```
import pandas as pd

data_filepath = "/content/drive/MyDrive/Kaggle/US_Accidents/US_Accidents_Dec20_Updated.csv"

df = pd.read_csv(data_filepath)
df.head(10)
```

	ID	Severity	Start_Time	End_Time	Start_Lat	Start_Lng	End_Lat	End_Lng	Distance(mi)	Description	Number	Street	Si
0	A-1	2	2019-05-21 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	Tanner Rd	
1	A-2	2	2019-10-07 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	Providence Branch Ln	
2	A-3	2	2020-12-13 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 Cruz Hwy	
3	A-4	2	2018-04-17 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	US Highway 395 S	
4	A-5	3	2016-08-31 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-595 W	
5	A-6	3	2018-10-17 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 W.T. Harris Blvd	
6	A-7	4	2019-12-12 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	I-70 E	
7	A-8	2	2019-12-21 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-60 W	
8	A-9	2	2018-05-23 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	Old Peytonsville Rd	
9	A-10	2	2019-01-30 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	Golden State Fwy S	

```
# Checking the columns in the data
df.columns
```

```
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')
```

```
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

```
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
```

```
df.describe()
```

	Severity	Start_Lat	Start_Lng	End_Lat	End_Lng	Distance(mi)	Number	Temperature(F)	Wind_Chill(F)	H
count	2.906610e+06	2.906610e+06	2.906610e+06	2.623789e+06	2.623789e+06	2.906610e+06	1.014938e+06	2.839386e+06	1.722751e+06	2
mean	2.288649e+00	3.653027e+01	-9.642676e+01	3.651733e+01	-9.620367e+01	3.980541e-01	6.789728e+03	6.098873e+01	5.499048e+01	6
std	5.541618e-01	5.013964e+00	1.775412e+01	5.016609e+00	1.765971e+01	1.592556e+00	1.697225e+04	1.845258e+01	2.219542e+01	2
min	1.000000e+00	2.455527e+01	-1.246238e+02	2.455527e+01	-1.246238e+02	0.000000e+00	0.000000e+00	-8.900000e+01	-8.900000e+01	1
25%	2.000000e+00	3.366453e+01	-1.178232e+02	3.364659e+01	-1.177020e+02	0.000000e+00	9.650000e+02	4.890000e+01	3.900000e+01	4
50%	2.000000e+00	3.609977e+01	-9.116690e+01	3.605898e+01	-9.105163e+01	0.000000e+00	3.093000e+03	6.300000e+01	5.800000e+01	6
75%	3.000000e+00	4.037505e+01	-8.085814e+01	4.033133e+01	-8.084679e+01	2.790000e-01	7.976000e+03	7.500000e+01	7.200000e+01	8
max	4.000000e+00	4.900220e+01	-6.711317e+01	4.907500e+01	-6.710924e+01	3.336300e+02	9.999997e+06	2.030000e+02	1.740000e+02	1

How many columns are numerical data?

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

```
14
```

Missing or incorrect values?

```
df.isnull().sum()
```

```
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
```

```
df.isna().sum()
```

```
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?

```
df.isna().sum().sort_values(ascending=False) * 100. / len(df)
```

```
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

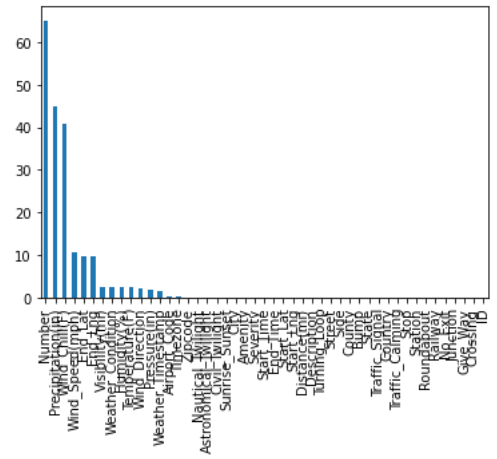
```
# Plotting a Pandas.Series data
missing_data = df.isna().sum().sort_values(ascending=False) * 100. / len(df)
```

```
type(missing_data) # we can directly plot the Pandas.Series using plot()
```

```
pandas.core.series.Series
```

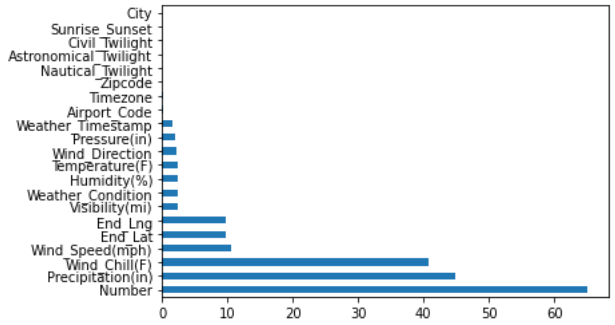
```
missing_data.plot(kind='bar')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd7016fd910>
```



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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd701690750>
```




```
# Printing all the columns
df.columns
```

```
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')
```

```
df.City.unique()
```

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

```
cities = df.City.unique()
len(cities)
```

```
11790
```

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

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

```
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
```

```
'New York' in cities
```

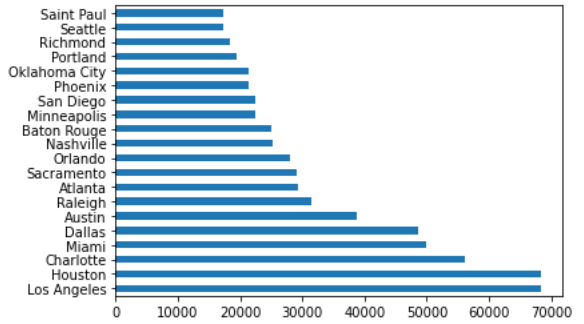
```
True
```

```
cities_by_accident["New York"]
```

```
7328
```

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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd701090c90>
```

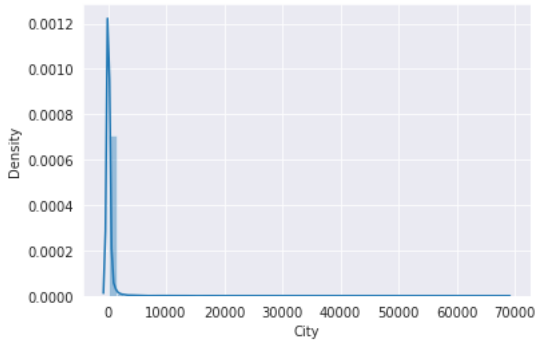



```
import seaborn as sns
sns.set_style("darkgrid")
```

Plotting all the citis by number of accidents accidents

```
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 rem
  warnings.warn(msg, FutureWarning)
<matplotlib.axes._subplots.AxesSubplot at 0x7fd6f25f0fd0>
```



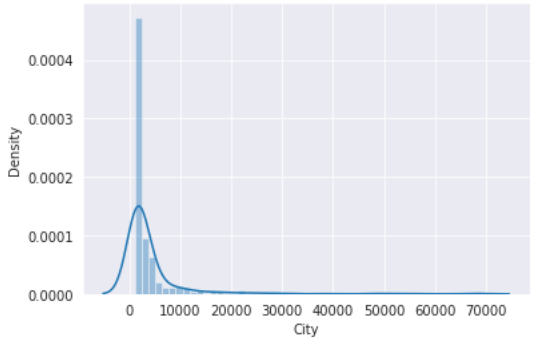

```
high_accident_cities = cities_by_accident[cities_by_accident >=1000] # having over 1000 accidents
low_accident_cities = cities_by_accident[cities_by_accident < 1000]  # having less than 1000 accidents
```

```
# Percentage of high accident cities
len(high_accident_cities) / len(cities_by_accident)
```

```
0.04351514123335313
```

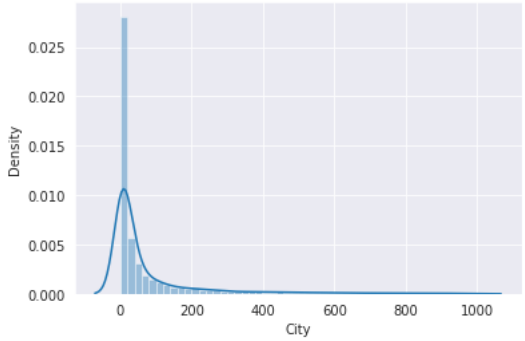
```
# Distribution of high accident cities
sns.distplot(high_accident_cities)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be rem
  warnings.warn(msg, FutureWarning)
<matplotlib.axes._subplots.AxesSubplot at 0x7fd6e7c7e2d0>
```



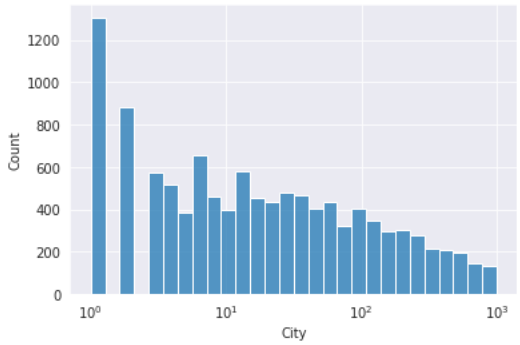

```
# Distribution of low accident cities
sns.distplot(low_accident_cities)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be rem
  warnings.warn(msg, FutureWarning)
<matplotlib.axes._subplots.AxesSubplot at 0x7fd6e7c66750>
```



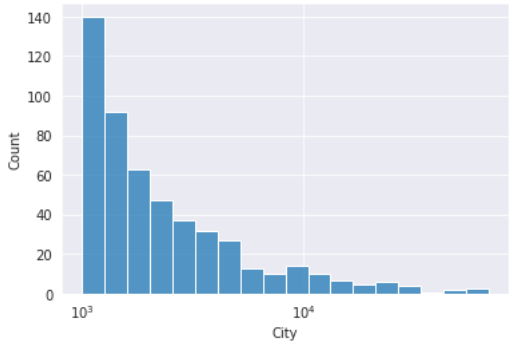

```
# Distribution of low accident cities
sns.histplot(low_accident_cities, log_scale=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd6e7a3f110>
```




```
# Distribution of high accident cities
sns.histplot(high_accident_cities, log_scale=True)
```

```
<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

```
cities_by_accident[cities_by_accident == 1]
```

```
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
```

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

```
'2019-05-21 08:29:55'
```

```
# converting date time to correct format
df.Start_Time = pd.to_datetime(df.Start_Time)
```

```
df.Start_Time[0]
```

```
Timestamp('2019-05-21 08:29:55')
```

```
# 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].hour, df.Start_Time[0].minute, df.Start_Time[0].second
```

```
(21, 5, 2019, 8, 29, 55)
```

Get the hour of the day for all the data

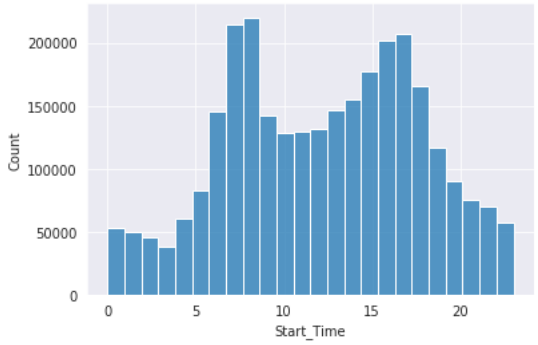
```
df.Start_Time.dt.hour
```

```
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

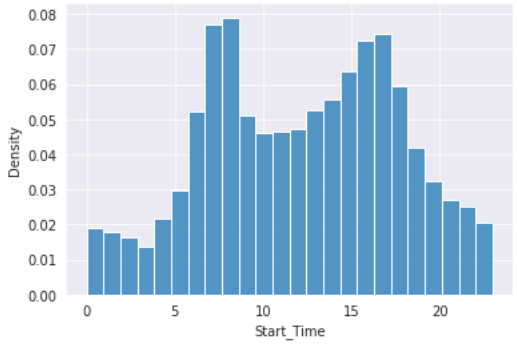
```
sns.histplot(df.Start_Time.dt.hour, bins=24)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd6e7b41090>
```



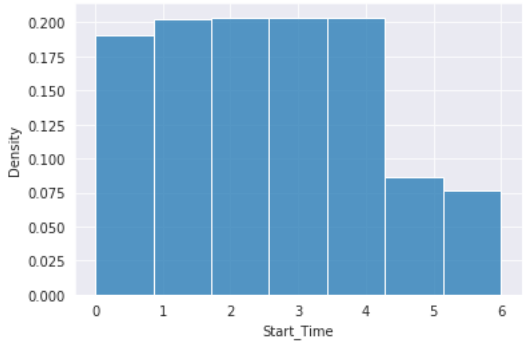

```
sns.histplot(df.Start_Time.dt.hour, bins=24, stat='density')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd6e792c210>
```



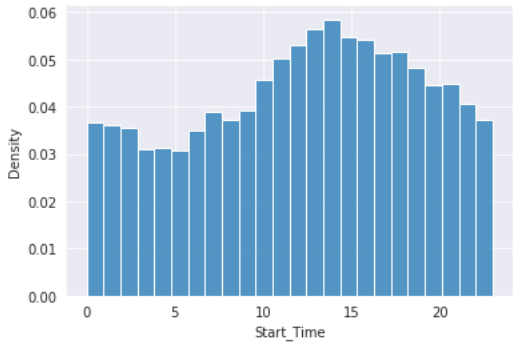

```
sns.histplot(df.Start_Time.dt.dayofweek, bins=7, stat='density')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd6e76aea50>
```



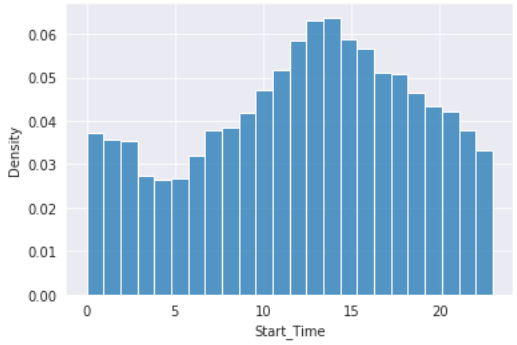
```
sundays_start_time = df.Start_Time[df.Start_Time.dt.dayofweek == 6]


sns.histplot(sundays_start_time.dt.hour, bins=24, stat='density')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd6e75ce750>
```



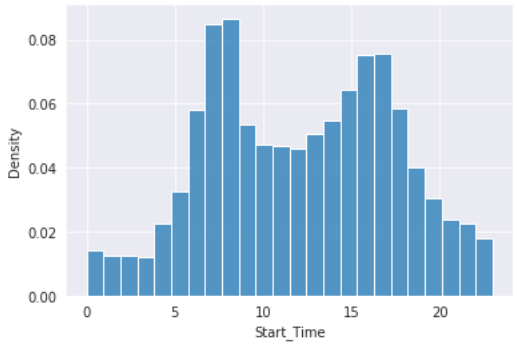

```
saturdays_start_time = df.Start_Time[df.Start_Time.dt.dayofweek == 5]
sns.histplot(saturdays_start_time.dt.hour, bins=24, stat='density')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd6e74f8190>
```



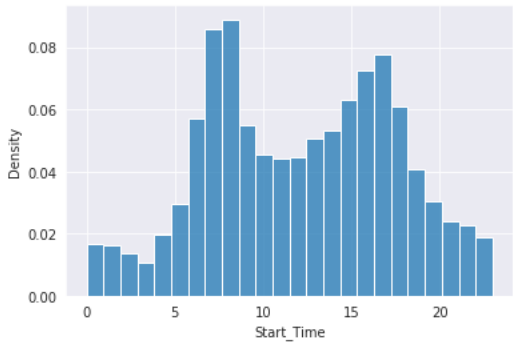

```
mondays_start_time = df.Start_Time[df.Start_Time.dt.dayofweek == 0]
sns.histplot(mondays_start_time.dt.hour, bins=24, stat='density')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd6e743a950>
```



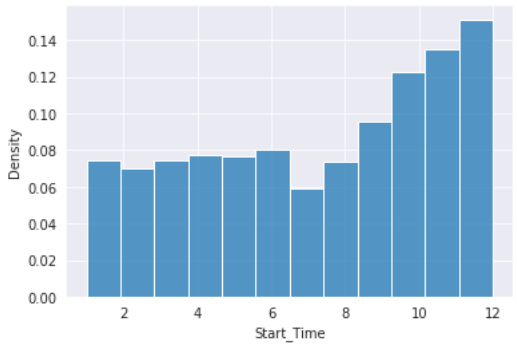

```
wednesdays_start_time = df.Start_Time[df.Start_Time.dt.dayofweek == 2]
sns.histplot(wednesdays_start_time.dt.hour, bins=24, stat='density')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd6e74ab610>
```



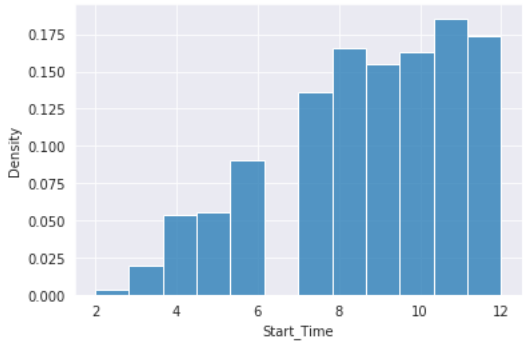

```
sns.histplot(df.Start_Time.dt.month, bins=12, stat='density')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd6e7457f50>
```



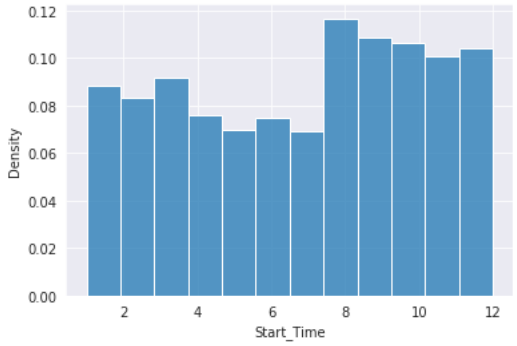

```
df_particular_year = df[df.Start_Time.dt.year == 2016]
sns.histplot(df_particular_year.Start_Time.dt.month, bins=12, stat='density')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd6e7348390>
```



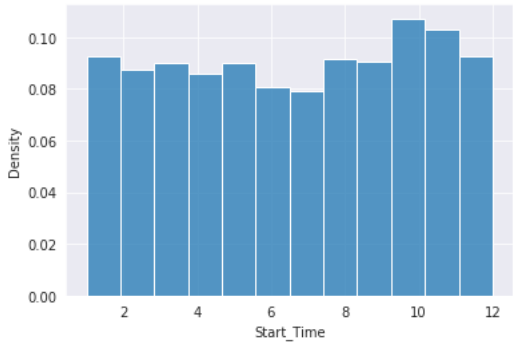

```
df_particular_year = df[df.Start_Time.dt.year == 2017]
sns.histplot(df_particular_year.Start_Time.dt.month, bins=12, stat='density')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd6e72e3510>
```



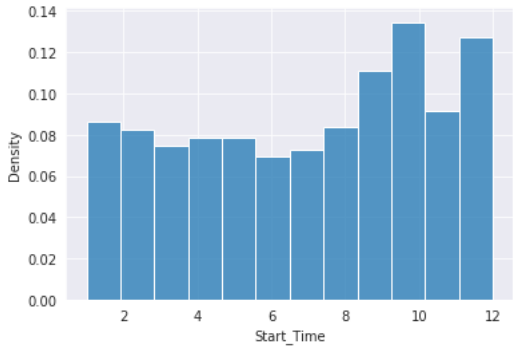

```
df_particular_year = df[df.Start_Time.dt.year == 2018]
sns.histplot(df_particular_year.Start_Time.dt.month, bins=12, stat='density')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd6e7273650>
```



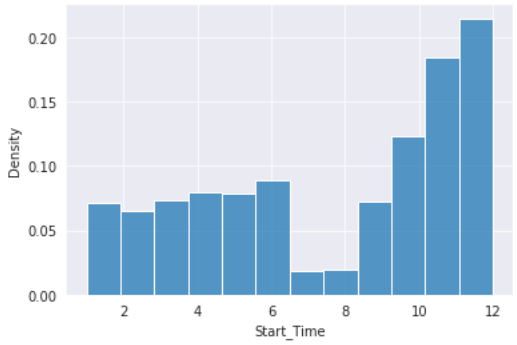

```
df_particular_year = df[df.Start_Time.dt.year == 2019]
sns.histplot(df_particular_year.Start_Time.dt.month, bins=12, stat='density')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd6e721ed10>
```




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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd6e71121d0>
```




```
df.Start_Lat
```

```
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
```

```
df.Start_Lng
```

```
0          -82.269157
1          -80.745560
2         -121.985052
3         -119.773781
4          -80.265091
```

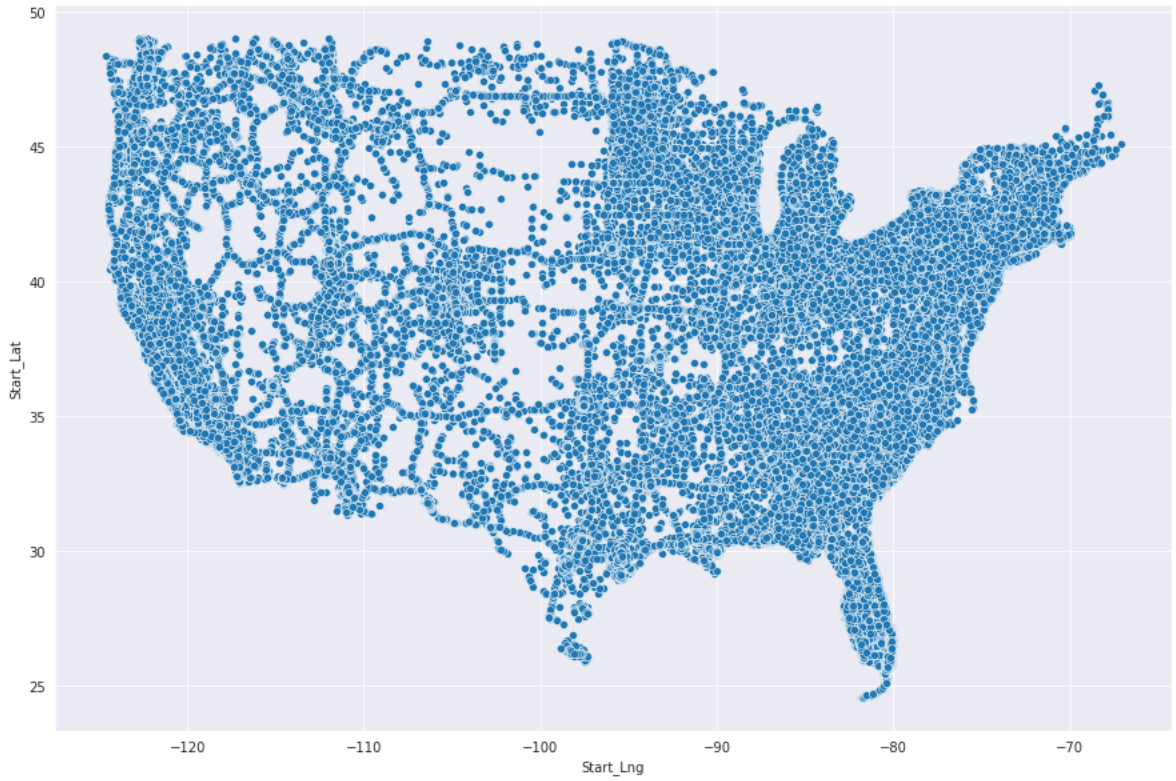
```
                ...
2906605    -95.399437
2906606   -117.342010
2906607    -80.335556
2906608   -111.952460
2906609   -104.748161
Name: Start_Lng, Length: 2906610, dtype: float64
```

```
import matplotlib.pyplot as plt
```

Plotting the latitudes and longitudes

```
plt.figure(figsize=(15,10))
sns.scatterplot(y=df.Start_Lat, x=df.Start_Lng)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd6e710cb90>
```




```
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
```

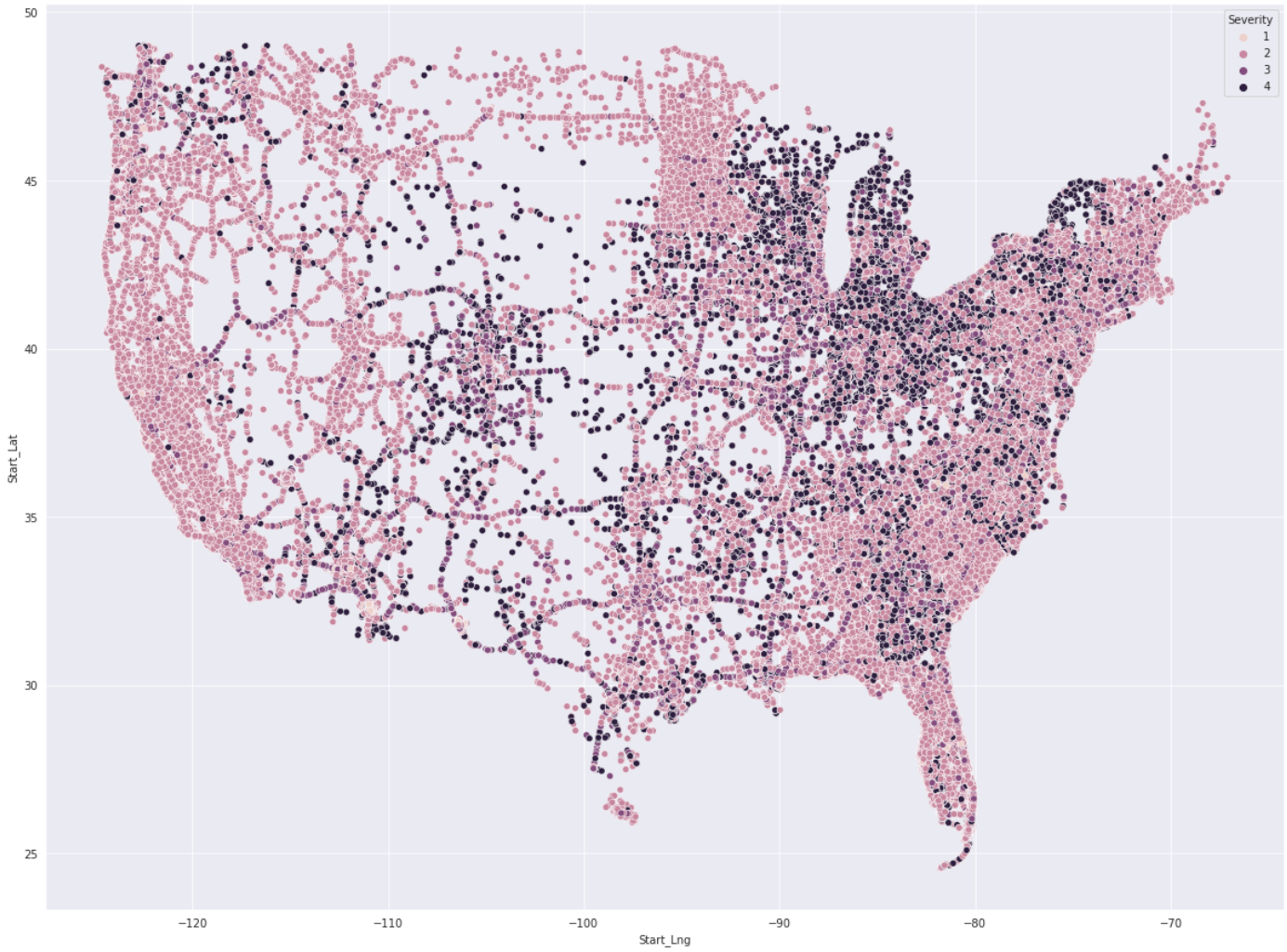
```
 22  Wind_Chill(F)          float64
 23  Humidity(%)            float64
 24  Pressure(in)           float64
 25  Visibility(mi)         float64
 26  Wind_Direction         object
 27  Wind_Speed(mph)        float64
 28  Precipitation(in)      float64
 29  Weather_Condition      object
 30  Amenity                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
```

```
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
<matplotlib.axes._subplots.AxesSubplot at 0x7fd6e76b5ed0>
/usr/local/lib/python3.7/dist-packages/google/colab/_event_manager.py:28: UserWarning: Creating legend with loc="best" can be slow with
  func(*args, **kwargs)
/usr/local/lib/python3.7/dist-packages/IPython/core/pylabtools.py:125: UserWarning: Creating legend with loc="best" can be slow with lar
```

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

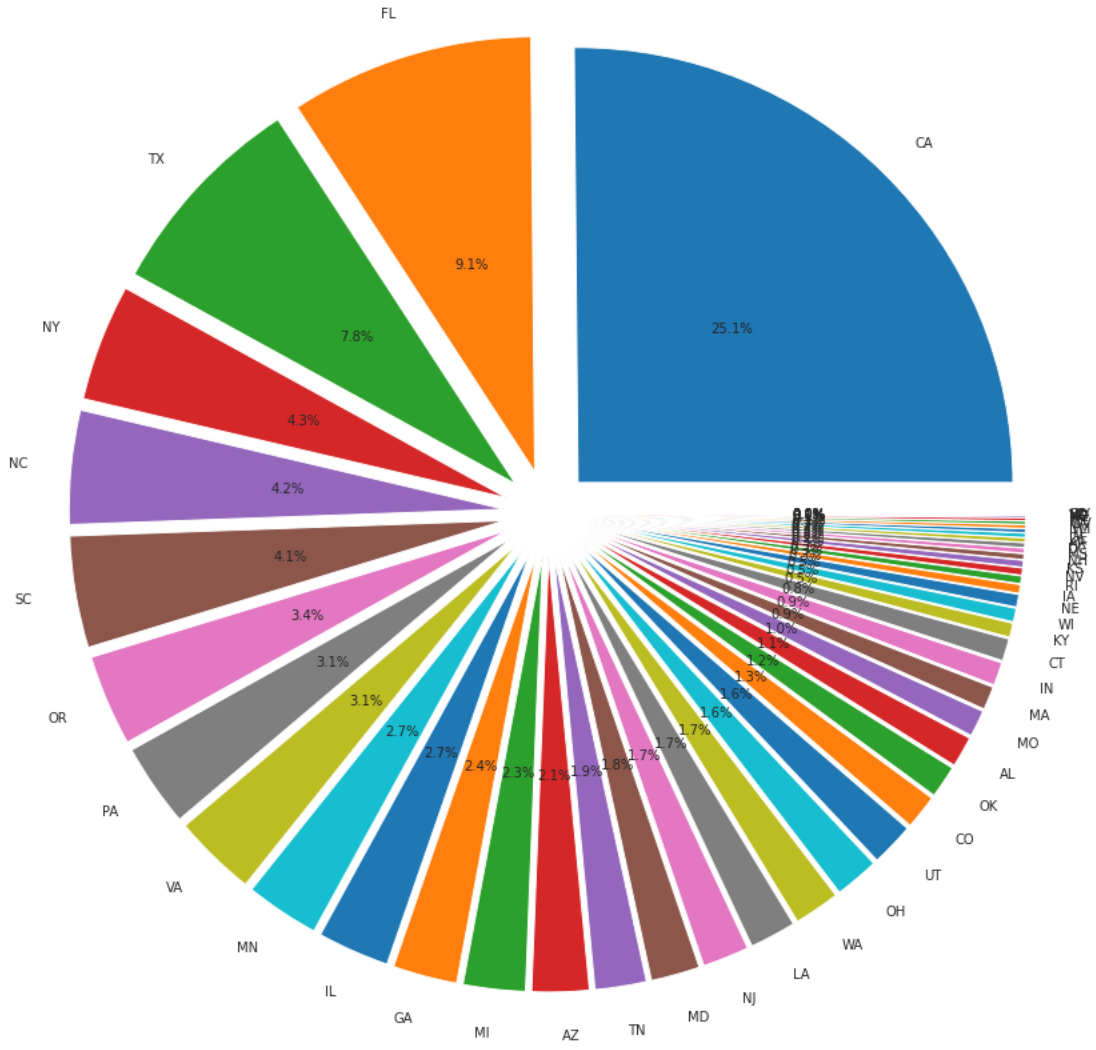
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd6e572e0d0>
```




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

```
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
```

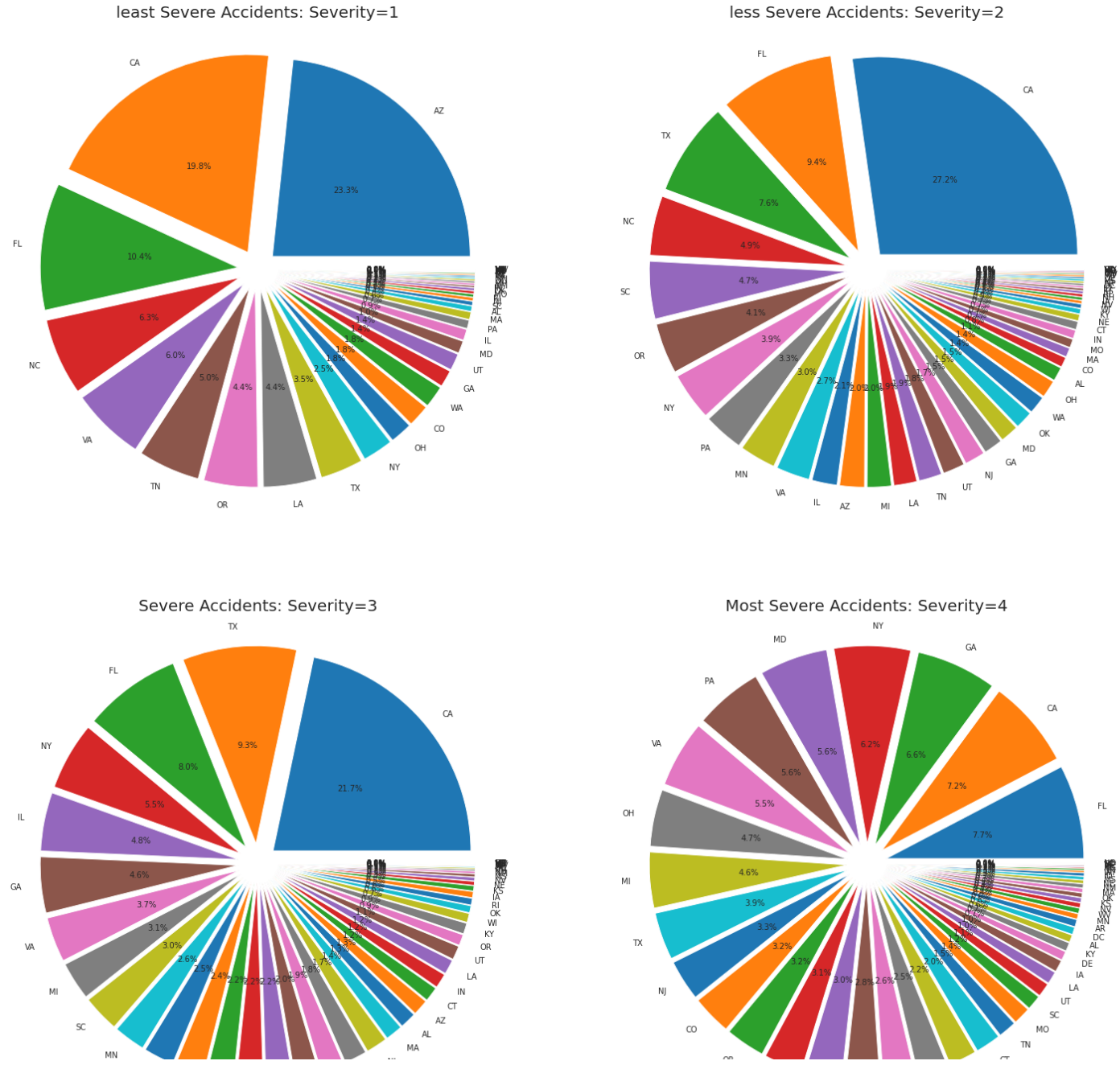
```
pie, ax = plt.subplots(figsize=[15,15])
labels = df.State.value_counts().keys()
plt.pie(x=df.State.value_counts(), autopct="%.1f%%", explode=[0.1]*len(df.State.value_counts()), labels=labels, pctdistance=0.5)
plt.show();
```




```
# 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()
```

```
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), labels=labels, pctdistance=0.5)
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), labels=labels, pctdistance=0.5)
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), labels=labels, pctdistance=0.5)
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), labels=labels, pctdistance=0.5)
plt.title("Most Severe Accidents: Severity=4", fontsize=20)
```

```
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)

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

```
[(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),
```

```
import random
```

```
df_sample = df.sample(10000)
```

```
df_sample.Start_Lat
```

```
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
```

```
df.columns
```

```
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')
```

```
df['Visibility(mi)']
```

```
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
```

```
df['Visibility(mi)'].value_counts()
```

```
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
```

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

	ID	Severity	Start_Time	End_Time	Start_Lat	Start_Lng	End_Lat	End_Lng	Distance(mi)	Description	Number	
6	A-7	4	2019-12-12 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	
40	A-41	4	2020-10-26 00:38:00	2020-10-26 02:41:25	40.428002	-79.926779	40.426058	-79.895801	1.635	Incident on I-376 EB near I-376 Road closed. T...	NaN	
146	A-147	4	2016-08-20 01:31:43	2016-08-20 07:31:43	40.645250	-75.425760	40.630360	-75.471790	2.624	Closed between PA-987/Airport Rd and Fullerton...	NaN	
167	A-168	4	2019-07-13 16:14:30	2019-07-13 16:42:44	43.849180	-84.020810	43.737277	-84.016961	7.734	Closed between Pinconning Rd/Exit 181 and Linw...	NaN	
196	A-197	4	2018-07-09 01:08:19	2018-07-09 07:08:19	32.719830	-117.117570	32.716810	-117.119700	0.243	Ramp closed to CA-15 - Road closed due to acci...	NaN	
...	...	...	...	...	...	...	...	...	...	...	...	
2906549	A-2906550	4	2018-08-26 00:25:54	2018-08-26 02:25:53	25.941730	-80.189260	25.950490	-80.181540	0.772	At I-95 (South) - Accident. Police activity on...	NaN	
2906554	A-2906555	4	2020-04-23 19:18:22	2020-04-23 19:47:31	39.144030	-84.559830	39.137680	-84.548380	0.754	Closed at Beekman St - Road closed due to acci...	NaN	
2906568	A-2906569	4	2019-11-09 03:55:18	2019-11-09 04:22:55	29.182560	-82.184640	29.191010	-82.184590	0.584	Closed at SR-40/Exit 352 - Road closed due to ...	NaN	
2906577	A-2906578	4	2017-03-07 16:01:38	2017-03-07 22:01:38	36.573645	-79.847221	36.559341	-79.826802	1.504	Closed at North Carolina/Virginia - Road close...	3030.0	M
2906591	A-2906592	4	2020-12-24 08:23:00	2020-12-24 11:25:20	41.785073	-86.139762	41.785570	-86.137596	0.117	Incident on US-12 EB near BRUSH RD Road closed...	29612.0	

114602 rows × 47 columns

```
(len(df[df['Visibility(mi)'] <=2]) / len(df) )* 100. # total percentage of accidents in which visibility was less than 2 miles
```

```
4.375578422973843
```

```
(len(df[(df['Visibility(mi)'] <=2) & (df['Severity'] ==4)]) / len(df) ) * 100. # total percentage of accidents in which visibility was less
```

```
0.21203395020315763
```

```
weather = df.Weather_Condition.value_counts()
```

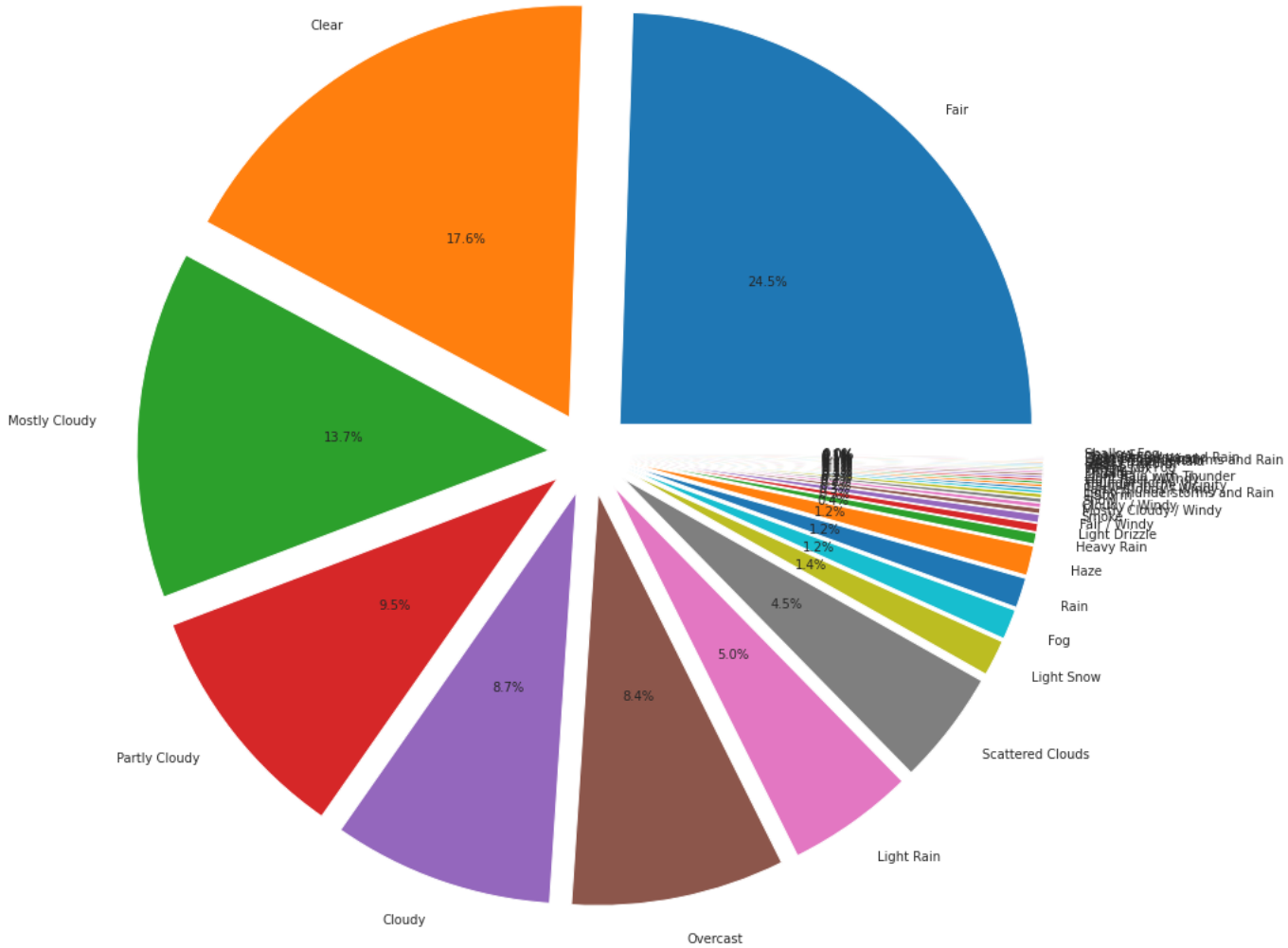
```
weather[weather > 1000]  # Kind of weather when no. of accidents were greater than 1000
```

```
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
```

```
import matplotlib.pyplot as plt
```

```
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 > 1000]), labels=labels, pctdistance=0.5)
plt.show();
```




```
df['Temperature(F)']
```

```
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
```

```
df['Temperature(F)'].value_counts()
```

```
 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
```

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

```
temperature.index
```

```
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)
```

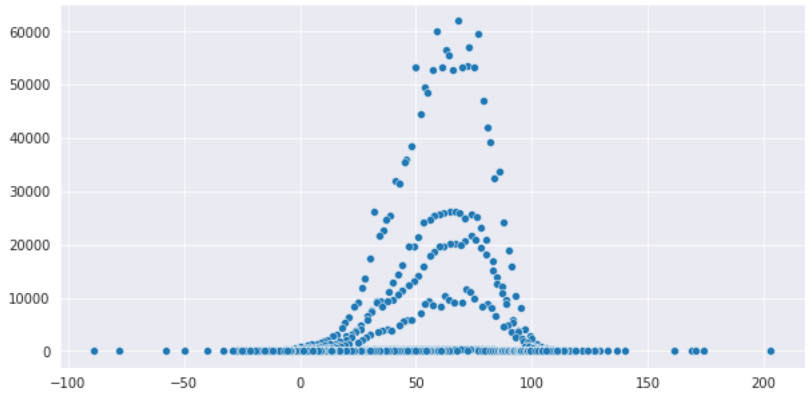
```
temperature.values
```

```
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.
```

```
import seaborn as sns
```

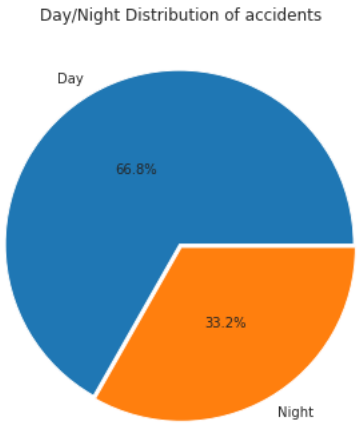
```
plt.figure(figsize=(10,5))
sns.scatterplot(x=temperature.index, y=temperature.values)
plt.show();
```




```
df.Sunrise_Sunset.value_counts()
```

```
Day      1941068
Night     965432
Name: Sunrise_Sunset, dtype: int64
```

```
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.Sunrise_Sunset.value_counts()), labels=labels, pctdistan
plt.title("Day/Night Distribution of accidents")
plt.show();
```




```
df.columns
```

```
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')
```

```
amenity = df.Amenity.groupby(df.Severity).value_counts()
amenity
```

```
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
```

```
amenity.index
```

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

```
no_exit = df.No_Exit.groupby(df.Severity).value_counts()
no_exit
```

```
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
```

```
railway = df.Railway.groupby(df.Severity).value_counts()
railway
```

```
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
```

```
traffic_calming = df.Traffic_Calming.groupby(df.Severity).value_counts()
traffic_calming
```

```
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
```

```
stop = df.Stop.groupby(df.Severity).value_counts()
stop
```

```
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
```

```
traffic_signal = df.Traffic_Signal.groupby(df.Severity).value_counts()
traffic_signal
```

```
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
```

```
give_way = df.Give_Way.groupby(df.Severity).value_counts()
give_way
```

```
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
```

```
bump = df.Bump.groupby(df.Severity).value_counts()
bump
```

```
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
```

```
crossing = df.Crossing.groupby(df.Severity).value_counts()
crossing
```

```
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
```

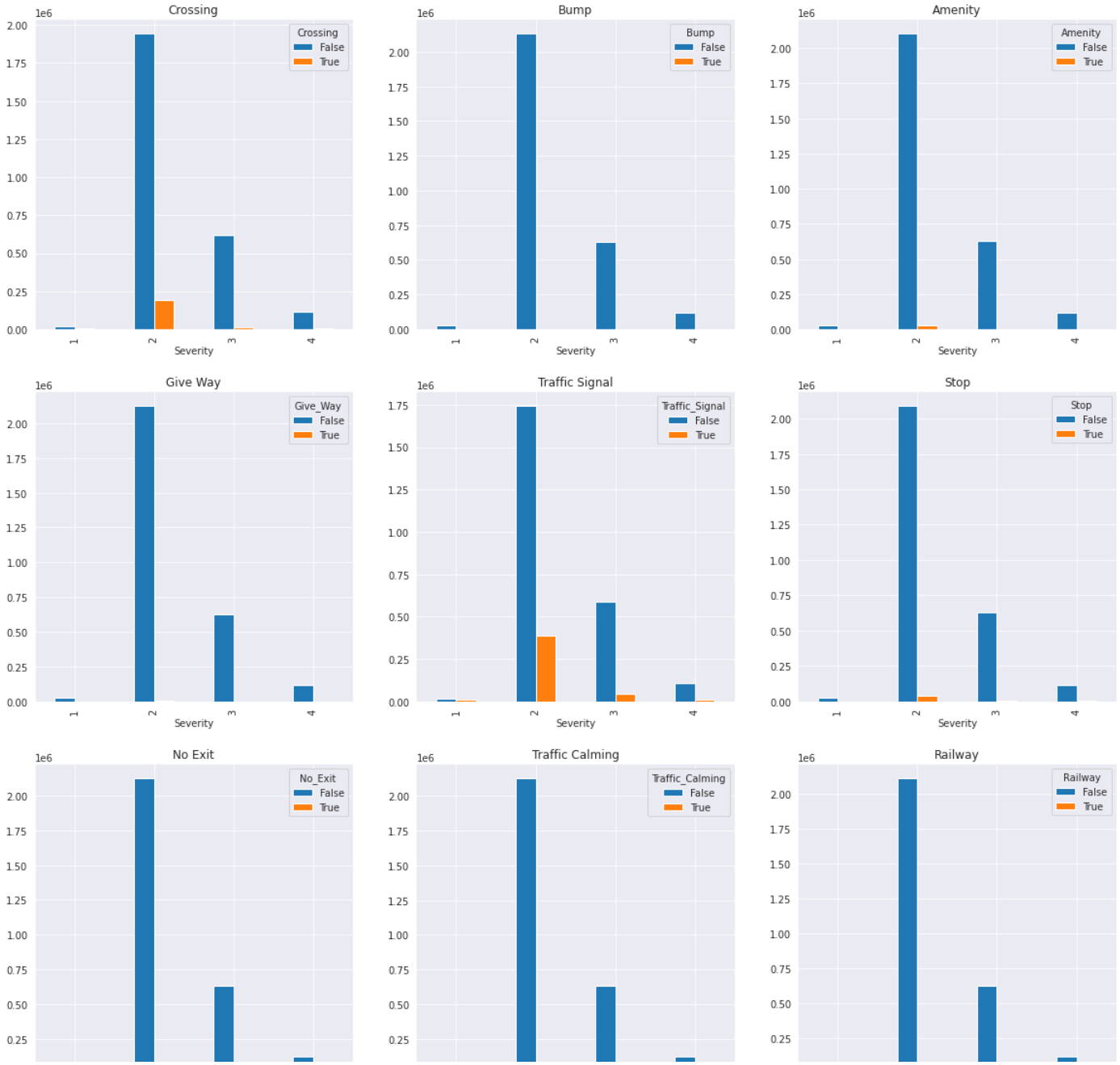
```
df.Turning_Loop.value_counts()
```

```
False    2906610
Name: Turning_Loop, dtype: int64
```

Plotting all the values

```
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")
```

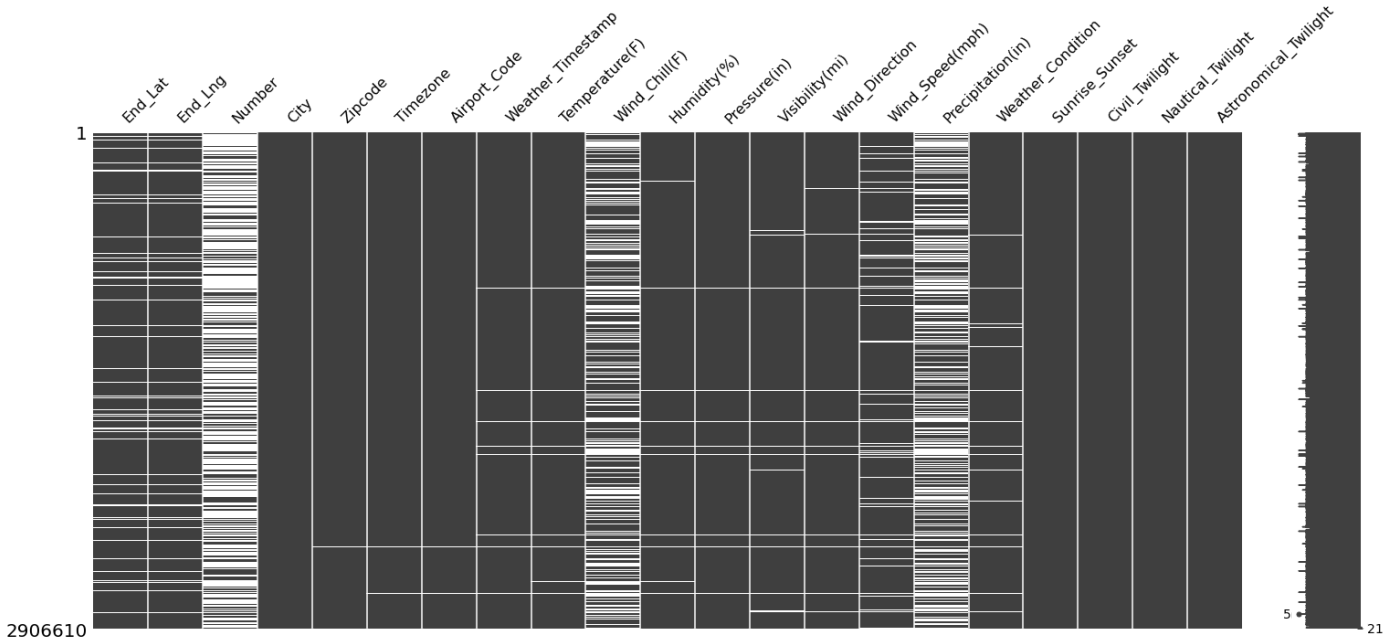
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd6ca355590>
```




```
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)',
```

```
mn.matrix(data[null_cols]);
```




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

```
new_data_b = new_data_a.dropna(subset = ['Visibility(mi)','Weather_Condition','Humidity(%)','Temperature(F)','Wind_Direction','Pressure(in)'
```

```
new_data_b.isnull().sum()
```

```
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
```

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

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

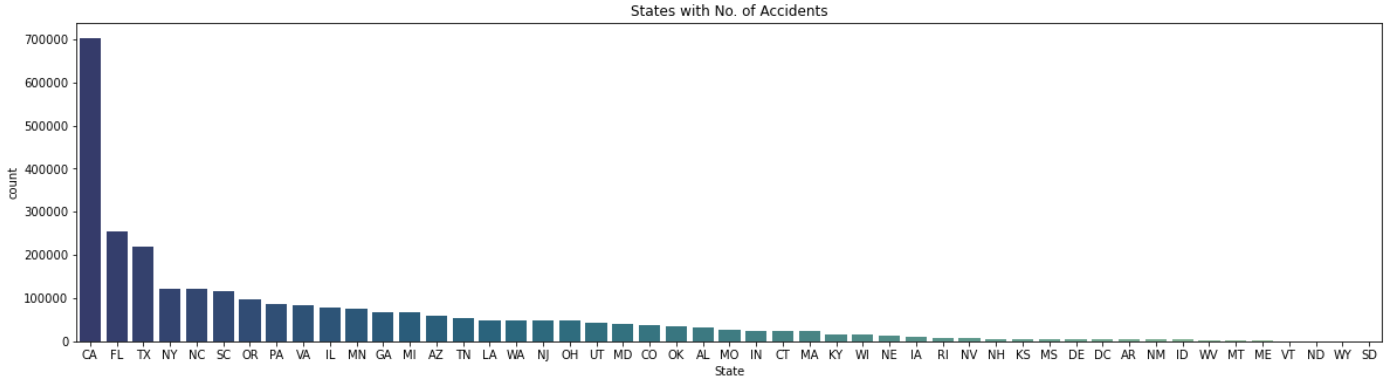
```
Severity                        0
Start_Time                      0
End_Time                        0
Start_Lat                       0
Start_Lng                       0
Distance(mi)                    0
Description                     0
Street                          0
Side                            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
```

```
state_counts = final_data["State"].value_counts()
fig = go.Figure(data=go.Choropleth(locations=state_counts.index, z=state_counts.values.astype(float), locationmode="USA-states", colorscale=
fig.update_layout(title_text="Number of Accidents for each State", geo_scope="usa")
fig.show()
```

```
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
```

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




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