This Project is related to a countrywide car accident dataset, which covers 49 states of the USA. The accident data are collected from February 2016 to Dec 2021, using multiple APIs that provide streaming traffic incident (or event) data. These APIs broadcast traffic data captured by a variety of entities, such as the US and state departments of transportation, law enforcement agencies, traffic cameras, and traffic sensors within the road-networks. Currently, there are about 2.8 million accident records in this datase.

Data is taken from the kaggle - https://www.kaggle.com/datasets/sobhanmoosavi/us-accidents

Lets import the libararies

[402…

```
import os
import pandas as pd
import  matplotlib.pyplot as plt
import numpy as np
import seaborn as sns


import matplotlib.mlab as mlab
import matplotlib
plt.style.use('ggplot')
from matplotlib.pyplot import figure

%matplotlib inline
matplotlib.rcParams['figure.figsize'] = (16,12)

pd.options.mode.chained_assignment = None
```

Here's the step by step process that i am going to follow :

1. Dataset is from Kaggle
2. Perform data preparation & cleaning using Pandas & Numpy
3. Perform exploratory analysis & visualization using Matplotlib & Seaborn
4. Asking & answer questions about the data
5. Summarizing & write a conclusion

Selecting a large real-world dataset from Kaggle

In [403...
```
df = pd.read_csv('US_Accidents_Dec21_updated.csv')
```

In [404...
```
df
```

Out[404...

	ID	Severity	Start_Time	End_Time	Start_Lat	Start_Lng	End_Lat	End_Lng	Distance(mi)	Description	...	Roundabout	Station	Stop	T
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...	...	False	False	False	
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.	...	False	False	False	
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.	...	False	False	False	
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.	...	False	False	False	
4	A-5	3	2016-02-	2016-02-08	39.172393	84.492792	39.170476	84.501798	0.500	At Mitchell Ave/Exit 6	...	False	False	False	

In [405…

```
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 [406…

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

Out[406…

```
ID                        0
Severity                  0
Start_Time                0
End_Time                  0
Start_Lat                 0
Start_Lng                 0
End_Lat                   0
End_Lng                   0
Distance(mi)              0
Description               0
Number              1743911
Street                    2
Side                      0
City                    137
County                    0
State                     0
Zipcode                1319
Country                   0
Timezone               3659
Airport_Code           9549
Weather_Timestamp     50736
Temperature(F)        69274
Wind_Chill(F)        469643
Humidity(%)           73092
Pressure(in)          59200
Visibility(mi)        70546
Wind_Direction        73775
Wind_Speed(mph)      157944
Precipitation(in)    549458
Weather_Condition     70636
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         2867
Civil_Twilight         2867
Nautical_Twilight      2867
```

The below code gives me the percentage of the null values in each coloumn and it is evedent that number coloumn has highest percentage of the null values

In [407…

```
for col in df.columns:
    pct_missing = np.mean(df[col].isnull())
    print('{} - {}%'.format(col, round(pct_missing*100)))
```

```
ID - 0%
Severity - 0%
Start_Time - 0%
End_Time - 0%
Start_Lat - 0%
Start_Lng - 0%
End_Lat - 0%
End_Lng - 0%
Distance(mi) - 0%
Description - 0%
Number - 61%
Street - 0%
Side - 0%
City - 0%
County - 0%
State - 0%
Zipcode - 0%
Country - 0%
Timezone - 0%
Airport_Code - 0%
Weather_Timestamp - 2%
Temperature(F) - 2%
Wind_Chill(F) - 17%
Humidity(%) - 3%
Pressure(in) - 2%
Visibility(mi) - 2%
Wind_Direction - 3%
Wind_Speed(mph) - 6%
Precipitation(in) - 19%
Weather_Condition - 2%
Amenity - 0%
Bump - 0%
Crossing - 0%
Give_Way - 0%
Junction - 0%
No_Exit - 0%
Railway - 0%
Roundabout - 0%
Station - 0%
```

In [409…

```
numeric_cols = df._get_numeric_data().columns
print(len(numeric_cols))
```

27

In [410…

```
numeric_df = df.select_dtypes(include=['float64','int64'])
print(len(numeric_df.columns))
```

14

lets plot the missing value in the graph

In [411…

```
missing_percent  = df.isnull().sum().sort_values(ascending = False) / len(df)
missing_percent
```

Out[411…

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
Weather_Timestamp	1.783125e-02
Airport_Code	3.356011e-03
Timezone	1.285961e-03
Nautical_Twilight	1.007612e-03
Civil_Twilight	1.007612e-03
Sunrise_Sunset	1.007612e-03
Astronomical_Twilight	1.007612e-03
Zipcode	4.635647e-04
City	4.814887e-05
Street	7.029032e-07
Country	0.000000e+00
Junction	0.000000e+00
Start_Time	0.000000e+00
End_Time	0.000000e+00
Start_Lat	0.000000e+00
Turning_Loop	0.000000e+00
Traffic_Signal	0.000000e+00
Traffic_Calming	0.000000e+00
Stop	0.000000e+00
Station	0.000000e+00

Lets plot the graph to get the missing values from the data frame

In [412…

```
missing_percent[missing_percent !=0] # this will give the only non zeroes values as i am going to plot the data in the graph i dodnt need the zeroes
```

Out[412…

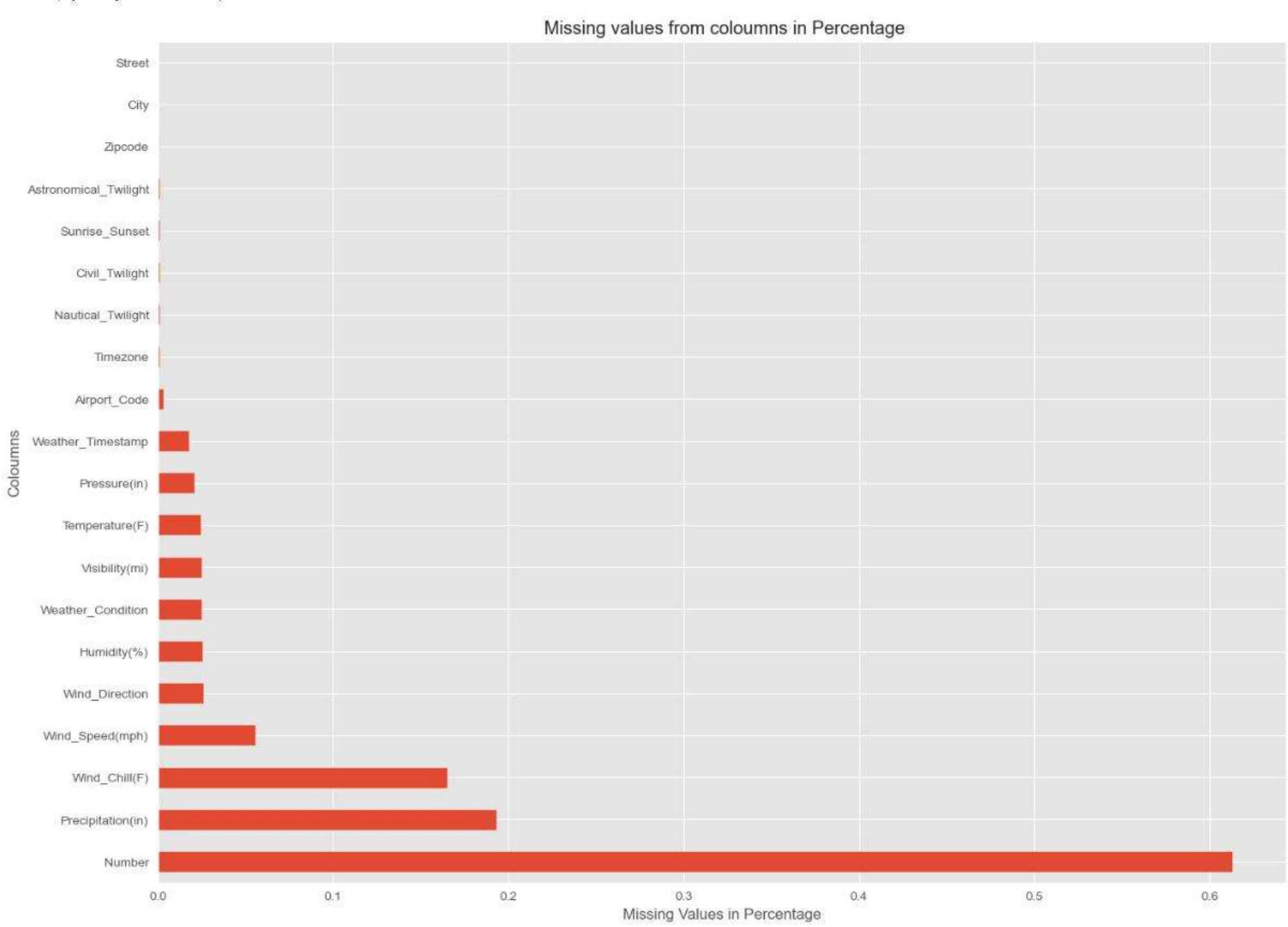
```
Number                  6.129003e-01
Precipitation(in)       1.931079e-01
Wind_Chill(F)           1.650568e-01
Wind_Speed(mph)         5.550967e-02
Wind_Direction          2.592834e-02
Humidity(%)             2.568830e-02
Weather_Condition       2.482514e-02
Visibility(mi)          2.479350e-02
Temperature(F)          2.434646e-02
Pressure(in)            2.080593e-02
Weather_Timestamp       1.783125e-02
Airport_Code            3.356011e-03
Timezone                1.285961e-03
Nautical_Twilight       1.007612e-03
Civil_Twilight          1.007612e-03
Sunrise_Sunset          1.007612e-03
Astronomical_Twilight   1.007612e-03
Zipcode                 4.635647e-04
City                    4.814887e-05
Street                  7.029032e-07
dtype: float64
```

In [413…

```
missing_percent[missing_percent !=0].plot(kind='barh').set(title= ' Missing values from coloumns in Percentage')
plt.xlabel('Missing Values in Percentage')
plt.ylabel('Coloumns')
```

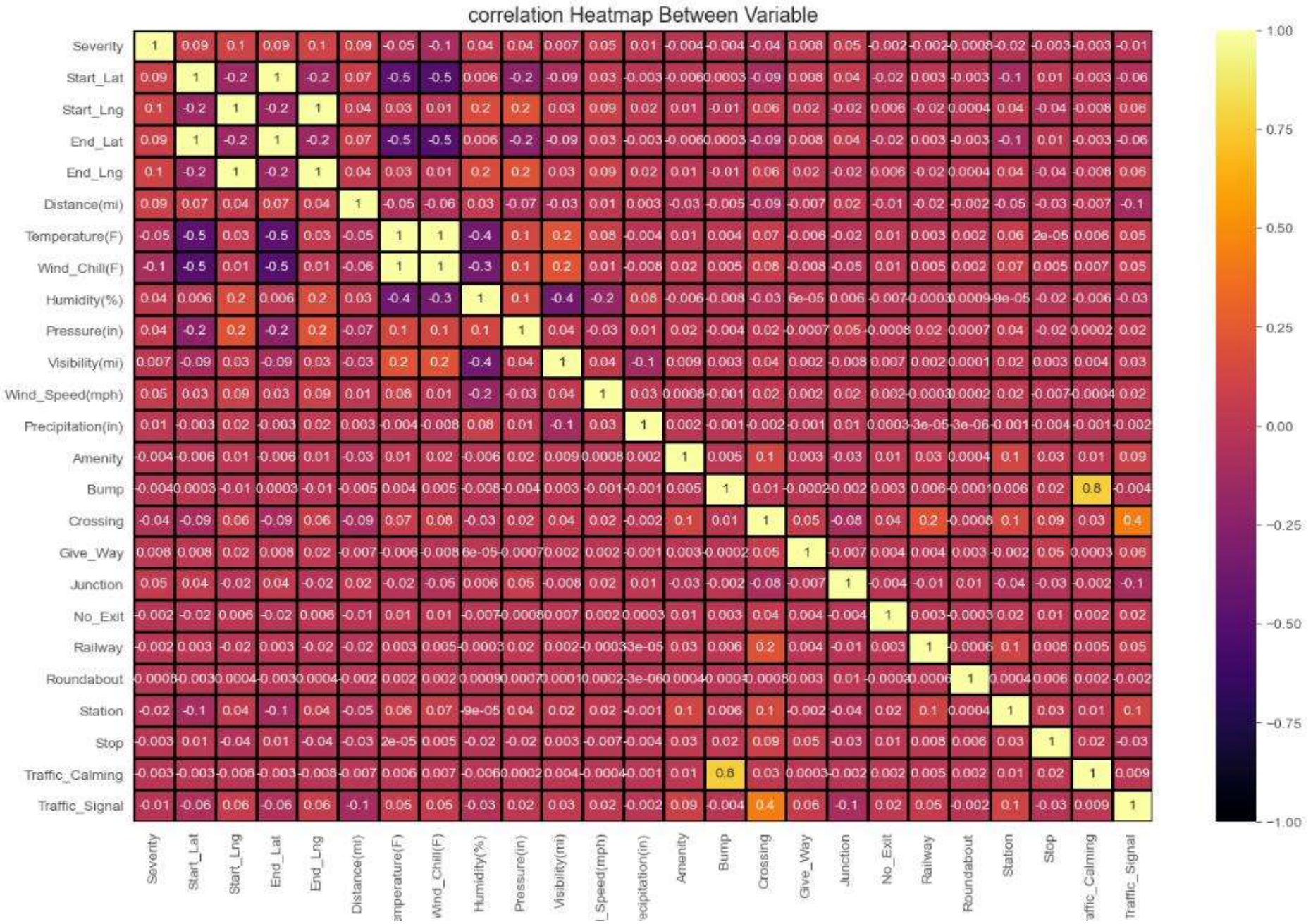
Out[413…

```
Text(0, 0.5, 'Coloumns')
```


Missing values from coloumns in Percentage
Street
City
Zipcode
Astronomical_Twilight
Sunrise_Sunset
Civil_Twilight
Nautical_Twilight
Timezone
Airport_Code
Weather_Timestamp
Pressure(in)
Temperature(F)
Visibility(mi)
Weather_Condition
Humidity(%)
Wind_Direction
Wind_Speed(mph)
Wind_Chill(F)
Precipitation(in)
Number
Coloumns
0.0
0.1
0.2
0.3
0.4
0.5
0.6
Missing Values in Percentage

In [416…

```
corr_df=df.drop(["Turning_Loop","Number"],axis=1).corr(method="pearson")
plt.figure(figsize=(16,10))
heatmap=sns.heatmap(corr_df, annot=True, fmt=".1g", vmin=-1, vmax=1, center=0, cmap="inferno", linewidths=1, linecolor="Black")
heatmap.set_title("correlation Heatmap Between Variable")
heatmap.set_xticklabels(heatmap.get_xticklabels(),rotation=90)
plt.show()
```



There are 11682 unique values of the cities in these data and "NEW YORK " has not been included

In [ ]:

In [418…
```
Cities = df.City.unique()
len(Cities)
```

Out[418… 11682

In [419…
```
city_by_accident =  df.City.value_counts()
city_by_accident
```

Out[419…
```
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
```

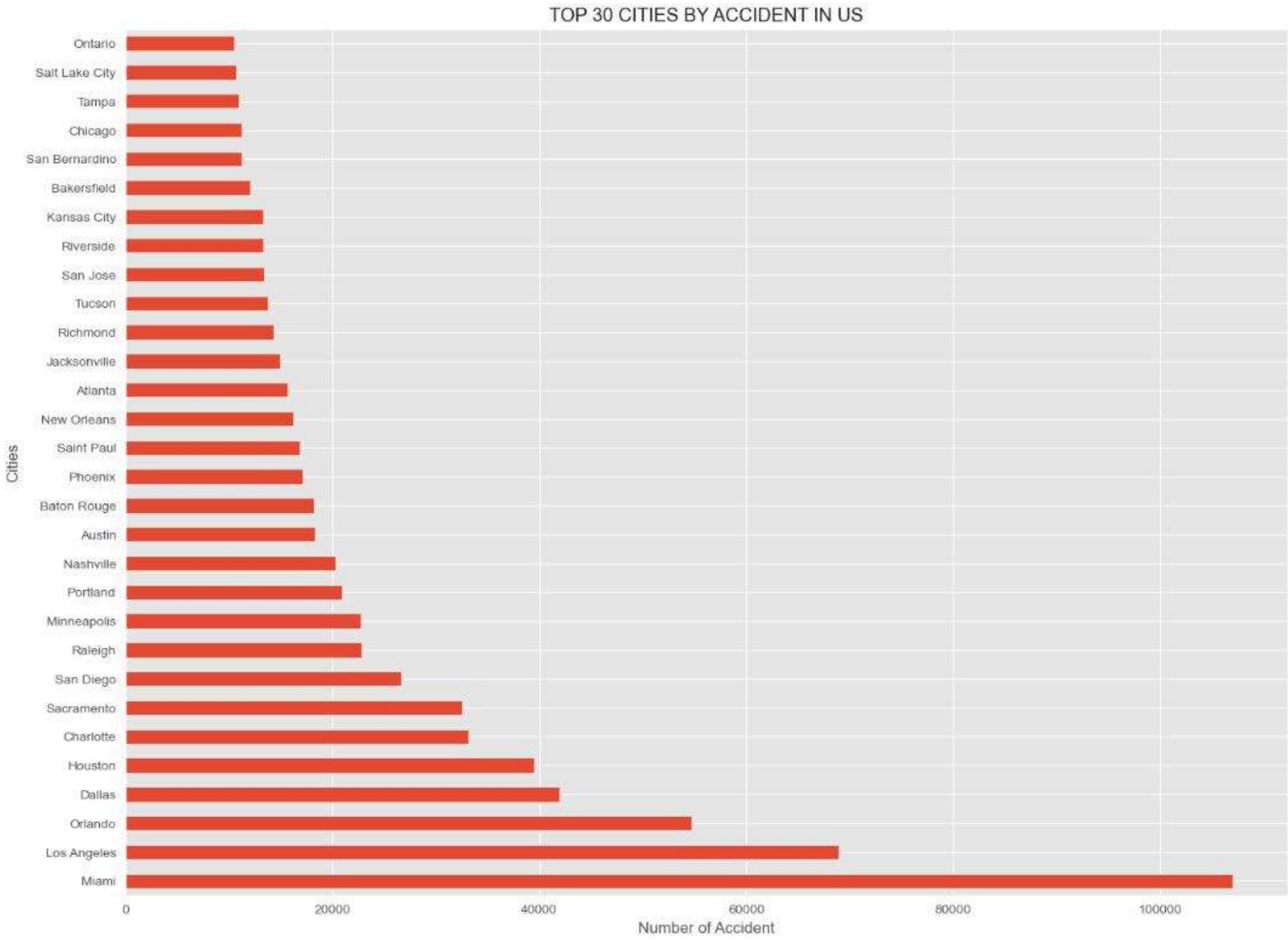
In [420…
```
city_by_accident[:20]
```

Out[420…
```
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
```

In [421…

```
city_by_accident[:30].plot(kind = 'barh').set(title= ' TOP 30 CITIES BY ACCIDENT IN US ')
plt.xlabel('Number of Accident')
plt.ylabel('Cities')
```

Out[421… `Text(0, 0.5, 'Cities')`



In [422…

```
high_accident_cities = city_by_accident[city_by_accident >= 1000]
low_accident_cities = city_by_accident[city_by_accident < 1000]
```

In [423…

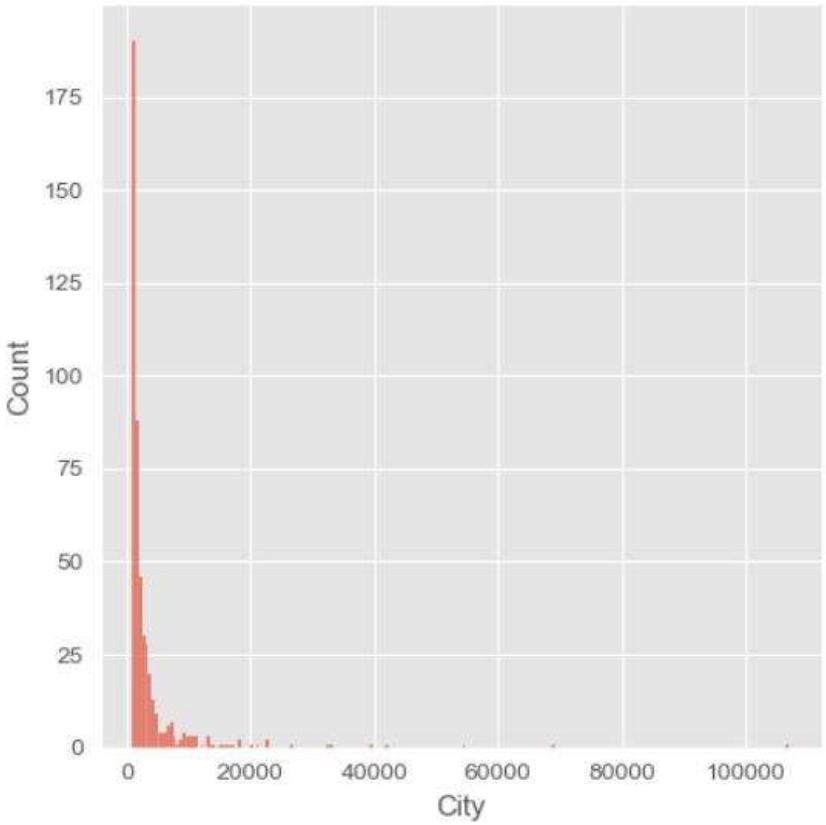
```
len (high_accident_cities) / len(Cities)
```

Out[423… 0.04245848313644924

In [424…

```
sns.displot(high_accident_cities)
```

Out[424… <seaborn.axisgrid.FacetGrid at 0x1538d5dc0d0>

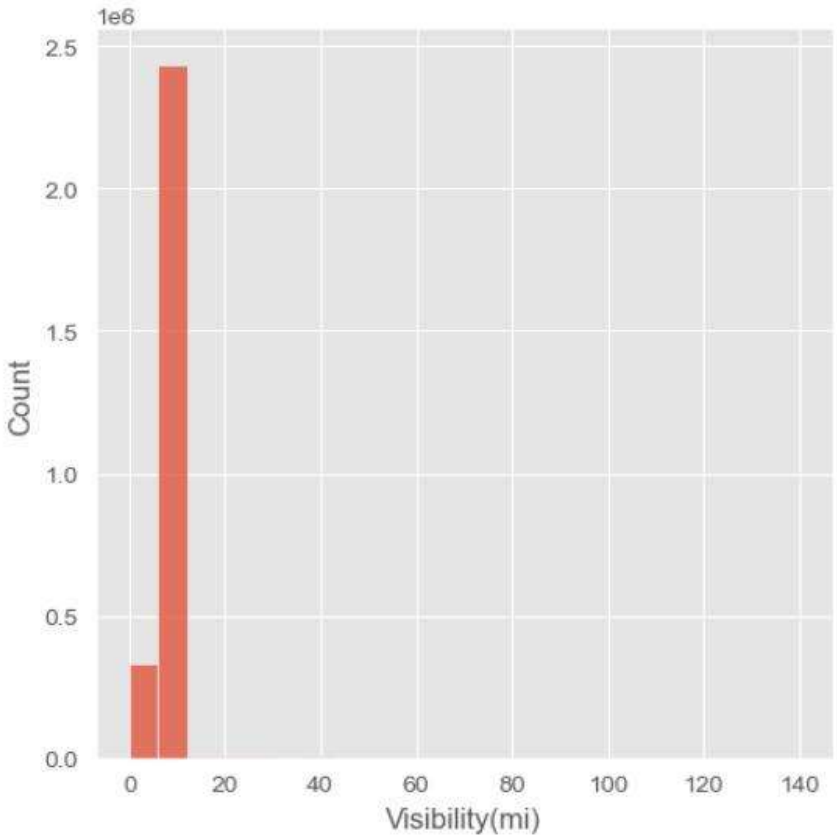


In [425…

```
column_name = "Visibility(mi)"
column_data = df[column_name]

sns.displot(column_data)
```

Out[425… <seaborn.axisgrid.FacetGrid at 0x153afc0bac0>



lets plot some other graphs for the cities based on other conditions

In [426...
```
df.Severity.unique()
```

Out[426...
```
array([3, 2, 4, 1], dtype=int64)
```

plot for the top 30 cities with Severity > 2

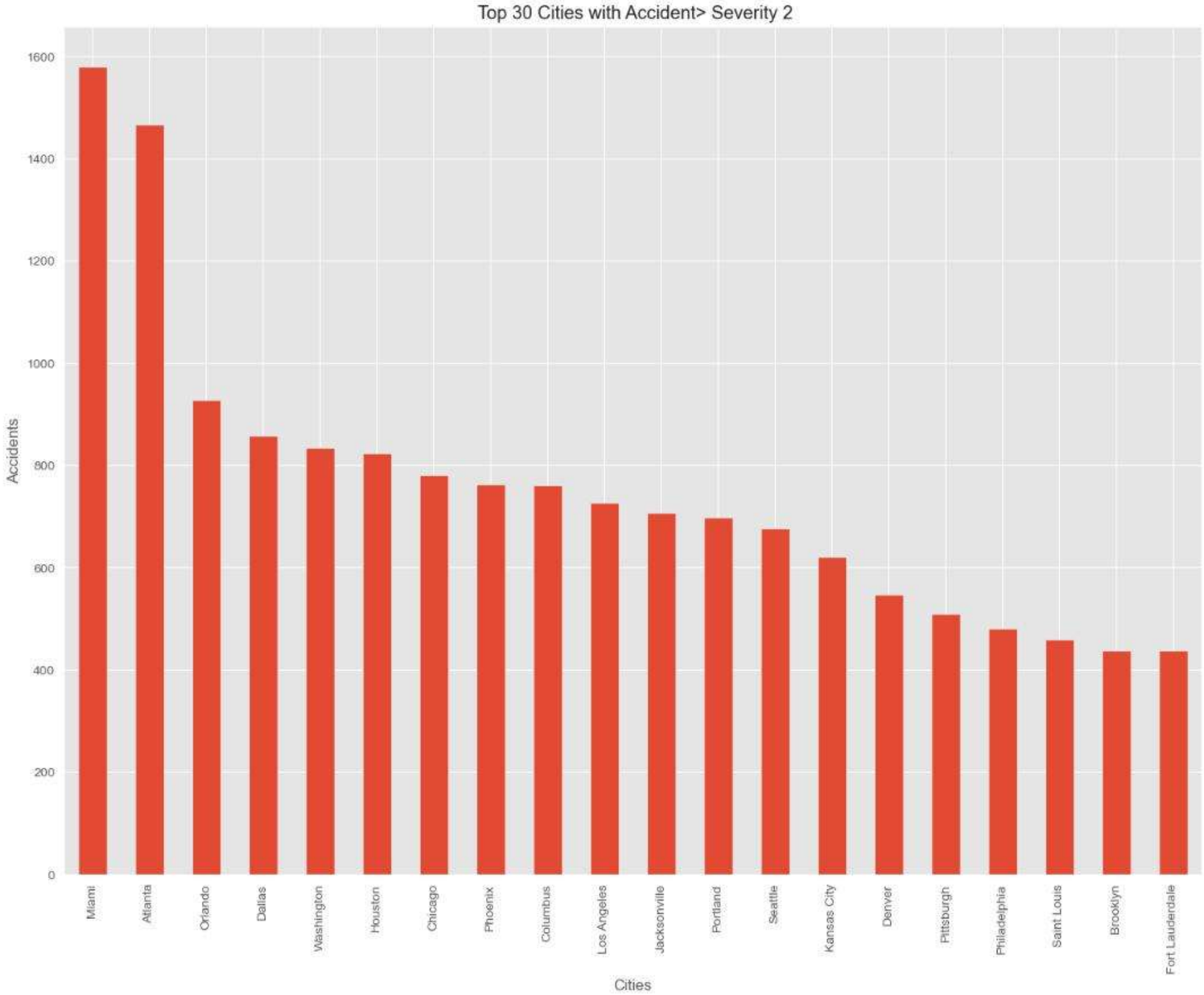
In [427...
```
df_sivr = df[df['Severity'] == 4]
```

In [428...
```
 df_sivr.City.value_counts()
```

Out[428...
```
Miami           1580
Atlanta         1467
Orlando          927
Dallas           858
Washington       835
                ...
Bladen             1
Lambsburg          1
Blackwater         1
Society Hill       1
Bridal Veil        1
Name: City, Length: 8660, dtype: int64
```

In [429...
```
df_sivr.City.value_counts().head(20).plot(kind='bar').set(title=' Top 30 Cities with Accident> Severity 2')
plt.xlabel('Cities')
plt.ylabel('Accidents')
```

Out[429...
```
Text(0, 0.5, 'Accidents')
```


Top 30 Cities with Accident> Severity 2
Accidents
0
200
400
600
800
1000
1200
1400
1600
Miami
Atlanta
Orlando
Dallas
Washington
Houston
Chicago
Phoenix
Columbus
Los Angeles
Jacksonville
Portland
Seattle
Kansas City
Denver
Pittsburgh
Philadelphia
Saint Louis
Brooklyn
Fort Lauderdale
Cities

In [431…

```
column_name = "Visibility(mi)"
column_data = df[column_name]
column_data
```

Out[431…

```
0          10.0
1          10.0
2          10.0
3          10.0
4          10.0
           ...
2845337    10.0
2845338    10.0
2845339    10.0
2845340    10.0
2845341     7.0
Name: Visibility(mi), Length: 2845342, dtype: float64
```

In [432…

```
column_data.unique()
```

Out[432…

```
array([1.00e+01, 3.00e+00, 5.00e-01, 1.80e+00, 1.00e+00, 6.00e+00,
       8.00e+00, 2.00e+00, 1.50e+00, 1.20e+00, 5.00e+00, 2.50e+00,
       4.00e+00, 8.00e-01, 7.00e+00, 9.00e+00,      nan, 7.50e-01,
       2.00e+01, 2.50e-01, 2.00e-01, 1.50e+01, 3.00e+01, 5.50e+00,
       1.30e+01, 6.00e+01, 5.00e+01, 3.50e+01, 2.50e+01, 1.00e-01,
       0.00e+00, 1.10e+01, 7.00e+01, 1.20e+01, 4.00e+01, 6.20e+00,
       1.11e+02, 2.20e+00, 3.50e+00, 1.90e+01, 1.05e+01, 4.50e+01,
       4.00e-01, 7.00e-01, 4.20e+00, 8.00e+01, 5.40e+01, 2.80e+00,
       6.00e-01, 1.10e+00, 1.60e+00, 9.00e-01, 1.20e-01, 3.80e-01,
       6.30e-01, 8.80e-01, 1.00e+02, 6.00e-02, 1.90e-01, 7.50e+01,
       1.40e+02, 2.30e+01, 9.00e+01, 1.60e+01, 1.20e+02, 9.90e-01,
       4.70e+01, 2.20e+01, 3.40e+01, 1.10e+02, 1.30e+02, 1.40e+00,
       1.90e+00, 6.30e+01, 4.30e+01, 3.60e+01, 1.40e+01])
```

In [433…

```
column_data.value_counts()
```

Out[433…

```
10.0    2230276
7.0       79649
9.0       68817
8.0       55955
5.0       53933
          ...
6.2           1
63.0          1
43.0          1
36.0          1
19.0          1
Name: Visibility(mi), Length: 76, dtype: int64
```

In [434…

