Exploratory Data Analysis on US accidents

Steps 1) Selecting large real world dataset 2)Data preparation & cleaning using pandas 3)Exploratory analysis & visualizarion using matplotlib & seaborn 4)Ask and answer questions 5)Summarize and write conclusions 6)Document, Publish & Present the notebook online

Data Preparation & Cleaning:

load file using pandas

In [2]:
```
import pandas as pd

data = pd.read_csv("C:\\Users\\Gudise Dharani\\Downloads\\archive (1)\\US_Accidents_Dec21_updated.csv")
len(data.columns)
```

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

look at some info about data & columns

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2845342 entries, 0 to 2845341
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: 773.4+ MB
```

In [4]:
```
data.describe()
```

Out[4]:

	Severity	Start_Lat	Start_Lng	End_Lat	End_Lng	Distance(mi)	Number	Temperature(F)	Wind_Chill(F)	Humidity(%)	Pressure(in)	Visibility(mi)	Wind_Speed(mph)	Precipitation(in)
count	2.845342e+06	2.845342e+06	2.845342e+06	2.845342e+06	2.845342e+06	2.845342e+06	1.101431e+06	2.776068e+06	2.375699e+06	2.772250e+06	2.786142e+06	2.774796e+06	2.687398e+06	2.295884e+06
mean	2.137572e+00	3.624520e+01	-9.711463e+01	3.624532e+01	-9.711439e+01	7.026779e-01	8.089408e+03	6.179356e+01	5.965823e+01	6.436545e+01	2.947234e+01	9.099391e+00	7.395044e+00	7.016940e-03
std	4.787216e-01	5.363797e+00	1.831782e+01	5.363873e+00	1.831763e+01	1.560361e+00	1.836009e+04	1.862263e+01	2.116097e+01	2.287457e+01	1.045286e+00	2.717546e+00	5.527454e+00	9.348831e-02
min	1.000000e+00	2.456603e+01	-1.245481e+02	2.456601e+01	-1.245457e+02	0.000000e+00	0.000000e+00	-8.900000e+01	-8.900000e+01	1.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
25%	2.000000e+00	3.344517e+01	-1.180331e+02	3.344628e+01	-1.180333e+02	5.200000e-02	1.270000e+03	5.000000e+01	4.600000e+01	4.800000e+01	2.931000e+01	1.000000e+01	3.500000e+00	0.000000e+00
50%	2.000000e+00	3.609861e+01	-9.241808e+01	3.609799e+01	-9.241772e+01	2.440000e-01	4.007000e+03	6.400000e+01	6.300000e+01	6.700000e+01	2.982000e+01	1.000000e+01	7.000000e+00	0.000000e+00
75%	2.000000e+00	4.016024e+01	-8.037243e+01	4.016105e+01	-8.037338e+01	7.640000e-01	9.567000e+03	7.600000e+01	7.600000e+01	8.300000e+01	3.001000e+01	1.000000e+01	1.000000e+01	0.000000e+00
