Task 05:- Analyze traffic accident data to identify patterns related to road conditions, weather, and time of day. Visualize accident hotspots and contributing factors.

Importing Required Libraries

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import missingno as mn
import plotly.graph_objects as go
pd.options.mode.chained_assignment = None
pd.options.display.max_columns = 999

# Input data files are available in the "../input/" directory.
# For example, running this (by clicking run or pressing Shift+Enter) will list all

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# Any results you write to the current directory are saved as output.
from warnings import filterwarnings
# Ignore warnings
filterwarnings(action='ignore')
```

Data Handeling

Loading Dataset into Pandas Dataframe

In [2]:
```
data_filepath = (r"C:\Users\kunal\Documents\PRODIGY Intership Material\PRODIGY_DS_0
data = pd.read_csv(data_filepath)
data.shape
data
```

Out[2]:

	ID	Source	Severity	Start_Time	End_Time	Start_Lat	Start_Lng	End_Lat	E
0	A-1	Source2	3	2016-02-08 05:46:00	2016-02-08 11:00:00	39.865147	-84.058723	NaN	
1	A-2	Source2	2	2016-02-08 06:07:59	2016-02-08 06:37:59	39.928059	-82.831184	NaN	
2	A-3	Source2	2	2016-02-08 06:49:27	2016-02-08 07:19:27	39.063148	-84.032608	NaN	
3	A-4	Source2	3	2016-02-08 07:23:34	2016-02-08 07:53:34	39.747753	-84.205582	NaN	
4	A-5	Source2	2	2016-02-08 07:39:07	2016-02-08 08:09:07	39.627781	-84.188354	NaN	
...	...	...	...	...	...	...	...	...	
7728389	A-7777757	Source1	2	2019-08-23 18:03:25	2019-08-23 18:32:01	34.002480	-117.379360	33.99888	-11
7728390	A-7777758	Source1	2	2019-08-23 19:11:30	2019-08-23 19:38:23	32.766960	-117.148060	32.76555	-11
7728391	A-7777759	Source1	2	2019-08-23 19:00:21	2019-08-23 19:28:49	33.775450	-117.847790	33.77740	-11
7728392	A-7777760	Source1	2	2019-08-23 19:00:21	2019-08-23 19:29:42	33.992460	-118.403020	33.98311	-11
7728393	A-7777761	Source1	2	2019-08-23 18:52:06	2019-08-23 19:21:31	34.133930	-117.230920	34.13736	-11

7728394 rows × 46 columns

Information

In [3]:
```
data.info()
```

```
<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
```

In [4]:
```
data.columns # Print all the columns present in the Dataset
```

Out[4]:
```
Index(['ID', 'Source', 'Severity', 'Start_Time', 'End_Time', 'Start_Lat',
       'Start_Lng', 'End_Lat', 'End_Lng', 'Distance(mi)', 'Description',
       'Street', 'City', 'County', 'State', 'Zipcode', 'Country', 'Timezone',
       'Airport_Code', 'Weather_Timestamp', 'Temperature(F)', 'Wind_Chill(F)',
       'Humidity(%)', 'Pressure(in)', 'Visibility(mi)', 'Wind_Direction',
       'Wind_Speed(mph)', 'Precipitation(in)', 'Weather_Condition', 'Amenity',
       'Bump', 'Crossing', 'Give_Way', 'Junction', 'No_Exit', 'Railway',
       'Roundabout', 'Station', 'Stop', 'Traffic_Calming', 'Traffic_Signal',
       'Turning_Loop', 'Sunrise_Sunset', 'Civil_Twilight', 'Nautical_Twilight',
       'Astronomical_Twilight'],
      dtype='object')
```

In [5]:
```
data.describe().T #Statistical Description of each column
```

Out[5]:

	count	mean	std	min	25%	50%	75
Severity	7728394.0	2.212384	0.487531	1.000000	2.000000	2.000000	2.0000
Start_Lat	7728394.0	36.201195	5.076079	24.554800	33.399631	35.823974	40.0849
Start_Lng	7728394.0	-94.702545	17.391756	-124.623833	-117.219396	-87.766616	-80.3536
End_Lat	4325632.0	36.261829	5.272905	24.566013	33.462070	36.183495	40.1789
End_Lng	4325632.0	-95.725570	18.107928	-124.545748	-117.754345	-88.027890	-80.2470
Distance(mi)	7728394.0	0.561842	1.776811	0.000000	0.000000	0.030000	0.4640
Temperature(F)	7564541.0	61.663286	19.013653	-89.000000	49.000000	64.000000	76.0000
Wind_Chill(F)	5729375.0	58.251048	22.389832	-89.000000	43.000000	62.000000	75.0000
Humidity(%)	7554250.0	64.831041	22.820968	1.000000	48.000000	67.000000	84.0000
Pressure(in)	7587715.0	29.538986	1.006190	0.000000	29.370000	29.860000	30.0300
Visibility(mi)	7551296.0	9.090376	2.688316	0.000000	10.000000	10.000000	10.0000
Wind_Speed(mph)	7157161.0	7.685490	5.424983	0.000000	4.600000	7.000000	10.4000
Precipitation(in)	5524808.0	0.008407	0.110225	0.000000	0.000000	0.000000	0.0000

Numerical Columns to deal with

In [6]:
```
# int, float and boolean data
print(data.count(numeric_only=True))
print("Total No. of Numerical Columns:", len(data.count(numeric_only=True)))
```

```
Severity              7728394
Start_Lat             7728394
Start_Lng             7728394
End_Lat               4325632
End_Lng               4325632
Distance(mi)          7728394
Temperature(F)        7564541
Wind_Chill(F)         5729375
Humidity(%)           7554250
Pressure(in)          7587715
Visibility(mi)        7551296
Wind_Speed(mph)       7157161
Precipitation(in)     5524808
Amenity               7728394
Bump                  7728394
Crossing              7728394
Give_Way              7728394
Junction              7728394
No_Exit               7728394
Railway               7728394
Roundabout            7728394
Station               7728394
Stop                  7728394
Traffic_Calming       7728394
Traffic_Signal        7728394
Turning_Loop          7728394
dtype: int64
Total No. of Numerical Columns: 26
```

Percentage of Missing Values

In [7]:
```
missing_values = data.isna().sum().sort_values(ascending=False)
missing_percentage = missing_values[missing_values!=0]/len(data)*100
print(" Percentage of Missing Values \n", missing_percentage)
```

```
 Percentage of Missing Values
 End_Lat                 44.029355
End_Lng                 44.029355
Precipitation(in)       28.512858
Wind_Chill(F)           25.865904
Wind_Speed(mph)          7.391355
Visibility(mi)           2.291524
Wind_Direction           2.267043
Humidity(%)              2.253301
Weather_Condition        2.244438
Temperature(F)           2.120143
Pressure(in)             1.820288
Weather_Timestamp        1.555666
Nautical_Twilight        0.300787
Civil_Twilight           0.300787
Sunrise_Sunset           0.300787
Astronomical_Twilight    0.300787
Airport_Code             0.292881
Street                   0.140637
Timezone                 0.101030
Zipcode                  0.024779
City                     0.003274
Description              0.000065
dtype: float64
```

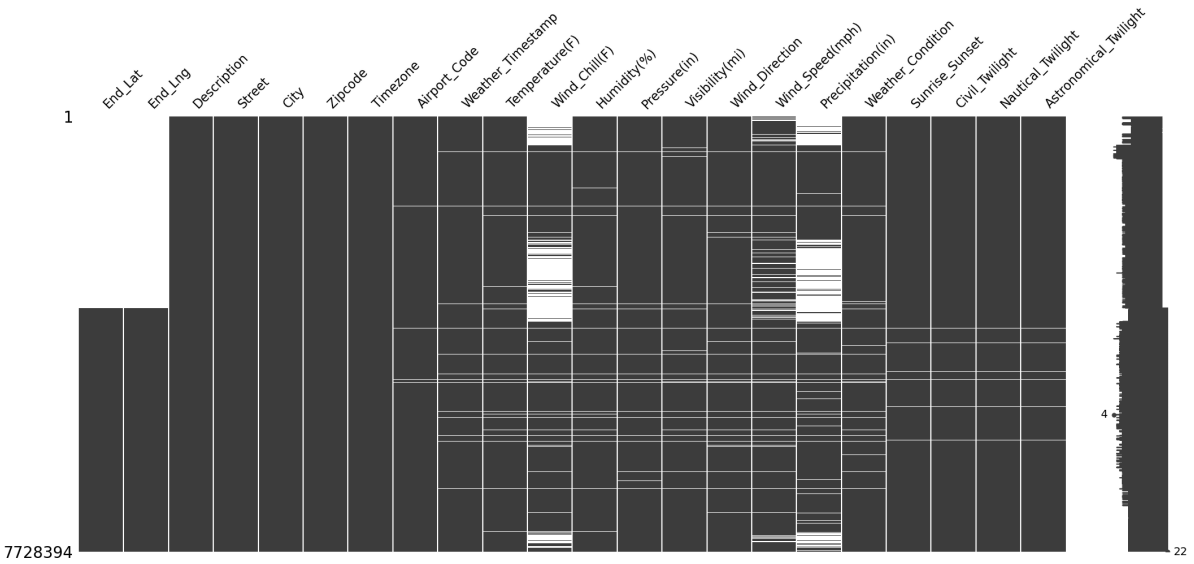
Getting List of Columns Having Null Values

In [8]:
```
null_cols = [i for i in data.columns if data[i].isnull().any()]
print(null_cols)
```

```
['End_Lat', 'End_Lng', 'Description', 'Street', 'City', 'Zipcode', 'Timezone', 'Ai
rport_Code', 'Weather_Timestamp', 'Temperature(F)', 'Wind_Chill(F)', 'Humidity
(%)', 'Pressure(in)', 'Visibility(mi)', 'Wind_Direction', 'Wind_Speed(mph)', 'Prec
ipitation(in)', 'Weather_Condition', 'Sunrise_Sunset', 'Civil_Twilight', 'Nautical
_Twilight', 'Astronomical_Twilight']
```

Checking Missing Values Pattern in Dataframe

In [9]:
```
mn.matrix(data[null_cols]);
```



Drop Four Columns [End_Lng, End_Lat] having high no. of Missing Values

In [12]:
```
new_data_a = data.drop(columns=["End_Lng", "End_Lat"], axis=0)
```

Drop Rows w.r.t to columns having least percentage of missing values (do not effect much to data with 4.2 million records)

In [16]:
```
new_data_b = new_data_a.dropna(subset = ['Visibility(mi)','Weather_Condition','Humi
```

In [17]:
```
new_data_b.isnull().sum()
```

Out[17]:

```
ID                          0
Source                      0
Severity                    0
Start_Time                  0
End_Time                    0
Start_Lat                   0
Start_Lng                   0
Distance(mi)                0
Description                 0
Street                  10214
City                        0
County                      0
State                       0
Zipcode                     0
Country                     0
Timezone                    0
Airport_Code                0
Weather_Timestamp           0
Temperature(F)              0
Wind_Chill(F)         1769892
Humidity(%)                 0
Pressure(in)                0
Visibility(mi)              0
Wind_Direction              0
Wind_Speed(mph)        375174
Precipitation(in)     2039619
Weather_Condition           0
Amenity                     0
Bump                        0
Crossing                    0
Give_Way                    0
Junction                    0
No_Exit                     0
Railway                     0
Roundabout                  0
Station                     0
Stop                        0
Traffic_Calming             0
Traffic_Signal              0
Turning_Loop                0
Sunrise_Sunset              0
Civil_Twilight              0
Nautical_Twilight           0
Astronomical_Twilight       0
dtype: int64
```

In [18]:

```
final_data = new_data_b.drop(columns = 'ID', axis=0)
```

In [19]:

```
final_data.isnull().sum()
```

Out[19]:

```
Source                       0
Severity                     0
Start_Time                   0
End_Time                     0
Start_Lat                    0
Start_Lng                    0
Distance(mi)                 0
Description                  0
Street                   10214
City                         0
County                       0
State                        0
Zipcode                      0
Country                      0
Timezone                     0
Airport_Code                 0
Weather_Timestamp            0
Temperature(F)               0
Wind_Chill(F)          1769892
Humidity(%)                  0
Pressure(in)                 0
Visibility(mi)               0
Wind_Direction               0
Wind_Speed(mph)         375174
Precipitation(in)      2039619
Weather_Condition            0
Amenity                      0
Bump                         0
Crossing                     0
Give_Way                     0
Junction                     0
No_Exit                      0
Railway                      0
Roundabout                   0
Station                      0
Stop                         0
Traffic_Calming              0
Traffic_Signal               0
Turning_Loop                 0
Sunrise_Sunset               0
Civil_Twilight               0
Nautical_Twilight            0
Astronomical_Twilight        0
dtype: int64
```

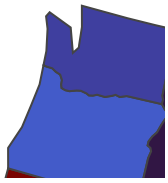
Exploratory Data Analysis

Location

In [25]:

```
state_counts = final_data["State"].value_counts()
fig = go.Figure(data=go.Choropleth(locations=state_counts.index, z=state_counts.val
fig.update_layout(title_text="Number of Accidents for each State", geo_scope="usa")
fig.show()
```

Number of Accidents for each State

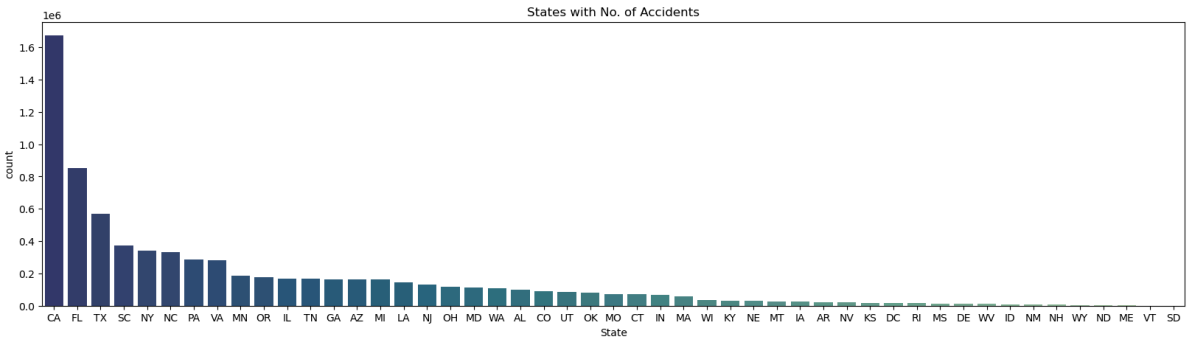


In [28]:
```
print("State Code: ", final_data.State.unique())
print("Total No. of State in Dataset: ", len(final_data.State.unique()))
```

```
State Code:  ['OH' 'WV' 'CA' 'FL' 'GA' 'SC' 'NE' 'IA' 'IL' 'MO' 'WI' 'IN' 'MI' 'N
J'
 'NY' 'CT' 'MA' 'RI' 'NH' 'PA' 'KY' 'MD' 'VA' 'DC' 'DE' 'TX' 'WA' 'OR'
 'AL' 'NC' 'AZ' 'TN' 'LA' 'MN' 'CO' 'OK' 'NV' 'UT' 'KS' 'NM' 'AR' 'MS'
 'ME' 'VT' 'WY' 'ID' 'ND' 'MT' 'SD']
Total No. of State in Dataset:  49
```

Total No. of State in Dataset: 49 There are 50 states in US New York not in dataset

In [29]:
```
fig, ax = plt.subplots(figsize = (20,5))
c = sns.countplot(x="State", data=final_data, orient = 'v', palette = "crest_r", or
c.set_title("States with No. of Accidents");
```



California (CA) is the 3rd most largest state of US after Texas (TX) and Alaska(AL) Also California (CA) is the most populated among all, followed by Texas (TX) Alaska (AL) is the

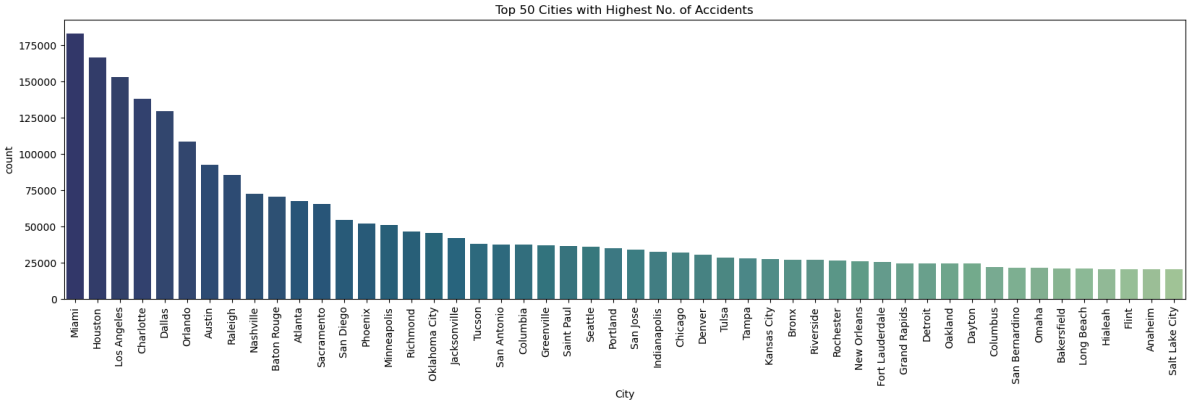
largest state but least populated state at 48th rank

In [30]:
```
print("City Code: ", final_data.City.unique())
print("Total No. of Cities in Dataset: ", len(final_data.City.unique()))
```

```
City Code:  ['Dayton' 'Reynoldsburg' 'Williamsburg' ... 'Ness City' 'Clarksdale'
 'American Fork-Pleasant Grove']
Total No. of Cities in Dataset:  12237
```

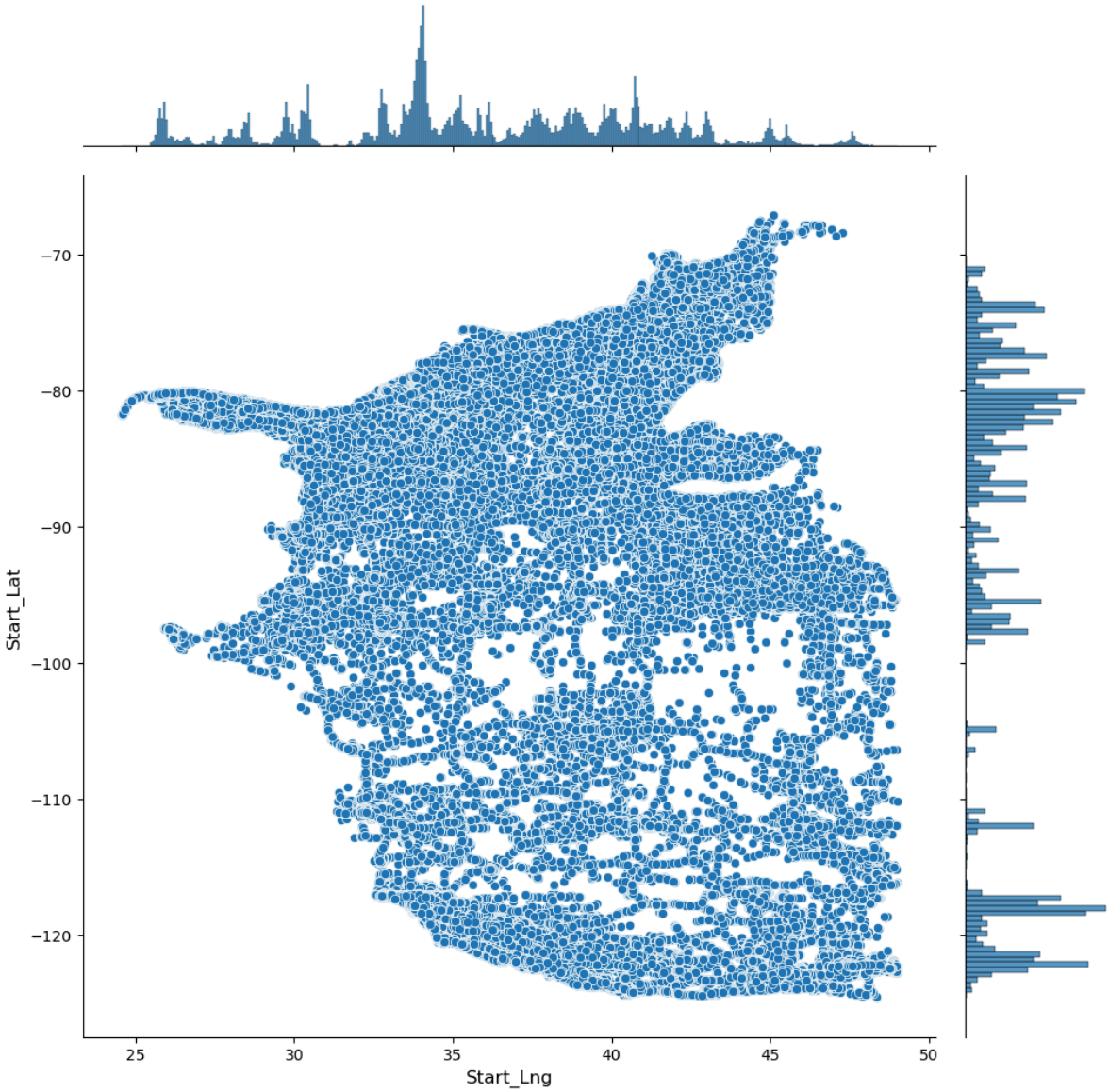
Total No. of Cities in Dataset: 11537 There were 19,502 incorporated places registered in the United States

In [31]:
```
fig, ax = plt.subplots(figsize = (20,5))
c = sns.countplot(x="City", data=final_data, order=final_data.City.value_counts().i
c.set_title("Top 50 Cities with Highest No. of Accidents")
c.set_xticklabels(c.get_xticklabels(), rotation=90)
plt.show()
```

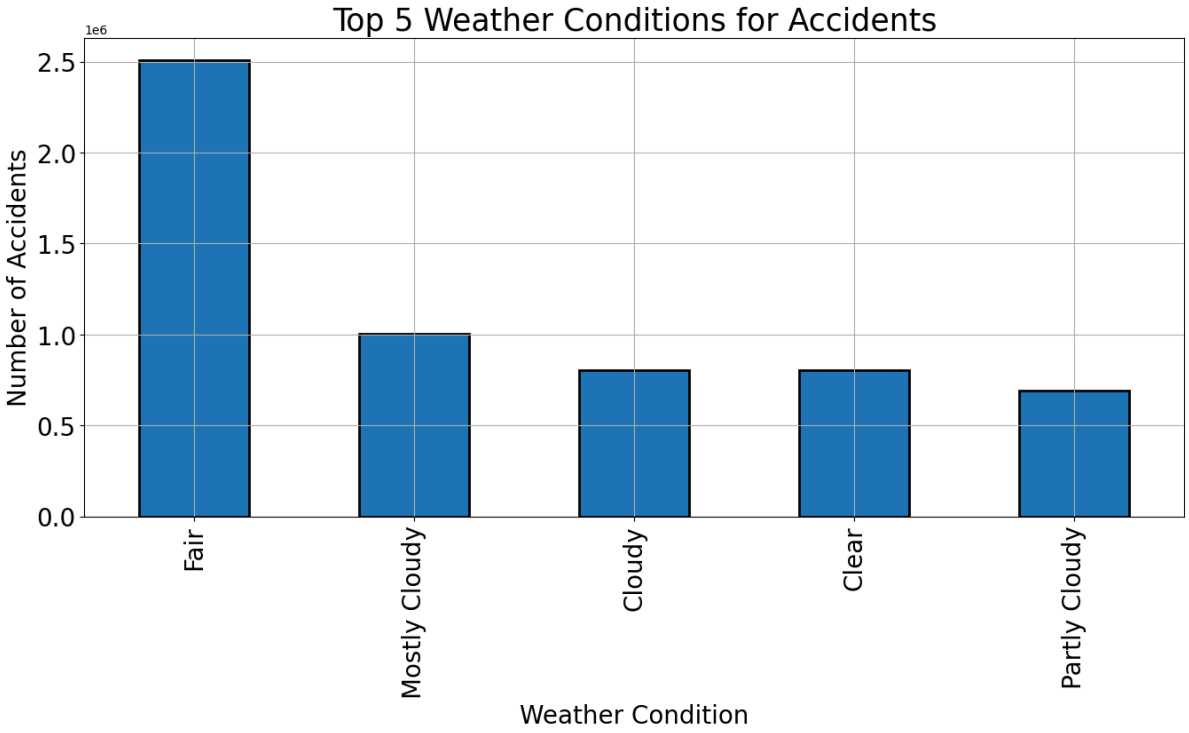


In [33]:
```
### Latitude and Longitude Jointplot

sns.jointplot(x=final_data.Start_Lat.values, y=final_data.Start_Lng.values, height=
plt.ylabel('Start_Lat', fontsize=12)
plt.xlabel('Start_Lng', fontsize=12)
plt.show()
```

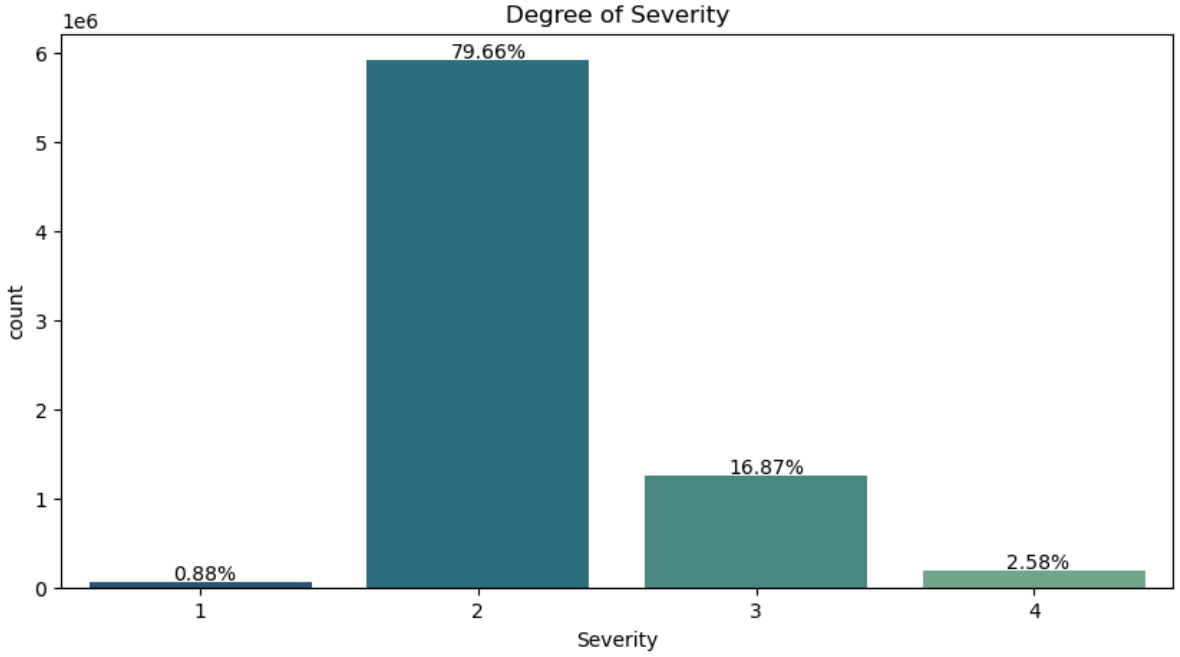


In [32]:
```
### Top 5 Weather Conditions for Accidents
fig, ax = plt.subplots(figsize=(16,7))
final_data['Weather_Condition'].value_counts().sort_values(ascending=False).head(5)
    width=0.5, edgecolor='k', align='center', linewidth=2, ax=ax)
plt.xlabel('Weather Condition', fontsize=20)
plt.ylabel('Number of Accidents', fontsize=20)
ax.tick_params(labelsize=20)
plt.title('Top 5 Weather Conditions for Accidents', fontsize=25)
plt.grid()
plt.ioff()
plt.show()
```



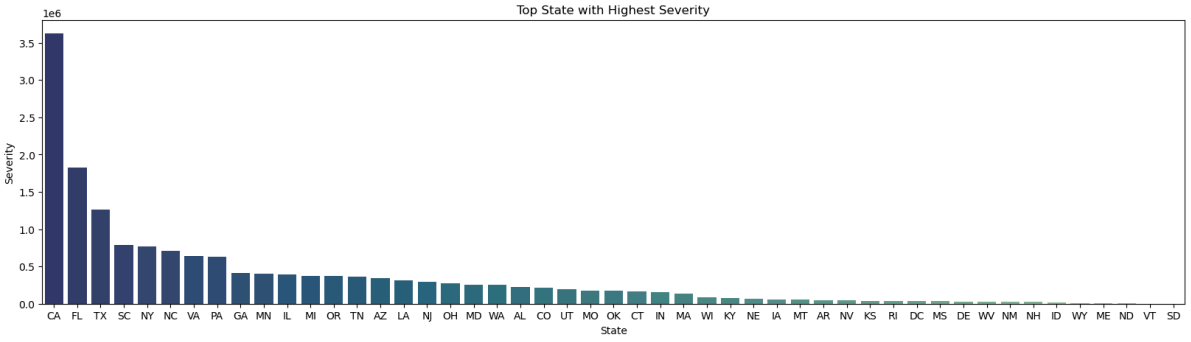
Severity

In [39]:
```
fig, ax = plt.subplots(figsize = (10,5))
c = sns.countplot(x="Severity", data=final_data, orient = 'v', palette = "crest_r")
c.set_title("Degree of Severity")
for i in ax.patches:
    count = "{:.2%}".format(i.get_height()/len(final_data.Severity))
    x = i.get_x()+i.get_width()-0.50
    y = i.get_height()+20000
    ax.annotate(count, (x, y))
plt.show()
```

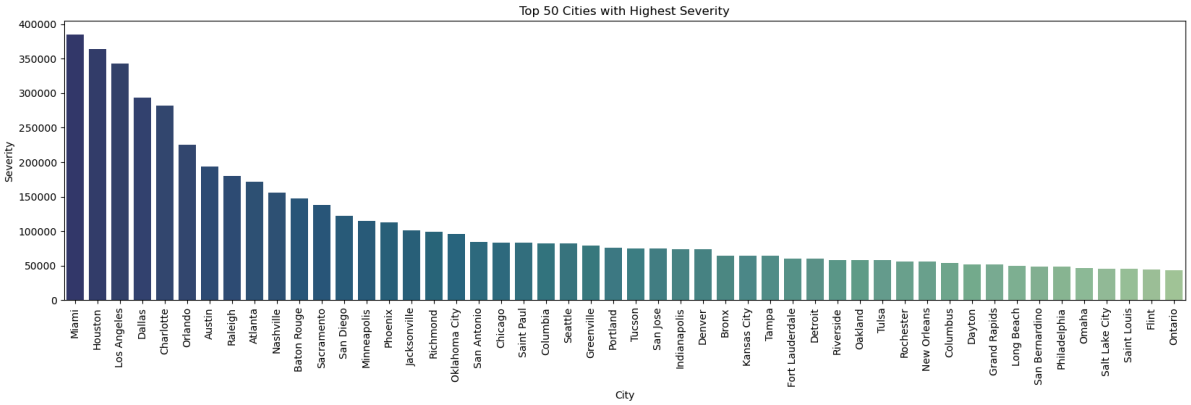


In [40]:
```
df_state = final_data.groupby('State').sum('Severity')[['Severity']]
df_state = df_state.reset_index().sort_values('Severity', ascending = False)[:50]
fig, ax = plt.subplots(figsize = (20,5))
c = sns.barplot(x = 'State', y = 'Severity', data = df_state, orient = 'v', palette
```

```
c.set_title("Top State with Highest Severity")
plt.show()
```



In [41]:
```
df_city = final_data.groupby('City').sum('Severity')[['Severity']]
df_city = df_city.reset_index().sort_values('Severity', ascending = False)[:50]
fig, ax = plt.subplots(figsize = (20,5))
c = sns.barplot(x = 'City', y = 'Severity', data = df_city, orient = 'v', palette =
c.set_title("Top 50 Cities with Highest Severity")
c.set_xticklabels(c.get_xticklabels(), rotation=90)
plt.show()
```



Weather Stimuli Impact

Related Columns

In [42]:
```
final_data.iloc[:10, 17:26]
```

Out[42]:

	Temperature(F)	Wind_Chill(F)	Humidity(%)	Pressure(in)	Visibility(mi)	Wind_Direction	Wind_S
0	36.9	NaN	91.0	29.68	10.0	Calm	
1	37.9	NaN	100.0	29.65	10.0	Calm	
2	36.0	33.3	100.0	29.67	10.0	SW	
3	35.1	31.0	96.0	29.64	9.0	SW	
4	36.0	33.3	89.0	29.65	6.0	SW	
5	37.9	35.5	97.0	29.63	7.0	SSW	
6	34.0	31.0	100.0	29.66	7.0	WSW	
7	34.0	31.0	100.0	29.66	7.0	WSW	
8	33.3	NaN	99.0	29.67	5.0	SW	
9	37.4	33.8	100.0	29.62	3.0	SSW	

Location Impact

Related Columns

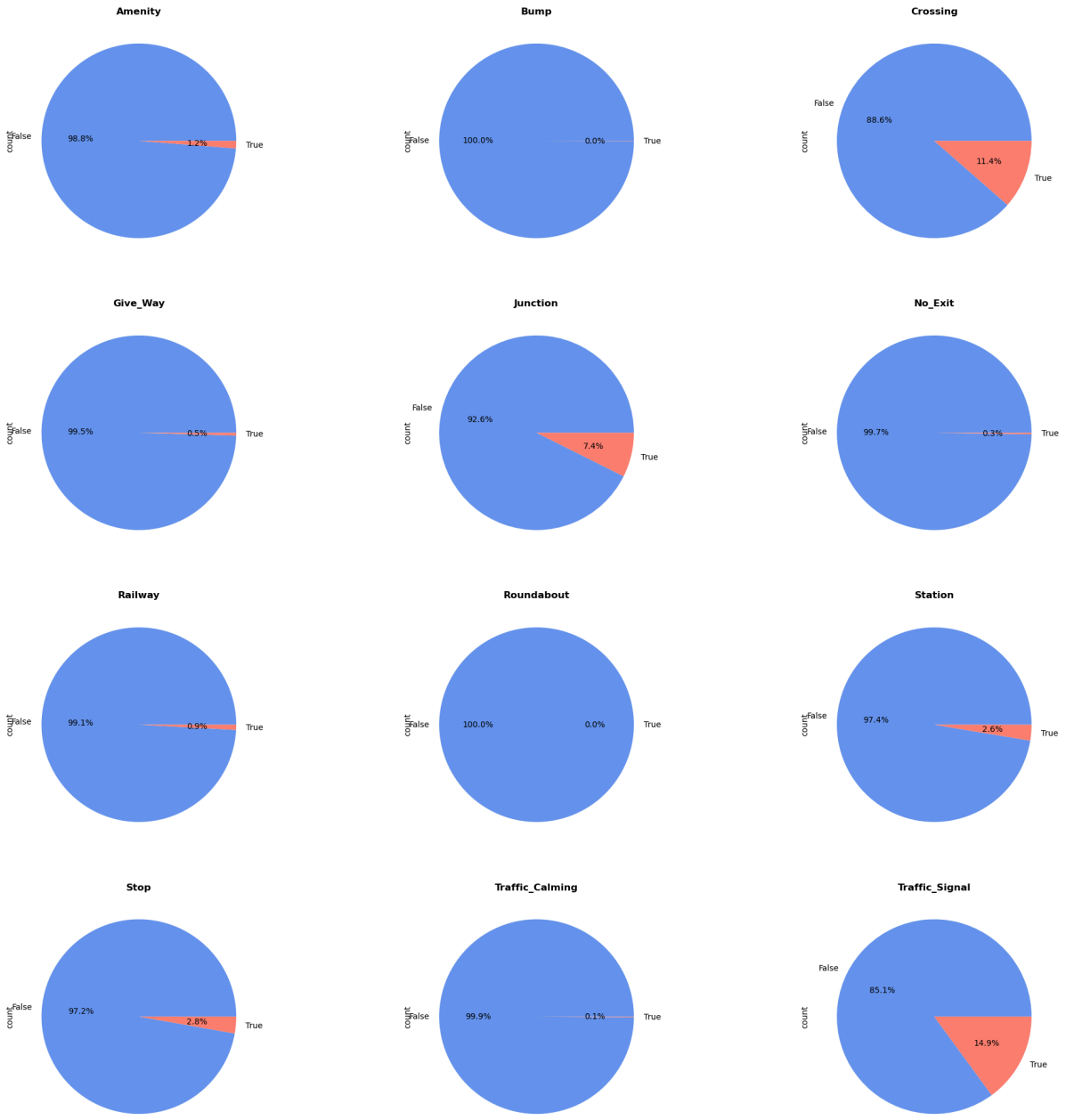
In [43]:
```
final_data.iloc[:10, 27:39]
```

Out[43]:

	Bump	Crossing	Give_Way	Junction	No_Exit	Railway	Roundabout	Station	Stop	Traffic_Cal
0	False	False	False	False	False	False	False	False	False	
1	False	False	False	False	False	False	False	False	False	
2	False	False	False	False	False	False	False	False	False	
3	False	False	False	False	False	False	False	False	False	
4	False	False	False	False	False	False	False	False	False	
5	False	False	False	False	False	False	False	False	False	
6	False	False	False	False	False	False	False	False	False	
7	False	False	False	False	False	False	False	False	False	
8	False	False	False	False	False	False	False	False	False	
9	False	False	False	False	False	False	False	False	False	

In [44]:
```
f,ax=plt.subplots(4,3,figsize=(25,25))
ax[0,0] = final_data['Amenity'].value_counts().plot.pie(autopct='%1.1f%%',ax=ax[0,0
ax[0,0].set_title("Amenity",fontweight ="bold")
ax[0,1] = final_data['Bump'].value_counts().plot.pie(autopct='%1.1f%%',ax=ax[0,1],c
ax[0,1].set_title("Bump",fontweight ="bold")
ax[0,2] = final_data['Crossing'].value_counts().plot.pie(autopct='%1.1f%%',ax=ax[0,
ax[0,2].set_title("Crossing",fontweight ="bold")
ax[1,0] = final_data['Give_Way'].value_counts().plot.pie(autopct='%1.1f%%',ax=ax[1,
ax[1,0].set_title("Give_Way",fontweight ="bold")
ax[1,1] = final_data['Junction'].value_counts().plot.pie(autopct='%1.1f%%',ax=ax[1,
ax[1,1].set_title("Junction",fontweight ="bold")
ax[1,2] = final_data['No_Exit'].value_counts().plot.pie(autopct='%1.1f%%',ax=ax[1,2
```

```
ax[1,2].set_title("No_Exit",fontweight ="bold")
ax[2,0] = final_data['Railway'].value_counts().plot.pie(autopct='%1.1f%%',ax=ax[2,0
ax[2,0].set_title("Railway",fontweight ="bold")
ax[2,1] = final_data['Roundabout'].value_counts().plot.pie(autopct='%1.1f%%',ax=ax[
ax[2,1].set_title("Roundabout",fontweight ="bold")
ax[2,2] = final_data['Station'].value_counts().plot.pie(autopct='%1.1f%%',ax=ax[2,2
ax[2,2].set_title("Station",fontweight ="bold")
ax[3,0] = final_data['Stop'].value_counts().plot.pie(autopct='%1.1f%%',ax=ax[3,0],c
ax[3,0].set_title("Stop",fontweight ="bold")
ax[3,1] = final_data['Traffic_Calming'].value_counts().plot.pie(autopct='%1.1f%%',a
ax[3,1].set_title("Traffic_Calming",fontweight ="bold")
ax[3,2] = final_data['Traffic_Signal'].value_counts().plot.pie(autopct='%1.1f%%',ax
ax[3,2].set_title("Traffic_Signal",fontweight ="bold")
plt.show()
```



In [ ]: