

# US\_Accidents\_Exploratory\_Data\_Analysis

## Steps to follow

- Select a large real\_world (more than 2.5M Records) dataset from kaggle
- Perform data preparation & cleaning using Pandas & Numpy
- Perform exploratory analysis & visualization using Matplotlib & Seaborn
- Ask & answer questions about the data in a jupyter notebook
- Summarize your inferences & write a conclusion

## Questions to ask from this dataset

- 1) Which 5 states have the highest number of accidents?
- 2) Top 100 cities in number of accidents
- 3) What time of the day are accidents most frequent in ?
- 4) Which days of the week have the most accidents ?
- 5) Which months have the most accidents?
- 6) What is the trend of accidents year over year (decreasing/increasing) ?

## Data Preparation and Cleaning

- Load the file using Pandas
- Look at some information about the data & the columns
- Fix any missing or incorrect values

## Import required libraries

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

```
In [4]: 1 df = pd.read_csv(r"G:\Udemy\DATA SCIENCE ineuron\EDA\US_Accidents_Dec21_updated.csv")
```

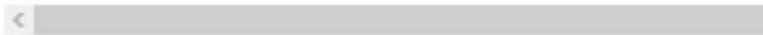
```
In [3]: 1 df.head()
```

```
In [3]: 1 df.head()
```

Out[3]:

	ID	Severity	Start_Time	End_Time	Start_Lat	Start_Lng	End_Lat	End_Lng	Distance(mi)	Description	...	Roundabc
0	A-1	3	2016-02-08 00:37:08	2016-02-08 06:37:08	40.108910	-83.092860	40.112060	-83.031870	3.230	Between Sawmill Rd/Exit 20 and OH-315/Olentang...	...	Fa
1	A-2	2	2016-02-08 05:56:20	2016-02-08 11:56:20	39.865420	-84.062800	39.865010	-84.048730	0.747	At OH-4/OH-235/Exit 41 - Accident.	...	Fa
2	A-3	2	2016-02-08 06:15:39	2016-02-08 12:15:39	39.102660	-84.524680	39.102090	-84.523960	0.055	At I-71/US-50/Exit 1 - Accident.	...	Fa
3	A-4	2	2016-02-08 06:51:45	2016-02-08 12:51:45	41.062130	-81.537840	41.062170	-81.535470	0.123	At Dart Ave/Exit 21 - Accident.	...	Fa
4	A-5	3	2016-02-08 07:53:43	2016-02-08 13:53:43	39.172393	-84.492792	39.170476	-84.501798	0.500	At Mitchell Ave/Exit 6 - Accident.	...	Fa

5 rows × 47 columns



```
In [6]: 1 df.shape
```

Out[6]: (2845342, 47)

```
In [4]: 1 len(df.columns)
```

Out[4]: 47

```
In [7]: 1 df.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   Description      object 
 9   Lat_Deg          float64
 10  Long_Deg         float64
 11  Lat_Min          float64
 12  Long_Min         float64
 13  Lat_Sec          float64
 14  Long_Sec         float64
 15  Lat_Hem          object 
 16  Long_Hem         object 
 17  Lat_Direction    object 
 18  Long_Direction   object 
 19  Lat_Deg_Direction float64
 20  Long_Deg_Direction float64
 21  Lat_Min_Direction float64
 22  Long_Min_Direction float64
 23  Lat_Sec_Direction float64
 24  Long_Sec_Direction float64
 25  Lat_Hem_Direction object 
 26  Long_Hem_Direction object 
 27  Lat_Direction_Deg float64
 28  Long_Direction_Deg float64
 29  Lat_Direction_Min float64
 30  Long_Direction_Min float64
 31  Lat_Direction_Sec float64
 32  Long_Direction_Sec float64
 33  Lat_Direction_Hem object 
 34  Long_Direction_Hem object 
 35  Lat_Direction_Deg_Direction float64
 36  Long_Direction_Deg_Direction float64
 37  Lat_Direction_Min_Direction float64
 38  Long_Direction_Min_Direction float64
 39  Lat_Direction_Sec_Direction float64
 40  Long_Direction_Sec_Direction float64
 41  Lat_Direction_Hem_Direction object 
 42  Long_Direction_Hem_Direction object 
 43  Lat_Direction_Deg_Direction_Deg float64
 44  Long_Direction_Deg_Direction_Deg float64
 45  Lat_Direction_Min_Direction_Deg float64
 46  Long_Direction_Min_Direction_Deg float64
 47  Lat_Direction_Sec_Direction_Deg float64
 48  Long_Direction_Sec_Direction_Deg float64
 49  Lat_Direction_Hem_Direction_Deg_Deg float64
 50  Long_Direction_Hem_Direction_Deg_Deg float64
```

In [7]: 1 df.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
```

```
In [8]: 1 df.describe().T
```

Out[8]:

		count	mean	std	min	25%	50%	75%	max
Severity		2845342.0	2.137572	0.478722	1.000000	2.000000	2.000000	2.000000	4.000000e+00
Start_Lat		2845342.0	36.245201	5.363797	24.566027	33.445174	36.098609	40.160243	4.900058e+01
Start_Lng		2845342.0	-97.114633	18.317819	-124.548074	-118.033113	-92.418076	-80.372431	-6.711317e+01
End_Lat		2845342.0	36.245321	5.363873	24.566013	33.446278	36.097987	40.161049	4.907500e+01
End_Lng		2845342.0	-97.114387	18.317632	-124.545748	-118.033331	-92.417718	-80.373383	-6.710924e+01
Distance(mi)		2845342.0	0.702678	1.560361	0.000000	0.052000	0.244000	0.764000	1.551860e+02
Number		1101431.0	8089.408114	18360.093995	0.000000	1270.000000	4007.000000	9567.000000	9.999997e+06
Temperature(F)		2776068.0	61.793556	18.622629	-89.000000	50.000000	64.000000	76.000000	1.960000e+02
Wind_Chill(F)		2375699.0	59.658231	21.160967	-89.000000	46.000000	63.000000	76.000000	1.960000e+02
Humidity(%)		2772250.0	64.365452	22.874568	1.000000	48.000000	67.000000	83.000000	1.000000e+02
Pressure(in)		2786142.0	29.472344	1.045286	0.000000	29.310000	29.820000	30.010000	5.890000e+01
Visibility(mi)		2774796.0	9.099391	2.717546	0.000000	10.000000	10.000000	10.000000	1.400000e+02
Wind_Speed(mph)		2687398.0	7.395044	5.527454	0.000000	3.500000	7.000000	10.000000	1.087000e+03
Precipitation(in)		2295884.0	0.007017	0.093488	0.000000	0.000000	0.000000	0.000000	2.400000e+01

```
In [5]: 1 # How many numeric columns in the dataset
```

```
2  
3 numerics = ['int16', 'int32', 'int64', 'float16', 'float32', 'float64']  
4  
5 numeric_df = df.select_dtypes(include=numerics)  
6 len(numeric_df.columns)  
7
```

Out[5]: 14

## Find the missing values

```
In [44]: 1 df.isna().sum().sort_values(ascending = False)[:10]
```

Out[44]:

Number	1743911
Precipitation(in)	549458
Wind_Chill(F)	469643
Wind_Speed(mph)	157944
Wind_Direction	73775
Humidity(%)	73092

## Find the missing values

```
In [44]: 1 df.isna().sum().sort_values(ascending = False)[:10]
```

```
Out[44]: Number          1743911
Precipitation(in)    549458
Wind_Chill(F)        469643
Wind_Speed(mph)      157944
Wind_Direction       73775
Humidity(%)          73092
Weather_Condition    70636
Visibility(mi)        70546
Temperature(F)        69274
Pressure(in)          59200
dtype: int64
```

## Percentage of missing values per columns

```
In [45]: 1 missing_percentage = df.isna().sum().sort_values(ascending = False) / len(df)
2 missing_percentage[:10]
```

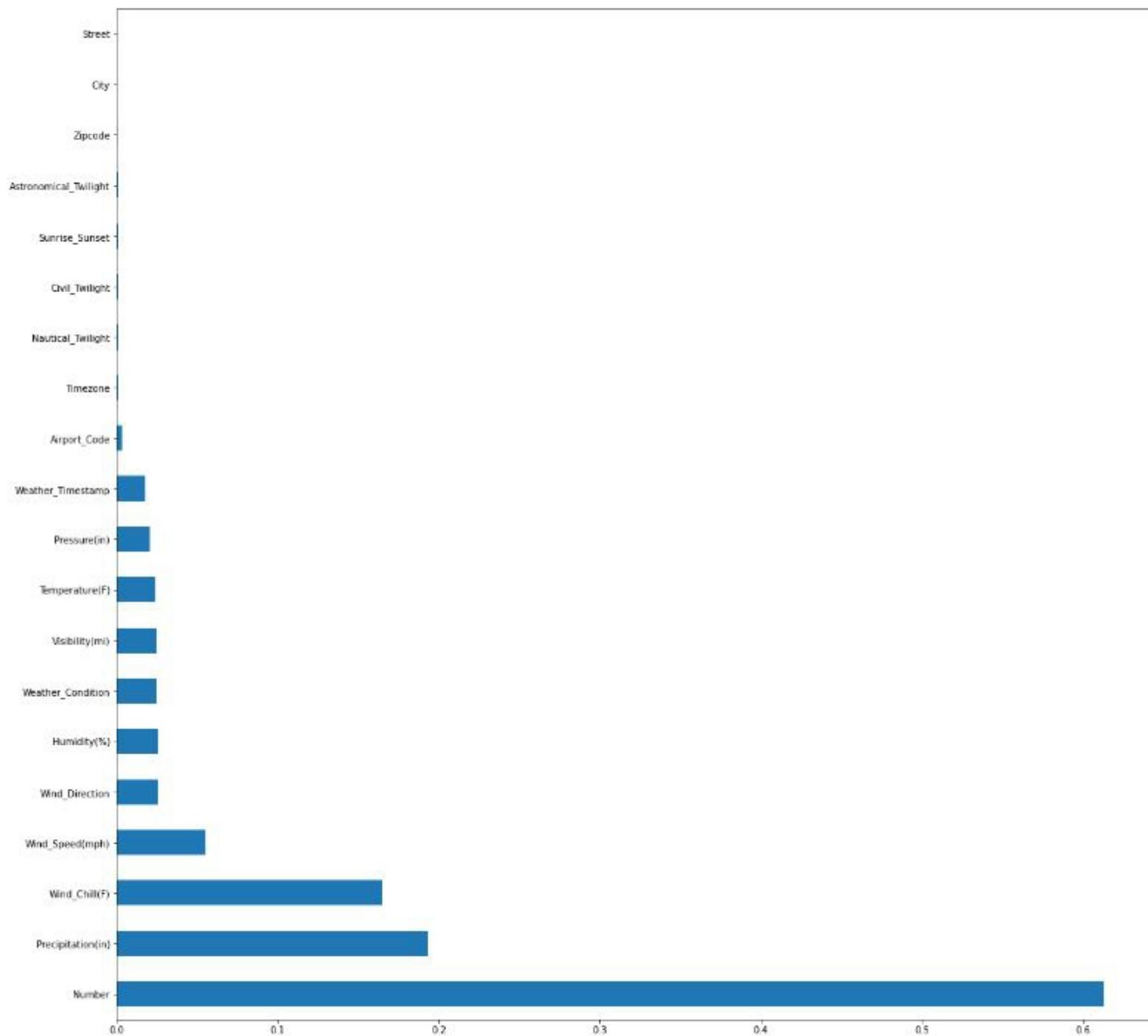
```
Out[45]: Number          0.612900
Precipitation(in)    0.193108
Wind_Chill(F)        0.165057
Wind_Speed(mph)      0.055510
Wind_Direction       0.025928
Humidity(%)          0.025688
Weather_Condition    0.024825
Visibility(mi)        0.024794
Temperature(F)        0.024346
Pressure(in)          0.020806
dtype: float64
```

```
In [11]: 1 # Shows only those columns which has missing values
2
3 missing_percentage[missing_percentage != 0]
```

```
Out[11]: Number          6.129003e-01
Precipitation(in)    1.931079e-01
Wind_Chill(F)        1.650568e-01
Wind_Speed(mph)      5.550967e-02
Wind_Direction       2.592834e-02
Humidity(%)          2.568830e-02
Weather_Condition    2.482514e-02
Visibility(mi)        2.479350e-02
Temperature(F)        2.434646e-02
Pressure(in)          2.080593e-02
```

```
In [12]: 1 missing_percentage[missing_percentage !=0].plot(kind = 'barh', figsize =(20,20))
```

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



## Columns we will analyse

1. City
2. Start\_Time
3. Start\_Lat, Start\_Lng
4. Temperature
5. Weather\_Condition

```
In [12]: 1 cities = df.City.unique()
2 len(cities)
```

```
Out[12]: 11682
```

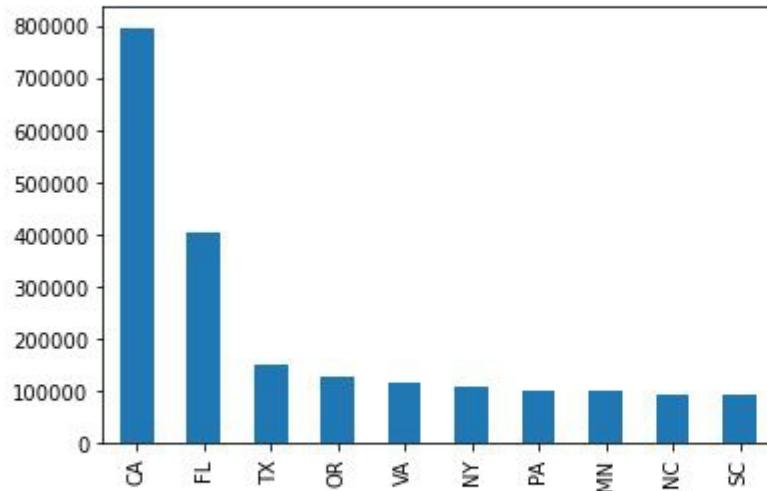
## Answer of Questions

### 1) Which 5 states have the highest number of accidents?

```
In [6]: 1 states_by_accident = df['State'].value_counts()[:10]
```

```
In [7]: 1 states_by_accident.plot(kind = 'bar')
```

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



- Observation

**5 States having the highest number of accidents are**

### 5 States having the highest number of accidents are

```
In [8]: 1 df['State'].value_counts()[:5]
```

```
Out[8]: CA    795868  
FL    401388  
TX    149037  
OR    126341  
VA    113535  
Name: State, dtype: int64
```

### 2) Top 100 cities in number of accidents

```
In [37]: 1 cities_by_accident = df.City.value_counts()  
2 cities_by_accident[:20]
```

```
Out[37]: Miami          106966  
Los Angeles      68956  
Orlando           54691  
Dallas             41979  
Houston            39448  
Charlotte          33152  
Sacramento         32559  
San Diego          26627  
Raleigh             22840  
Minneapolis        22768  
Portland            20944  
Nashville           20267  
Austin              18301  
Baton Rouge         18182  
Phoenix              17143  
Saint Paul           16869  
New Orleans          16251  
Atlanta              15622  
Jacksonville         14967  
Richmond             14349  
Name: City, dtype: int64
```

### 3) What time of the day are accidents most frequent in ?

```
In [16]: 1 df.Start_Time[0]
```

```
Out[16]: '2016-02-08 00:37:08'
```

### 3) What time of the day are accidents most frequent in ?

```
In [16]: 1 df.Start_Time[0]
```

```
Out[16]: '2016-02-08 00:37:08'
```

```
In [17]: 1 df.Start_Time = pd.to_datetime(df.Start_Time)
```

```
In [18]: 1 # convert Datetime to hour  
2  
3 pd.DatetimeIndex(df['Start_Time']).hour
```

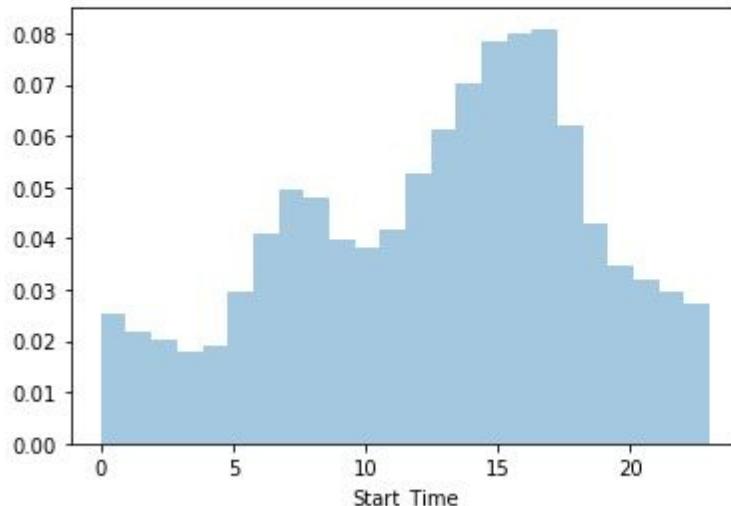
```
Out[18]: Int64Index([ 0,  5,  6,  6,  7,  8,  8, 11, 14, 15,  
...  
    17, 17, 17, 17, 18, 18, 19, 19, 19, 18],  
   dtype='int64', name='Start_Time', length=2845342)
```

```
In [12]: 1 # Plot Histogram  
2  
3 sns.distplot(pd.DatetimeIndex(df['Start_Time']).hour, bins = 24, kde = False, norm_hist= True)
```

```
C:\Users\SAHILJOSAN\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 `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
```

```
    warnings.warn(msg, FutureWarning)
```

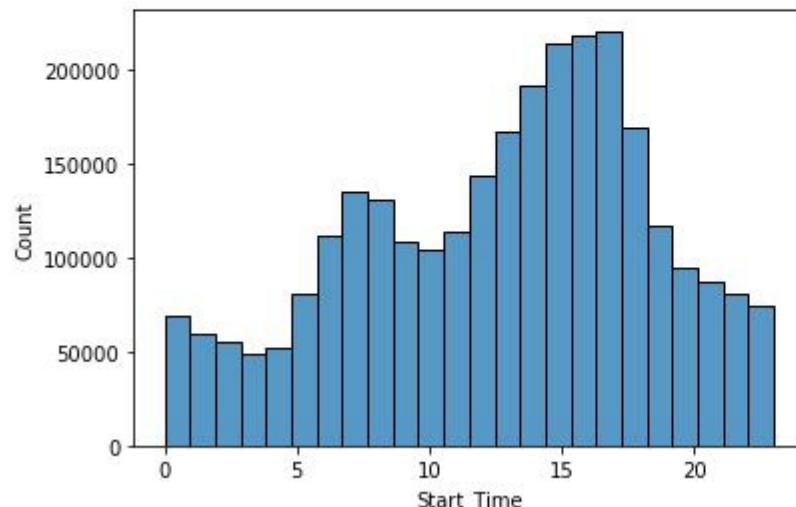
```
Out[12]: <AxesSubplot:xlabel='Start_Time'>
```



## At which time of the day highest accidents occur

```
In [94]: 1 sns.histplot(data = df, x = pd.DatetimeIndex(df['Start_Time']).hour, bins = 24)
```

```
Out[94]: <AxesSubplot:xlabel='Start_Time', ylabel='Count'>
```



- Observation

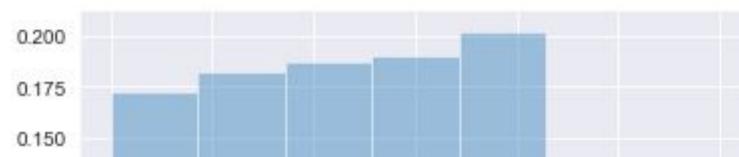
**A high number of the accidents occur between 3 pm to 5 pm**

## Is the distribution of accidents by hour the same on weekends as on weekdays?

```
In [28]: 1 sns.distplot(df.Start_Time.dt.dayofweek, bins = 7, kde = False, norm_hist= True)
```

```
C:\Users\SAHILJOSAN\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 `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

```
Out[28]: <AxesSubplot:xlabel='Start_Time'>
```

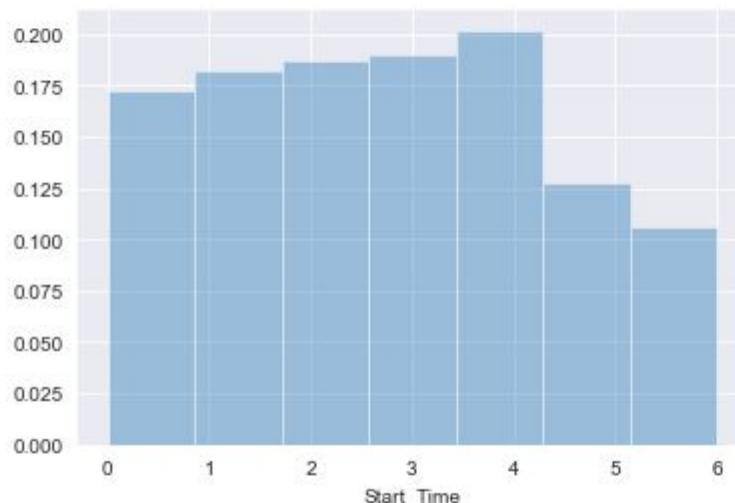


## Is the distribution of accidents by hour the same on weekends as on weekdays?

```
In [28]: 1 sns.distplot(df.Start_Time.dt.dayofweek, bins = 7, kde = False, norm_hist= True)
```

```
C:\Users\SAHILJOSAN\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 `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).  
warnings.warn(msg, FutureWarning)
```

```
Out[28]: <AxesSubplot:xlabel='Start_Time'>
```



- Observation

**From the graph it is seen that on weekdays more number of accidents occur compare to weekends**

**Over 1110 cities have reported just 1 accident (need to investigate)**

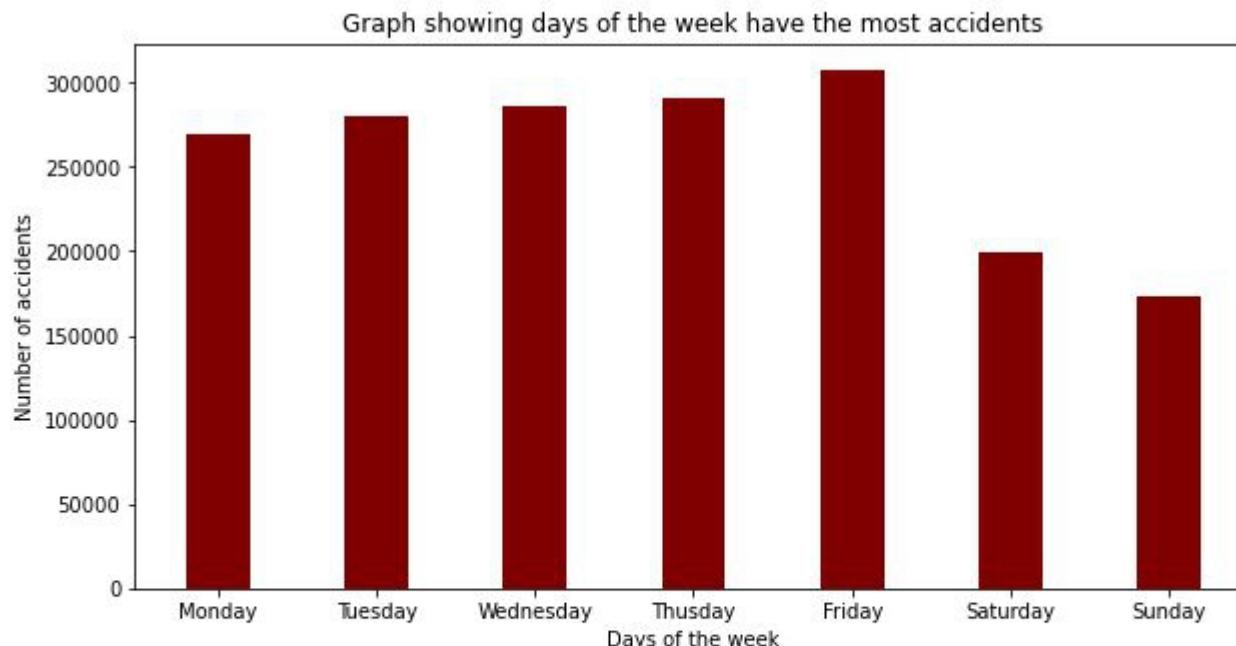
```
In [29]: 1 cities_by_accident[cities_by_accident == 1]
```

```
Out[29]: Carney                      1  
Waverly Hall                   1  
Center Sandwich                 1  
Glen Flora                     1  
Sulphur Springs                 1  
..
```

```
Sulphur Springs          1
Ridgedale                ..
Sekiu                     1
Wooldridge               1
Bullock                   1
American Fork-Pleasant Grove  1
Name: City, Length: 1110, dtype: int64
```

#### 4) Which days of the week have the most accidents ?

```
In [34]: 1 fig = plt.figure(figsize = (10, 5))
2
3 # creating the bar plot
4 plt.bar(days,list_days_of_the_week, color ='maroon',width = 0.4)
5
6 plt.xlabel("Days of the week")
7 plt.ylabel("Number of accidents")
8 plt.title("Graph showing days of the week have the most accidents")
9 plt.show()
```



- Observation

**Maximum number of accidents occurs on Fridays**

```
In [20]: 1 list_days_of_the_week = []
2 for i in range(0,7):
3     list_days_of_the_week.append(len(df.Start_Time[df.Start_Time.dt.dayofweek == (i)].unique()))
4 list_days_of_the_week
```

```
Out[20]: [268929, 280241, 286551, 291495, 307609, 198976, 173510]
```

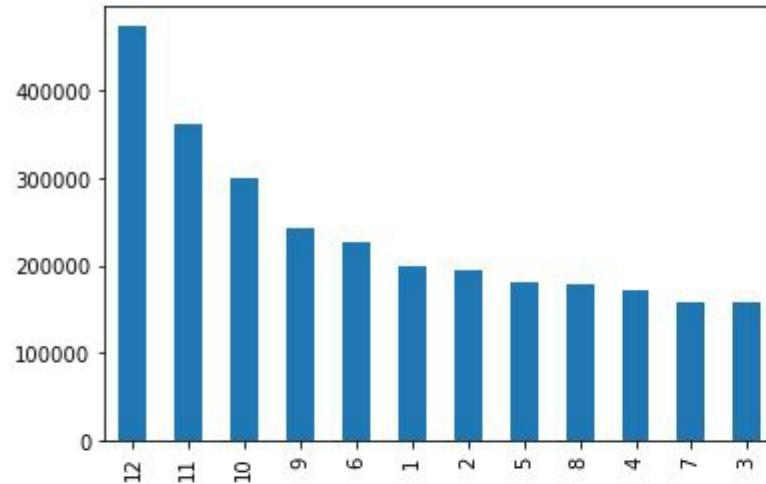
```
In [28]: 1 days = ["Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "Sunday"]
2 days
```

```
Out[28]: ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday']
```

## 5) Which months have the most accidents?

```
In [44]: 1 df.Start_Time.dt.month.value_counts().plot(kind = "bar")
```

```
Out[44]: <AxesSubplot:>
```



## 6) What is the trend of accidents year over year (decreasing/increasing) ?

```
In [38]: 1 trend_of_accident = df.Start_Time.dt.year.value_counts()
2 trend_of_accident.plot(kind= "bar")
```

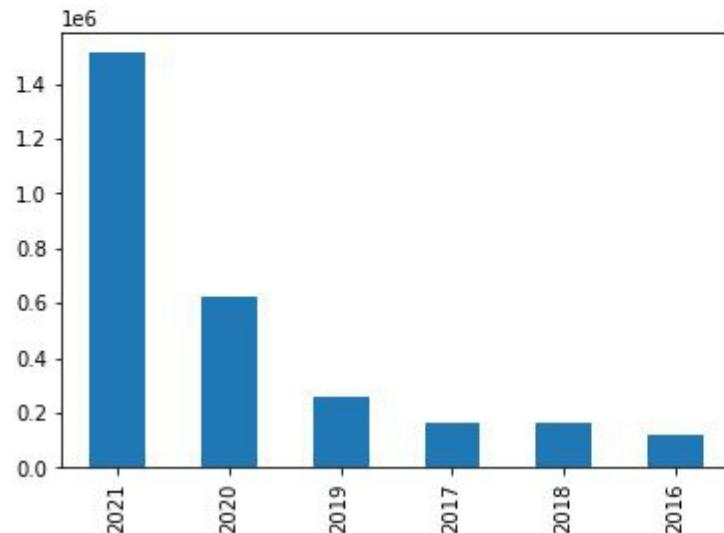
```
Out[38]: <AxesSubplot:>
```



## 6) What is the trend of accidents year over year (decreasing/increasing) ?

```
In [38]: 1 trend_of_accident = df.Start_Time.dt.year.value_counts()  
2 trend_of_accident.plot(kind= "bar")
```

Out[38]: <AxesSubplot:>



```
In [86]: 1 df.Start_Time.dt.year.value_counts()
```

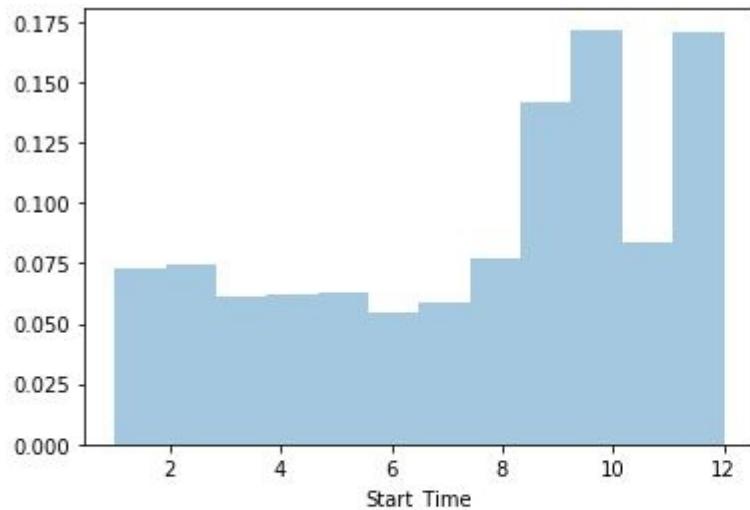
```
Out[86]: 2021    1511745  
2020     625864  
2019     258615  
2017     163918  
2018     163176  
2016     122024  
Name: Start_Time, dtype: int64
```

## Accidents occur in the year 2019

```
In [78]: 1 # Data of 2019  
2  
3 df_2019 = df[df.Start_Time.dt.year == 2019]  
4 sns.distplot(df_2019.Start_Time.dt.month, bins = 12, kde = False, norm_hist = True)
```

C:\Users\SAHILJOSAN\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 `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).

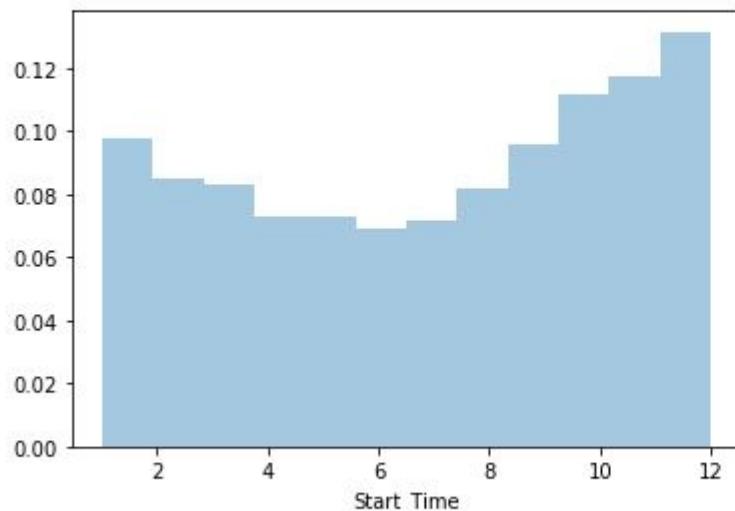
```
warnings.warn(msg, FutureWarning)
```



## Accidents occur in the year 2018

```
In [79]: 1 # Data of 2018
          2
          3 df_2018 = df[df.Start_Time.dt.year == 2018]
          4 sns.distplot(df_2018.Start_Time.dt.month, bins = 12, kde = False, norm_hist = True)
```

```
Out[79]: <AxesSubplot:xlabel='Start_Time'>
```

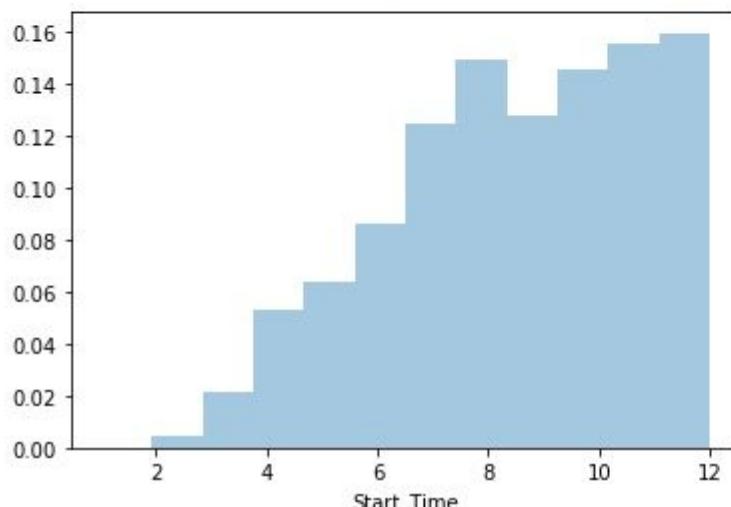


## Accidents occur in the year 2016

In [80]:

```
1 # Data of 2016
2
3 df_2016 = df[df.Start_Time.dt.year == 2016]
4 sns.distplot(df_2016.Start_Time.dt.month, bins = 12, kde = False, norm_hist = True)
```

Out[80]:



- Observation

**Much Data is missing for 2016. Maybe even 2017**

## High Accident Cities and low accident cities

In [19]:

```
1 high_accident_cities = cities_by_accident[cities_by_accident >= 1000]
2 low_accident_cities = cities_by_accident[cities_by_accident < 1000]
```

In [20]:

```
1 len(high_accident_cities)
```

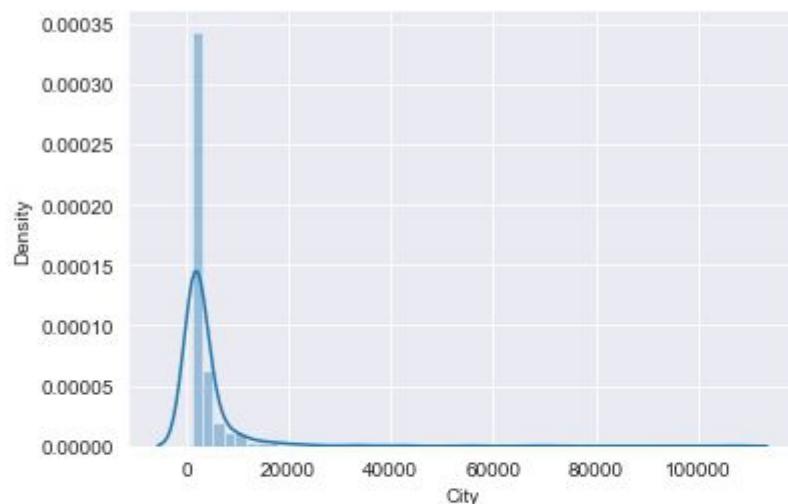
Out[20]:

496

In [21]:

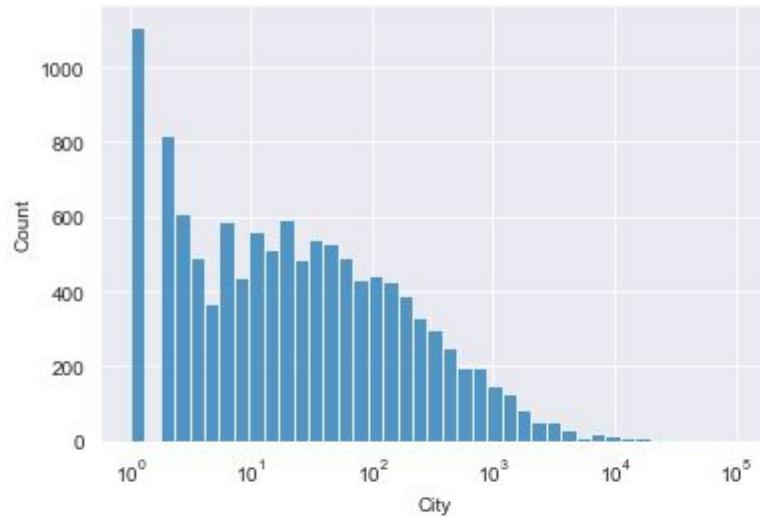
```
1 sns.distplot(high_accident_cities)
```

C:\Users\SAHILJOSAN\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 `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level funct



```
In [22]: 1 sns.histplot(cities_by_accident, log_scale = True)
```

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



## Analysis basis on Start Latitude & Longitude

```
In [28]: 1 df.Start_Lat[:5]
```

```
Out[28]: 0    40.108910
1    39.865420
2    39.102660
3    41.062130
4    39.172393
Name: Start_Lat, dtype: float64
```

## Analysis basis on Start Latitude & Longitude

```
In [28]: 1 df.Start_Lat[:5]
```

```
Out[28]: 0    40.108910
1    39.865420
2    39.102660
3    41.062130
4    39.172393
Name: Start_Lat, dtype: float64
```

```
In [29]: 1 df.Start_Lng[:5]
```

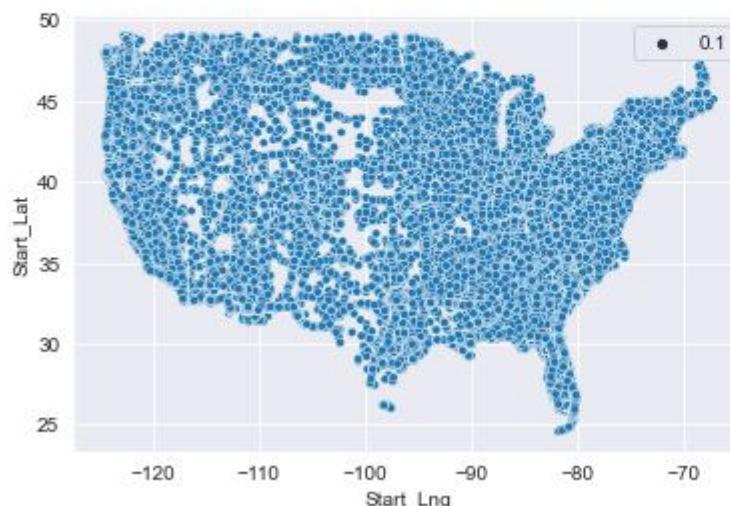
```
Out[29]: 0   -83.092860
1   -84.062800
2   -84.524680
3   -81.537840
4   -84.492792
Name: Start_Lng, dtype: float64
```

## Scatterplot using Start\_Lng and Start\_Lat columns

- By using scatterplot we can see the area in the country where accident occurs

```
In [84]: 1 sns.scatterplot(x = df.Start_Lng, y=df.Start_Lat, size = 0.1)
```

```
Out[84]: <AxesSubplot:xlabel='Start_Lng', ylabel='Start_Lat'>
```

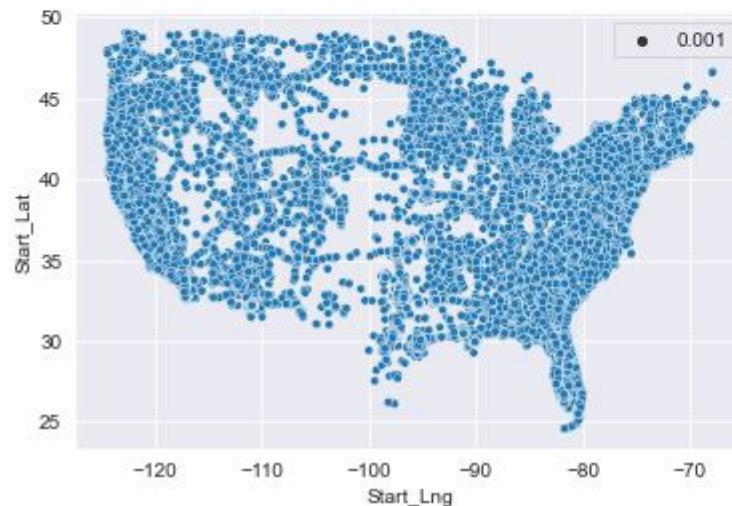


**Data is very big so we have take 0.1% sample of whole dataset**

In [86]:

```
1 # Take sample of the dataset and plot it
2
3 sample_df = df.sample(int(0.1 * len(df)))
4
5 sns.scatterplot(x = sample_df.Start_Lng, y = sample_df.Start_Lat, size = 0.001)
```

Out[86]: <AxesSubplot:xlabel='Start\_Lng', ylabel='Start\_Lat'>



**Import Folium to plot the markers on google map**

In [46]:

```
1 import folium
```

In [47]:

```
1 lat,lng = df.Start_Lat[0], df.Start_Lng[0]
2 lat,lng
```

Out[47]: (40.10891, -83.09286)

**Plot 1 marker on the map**

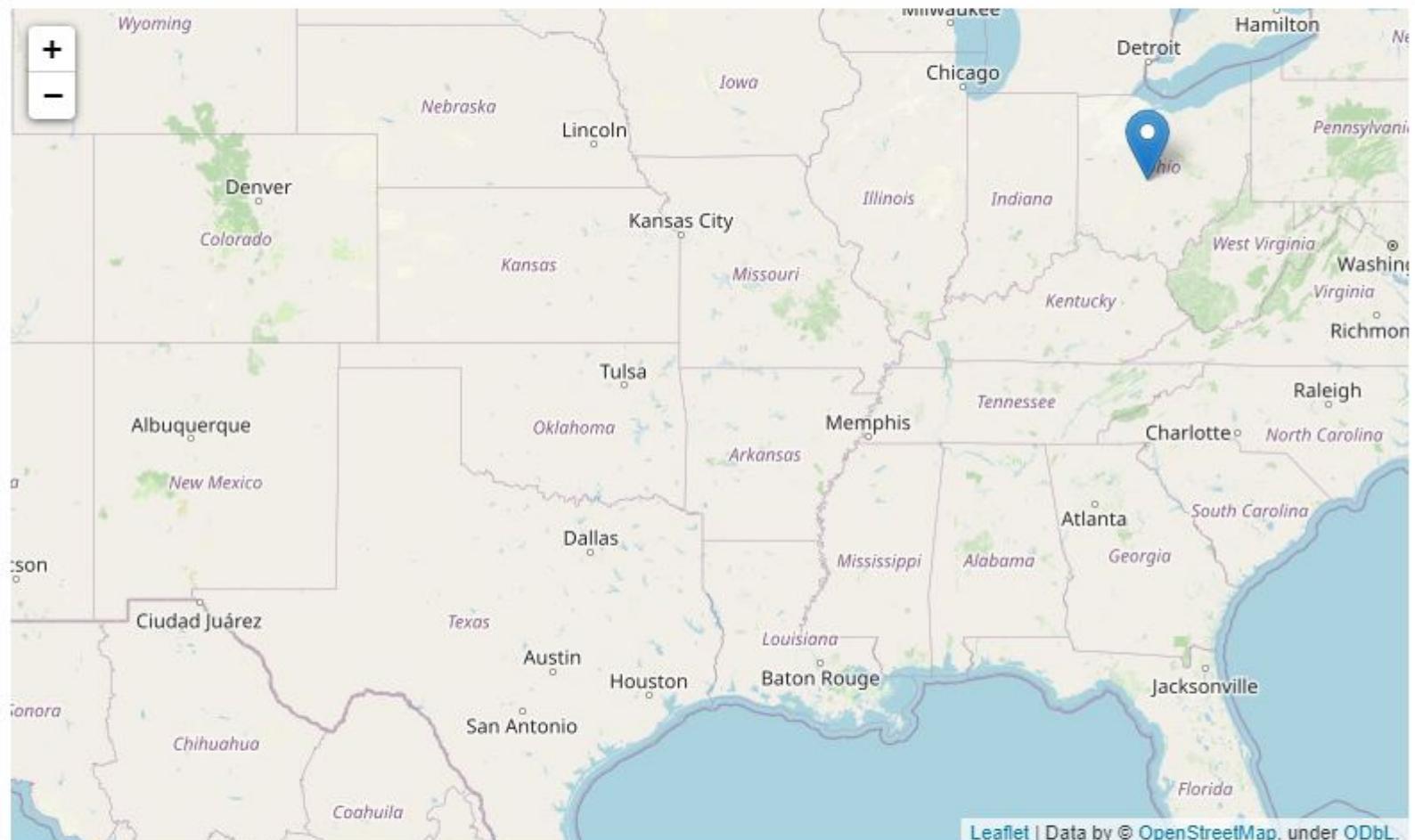
In [48]:

```
1 # plot 1 marker on the map
2
3 map = folium.Map()
4 marker = folium.Marker((lat,lng))
5 marker.add_to(map)
6 map
```

## Plot 1 marker on the map

```
In [48]: 1 # plot 1 marker on the map  
2  
3 map = folium.Map()  
4 marker = folium.Marker((lat,lng))  
5 marker.add_to(map)  
6 map
```

Out[48]:



```
In [49]: 1 list(zip(list(df.Start_Lat), list(df.Start_Lng)))[::10]
```

Out[49]: [(40.10891, -83.09286),  
(39.86542, -84.0628),  
(39.10266, -84.52468),  
(41.06213, -81.53784),  
(39.172393, -84.49279200000002),  
(39.06324, -84.03243),  
(39.77565, -84.18603),  
(41.37531, -81.82016999999998),

## To plot multiple markers on the map

- We have to take 0.01% of sample of whole data

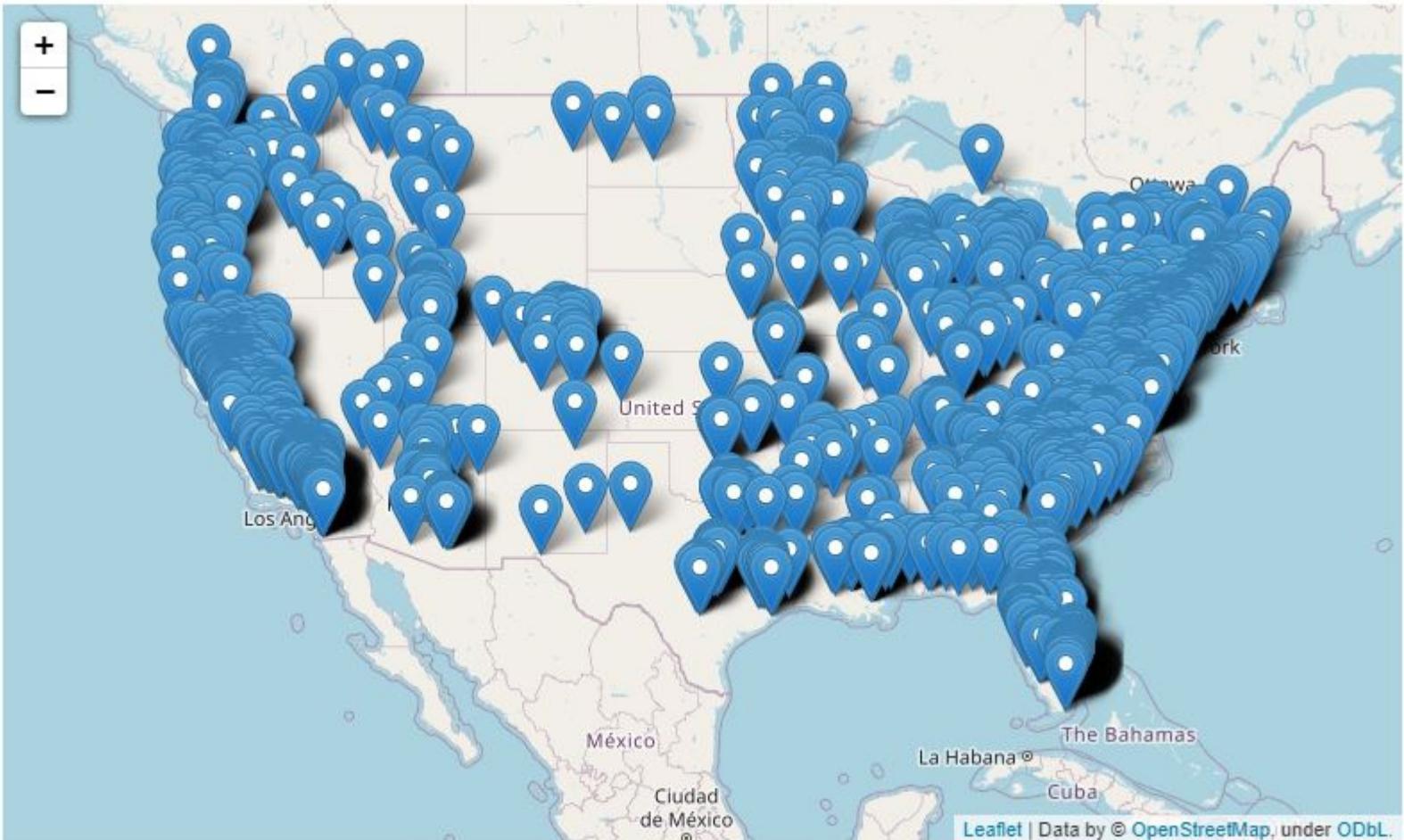
```
In [50]: 1 sample_df = df.sample(int(0.001 * len(df)))
```

```
In [51]: 1 locations = sample_df[['Start_Lat', 'Start_Lng']]  
2 locationlist = locations.values.tolist()  
3 len(locationlist)  
4 locationlist[7]
```

```
Out[51]: [26.775749, -80.097928]
```

```
In [52]: 1 map = folium.Map()  
2 for point in range(0, len(locationlist)):  
3     marker = folium.Marker((locationlist[point]))  
4     marker.add_to(map)  
5 map
```

```
Out[52]:
```



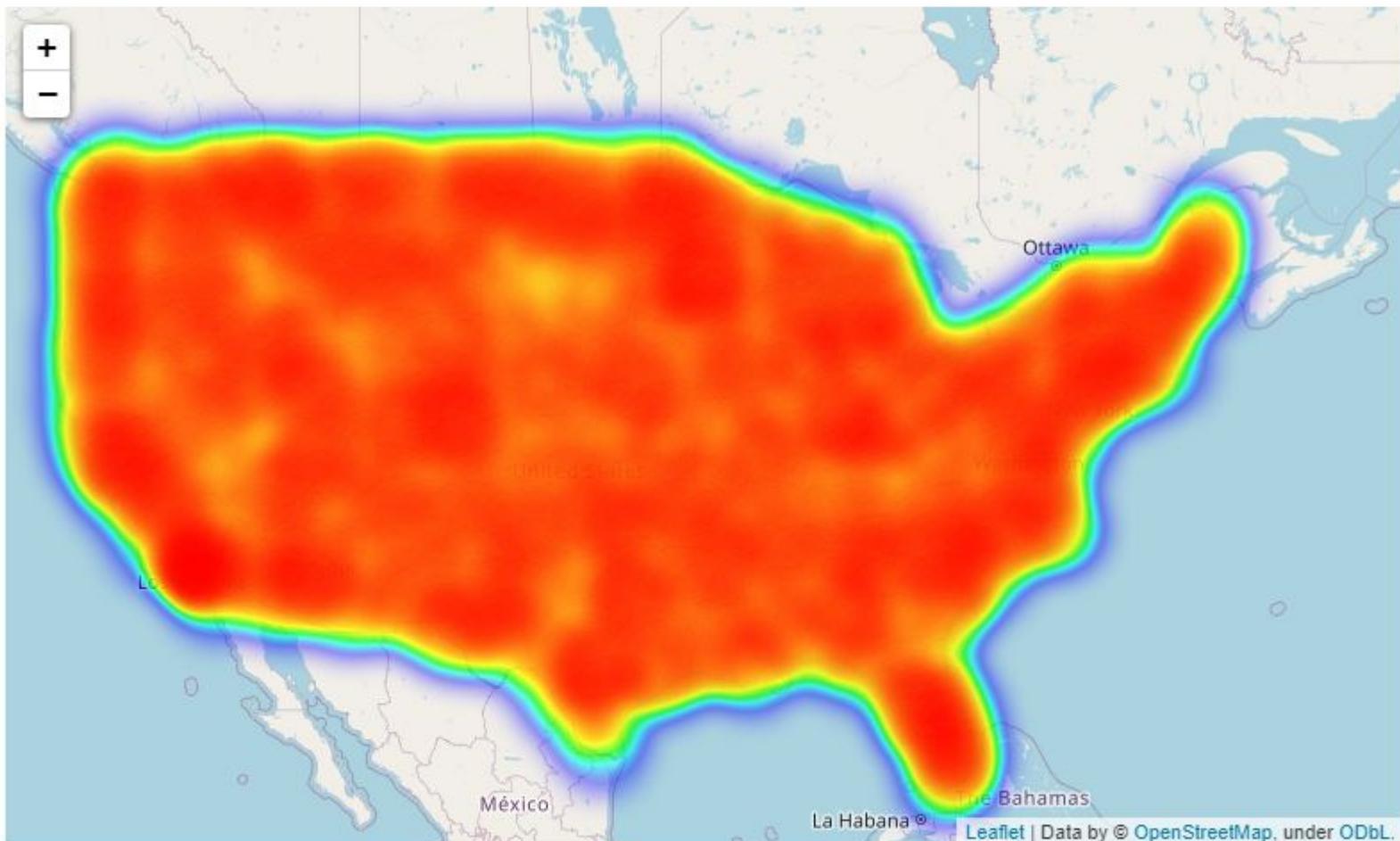
- Observation

The above maps mark's the states where accidents occurred

### Heatmap of area where accidents occurred using Start\_Lat,Start\_Lng

```
In [53]: 1 from folium import plugins  
2 from folium.plugins import HeatMap  
3  
4 map = folium.Map()  
5 HeatMap(zip(list(df.Start_Lat), list(df.Start_Lng))).add_to(map)  
6 map
```

Out[53]:



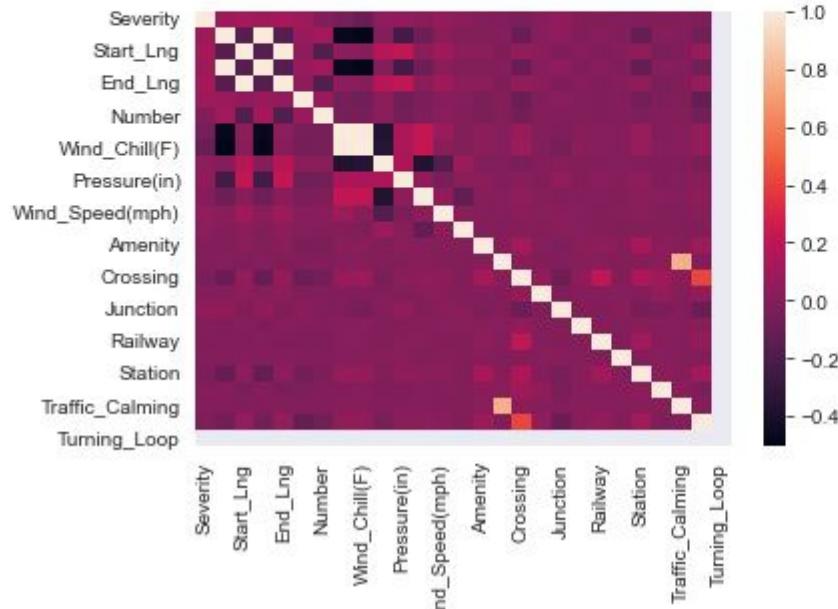
```
In [52]: 1 df.columns
```

```
Out[52]: Index(['ID', 'Severity', 'Start_Time', 'End_Time', 'Start_Lat', 'Start_Lng',
```

## Heatmap of correlation

In [49]: 1 sns.heatmap(df.corr())

Out[49]: <AxesSubplot:>



## Summary and Conclusion

### Insights

- Less than 5% of cities have more than 1000 yearly accidents
- We have seen top 5 states having more number of accidents
- We have seen top 100 cities having more number of accidents
- High number of accidents occur between 3 pm to 5 pm
- Over 1110 cities have reported just 1 accident (need to investigate)
- Maximum number of accidents occurs on Fridays
- December month have reported the maximum number of accidents
- Much Data of 2016 and 2017 is missing, still from the graph we can say that the trend of accidents increases year by year
- We have used scatterplot using Start\_longitude, Start\_latitude
- We have used Folium to plot the markers on google map
- We have seen the Heatmap of areas where accidents occurred using Start\_lat, Start\_Lng