max	4.000000e+00	4.900058e+01	-6.711317e+01	4.907500e+01	-6.710924e+01	1.551860e+02	9.999997e+06	1.960000e+02	1.960000e+02	1.000000e+02	5.890000e+01	1.400000e+02	1.087000e+03	2.400000e+01

In [5]:
```
numeric = ['float64','int64']
new = data.select_dtypes(include=numeric)
len(new.columns)
```

Out[5]:
```
14
```

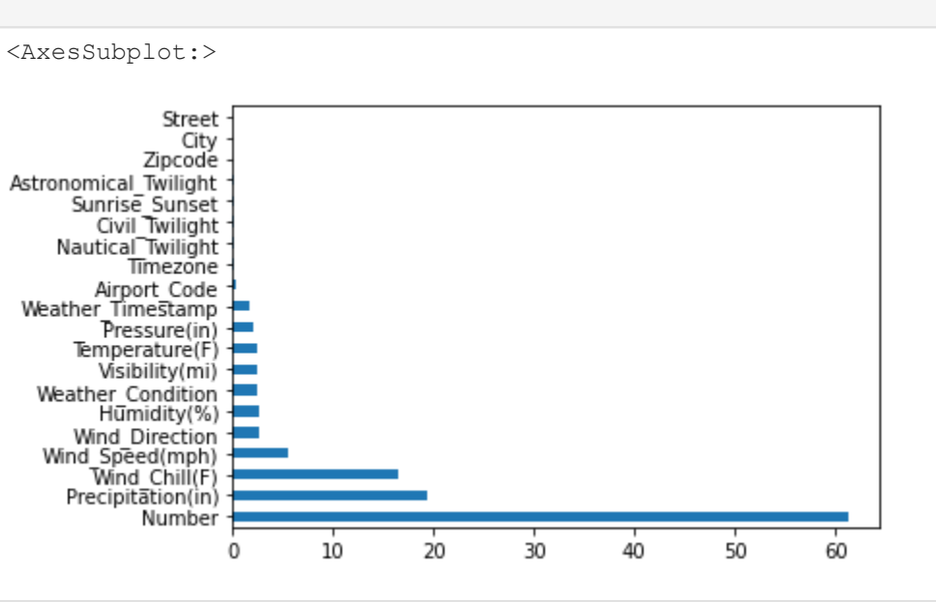
Fix any missing values or incorrect values

In [6]:
```
missing_values = (data.isna().sum().sort_values(ascending=False)/len(data))* 100
missing_values
```

Out[6]:
```
Number                   61.290031
Precipitation(in)        19.310789
Wind_Chill(F)            16.505678
Wind_Speed(mph)           5.550967
Wind_Direction            2.592834
Humidity(%)               2.568830
Weather_Condition         2.482514
Visibility(mi)            2.479350
Temperature(F)            2.434646
Pressure(in)              2.080593
Weather_Timestamp         1.783125
Airport_Code              0.335601
Timezone                  0.128596
Nautical_Twilight         0.100761
Civil_Twilight            0.100761
Sunrise_Sunset            0.100761
Astronomical_Twilight     0.100761
Zipcode                   0.046356
City                      0.004815
Street                    0.000070
Country                   0.000000
Junction                  0.000000
Start_Time                0.000000
End_Time                  0.000000
Start_Lat                 0.000000
Turning_Loop              0.000000
Traffic_Signal            0.000000
Traffic_Calming           0.000000
Stop                      0.000000
Station                   0.000000
Roundabout                0.000000
Railway                   0.000000
No_Exit                   0.000000
Crossing                  0.000000
Give_Way                  0.000000
Bump                      0.000000
Amenity                   0.000000
Start_Lng                 0.000000
End_Lat                   0.000000
End_Lng                   0.000000
Distance(mi)              0.000000
Description               0.000000
Severity                  0.000000
Side                      0.000000
County                    0.000000
State                     0.000000
ID                        0.000000
dtype: float64
```

In [7]:
```
missing_values[missing_values != 0].plot(kind='barh')
```

Out[7]:
```
<AxesSubplot:>
```



In [8]:
```
missing_values[missing_values != 0]
```

Out[8]:
```
Number                   61.290031
Precipitation(in)        19.310789
Wind_Chill(F)            16.505678
Wind_Speed(mph)           5.550967
Wind_Direction            2.592834
Humidity(%)               2.568830
Weather_Condition         2.482514
Visibility(mi)            2.479350
Temperature(F)            2.434646
Pressure(in)              2.080593
Weather_Timestamp         1.783125
Airport_Code              0.335601
Timezone                  0.128596
Nautical_Twilight         0.100761
Civil_Twilight            0.100761
Sunrise_Sunset            0.100761
Astronomical_Twilight     0.100761
Zipcode                   0.046356
City                      0.004815
Street                    0.000070
dtype: float64
```

Exploratory Analysis & Visualization:

Pick up columns for analysis:

1)City 2)Start time 3)Start latitude 4)Start longitude 5)Temperature

City:

In [9]:
```
len(data['City'].unique())
```

Out[9]:
```
11682
```

In [10]:
```
cities_by_accidents = data['City'].value_counts()
cities_by_accidents
```

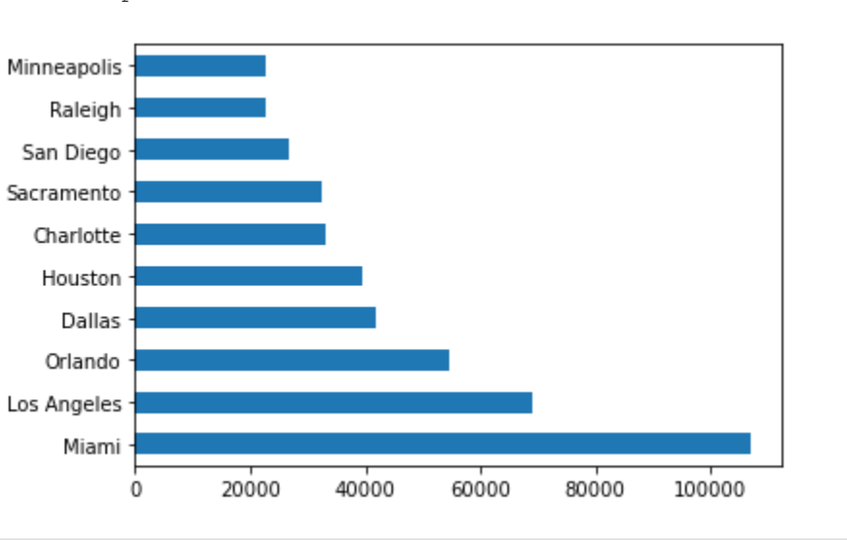
Out[10]:
```
Miami                            106966
Los Angeles                       68956
Orlando                           54691
Dallas                            41979
Houston                           39448
                                  ...
Ridgedale                             1
Sekiu                                 1
Wooldridge                            1
Bullock                               1
American Fork-Pleasant Grove          1
Name: City, Length: 11681, dtype: int64
```

In [11]:
```
cities_by_accidents[:10]
```

Out[11]:
```
Miami          106966
Los Angeles     68956
Orlando         54691
Dallas          41979
Houston         39448
Charlotte       33152
Sacramento      32559
San Diego       26627
Raleigh         22840
Minneapolis     22768
Name: City, dtype: int64
```

In [12]:
```
cities_by_accidents[:10].plot(kind='barh')
```

Out[12]:
```
<AxesSubplot:>
```

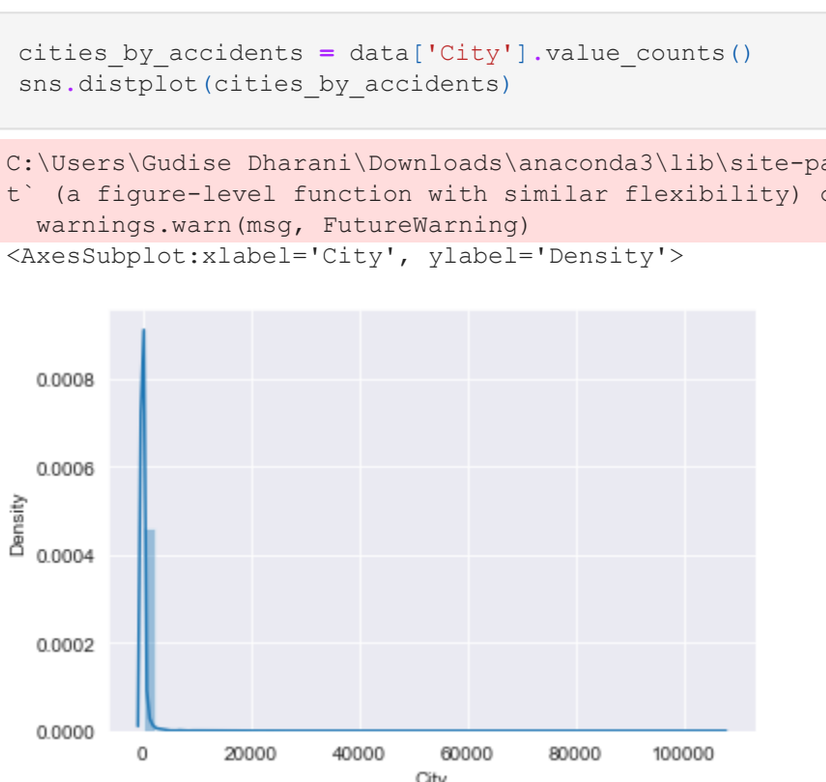


In [13]:
```
import seaborn as sns
sns.set_style('darkgrid')
```

In [14]:
```
cities_by_accidents = data['City'].value_counts()
sns.distplot(cities_by_accidents)
```

```
C:\Users\Gudise Dharani\Downloads\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displo
t` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[14]:
```
<AxesSubplot:xlabel='City', ylabel='Density'>
```



In [15]:
```
high_accident_cities = cities_by_accidents[cities_by_accidents >= 1000]
low_accident_cities = cities_by_accidents[cities_by_accidents < 1000]
```

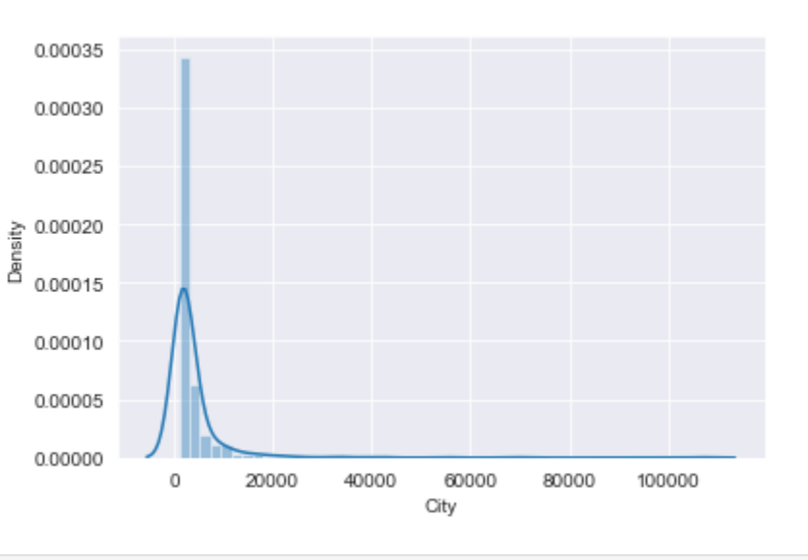
In [16]:
```
(len(high_accident_cities)/len(cities_by_accidents))*100
```

Out[16]:
```
4.246211796935194
```

In [17]:
```
sns.distplot(high_accident_cities)
```

```
C:\Users\Gudise Dharani\Downloads\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displo
t` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

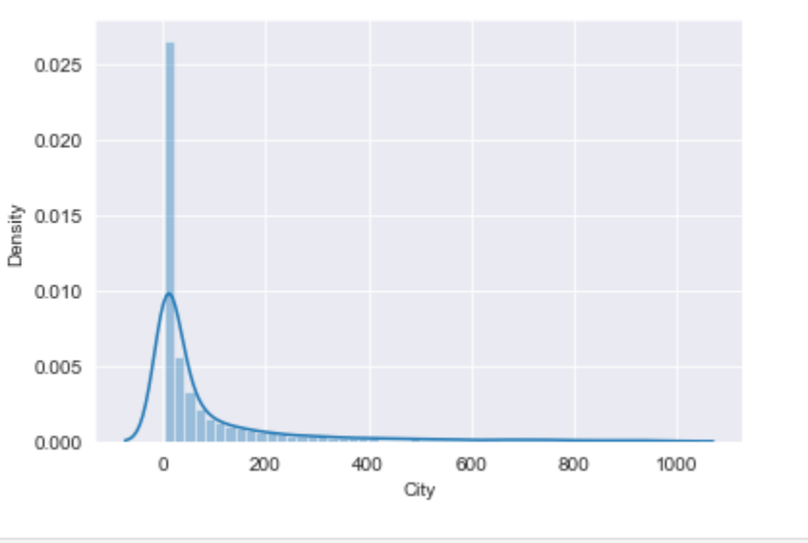
Out[17]:
```
<AxesSubplot:xlabel='City', ylabel='Density'>
```



In [18]:
```
sns.distplot(low_accident_cities)
```

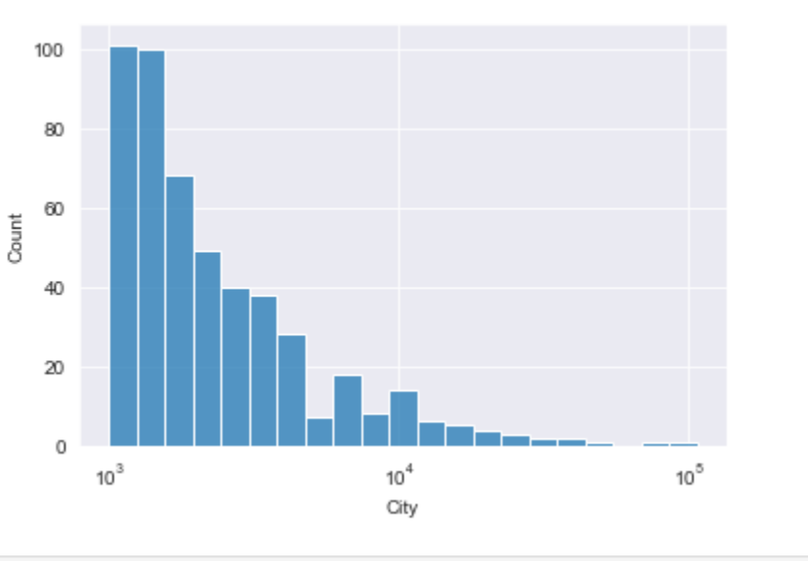
```
C:\Users\Gudise Dharani\Downloads\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displo
t` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[18]:
```
<AxesSubplot:xlabel='City', ylabel='Density'>
```



In [19]:
```
sns.histplot(high_accident_cities, log_scale=True)
```

Out[19]:
```
<AxesSubplot:xlabel='City', ylabel='Count'>
```



In [20]:
```
(data['City'].value_counts() == 1).value_counts()
```

Out[20]:
```
False    10571
True      1110
Name: City, dtype: int64
```

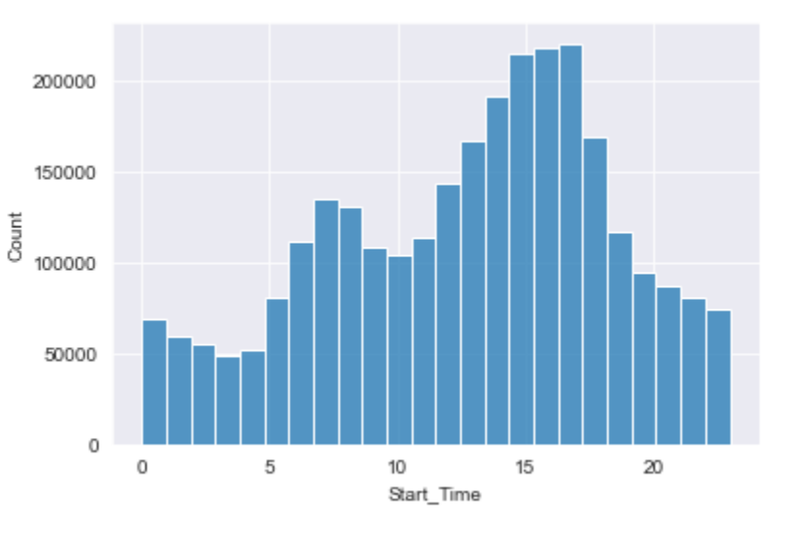
Start time:

In [21]:
```
data['Start_Time'] = pd.to_datetime(data['Start_Time'])
data['Start_Time'][:10]
```

Out[21]:
```
0   2016-02-08 00:37:08
1   2016-02-08 05:56:20
2   2016-02-08 06:15:39
3   2016-02-08 06:51:45
4   2016-02-08 07:53:43
5   2016-02-08 08:16:57
6   2016-02-08 08:15:41
7   2016-02-08 11:51:46
8   2016-02-08 14:19:57
9   2016-02-08 15:16:43
Name: Start_Time, dtype: datetime64[ns]
```

In [22]:
```
sns.histplot(data['Start_Time'].dt.hour, bins=24)
```

Out[22]:
```
<AxesSubplot:xlabel='Start_Time', ylabel='Count'>
```



Start Latitude & Start Longitude:

In [23]:
```
data['Start_Lat']
```

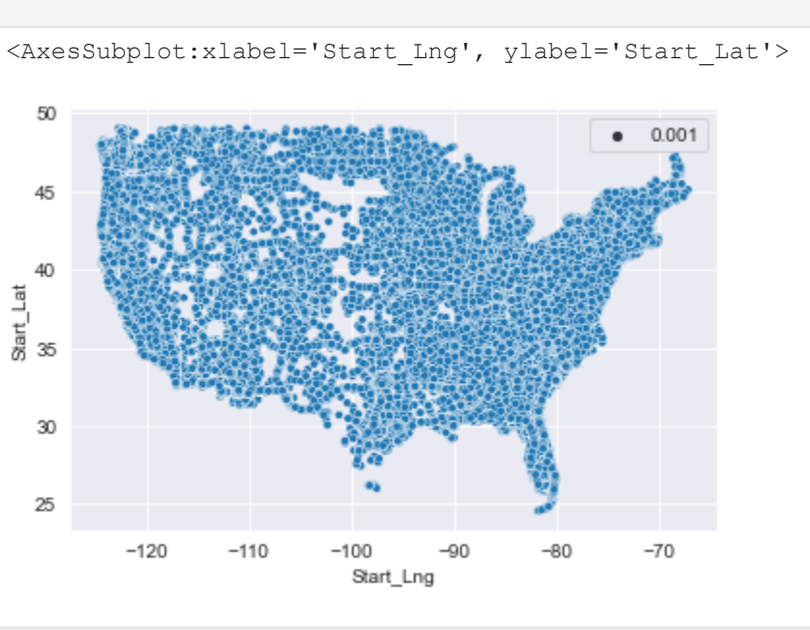
Out[23]:
```
0          40.108910
1          39.865420
2          39.102660
3          41.062130
4          39.172393
             ...
2845337    34.002480
2845338    32.766960
2845339    33.775450
2845340    33.992460
2845341    34.133930
Name: Start_Lat, Length: 2845342, dtype: float64
```

In [24]:
```
data['Start_Lng']
```

Out[24]:
```
0          -83.092860
1          -84.062800
2          -84.524680
3          -81.537840
4          -84.492792
              ...
2845337   -117.379360
2845338   -117.148060
2845339   -117.847790
2845340   -118.403020
2845341   -117.230920
Name: Start_Lng, Length: 2845342, dtype: float64
```

In [25]:
```
sns.scatterplot(x=data['Start_Lng'],y=data['Start_Lat'], size=0.001)
```

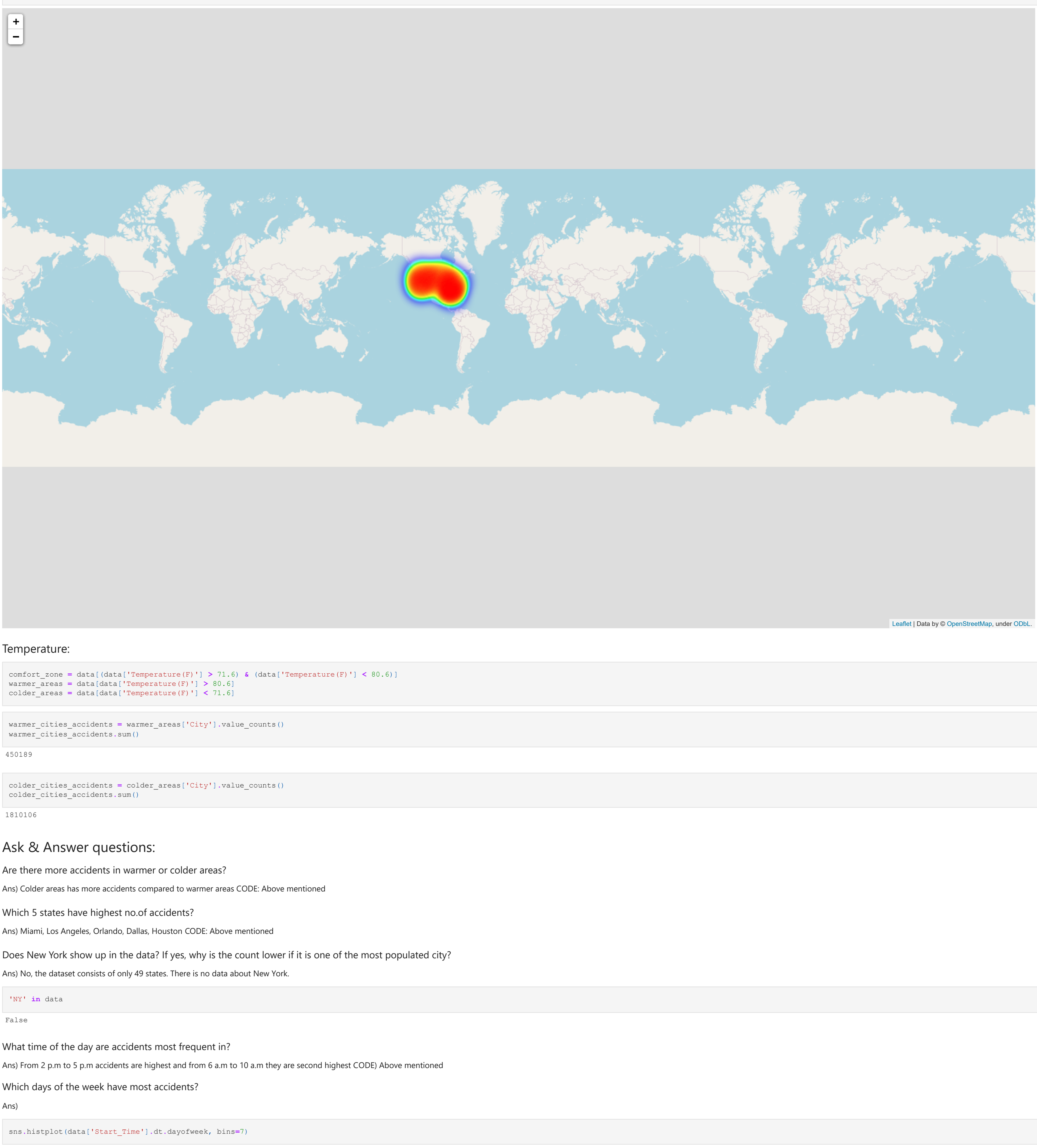
Out[25]:
```
<AxesSubplot:xlabel='Start_Lng', ylabel='Start_Lat'>
```



In [26]:
```
import folium
from folium.plugins import HeatMap
```

In [27]:
```
map = folium.Map()
HeatMap(list(zip(list(data['Start_Lat']), list(data['Start_Lng'])))).add_to(map)
map
```

Out[27]:



Temperature:

In [28]:
```
comfort_zone = data[(data['Temperature(F)'] > 71.6) & (data['Temperature(F)'] < 80.6)]
warmer_areas = data[data['Temperature(F)'] > 80.6]
colder_areas = data[data['Temperature(F)'] < 71.6]
```

In [29]:
```
warmer_cities_accidents = warmer_areas['City'].value_counts()
warmer_cities_accidents.sum()
```

Out[29]:
```
450189
```

In [30]:
```
colder_cities_accidents = colder_areas['City'].value_counts()
colder_cities_accidents.sum()
```

Out[30]:
```
1810106
```

Ask & Answer questions:

Are there more accidents in warmer or colder areas?

Ans) Colder areas has more accidents compared to warmer areas CODE: Above mentioned

Which 5 states have highest no.of accidents?

Ans) Miami, Los Angeles, Orlando, Dallas, Houston CODE: Above mentioned

Does New York show up in the data? If yes, why is the count lower if it is one of the most populated city?

Ans) No, the dataset consists of only 49 states. There is no data about New York.

In [31]:
```
'NY' in data
```

Out[31]:
```
False
```

What time of the day are accidents most frequent in?

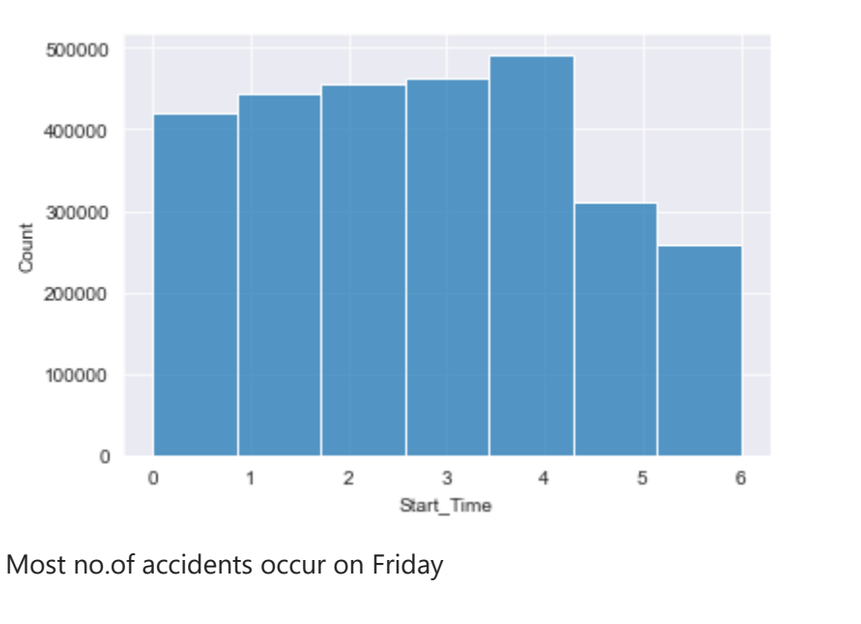
Ans) From 2 p.m to 5 p.m accidents are highest and from 6 a.m to 10 a.m they are second highest CODE) Above mentioned

Which days of the week have most accidents?

Ans)

In [32]:
```
sns.histplot(data['Start_Time'].dt.dayofweek, bins=7)
```

Out[32]:
```
<AxesSubplot:xlabel='Start_Time', ylabel='Count'>
```



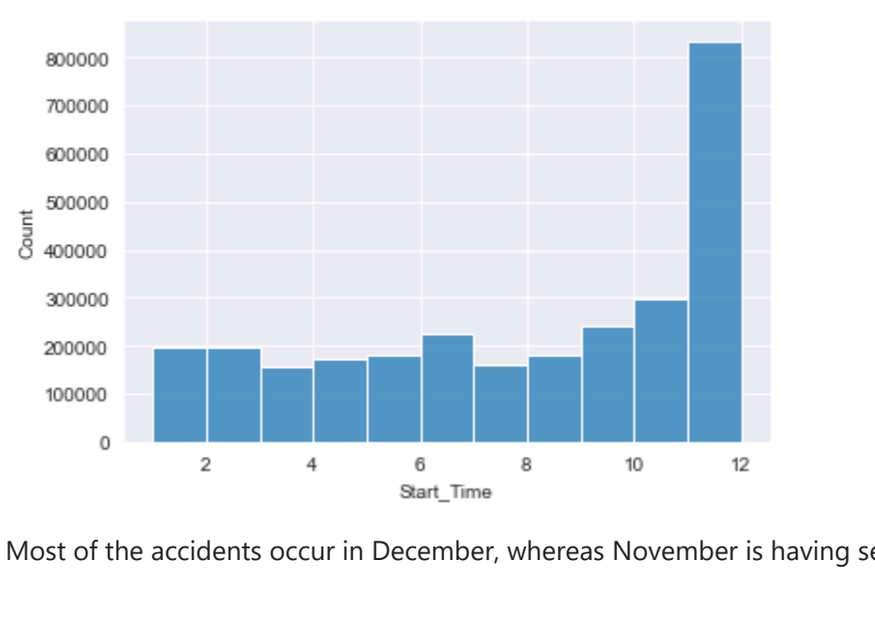
Most no.of accidents occur on Friday

Which months have the most accidents?

Ans)

In [33]:
```
sns.histplot(data['Start_Time'].dt.month, bins=11)
```

Out[33]:
```
<AxesSubplot:xlabel='Start_Time', ylabel='Count'>
```



Most of the accidents occur in December, whereas November is having second highest no.of accidents.But there is no data regarding January in the dataset.

Summary and Conclusion:

Insights:

- No data from New York
- Less than 5% of cities have more than 1000 yearly accidents
- The no.of accidents per city decreases exponentially
- Over 1110 cities have reported just one accident

In [ ]: