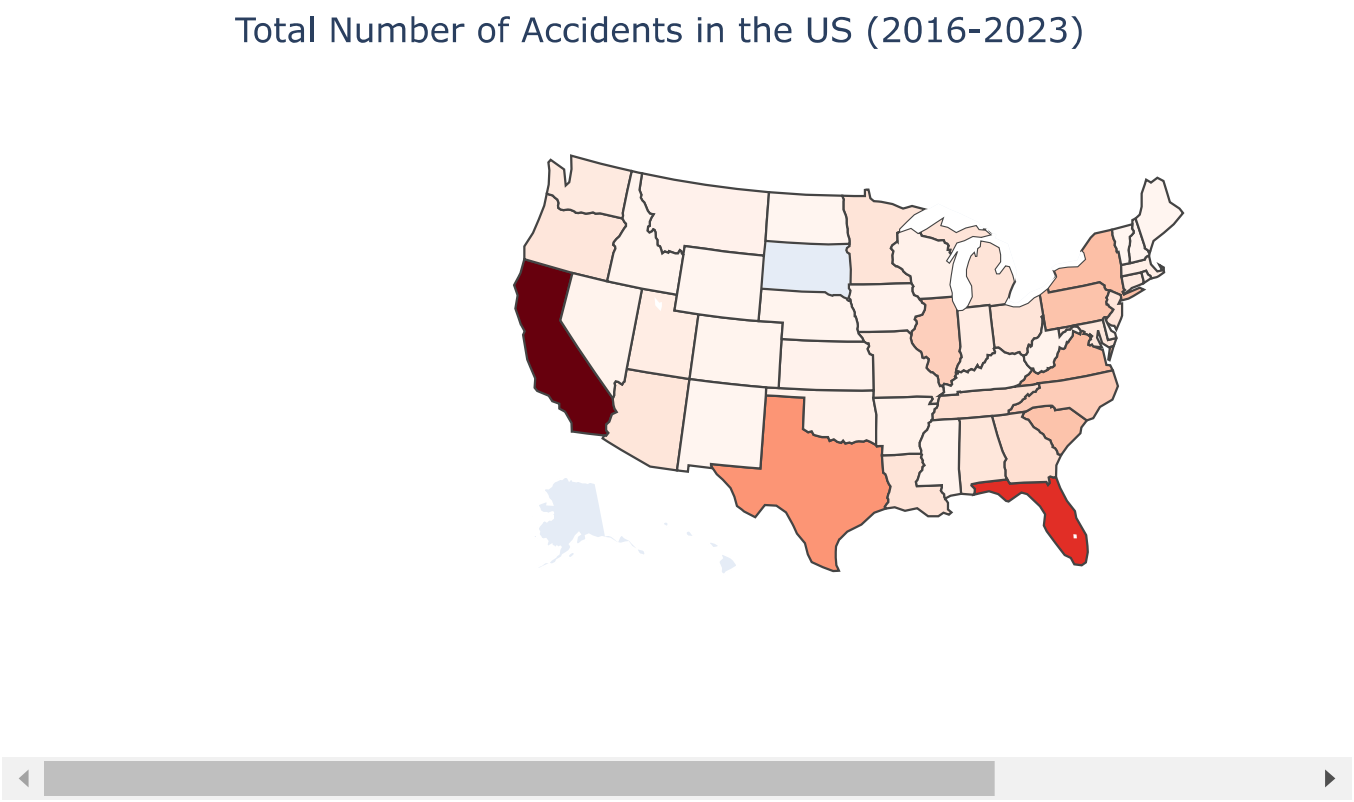
In [35]:
```python
# Create a subplot for Weather Category, Weather Intensity, and City
fig = make_subplots(rows=1, cols=4, subplot_titles=['Weather Category', 'Weather Ir

# Plot Weather Category (bar plot)
weather_cat_counts = df['Weather_Category'].value_counts()
fig.add_trace(go.Bar(x=weather_cat_counts.index, y=weather_cat_counts.values, marke

# Plot Weather Intensity (bar plot)
weather_intensity_counts = df['Weather_Intensity'].value_counts()
fig.add_trace(go.Bar(x=weather_intensity_counts.index, y=weather_intensity_counts.v

# Plot City (bar plot)
city_counts = df['City'].value_counts().nlargest(15)
fig.add_trace(go.Bar(x=city_counts.index, y=city_counts.values, marker_color='greer

# Plot County (bar plot)
county_counts = df['County'].value_counts().nlargest(15)
fig.add_trace(go.Bar(x=county_counts.index, y=county_counts.values, marker_color='p

fig.update_layout(title='Weather Category, Intensity, Top Cities and Top Counties',
                  height=340, width=1150, showlegend=False)

fig.show(config={'staticPlot': True})
```



- Most accidents occur under regular weather intensity.
- Cities with higher populations and more traffic density tend to have more accidents.

In [36]:
```python
# Plot State (choropleth map)
state_counts = df['State'].value_counts().reset_index()
state_counts.columns = ['State', 'Counts']

# Create choropleth map for State
```

```
fig = go.Figure(data=go.Choropleth(
    locations=state_counts['State'],
    z=state_counts['Counts'],
    locationmode='USA-states',
    colorscale='Reds',
    colorbar_title='Number of Accidents'
))

fig.update_layout(
    title_text='Total Number of Accidents in the US (2016-2023)',
    geo=dict(scope='usa', projection_type='albers usa'),
    height=400, width=800,
    showlegend=True,
    barmode='group',
)

# Show the plot
fig.show(config={'staticPlot': True})
```



- States with higher populations and extensive road networks like Californaia tend to report more accidents.

2. Multivariate Analysis

Numerical variables

In [37]:
```
numerical_cols.remove("Month")
numerical_cols.remove("Day_of_Week")
```

In [38]:
```
coordinate_vars = ["Start_Lng", "Start_Lat"]
```

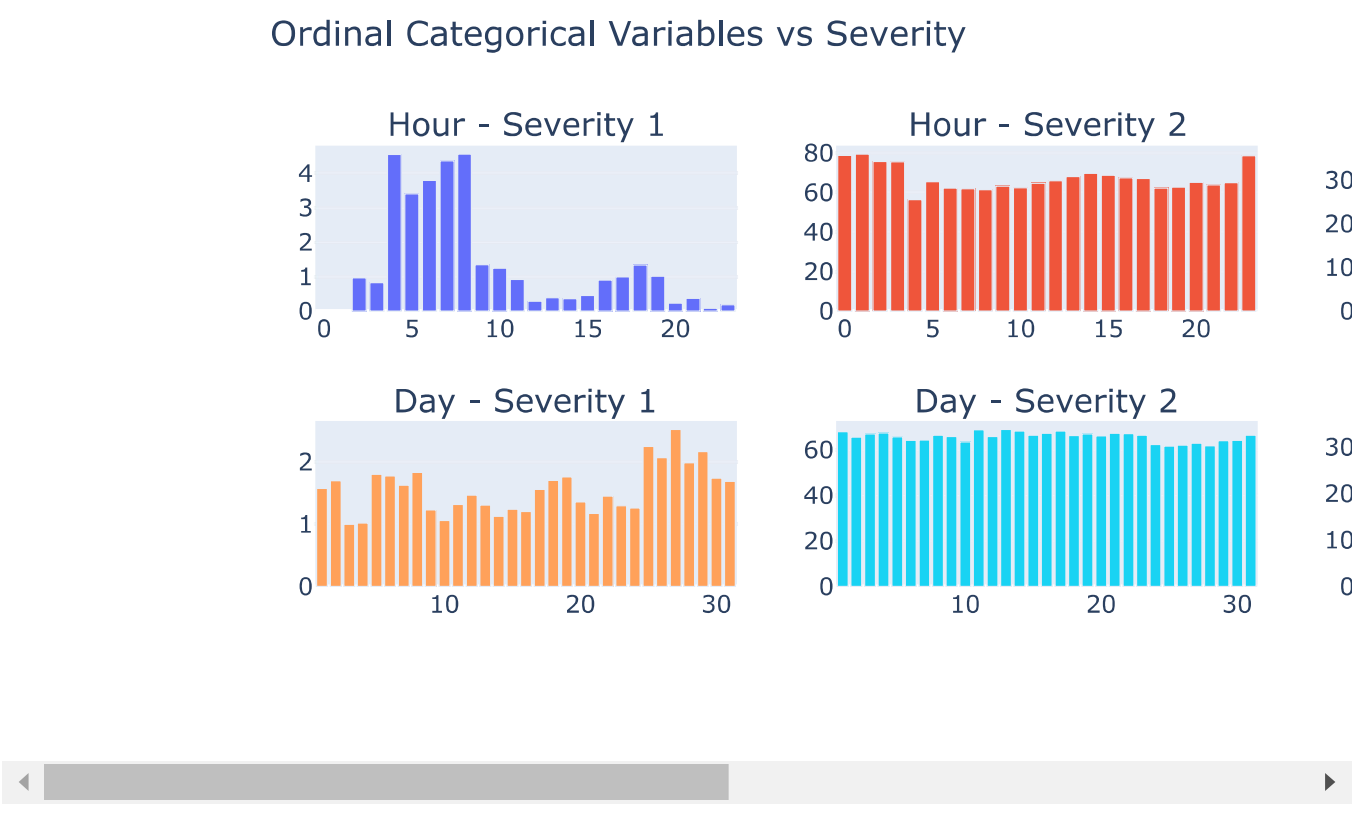
In [39]:
```
#%%time
#fig = px.scatter(df, x='Start_Lng', y='Start_Lat', color='Severity', opacity=0.5)
```

```
#fig.update_layout(title='Start_Lat vs Start_Lng', height=450, width=780)
#fig.show(config={'staticPlot': True})
```

- A scatterplot of coordinates (`Start_Lat` and `Start_Lng`) show that the most severe accidents occured in the most populated cities--mostly in the Eastern side of the US. We can also notice accidents along interstate highways.

In [40]:
```
time_vars = ['Hour', 'Day']
```

In [41]:
```
severities = sorted(df['Severity'].unique())
subplot_titles = [f'{time_vars[i]} - Severity {severities[j]}' for i in range(len(t

# Plot time variables
fig = make_subplots(rows=2, cols=4, subplot_titles=subplot_titles, shared_yaxes=Fal

for i, var in enumerate(time_vars):
    var_counts = df[var].value_counts().sort_index()
    for j, severity in enumerate(severities):
        df_grouped = df[df['Severity'] == severity][var].value_counts().sort_index(
        df_grouped = df_grouped.div(var_counts, fill_value=float('NaN'))
        fig.add_trace(go.Bar(x=df_grouped.index, y=df_grouped.values * 100, name=f'

fig.update_layout(height=400, width=1150, title='Ordinal Categorical Variables vs S
fig.show(config={'staticPlot': True})
```



Time of the Day:

- Severity 1: Most accidents occur early in the morning, peaking at 7 AM, with another, less intense peak around 5 PM.
- Severity 2 and 3: Accidents follow commuting trends, with higher frequencies during rush hours (5 AM-9 AM and 1 PM-7 PM).

- Severity 4: The distribution is more uniform throughout the day, with a slight increase in the later hours.

Day of the Month:

- The distribution of accidents is fairly uniform throughout the month for all severity levels with a slight trend at the end of the month.

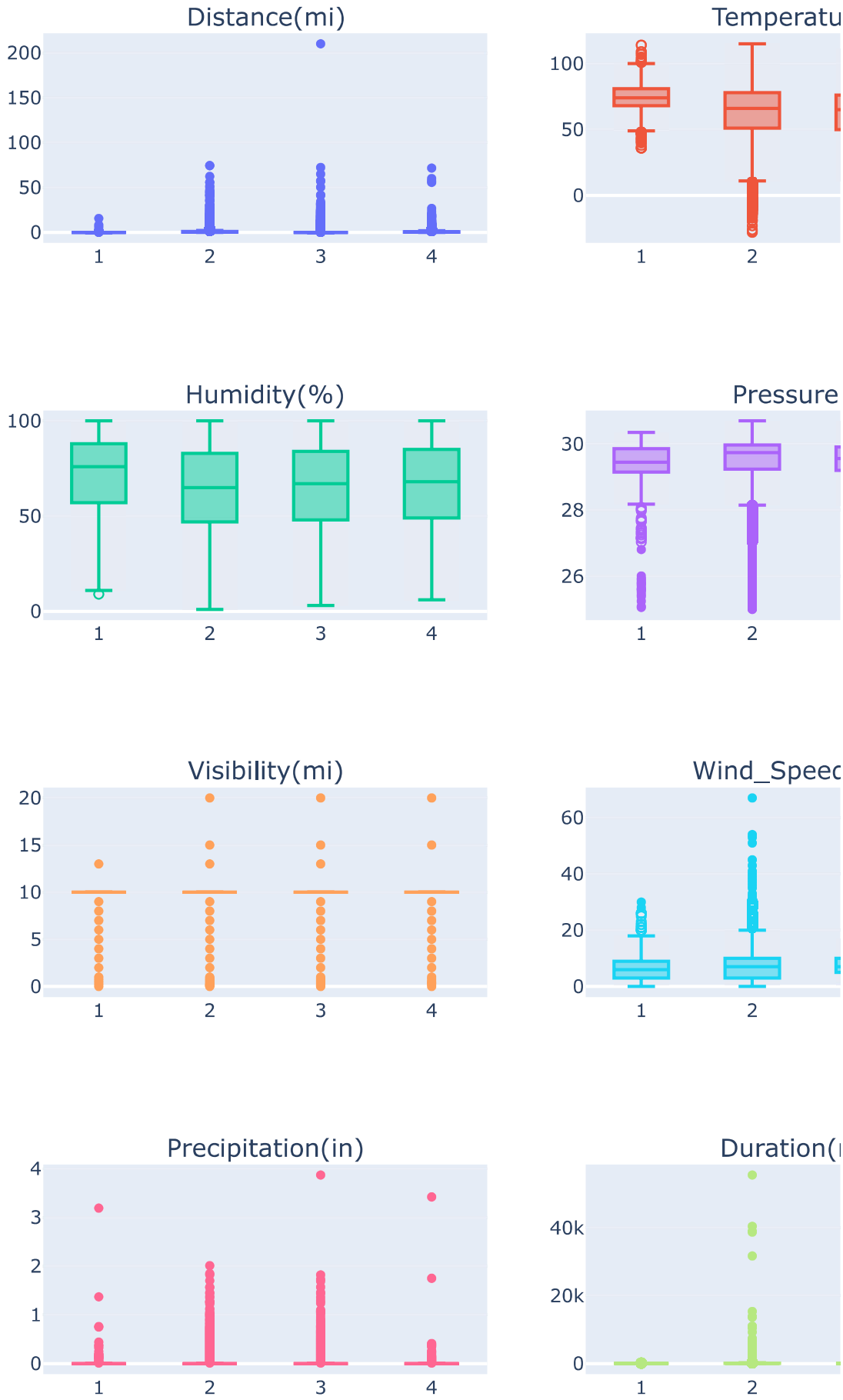
In [42]:
```
# Separate ordinal categorical variables from numerical variables
ordinal_cols = ["Month", "Day_of_Week"]
```

In [43]:
```
num_vars = [var for var in numerical_cols if var not in time_vars and var not in or
num_vars.remove("Year")
num_vars.remove("Severity")
len(num_vars)
```

Out[43]:
```
8
```

In [44]:
```
%%time
fig = make_subplots(rows=4, cols=2, subplot_titles=num_vars)
for i, var in enumerate(num_vars):
    row = i // 2 + 1
    col = i % 2 + 1
    fig.add_trace(go.Box(
        y=df[var], x=df['Severity'],
        name=var,
        boxpoints="suspectedoutliers"
        #points='suspectedoutliers', box_visible=True, meanline_visible=True
    ),
                  row=row, col=col)
    fig.update_layout(height=1050, width=800, title='Numerical Variables vs Severit

fig.show(config={'staticPlot': True})
```

Numerical Variables vs Severity

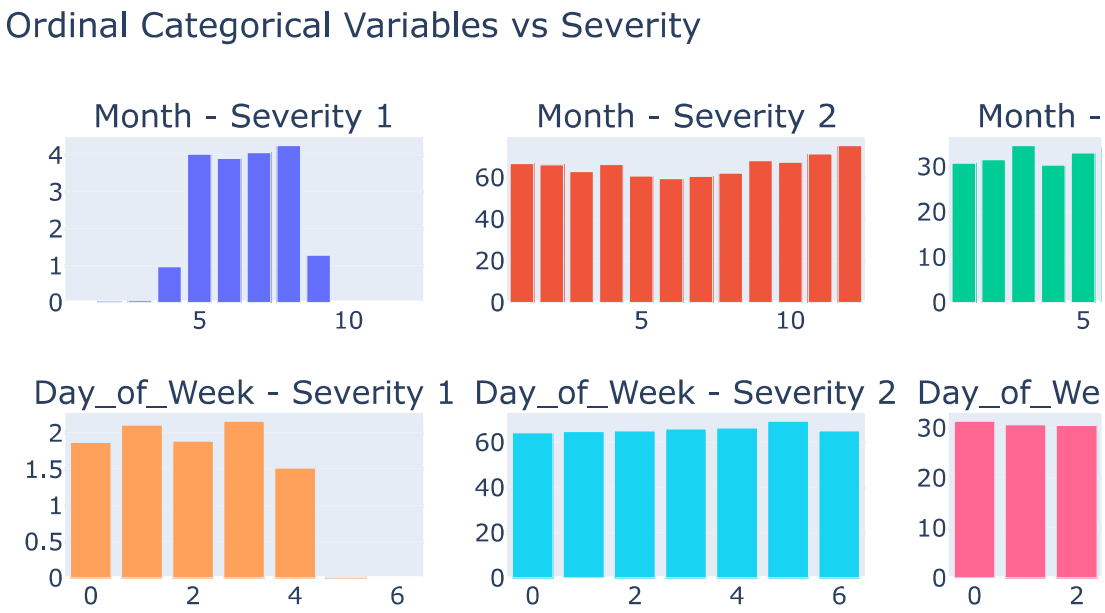


```
CPU times: total: 1.28 s
```

- Distance affected: Overall, the distance slightly increases with severity, indicating more severe accidents may disrupt a longer stretch of road.
- Temperature: Higher severity accidents occur in a wider range of temperatures, but often cooler conditions.
- Humidity and Pressure: Do not show strong differentiation between severity levels.
- Visibility: Generally relatively high for all severities, but slightly lower for more severe accidents.
- Wind Speed: Higher wind speeds are associated with more severe accidents.
- Precipitation: Low across all severities, but slightly higher for more severe accidents.
- Duration: Increases with severity, indicating more severe accidents take longer to resolve.

Ordinal Categorical variables

In [45]:
```
severities = sorted(df['Severity'].unique())
subplot_titles = [f'{ordinal_cols[i]} - Severity {severities[j]}' for i in range(le

# Plot time variables
fig = make_subplots(rows=2, cols=4, subplot_titles=subplot_titles, shared_yaxes=Fal

for i, var in enumerate(ordinal_cols):
    var_counts = df[var].value_counts().sort_index()
    for j, severity in enumerate(severities):
        df_grouped = df[df['Severity'] == severity][var].value_counts().sort_index(
        df_grouped = df_grouped.div(var_counts, fill_value=float('NaN'))
        fig.add_trace(go.Bar(x=df_grouped.index, y=df_grouped.values * 100, name=f'

fig.update_layout(height=400, width=1000, title='Ordinal Categorical Variables vs S
fig.show(config={'staticPlot': True})
```



Month of the year:

- Accidents are relatively evenly distributed across the months for all severity levels.
- There is a slight decrease in accidents of severity 3 during the winter months (November to February) while we notice an increase in accidents of severity 4 in December, likely due to weather or end of the year celebrations. Similarly, accidents of severity 1 peak during summer time.

Day of Week:

- Accidents are relatively evenly distributed across the weekdays.
- Weekdays, especially Fridays, have higher accident rates across all severities due to increased traffic and commuting. Weekends generally see lower accident rates, indicating safer driving conditions.

Categorical variables

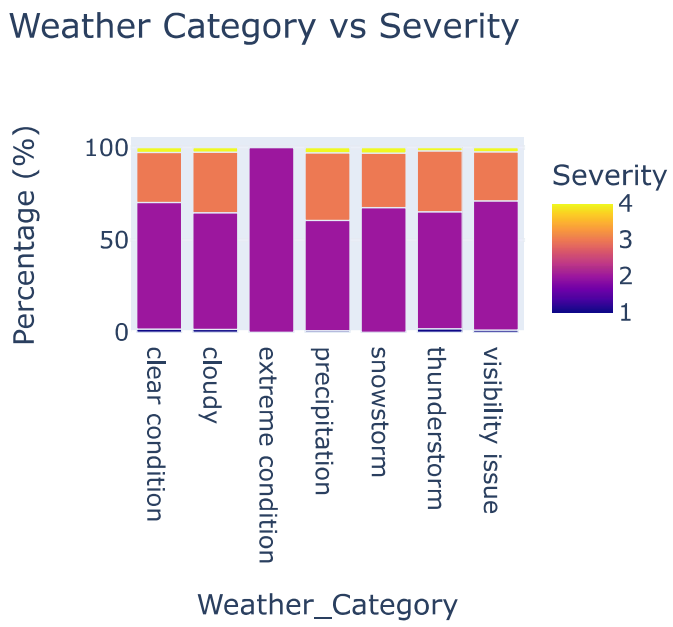
To simplify our analysis, we decided to drop the "City" and "County" variables due to their high cardinality and the lack of additional context such as population size. This makes them less useful for our current scope. By excluding them, we can focus on more immediately actionable and interpretable variables.

In [46]:
```
categorical_cols.remove("City")
categorical_cols.remove("County")
categorical_cols.remove("Weather_Intensity")
```

In [47]:
```
len(categorical_cols)
```

Out[47]:
```
2
```

In [48]:
```
df_weather_grouped = df.groupby(['Weather_Category', 'Severity']).size().reset_inde
df_weather_grouped['Percentage'] = df_weather_grouped.groupby('Weather_Category', g

# Plotting
fig = px.bar(df_weather_grouped, x='Weather_Category', y='Percentage', color='Sever
             category_orders={'Severity': ['1', '2', '3', '4']},
             labels={'Percentage': 'Percentage (%)'},
             title='Weather Category vs Severity')
fig.update_layout(showlegend=True, height=350, width=370)
fig.show(config={'staticPlot': True})
```

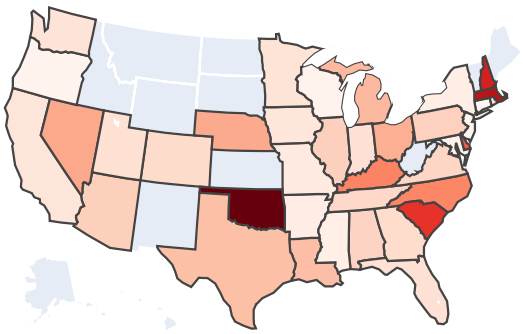


- Severity 1: Generally, all weather conditions contribute to relatively low proportions of severity 1 accidents, indicating that driving conditions may be safer under these circumstances.
- Severity 2: Clear, cloudy, and various adverse weather conditions significantly increase the likelihood of severity 2 accidents, with precipitation, snowstorms, thunderstorms, and visibility issues having the highest impact.
- Severity 3 and 4: While severity 2 dominates across weather categories, severe accidents (severity 3 and 4) are less frequent but still notable under adverse weather conditions such as precipitation, snowstorms, thunderstorms, and visibility issues (smoke, fog, etc.)

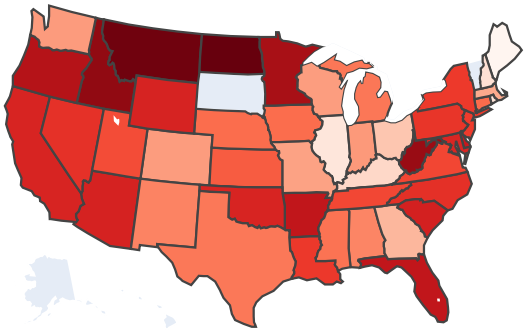
In [49]:
```
severities = sorted(df['Severity'].unique())
# Store counts accross all severity levels
all_severity_state_counts = df["State"].value_counts().sort_index()
# Create a list to hold each choropleth map figure
figs = []
# Loop through each severity level and create a choropleth map
for severity in severities:
    # Filter data for the current severity level
    severity_data = df[df['Severity'] == severity]
    # Count occurrences of each state for the current severity level
    state_counts = severity_data['State'].value_counts()
    state_counts = state_counts.div(all_severity_state_counts).reset_index()
    state_counts.columns = ['State', 'Proportion']
    # Create choropleth map for the current severity level
    fig = go.Figure(data=go.Choropleth(
```

```
        locations=state_counts['State'],
        z=state_counts['Proportion'],
        locationmode='USA-states',
        colorscale='Reds',
        colorbar=dict(title='Proportion'),
        hoverinfo='location+z'
    ))
    # Update layout for the choropleth map
    fig.update_layout(
        height=350, width=700,
        title=f'Proportion of Severity {severity} Accidents by State',
        geo=dict(scope='usa', projection_type='albers usa'),
    )
    # Add the figure to the list
    figs.append(fig)

# Display each choropleth map separately
for fig in figs:
    fig.show(config={'staticPlot': True})
```

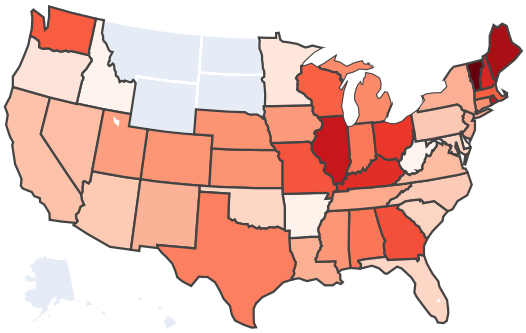
Proportion of Severity 1 Accidents by State



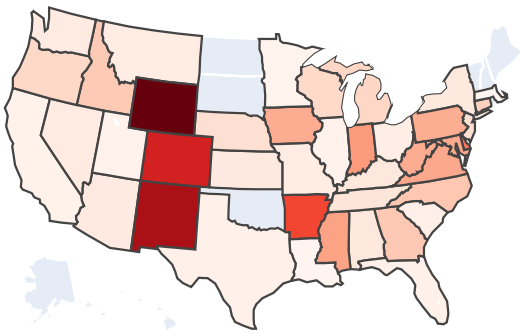
Proportion of Severity 2 Accidents by State



Proportion of Severity 3 Accidents by State



Proportion of Severity 4 Accidents by State



- Generally, severity 1 accidents show less variation across states, suggesting that factors influencing minor accidents might be more uniformly managed or influenced.
- States vary widely in terms of severity 2 accident proportions, with some states experiencing significantly higher rates, potentially due to factors like population density, weather conditions, or road infrastructure.
- There is more variability across states for more severe accidents (severity 3 and 4), with some states having higher proportions compared to others.

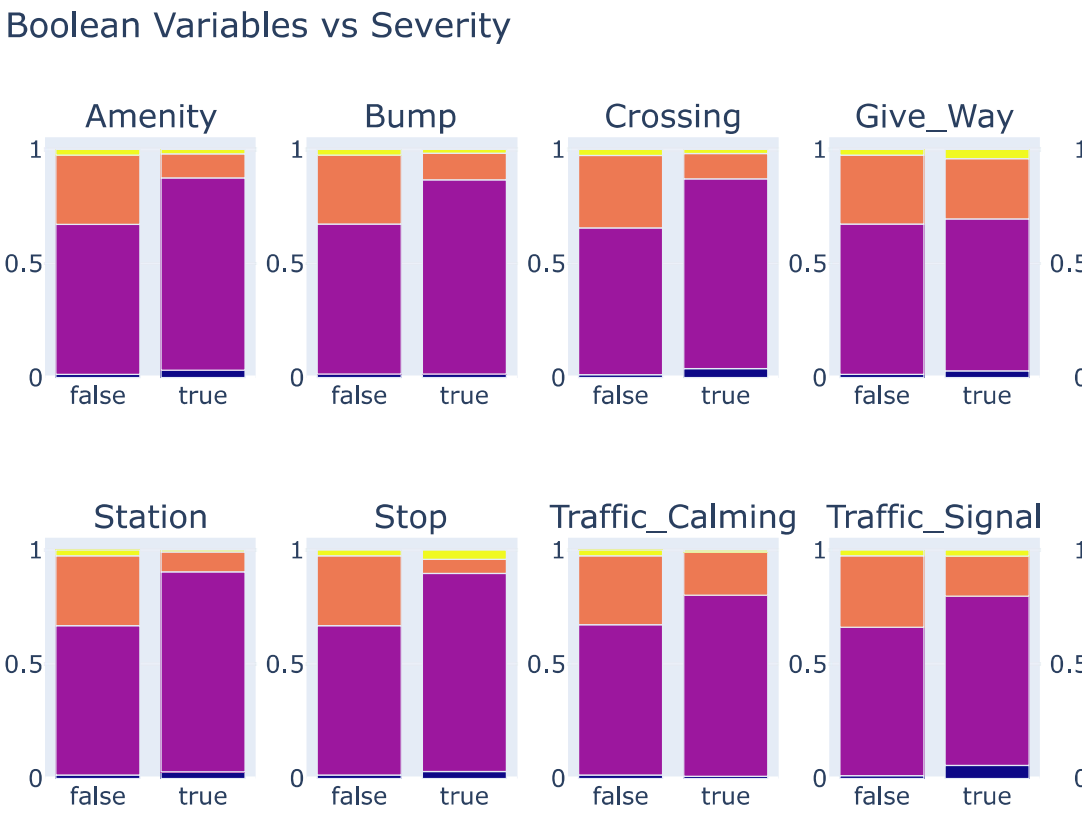
Boolean variables

In [50]:
```
len(boolean_cols)
```

Out[50]:
```
15
```

In [51]:

```
num_plots = len(boolean_cols)
num_cols = 8
num_rows = math.ceil(num_plots / num_cols)

fig = make_subplots(rows=num_rows, cols=num_cols, subplot_titles=boolean_cols)

# Loop through each boolean variable and create a stacked bar plot
for i, var in enumerate(boolean_cols):
    row = i // num_cols + 1
    col = i % num_cols + 1

    # Group by boolean variable and Severity, then calculate percentage
    df_grouped = df.groupby([var, 'Severity']).size().reset_index(name='Count')
    df_grouped['Proportion'] = df_grouped.groupby(var, group_keys=False)['Count'].a

    # Plotting
    fig_bar = px.bar(df_grouped, x=var, y='Proportion', color='Severity', barmode='
                     category_orders={'Severity': ['1', '2', '3', '4']},
                     labels={'Proportion': 'Proportion (%)'},
                     title=f'{var} vs Severity').data

    # Add traces to subplot
    for trace in fig_bar:
        fig.add_trace(trace, row=row, col=col)

# Update layout
fig.update_layout(
    title='Boolean Variables vs Severity',
    height=500, width=1200,
    showlegend=True
)

fig.show(config={'staticPlot': True})
```



- Strong Correlation with Severity 2: Amenities, bumps, crossings, no exit, stations, stop signs, and traffic calming measures.
- Moderate Correlation with Severity 2: Railways, traffic signals, and nighttime conditions.
- High Correlation with Severity 3 and 4: Highways, junctions, and nighttime conditions.

3. Correlation Analysis

In [52]:
```
# Store results
correlations = {'Variable': [], 'Correlation': [], 'Type': []}
```

In [53]:
```
# Spearman's Rank Correlation for numerical and time-related variables
for var in num_vars + time_vars:
    corr, _ = spearmanr(df[var], df['Severity'])
    correlations['Variable'].append(var)
    correlations['Correlation'].append(corr)
    correlations['Type'].append('Numerical/Time')
```

In [54]:
```
# Chi-Square and Cramér's V for categorical variables
for var in categorical_cols:
    contingency_table = pd.crosstab(df[var], df['Severity'])
    chi2, p, dof, expected = chi2_contingency(contingency_table)
    cramers_v = np.sqrt(chi2 / (df.shape[0] * (min(contingency_table.shape) - 1)))
    correlations['Variable'].append(var)
    correlations['Correlation'].append(cramers_v)
    correlations['Type'].append('Categorical')
```

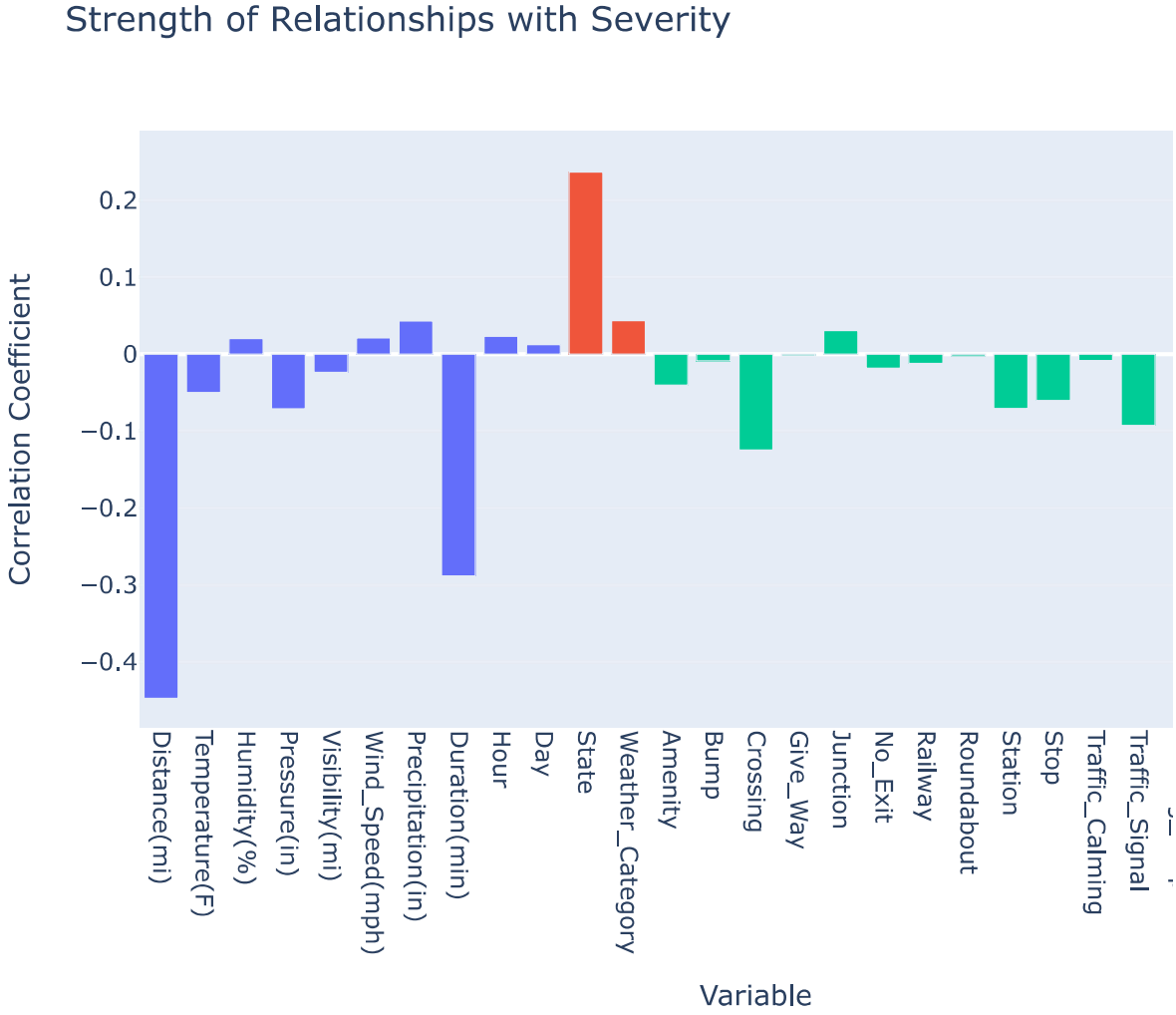
In [55]:
```
# Point-Biserial Correlation for boolean variables
for var in boolean_cols:
    corr, _ = pointbiserialr(df[var], df['Severity'])
    correlations['Variable'].append(var)
    correlations['Correlation'].append(corr)
    correlations['Type'].append('Boolean')
```

```
C:\Users\ngoum\anaconda3\envs\dl-env\lib\site-packages\scipy\stats\stats.py:4023:
PearsonRConstantInputWarning:

An input array is constant; the correlation coefficient is not defined.
```

In [56]:
```
# Spearman's Rank Correlation for ordinal variables
for var in ordinal_cols:
    # Encode ordinal variables to numerical
    df[var] = df[var].astype('category').cat.codes
    corr, _ = spearmanr(df[var], df['Severity'])
    correlations['Variable'].append(var)
    correlations['Correlation'].append(corr)
    correlations['Type'].append('Ordinal')
```

In [57]:
```
# Convert to DataFrame
correlations_df = pd.DataFrame(correlations)
```

In [58]:
```
fig = go.Figure()

# Iterate over unique types to create grouped bars
for type_name in correlations_df['Type'].unique():
    df_filtered = correlations_df[correlations_df['Type'] == type_name]

    fig.add_trace(
        go.Bar(x=df_filtered['Variable'], y=df_filtered['Correlation'], name=type_r
    )

# Update layout
fig.update_layout(
    title='Strength of Relationships with Severity',
    xaxis_title='Variable',
    yaxis_title='Correlation Coefficient',
    xaxis_tickangle=90,
    legend_title='Type',
    legend=dict(x=1, y=1, traceorder='normal'),
    barmode='group',
    height=550,
    width=850
)

fig.show(config={'staticPlot': True})
```



In [59]:
```
corr_threshold = 0.1
```

In [60]:
```
correlations_df[np.abs(correlations_df["Correlation"]) >= corr_threshold]
```

Out[60]:

	Variable	Correlation	Type
0	Distance(mi)	-0.447460	Numerical/Time
7	Duration(min)	-0.288564	Numerical/Time
10	State	0.236950	Categorical
14	Crossing	-0.124546	Boolean
26	Is_Highway	0.251375	Boolean

Some variables have a relatively strong relationship with the severity of the accident. We could use them as features in a predictive model.

- `Distance(mi)` : Strong negative correlation (-0.447) suggests it's a significant predictor.
- `Duration(min)` : Strong negative correlation (-0.289) indicates it should be included.
- `State` : Moderate positive correlation (0.237) implies state-specific factors affecting severity should be accounted for.

- `Weather_Category` : Slight positive correlation (0.044) and domain knowledge suggests considering weather conditions.
- `Is_Highway` : Moderate positive correlation (0.251) indicates the importance of distinguishing accidents on highways.
- `Crossing` : Moderate negative correlation (-0.125) highlights the impact of accidents involving crossings.
- `Hour` : Hourly patterns can directly relate to traffic conditions, commuter behavior, and visibility, which are critical factors influencing accident severity.

In [ ]: