US Accidents Exploratory Data Analysis

Details

-Source-Kaggle

-Information about accidents

-can be useful to prevent accidents

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
df = pd.read_csv('US_Accidents_March23.csv', encoding= 'unicode_escape',low_memory=
```

In [40]:
```
df.head()
```

Out[40]:

	ID	Source	Severity	Start_Time	End_Time	Start_Lat	Start_Lng	End_Lat	End_Lng	Distanc
0	A-1	Source2	3	2016-02-08 05:46:00	2016-02-08 11:00:00	39.865147	-84.058723	NaN	NaN	
1	A-2	Source2	2	2016-02-08 06:07:59	2016-02-08 06:37:59	39.928059	-82.831184	NaN	NaN	
2	A-3	Source2	2	2016-02-08 06:49:27	2016-02-08 07:19:27	39.063148	-84.032608	NaN	NaN	
3	A-4	Source2	3	2016-02-08 07:23:34	2016-02-08 07:53:34	39.747753	-84.205582	NaN	NaN	
4	A-5	Source2	2	2016-02-08 07:39:07	2016-02-08 08:09:07	39.627781	-84.188354	NaN	NaN	

5 rows × 46 columns

In [41]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2879985 entries, 0 to 2879984
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                object
 30  Bump                   object
 31  Crossing               object
 32  Give_Way               object
 33  Junction               object
 34  No_Exit                object
 35  Railway                object
 36  Roundabout             object
 37  Station                object
 38  Stop                   object
 39  Traffic_Calming        object
 40  Traffic_Signal         object
 41  Turning_Loop           object
 42  Sunrise_Sunset         object
 43  Civil_Twilight         object
 44  Nautical_Twilight      object
 45  Astronomical_Twilight  object
dtypes: float64(12), int64(1), object(33)
memory usage: 1010.7+ MB
```

In [42]:
```
df.describe()
```

Out[42]:

	Severity	Start_Lat	Start_Lng	End_Lat	End_Lng	Distance(mi)	Temperature(
count	2.879985e+06	2.879985e+06	2.879985e+06	0.0	0.0	2.879985e+06	2.832021e+(
mean	2.325735e+00	3.609884e+01	-9.348430e+01	NaN	NaN	2.184002e-01	6.306505e+(
std	5.063891e-01	4.807686e+00	1.641337e+01	NaN	NaN	1.661789e+00	1.832914e+(
min	1.000000e+00	2.455480e+01	-1.245344e+02	NaN	NaN	0.000000e+00	-8.900000e+(
25%	2.000000e+00	3.321257e+01	-1.109220e+02	NaN	NaN	0.000000e+00	5.110000e+(
50%	2.000000e+00	3.539169e+01	-8.727915e+01	NaN	NaN	0.000000e+00	6.500000e+(
75%	3.000000e+00	3.998390e+01	-8.084421e+01	NaN	NaN	0.000000e+00	7.700000e+(
max	4.000000e+00	4.900220e+01	-6.755331e+01	NaN	NaN	4.417500e+02	2.030000e+(

In [43]:
```
#finding missing and incorrect values
```

In [44]:
```
pd.isnull(df).sum()
```

Out[44]:

```
ID                          0
Source                      0
Severity                    0
Start_Time                  0
End_Time                    0
Start_Lat                   0
Start_Lng                   0
End_Lat                2879985
End_Lng                2879985
Distance(mi)                0
Description                 5
Street                   1712
City                       56
County                      0
State                       0
Zipcode                   400
Country                     0
Timezone                 2272
Airport_Code             5387
Weather_Timestamp       33359
Temperature(F)          47964
Wind_Chill(F)         1052101
Humidity(%)             51430
Pressure(in)            39942
Visibility(mi)          54039
Wind_Direction          47683
Wind_Speed(mph)        260854
Precipitation(in)     1122612
Weather_Condition       53196
Amenity                     1
Bump                        1
Crossing                    1
Give_Way                    1
Junction                    1
No_Exit                     1
Railway                     1
Roundabout                  1
Station                     1
Stop                        1
Traffic_Calming             1
Traffic_Signal              1
Turning_Loop                1
Sunrise_Sunset           1670
Civil_Twilight           1670
Nautical_Twilight        1670
Astronomical_Twilight    1670
dtype: int64
```

In [51]:

```
pd.isnull(df).sum().sort_values(ascending=False)
```

Out[51]:

```
End_Lat                 2879985
End_Lng                 2879985
Precipitation(in)       1122612
Wind_Chill(F)           1052101
Wind_Speed(mph)          260854
Visibility(mi)            54039
Weather_Condition         53196
Humidity(%)               51430
Temperature(F)            47964
Wind_Direction            47683
Pressure(in)              39942
Weather_Timestamp         33359
Airport_Code               5387
Timezone                   2272
Street                     1712
Sunrise_Sunset             1670
Civil_Twilight             1670
Nautical_Twilight          1670
Astronomical_Twilight      1670
Zipcode                     400
City                         56
Description                   5
Give_Way                      1
Bump                          1
Roundabout                    1
Railway                       1
No_Exit                       1
Junction                      1
Crossing                      1
Traffic_Calming               1
Amenity                       1
Stop                          1
Traffic_Signal                1
Turning_Loop                  1
Station                       1
End_Time                      0
Start_Time                    0
Severity                      0
Country                       0
Start_Lat                     0
Start_Lng                     0
Distance(mi)                  0
Source                        0
County                        0
State                         0
ID                            0
dtype: int64
```

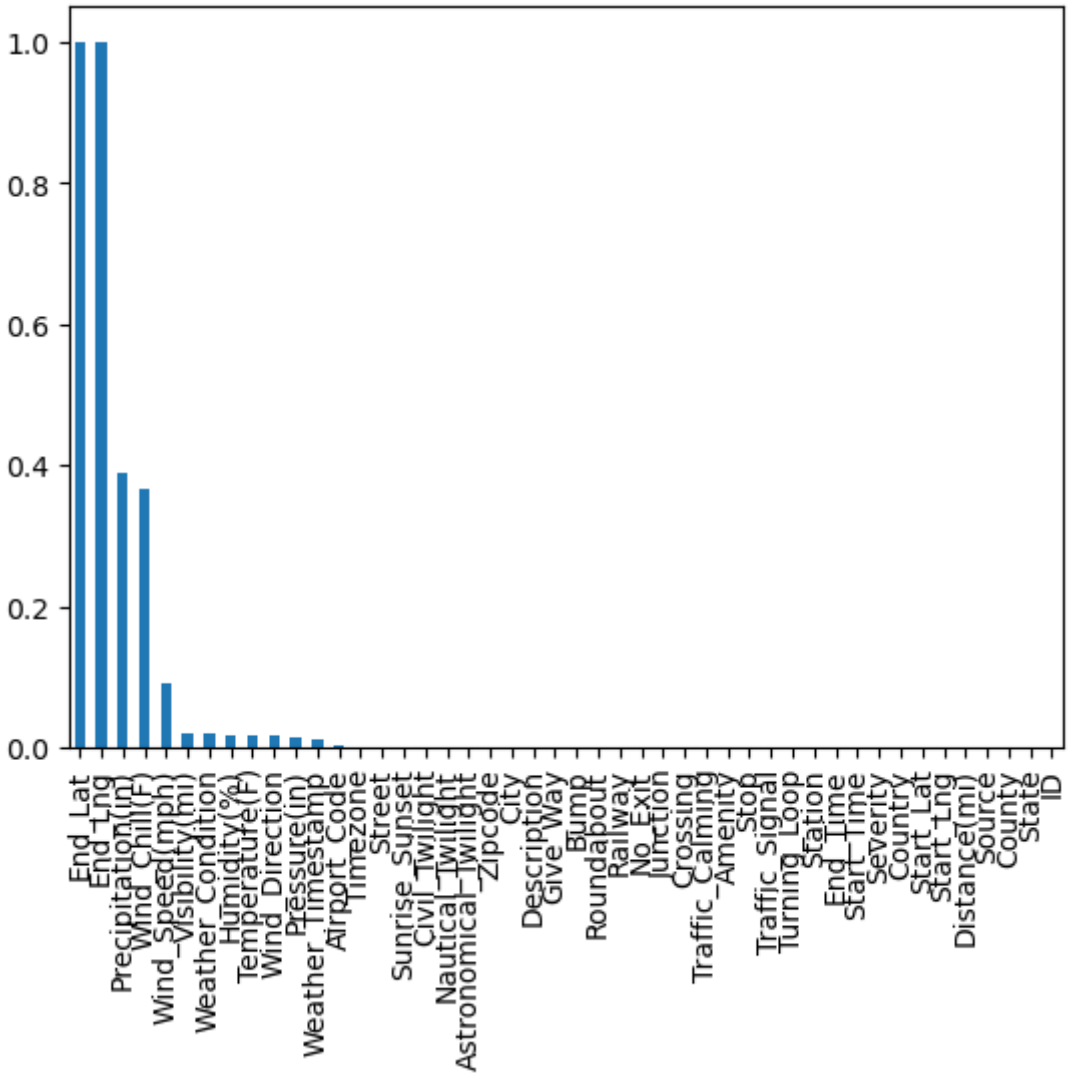
In [54]:

```
#calculating missing percentage
missing_percentages= pd.isnull(df).sum().sort_values(ascending=False)/len(df)
missing_percentages
```

Out[54]:
```
End_Lat                 1.000000e+00
End_Lng                 1.000000e+00
Precipitation(in)       3.897979e-01
Wind_Chill(F)           3.653147e-01
Wind_Speed(mph)         9.057478e-02
Visibility(mi)          1.876364e-02
Weather_Condition       1.847093e-02
Humidity(%)             1.785773e-02
Temperature(F)          1.665425e-02
Wind_Direction          1.655668e-02
Pressure(in)            1.386882e-02
Weather_Timestamp       1.158305e-02
Airport_Code            1.870496e-03
Timezone                7.888930e-04
Street                  5.944475e-04
Sunrise_Sunset          5.798641e-04
Civil_Twilight          5.798641e-04
Nautical_Twilight       5.798641e-04
Astronomical_Twilight   5.798641e-04
Zipcode                 1.388896e-04
City                    1.944455e-05
Description             1.736120e-06
Give_Way                3.472240e-07
Bump                    3.472240e-07
Roundabout              3.472240e-07
Railway                 3.472240e-07
No_Exit                 3.472240e-07
Junction                3.472240e-07
Crossing                3.472240e-07
Traffic_Calming         3.472240e-07
Amenity                 3.472240e-07
Stop                    3.472240e-07
Traffic_Signal          3.472240e-07
Turning_Loop            3.472240e-07
Station                 3.472240e-07
End_Time                0.000000e+00
Start_Time              0.000000e+00
Severity                0.000000e+00
Country                 0.000000e+00
Start_Lat               0.000000e+00
Start_Lng               0.000000e+00
Distance(mi)            0.000000e+00
Source                  0.000000e+00
County                  0.000000e+00
State                   0.000000e+00
ID                      0.000000e+00
dtype: float64
```

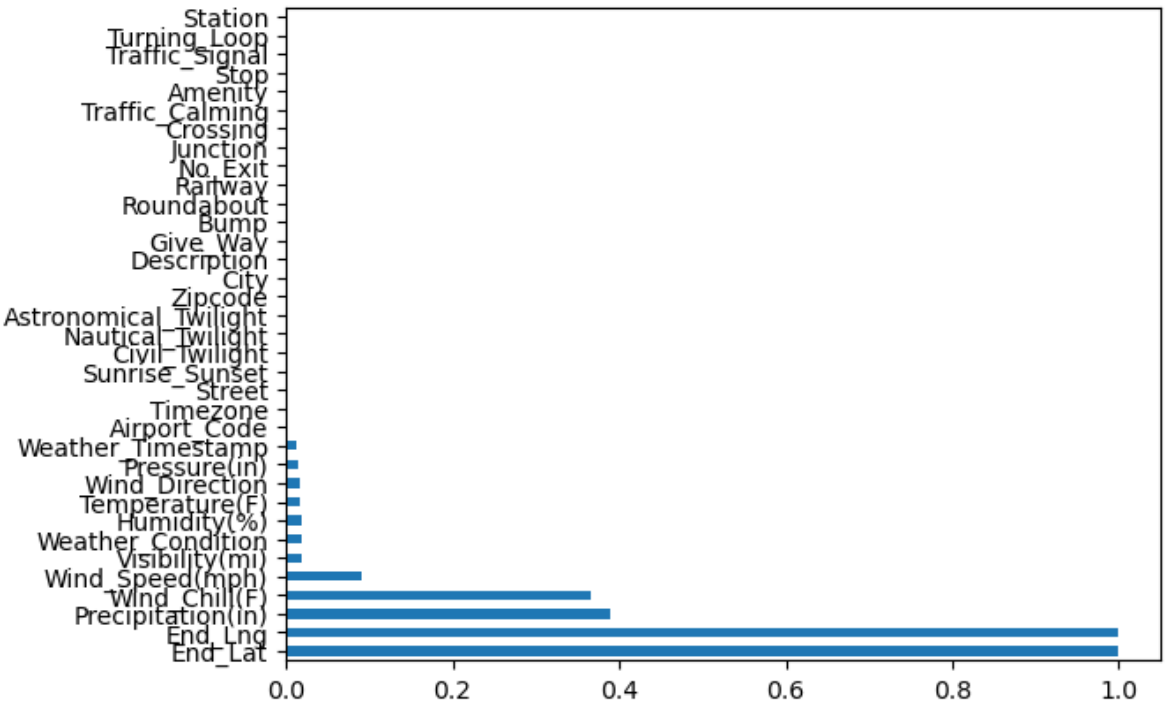
In [56]:
```
missing_percentages.plot(kind='bar')
```

Out[56]:
```
<Axes: >
```



In [57]:
```
missing_percentages[missing_percentages != 0].plot(kind='barh')
```

Out[57]:
```
<Axes: >
```



Exploratory Data Analysis and Visualization

In [59]:
```
df.columns
```

Out[59]:
```
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 [61]:
```
#colums we will analyse are
# 1.City
# 2.start time
# 3.Start_Lng,Start_Lat
# 4.Temperature
# 5.Weather Condition
```

In [66]:
```
cities=df.City.unique()
len(cities)
```

Out[66]:
```
11022
```

CITY

In [111…
```
df.City
```

Out[111]:
```
0               Dayton
1          Reynoldsburg
2          Williamsburg
3               Dayton
4               Dayton
               ...
2879980    Indianapolis
2879981      Louisville
2879982    Indianapolis
2879983       Pendleton
2879984       Pendleton
Name: City, Length: 2879985, dtype: object
```

In [68]:
```
cities_by_accident= df.City.value_counts()
cities_by_accident
```

Out[68]:
```
Houston            93660
Charlotte          68362
Dallas             63531
Austin             58490
Los Angeles        55294
                   ...
Robert                 1
West Burlington        1
Osawatomie             1
Bean Station           1
Bosler                 1
Name: City, Length: 11021, dtype: int64
```

In [70]:
```
cities_by_accident[:2 0]
```

Out[70]:
```
Houston          93660
Charlotte        68362
Dallas           63531
Austin           58490
Los Angeles      55294
Raleigh          39390
Atlanta          34811
Nashville        34612
Oklahoma City    34150
Baton Rouge      33638
Miami            28804
Orlando          24872
San Antonio      23679
Phoenix          23249
Greenville       22022
Sacramento       20562
Seattle          19903
Richmond         18931
Columbia         18285
Jacksonville     16682
Name: City, dtype: int64
```

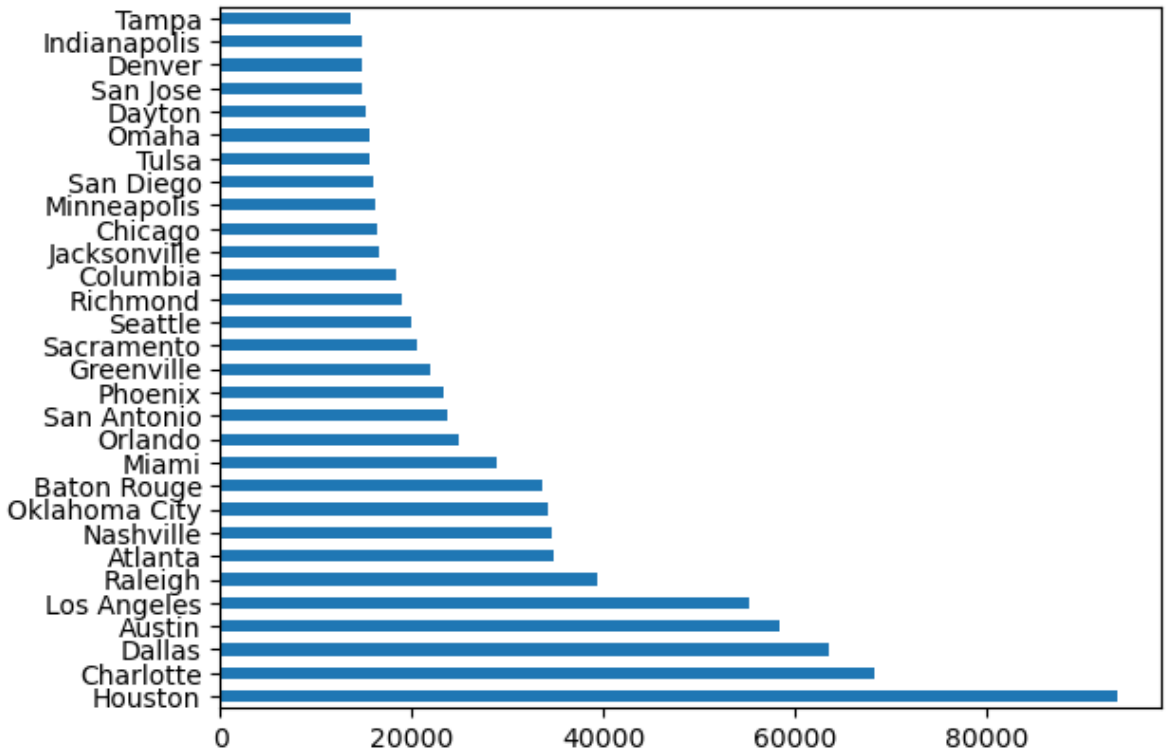
In [71]:
```
'New York' in df.City
```

Out[71]:
```
False
```

In [72]:
```
#it means this file doesnt contain data about New York because
```

In [73]:
```
cities_by_accident[:30].plot(kind='barh')
```

Out[73]:
```
<Axes: >
```



In [74]:
```
sns.set_style("darkgrid")
```

In [91]:
```
high_accident_cities = cities_by_accident[cities_by_accident >= 1000]
high_accident_cities
```

Out[91]:
```
Houston        93660
Charlotte      68362
Dallas         63531
Austin         58490
Los Angeles    55294
               ...
Asheville       1011
Delaware        1010
Belton          1004
Bridgeport      1004
Warwick         1004
Name: City, Length: 468, dtype: int64
```

In [92]:
```
low_accident_cities = cities_by_accident[cities_by_accident < 1000]
low_accident_cities
```

Out[92]:
```
Lisle              998
Arcadia            998
Newberry           996
Zachary            995
Royal Oak          994
                  ...
Robert               1
West Burlington      1
Osawatomie           1
Bean Station         1
Bosler               1
Name: City, Length: 10553, dtype: int64
```

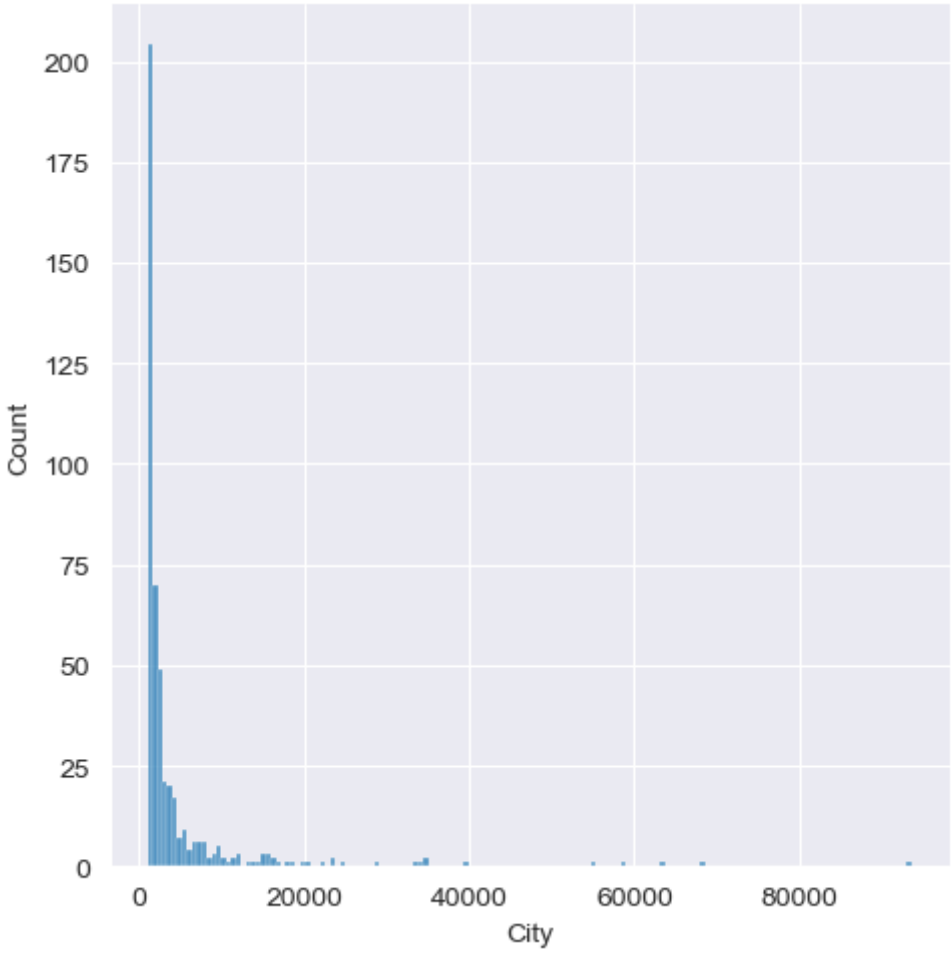
In [94]:
```
len(high_accident_cities)/len(cities)
```

Out[94]:
```
0.04246053347849755
```

In [95]:
```
#less than 5% of cities have more than thousand yearly accidents
```

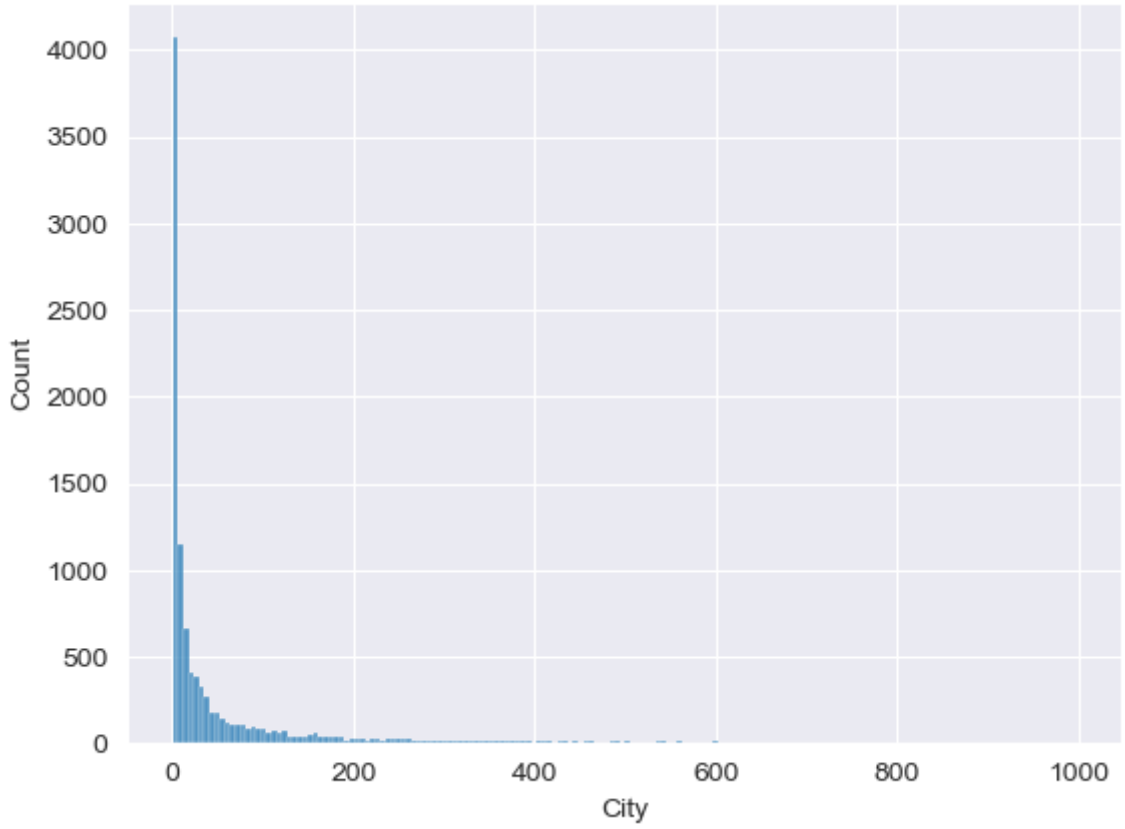
In [99]:
```
sns.histplot(high_accident_cities)
```

Out[99]:
```
<seaborn.axisgrid.FacetGrid at 0x1a0a6cb3640>
```



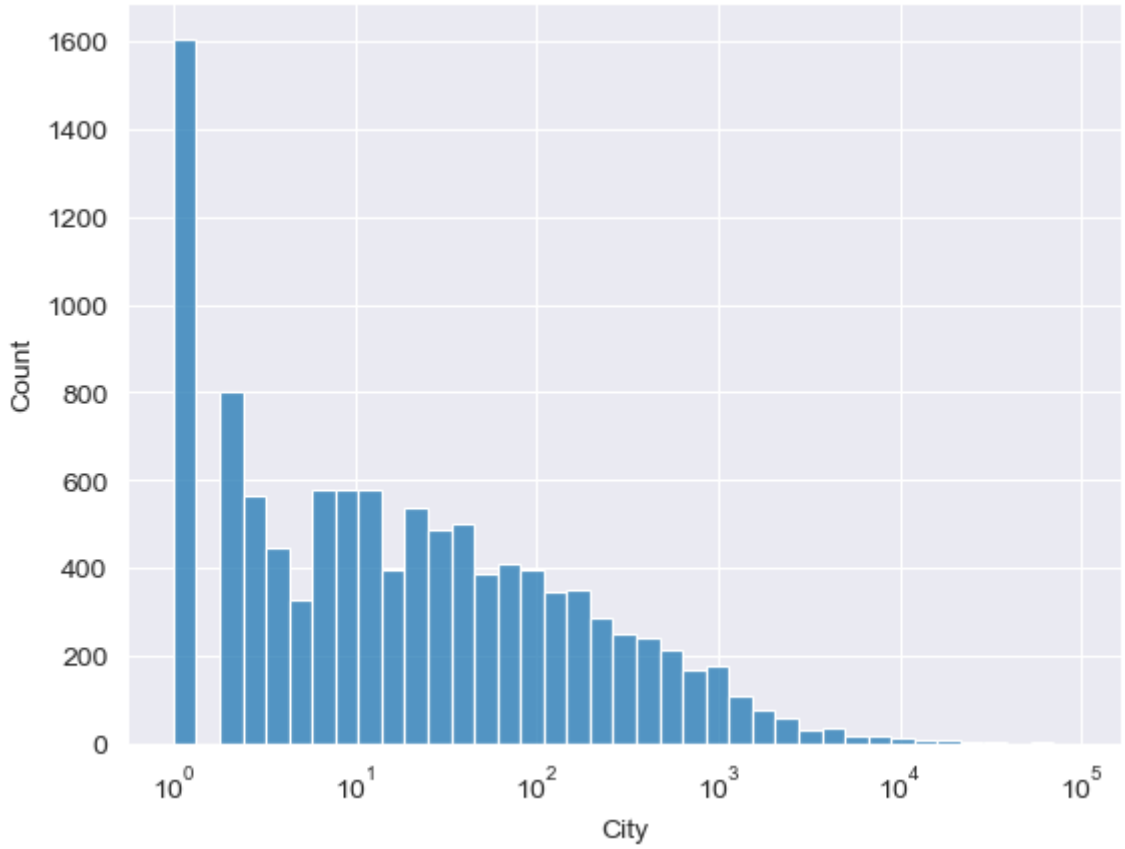
In [106…
```
sns.histplot(low_accident_cities)
#no. of accidents per city decreases exponentialy
```

Out[106]:
```
<Axes: xlabel='City', ylabel='Count'>
```



In [103…
```
sns.histplot(cities_by_accident, log_scale=True)
```

Out[103]:
```
<Axes: xlabel='City', ylabel='Count'>
```



In [104…
```
cities_by_accident[cities_by_accident == 1]
```

Out[104]:
```
Bouton             1
Guys Mills         1
Vevay              1
Walsh              1
Antimony           1
                  ..
Robert             1
West Burlington    1
Osawatomie         1
Bean Station       1
Bosler             1
Name: City, Length: 1605, dtype: int64
```

In [105…
```
#over 1600 cities have reported just 1 accident
```

Start time

In [113…
```
df.Start_Time
```

Out[113]:
```
0          2016-02-08 05:46:00
1          2016-02-08 06:07:59
2          2016-02-08 06:49:27
3          2016-02-08 07:23:34
4          2016-02-08 07:39:07
                   ...
2879980    2018-04-17 11:16:45
2879981    2018-04-17 11:20:23
2879982    2018-04-17 12:41:10
2879983    2018-04-17 14:52:45
2879984    2018-04-17 14:59:10
Name: Start_Time, Length: 2879985, dtype: object
```

In [114…
```
df.Start_Time[0]
```

Out[114]:
```
'2016-02-08 05:46:00'
```

In [115…
```
#this is a string we will change into date format
```

In [5]:
```
df.Start_Time=pd.to_datetime(df.Start_Time)
```

In [6]:
```
df.Start_Time[0]
```

Out[6]:
```
Timestamp('2016-02-08 05:46:00')
```

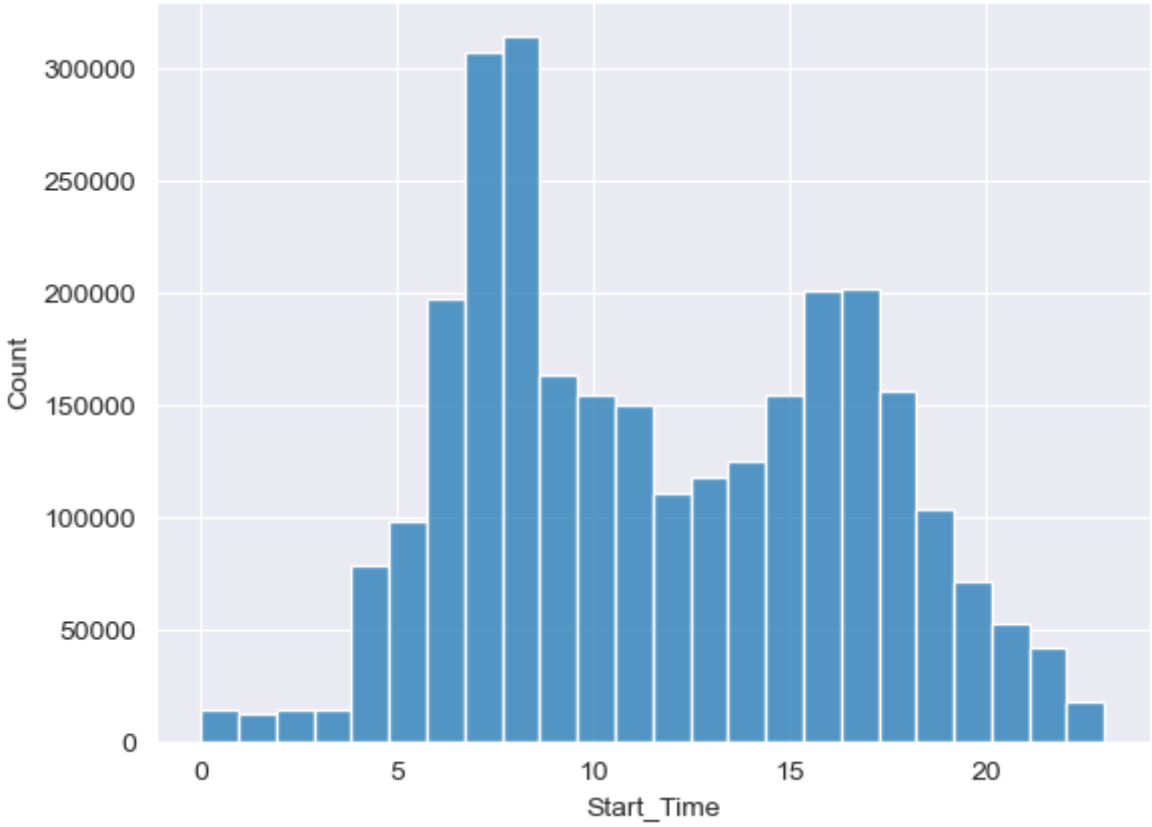
In [119…
```
#we cant plot with Start_Time column because it is just a timestamp
#so will pull out the pieces of information form it
```

In [123…
```
df.Start_Time.dt.hour
```

Out[123]:
```
0           5
1           6
2           6
3           7
4           7
           ..
2879980    11
2879981    11
2879982    12
2879983    14
2879984    14
Name: Start_Time, Length: 2879985, dtype: int64
```

In [137…
```
sns.histplot(df.Start_Time.dt.hour, bins=24)
```

Out[137]:
```
<Axes: xlabel='Start_Time', ylabel='Count'>
```

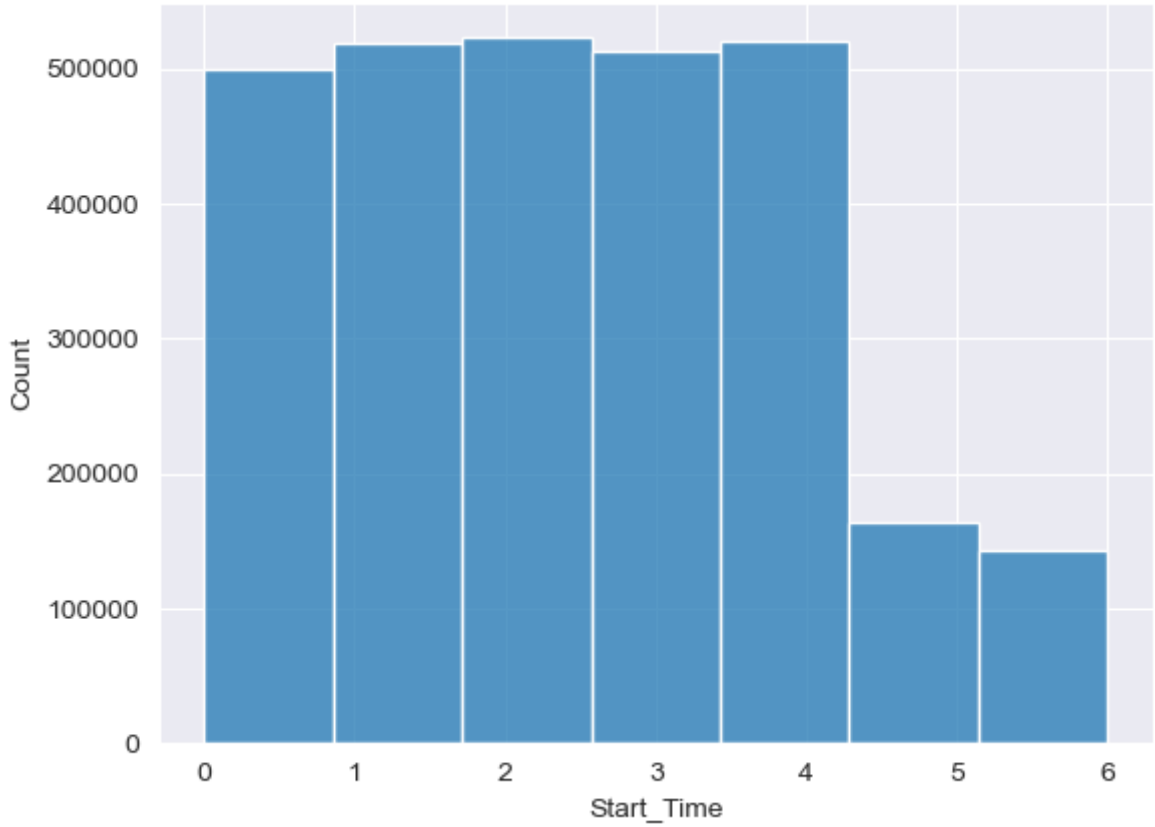


In [139…
```
#high percentage of accidents occur between 6am to 10am(probably people in a hurry
#next high occurence is between 3pm to 7pm
```

dayofweek

In [148…
```
sns.histplot(df.Start_Time.dt.dayofweek, bins=7)
```

Out[148]:
```
<Axes: xlabel='Start_Time', ylabel='Count'>
```



In [3]:
```
#is the distribution of accidents by hour is the same on weekends as weekdays.
```

In [7]:
```
df.Start_Time.dt.dayofweek == 6
```

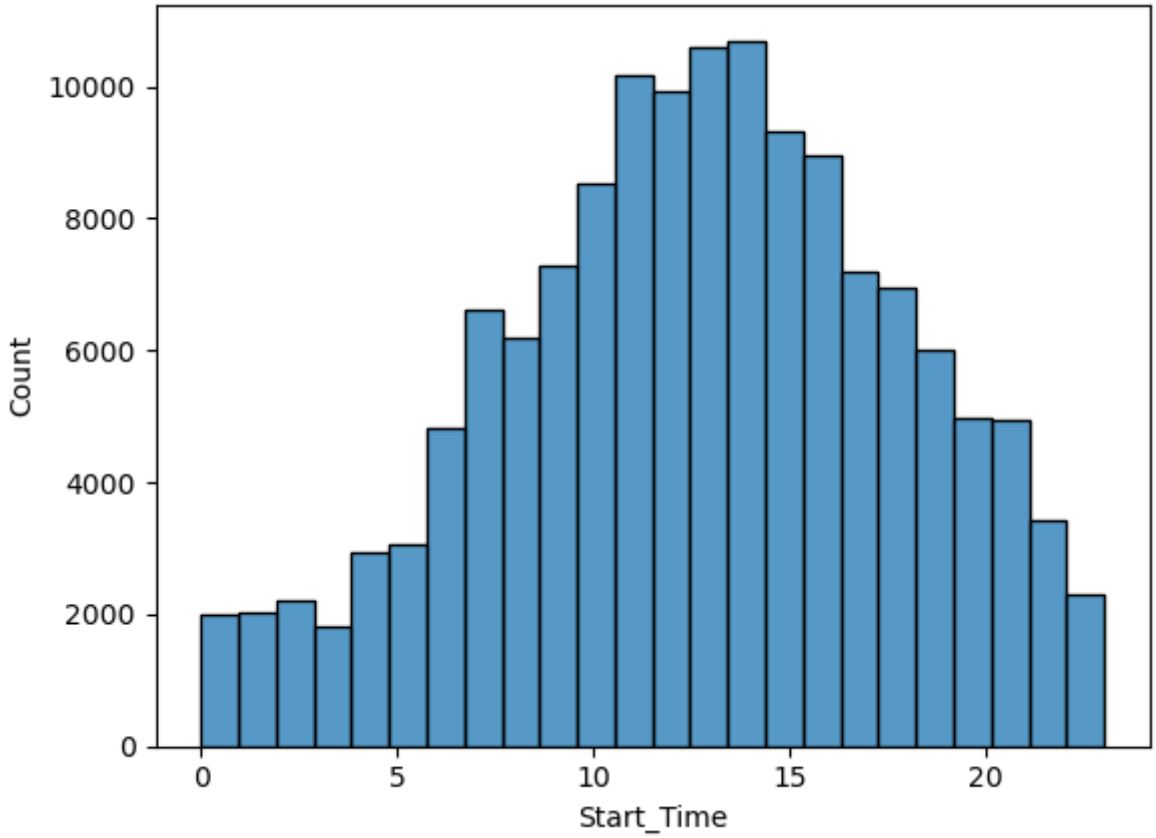
Out[7]:
```
0          False
1          False
2          False
3          False
4          False
           ...
2879980    False
2879981    False
2879982    False
2879983    False
2879984    False
Name: Start_Time, Length: 2879985, dtype: bool
```

In [9]:
```
sundays_start_time = df.Start_Time[df.Start_Time.dt.dayofweek == 6]
sundays_start_time
```

Out[9]:
```
135       2016-02-14 10:13:00
136       2016-02-14 10:49:23
137       2016-02-14 10:41:57
138       2016-02-14 18:15:23
139       2016-02-14 19:17:01
                  ...
2879371   2018-04-15 21:58:18
2879372   2018-04-15 22:26:34
2879373   2018-04-15 22:36:56
2879374   2018-04-15 22:39:26
2879375   2018-04-15 23:53:20
Name: Start_Time, Length: 142917, dtype: datetime64[ns]
```

In [12]:
```
sns.histplot(sundays_start_time.dt.hour, bins=24,kde=False)
```

Out[12]:
```
<Axes: xlabel='Start_Time', ylabel='Count'>
```

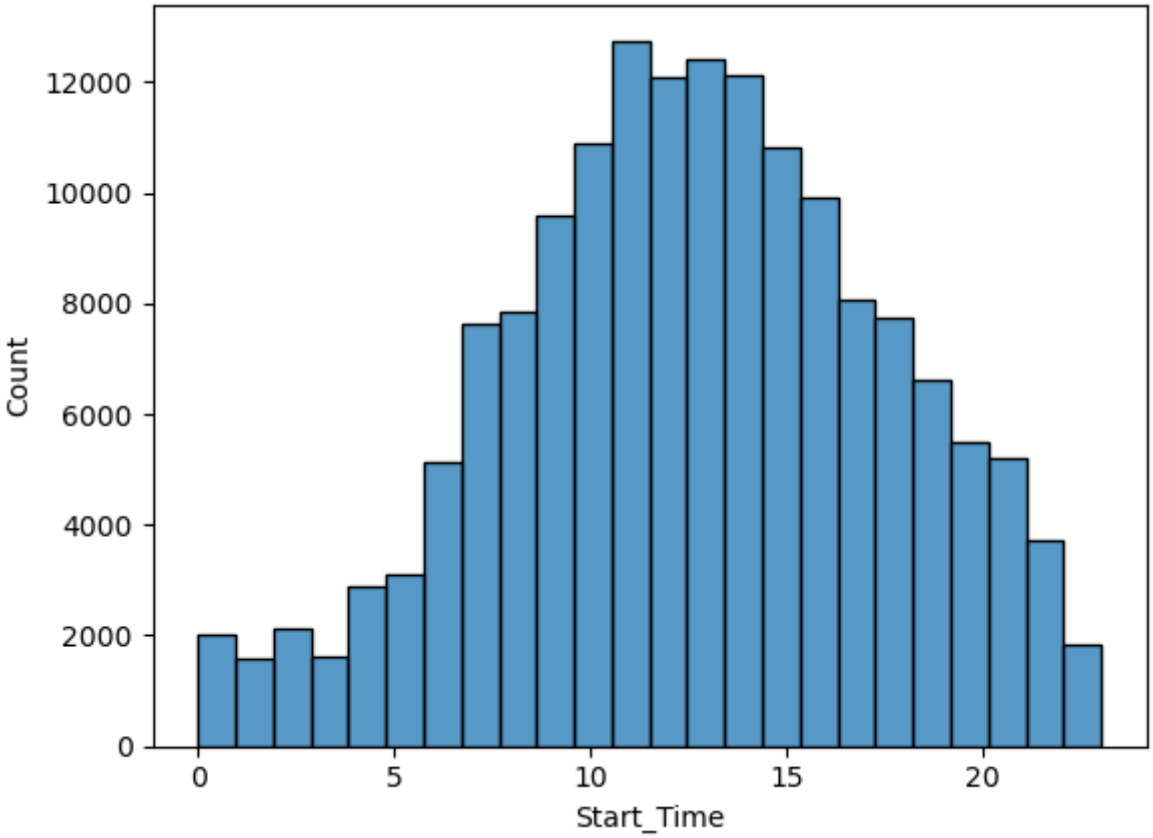


In [13]:
```
saturdays_start_time = df.Start_Time[df.Start_Time.dt.dayofweek == 5]
saturdays_start_time
```

Out[13]:
```
129       2016-02-13 11:05:00
130       2016-02-13 11:05:21
131       2016-02-13 11:17:01
132       2016-02-13 11:25:42
133       2016-02-13 12:56:31
                  ...
2876767   2018-04-14 23:11:17
2876768   2018-04-14 23:16:24
2876769   2018-04-14 23:36:26
2876770   2018-04-14 23:37:35
2876771   2018-04-14 23:39:07
Name: Start_Time, Length: 163222, dtype: datetime64[ns]
```

In [14]:
```
sns.histplot(saturdays_start_time.dt.hour, bins=24,kde=False)
```

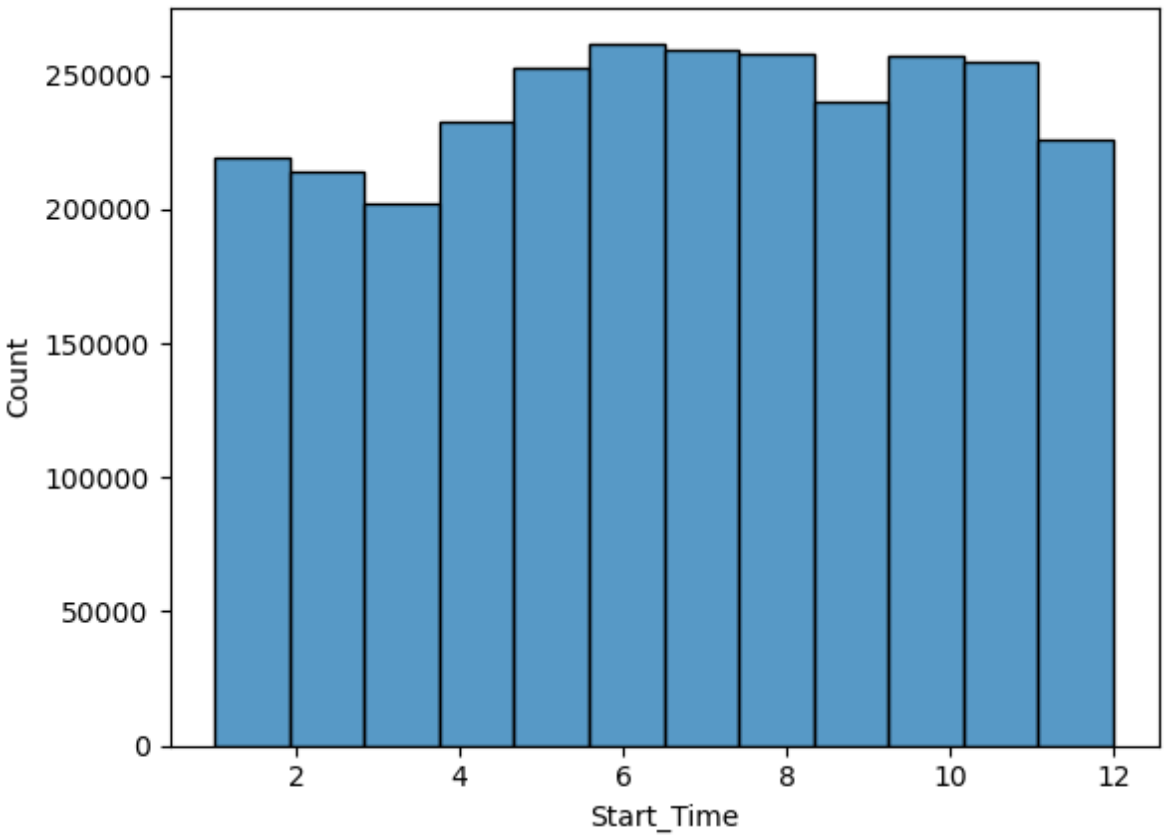
Out[14]:
```
<Axes: xlabel='Start_Time', ylabel='Count'>
```



In [15]:
```
#on weekends the peak occurs between 10am to 4pm unlike weekdays
```

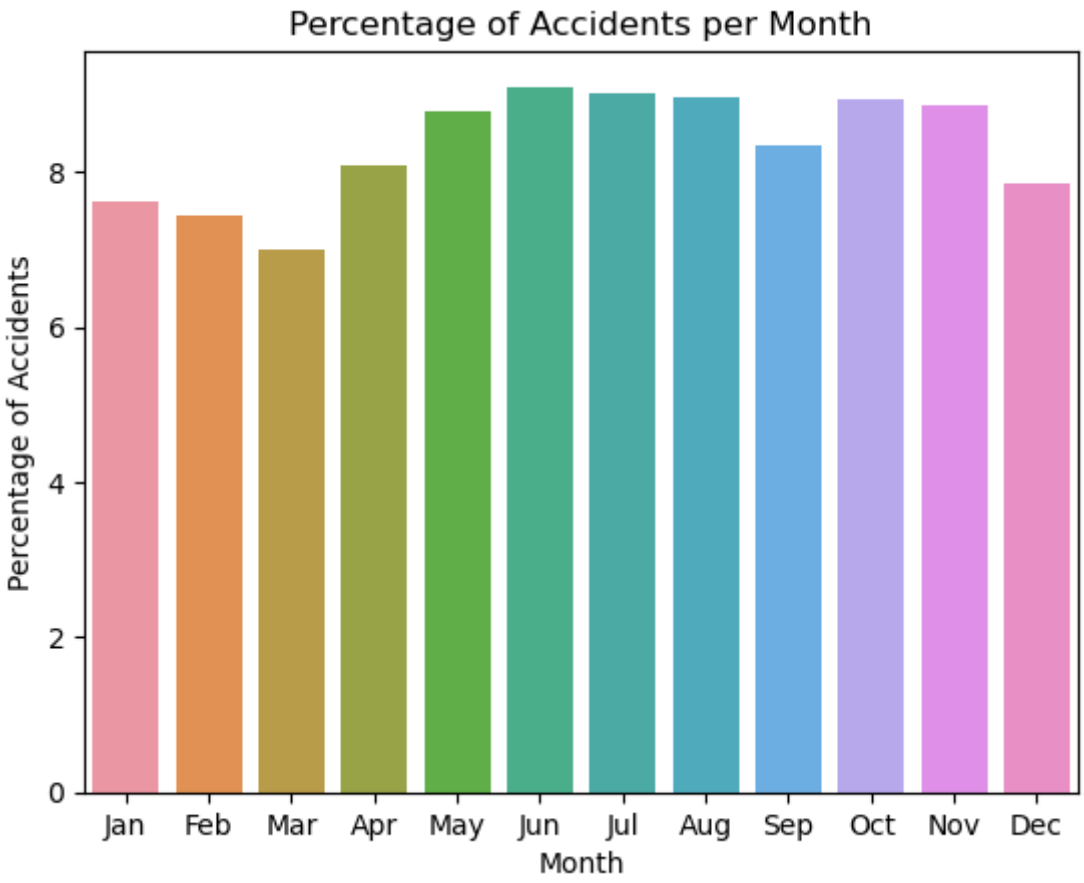
In [21]:
```
sns.histplot(df.Start_Time.dt.month, bins=12,kde=False)
```

Out[21]:
```
<Axes: xlabel='Start_Time', ylabel='Count'>
```



In [23]:
```
accidents_per_month = df.Start_Time.dt.month.value_counts()
```

```
# Calculate the total number of accidents
total_accidents = accidents_per_month.sum()

# Calculate the percentage of accidents per month
percentage_per_month = (accidents_per_month / total_accidents) * 100

# Create a histogram plot
sns.barplot(x=percentage_per_month.index, y=percentage_per_month.values)
plt.xlabel('Month')
plt.ylabel('Percentage of Accidents')
plt.title('Percentage of Accidents per Month')
plt.xticks(range(0, 12), ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug',
plt.show()
```

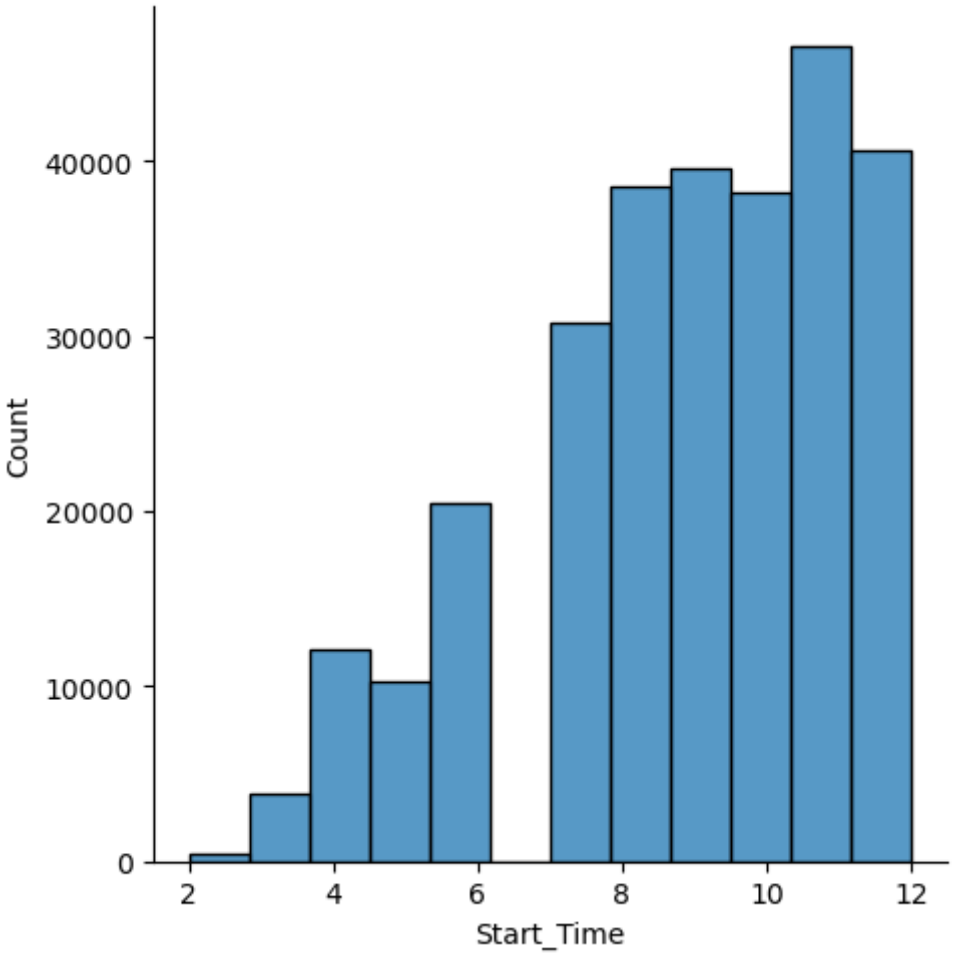


In [26]:
```
accidents_per_month
```

Out[26]:
```
6     262044
7     259529
8     258240
10    257325
11    255107
5     252825
9     240419
4     232805
12    226223
1     219225
2     214339
3     201904
Name: Start_Time, dtype: int64
```

In [44]:
```
df_2016=df[df.Start_Time.dt.year == 2016]
sns.displot(df_2016.Start_Time.dt.month, bins=12,kde=False)
```

Out[44]:
```
<seaborn.axisgrid.FacetGrid at 0x1d9a413df00>
```

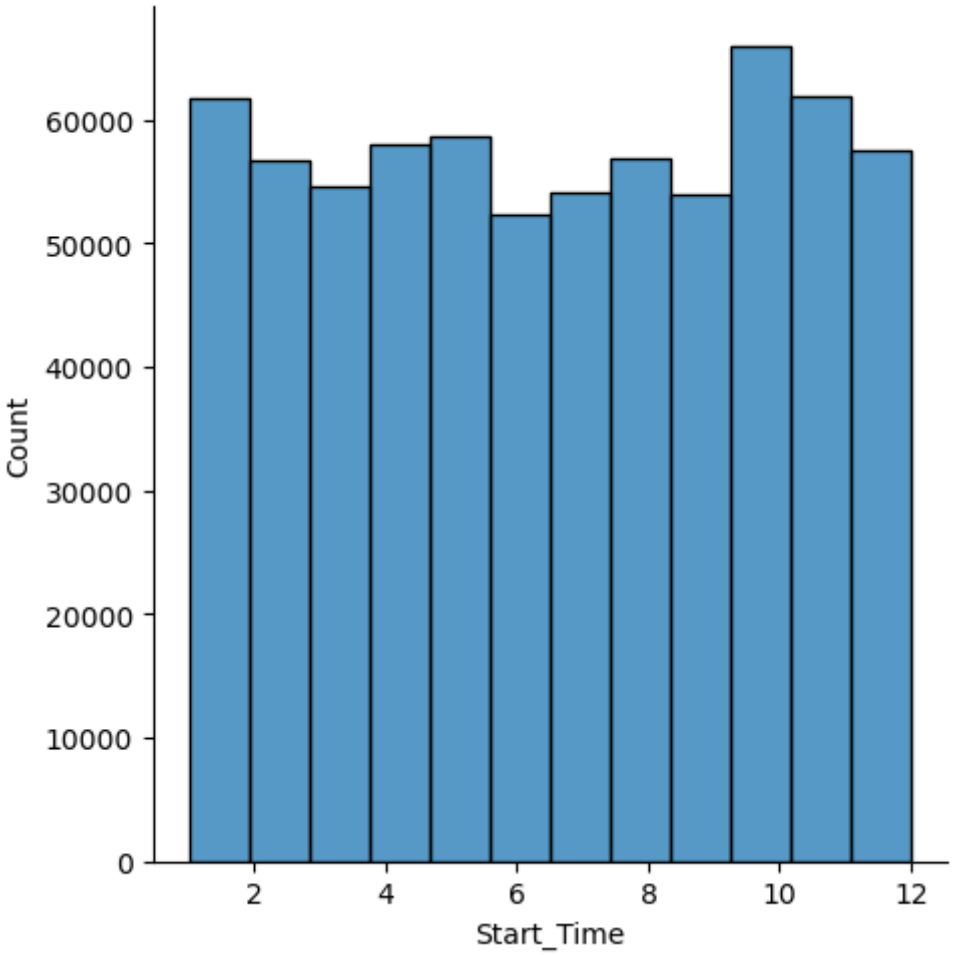


In [46]:
```
#much data is missing for 2016
```

In [47]:
```
#lets look at the sources of data
```

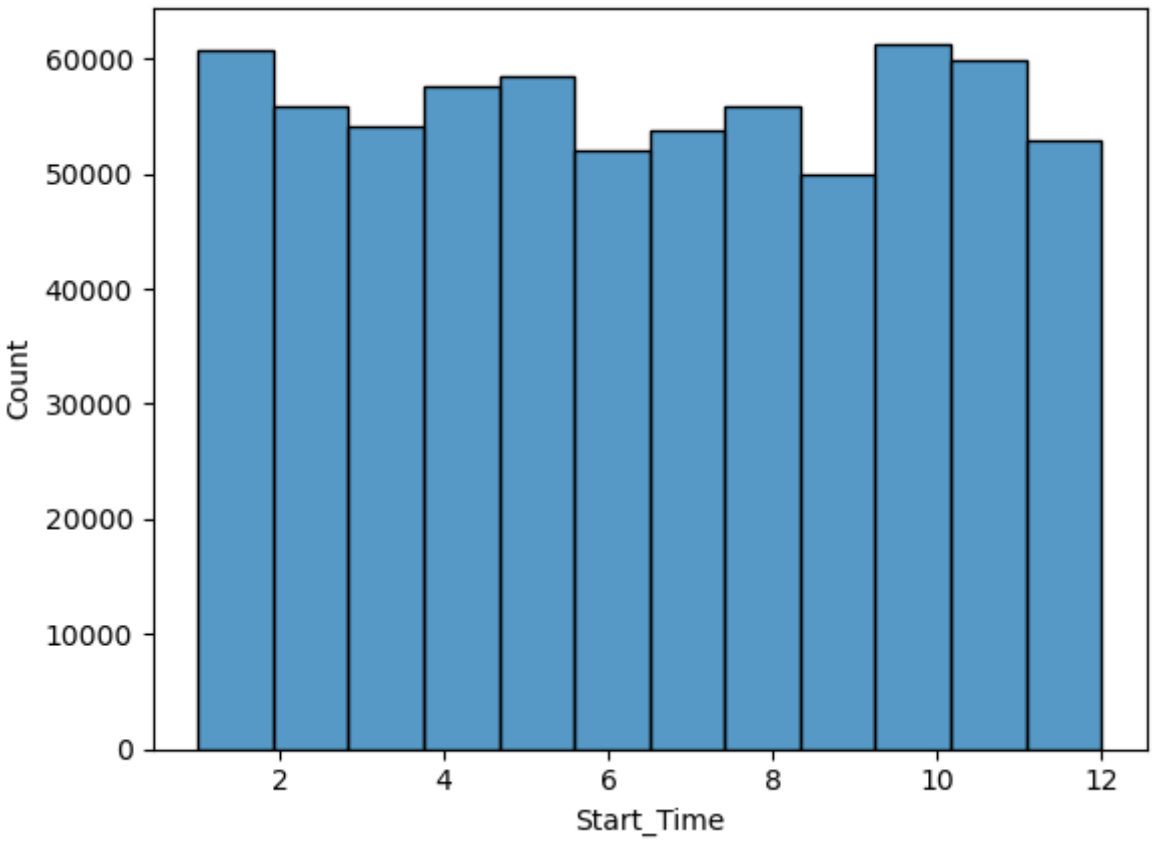
In [48]:
```
df_2019=df[df.Start_Time.dt.year == 2019]
sns.displot(df_2019.Start_Time.dt.month, bins=12,kde=False)
```

Out[48]:
```
<seaborn.axisgrid.FacetGrid at 0x1d9a7f9bee0>
```



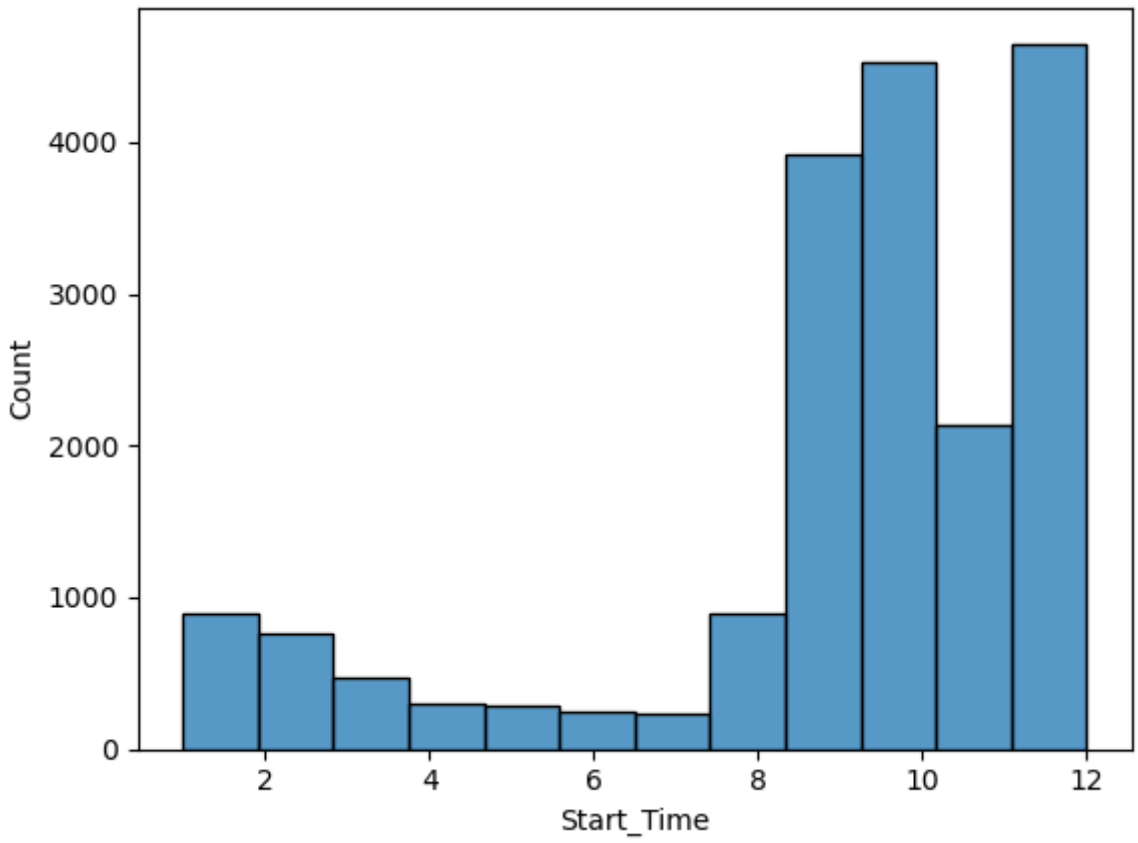
In [61]:
```
df_2019=df[df.Start_Time.dt.year == 2019]
df_2019_Bing=df_2019[df_2019.Source == 'Source2']
sns.histplot(df_2019_Bing.Start_Time.dt.month, bins=12,kde=False)
```

Out[61]:
```
<Axes: xlabel='Start_Time', ylabel='Count'>
```



In [62]:
```
df_2019=df[df.Start_Time.dt.year == 2019]
df_2019_Bing=df_2019[df_2019.Source == 'Source3']
sns.histplot(df_2019_Bing.Start_Time.dt.month, bins=12,kde=False)
```

Out[62]:
```
<Axes: xlabel='Start_Time', ylabel='Count'>
```



In [63]:
```
#there seems to an issue with the source 3 data
```

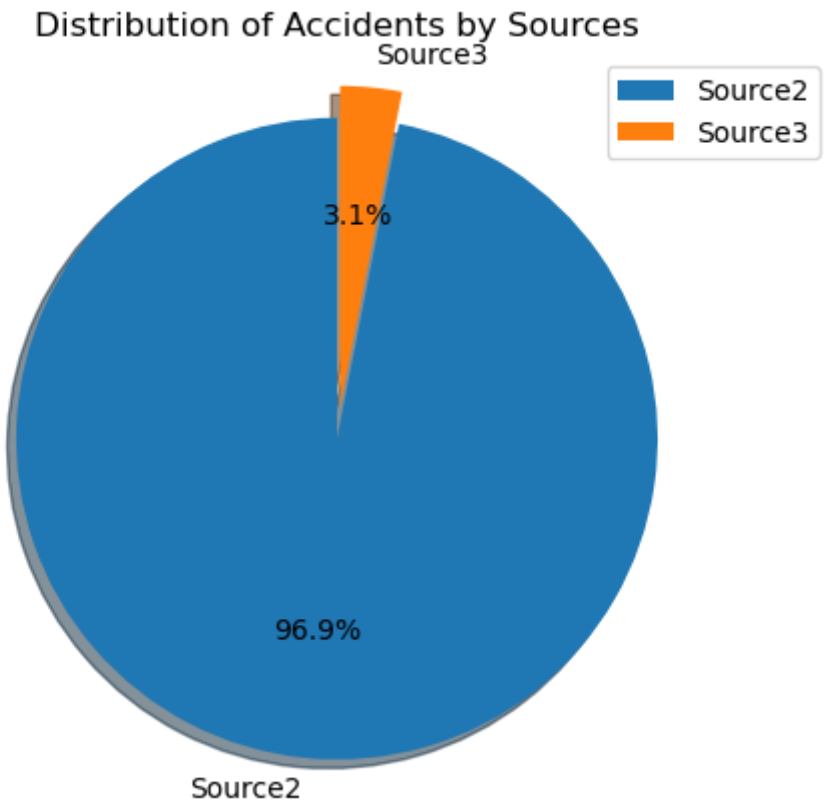
In [65]:
```
df.Source.unique()
```

Out[65]:
```
array(['Source2', 'Source3'], dtype=object)
```

In [68]:
```
df.Source.value_counts()
```

Out[68]:
```
Source2    2790802
Source3      89183
Name: Source, dtype: int64
```

In [81]:
```
source_cnt=df.Source.value_counts()
labels=['Source2','Source3']
explode = (0.1, 0)


plt.pie(source_cnt, startangle=90, labels=labels, explode=explode, autopct='%1.1f%%
plt.axis('equal')


plt.legend(labels, loc="best")


plt.title('Distribution of Accidents by Sources')

plt.show()
```



In [82]:
```
#excluding source3 data seems to have issues
```

Start Latitude & Longitude

In [84]:
```
df.Start_Lat
```

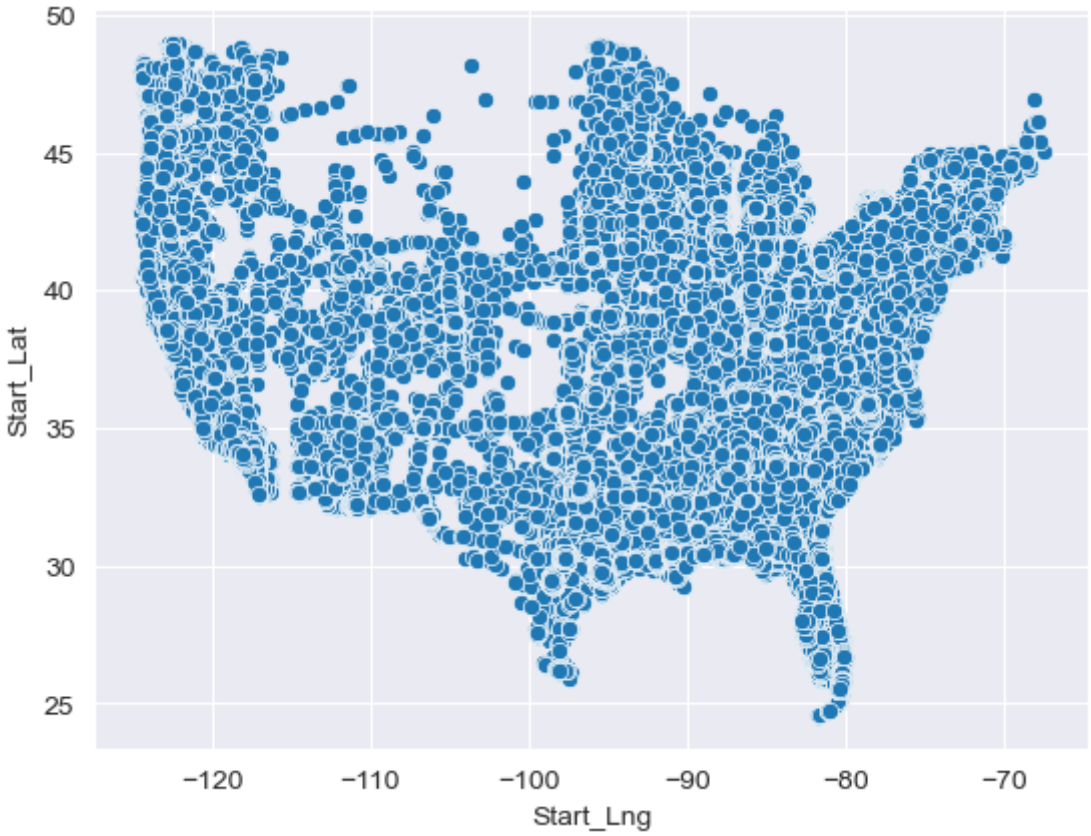
Out[84]:
```
0          39.865147
1          39.928059
2          39.063148
3          39.747753
4          39.627781
             ...
2879980    39.680641
2879981    38.247238
2879982    39.766941
2879983    39.993881
2879984    40.004913
Name: Start_Lat, Length: 2879985, dtype: float64
```

In [85]:
```
df.Start_Lng
```

Out[85]:
```
0         -84.058723
1         -82.831184
2         -84.032608
3         -84.205582
4         -84.188354
             ...
2879980   -86.082512
2879981   -85.700569
2879982   -86.142792
2879983   -85.843201
2879984   -85.772736
Name: Start_Lng, Length: 2879985, dtype: float64
```

In [92]:
```
sns.set_style('darkgrid')
sns.scatterplot(x=df.Start_Lng,y=df.Start_Lat)
```

Out[92]:
```
<Axes: xlabel='Start_Lng', ylabel='Start_Lat'>
```

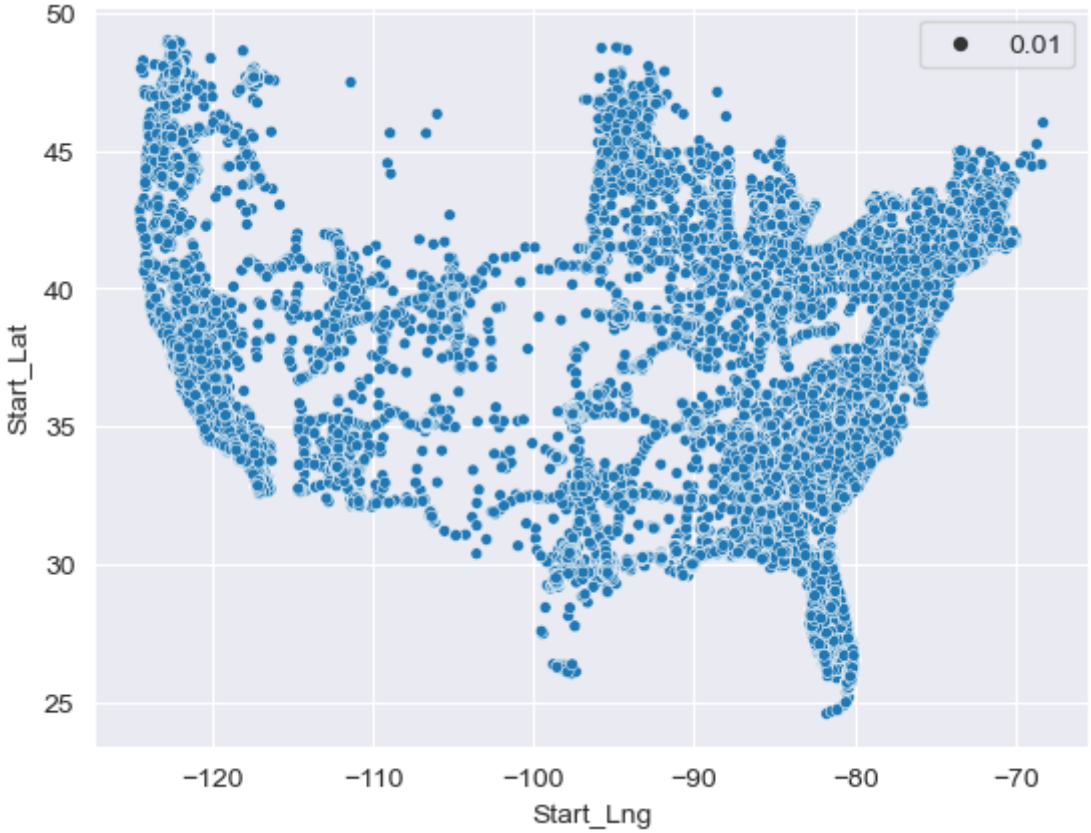


In [87]:
```
#lets plot on the smaller sample from the data
```

In [93]:
```
sample_df = df.sample(int(0.1*len(df)))
```

In [94]:
```
sns.scatterplot(x=sample_df.Start_Lng, y=sample_df.Start_Lat,size=0.01)
```

Out[94]:
```
<Axes: xlabel='Start_Lng', ylabel='Start_Lat'>
```



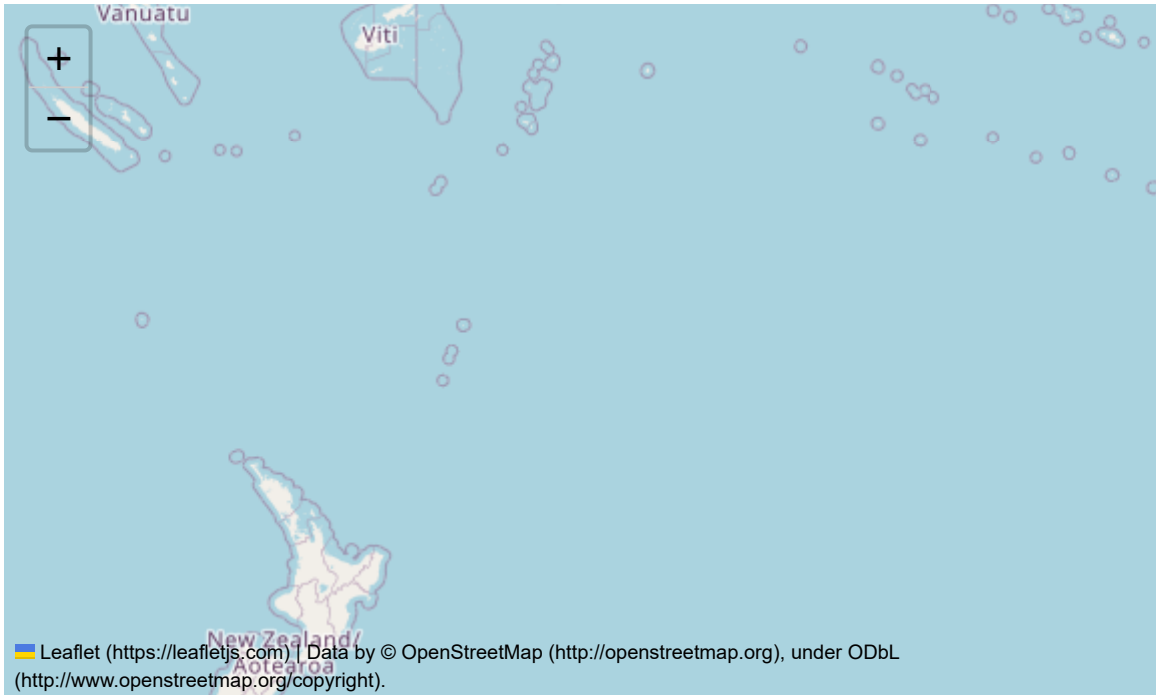
In [96]:
```
!pip install folium
```

```
Collecting folium
  Downloading folium-0.14.0-py2.py3-none-any.whl (102 kB)
     ---------------------------------------- 102.3/102.3 kB 979.2 kB/s eta 0:00:00
Requirement already satisfied: requests in c:\users\dell\anaconda3\lib\site-packag
es (from folium) (2.28.1)
Requirement already satisfied: numpy in c:\users\dell\anaconda3\lib\site-packages
(from folium) (1.23.5)
Collecting branca>=0.6.0
  Downloading branca-0.6.0-py3-none-any.whl (24 kB)
Requirement already satisfied: jinja2>=2.9 in c:\users\dell\anaconda3\lib\site-pac
kages (from folium) (3.1.2)
Requirement already satisfied: MarkupSafe>=2.0 in c:\users\dell\anaconda3\lib\site
-packages (from jinja2>=2.9->folium) (2.1.1)
Requirement already satisfied: charset-normalizer<3,>=2 in c:\users\dell\anaconda3
\lib\site-packages (from requests->folium) (2.0.4)
Requirement already satisfied: idna<4,>=2.5 in c:\users\dell\anaconda3\lib\site-pa
ckages (from requests->folium) (3.4)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\dell\anaconda3\lib\s
ite-packages (from requests->folium) (2022.12.7)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in c:\users\dell\anaconda3\li
b\site-packages (from requests->folium) (1.26.14)
Installing collected packages: branca, folium
Successfully installed branca-0.6.0 folium-0.14.0
```

In [1]:
```
import folium
```

In [2]:
```
folium.Map()
```

Out[2]:



In [100…

```
lat, lon = df.Start_Lat[0],df.Start_Lng[0]
lat, lon
```

Out[100]:

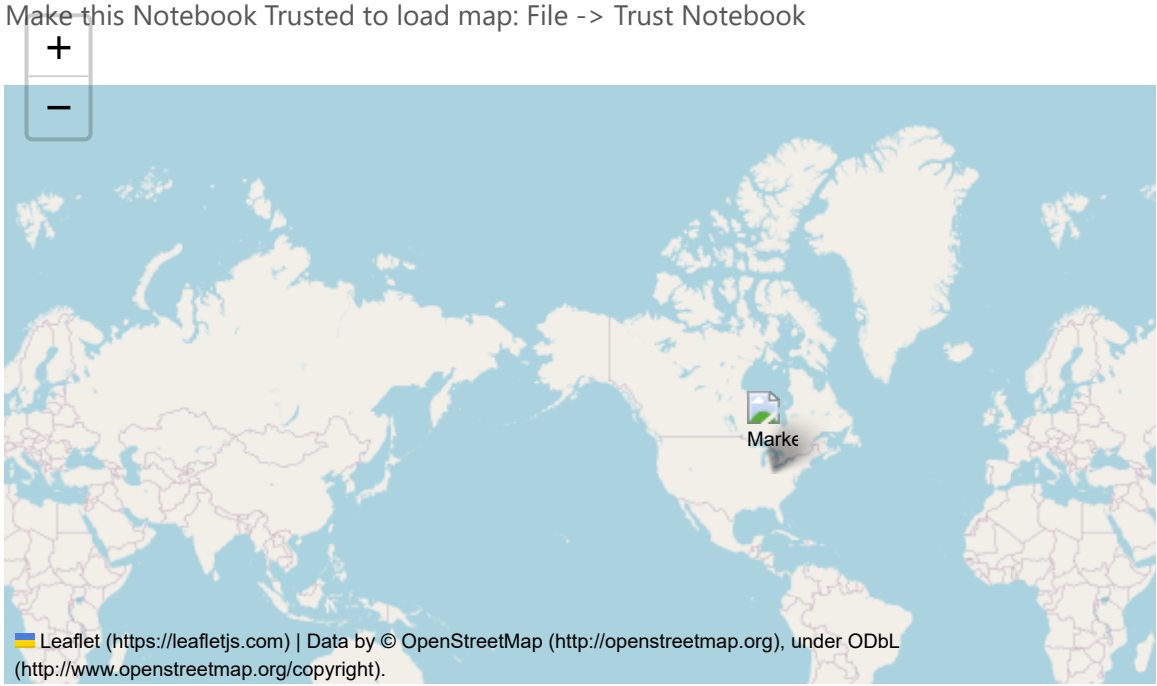
```
(39.865147, -84.058723)
```

In [101…

```
map=folium.Map()
marker=folium.Marker((lat,lon))
marker.add_to(map)
map
```

Out[101]:

Make this Notebook Trusted to load map: File -> Trust Notebook



In [ ]:

```
map=folium.Map()
```

In [104…

```
from folium import plugins
from folium.plugins import HeatMap
```

In [107…

```
list(zip(list(df.Start_Lat),list(df.Start_Lng)))
```

```
(37.750488, -121.379982),
(37.37682, -121.941536),
(37.38965200000001, -122.163582),
(37.70306, -122.07856),
(37.323593, -121.940826),
(37.690514, -121.918709),
(37.656654, -121.901588),
(37.753693, -122.151398),
(38.227737, -122.120537),
(37.769375, -122.405533),
(37.696819, -122.071869),
(37.8745, -122.306038),
(37.752502, -122.403008),
(37.7883, -121.300407),
(38.860706, -121.300247),
(37.86365900000001, -121.218956),
(37.54723, -122.37265),
(37.33442700000001, -121.936485),
(36.950603, -121.523094),
(38.660831, -121.337288),
(38.222748, -122.12883),
(37.343258, -121.846283),
(37.777836, -121.317451),
(38.722092, -121.22583799999998),
(38.246796, -122.627808),
(38.661053, -121.359779),
(37.813274, -121.034325),
(37.731277, -122.435219),
(37.315239, -121.914902),
(37.328579, -121.871277),
(37.882004, -121.641479),
(38.835583, -121.168533),
(37.933201, -122.046196),
(37.657974, -121.902954),
(38.562939, -121.641922),
(38.022778, -121.965698),
(37.656654, -121.901588),
(38.690273, -121.392136),
(38.68111, -121.333244),
(38.653061, -121.070541),
...]
```

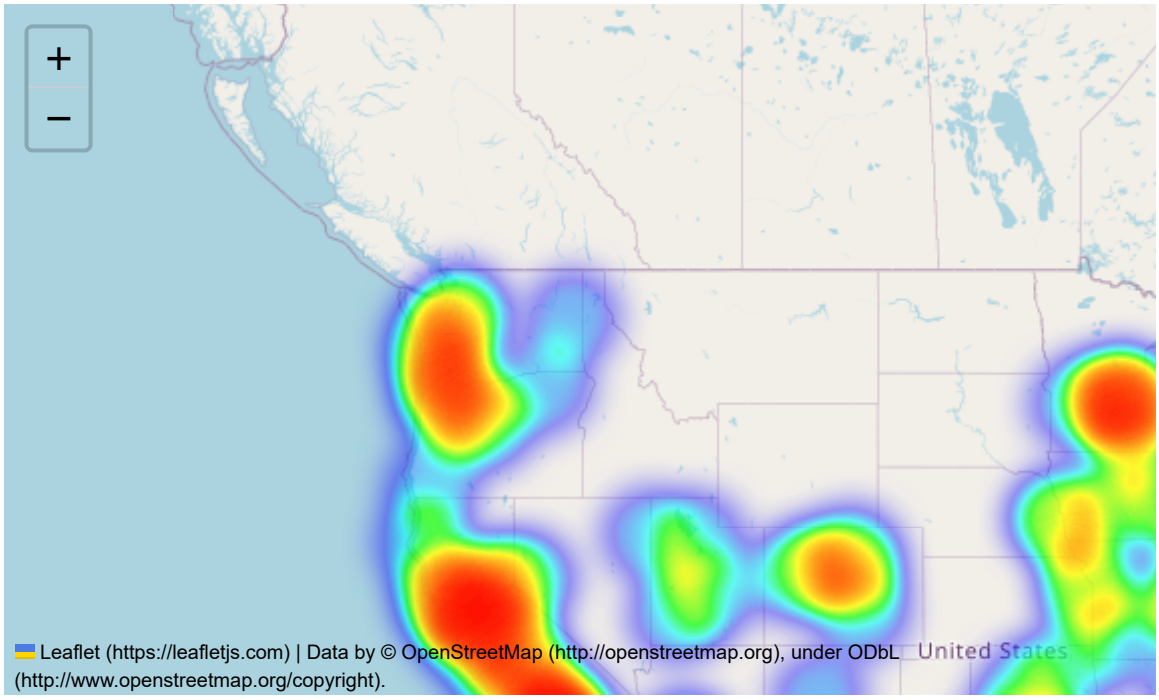
In [115…

```
sample_df = df.sample(int(0.001*len(df)))
lat_lon_pairs = list(zip(list(sample_df.Start_Lat),list(sample_df.Start_Lng)))
```

In [116…

```
map=folium.Map()
HeatMap(lat_lon_pairs).add_to(map)
map
```

Out[116]:



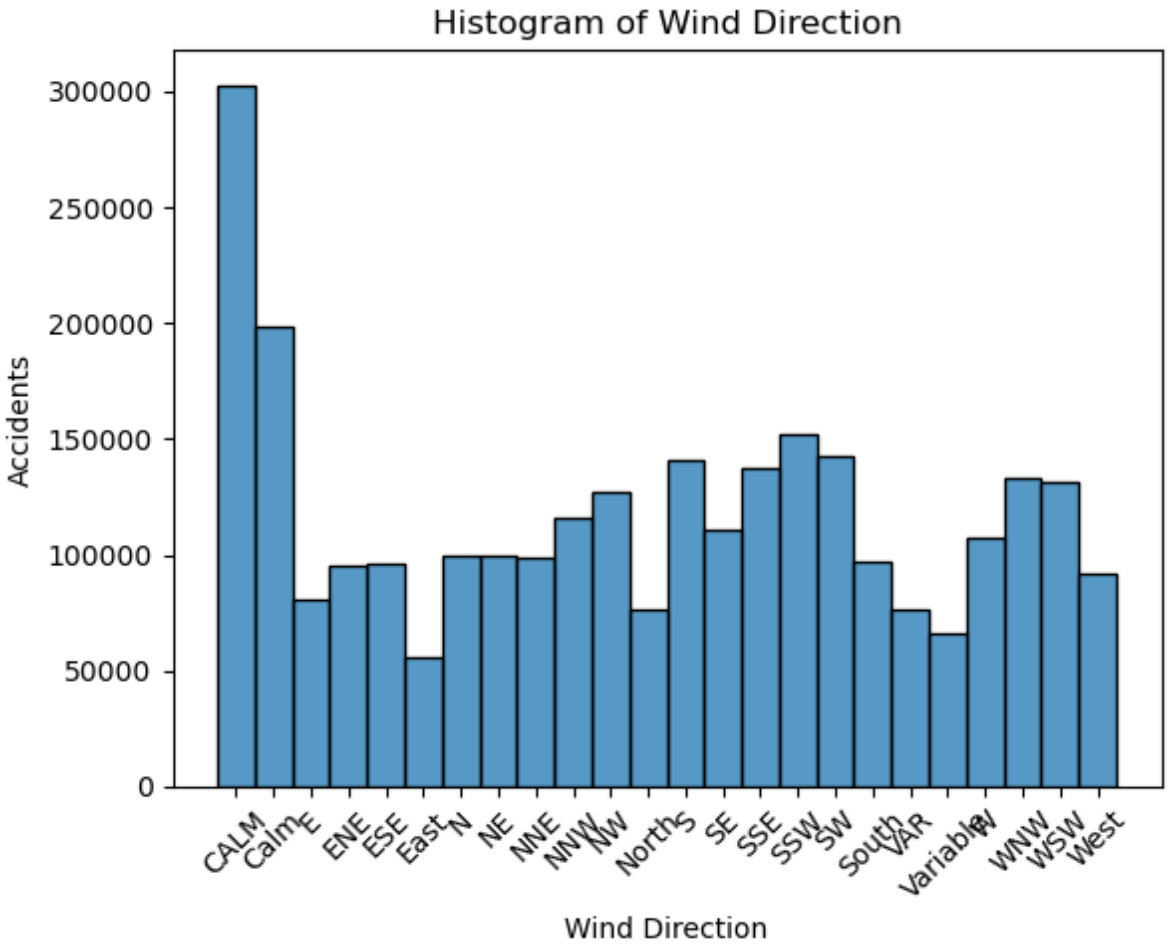
In [6]:
```
df.columns
```

Out[6]:
```
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 [16]:
```
df.Wind_Direction.value_counts()
```

Out[16]:
```
CALM        302485
Calm        198154
SSW         151767
SW          142251
S           141047
SSE         137418
WNW         133248
WSW         131687
NW          126719
NNW         115991
SE          110786
W           107134
N            99526
NE           99212
NNE          98990
South        97392
ESE          96140
ENE          95677
West         91499
E            80999
North        76362
VAR          76120
Variable     65679
East         56019
Name: Wind_Direction, dtype: int64
```

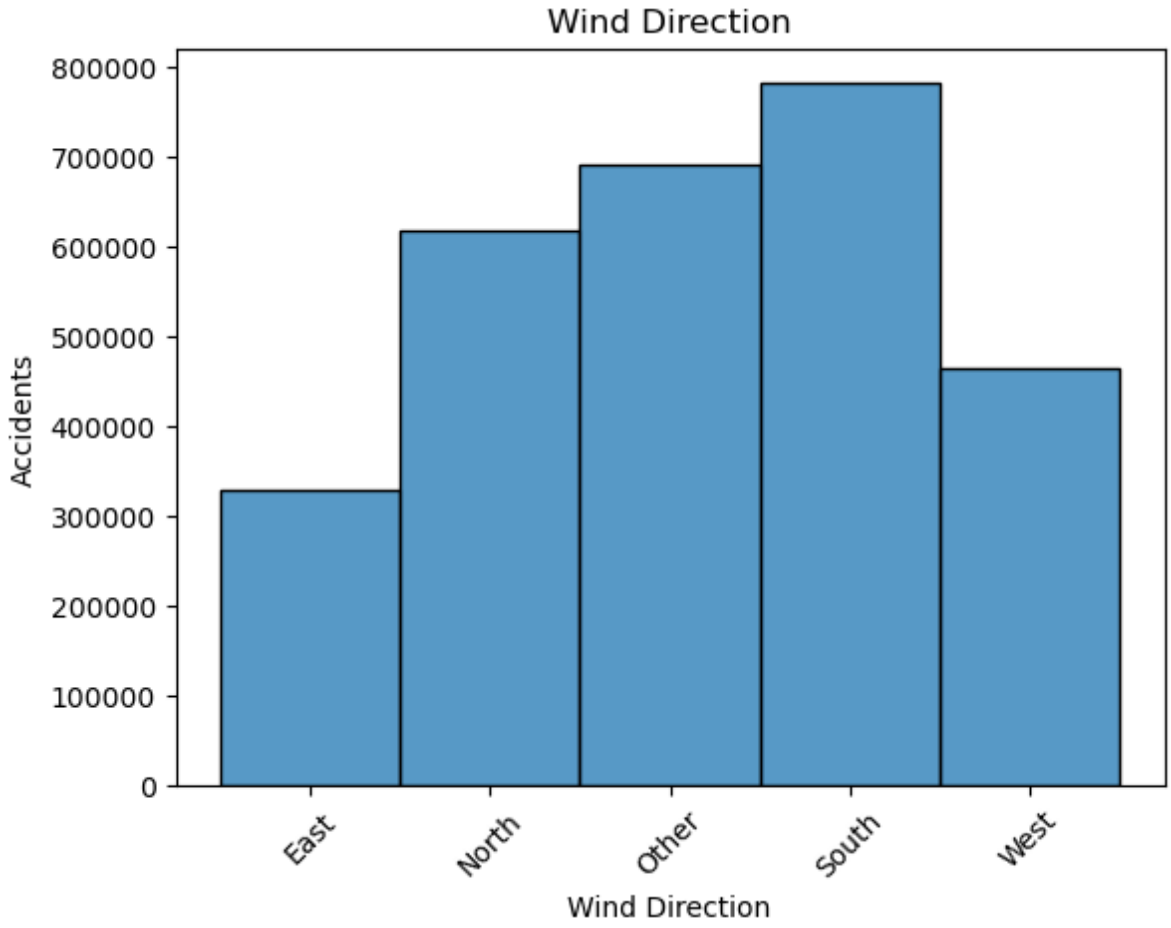
In [20]:
```
sns.histplot(df.Wind_Direction.sort_values())
plt.xlabel('Wind Direction')
plt.ylabel('Accidents')
plt.title('Histogram of Wind Direction')
plt.xticks(rotation=45)
plt.show()
```



In [ ]:
```
#Variability in Naming: There is variability in how wind directions are named,
#such as "Calm" and "CALM" or "Variable" and "VAR."
#This inconsistency might be due to different data entry practices or conventions.
```

In [3]:
```
# Grouping wind directions into cardinal or intermediate directions
def bin_wind_direction(Wind_Direction):
    if Wind_Direction in ['N', 'NE', 'NNE', 'NW', 'NNW', 'North']:
        return 'North'
    elif Wind_Direction in ['S', 'SE', 'SSE', 'SW', 'SSW', 'South']:
        return 'South'
    elif Wind_Direction in ['E', 'ENE', 'ESE', 'East']:
        return 'East'
    elif Wind_Direction in ['W', 'WNW', 'WSW', 'West']:
        return 'West'
    else:
        return 'Other'

df['Wind_Direction_Binned'] = df.Wind_Direction.apply(bin_wind_direction)
```

In [4]:
```
sns.histplot(df.Wind_Direction_Binned.sort_values())
plt.xlabel('Wind Direction')
plt.ylabel('Accidents')
plt.title('Wind Direction')
plt.xticks(rotation=45)
plt.show()
```



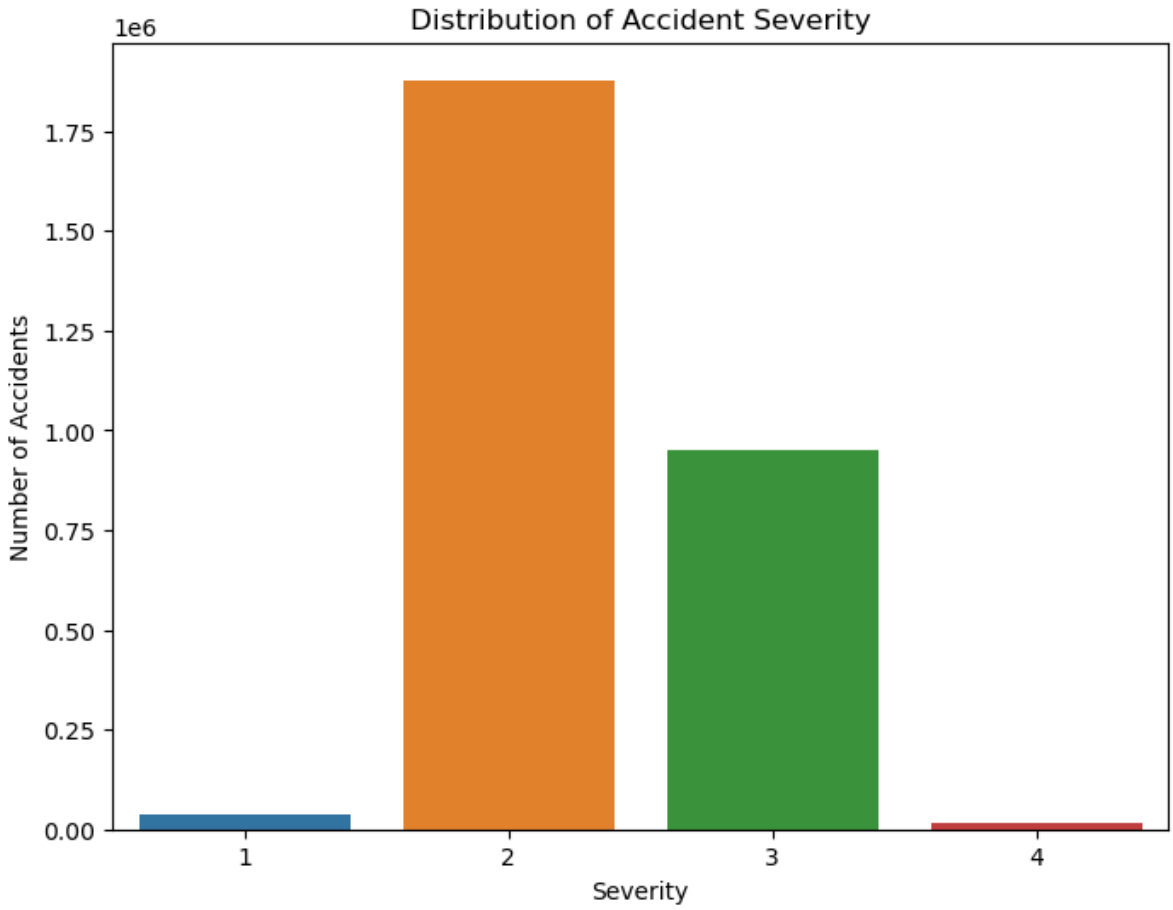
In [5]:
```
#most number of accidents happen when the wind have cardinal or intermediate direct
```

Severity Analysis

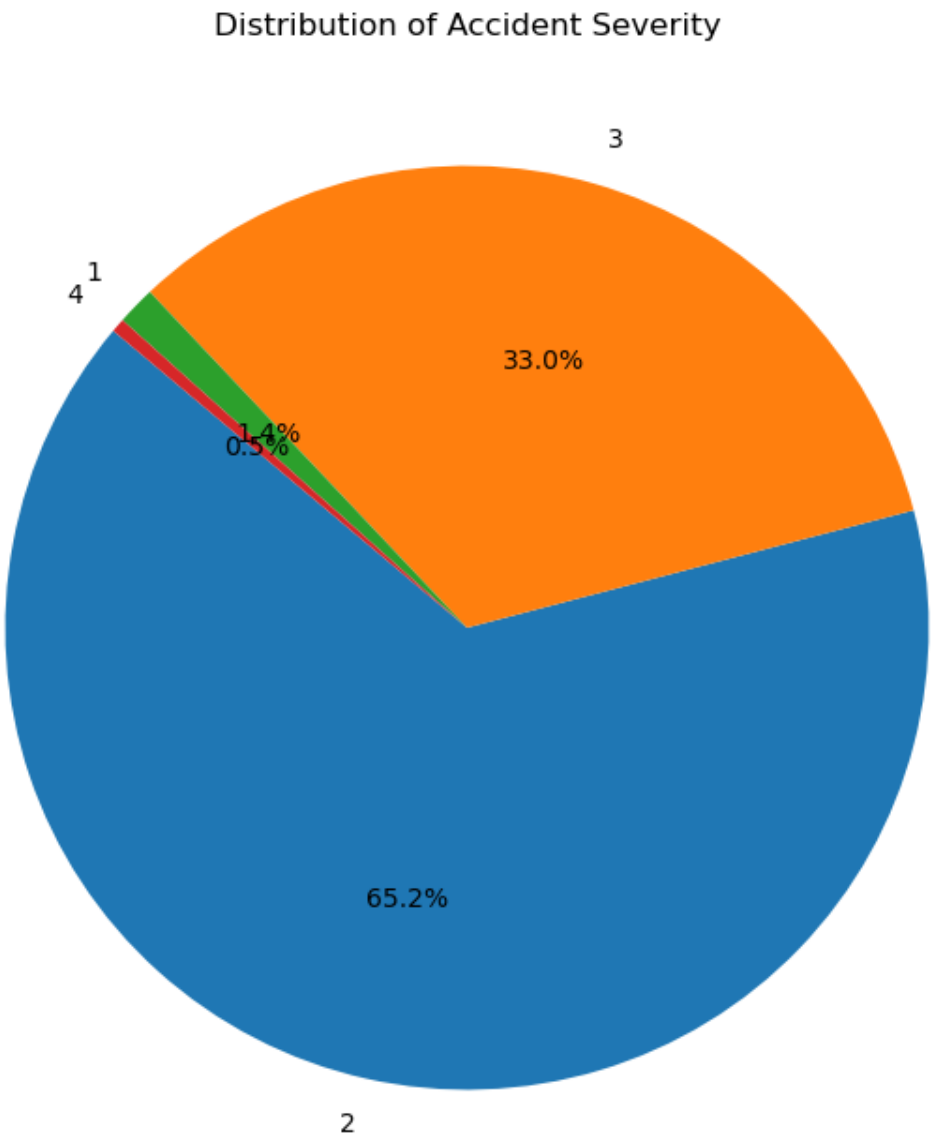
In [7]:
```
df.Severity.unique()
```

Out[7]:
```
array([3, 2, 1, 4], dtype=int64)
```

In [12]:
```
# Severity Distribution
plt.figure(figsize=(8, 6))
sns.countplot(data=df, x='Severity')
plt.xlabel('Severity')
plt.ylabel('Number of Accidents')
plt.title('Distribution of Accident Severity')
plt.show()
```

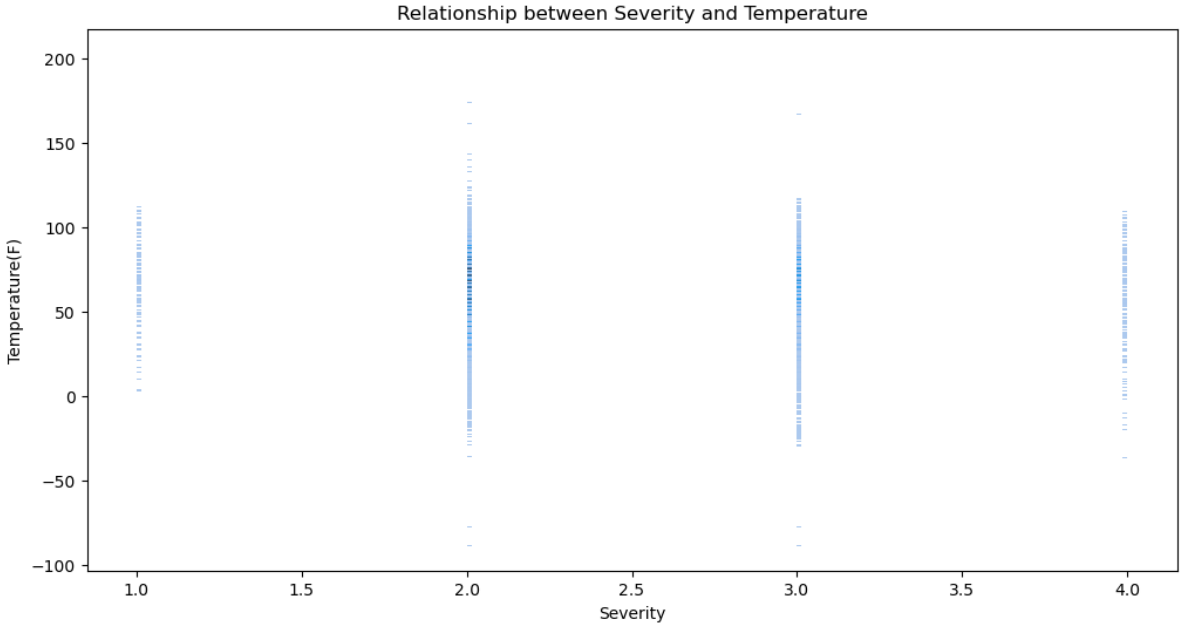


In [15]:
```
severity_counts = df['Severity'].value_counts()
labels = severity_counts.index
sizes = severity_counts.values

plt.figure(figsize=(8, 8))
plt.pie(sizes, labels=labels, autopct='%1.1f%%', startangle=140)
plt.title('Distribution of Accident Severity')
plt.show()
```

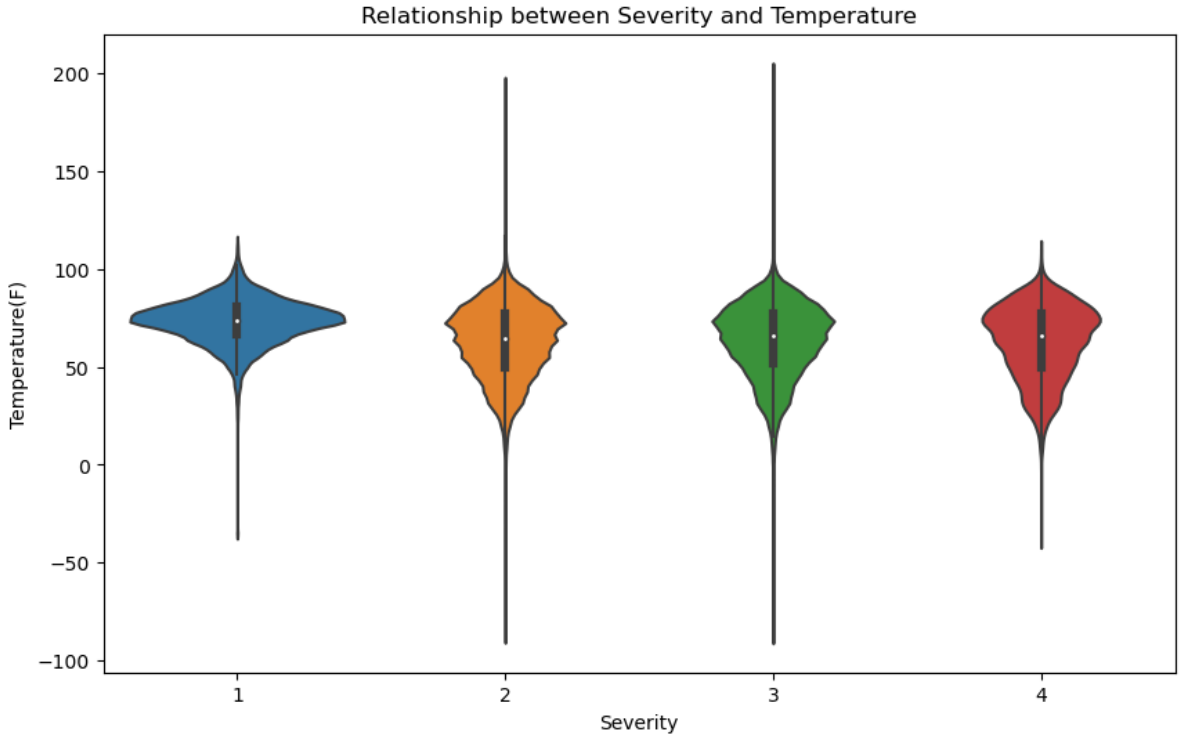


In [17]:
```
#it is seen that 65% of accidents are of level 2 severity
#only 2% of accidents are of severity 1 and 4
```

In [19]:
```
# Relationship between Severity and Weather Conditions
plt.figure(figsize=(12, 6))
sns.histplot(data=df, x='Severity', y='Temperature(F)')
plt.xlabel('Severity')
plt.ylabel('Temperature(F)')
plt.title('Relationship between Severity and Temperature')
plt.show()
```



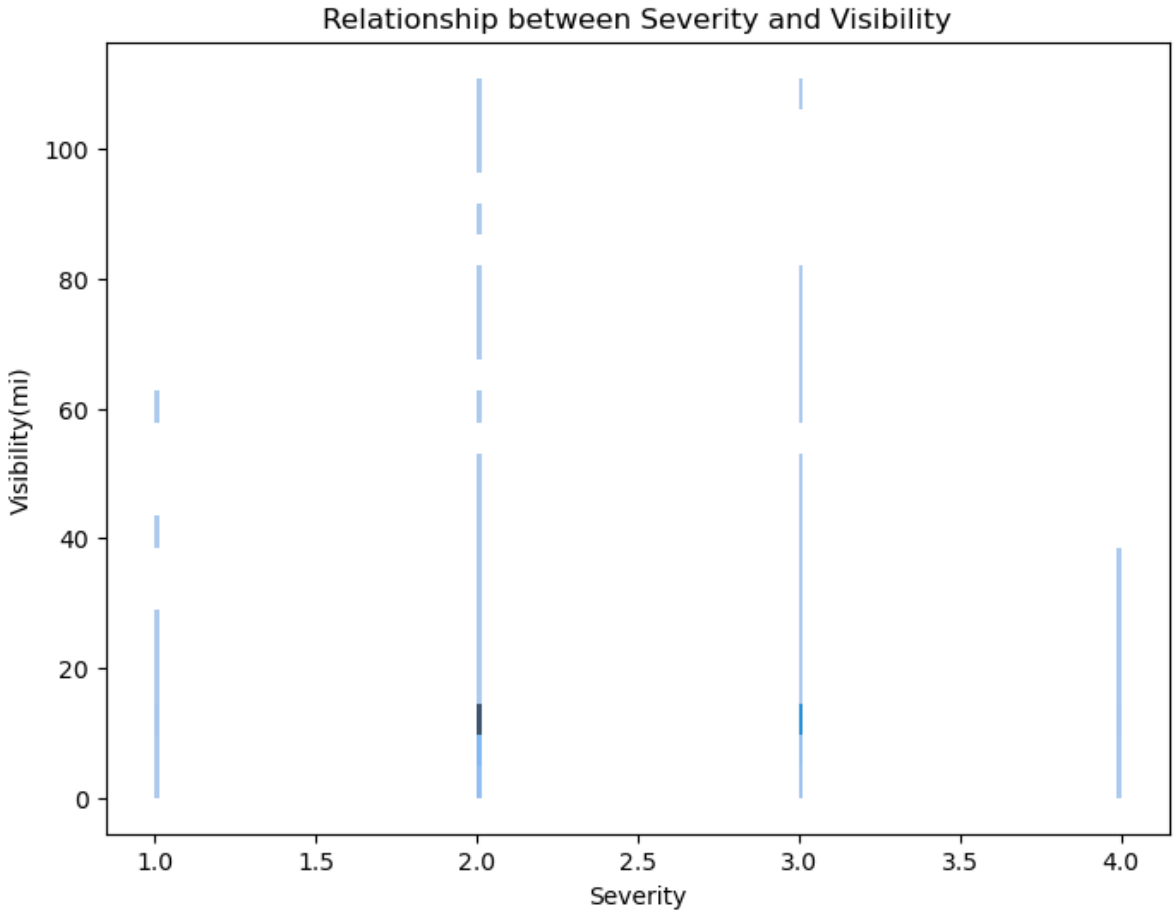
In [20]:
```
plt.figure(figsize=(10, 6))
sns.violinplot(data=df, x='Severity', y='Temperature(F)')
plt.xlabel('Severity')
plt.ylabel('Temperature(F)')
plt.title('Relationship between Severity and Temperature')
plt.show()
```



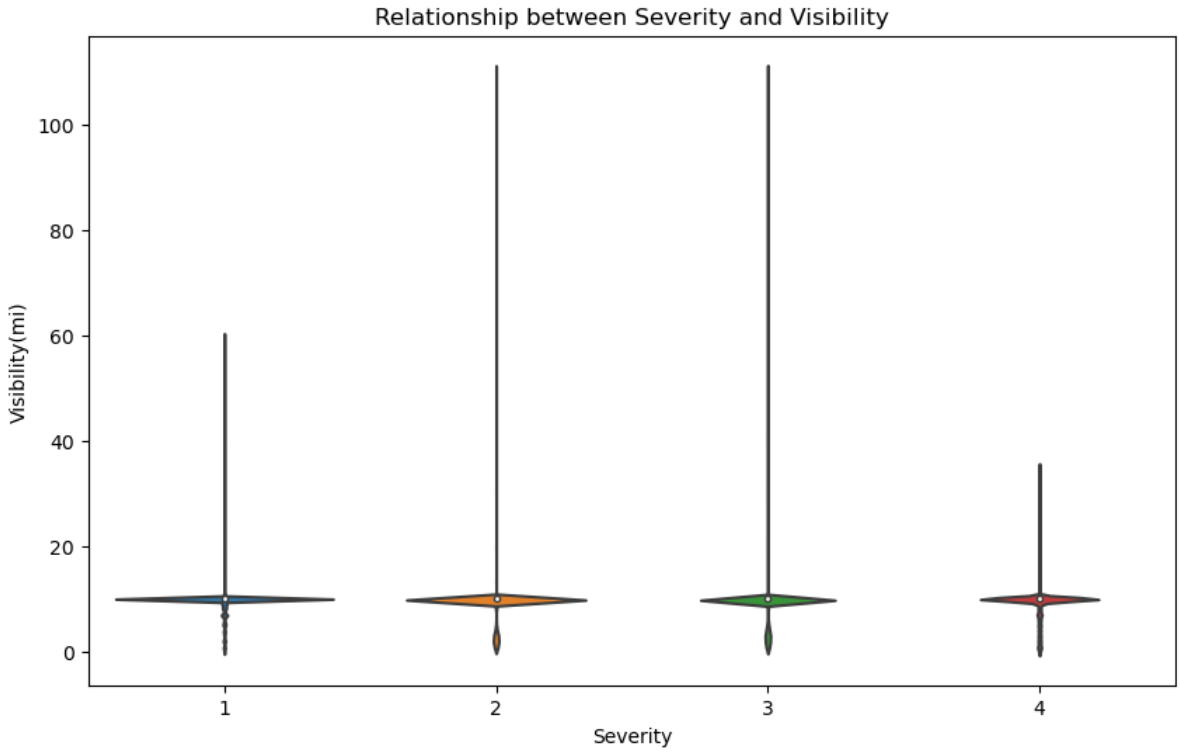
In [26]:
```
df.columns
```

Out[26]:
```
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', 'Wind_Direction_Binned'],
      dtype='object')
```

In [35]:
```
plt.figure(figsize=(8, 6))
sns.histplot(data=df, x='Severity', y='Visibility(mi)')
plt.xlabel('Severity')
plt.ylabel('Visibility(mi)')
plt.title('Relationship between Severity and Visibility')
plt.show()
```

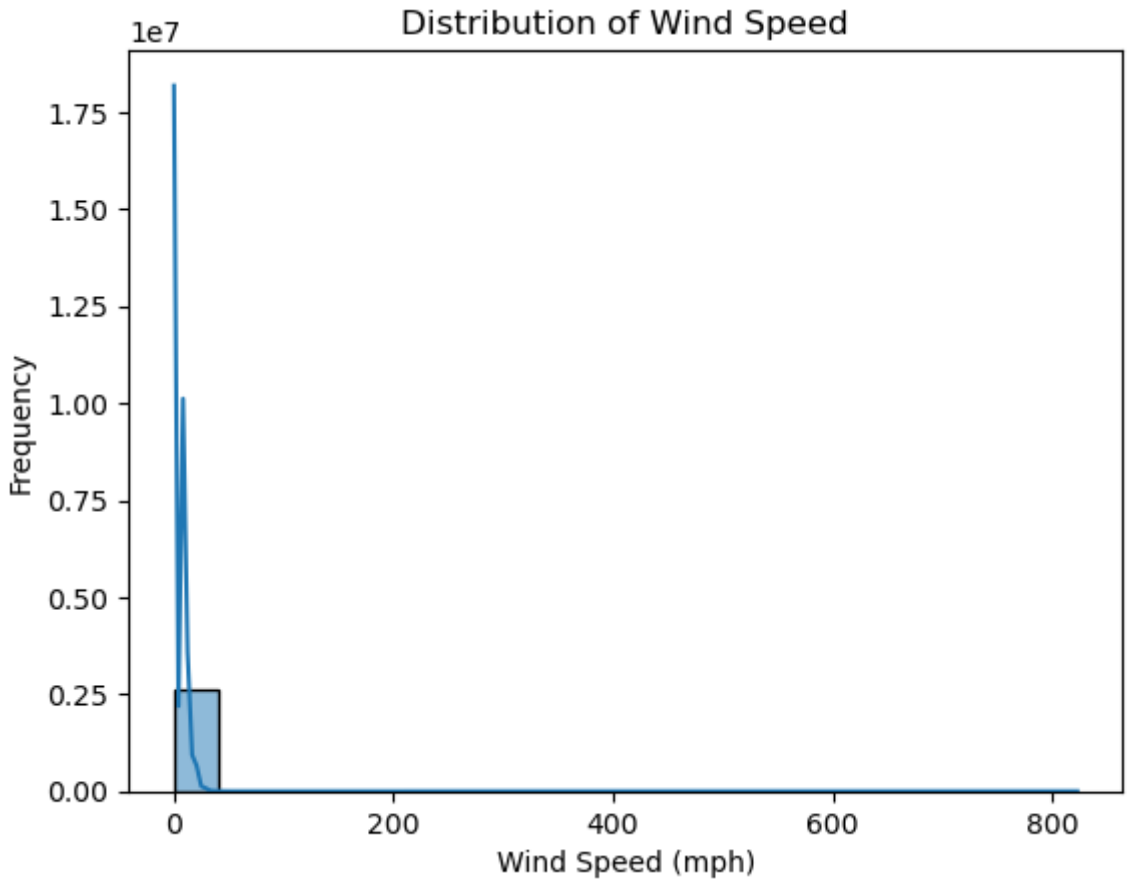


In [37]:
```
plt.figure(figsize=(10, 6))
sns.violinplot(data=df, x='Severity', y='Visibility(mi)')
plt.xlabel('Severity')
plt.ylabel('Visibility(mi)')
plt.title('Relationship between Severity and Visibility')
plt.show()
```



In [ ]:
```
#mostly accidents occur at low visibility between 0 to 2o
```

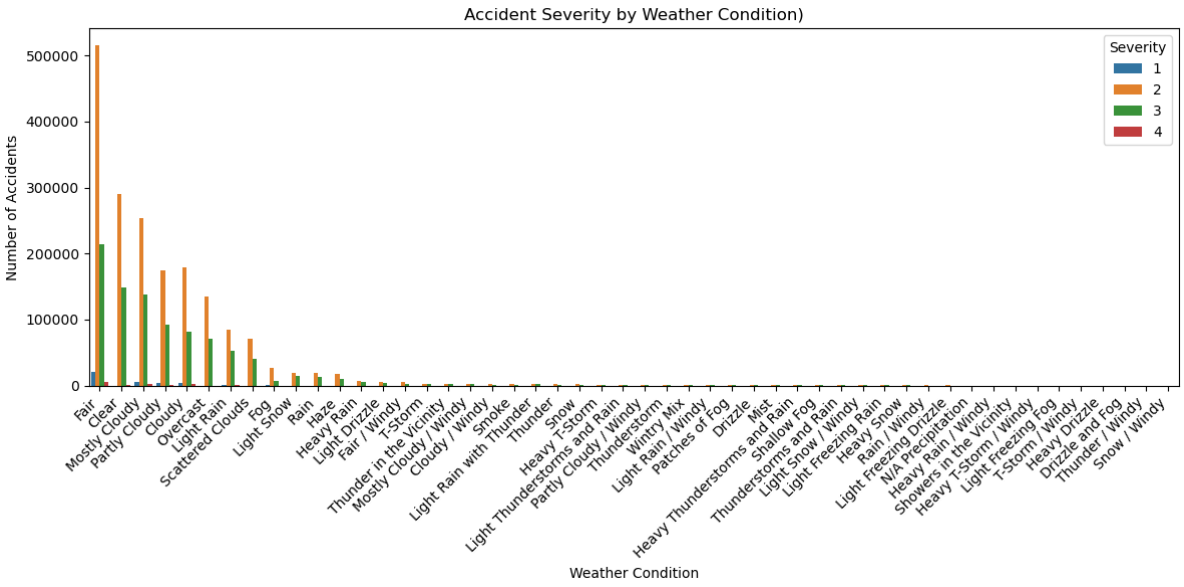
In [38]:
```
sns.histplot(data=df, x='Wind_Speed(mph)', bins=20, kde=True)
plt.xlabel('Wind Speed (mph)')
plt.ylabel('Frequency')
plt.title('Distribution of Wind Speed')
plt.show()
```



In [52]:
```
df.Weather_Condition.value_counts()[:20]
```

Out[52]:
```
Fair                        755628
Clear                       439462
Mostly Cloudy               399204
Partly Cloudy               272622
Cloudy                      266753
Overcast                    206469
Light Rain                  138577
Scattered Clouds            111401
Fog                          35004
Light Snow                   34444
Rain                         32652
Haze                         28780
Heavy Rain                   12812
Light Drizzle                 9888
Fair / Windy                  9017
T-Storm                       5976
Thunder in the Vicinity       5142
Mostly Cloudy / Windy         4911
Cloudy / Windy                4882
Smoke                         4777
Name: Weather_Condition, dtype: int64
```

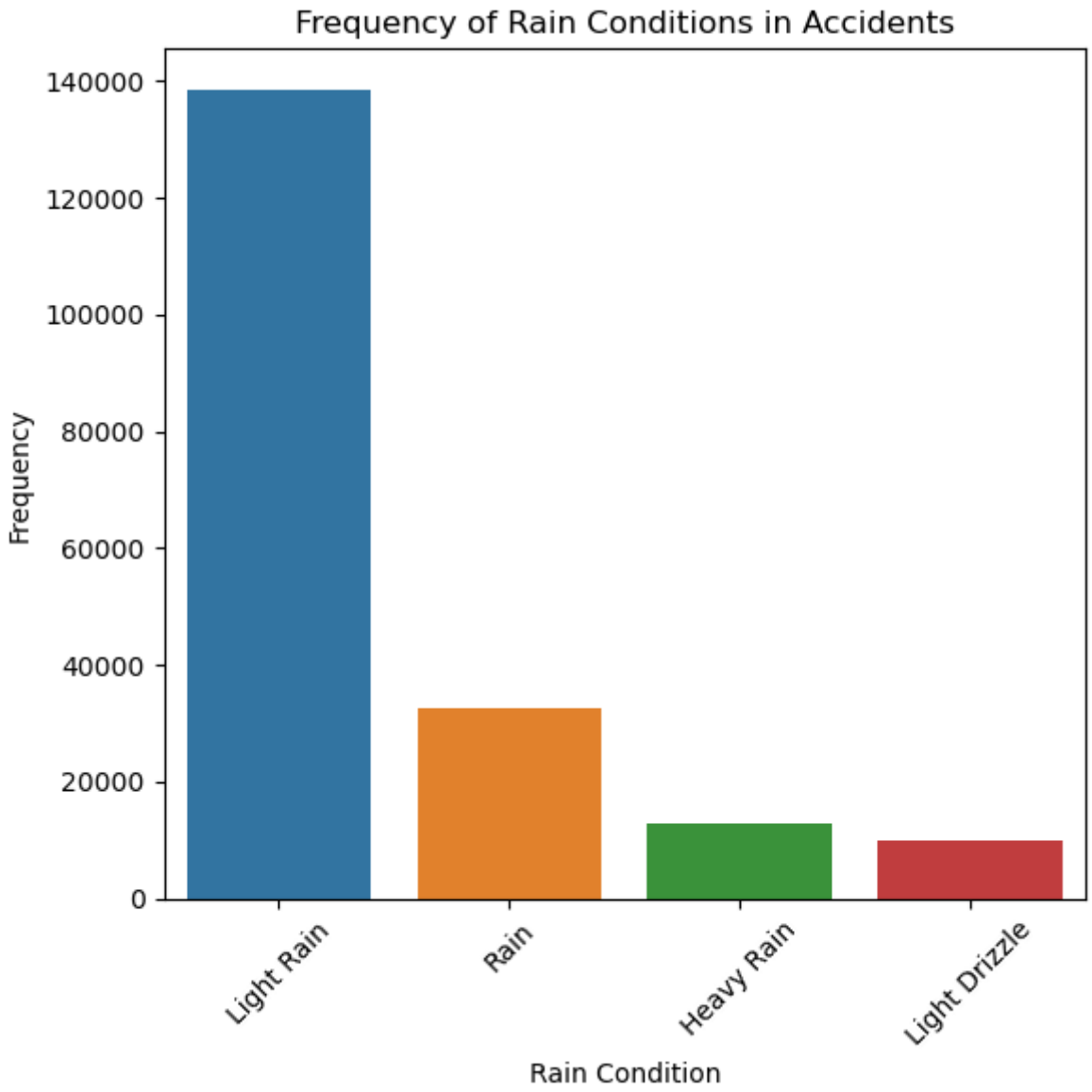
In [55]:
```
weather_conditions_subset = df['Weather_Condition'].value_counts().index[:50]

plt.figure(figsize=(12, 6))
sns.countplot(data=df[df['Weather_Condition'].isin(weather_conditions_subset)], x=
plt.xlabel('Weather Condition')
plt.ylabel('Number of Accidents')
plt.title('Accident Severity by Weather Condition)')
plt.xticks(rotation=45, ha='right')
plt.legend(title='Severity', loc='upper right')
plt.tight_layout()
plt.show()
```

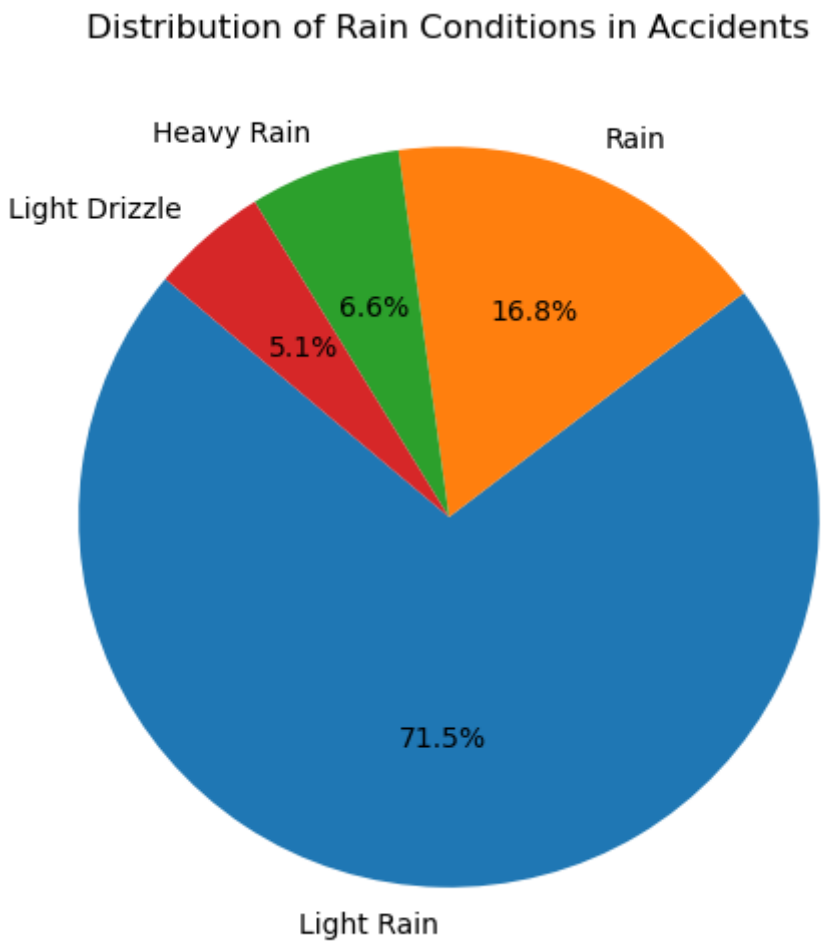


rain analysis

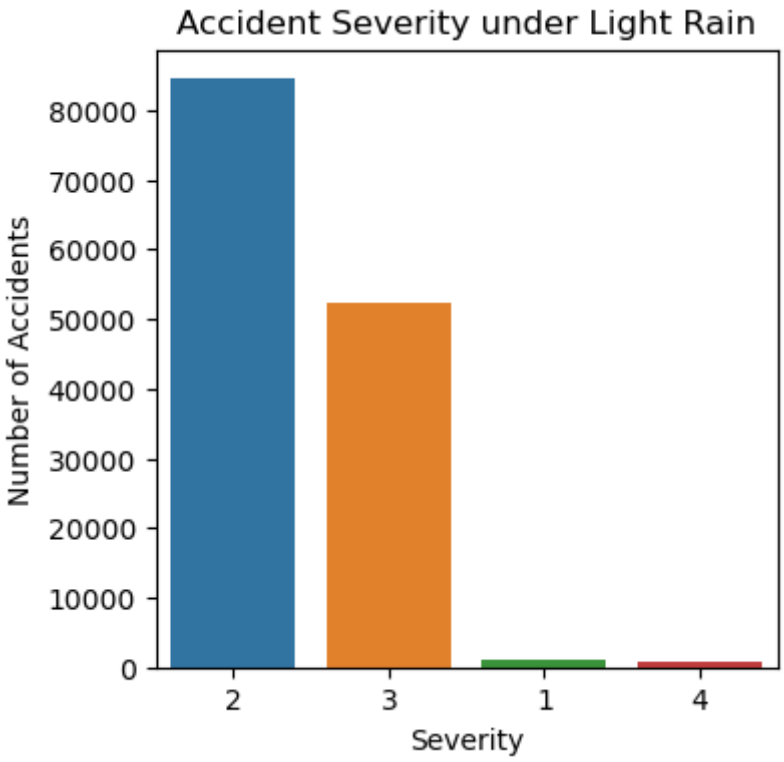
In [71]:
```
rain_conditions = ['Light Rain', 'Rain', 'Heavy Rain', 'Light Drizzle']

# Filter the DataFrame to include only rows with rainy conditions
rainy_df = df[df['Weather_Condition'].isin(rain_conditions)]

# Calculate frequency of each rainy condition
rain_condition_counts = rainy_df['Weather_Condition'].value_counts()
```

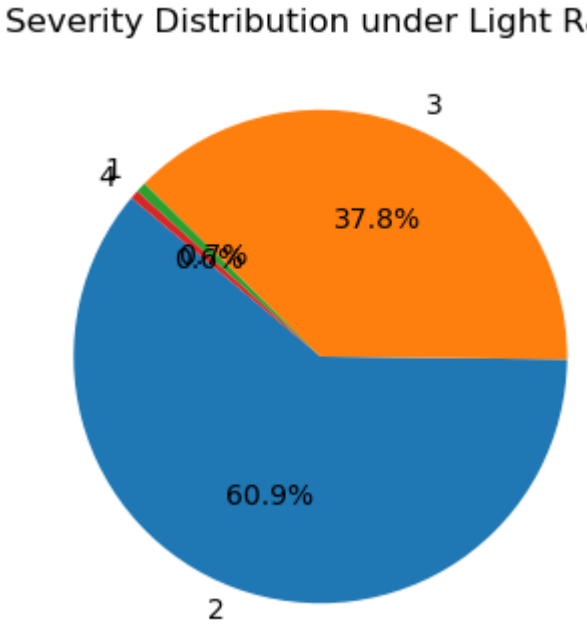
```
# Create a bar plot for rain conditions and their frequency
plt.figure(figsize=(6, 6))
sns.barplot(x=rain_condition_counts.index, y=rain_condition_counts.values)
plt.xlabel('Rain Condition')
plt.ylabel('Frequency')
plt.title('Frequency of Rain Conditions in Accidents')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()

rain_condition_counts = rainy_df['Weather_Condition'].value_counts()

# Create a pie chart for rain condition distribution
plt.figure(figsize=(6, 6))
plt.pie(rain_condition_counts.values, labels=rain_condition_counts.index, autopct=
plt.title('Distribution of Rain Conditions in Accidents')
plt.show()
```





In [63]:
```
# Plot for Light Rain
plt.figure(figsize=(4, 4))
sns.countplot(data=rainy_df[rainy_df['Weather_Condition'] == 'Light Rain'], x='Sev
plt.xlabel('Severity')
plt.ylabel('Number of Accidents')
plt.title('Accident Severity under Light Rain')

plt.show()

# Pie chart for severity distribution under Light Rain
plt.figure(figsize=(4, 4))
severity_counts = rainy_df[rainy_df['Weather_Condition'] == 'Light Rain']['Severity
labels = severity_counts.index
sizes = severity_counts.values
plt.pie(sizes, labels=labels, autopct='%1.1f%%', startangle=140)
plt.title('Severity Distribution under Light Rain')
plt.show()
```

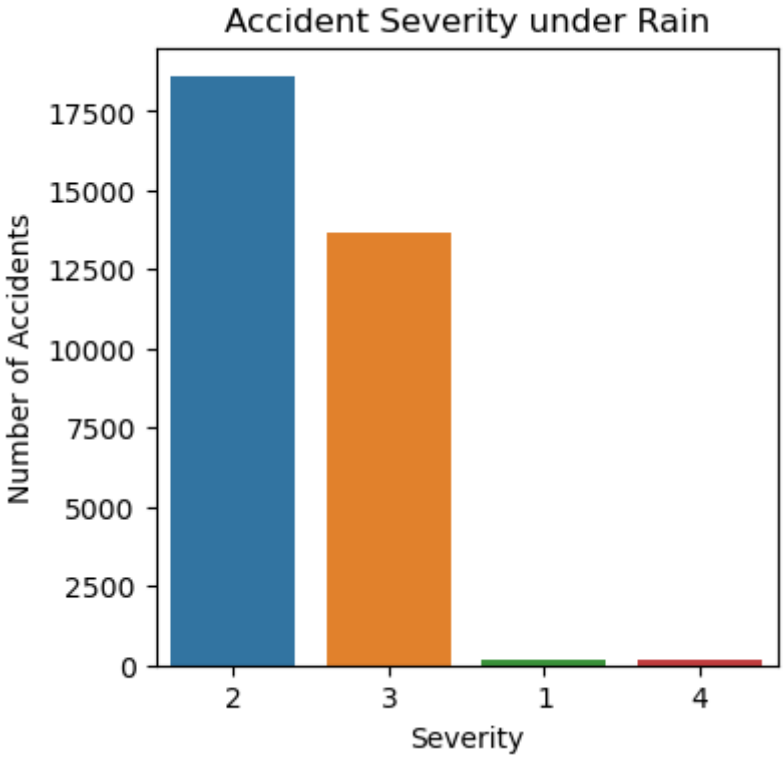


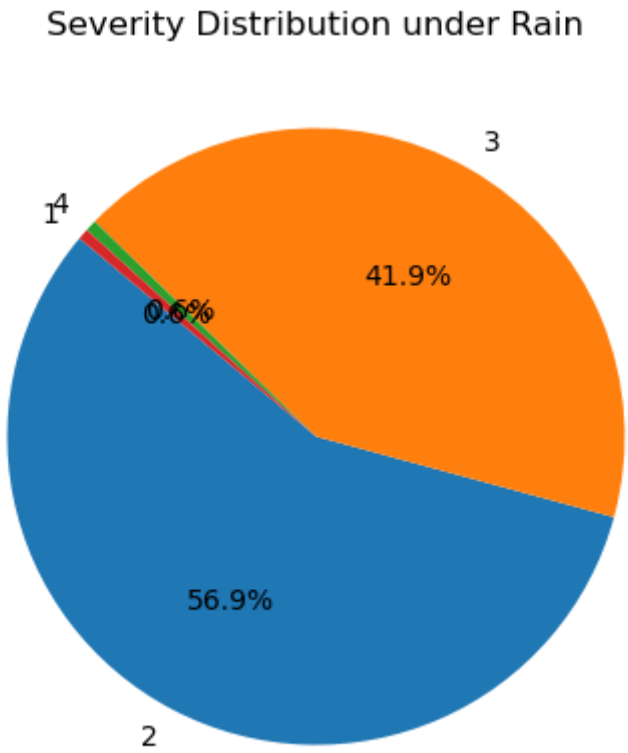


In [ ]:
```
#under accidents because of rain the 71% of accidents occur due to light rain only
#only 6% under heavy rain
```

In [65]:
```
# Plot for Rain
plt.figure(figsize=(4, 4))
sns.countplot(data=rainy_df[rainy_df['Weather_Condition'] == 'Rain'], x='Severity'
plt.xlabel('Severity')
plt.ylabel('Number of Accidents')
plt.title('Accident Severity under Rain')
plt.show()

# Pie chart for severity distribution under Rain
plt.figure(figsize=(5, 5))
severity_counts = rainy_df[rainy_df['Weather_Condition'] == 'Rain']['Severity'].va
labels = severity_counts.index
sizes = severity_counts.values
plt.pie(sizes, labels=labels, autopct='%1.1f%%', startangle=140)
```

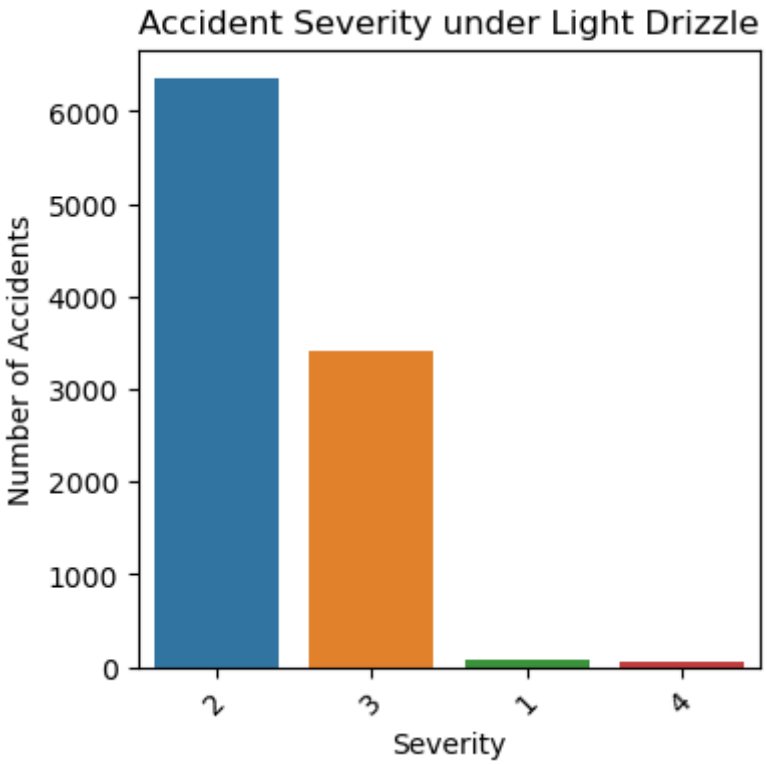
```
plt.title('Severity Distribution under Rain')
plt.show()
```

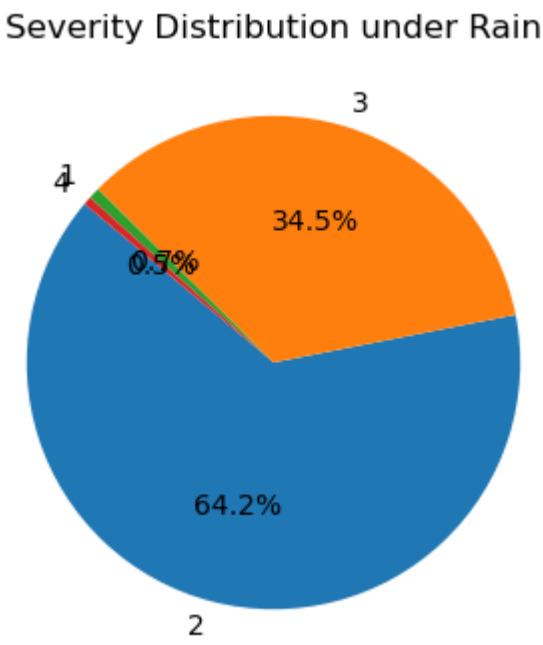




In [67]:
```
plt.figure(figsize=(4, 4))
sns.countplot(data=rainy_df[rainy_df['Weather_Condition'] == 'Light Drizzle'], x='
plt.xlabel('Severity')
plt.ylabel('Number of Accidents')
plt.title('Accident Severity under Light Drizzle')
plt.xticks(rotation=45)
plt.show()

plt.figure(figsize=(4, 4))
```

```
severity_counts = rainy_df[rainy_df['Weather_Condition'] == 'Light Drizzle']['Sever
labels = severity_counts.index
sizes = severity_counts.values
plt.pie(sizes, labels=labels, autopct='%1.1f%%', startangle=140)
plt.title('Severity Distribution under Rain')
plt.show()
```

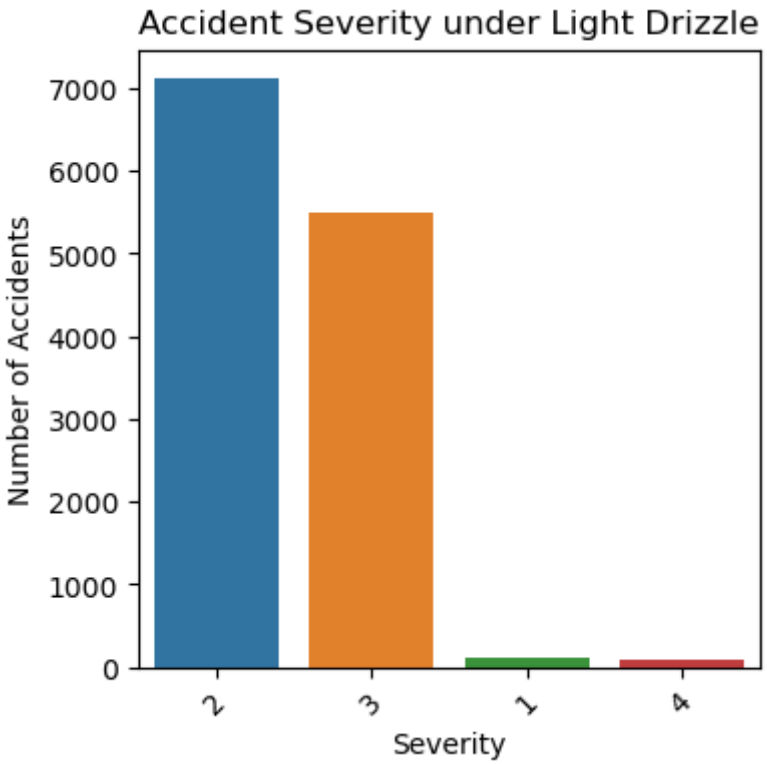


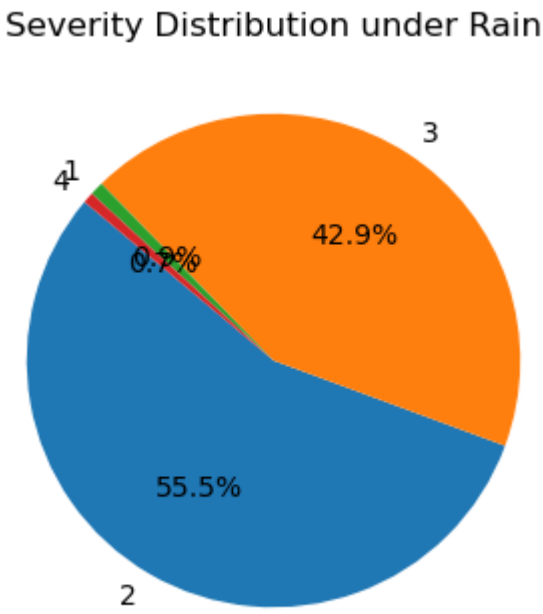


In [72]:

```
plt.figure(figsize=(4, 4))
sns.countplot(data=rainy_df[rainy_df['Weather_Condition'] ==  'Heavy Rain'], x='Se
plt.xlabel('Severity')
plt.ylabel('Number of Accidents')
plt.title('Accident Severity under Light Drizzle')
plt.xticks(rotation=45)
plt.show()

plt.figure(figsize=(4, 4))
severity_counts = rainy_df[rainy_df['Weather_Condition'] == 'Heavy Rain']['Severity
```

```
labels = severity_counts.index
sizes = severity_counts.values
plt.pie(sizes, labels=labels, autopct='%1.1f%%', startangle=140)
plt.title('Severity Distribution under Rain')
plt.show()
```

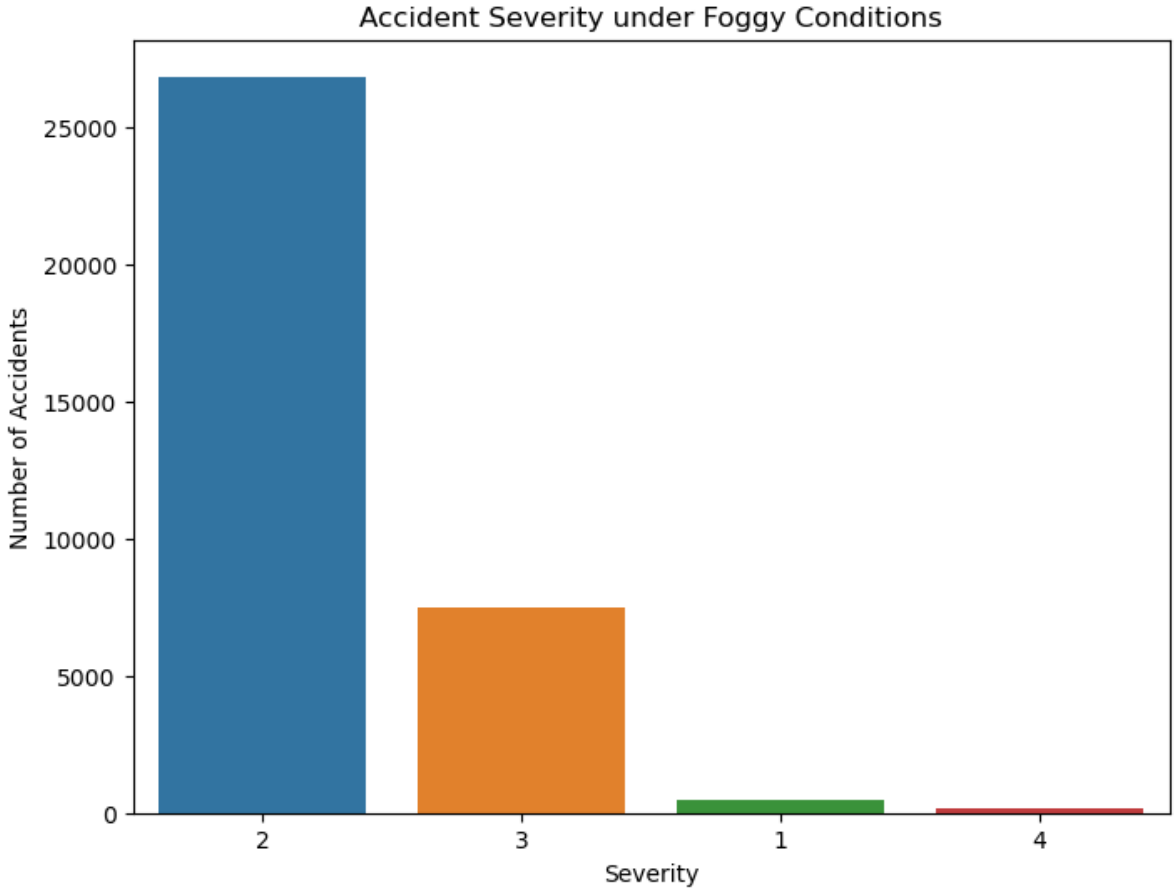


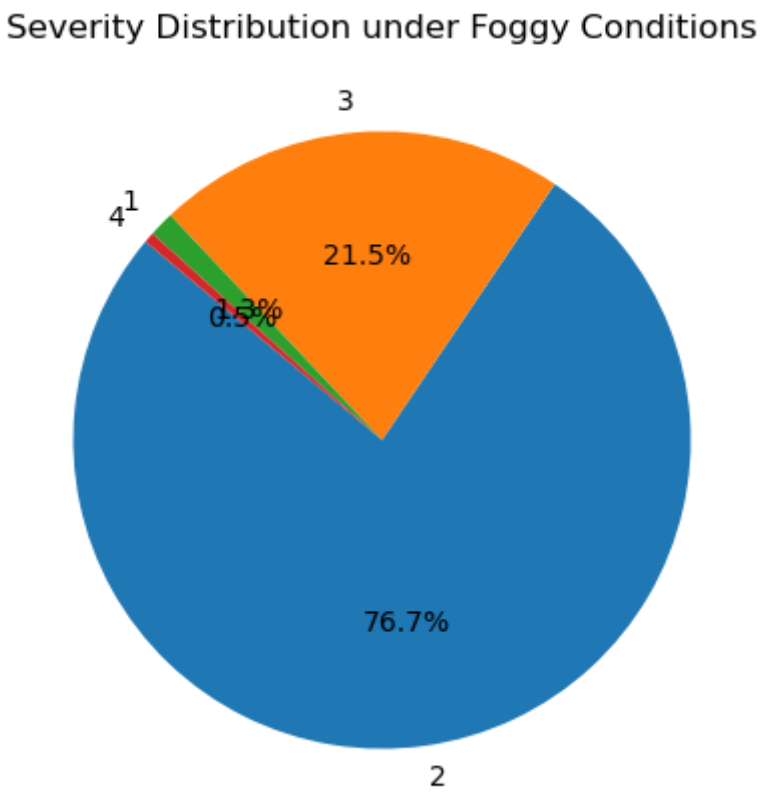


Fog

In [73]:
```
# List of foggy conditions
fog_conditions = ['Fog']

# Filter the DataFrame to include only rows with foggy conditions
foggy_df = df[df['Weather_Condition'].isin(fog_conditions)]

# Visualization - Countplot
plt.figure(figsize=(8, 6))
```

```
sns.countplot(data=foggy_df, x='Severity', order=foggy_df['Severity'].value_counts
plt.xlabel('Severity')
plt.ylabel('Number of Accidents')
plt.title('Accident Severity under Foggy Conditions')
plt.show()

# Pie chart for severity distribution under foggy conditions
plt.figure(figsize=(5, 5))
severity_counts = foggy_df['Severity'].value_counts()
labels = severity_counts.index
sizes = severity_counts.values
plt.pie(sizes, labels=labels, autopct='%1.1f%%', startangle=140)
plt.title('Severity Distribution under Foggy Conditions')
plt.show()

# Statistical Analysis
severity_percentage = (foggy_df['Severity'].value_counts() / len(foggy_df)) * 100
print("Percentage of Accidents by Severity under Foggy Conditions:")
print(severity_percentage)
```






```
Percentage of Accidents by Severity under Foggy Conditions:
2    76.665524
3    21.460405
1     1.325563
4     0.548509
Name: Severity, dtype: float64
```

In [75]:
```
# 76% of accidents are under severity 2 when the weather condition is Fog
```

Summary and Conclusion

No data from New York

The number of accidents per city decreases exponentially

Less than 5% of cities have more than 1000 yearly accidents

Over 1600 cities have reported just one accident(need to investigate)

high percentage of accidents occur between 6am to 10am(probably people in a hurry to get to work)

next high occurence is between 3pm to 7pm

on weekends the peak occurs between 10am to 4pm unlike weekdays

much data is missing for 2016

there is an issue with the source 3 data

Variability in Naming: There is variability in how wind directions are named,such as "Calm" and "CALM" or "Variable" and "VAR." This

inconsistency might be due to different data entry practices or conventions.

it is seen that 65% of accidents are of level 2 severity and only 2% of accidents are of severity 1 and 4

mostly accidents occur at low visibility between 0 to 2o

under accidents because of rain the 71% of accidents occur due to light rain and only 6% under heavy rain

76% of accidents are under severity 2 when the weather condition is Fog