```
df.groupby('Visibility(mi)').sum()
```

Out[434…

Visibility(mi)	Severity	Start_Lat	Start_Lng	End_Lat	End_Lng	Distance(mi)	Number	Temperature(F)	Wind_Chill(F)	Humidity(%)	…	G
0.00	6740	121696.002286	-345542.648775	121693.867970	-345541.623890	2640.523	16962088.0	145393.1	135920.9	304329.0	…	
0.06	241	4291.554439	-11789.981693	4291.530405	-11789.851954	56.396	306111.0	5744.0	5675.0	10918.0	…	
0.10	656	10582.519110	-24852.014539	10582.352133	-24851.338424	191.842	759783.0	11488.8	2386.2	25478.0	…	
0.12	1507	27008.606095	-75594.326261	27008.576841	-75595.571476	460.346	2439137.0	34239.0	33405.0	69741.0	…	
0.19	24	440.142295	-1435.345402	440.124071	-1435.360854	3.913	36620.0	554.0	520.0	1187.0	…	
…	…	…	…	…	…	…	…	…	…	…	…	
110.00	2	44.370840	-71.297090	44.364250	-71.329370	1.658	585.0	32.0	26.0	30.0	…	
111.00	3	32.997580	-96.671150	33.004110	-96.714290	2.540	0.0	93.2	0.0	49.0	…	
120.00	12	177.608659	-284.415077	177.582958	-284.468103	6.753	2725.0	37.0	-56.0	104.0	…	
130.00	2	44.269590	-71.472460	44.268620	-71.468810	0.193	1904.0	21.0	2.0	33.0	…	
140.00	4	72.738212	-238.714317	72.736036	-238.716267	0.290	2580.0	114.2	66.0	95.0	…	

76 rows × 26 columns

In [435…

```
df_vis = df[df['Visibility(mi)'] < 5]
```

In [436…

```
df_vis.City.value_counts()
```

Out[436…

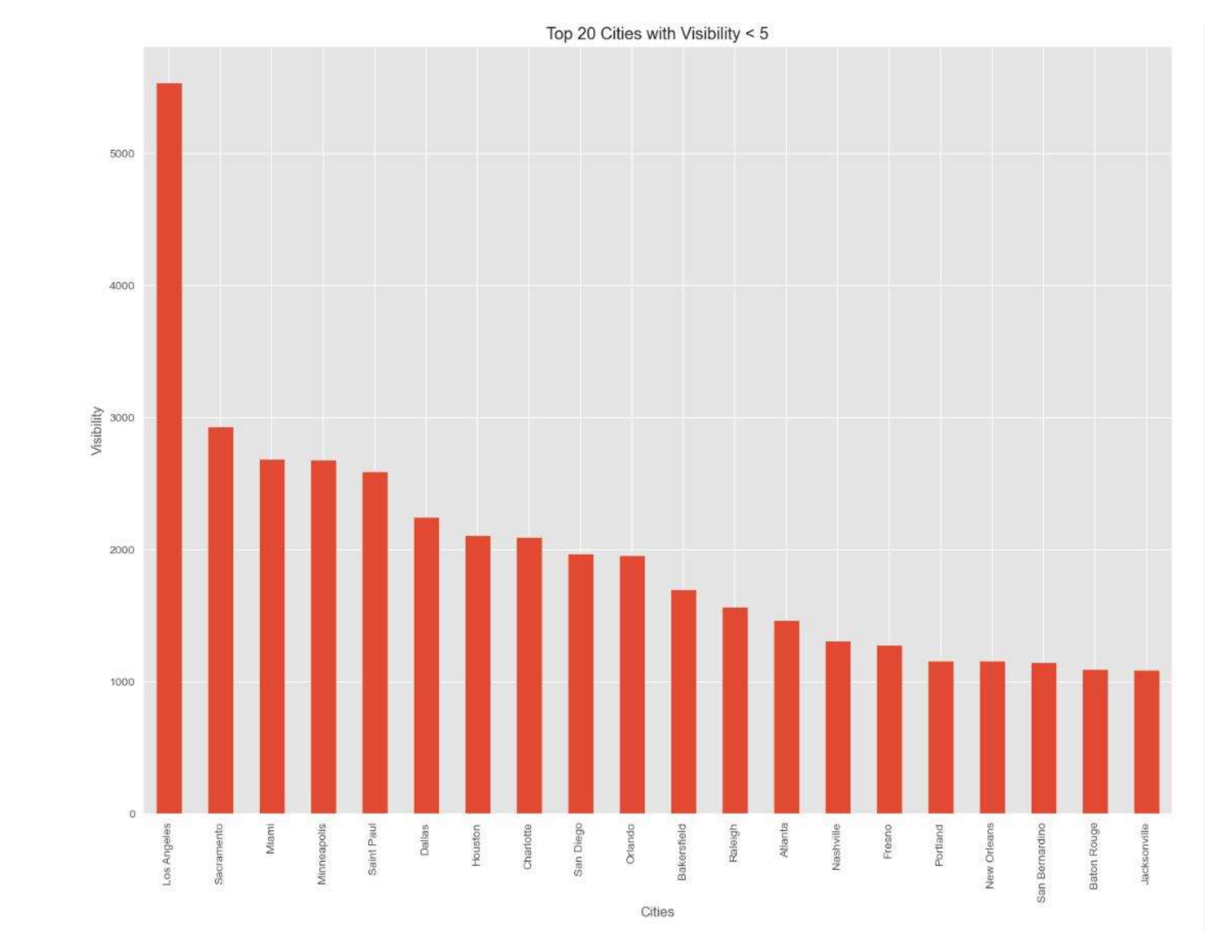
```
Los Angeles         5534
Sacramento          2931
Miami               2684
Minneapolis         2680
Saint Paul          2592
                    ...
Oceano                 1
Garnet Valley          1
Hurdsfield             1
Farmington Hills       1
Tubac                  1
Name: City, Length: 7073, dtype: int64
```

In [437…

```
df_vis.City.value_counts().head(20).plot(kind='bar').set(title=' Top 20 Cities with Visibility < 5 ')
plt.xlabel('Cities')
plt.ylabel('Visibility')
```

Out[437…

```
Text(0, 0.5, 'Visibility')
```


Top 20 Cities with Visibility < 5
Visibility
0
1000
2000
3000
4000
5000
Los Angeles
Sacramento
Miami
Minneapolis
Saint Paul
Dallas
Houston
Charlotte
San Diego
Orlando
Bakersfield
Raleigh
Atlanta
Nashville
Fresno
Portland
New Orleans
San Bernardino
Baton Rouge
Jacksonville
Cities

lets analyize the start time data

In [438…
```
df.Start_Time
```

Out[438…
```
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
                  ...
2845337    2019-08-23 18:03:25
2845338    2019-08-23 19:11:30
2845339    2019-08-23 19:00:21
2845340    2019-08-23 19:00:21
2845341    2019-08-23 18:52:06
Name: Start_Time, Length: 2845342, dtype: object
```

In [439…
```
df.Start_Time = pd.to_datetime(df.Start_Time)
```

In [440…
```
df.Start_Time[0]
```

Out[440…
```
Timestamp('2016-02-08 00:37:08')
```

In [441…
```
df.Start_Time.dt.hour
```

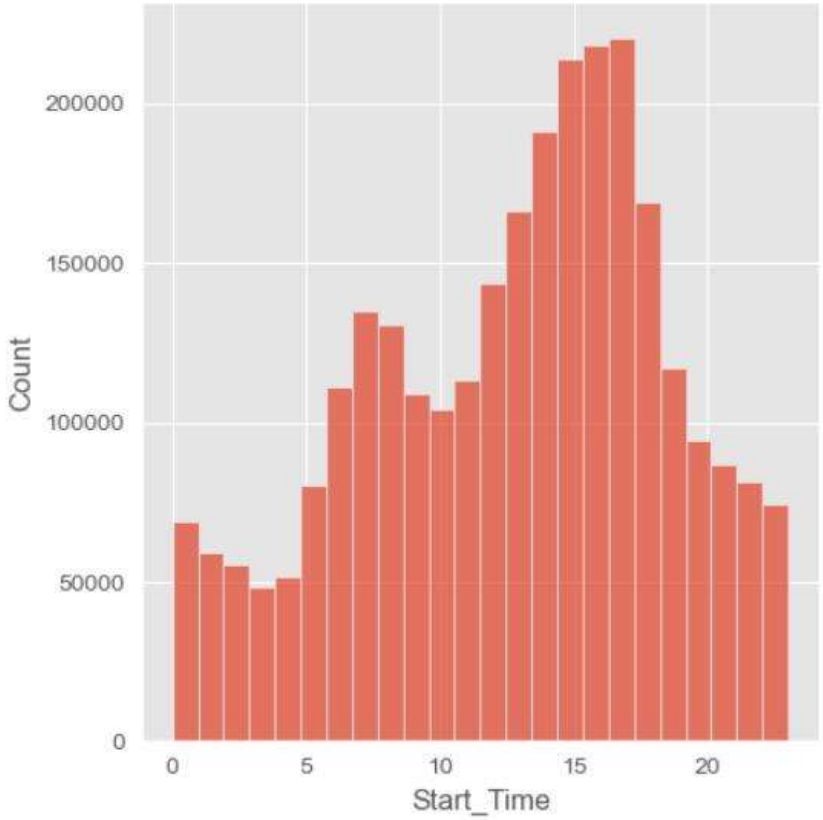
Out[441…
```
0          0
1          5
2          6
3          6
4          7
          ..
2845337    18
2845338    19
2845339    19
2845340    19
2845341    18
Name: Start_Time, Length: 2845342, dtype: int64
```

In [442…

```
sns.displot(df.Start_Time.dt.hour , bins = 24 , kde = False)
```

Out[442…

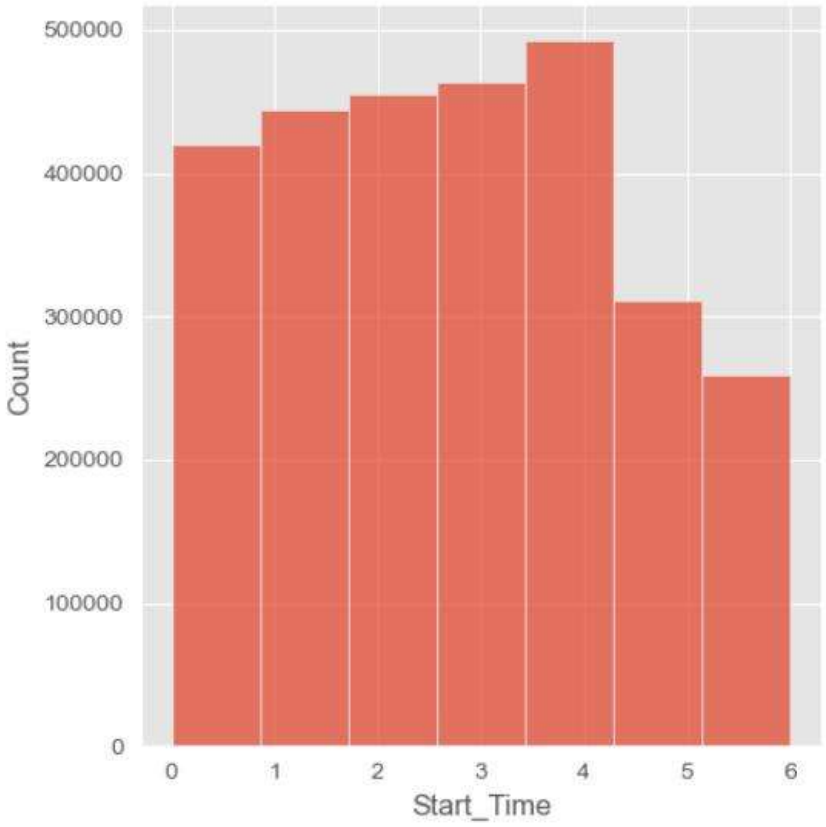
```
<seaborn.axisgrid.FacetGrid at 0x153c56816d0>
```



In [443…

```
sns.displot(df.Start_Time.dt.dayofweek , bins = 7 , kde = False)
```

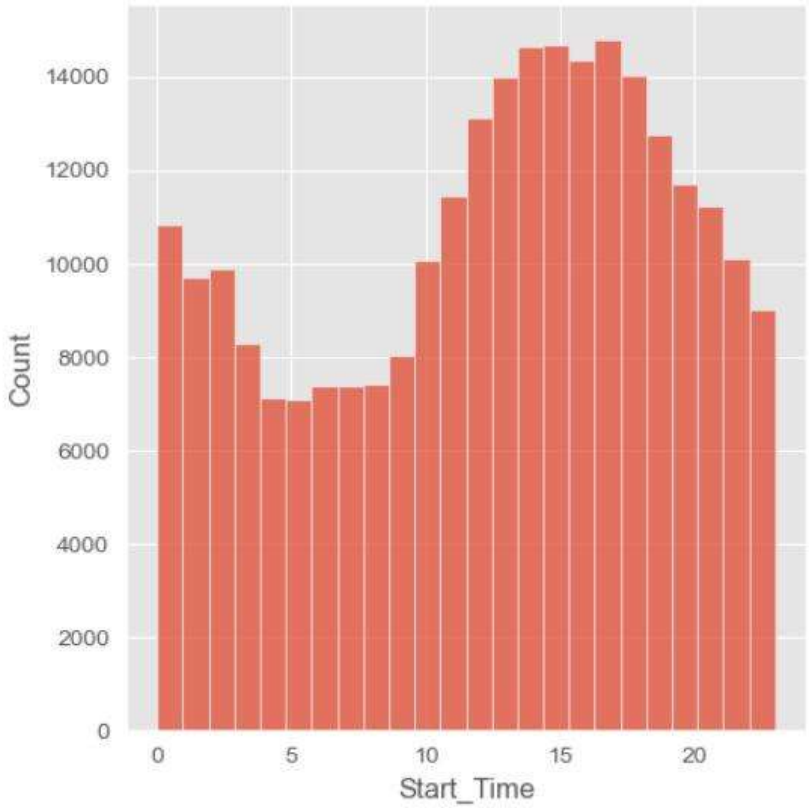
Out[443… <seaborn.axisgrid.FacetGrid at 0x153c566bc40>



In [444…
```
sunday_start_time = df.Start_Time[df.Start_Time.dt.dayofweek==6]
```

In [445…
```
sns.displot(sunday_start_time.dt.hour , bins = 24 , kde = False)
```

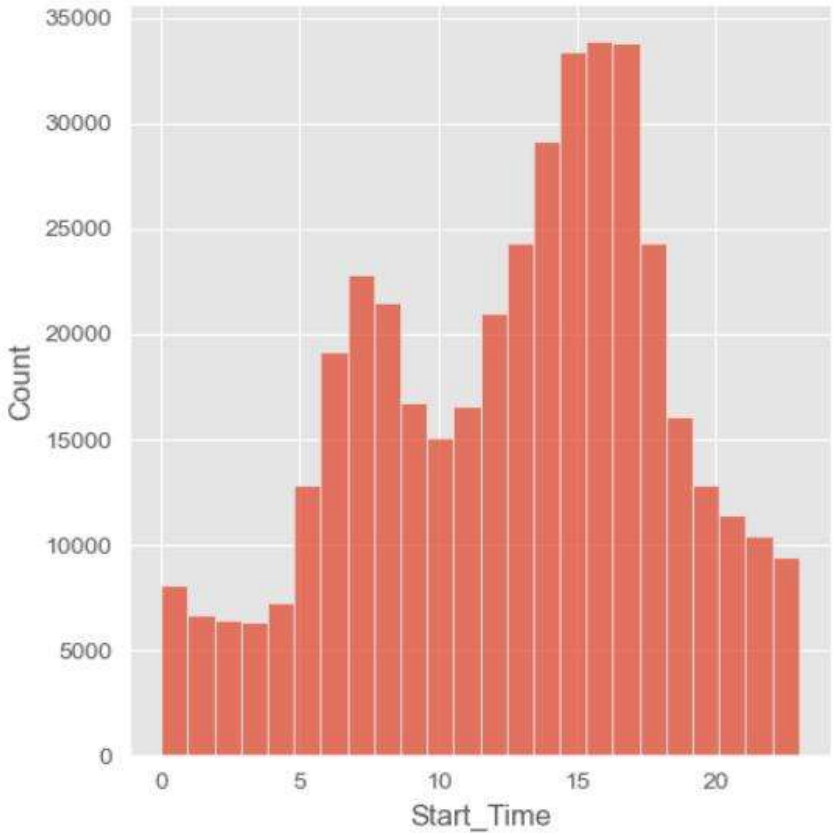
Out[445… <seaborn.axisgrid.FacetGrid at 0x153c5610250>



In [446…

```
mondy_start_time = df.Start_Time[df.Start_Time.dt.dayofweek==0]
sns.displot(mondy_start_time.dt.hour , bins = 24 , kde = False)
```

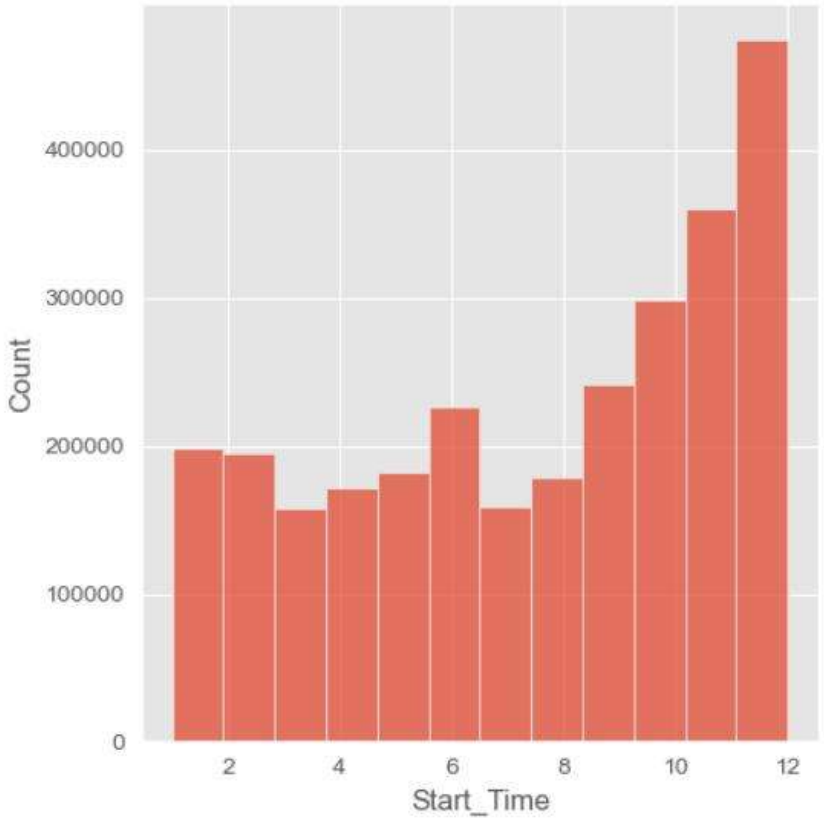
Out[446… <seaborn.axisgrid.FacetGrid at 0x1560c13ba90>



In [447…

```
sns.displot(df.Start_Time.dt.month , bins = 12 , kde = False)
```

Out[447… <seaborn.axisgrid.FacetGrid at 0x153c5559b20>

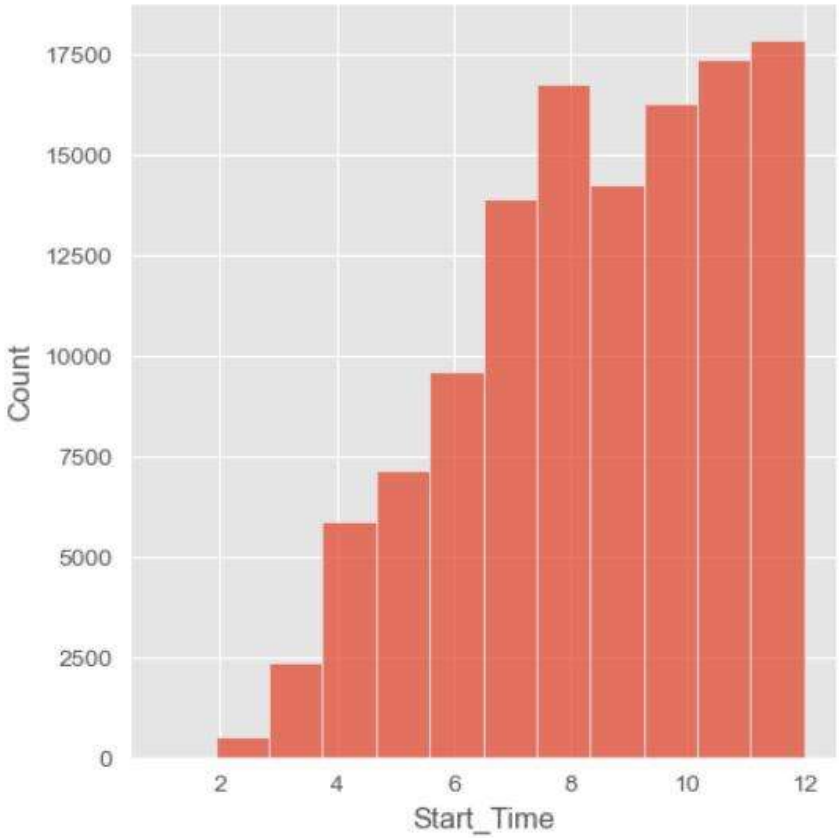


In [448…

```
df_yr = df[df.Start_Time.dt.year == 2016]

sns.displot(df_yr.Start_Time.dt.month , bins = 12 , kde = False)
```

Out[448… <seaborn.axisgrid.FacetGrid at 0x153c584f580>



In [ ]:

In [449…

```
df['Hour'] = pd.to_datetime(df['Start_Time']).dt.hour
df['Minute'] = pd.to_datetime(df['Start_Time']).dt.minute
df['Count'] = 1
df.head(10)
```

Out[449…

	ID	Severity	Start_Time	End_Time	Start_Lat	Start_Lng	End_Lat	End_Lng	Distance(mi)	Description	...	Traffic_Calming	Traffic_Signal	Turning_Loop
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...	...	False	False	False
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.	...	False	False	False
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.	...	False	False	False
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.	...	False	False	False
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.	...	False	False	False
5	A-6	2	2016-02-08 08:16:57	2016-02-08 14:16:57	39.063240	-84.032430	39.067310	-84.058510	1.427	At Dela Palma Rd - Accident.	...	False	True	False
6	A-7	2	2016-02-08 08:15:41	2016-02-08 14:15:41	39.775650	-84.186030	39.772750	-84.188050	0.227	At OH-4/Exit 54 - Accident.	...	False	False	False
7	A-8	2	2016-02-08 11:51:46	2016-02-08 17:51:46	41.375310	-81.820170	41.367860	-81.821740	0.521	At Bagley Rd/Exit 235 - Accident.	...	False	False	False

In [451…
```
df_hr = df[df['Hour'] == 17]
```

In [452…
```
df_hr.City.value_counts()
```

Out[452…
```
Miami               8059
Orlando             5687
Los Angeles         4833
Houston             3466
Sacramento          3204
                    ...
Left Hand              1
Clara City             1
Milton-Freewater       1
Longboat Key           1
Jamieson               1
Name: City, Length: 6752, dtype: int64
```

In [ ]:
```

```

In [453…
```
hour_counts = df['Hour'].value_counts()  #-------- refered this method using the stack over flow

# find the hour with the most accidents
most_accidents_hour = hour_counts.idxmax()

# display the hour with the most accidents
print("The hour with the most accidents is:", most_accidents_hour)
```

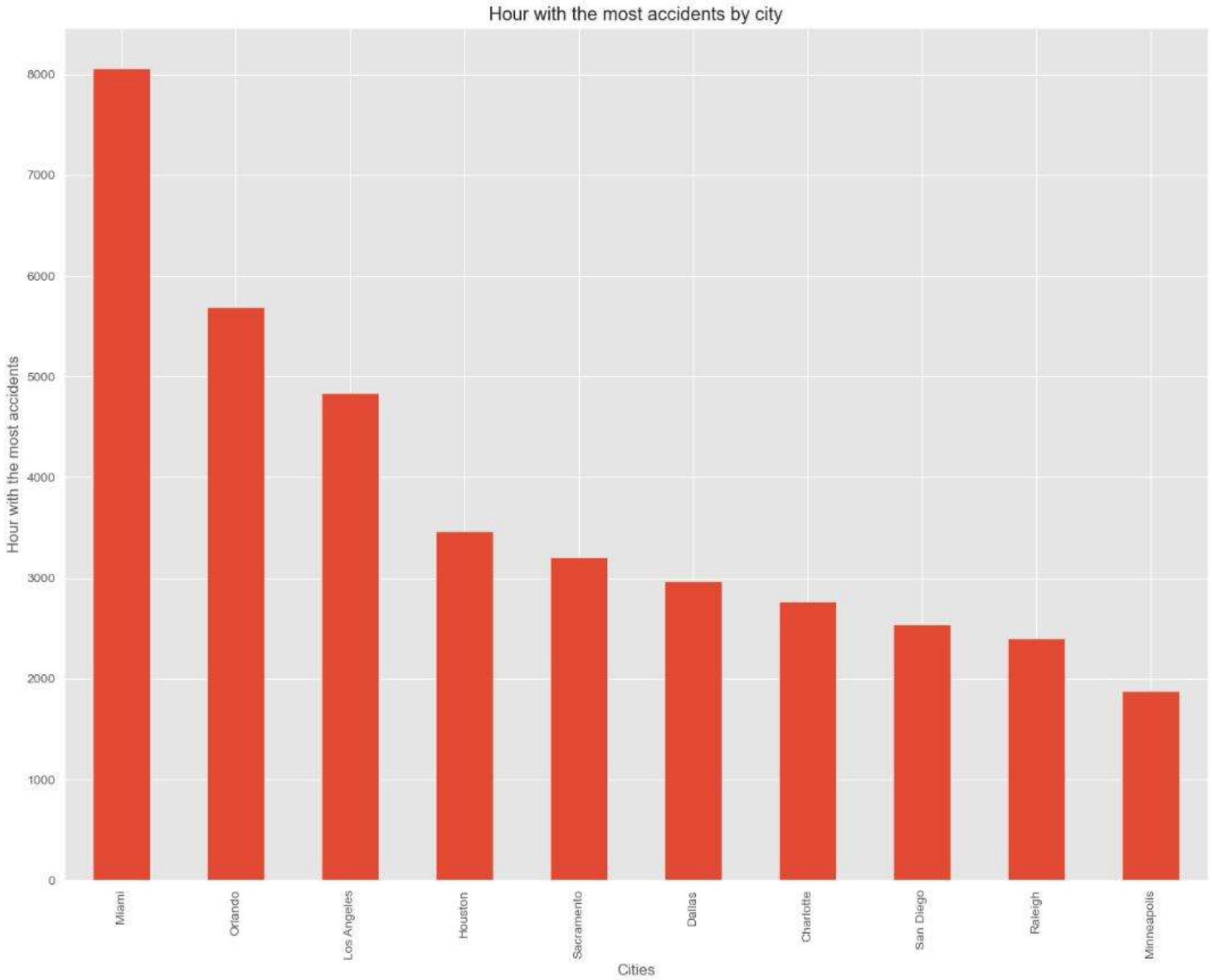
```
The hour with the most accidents is: 17
```

In [454…
```
hour_counts = df['Hour'].value_counts()
most_accidents_hour = hour_counts.idxmax()
most_accidents_hour
```

Out[454… 17

In [455…
```
df_hr.City.value_counts().head(10).plot(kind='bar').set(title=' Hour with the most accidents by city ')
plt.xlabel('Cities')
plt.ylabel('Hour with the most accidents')
```

Out[455… Text(0, 0.5, 'Hour with the most accidents')



In [456...

```
keys = [pair for pair, df in df.groupby(['Hour'])]

plt.plot(keys, df.groupby(['Hour']).count()['Count'])
plt.xticks(keys)
plt.xlabel('Hour in accident occurs')
plt.show()
```

Lattitude and langitude

In [457…
```
df.Start_Lat
```

Out[457…
```
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 [320…
```
df.Start_Lng
```

Out[320…
```
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
```

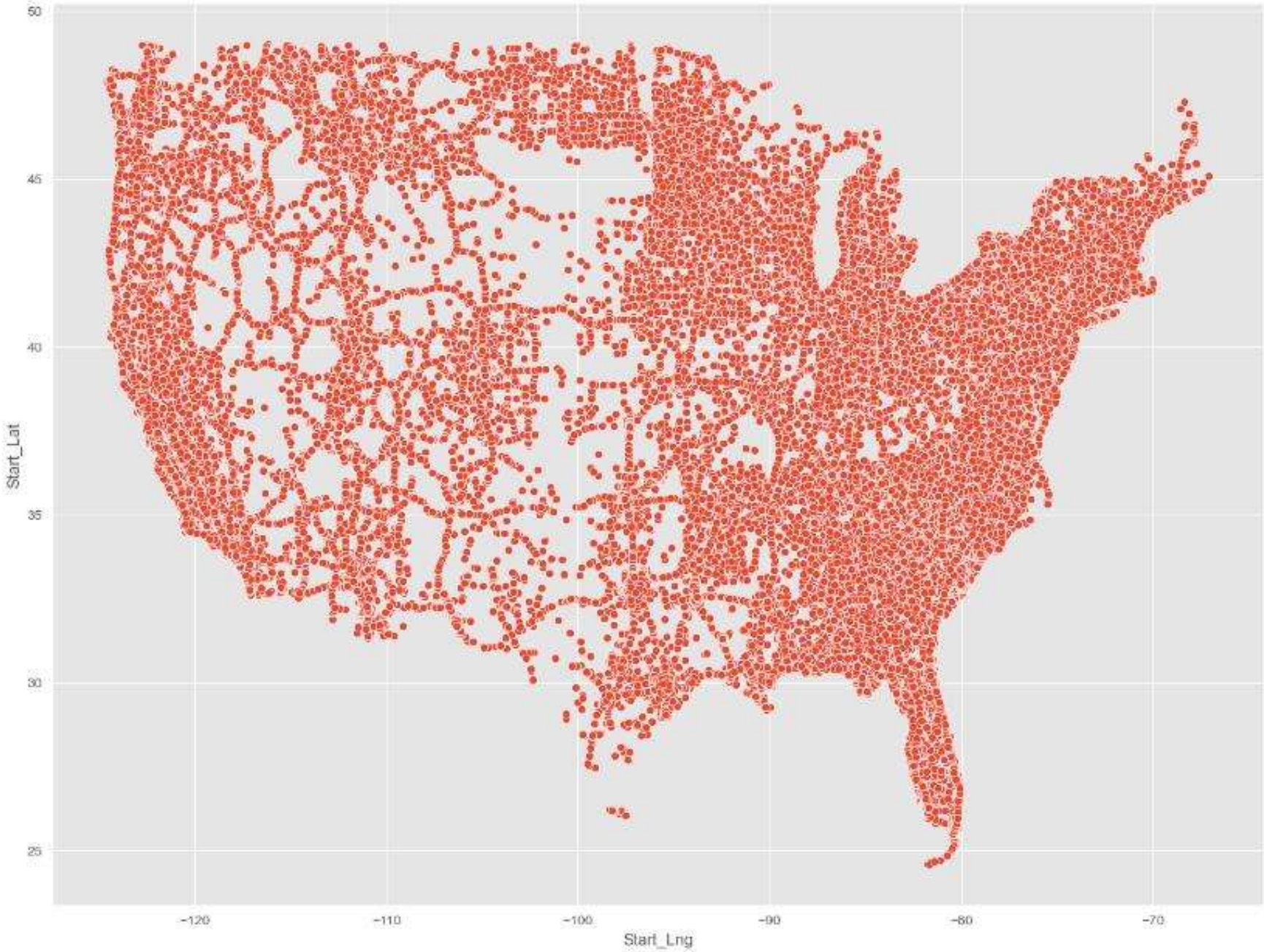
The below graph gives clear explanation of high number of accidents in the coastel areas as the population in the coast are higher

In [322…

```
sns.scatterplot(x= df.Start_Lng , y=df.Start_Lat )
```

Out[322…

```
<AxesSubplot:xlabel='Start_Lng', ylabel='Start_Lat'>
```



Findings:

1. There is no data for the new york in the data set
2. Miami top the list with city having severity of 4
3. los angels top the list for the city with highest numeber of accident for visibility < 10 and also < 5
4. only 4% of the cities has the accident numbers > 1000
5. At 17:00 hours the maximum accidents occur with miami toping the list.
6. Most of the accident occurs during the 3 PM to 6 PM.