In [7]:
```
import pandas as pd
```

In [6]:
```
data_filepath = "/content/drive/MyDrive/Kaggle/US_Accidents/US_Accidents_Dec20_Updated
```

In [8]:
```
df = pd.read_csv(data_filepath)
df.head(10)
```

art_Time	End_Time	Start_Lat	Start_Lng	End_Lat	End_Lng	Distance(mi)	Description	Number	S
2019-05-08:29:55	2019-05-21 09:29:40	34.808868	-82.269157	34.808868	-82.269157	0.000	Accident on Tanner Rd at Pennbrooke Ln.	439.0	Tanr
2019-10-17:43:09	2019-10-07 19:42:50	35.090080	-80.745560	35.090080	-80.745560	0.000	Accident on Houston Branch Rd at Providence Br...	3299.0	Provi Bran
2020-12-21:53:00	2020-12-13 22:44:00	37.145730	-121.985052	37.165850	-121.988062	1.400	Stationary traffic on CA-17 from Summit Rd (CA...	NaN	Santa
2018-04-16:51:23	2018-04-17 17:50:46	39.110390	-119.773781	39.110390	-119.773781	0.000	Accident on US-395 Southbound at Topsy Ln.	NaN	Hig
2016-08-17:40:49	2016-08-31 18:10:49	26.102942	-80.265091	26.102942	-80.265091	0.000	Accident on I-595 Westbound at Exit 4 / Pine I...	NaN	I-5
2018-10-16:40:36	2018-10-17 17:10:18	35.348240	-80.847221	35.348240	-80.847221	0.000	Three lanes blocked due to accident on I-77 No...	NaN	W Harri
2019-12-09:48:52	2019-12-12 10:18:05	39.523970	-107.777000	39.565780	-107.516950	14.153	Closed between CO-13/Taughenbaugh Blvd/Exit 90...	NaN	
2019-12-23:59:00	2019-12-22 00:32:06	34.034017	-118.026972	34.034017	-118.026972	0.000	At CA-60/Pomona Fwy - Accident.	NaN	CA-
2018-05-16:50:24	2018-05-23 22:50:24	35.863490	-86.831680	35.849480	-86.832530	0.969	At TN-248/Peytonsville Rd/Exit 61 - Accident. ...	425.0	Peyton
2019-01-08:44:18	2019-01-30 09:14:17	34.426330	-118.585100	34.420220	-118.581900	0.460	At Magic Mountain Pky - Accident. Hard shoulde...	NaN	G State

In [9]:
```
# Checking the columns in the data
df.columns
```

Out[9]:
```
Index(['ID', 'Severity', 'Start_Time', 'End_Time', 'Start_Lat', 'Start_Lng',
       'End_Lat', 'End_Lng', 'Distance(mi)', 'Description', 'Number', 'Street',
       'Side', '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 [10]:
```
print("Number of columns: ",len(df.columns))
print("Number of rows: ",len(df))
```

```
Number of columns:  47
Number of rows:  2906610
```

Gathering information about the dataset

- Missing values
- Null values
- Type of data in the file

In [11]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2906610 entries, 0 to 2906609
Data columns (total 47 columns):
 #   Column                 Dtype
---  ------                 -----
 0   ID                     object
 1   Severity               int64
 2   Start_Time             object
 3   End_Time               object
 4   Start_Lat              float64
 5   Start_Lng              float64
 6   End_Lat                float64
 7   End_Lng                float64
 8   Distance(mi)           float64
 9   Description            object
 10  Number                 float64
 11  Street                 object
 12  Side                   object
 13  City                   object
 14  County                 object
 15  State                  object
 16  Zipcode                object
 17  Country                object
 18  Timezone               object
 19  Airport_Code           object
 20  Weather_Timestamp      object
 21  Temperature(F)         float64
 22  Wind_Chill(F)          float64
 23  Humidity(%)            float64
 24  Pressure(in)           float64
 25  Visibility(mi)         float64
 26  Wind_Direction         object
 27  Wind_Speed(mph)        float64
 28  Precipitation(in)      float64
 29  Weather_Condition      object
 30  Amenity                bool
 31  Bump                   bool
 32  Crossing               bool
 33  Give_Way               bool
 34  Junction               bool
 35  No_Exit                bool
 36  Railway                bool
 37  Roundabout             bool
 38  Station                bool
 39  Stop                   bool
 40  Traffic_Calming        bool
 41  Traffic_Signal         bool
 42  Turning_Loop           bool
 43  Sunrise_Sunset         object
 44  Civil_Twilight         object
 45  Nautical_Twilight      object
 46  Astronomical_Twilight  object
dtypes: bool(13), float64(13), int64(1), object(20)
memory usage: 790.0+ MB
```

In [12]:
```
df.describe()
```

Out[12]:

	Severity	Start_Lat	Start_Lng	End_Lat	End_Lng	Distance(mi)	N
count	2.906610e+06	2.906610e+06	2.906610e+06	2.623789e+06	2.623789e+06	2.906610e+06	1.0149
mean	2.288649e+00	3.653027e+01	-9.642676e+01	3.651733e+01	-9.620367e+01	3.980541e-01	6.7897
std	5.541618e-01	5.013964e+00	1.775412e+01	5.016609e+00	1.765971e+01	1.592556e+00	1.6972
min	1.000000e+00	2.455527e+01	-1.246238e+02	2.455527e+01	-1.246238e+02	0.000000e+00	0.0000
25%	2.000000e+00	3.366453e+01	-1.178232e+02	3.364659e+01	-1.177020e+02	0.000000e+00	9.6500
50%	2.000000e+00	3.609977e+01	-9.116690e+01	3.605898e+01	-9.105163e+01	0.000000e+00	3.0930
75%	3.000000e+00	4.037505e+01	-8.085814e+01	4.033133e+01	-8.084679e+01	2.790000e-01	7.9760
max	4.000000e+00	4.900220e+01	-6.711317e+01	4.907500e+01	-6.710924e+01	3.336300e+02	9.9999

How many columns are numerical data?

In [13]:
```
len(df.select_dtypes(['int64', 'float64']).columns)
```

Out[13]: 14

Missing or incorrect values?

In [14]:
```
df.isnull().sum()
```

Out[14]:

ID	0
Severity	0
Start_Time	0
End_Time	0
Start_Lat	0
Start_Lng	0
End_Lat	282821
End_Lng	282821
Distance(mi)	0
Description	0
Number	1891672
Street	0
Side	0
City	108
County	0
State	0
Zipcode	1114
Country	0
Timezone	3430
Airport_Code	6608
Weather_Timestamp	46917
Temperature(F)	67224
Wind_Chill(F)	1183859
Humidity(%)	71270
Pressure(in)	56908
Visibility(mi)	72078
Wind_Direction	63474
Wind_Speed(mph)	307163
Precipitation(in)	1301326
Weather_Condition	71851
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	110
Civil_Twilight	110
Nautical_Twilight	110
Astronomical_Twilight	110

dtype: int64

In [15]:
```
df.isna().sum()
```

Out[15]:

ID	0
Severity	0
Start_Time	0
End_Time	0
Start_Lat	0
Start_Lng	0
End_Lat	282821
End_Lng	282821
Distance(mi)	0
Description	0
Number	1891672
Street	0
Side	0
City	108
County	0
State	0
Zipcode	1114
Country	0
Timezone	3430
Airport_Code	6608
Weather_Timestamp	46917
Temperature(F)	67224
Wind_Chill(F)	1183859
Humidity(%)	71270
Pressure(in)	56908
Visibility(mi)	72078
Wind_Direction	63474
Wind_Speed(mph)	307163
Precipitation(in)	1301326
Weather_Condition	71851
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	110
Civil_Twilight	110
Nautical_Twilight	110
Astronomical_Twilight	110

dtype: int64

Finding the percantage of missing data per columns?

In [16]:
```
df.isna().sum().sort_values(ascending=False) * 100. / len(df)
```

Out[16]:
```
Number                 65.081728
Precipitation(in)      44.771263
Wind_Chill(F)          40.729888
Wind_Speed(mph)        10.567740
End_Lat                 9.730270
End_Lng                 9.730270
Visibility(mi)          2.479796
Weather_Condition       2.471986
Humidity(%)             2.451997
Temperature(F)          2.312797
Wind_Direction          2.183781
Pressure(in)            1.957882
Weather_Timestamp       1.614148
Airport_Code            0.227344
Timezone                0.118007
Zipcode                 0.038326
Nautical_Twilight       0.003784
Astronomical_Twilight   0.003784
Civil_Twilight          0.003784
Sunrise_Sunset          0.003784
City                    0.003716
Amenity                 0.000000
Severity                0.000000
Start_Time              0.000000
End_Time                0.000000
Start_Lat               0.000000
Start_Lng               0.000000
Distance(mi)            0.000000
Description             0.000000
Turning_Loop            0.000000
Street                  0.000000
Side                    0.000000
County                  0.000000
Bump                    0.000000
State                   0.000000
Traffic_Signal          0.000000
Country                 0.000000
Traffic_Calming         0.000000
Stop                    0.000000
Station                 0.000000
Roundabout              0.000000
Railway                 0.000000
No_Exit                 0.000000
Junction                0.000000
Give_Way                0.000000
Crossing                0.000000
ID                      0.000000
dtype: float64
```

Plotting the missing percentages

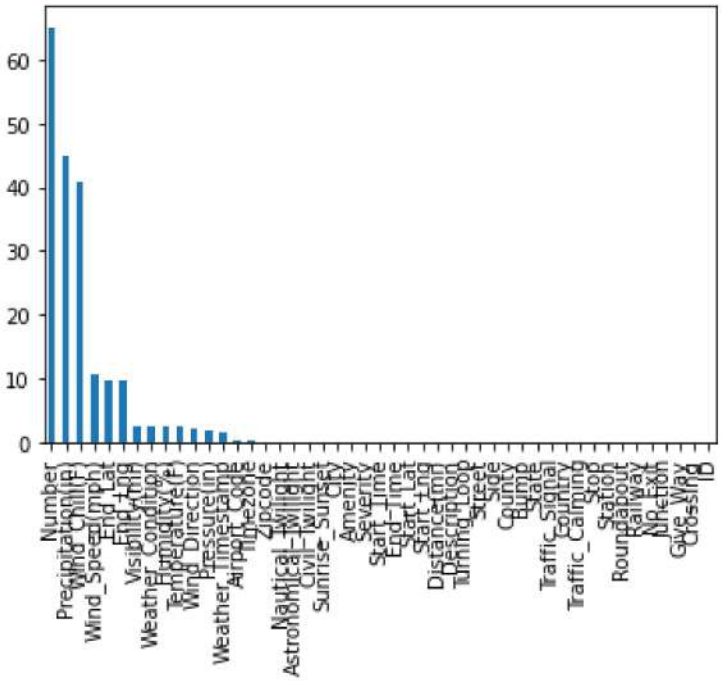
In [17]:
```
# Plotting a Pandas.Series data
missing_data = df.isna().sum().sort_values(ascending=False) * 100. / len(df)
```

In [18]:
```
type(missing_data) # we can directly plot the Pandas.Series using plot()
```

Out[18]:
```
pandas.core.series.Series
```

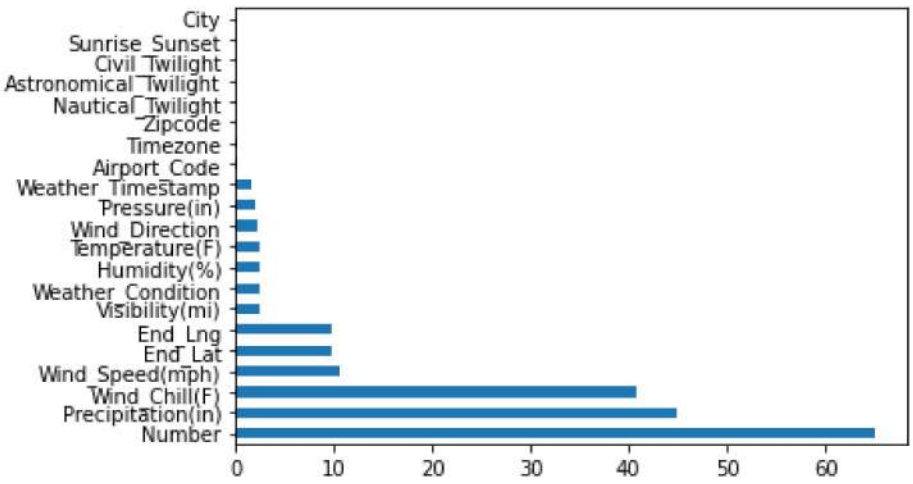
In [19]:
```
missing_data.plot(kind='bar')
```

Out[19]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd7016fd910>
```



In [20]:
```
missing_data[missing_data!=0].plot(kind='barh')
```

Out[20]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd701690750>
```



In [102…
```
# Printing all the columns
df.columns
```

Out[102]:
```
Index(['ID', 'Severity', 'Start_Time', 'End_Time', 'Start_Lat', 'Start_Lng',
       'End_Lat', 'End_Lng', 'Distance(mi)', 'Description', 'Number', 'Street',
       'Side', '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 [22]:
```
df.City.unique()
```

Out[22]:
```
array(['Greenville', 'Charlotte', 'Los Gatos', ..., 'Allons', 'Adolphus',
       'Gowanda'], dtype=object)
```

In [23]:
```
cities = df.City.unique()
len(cities)
```

Out[23]:
```
11790
```

Getting the number of accidents in each city over all years (2016-2020)

In [24]:
```
cities_by_accident = df.City.value_counts()
cities_by_accident[:20]
```

Out[24]:
```
Los Angeles      68411
Houston          68265
Charlotte        56176
Miami            49965
Dallas           48525
Austin           38808
Raleigh          31355
Atlanta          29244
Sacramento       28984
Orlando          28092
Nashville        25277
Baton Rouge      25080
Minneapolis      22469
San Diego        22329
Phoenix          21370
Oklahoma City    21292
Portland         19432
Richmond         18343
Seattle          17384
Saint Paul       17266
Name: City, dtype: int64
```

In [25]:
```
'New York' in cities
```

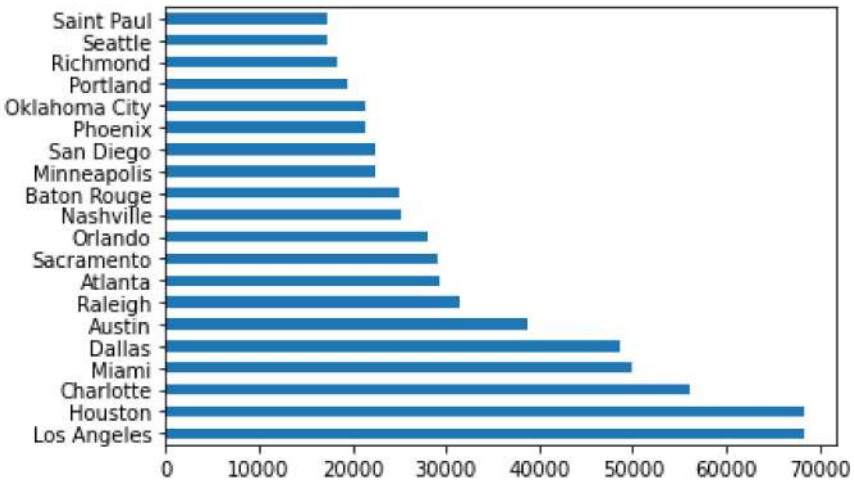
Out[25]:
```
True
```

In [26]:
```
cities_by_accident["New York"]
```

Out[26]:
```
7328
```

In [27]:
```
cities_by_accident[:20].plot(kind='barh')
```

Out[27]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd701090c90>
```



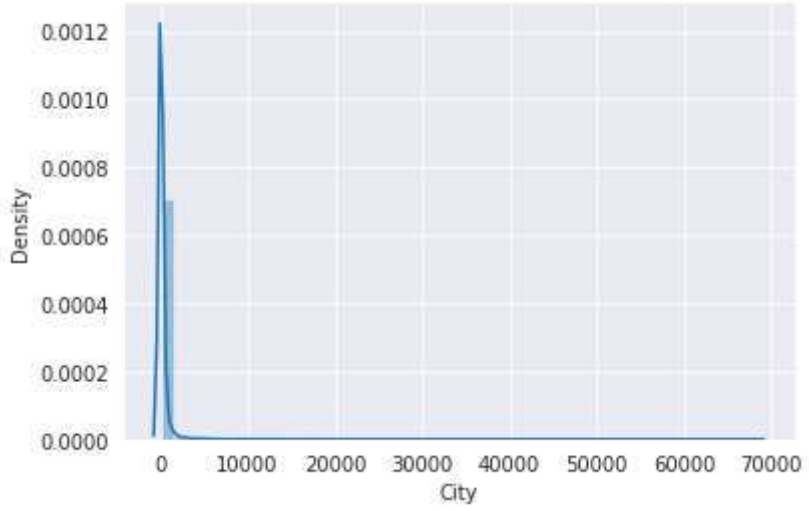
In [28]:
```
import seaborn as sns
sns.set_style("darkgrid")
```

Plotting all the citis by number of accidents accidents

In [29]:
```
sns.distplot(cities_by_accident)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2557: FutureWarning:
`distplot` is a deprecated function and will be removed in a future version. Please a
dapt your code to use either `displot` (a figure-level function with similar flexibil
ity) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[29]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd6f25f0fd0>
```



In [30]:
```
high_accident_cities = cities_by_accident[cities_by_accident >=1000] # having over 100
low_accident_cities = cities_by_accident[cities_by_accident < 1000]  # having less tha
```

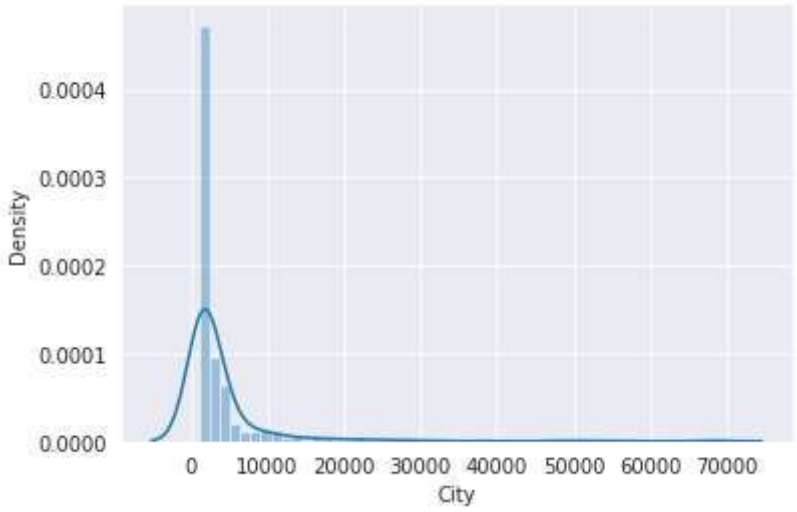
In [31]:
```
# Percentage of high accident cities
len(high_accident_cities) / len(cities_by_accident)
```

Out[31]:
```
0.04351514123335313
```

In [32]:
```
# Distribution of high accident cities
sns.distplot(high_accident_cities)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2557: FutureWarning:
`distplot` is a deprecated function and will be removed in a future version. Please a
dapt your code to use either `displot` (a figure-level function with similar flexibil
ity) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

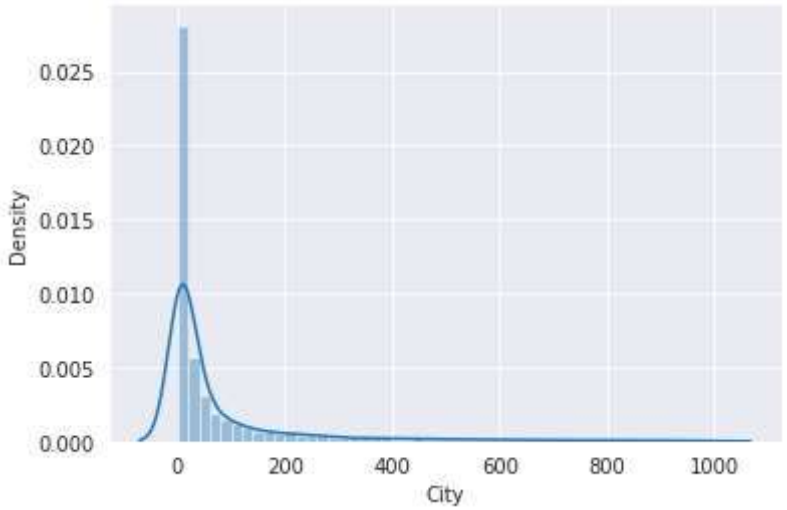
Out[32]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd6e7c7e2d0>
```



In [33]:
```
# Distribution of low accident cities
sns.distplot(low_accident_cities)
```

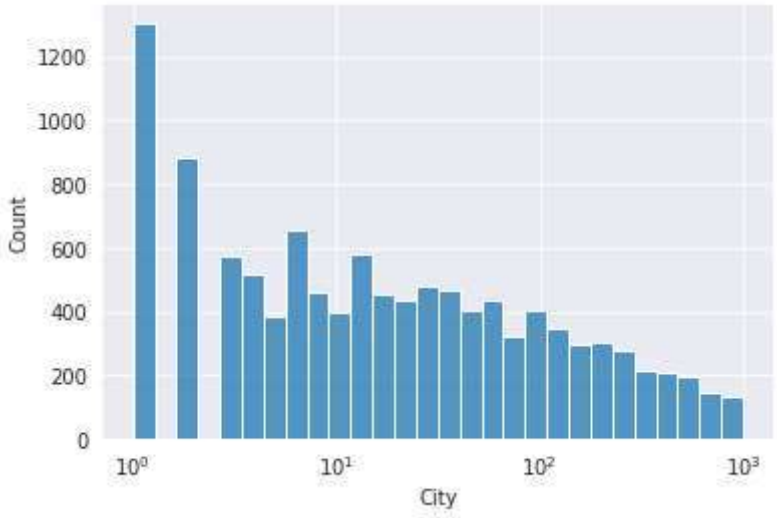
```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2557: FutureWarning:
`distplot` is a deprecated function and will be removed in a future version. Please a
dapt your code to use either `displot` (a figure-level function with similar flexibil
ity) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[33]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd6e7c66750>
```



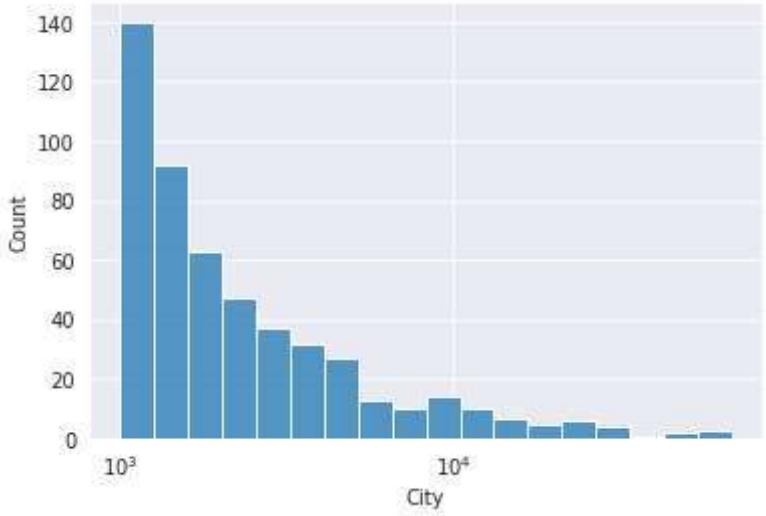
In [34]:
```
# Distribution of low accident cities
sns.histplot(low_accident_cities, log_scale=True)
```

Out[34]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd6e7a3f110>
```



In [35]:
```
# Distribution of high accident cities
sns.histplot(high_accident_cities, log_scale=True)
```

Out[35]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd6e7a57650>
```



There are alos cities which have reported just 1 accident in 4 years.

This could be an indication of some missing data/ irregularities or the impact of population, per-capitaincome, government spending, average age of city, etc. as hypothesised earlier

In [36]:
```
cities_by_accident[cities_by_accident == 1]
```

Out[36]:
```
Clinchco                  1
Conemaugh                 1
Beardstown                1
Tompkinsville             1
Fairchild Air Force Base  1
                         ..
Manitowish Waters         1
Polo                      1
East Dorset               1
Marine City               1
Wardsboro                 1
Name: City, Length: 1306, dtype: int64
```

In [37]:
```
#checking out an entry
df.Start_Time[0]
```

Out[37]:
```
'2019-05-21 08:29:55'
```

In [38]:
```
# converting date time to correct format
df.Start_Time = pd.to_datetime(df.Start_Time)
```

In [39]:
```
df.Start_Time[0]
```

Out[39]:
```
Timestamp('2019-05-21 08:29:55')
```

In [40]:
```
# Segregating the different aspects of date-time
df.Start_Time[0].day, df.Start_Time[0].month, df.Start_Time[0].year, df.Start_Time[0].
```

Out[40]:
```
(21, 5, 2019, 8, 29, 55)
```

Get the hour of the day for all the data

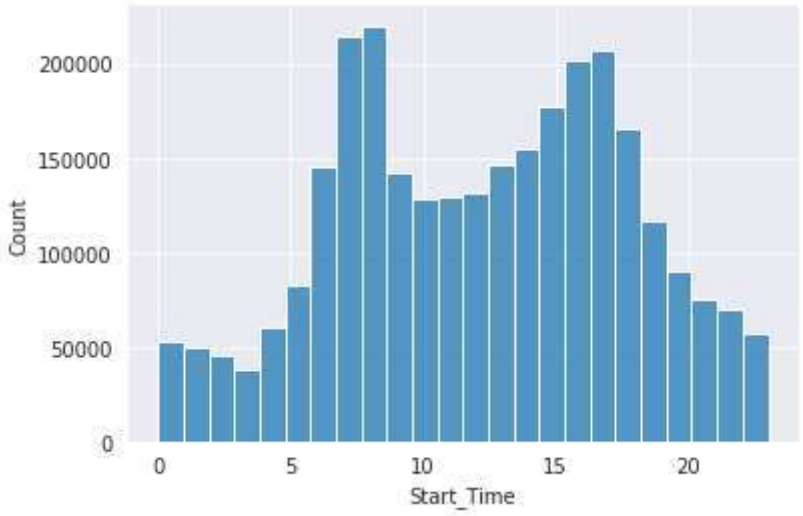
In [41]:
```
df.Start_Time.dt.hour
```

Out[41]:
```
0           8
1          17
2          21
3          16
4          17
           ..
2906605     8
2906606     2
2906607    12
2906608    22
2906609    13
Name: Start_Time, Length: 2906610, dtype: int64
```

Plotting the density distribution and count distribution of accidents at each hour of the day

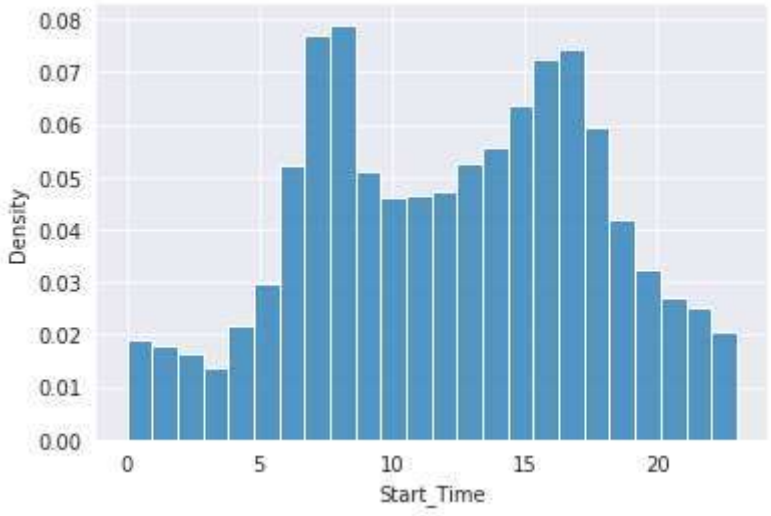
In [42]:
```
sns.histplot(df.Start_Time.dt.hour, bins=24)
```

Out[42]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd6e7b41090>
```



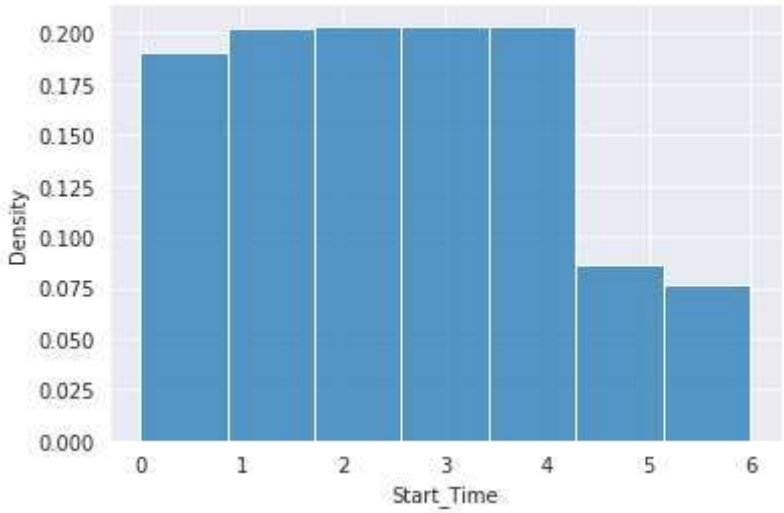
In [43]:
```
sns.histplot(df.Start_Time.dt.hour, bins=24, stat='density')
```

Out[43]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd6e792c210>
```



In [44]:
```
sns.histplot(df.Start_Time.dt.dayofweek, bins=7, stat='density')
```

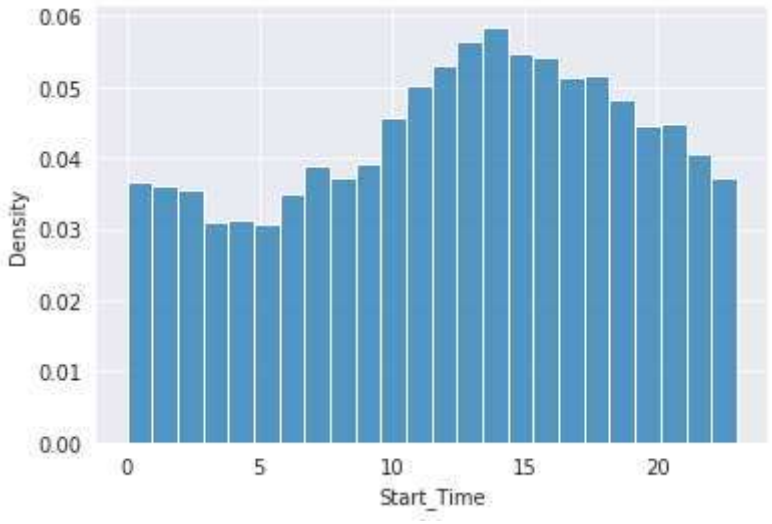
Out[44]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd6e76aea50>
```



In [45]:
```
sundays_start_time = df.Start_Time[df.Start_Time.dt.dayofweek == 6]
```

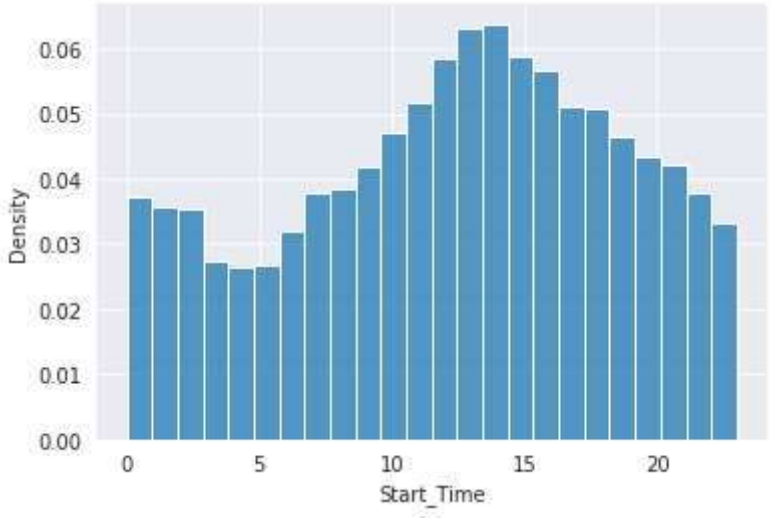
In [46]:
```
sns.histplot(sundays_start_time.dt.hour, bins=24, stat='density')
```

Out[46]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd6e75ce750>
```



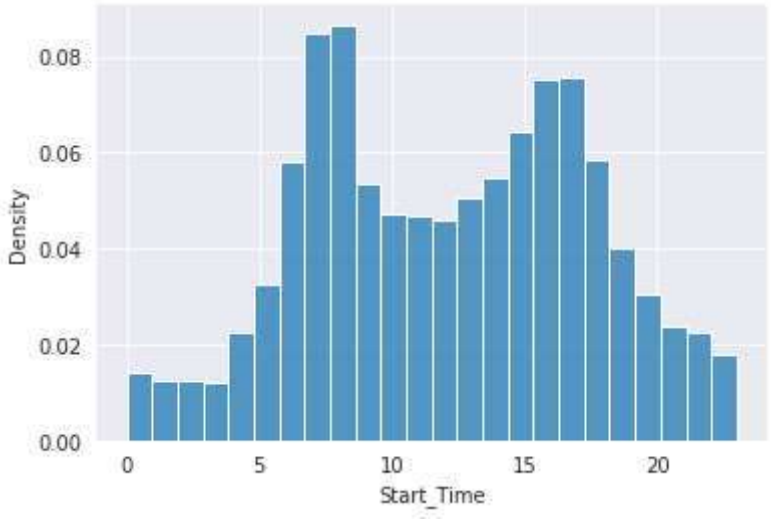
In [47]:
```
saturdays_start_time = df.Start_Time[df.Start_Time.dt.dayofweek == 5]
sns.histplot(saturdays_start_time.dt.hour, bins=24, stat='density')
```

Out[47]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd6e74f8190>
```



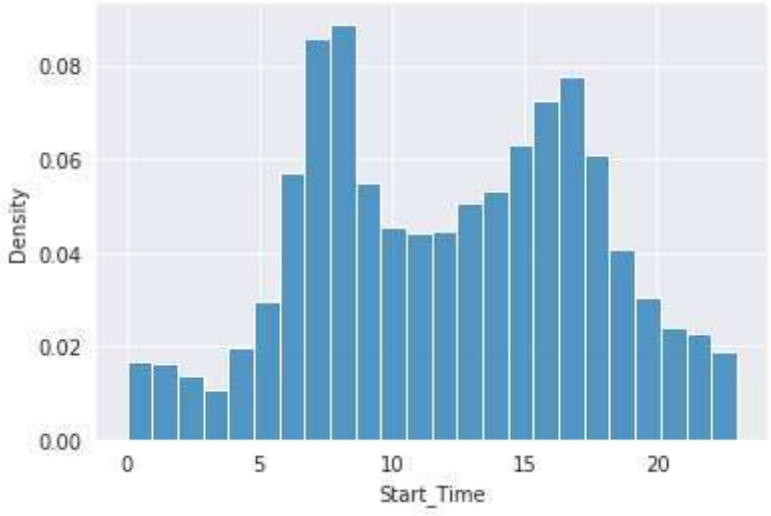
In [48]:
```
mondays_start_time = df.Start_Time[df.Start_Time.dt.dayofweek == 0]
sns.histplot(mondays_start_time.dt.hour, bins=24, stat='density')
```

Out[48]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd6e743a950>
```



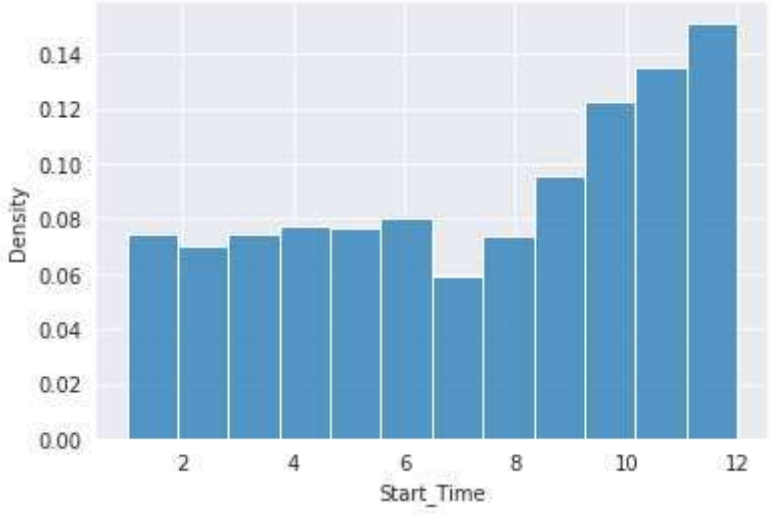
In [103…
```
wednesdays_start_time = df.Start_Time[df.Start_Time.dt.dayofweek == 2]
sns.histplot(wednesdays_start_time.dt.hour, bins=24, stat='density')
```

Out[103]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd6e74ab610>
```



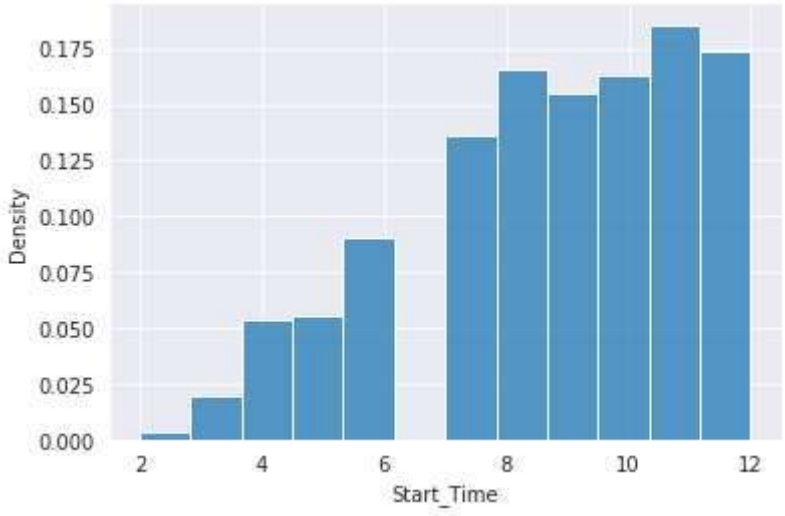
In [49]:
```
sns.histplot(df.Start_Time.dt.month, bins=12, stat='density')
```

Out[49]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd6e7457f50>
```



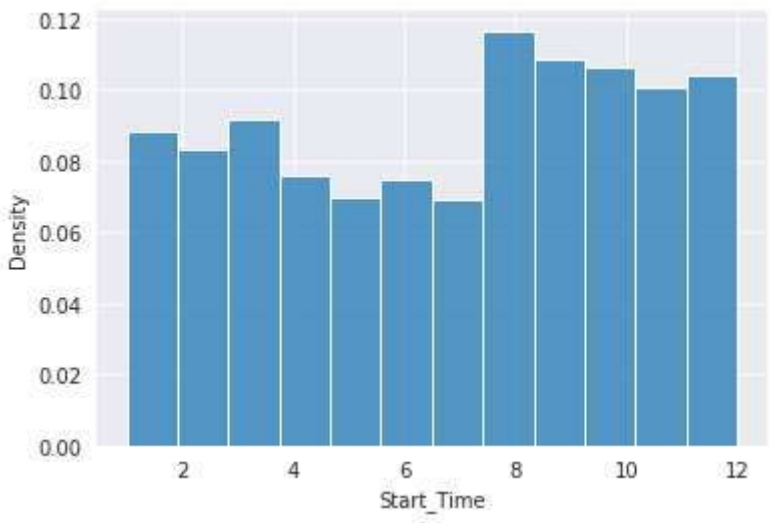
In [50]:
```
df_particular_year = df[df.Start_Time.dt.year == 2016]
sns.histplot(df_particular_year.Start_Time.dt.month, bins=12, stat='density')
```

Out[50]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd6e7348390>
```



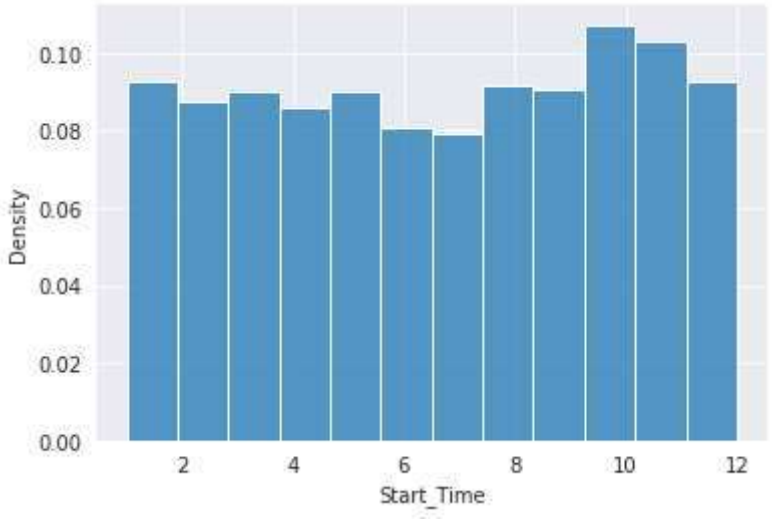
In [51]:
```
df_particular_year = df[df.Start_Time.dt.year == 2017]
sns.histplot(df_particular_year.Start_Time.dt.month, bins=12, stat='density')
```

Out[51]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd6e72e3510>
```



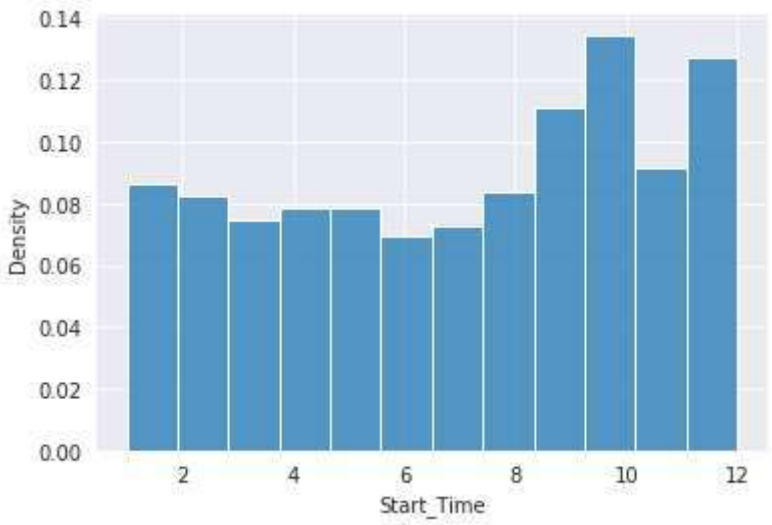
In [52]:
```
df_particular_year = df[df.Start_Time.dt.year == 2018]
sns.histplot(df_particular_year.Start_Time.dt.month, bins=12, stat='density')
```

Out[52]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd6e7273650>
```



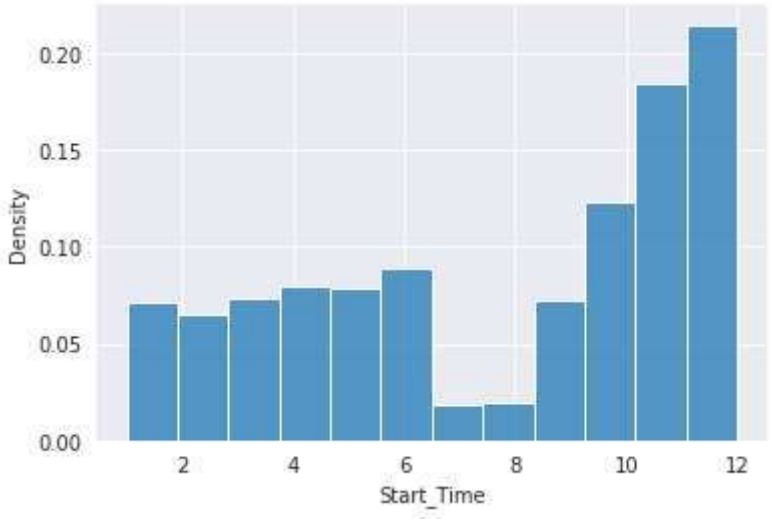
In [53]:
```
df_particular_year = df[df.Start_Time.dt.year == 2019]
sns.histplot(df_particular_year.Start_Time.dt.month, bins=12, stat='density')
```

Out[53]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd6e721ed10>
```



In [54]:
```
df_particular_year = df[df.Start_Time.dt.year == 2020]
sns.histplot(df_particular_year.Start_Time.dt.month, bins=12, stat='density')
```

Out[54]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd6e71121d0>
```



In [55]:
```
df.Start_Lat
```

Out[55]:
```
0          34.808868
1          35.090080
2          37.145730
3          39.110390
4          26.102942
             ...
2906605    29.813824
2906606    34.068890
2906607    25.702200
2906608    40.660140
2906609    38.831749
Name: Start_Lat, Length: 2906610, dtype: float64
```

In [56]:
```
df.Start_Lng
```

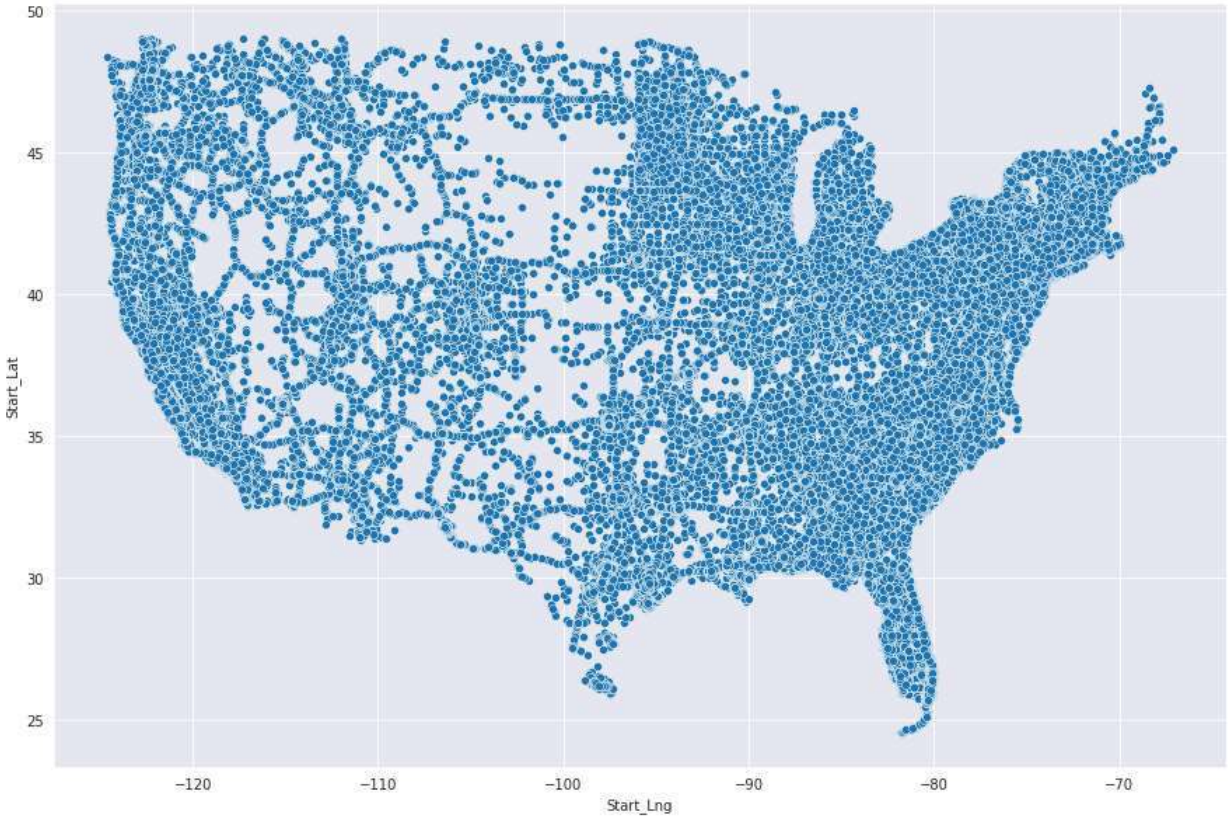
Out[56]:
```
0          -82.269157
1          -80.745560
2         -121.985052
3         -119.773781
4          -80.265091
              ...
2906605    -95.399437
2906606   -117.342010
2906607    -80.335556
2906608   -111.952460
2906609   -104.748161
Name: Start_Lng, Length: 2906610, dtype: float64
```

In [57]:
```
import matplotlib.pyplot as plt
```

Plotting the latitudes and longitudes

In [58]:
```
plt.figure(figsize=(15,10))
sns.scatterplot(y=df.Start_Lat, x=df.Start_Lng)
```

Out[58]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd6e710cb90>
```



In [104…

```
df.info()
```

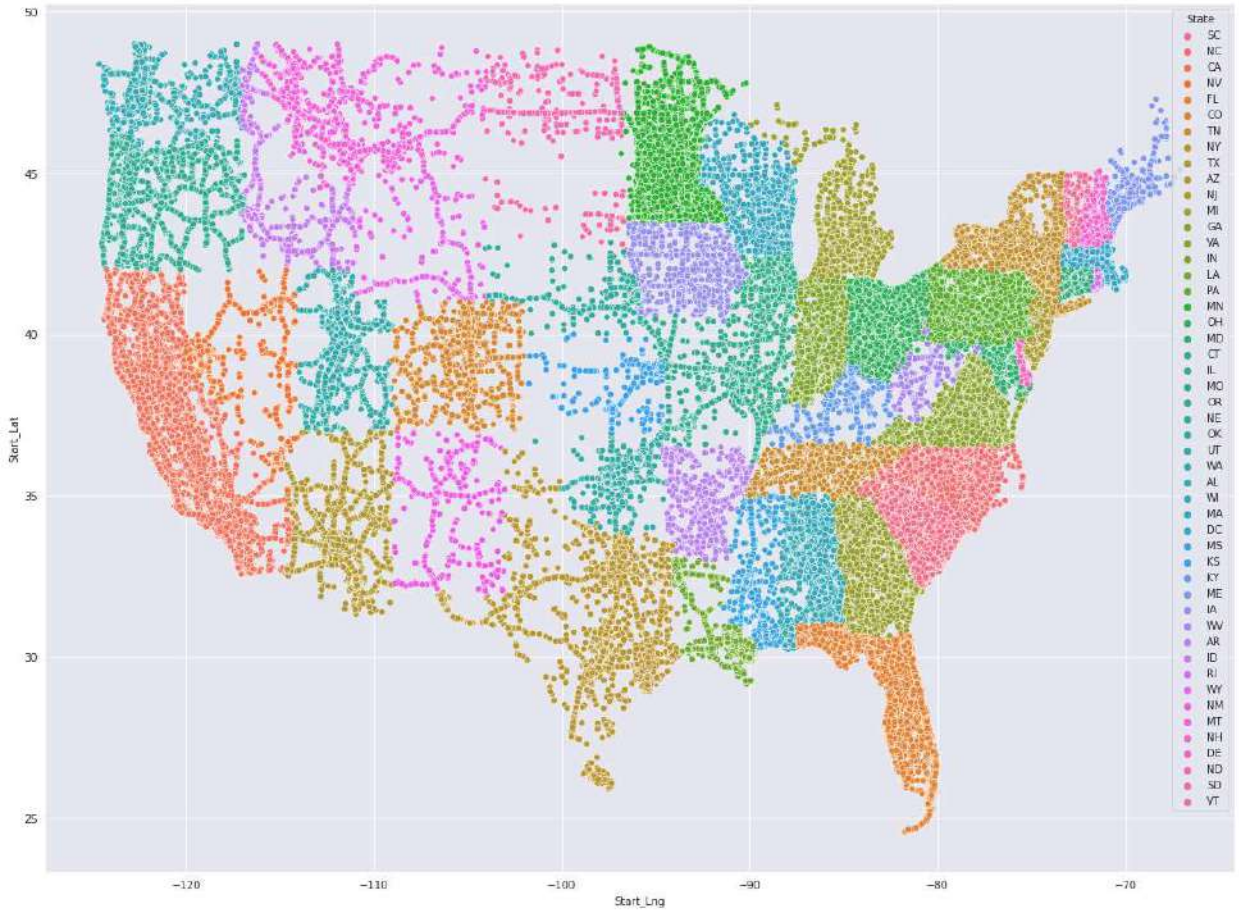
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2906610 entries, 0 to 2906609
Data columns (total 47 columns):
 #   Column                 Dtype
---  ------                 -----
 0   ID                     object
 1   Severity               int64
 2   Start_Time             datetime64[ns]
 3   End_Time               object
 4   Start_Lat              float64
 5   Start_Lng              float64
 6   End_Lat                float64
 7   End_Lng                float64
 8   Distance(mi)           float64
 9   Description            object
 10  Number                 float64
 11  Street                 object
 12  Side                   object
 13  City                   object
 14  County                 object
 15  State                  object
 16  Zipcode                object
 17  Country                object
 18  Timezone               object
 19  Airport_Code           object
 20  Weather_Timestamp      object
 21  Temperature(F)         float64
 22  Wind_Chill(F)          float64
 23  Humidity(%)            float64
 24  Pressure(in)           float64
 25  Visibility(mi)         float64
 26  Wind_Direction         object
 27  Wind_Speed(mph)        float64
 28  Precipitation(in)      float64
 29  Weather_Condition      object
 30  Amenity                bool
 31  Bump                   bool
 32  Crossing               bool
 33  Give_Way               bool
 34  Junction               bool
 35  No_Exit                bool
 36  Railway                bool
 37  Roundabout             bool
 38  Station                bool
 39  Stop                   bool
 40  Traffic_Calming        bool
 41  Traffic_Signal         bool
 42  Turning_Loop           bool
 43  Sunrise_Sunset         object
 44  Civil_Twilight         object
 45  Nautical_Twilight      object
 46  Astronomical_Twilight  object
dtypes: bool(13), datetime64[ns](1), float64(13), int64(1), object(19)
memory usage: 790.0+ MB
```

In [60]:
```
plt.figure(figsize=(20,15))
sns.scatterplot(y=df.Start_Lat, x=df.Start_Lng, hue=df.State)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2906610 entries, 0 to 2906609
Data columns (total 47 columns):
 #   Column                 Dtype
---  ------                 -----
 0   ID                     object
 1   Severity               int64
 2   Start_Time             datetime64[ns]
 3   End_Time               object
 4   Start_Lat              float64
 5   Start_Lng              float64
 6   End_Lat                float64
 7   End_Lng                float64
 8   Distance(mi)           float64
 9   Description            object
 10  Number                 float64
 11  Street                 object
 12  Side                   object
 13  City                   object
 14  County                 object
 15  State                  object
 16  Zipcode                object
 17  Country                object
 18  Timezone               object
 19  Airport_Code           object
 20  Weather_Timestamp      object
 21  Temperature(F)         float64
 22  Wind_Chill(F)          float64
 23  Humidity(%)            float64
 24  Pressure(in)           float64
 25  Visibility(mi)         float64
 26  Wind_Direction         object
 27  Wind_Speed(mph)        float64
 28  Precipitation(in)      float64
 29  Weather_Condition      object
 30  Amenity                bool
 31  Bump                   bool
 32  Crossing               bool
 33  Give_Way               bool
 34  Junction               bool
 35  No_Exit                bool
 36  Railway                bool
 37  Roundabout             bool
 38  Station                bool
 39  Stop                   bool
 40  Traffic_Calming        bool
 41  Traffic_Signal         bool
 42  Turning_Loop           bool
 43  Sunrise_Sunset         object
 44  Civil_Twilight         object
 45  Nautical_Twilight      object
 46  Astronomical_Twilight  object
dtypes: bool(13), datetime64[ns](1), float64(13), int64(1), object(19)
memory usage: 790.0+ MB
```

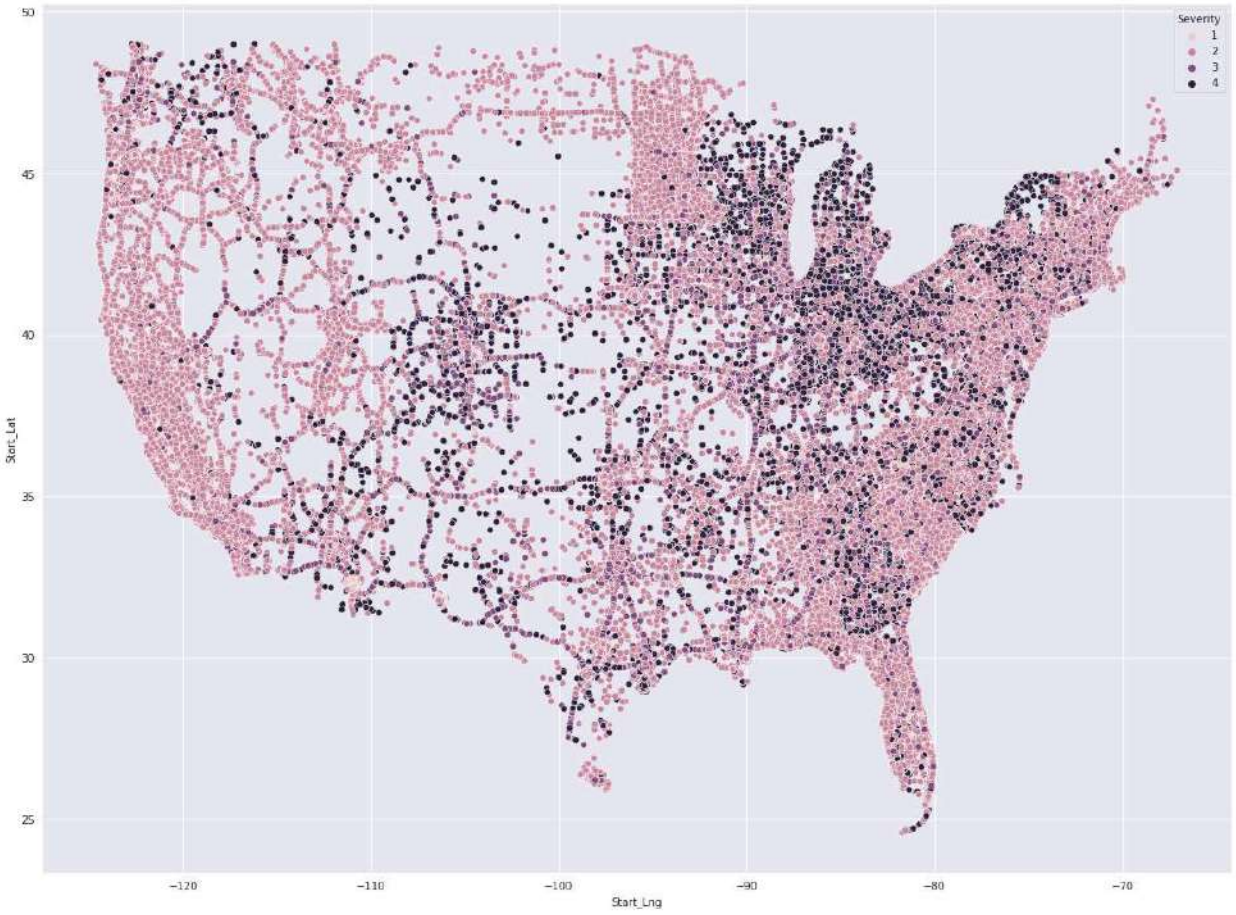
Out[60]: <matplotlib.axes._subplots.AxesSubplot at 0x7fd6e76b5ed0>

```
/usr/local/lib/python3.7/dist-packages/google/colab/_event_manager.py:28: UserWarnin
g: Creating legend with loc="best" can be slow with large amounts of data.
  func(*args, **kwargs)
/usr/local/lib/python3.7/dist-packages/IPython/core/pylabtools.py:125: UserWarning: C
reating legend with loc="best" can be slow with large amounts of data.
  fig.canvas.print_figure(bytes_io, **kw)
```



In [61]:
```
plt.figure(figsize=(20,15))
sns.scatterplot(y=df.Start_Lat, x=df.Start_Lng, hue=df.Severity)
```

Out[61]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd6e572e0d0>
```

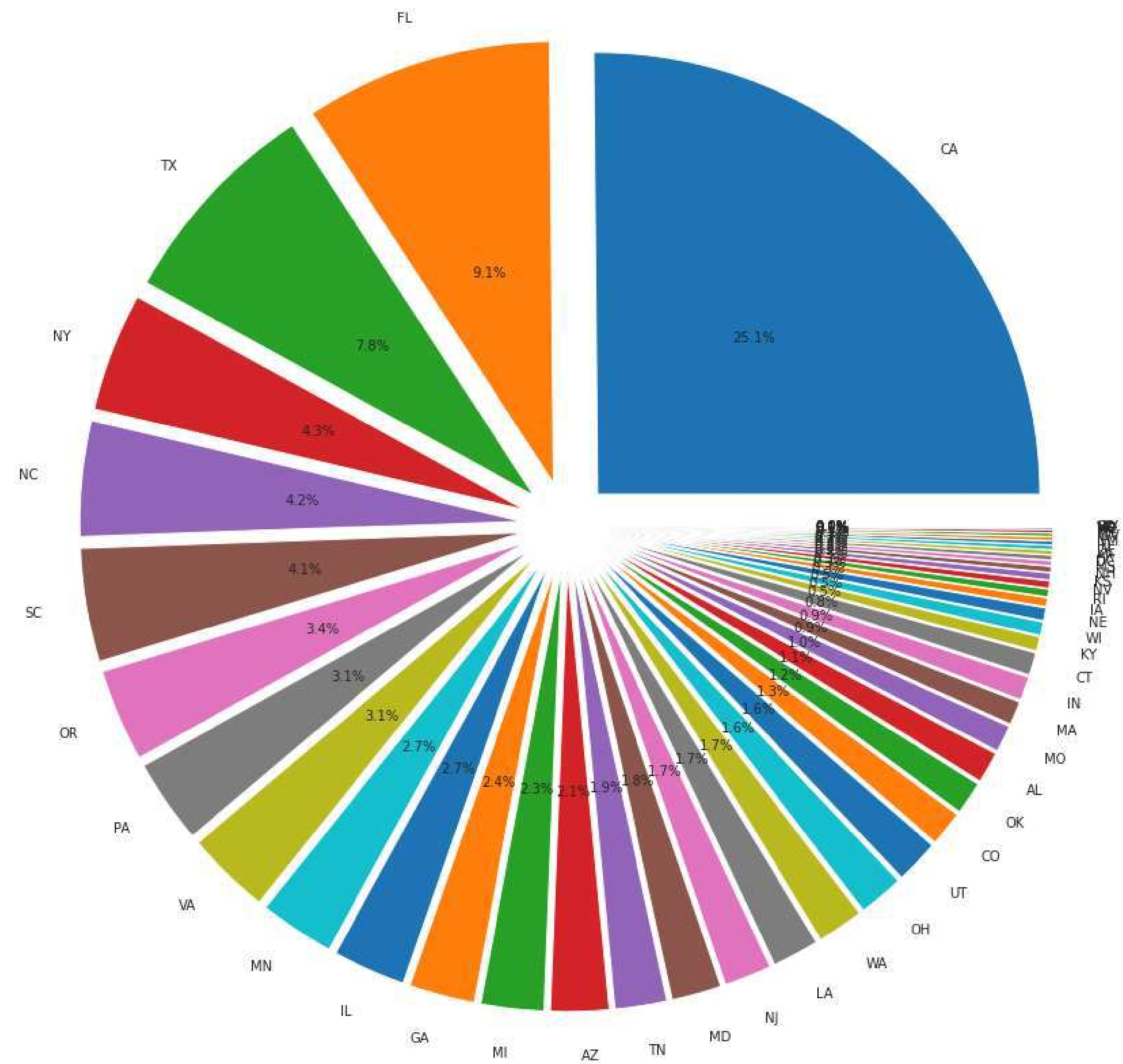


In [62]:
```
df.State.value_counts()[:25]
```

Out[62]:
```
CA    730744
FL    263300
TX    226640
NY    126176
NC    122797
SC    120462
OR     98352
PA     89745
VA     89730
MN     79712
IL     77626
GA     69536
MI     67073
AZ     61707
TN     55495
MD     52755
NJ     50214
LA     50103
WA     49455
OH     47836
UT     46897
CO     37280
OK     35105
AL     33290
MO     28674
Name: State, dtype: int64
```

In [63]:
```
pie, ax = plt.subplots(figsize=[15,15])
labels = df.State.value_counts().keys()
```

```
plt.pie(x=df.State.value_counts(), autopct="%.1f%%", explode=[0.1]*len(df.State.value_
plt.show();
```

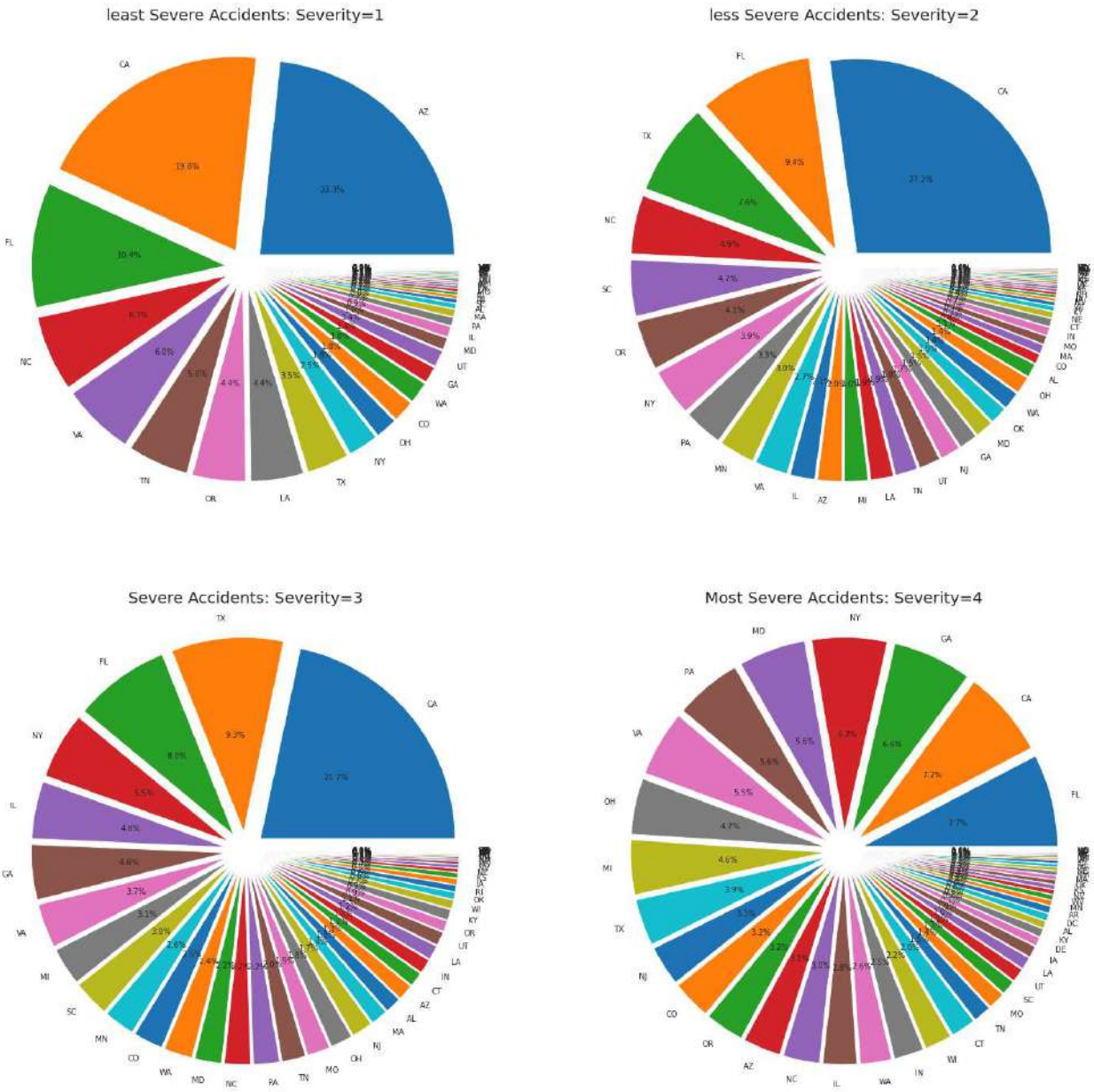


In [64]:
```
# Segregating accidents on the basis of severity
severe_accidents_4 = df[df.Severity==4].State.value_counts()
severe_accidents_3 = df[df.Severity==3].State.value_counts()
severe_accidents_2 = df[df.Severity==2].State.value_counts()
severe_accidents_1 = df[df.Severity==1].State.value_counts()
```

In [65]:
```
fig, ax1 = plt.subplots(figsize=[25,25])
ax1 = plt.subplot2grid((2,2),(0,0))
labels = severe_accidents_1.keys()
plt.pie(x=severe_accidents_1, autopct="%.1f%%", explode=[0.1]*len(severe_accidents_1),
plt.title("least Severe Accidents: Severity=1", fontsize=20)

ax1 = plt.subplot2grid((2,2),(0,1))
labels = severe_accidents_2.keys()
plt.pie(x=severe_accidents_2, autopct="%.1f%%", explode=[0.1]*len(severe_accidents_2),
plt.title("less Severe Accidents: Severity=2", fontsize=20)

ax1 = plt.subplot2grid((2,2),(1,0))
```

```
labels = severe_accidents_3.keys()
plt.pie(x=severe_accidents_3, autopct="%.1f%%", explode=[0.1]*len(severe_accidents_3),
plt.title("Severe Accidents: Severity=3", fontsize=20)

ax1 = plt.subplot2grid((2,2),(1,1))
labels = severe_accidents_4.keys()
plt.pie(x=severe_accidents_4, autopct="%.1f%%", explode=[0.1]*len(severe_accidents_4),
plt.title("Most Severe Accidents: Severity=4", fontsize=20)
```

Out[65]:
```
Text(0.5, 1.0, 'Most Severe Accidents: Severity=4')
```



Inferences from the above plot

- California generally seems to have the most accidents (in all categories)

In [67]:
```
list(zip(list(df.Start_Lat), list(df.Start_Lng)))
```

Out[67]:
```
[(34.808868, -82.26915699999998),
 (35.09008, -80.74556),
 (37.14573, -121.985052),
 (39.11039, -119.773781),
 (26.102942, -80.265091),
 (35.34824000000001, -80.84722099999998),
 (39.52397, -107.777),
 (34.034017, -118.026972),
 (35.86349000000001, -86.83168),
 (34.42633, -118.5851),
 (28.021709, -82.203583),
 (40.91221, -73.875099),
 (32.86693, -96.66617),
 (32.265141, -110.90358700000002),
 (41.05982, -74.25092),
 (29.723339000000006, -95.497337),
 (34.103172, -118.249969),
 (34.186595000000004, -117.439427),
 (42.501929, -82.918056),
 (41.556862, -73.779556),
 (33.918056, -84.33802800000002),
 (35.596561, -78.759743),
 (29.640491, -95.482445),
 (37.40691, -79.913933),
 (40.9122, -73.88461),
 (37.994461, -122.069885),
 (32.87109, -80.010628),
 (30.426109000000004, -97.753906),
 (33.774159000000004, -118.049783),
 (43.22039, -85.500961),
 (25.684458, -80.445924),
 (43.003693, -78.412064),
 (39.922646, -86.11689),
 (30.420996, -91.140549),
 (35.23932999999999, -80.856415),
 (35.05473, -80.85037),
 (34.037781, -117.320625),
 (37.504467, -77.084549),
 (33.38548, -111.9432),
 (43.100197, -77.547005),
 (40.428002, -79.92677900000002),
 (38.510303, -121.464525),
 (42.764778, -73.760338),
 (28.698406, -82.451477),
 (32.7584, -97.25763),
 (40.619857, -122.365844),
 (33.913502, -118.143219),
 (44.852409, -93.247139),
 (29.690945000000006, -95.417068),
 (34.03542, -118.274465),
 (26.61341, -80.068784),
 (42.972679, -85.677254),
 (25.685477, -80.414796),
 (46.325429, -94.962033),
 (39.826813, -84.908905),
 (38.644922, -121.383059),
 (35.25935, -80.77776),
 (40.070709, -83.134422),
 (38.48819, -77.38915),
 (37.869022, -77.453568),
```

```
(43.005009, -83.684975),
(42.967983, -85.683899),
(42.972506, -73.831128),
(34.134365, -117.957603),
(40.353738, -74.472659),
(32.78758, -96.79058),
(31.759741, -106.484001),
(32.857914, -96.651512),
(30.334675, -81.670998),
(33.940923, -117.249238),
(28.981974, -81.98734300000002),
(33.4737, -112.26643),
(32.805706, -117.156586),
(38.864871, -76.846793),
(41.755451, -72.650444),
(37.8945, -122.116587),
(39.617117, -120.591355),
(36.577457, -121.908821),
(33.900101, -84.447914),
(42.928024, -83.691177),
(39.71571, -84.219994),
(41.773296, -87.993965),
(37.328154, -121.111046),
(44.95075, -93.09934),
(34.007862, -117.52449),
(26.429920000000006, -80.090312),
(35.25833499999999, -80.794762),
(38.80037, -77.11725),
(35.266849, -119.030632),
(38.384973, -75.422516),
(37.884749, -122.308882),
(44.570877, -92.645325),
(40.703041, -73.566048),
(43.077332, -77.546585),
(29.702297, -95.359451),
(41.80175, -88.16736999999998),
(40.969318, -73.77409399999998),
(35.07973, -90.06555),
(42.229309, -88.144798),
(40.67361500000001, -73.857185),
(34.247787, -118.42585),
(37.010113, -94.548073),
(39.63915, -75.09758000000002),
(45.52641, -122.47999),
(33.043757, -111.987923),
(32.295361, -111.011978),
(41.256569, -95.940956),
(40.68459, -73.64304),
(29.849245, -95.411613),
(39.301532, -121.671249),
(34.320664, -118.495922),
(35.059368, -85.131248),
(34.153442, -118.6521),
(39.5652, -104.87226),
(44.88001, -93.14115),
(40.931221, -73.855896),
(43.875922, -95.113021),
(38.000336, -121.260314),
(38.79095, -121.29229),
(42.495152, -82.919205),
```

```
(37.74219, -121.525634),
(34.06514, -118.00026),
(33.713829, -112.285217),
(35.45930900000001, -97.491463),
(29.70736, -95.25416),
(33.95966, -81.22747),
(35.963348, -79.018662),
(33.923618, -98.503296),
(34.052890999999995, -117.548149),
(34.063072, -118.249306),
(40.820934, -78.01289799999998),
(40.69675, -111.94854),
(35.590382, -78.63269),
(43.18084, -77.59918),
(37.756392, -122.148785),
(25.942696, -80.20109599999998),
(38.799801, -77.598946),
(40.070331, -75.151593),
(39.78436, -105.08863999999998),
(45.664909, -122.434761),
(34.005801, -118.176035),
(39.966751, -76.705742),
(34.753899, -86.740997),
(33.884048, -118.351929),
(38.454845, -120.867759),
(34.04821, -118.26597),
(40.64525, -75.42576),
(39.01001, -122.05225),
(32.3116, -80.979507),
(36.171581, -86.869011),
(34.14112, -118.17972),
(33.92638, -118.378448),
(30.424627, -97.671585),
(34.071883, -117.872309),
(44.347912, -88.737572),
(42.430195, -71.10301199999998),
(45.58187, -122.67851),
(41.9589, -71.06367),
(35.375984, -118.931871),
(30.403622, -81.564785),
(34.201108000000005, -118.473406),
(38.22317, -122.61118),
(47.650143, -122.347412),
(46.21548, -123.86201),
(25.627335, -80.37943299999998),
(39.924939, -86.58699399999998),
(38.918324, -76.95503199999997),
(43.84918, -84.02081),
(26.456837, -80.089449),
(41.224018, -96.043098),
(41.177315, -73.17304200000002),
(39.014515, -77.48455),
(30.464523, -90.917725),
(45.542439, -122.484029),
(34.40578, -88.9207),
(38.017952, -121.894058),
(30.456015, -97.666321),
(41.27264, -72.970589),
(45.123852, -93.33902),
(34.557976000000004, -118.132591),
```

```
(39.841164, -75.404449),
(34.165503, -118.492347),
(45.531761, -122.565521),
(38.358604, -121.934029),
(35.585209000000006, -97.5672),
(27.76623, -82.768204),
(40.611329, -111.90618),
(35.84600800000001, -78.836472),
(33.931194, -117.993202),
(40.18096, -74.10131),
(33.544922, -81.677643),
(37.47018, -121.90886),
(35.221828, -80.828453),
(29.784912, -95.688347),
(44.31172, -93.99379),
(32.958290000000005, -97.14342),
(32.71983, -117.11757),
(39.26285, -76.56695),
(33.46147, -112.28829),
(40.903866, -74.15786700000002),
(33.68363, -117.87666000000002),
(34.693901000000004, -82.733963),
(29.21297, -81.10059),
(34.82976, -82.29406),
(39.632698, -104.906487),
(42.26919, -83.2251),
(39.976933, -75.198631),
(38.521811, -78.836011),
(38.657713, -121.363862),
(39.105335, -94.709862),
(37.267, -80.13744200000002),
(34.036971, -118.43854),
(33.49411, -86.330162),
(29.797159000000004, -95.031898),
(30.337242, -81.669167),
(35.11739, -81.306679),
(36.521611, -121.44837),
(33.895256, -84.263733),
(44.86555, -93.29896),
(32.768566, -96.674858),
(25.734321, -80.251918),
(34.698074, -82.30504599999998),
(29.79659, -95.579627),
(33.651927, -84.497727),
(37.948418, -121.300972),
(38.938068, -94.398103),
(33.65761, -117.74564),
(34.03078, -118.21791),
(29.186419, -82.185303),
(40.355076, -76.379272),
(32.472916, -93.796127),
(27.95956, -82.42554),
(37.634975, -122.086945),
(40.484029, -74.302049),
(33.997040000000005, -117.931015),
(37.789969, -121.291099),
(39.7174, -86.12572),
(45.828438, -122.843275),
(28.348536, -81.56498),
(47.463745, -122.29391499999998),
```

```
(38.23583, -85.61639),
(26.31321, -80.117207),
(40.68915300000001, -74.3014),
(35.162048, -80.970139),
(33.516411, -81.729156),
(34.09943, -117.846367),
(33.732585, -117.989329),
(36.77562, -119.784837),
(34.034725, -117.974474),
(34.259491, -118.437836),
(39.74004, -84.194923),
(38.432375, -122.715965),
(37.622681, -77.457741),
(39.865639, -88.972939),
(35.423435, -97.618668),
(37.696018, -122.115387),
(39.651718, -86.05403100000002),
(39.327599, -77.582863),
(45.041573, -93.28492),
(34.061809000000004, -118.457189),
(41.580649, -72.899661),
(44.736202, -119.191689),
(43.04277, -124.12115),
(26.46674, -80.08927),
(35.304, -80.84886),
(33.89956, -117.47298),
(38.008226, -121.871205),
(28.285931, -80.608664),
(42.385137, -83.280821),
(37.620403, -122.425995),
(33.908318, -117.882774),
(44.144381, -69.925377),
(44.030651, -92.488159),
(35.496719, -97.601158),
(40.817029, -73.836497),
(33.994858, -80.958282),
(41.597691, -93.718437),
(35.557968, -97.638351),
(33.461964, -112.151688),
(37.64955, -122.45233),
(40.654079, -74.007675),
(36.34705, -86.87468),
(30.466084, -97.844037),
(43.043734, -83.723338),
(26.70676, -80.11098),
(40.196327, -76.108688),
(38.562206, -121.487221),
(39.062382, -94.6259),
(39.343011, -120.337151),
(36.175961, -86.76619699999998),
(42.94145, -78.76647),
(29.734968, -95.347444),
(33.946648, -118.368512),
(35.551079, -97.565651),
(42.36147, -83.07294),
(34.14565, -118.75722),
(42.397328, -121.36068),
(37.54805, -77.35896),
(41.17161, -73.191683),
(40.79871, -74.189888),
```

```
(40.247822, -75.635788),
(34.433743, -118.385757),
(28.66436, -81.38871999999998),
(44.11386, -123.18217),
(35.094120000000004, -81.854828),
(42.167667, -71.937218),
(41.877819, -93.571083),
(33.535, -112.186234),
(40.70396, -111.93896),
(33.18652, -97.108223),
(34.185817, -118.217363),
(38.597375, -121.419965),
(39.88203, -86.244186),
(40.160965, -76.392044),
(34.121311, -117.514503),
(38.258007, -85.765343),
(39.187038, -77.169777),
(34.028813, -117.575703),
(42.489758, -73.677544),
(34.892540000000004, -82.160202),
(33.996331, -117.92722),
(40.73724, -73.172539),
(38.111683, -85.672958),
(43.138062, -77.60041),
(33.291782, -111.904217),
(34.994781, -81.978729),
(42.666294, -73.75489),
(44.912191, -122.97781299999998),
(30.350527000000003, -97.500694),
(33.089451000000004, -117.301628),
(34.93185, -81.08152),
(26.564581, -81.856346),
(38.59085, -121.50469),
(32.90871, -96.899732),
(34.436634000000005, -82.68933100000002),
(38.989737, -76.626846),
(34.002491, -118.411911),
(40.065258, -75.320206),
(37.552776, -77.443588),
(29.9518, -90.06726),
(34.051727, -80.923485),
(34.856169, -82.267323),
(30.63799, -90.5242),
(34.044212, -118.169844),
(38.684384, -121.72836),
(45.93963, -123.91922),
(28.51001, -81.248032),
(42.153564, -87.988268),
(39.250969, -76.681099),
(39.07931, -94.49073),
(30.44211, -91.0401),
(40.284412, -79.39363),
(41.668731, -72.844148),
(40.776439, -73.467088),
(32.475221999999995, -93.797218),
(43.297569, -73.678947),
(34.861706, -82.276009),
(30.36783, -81.66879),
(25.87242, -80.20888000000002),
(37.539156, -77.333116),
```

```
(30.435011, -84.201233),
(33.5411, -84.267738),
(39.20583, -76.68930999999998),
(29.813528, -95.317154),
(39.88358, -75.24481),
(41.843773, -87.9702),
(41.87688, -88.23516),
(42.808024, -121.835043),
(32.792156, -96.729103),
(34.066146999999994, -117.509277),
(25.945438, -80.186628),
(40.285888, -76.821141),
(38.660194, -121.071808),
(33.90214, -118.18644),
(42.14015, -121.88498),
(28.66403, -81.27516899999998),
(38.33462, -77.49722),
(40.812576, -74.39267),
(36.845222, -76.283592),
(33.901752, -118.370316),
(33.8541, -117.79493),
(41.49366, -94.23755),
(29.91696, -95.41274),
(45.082928, -93.402077),
(35.39197, -97.49846),
(35.2943, -118.757423),
(38.743896, -90.412605),
(37.282066, -121.808708),
(33.992068, -118.402353),
(33.778082, -117.789865),
(38.544779, -121.47398),
(38.55085800000001, -121.422653),
(40.83593, -73.870201),
(45.562813, -122.590408),
(47.46919000000001, -122.26836000000002),
(40.888208, -80.723656),
(25.733028, -80.318649),
(42.36298, -83.12684),
(33.67016, -117.689803),
(38.93532, -94.72042),
(40.24158900000001, -75.575129),
(35.82915900000001, -78.884323),
(45.27793, -123.010934),
(40.39204, -74.09959),
(41.22699, -95.990227),
(27.960513, -82.75910999999998),
(41.04086, -73.83776),
(36.206554, -86.77624499999997),
(34.24316, -117.28183),
(33.403724, -84.599029),
(29.905119, -95.31073),
(29.677942, -95.352493),
(27.388454, -82.487072),
(39.15658, -76.60065),
(30.877176, -88.044342),
(30.139681, -81.537231),
(34.15311, -118.36934),
(33.46222, -112.047433),
(34.06132, -117.396399),
(39.74847000000001, -77.56276700000002),
```

```
(34.128815, -118.346886),
(32.17783, -110.975853),
(32.779443, -117.112515),
(37.959408, -120.236236),
(39.020229, -94.175552),
(40.426502, -80.05695300000002),
(38.299835, -122.034061),
(44.91861, -122.99738),
(39.870079, -75.005981),
(42.99213, -83.71045),
(36.02177800000001, -86.654488),
(34.19957, -118.338249),
(42.03027, -88.27771),
(31.77798, -106.46249),
(35.240379, -80.920937),
(33.411137, -86.97020699999999),
(26.203583, -80.141755),
(38.85297, -77.39779),
(38.93507, -94.733157),
(47.16736, -122.47563),
(41.299488, -96.014259),
(38.714451, -90.447556),
(37.34535, -77.339744),
(29.840179, -95.38967),
(32.663940000000004, -96.90086),
(45.44009000000001, -122.56884),
(33.551689, -117.672981),
(34.774818, -80.790314),
(42.24531, -71.199),
(28.069836, -82.354057),
(39.12569000000001, -121.57576),
(37.846089, -122.48667),
(39.154588, -94.587209),
(38.86012, -77.17886999999997),
(41.88232100000001, -112.481903),
(38.45626, -121.94223999999998),
(41.99173, -87.87095),
(41.989578, -87.86792),
(38.84218, -104.81505),
(40.440636, -122.287979),
(28.171101, -81.301735),
(29.625639000000003, -95.605909),
(36.26545, -86.664017),
(35.2719, -89.98331999999998),
(34.065228000000005, -117.806824),
(30.23708, -93.25941),
(42.97337, -85.64375),
(32.778492, -96.916878),
(39.75475, -86.14495),
(33.87631, -118.12106),
(41.033882, -74.07123100000003),
(33.860405, -118.31263),
(32.223412, -110.909508),
(44.95071, -93.101265),
(41.595047, -73.764015),
(44.848248, -93.085258),
(33.916758, -118.285871),
(25.875916, -80.169012),
(42.218056, -87.845528),
(40.944124, -77.729827),
```

```
(37.7855, -122.391823),
(42.291234, -87.904834),
(40.66045, -73.8672),
(42.279732, -87.87854),
(28.74878, -81.93167),
(40.59617, -75.33734),
(39.53414, -77.920006),
(33.0448, -117.28601),
(34.739841, -86.662178),
(40.792934, -73.82396700000002),
(35.761955, -78.744987),
(42.298378, -83.878075),
(40.891945, -79.81309499999998),
(37.929001, -122.057755),
(32.93891, -80.04954000000002),
(42.874249, -78.787849),
(47.649307, -122.34946399999998),
(47.668671, -122.376221),
(41.565275, -72.774369),
(37.37431, -77.40599),
(34.177111, -118.4685),
(41.190815, -96.08139),
(37.192772, -77.327789),
(40.552364, -74.31774899999998),
(34.293619, -118.470243),
(40.62268, -80.09659),
(40.704742, -73.995041),
(38.646744, -90.530266),
(32.811832, -96.817413),
(40.80842, -73.94493),
(30.428614000000003, -97.681252),
(37.55187, -122.30963),
(30.158553, -97.776615),
(26.61306, -81.793472),
(40.629429, -111.90229),
(43.012321, -85.657928),
(34.366240999999995, -117.65341200000002),
(35.827309, -78.620354),
(34.148075, -118.469818),
(32.749599, -116.92968799999998),
(29.68305, -95.349144),
(35.27271500000001, -80.836258),
(36.990612, -121.557852),
(42.51616, -73.60991),
(44.76519, -93.78126),
(37.699375, -121.92906200000002),
(45.222591, -93.029099),
(38.906714, -120.842419),
(36.239861, -119.42112),
(30.656706, -87.905559),
(34.016742, -118.173354),
(37.316757, -121.922081),
(26.16483, -80.23935999999998),
(39.572697, -119.807961),
(33.49127, -86.82158000000003),
(40.764152, -73.724091),
(40.644755, -77.293184),
(39.665768, -121.743866),
(34.641855, -118.209961),
(28.630995, -81.387535),
```

```
(40.263206, -76.181976),
(30.44323, -91.153587),
(40.70204, -73.48705),
(33.922749, -118.029207),
(33.912348, -117.446456),
(37.898679, -122.070733),
(34.704551, -82.64201),
(42.212754, -71.14894699999998),
(33.431629, -79.142303),
(32.773254, -96.856499),
(30.069355, -95.183022),
(34.07893, -118.227524),
(39.092316, -104.862534),
(40.130443, -76.873456),
(43.26641, -119.83921),
(32.825161, -86.620613),
(42.98843, -87.984047),
(39.752129, -84.176109),
(35.888927, -78.71189100000002),
(33.58017, -117.6716),
(29.378901000000006, -98.403954),
(30.135435, -92.009377),
(42.792507, -73.762527),
(39.16785, -76.62316),
(43.83709, -95.15798),
(38.67138, -121.58608999999998),
(38.818534, -76.91265899999998),
(37.76345300000001, -77.460447),
(40.255001, -75.64179200000002),
(37.21507, -76.59236999999997),
(32.873901000000004, -96.897476),
(41.322926, -95.956635),
(40.96344000000001, -81.465019),
(42.35776, -71.17735),
(47.2564, -122.51485),
(40.111813, -75.28514100000002),
(29.739517, -95.381279),
(47.7086, -122.32862),
(33.739643, -84.328194),
(32.69132800000001, -96.823269),
(42.912819, -85.647102),
(47.60147, -122.32485),
(36.606567, -119.662682),
(37.387119, -77.419726),
(38.691372, -104.719327),
(39.529675, -119.829788),
(41.668608, -72.922841),
(38.16735, -122.20279),
(34.019241, -118.422203),
(42.01990900000001, -91.550224),
(36.931141, -76.266922),
(38.87091, -76.98875),
(33.90034, -117.47285),
(41.90542, -87.98835),
(34.762852, -82.545387),
(39.157853, -76.831834),
(36.666962, -121.756935),
(35.921349, -84.064949),
(44.918663, -73.739655),
(33.815277, -118.185081),
```

```
(42.334366, -71.095314),
(43.554192, -73.40007),
(38.654709, -121.036942),
(38.321544, -121.32546),
(28.697873, -82.104239),
(33.946217, -117.38871),
(39.23406, -76.70889),
(33.83368, -84.33232),
(33.753777, -118.29126),
(35.151089, -118.823372),
(33.751446, -84.446114),
(34.23274, -118.47299),
(33.563098, -117.656415),
(42.02602, -87.81624000000002),
(40.14737, -84.21929),
(38.23587, -85.658455),
(37.639245, -121.007109),
(43.02751, -88.01263),
(39.514629, -118.856812),
(33.930969, -118.300323),
(36.170837, -86.76417),
(32.612583, -83.742805),
(42.42837100000001, -71.25882),
(38.24572, -122.625526),
(35.073608, -85.18820699999998),
(38.56368300000001, -121.636625),
(45.163956, -93.276695),
(40.029678, -76.493042),
(37.362774, -121.887871),
(29.766043, -95.455971),
(29.729774, -95.605743),
(27.95617, -82.4642),
(33.764568, -84.384727),
(42.494469, -82.938179),
(37.710659, -122.166817),
(34.114746000000004, -80.888672),
(26.010791, -80.166826),
(29.784473, -95.468704),
(27.885455, -82.787019),
(34.331055, -81.609154),
(32.624374, -83.36969),
(30.35552, -81.66866999999998),
(32.515087, -93.748596),
(30.340569, -97.554581),
(33.78989, -84.49068),
(34.03119, -118.03068),
(34.145805, -81.23786199999998),
(25.686405, -80.35164499999998),
(34.257332, -118.853742),
(35.748638, -78.834824),
(25.963083, -80.15353499999998),
(33.963913, -117.324977),
(42.61479, -87.95227),
(35.258881, -89.871643),
(35.93572, -79.33326),
(41.07552, -73.802017),
(44.96603, -93.265116),
(42.264896, -83.271271),
(33.54546, -112.047539),
(32.830425, -117.23262),
```

```
(43.001141, -77.670395),
(33.87628, -118.13109),
(34.741413, -86.678698),
(38.596024, -121.401497),
(38.011761, -121.86622),
(33.782509999999995, -118.35237),
(27.441839, -82.459595),
(38.231153000000006, -122.112747),
(37.603195, -122.064285),
(37.31749, -121.935356),
(33.687782, -117.906548),
(44.879065, -93.002742),
(43.04929, -75.233482),
(42.909031, -83.604378),
(35.75377, -78.64441),
(40.37692, -111.823003),
(29.843294, -82.530411),
(38.959431, -85.845001),
(35.217152, -80.82399000000002),
(33.791645, -84.391693),
(38.790883, -77.176147),
(37.742855, -122.198219),
(32.774640000000005, -117.13446),
(33.555344, -112.043892),
(39.83181, -75.42732),
(33.483948, -86.82634),
(28.450769, -82.557404),
(34.15005900000001, -118.468437),
(34.559639000000004, -111.904762),
(34.617359, -82.324852),
(36.880207, -119.976631),
(42.36788, -87.89508000000002),
(36.751447, -119.793557),
(43.196243, -77.576782),
(34.74968, -82.50083199999997),
(34.145939, -84.516632),
(25.883393, -80.361758),
(30.137239, -92.066566),
(40.70215, -73.48571),
(38.26073, -122.0606),
(42.978516, -77.232529),
(44.775101, -93.288506),
(46.93051, -120.502127),
(44.78704000000001, -68.80843),
(37.33453, -122.027931),
(42.231777, -87.830811),
(39.811752, -86.176788),
(41.932301, -87.678116),
(34.12536, -118.47654),
(39.87425, -75.10239),
(38.85122, -77.07574),
(41.099137, -73.440099),
(31.771512, -106.377172),
(39.210904, -120.783464),
(25.889368, -80.206795),
(33.418018, -86.687416),
(33.96021, -117.908268),
(40.4189, -74.208412),
(41.0613, -73.51018),
(27.792072, -82.66069),
```

```
(34.042334000000004, -118.773637),
(34.033696, -118.361083),
(35.25859000000001, -112.14472),
(39.964121, -77.24828199999997),
(39.871022, -75.10212),
(30.412809000000006, -97.696831),
(37.702255, -121.744728),
(34.17626, -83.91631),
(40.653603, -111.98176200000002),
(29.73068, -95.484161),
(35.229172, -80.865356),
(34.129452, -86.41124),
(40.060783, -76.359779),
(30.475215, -91.111308),
(41.8153, -87.63048),
(36.141407, -86.666557),
(34.381173, -118.413408),
(41.178349, -73.19519),
(34.082256, -117.698196),
(34.003284, -117.380767),
(28.37145, -81.40438),
(29.675388, -95.550169),
(41.94935200000001, -71.340848),
(30.543131, -91.028587),
(40.71806, -112.11437),
(40.25008, -75.388733),
(38.012596, -122.002274),
(40.06489600000001, -75.32122),
(38.617735, -121.382879),
(30.18701, -81.73946),
(34.028675, -118.228828),
(34.037999, -118.276449),
(34.912468, -82.429283),
(44.94409, -93.07786),
(36.758495, -76.30364399999998),
(29.61107, -95.4883),
(40.854481, -73.96579),
(37.670472, -122.110379),
(34.129723, -117.933701),
(33.58918, -111.83617),
(40.28586, -74.71434),
(33.738712, -117.091906),
(36.66982, -80.708903),
(34.021034, -118.195198),
(44.951866, -93.146957),
(32.898563, -96.769188),
(41.82402, -78.35730699999998),
(39.59483, -104.802345),
(34.837709999999994, -82.28967),
(32.79575, -96.6922),
(37.09657, -80.563202),
(37.540244, -120.895759),
(33.876289, -118.102577),
(33.394573, -80.64431),
(34.16108, -118.46961),
(40.59523, -73.65686),
(33.85611, -118.00633),
(34.865355, -82.433279),
(33.8784, -118.19254),
(35.777813, -78.560852),
```

```
(33.787947, -117.880076),
(44.81494, -120.93174),
(46.301641, -94.267719),
(39.77345, -104.83529),
(41.95929, -88.089287),
(40.709063, -73.819822),
(40.683006, -75.280905),
(41.581806, -93.703514),
(44.952644, -93.041512),
(34.035315999999995, -118.149963),
(41.73481, -72.66427),
(34.392521, -118.472832),
(34.82251, -82.288048),
(34.908939000000004, -91.196554),
(34.070496, -117.392105),
(34.124851, -117.509501),
(27.903095, -82.636016),
(33.230209, -86.80720500000002),
(37.595968, -122.059106),
(30.008568, -90.219955),
(36.07215, -86.73288000000002),
(41.477081, -74.312302),
(34.186409000000005, -80.211472),
(34.230522, -79.798828),
(30.35021, -81.66854000000002),
(42.17422, -84.658953),
(33.392072, -117.174211),
(47.478865, -92.871927),
(43.65889, -116.657),
(33.88998, -117.50985),
(29.735954, -95.410675),
(29.784895, -95.64492),
(43.01559, -121.82597),
(37.800674, -121.192345),
(37.326692, -121.94071),
(42.75262, -70.94408),
(33.674887, -117.8384),
(43.594297, -116.448243),
(39.95111, -75.18283000000002),
(40.004341, -83.15387700000002),
(28.480616, -81.310112),
(37.886386, -120.999881),
(37.75015300000001, -121.138657),
(33.72029000000001, -84.50253000000002),
(38.630543, -121.498053),
(28.548094, -81.260651),
(41.93976, -85.65168),
(30.389618, -87.067521),
(32.709896, -97.024734),
(33.80271, -84.38578199999998),
(39.326984, -120.39026),
(42.378181, -83.06276700000002),
(42.06832, -88.29720999999998),
(48.07196, -122.111336),
(39.268185, -84.36068),
(33.933499, -117.000073),
(45.69356, -120.382422),
(30.260363, -97.794464),
(34.06446, -118.00388),
(39.102356, -84.498451),
```

```
(32.67099, -96.97586),
(32.734257, -96.688438),
(45.29932, -93.51108),
(37.771553, -121.810371),
(35.715639, -78.58413399999998),
(34.00634, -118.41371200000002),
(38.77343, -121.24282),
(35.8242, -78.587456),
(37.608707, -122.068192),
(34.155396, -118.291702),
(40.430185, -79.880839),
(39.07679, -77.6064),
(41.85697, -88.16093000000002),
(40.844837, -73.904137),
(40.91078, -74.50895),
(45.575668, -94.124184),
(39.69667000000001, -104.81922),
(34.442143, -119.784885),
(40.936253, -123.625681),
(45.375734, -122.58606),
(33.916225, -118.223022),
(39.651531, -77.800186),
(40.044579, -76.490997),
(32.751759, -96.770355),
(32.852734000000005, -79.98790699999998),
(30.41016, -84.137787),
(34.25581, -111.256042),
(37.8952, -122.11238),
(41.90015, -87.66059),
(33.928740000000005, -118.25041),
(29.9466, -95.41623),
(39.11364, -94.892532),
(41.527344, -73.916061),
(34.916744, -82.00431800000001),
(29.981182, -90.11367),
(35.106468, -80.76664699999998),
(41.029095, -74.251366),
(33.934631, -84.494003),
(30.23735, -93.26431),
(34.078987, -118.291693),
(47.661381, -122.336151),
(37.807884, -121.297215),
(45.062237, -93.159867),
(36.845505, -76.196342),
(32.746639, -97.30658),
(33.629433, -84.40289),
(32.759537, -97.062378),
(40.42476, -76.52409),
(28.041037, -82.558578),
(40.847298, -72.883698),
(33.968163, -118.16787),
(32.67951, -96.82267),
(34.01334, -117.34403),
(33.954479, -80.924461),
(28.001841, -82.30246),
(32.792442, -96.685028),
(35.05104, -80.76953),
(37.768478, -122.006729),
(32.95215, -117.10351),
(35.833744, -77.978882),
```

```
(34.02502, -117.73398),
(37.092485, -80.573437),
(34.025956, -118.040102),
(45.079498, -93.052864),
(46.23506, -123.87196000000002),
(40.760189, -74.051147),
(33.75119, -84.4963),
(45.77352, -92.8301),
(33.74071, -117.83391),
(34.018436, -117.984634),
(25.94235, -80.34227),
(38.94522, -94.85307),
(35.221835, -92.833131),
(34.155727, -118.431236),
(45.2133, -122.82353),
(30.10803, -95.435562),
(41.213799, -96.110283),
(34.147237, -118.69888600000002),
(37.892591, -122.159524),
(34.020633000000004, -118.192258),
(34.186754, -118.474275),
(42.40348, -83.09566),
(38.64056400000001, -121.47773),
(40.5018, -74.44404),
(33.901897, -118.291794),
(42.19334, -120.37615),
(36.261977, -86.738043),
(36.289661, -119.117767),
(39.884051, -76.160831),
(40.10962, -79.38226),
(41.035675, -73.816696),
(25.630294, -80.41471800000002),
(41.96341, -87.74721),
(25.77773, -80.302995),
(34.878521, -82.361328),
(36.95016500000001, -121.471695),
(38.737267, -77.192484),
(40.03244, -111.758102),
(33.90651, -84.43203000000003),
(27.576923, -82.42546999999998),
(38.911391, -76.579894),
(33.856840000000005, -118.284809),
(43.327518, -73.68463100000002),
(38.863026, -77.369875),
(47.608002, -122.29628),
(37.759823, -122.392258),
(33.797779, -118.293724),
(41.219711, -95.953331),
(33.986732, -117.906235),
(45.244923, -123.047989),
(33.7677, -117.92028),
(37.390098, -121.991977),
(30.224709000000004, -92.052452),
(34.104040000000005, -118.319696),
(36.168259, -86.782036),
(35.15279, -80.774117),
(29.872042, -95.262558),
(34.303802000000005, -118.478905),
(38.628149, -120.208637),
(34.055695, -118.451965),
```

```
(34.15424, -118.375259),
(35.76886, -78.736099),
(40.85306, -73.96011),
(39.514046, -77.319283),
(30.184739, -81.637291),
(45.053483, -93.058029),
(35.827312, -78.620384),
(27.567875, -82.568765),
(30.160873, -81.747263),
(42.783209, -86.07985),
(42.840167, -73.483921),
(42.945187, -83.62281800000002),
(27.93755, -82.32975),
(33.575444, -81.740948),
(43.174896, -77.613722),
(44.697693, -93.288147),
(33.77817, -117.86739),
(29.624031, -95.430809),
(30.425835, -91.010925),
(34.00444, -118.4122),
(28.727290000000004, -81.60181),
(38.84737, -77.42849),
(26.13603, -80.219009),
(33.743111, -118.002724),
(26.6217, -81.71521),
(36.941958, -80.917374),
(38.16722, -122.20299),
(44.784027, -93.20884),
(37.56174, -122.27629),
(26.45408, -80.17444),
(40.758507, -73.854675),
(28.635424, -81.448527),
(38.70697, -77.223),
(40.785793, -77.8545),
(35.83492000000001, -78.60642),
(42.066444, -88.22333499999998),
(45.069469, -93.285942),
(47.636021, -122.227669),
(42.912641, -85.626609),
(45.507012, -122.670754),
...]
```

In [69]:
```
import random
```

In [70]:
```
df_sample = df.sample(10000)
```

In [71]:
```
df_sample.Start_Lat
```

Out[71]:
```
668907     41.938133
448021     33.385742
4511       30.272743
1371228    35.863110
2211536    33.553101
             ...
1029356    38.697596
1149812    42.525455
1018798    43.118233
175231     34.088684
1985548    32.632549
Name: Start_Lat, Length: 10000, dtype: float64
```

In [72]:
```
df.columns
```

Out[72]:
```
Index(['ID', 'Severity', 'Start_Time', 'End_Time', 'Start_Lat', 'Start_Lng',
       'End_Lat', 'End_Lng', 'Distance(mi)', 'Description', 'Number', 'Street',
       'Side', '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 [112…
```
df['Visibility(mi)']
```

Out[112]:
```
0          10.0
1          10.0
2          10.0
3          10.0
4          10.0
           ...
2906605     9.0
2906606    10.0
2906607    10.0
2906608    10.0
2906609    10.0
Name: Visibility(mi), Length: 2906610, dtype: float64
```

In [113…
```
df['Visibility(mi)'].value_counts()
```

Out[113]:
```
10.0     2260327
7.0        87566
9.0        75270
8.0        60090
5.0        56646
           ...
3.2            1
19.0           1
54.0           1
101.0          1
130.0          1
Name: Visibility(mi), Length: 81, dtype: int64
```

In [74]:
```
df[(df.Severity == 4) & (df['Visibility(mi)'] <=10)]  # data when severity is high and
```

Out[74]:

	ID	Severity	Start_Time	End_Time	Start_Lat	Start_Lng	End_Lat	End_Lng	Di
6	A-7	4	2019-12-12 09:48:52	2019-12-12 10:18:05	39.523970	-107.777000	39.565780	-107.516950	
40	A-41	4	2020-10-26 00:38:00	2020-10-26 02:41:25	40.428002	-79.926779	40.426058	-79.895801	
146	A-147	4	2016-08-20 01:31:43	2016-08-20 07:31:43	40.645250	-75.425760	40.630360	-75.471790	
167	A-168	4	2019-07-13 16:14:30	2019-07-13 16:42:44	43.849180	-84.020810	43.737277	-84.016961	
196	A-197	4	2018-07-09 01:08:19	2018-07-09 07:08:19	32.719830	-117.117570	32.716810	-117.119700	
...	...	...	...	...	...	...	...	...	
2906549	A-2906550	4	2018-08-26 00:25:54	2018-08-26 02:25:53	25.941730	-80.189260	25.950490	-80.181540	
2906554	A-2906555	4	2020-04-23 19:18:22	2020-04-23 19:47:31	39.144030	-84.559830	39.137680	-84.548380	
2906568	A-2906569	4	2019-11-09 03:55:18	2019-11-09 04:22:55	29.182560	-82.184640	29.191010	-82.184590	
2906577	A-2906578	4	2017-03-07 16:01:38	2017-03-07 22:01:38	36.573645	-79.847221	36.559341	-79.826802	
2906591	A-2906592	4	2020-12-24 08:23:00	2020-12-24 11:25:20	41.785073	-86.139762	41.785570	-86.137596	

114602 rows × 47 columns

In [106…
```
(len(df[df['Visibility(mi)'] <=2]) / len(df) )* 100. # total percentage of accidents i
```

Out[106]:
```
4.375578422973843
```

In [107…
```
(len(df[(df['Visibility(mi)'] <=2) & (df['Severity'] ==4)]) / len(df) ) * 100. # total
```

Out[107]:
```
0.21203395020315763
```

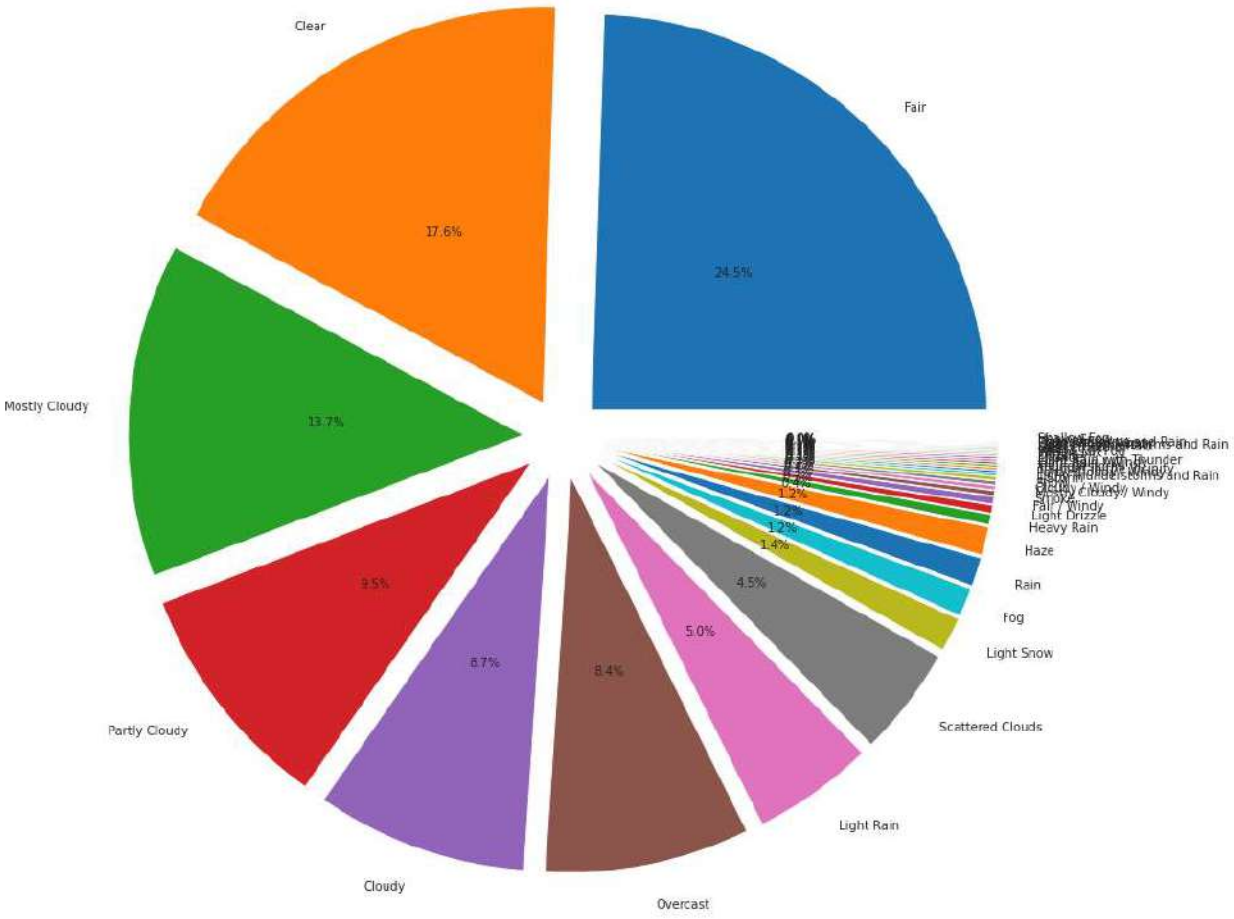
In [78]:
```
weather = df.Weather_Condition.value_counts()
```

In [79]:
```
weather[weather > 1000]  # Kind of weather when no. of accidents were greater than 100
```

Out[79]:
```
Fair                            692680
Clear                           498925
Mostly Cloudy                   386122
Partly Cloudy                   268851
Cloudy                          245054
Overcast                        237068
Light Rain                      140946
Scattered Clouds                127090
Light Snow                       39941
Fog                              33424
Rain                             33383
Haze                             32993
Heavy Rain                       12340
Light Drizzle                     9484
Fair / Windy                      9121
Smoke                             6037
Mostly Cloudy / Windy             5100
Cloudy / Windy                    4773
Snow                              4589
T-Storm                           3313
Light Thunderstorms and Rain      3089
Partly Cloudy / Windy             3054
Thunder in the Vicinity           2829
Thunderstorm                      2801
Light Rain / Windy                2653
Light Rain with Thunder           2595
Thunder                           2314
Drizzle                           2025
Wintry Mix                        1875
Patches of Fog                    1854
Mist                              1757
Heavy T-Storm                     1748
Light Freezing Rain               1489
Heavy Thunderstorms and Rain      1475
Light Snow / Windy                1418
Thunderstorms and Rain            1415
Heavy Snow                        1232
Shallow Fog                       1150
Name: Weather_Condition, dtype: int64
```

In [80]:
```
import matplotlib.pyplot as plt
```

In [81]:
```
pie, ax = plt.subplots(figsize=[15,15])
labels = weather[weather > 1000].keys()
plt.pie(x=weather[weather > 1000], autopct="%.1f%%", explode=[0.1]*len(weather[weather
plt.show();
```



In [111…
```
df['Temperature(F)']
```

Out[111]:
```
0          76.0
1          76.0
2          51.0
3          53.6
4          84.2
           ...
2906605    84.2
2906606    46.9
2906607    76.0
2906608    27.0
2906609    51.1
Name: Temperature(F), Length: 2906610, dtype: float64
```

In [119…
```
df['Temperature(F)'].value_counts()
```

Out[119]:
```
68.0     62008
59.0     60192
77.0     59625
73.0     57029
63.0     56585
         ...
132.6        1
1.6          1
-19.3        1
-5.4         1
-21.5        1
Name: Temperature(F), Length: 822, dtype: int64
```

In [83]:
```
temperature = df['Temperature(F)'].value_counts()
```

In [120…
```
temperature.index
```

Out[120]:
```
Float64Index([ 68.0,  59.0,  77.0,  73.0,  63.0,  64.0,  72.0,  70.0,  61.0,
               75.0,
              ...
              170.6, -12.1, 161.6, 203.0, -15.2, 132.6,   1.6, -19.3,  -5.4,
              -21.5],
             dtype='float64', length=822)
```

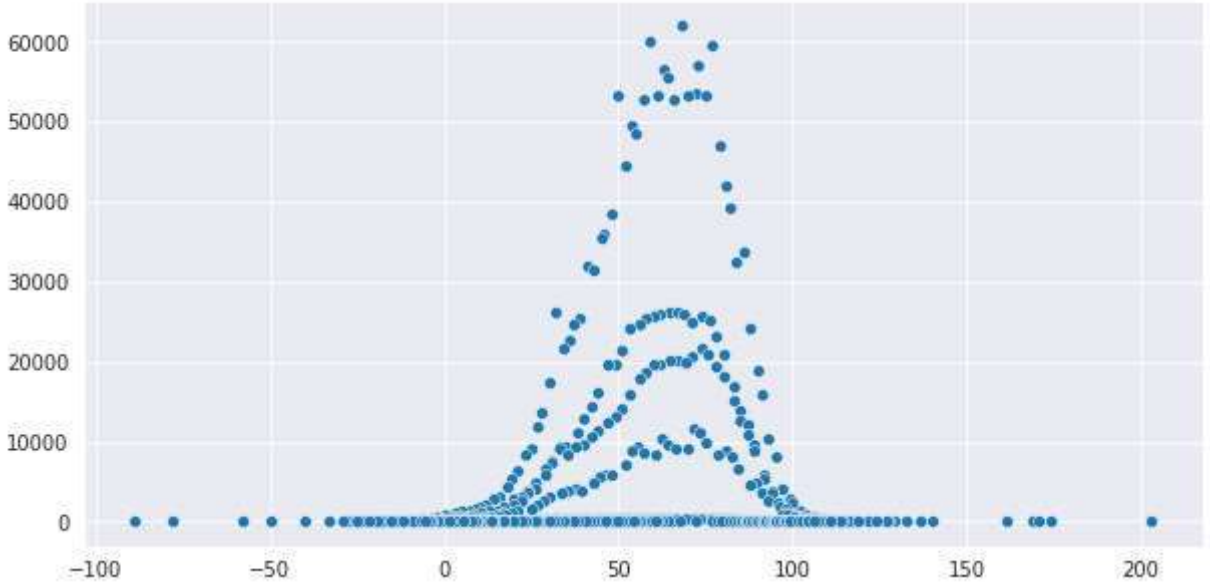
In [121…
```
temperature.values
```

Out[121]:
```
array([62008, 60192, 59625, 57029, 56585, 55573, 53610, 53400, 53399,
       53394, 53284, 52807, 52794, 49587, 48477, 46922, 44543, 42121,
       39295, 38519, 36097, 35541, 33786, 32382, 32046, 31512, 26320,
       26300, 26291, 26010, 25873, 25714, 25709, 25532, 25471, 25108,
       25064, 24747, 24625, 24278, 24151, 23135, 22800, 21793, 21706,
       21325, 20935, 20869, 20661, 20270, 20094, 20004, 19785, 19661,
       19660, 19570, 19424, 18874, 18686, 18266, 18049, 17393, 16880,
       16052, 15940, 15820, 15213, 14382, 14228, 13897, 13690, 13164,
       12810, 12601, 12519, 12234, 11984, 11655, 11443, 11261, 11085,
       10824, 10676, 10471, 10287,  9912,  9741,  9671,  9557,  9516,
        9458,  9339,  9331,  9252,  9178,  9144,  9062,  9010,  8895,
        8848,  8755,  8504,  8413,  8356,  8302,  8248,  8132,  7652,
        7462,  7089,  6676,  6514,  6487,  5933,  5853,  5769,  5764,
        5515,  5457,  5381,  4860,  4819,  4767,  4755,  4355,  4248,
        4222,  4190,  4000,  3848,  3770,  3768,  3623,  3581,  3559,
        3504,  3226,  3121,  3072,  2933,  2868,  2811,  2722,  2683,
        2612,  2476,  2317,  2160,  2048,  2037,  1875,  1827,  1803,
        1728,  1640,  1579,  1470,  1384,  1309,  1290,  1286,  1254,
        1140,  1127,  1086,  1058,  1033,   990,   985,   964,   912,
         903,   844,   838,   815,   791,   634,   631,   625,   601,
         594,   588,   568,   524,   506,   503,   498,   490,   443,
         435,   435,   434,   416,   409,   407,   407,   400,   391,
         386,   379,   362,   362,   357,   355,   353,   352,   350,
         350,   349,   347,   345,   345,   343,   331,   331,   330,
         328,   328,   325,   325,   323,   322,   321,   317,   317,
         315,   315,   315,   314,   311,   310,   310,   309,   309,
         308,   308,   307,   306,   306,   305,   304,   301,   298,
         298,   298,   298,   297,   297,   295,   293,   293,   293,
         292,   292,   292,   290,   290,   289,   289,   289,   289,
         286,   286,   286,   285,   285,   284,   284,   281,   280,
         278,   278,   278,   277,   277,   275,   274,   274,   273,
         272,   272,   271,   271,   268,   267,   267,   266,   265,
         264,   262,   262,   261,   260,   260,   259,   258,   257,
         257,   257,   255,   254,   253,   253,   253,   252,   252,
         250,   250,   250,   250,   249,   248,   248,   248,   247,
         246,   245,   244,   244,   244,   243,   243,   242,   242,
         242,   241,   241,   240,   238,   238,   237,   236,   234,
         234,   234,   234,   232,   231,   230,   230,   229,   228,
         226,   226,   226,   225,   225,   225,   224,   224,   224,
         224,   223,   223,   222,   222,   222,   222,   221,   221,
         220,   219,   219,   219,   218,   218,   217,   217,   217,
         216,   215,   214,   214,   214,   213,   212,   212,   212,
         211,   211,   210,   209,   209,   208,   207,   206,   206,
         206,   206,   204,   204,   204,   204,   203,   201,   200,
         199,   199,   199,   196,   196,   195,   192,   192,   192,
         191,   190,   190,   190,   189,   189,   188,   188,   187,
         187,   186,   185,   184,   183,   183,   183,   182,   181,
         179,   179,   178,   175,   174,   173,   173,   170,   170,
         170,   170,   169,   168,   167,   166,   166,   165,   165,
         164,   164,   163,   163,   162,   161,   160,   159,   158,
         156,   156,   156,   154,   153,   151,   149,   148,   147,
         146,   145,   144,   144,   143,   143,   141,   138,   138,
         136,   136,   135,   134,   131,   131,   130,   129,   126,
         126,   126,   123,   122,   122,   120,   119,   119,   118,
         118,   114,   114,   112,   112,   108,   106,   105,   105,
         103,    99,    98,    98,    98,    95,    94,    94,    92,
          92,    91,    89,    88,    87,    84,    84,    84,    82,
          80,    77,    76,    75,    75,    72,    72,    72,    72,
          72,    71,    71,    69,    69,    69,    68,    68,    67,
          67,    67,    65,    64,    64,    61,    61,    60,    59,
```

```
56,    56,    56,    55,    55,    55,    54,    53,    53,
52,    52,    52,    51,    51,    50,    49,    46,    44,
44,    42,    41,    41,    41,    40,    40,    39,    39,
39,    38,    37,    37,    36,    36,    36,    35,    35,
35,    35,    34,    33,    33,    33,    32,    32,    32,
32,    32,    31,    31,    30,    29,    29,    29,    29,
28,    28,    28,    27,    27,    27,    27,    25,    25,
25,    25,    24,    24,    24,    24,    24,    23,    23,
23,    23,    23,    22,    22,    22,    21,    20,    19,
19,    19,    18,    18,    18,    18,    17,    17,    16,
16,    16,    15,    15,    15,    14,    14,    14,    14,
14,    14,    14,    14,    14,    14,    13,    13,    13,
13,    13,    13,    12,    12,    12,    12,    12,    12,
11,    11,    11,    11,    10,    10,    10,    10,    10,
10,     9,     9,     9,     9,     9,     9,     9,     9,
 8,     8,     8,     8,     8,     8,     8,     8,     8,
 8,     8,     7,     7,     7,     7,     7,     7,     7,
 7,     7,     7,     7,     7,     7,     7,     7,     7,
 6,     6,     6,     6,     6,     6,     6,     6,     6,
 6,     6,     5,     5,     5,     5,     5,     5,     5,
 4,     4,     4,     4,     4,     4,     4,     4,     4,
 4,     4,     3,     3,     3,     3,     3,     3,     3,
 3,     3,     3,     3,     3,     3,     3,     3,     3,
 3,     3,     3,     3,     3,     3,     3,     2,     2,
 2,     2,     2,     2,     2,     2,     2,     2,     2,
 2,     2,     2,     2,     2,     2,     2,     2,     2,
 2,     2,     2,     1,     1,     1,     1,     1,     1,
 1,     1,     1,     1,     1,     1,     1,     1,     1,
 1,     1,     1,     1,     1,     1,     1,     1,     1,
 1,     1,     1,     1,     1,     1,     1,     1,     1,
 1,     1,     1,     1,     1,     1,     1,     1,     1,
 1,     1,     1])
```

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

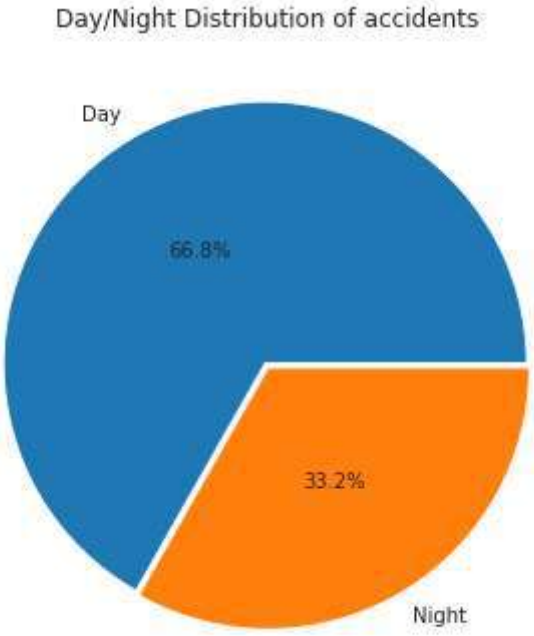
In [129…
```
plt.figure(figsize=(10,5))
sns.scatterplot(x=temperature.index, y=temperature.values)
plt.show();
```



In [100…
```
df.Sunrise_Sunset.value_counts()
```

Out[100]:
```
Day      1941068
Night     965432
Name: Sunrise_Sunset, dtype: int64
```

In [101…
```
pie, ax = plt.subplots(figsize=[6,6])
labels = df.Sunrise_Sunset.value_counts().keys()
plt.pie(x=df.Sunrise_Sunset.value_counts(), autopct="%.1f%%", explode=[0.01]*len(df.Su
plt.title("Day/Night Distribution of accidents")
plt.show();
```



In [88]:
```
df.columns
```

Out[88]:
```
Index(['ID', 'Severity', 'Start_Time', 'End_Time', 'Start_Lat', 'Start_Lng',
       'End_Lat', 'End_Lng', 'Distance(mi)', 'Description', 'Number', 'Street',
       'Side', '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 [86]:
```
amenity = df.Amenity.groupby(df.Severity).value_counts()
amenity
```

Out[86]:
```
Severity  Amenity
1         False        28261
          True           490
2         False      2102046
          True         27217
3         False       627000
          True          2452
4         False       117933
          True          1211
Name: Amenity, dtype: int64
```

In [87]:
```
amenity.index
```

Out[87]:
```
MultiIndex([(1, False),
            (1,  True),
            (2, False),
            (2,  True),
            (3, False),
            (3,  True),
            (4, False),
            (4,  True)],
           names=['Severity', 'Amenity'])
```

In [89]:
```
no_exit = df.No_Exit.groupby(df.Severity).value_counts()
no_exit
```

Out[89]:
```
Severity  No_Exit
1         False        28631
          True           120
2         False      2126312
          True          2951
3         False       628809
          True           643
4         False       119000
          True           144
Name: No_Exit, dtype: int64
```

In [90]:
```
railway = df.Railway.groupby(df.Severity).value_counts()
railway
```

Out[90]:
```
Severity  Railway
1         False        28209
          True           542
2         False      2108779
          True         20484
3         False       625458
          True          3994
4         False       118237
          True           907
Name: Railway, dtype: int64
```

In [91]:
```
traffic_calming = df.Traffic_Calming.groupby(df.Severity).value_counts()
traffic_calming
```

Out[91]:
```
Severity  Traffic_Calming
1         False              28738
          True                  13
2         False            2128279
          True                984
3         False             629186
          True                266
4         False             119100
          True                 44
Name: Traffic_Calming, dtype: int64
```

In [92]:
```
stop = df.Stop.groupby(df.Severity).value_counts()
stop
```

Out[92]:
```
Severity  Stop
1         False      28251
          True         500
2         False    2088803
          True       40460
3         False     626761
          True        2691
4         False     117341
          True        1803
Name: Stop, dtype: int64
```

In [93]:
```
traffic_signal = df.Traffic_Signal.groupby(df.Severity).value_counts()
traffic_signal
```

Out[93]:
```
Severity  Traffic_Signal
1         False              16201
          True               12550
2         False            1742209
          True              387054
3         False              587341
          True               42111
4         False              107194
          True               11950
Name: Traffic_Signal, dtype: int64
```

In [94]:
```
give_way = df.Give_Way.groupby(df.Severity).value_counts()
give_way
```

Out[94]:
```
Severity  Give_Way
1         False         28658
          True             93
2         False       2122933
          True           6330
3         False        628086
          True           1366
4         False        118713
          True            431
Name: Give_Way, dtype: int64
```

In [95]:
```
bump = df.Bump.groupby(df.Severity).value_counts()
bump
```

Out[95]:
```
Severity  Bump
1         False      28741
          True          10
2         False    2128794
          True         469
3         False     629364
          True          88
4         False     119132
          True          12
Name: Bump, dtype: int64
```

In [96]:
```
crossing = df.Crossing.groupby(df.Severity).value_counts()
crossing
```

Out[96]:
```
Severity  Crossing
1         False         19673
          True           9078
2         False       1940027
          True         189236
3         False         614822
          True          14630
4         False         113159
          True           5985
Name: Crossing, dtype: int64
```

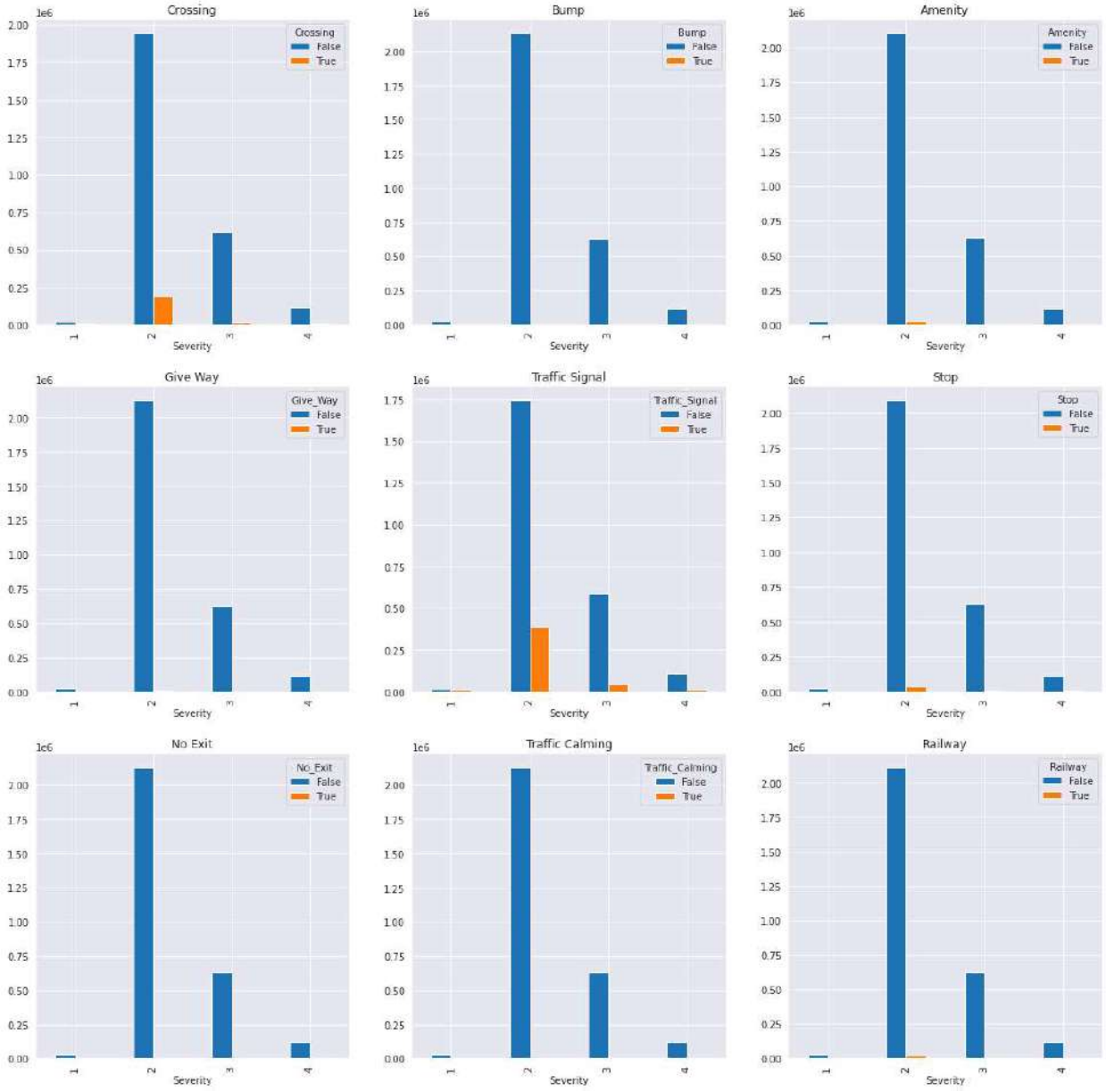
In [97]:
```
df.Turning_Loop.value_counts()
```

Out[97]:
```
False    2906610
Name: Turning_Loop, dtype: int64
```

Plotting all the values

In [99]:
```
fig, ax = plt.subplots(3,3, figsize=(20, 20))

crossing.unstack().plot(kind='bar', ax=ax[0,0], title="Crossing")
bump.unstack().plot(kind='bar', ax=ax[0,1], title="Bump")
amenity.unstack().plot(kind='bar', ax=ax[0,2], title="Amenity")
give_way.unstack().plot(kind='bar', ax=ax[1,0], title="Give Way")
traffic_signal.unstack().plot(kind='bar', ax=ax[1,1], title="Traffic Signal")
stop.unstack().plot(kind='bar', ax=ax[1,2], title="Stop")
no_exit.unstack().plot(kind='bar', ax=ax[2,0], title="No Exit")
traffic_calming.unstack().plot(kind='bar', ax=ax[2,1], title="Traffic Calming")
railway.unstack().plot(kind='bar', ax=ax[2,2], title="Railway")
```

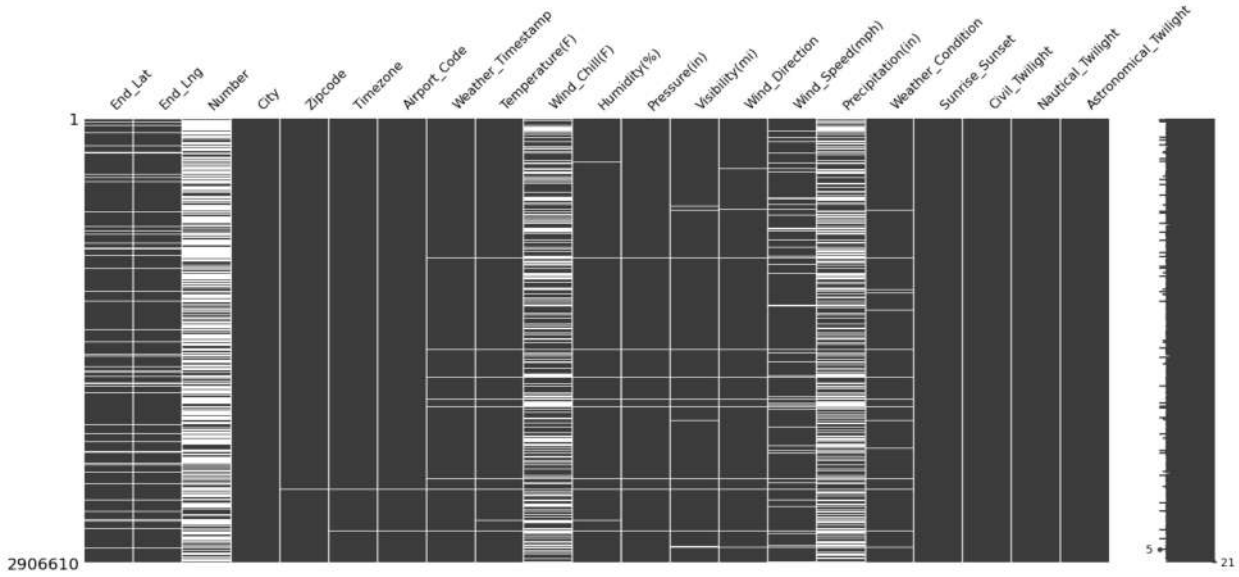
Out[99]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd6ca355590>
```



In [16]:
```
null_cols = [i for i in data.columns if data[i].isnull().any()]
print(null_cols)
```

```
['End_Lat', 'End_Lng', 'Number', 'City', 'Zipcode', 'Timezone', 'Airport_Code', 'Weather_Timestamp', 'Temperature(F)', 'Wind_Chill(F)', 'Humidity(%)', 'Pressure(in)', 'Visibility(mi)', 'Wind_Direction', 'Wind_Speed(mph)', 'Precipitation(in)', 'Weather_Condition', 'Sunrise_Sunset', 'Civil_Twilight', 'Nautical_Twilight', 'Astronomical_Twilight']
```

In [17]:
```
mn.matrix(data[null_cols]);
```



In [19]:
```
new_data_a = data.drop(columns=["End_Lng", "End_Lat", "Number"], axis=0)
```

In [20]:
```
new_data_b = new_data_a.dropna(subset = ['Visibility(mi)','Weather_Condition','Humidit
```

In [21]:
```
new_data_b.isnull().sum()
```

Out[21]:

ID	0
Severity	0
Start_Time	0
End_Time	0
Start_Lat	0
Start_Lng	0
Distance(mi)	0
Description	0
Street	0
Side	0
City	0
County	0
State	0
Zipcode	0
Country	0
Timezone	0
Airport_Code	0
Weather_Timestamp	0
Temperature(F)	0
Wind_Chill(F)	1090741
Humidity(%)	0
Pressure(in)	0
Visibility(mi)	0
Wind_Direction	0
Wind_Speed(mph)	229916
Precipitation(in)	1228285
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 [22]:
```
final_data = new_data_b.drop(columns = 'ID', axis=0)
```

In [23]:
```
final_data.isnull().sum()
```

Out[23]:

Severity	0
Start_Time	0
End_Time	0
Start_Lat	0
Start_Lng	0
Distance(mi)	0
Description	0
Street	0
Side	0
City	0
County	0
State	0
Zipcode	0
Country	0
Timezone	0
Airport_Code	0
Weather_Timestamp	0
Temperature(F)	0
Wind_Chill(F)	1090741
Humidity(%)	0
Pressure(in)	0
Visibility(mi)	0
Wind_Direction	0
Wind_Speed(mph)	229916
Precipitation(in)	1228285
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

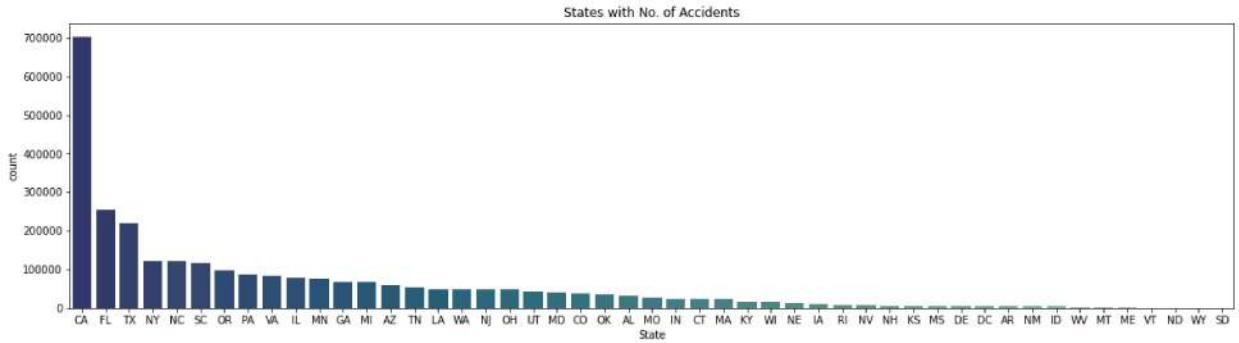
dtype: int64

In [26]:
```
state_counts = final_data["State"].value_counts()
fig = go.Figure(data=go.Choropleth(locations=state_counts.index, z=state_counts.values
fig.update_layout(title_text="Number of Accidents for each State", geo_scope="usa")
fig.show()
```

In [27]:
```
print("State Code: ", final_data.State.unique())
print("Total No. of State in Dataset: ", len(final_data.State.unique()))
```

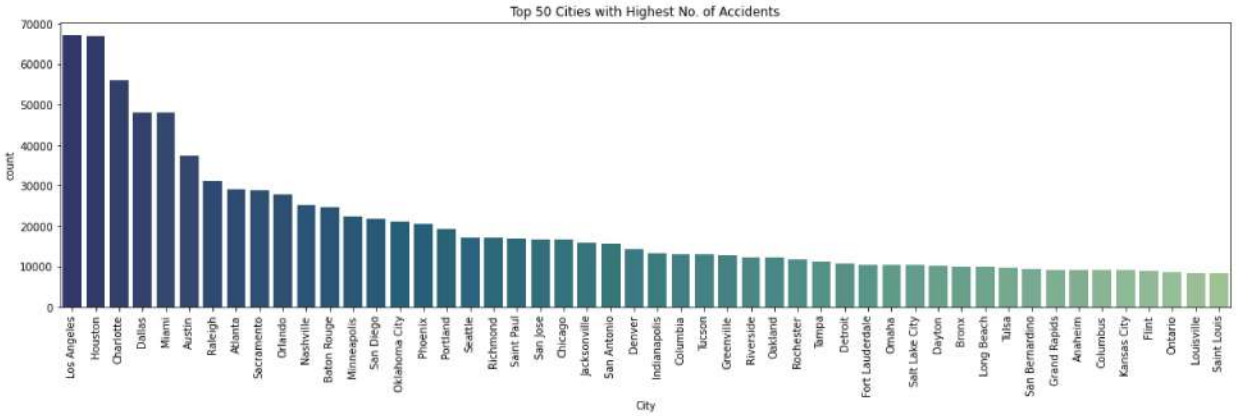
```
State Code:  ['SC' 'NC' 'CA' 'NV' 'FL' 'CO' 'TN' 'NY' 'TX' 'AZ' 'NJ' 'MI' 'GA' 'VA'
 'IN' 'LA' 'PA' 'MN' 'OH' 'CT' 'IL' 'MD' 'MO' 'OR' 'NE' 'OK' 'UT' 'WA'
 'AL' 'WI' 'MA' 'DC' 'MS' 'KY' 'ME' 'IA' 'KS' 'WV' 'AR' 'ID' 'RI' 'WY'
 'NM' 'MT' 'NH' 'DE' 'ND' 'SD' 'VT']
Total No. of State in Dataset:  49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

In [28]:
```
fig, ax = plt.subplots(figsize = (20,5))
c = sns.countplot(x="State", data=final_data, orient = 'v', palette = "crest_r", order
c.set_title("States with No. of Accidents");
```



In [30]:
```
fig, ax = plt.subplots(figsize = (20,5))
c = sns.countplot(x="City", data=final_data, order=final_data.City.value_counts().iloc
```

```
c.set_title("Top 50 Cities with Highest No. of Accidents")
c.set_xticklabels(c.get_xticklabels(), rotation=90)
plt.show()
```

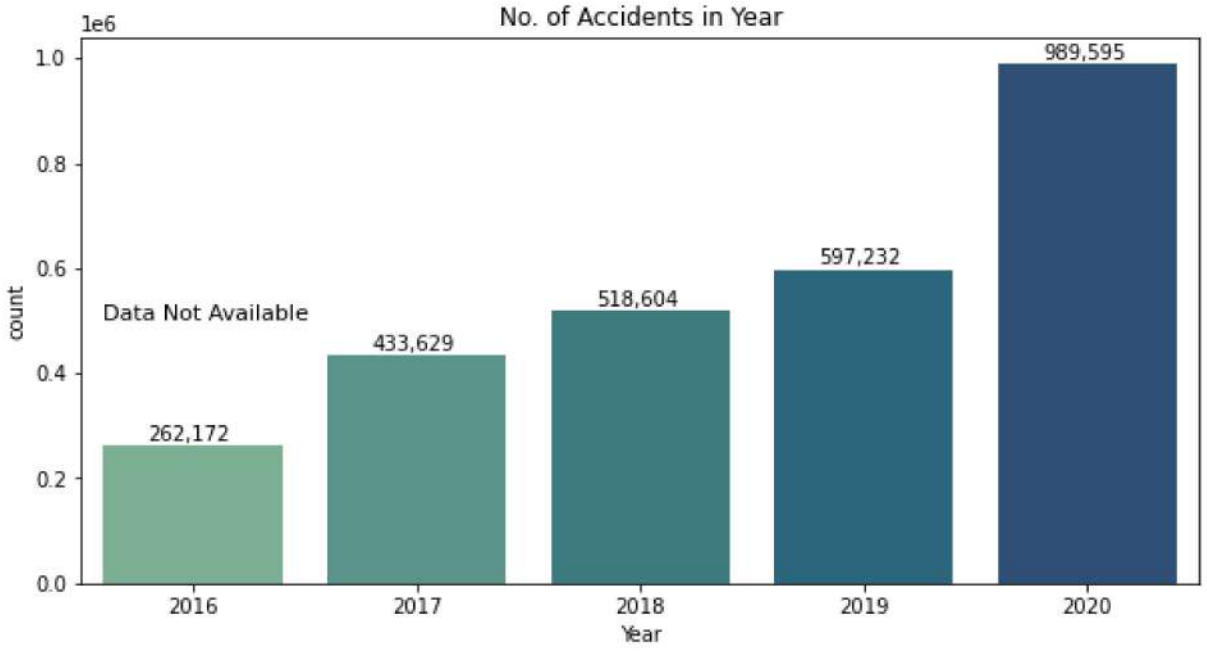


In [31]:
```
final_data.Start_Time = pd.to_datetime(final_data.Start_Time)
final_data.Start_Time[0]
```

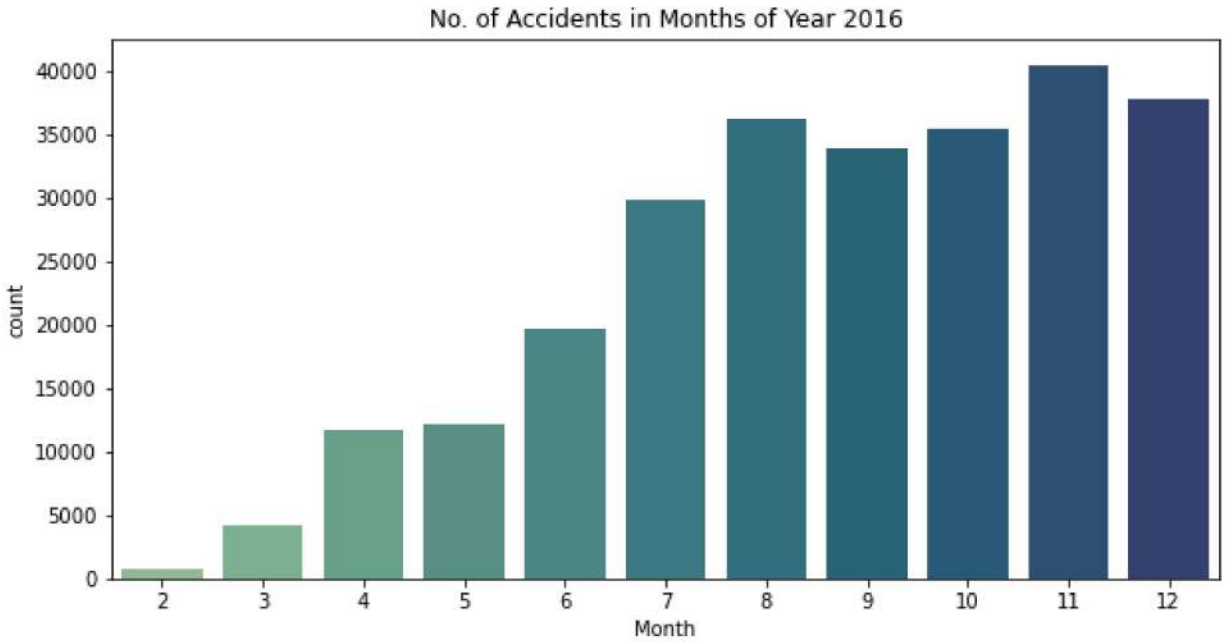
Out[31]:
```
Timestamp('2019-05-21 08:29:55')
```

In [32]:
```
final_data['Month'] = final_data['Start_Time'].dt.month
final_data['Year'] = final_data['Start_Time'].dt.year
final_data['Hour'] = final_data['Start_Time'].dt.hour
final_data['Weekday'] = final_data['Start_Time'].dt.weekday
#yearly data subset
data_2016 = final_data[final_data.Start_Time.dt.year == 2016]
data_2017 = final_data[final_data.Start_Time.dt.year == 2017]
data_2018 = final_data[final_data.Start_Time.dt.year == 2018]
data_2019 = final_data[final_data.Start_Time.dt.year == 2019]
data_2020 = final_data[final_data.Start_Time.dt.year == 2020]
data_2017_2019 = final_data[(final_data["Year"] >= 2017) & (final_data["Year"] <= 2019
```

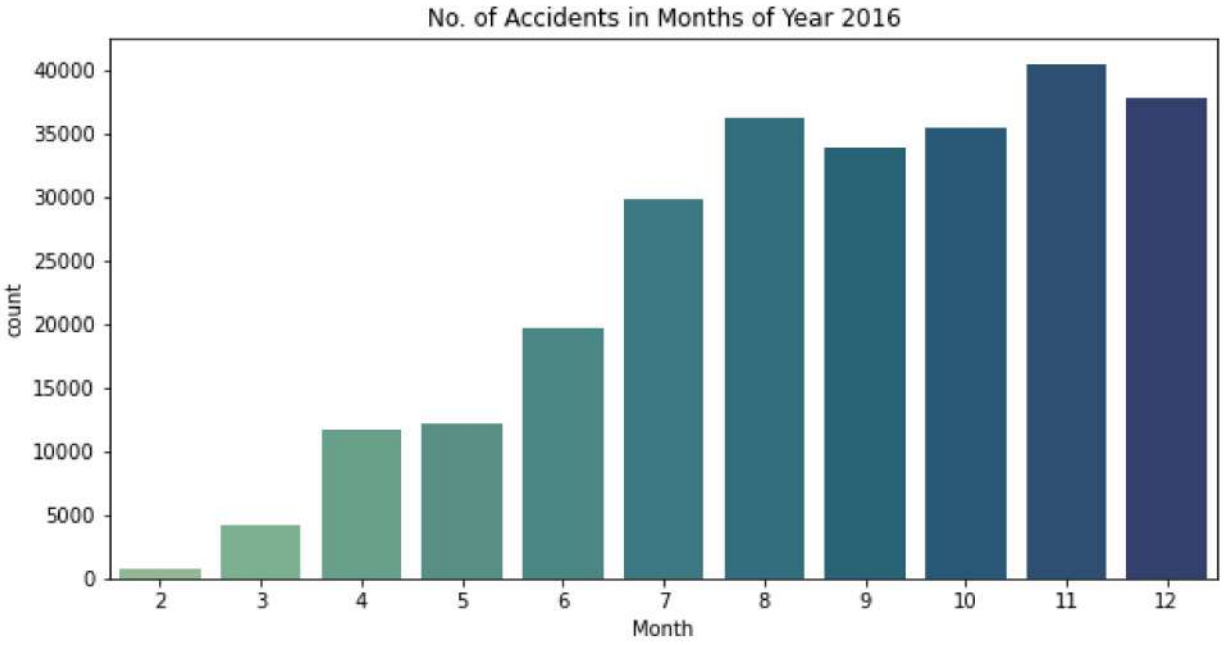
In [33]:
```
fig, ax = plt.subplots(figsize = (10,5))
c = sns.countplot(x="Year", data=final_data, orient = 'v', palette = "crest")
plt.annotate('Data Not Available',xy=(-0.4,500000), fontsize=11)
c.set_title("No. of Accidents in Year")
for i in ax.patches:
    count = '{:,.0f}'.format(i.get_height())
    x = i.get_x()+i.get_width()-0.60
    y = i.get_height()+10000
    ax.annotate(count, (x, y))
plt.show()
```



In [34]:
```
fig, ax = plt.subplots(figsize = (10,5))
c = sns.countplot(x="Month", data=data_2016, orient = 'v', palette = "crest")
plt.annotate('Data Not Available',xy=(2,50000), fontsize=11)
c.set_title("No. of Accidents in Months of Year 2016")
plt.show()
```



In [34]:
```
fig, ax = plt.subplots(figsize = (10,5))
c = sns.countplot(x="Month", data=data_2016, orient = 'v', palette = "crest")
plt.annotate('Data Not Available',xy=(2,50000), fontsize=11)
c.set_title("No. of Accidents in Months of Year 2016")
plt.show()
```



In [36]:
```
fig, ax = plt.subplots(figsize = (10,5))
c = sns.countplot(x="Month", data=data_2020, orient = 'v', palette = "crest")
plt.annotate('Covid-19 Pandemic',xy=(2,150000), fontsize=12)
plt.annotate("[",xy=(0,0),xytext=(1.9,150000),arrowprops={'arrowstyle':'-|>'}, fontsiz
plt.annotate("]",xy=(10,0),xytext=(4.5,150000),arrowprops={'arrowstyle':'-|>'}, fontsi
c.set_title("No. of Accidents in Month of Year 2020")
plt.show()
```