[18]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

```
/kaggle/input/us-accidents/US_Accidents_March23.csv
```

1 Import the dataset and load the file

[19]:
```
my_filepath = '../input/us-accidents/US_Accidents_Dec21_updated.csv'
```

[20]:
```
df = pd.read_csv('/kaggle/input/us-accidents/US_Accidents_March23.csv')
df.head()
```

[20]:

	ID	Source	Severity	Start_Time	End_Time	Start_Lat	Start_Lng	End_Lat	End_Lng	Distance(mi)	...	Roundabout
0	A-1	Source2	3	2016-02-08 05:46:00	2016-02-08 11:00:00	39.865147	-84.058723	NaN	NaN	0.01	.	False
1	A-2	Source2	2	2016-02-08 06:07:59	2016-02-08 06:37:59	39.928059	-82.831184	NaN	NaN	0.01	.	False
2	A-3	Source2	2	2016-02-08 06:49:27	2016-02-08 07:19:27	39.063148	-84.032608	NaN	NaN	0.01	.	False
3	A-4	Source2	3	2016-02-08 07:23:34	2016-02-08 07:53:34	39.747753	-84.205582	NaN	NaN	0.01	.	False
4	A-5	Source2	2	2016-02-08 07:39:07	2016-02-08 08:09:07	39.627781	-84.188354	NaN	NaN	0.01	.	False

```
  Station   Stop Traffic_Calming Traffic_Signal Turning_Loop Sunrise_Sunset  \
0   False  False           False          False        False          Night
1   False  False           False          False        False          Night
2   False  False           False           True        False          Night
3   False  False           False          False        False          Night
4   False  False           False           True        False            Day

  Civil_Twilight Nautical_Twilight Astronomical_Twilight
0          Night             Night                 Night
1          Night             Night                   Day
2          Night               Day                   Day
3            Day               Day                   Day
4            Day               Day                   Day

[5 rows x 46 columns]
```

2 Data Prepararion and data cleaning

[21]:
```
df.columns
```

[21]:
```
Index(['ID', 'Source', 'Severity', 'Start_Time', 'End_Time', 'Start_Lat',
       'Start_Lng', 'End_Lat', 'End_Lng', 'Distance(mi)', 'Description',
       'Street', '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')
```

[22]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7728394 entries, 0 to 7728393
Data columns (total 46 columns):
 #   Column                 Dtype
---  ------                 -----
 0   ID                     object
 1   Source                 object
 2   Severity               int64
 3   Start_Time             object
 4   End_Time               object
 5   Start_Lat              float64
```

```
 6   Start_Lng              float64
 7   End_Lat                float64
 8   End_Lng                float64
 9   Distance(mi)           float64
 10  Description            object
 11  Street                 object
 12  City                   object
 13  County                 object
 14  State                  object
 15  Zipcode                object
 16  Country                object
 17  Timezone               object
 18  Airport_Code           object
 19  Weather_Timestamp      object
 20  Temperature(F)         float64
 21  Wind_Chill(F)          float64
 22  Humidity(%)            float64
 23  Pressure(in)           float64
 24  Visibility(mi)         float64
 25  Wind_Direction         object
 26  Wind_Speed(mph)        float64
 27  Precipitation(in)      float64
 28  Weather_Condition      object
 29  Amenity                bool
 30  Bump                   bool
 31  Crossing               bool
 32  Give_Way               bool
 33  Junction               bool
 34  No_Exit                bool
 35  Railway                bool
 36  Roundabout             bool
 37  Station                bool
 38  Stop                   bool
 39  Traffic_Calming        bool
 40  Traffic_Signal         bool
 41  Turning_Loop           bool
 42  Sunrise_Sunset         object
 43  Civil_Twilight         object
 44  Nautical_Twilight      object
 45  Astronomical_Twilight  object
dtypes: bool(13), float64(12), int64(1), object(20)
memory usage: 2.0+ GB
```

[23]:
```
df.describe()
```

[23]:
```
            Severity     Start_Lat     Start_Lng       End_Lat       End_Lng  \
count  7.728394e+06  7.728394e+06  7.728394e+06  4.325632e+06  4.325632e+06
```

```
mean   2.212384e+00  3.620119e+01 -9.470255e+01  3.626183e+01 -9.572557e+01
std    4.875313e-01  5.076079e+00  1.739176e+01  5.272905e+00  1.810793e+01
min    1.000000e+00  2.455480e+01 -1.246238e+02  2.456601e+01 -1.245457e+02
25%    2.000000e+00  3.339963e+01 -1.172194e+02  3.346207e+01 -1.177543e+02
50%    2.000000e+00  3.582397e+01 -8.776662e+01  3.618349e+01 -8.802789e+01
75%    2.000000e+00  4.008496e+01 -8.035368e+01  4.017892e+01 -8.024709e+01
max    4.000000e+00  4.900220e+01 -6.711317e+01  4.907500e+01 -6.710924e+01

       Distance(mi)  Temperature(F)  Wind_Chill(F)   Humidity(%)  \
count  7.728394e+06    7.564541e+06   5.729375e+06  7.554250e+06
mean   5.618423e-01    6.166329e+01   5.825105e+01  6.483104e+01
std    1.776811e+00    1.901365e+01   2.238983e+01  2.282097e+01
min    0.000000e+00   -8.900000e+01  -8.900000e+01  1.000000e+00
25%    0.000000e+00    4.900000e+01   4.300000e+01  4.800000e+01
50%    3.000000e-02    6.400000e+01   6.200000e+01  6.700000e+01
75%    4.640000e-01    7.600000e+01   7.500000e+01  8.400000e+01
max    4.417500e+02    2.070000e+02   2.070000e+02  1.000000e+02

       Pressure(in)  Visibility(mi)  Wind_Speed(mph)  Precipitation(in)
count  7.587715e+06    7.551296e+06     7.157161e+06       5.524808e+06
mean   2.953899e+01    9.090376e+00     7.685490e+00       8.407210e-03
std    1.006190e+00    2.688316e+00     5.424983e+00       1.102246e-01
min    0.000000e+00    0.000000e+00     0.000000e+00       0.000000e+00
25%    2.937000e+01    1.000000e+01     4.600000e+00       0.000000e+00
50%    2.986000e+01    1.000000e+01     7.000000e+00       0.000000e+00
75%    3.003000e+01    1.000000e+01     1.040000e+01       0.000000e+00
max    5.863000e+01    1.400000e+02     1.087000e+03       3.647000e+01
```

[24]:
```
numeric_df = df.select_dtypes(include='number')
numeric_df.columns
```

[24]:
```
Index(['Severity', 'Start_Lat', 'Start_Lng', 'End_Lat', 'End_Lng',
       'Distance(mi)', 'Temperature(F)', 'Wind_Chill(F)', 'Humidity(%)',
       'Pressure(in)', 'Visibility(mi)', 'Wind_Speed(mph)',
       'Precipitation(in)'],
      dtype='object')
```

2.1 Looking for missing values

[25]:
```
missing_percent = df.isna().sum().sort_values(ascending = False)/len(df)
missing_percent
```

[25]:
```
End_Lat                 4.402935e-01
End_Lng                 4.402935e-01
Precipitation(in)       2.851286e-01
Wind_Chill(F)           2.586590e-01
Wind_Speed(mph)         7.391355e-02
```

```
Visibility(mi)           2.291524e-02
Wind_Direction           2.267043e-02
Humidity(%)              2.253301e-02
Weather_Condition        2.244438e-02
Temperature(F)           2.120143e-02
Pressure(in)             1.820288e-02
Weather_Timestamp        1.555666e-02
Nautical_Twilight        3.007869e-03
Civil_Twilight           3.007869e-03
Sunrise_Sunset           3.007869e-03
Astronomical_Twilight    3.007869e-03
Airport_Code             2.928810e-03
Street                   1.406372e-03
Timezone                 1.010300e-03
Zipcode                  2.477876e-04
City                     3.273643e-05
Description              6.469649e-07
Traffic_Signal           0.000000e+00
Roundabout               0.000000e+00
Station                  0.000000e+00
Stop                     0.000000e+00
Traffic_Calming          0.000000e+00
Country                  0.000000e+00
Turning_Loop             0.000000e+00
No_Exit                  0.000000e+00
End_Time                 0.000000e+00
Start_Time               0.000000e+00
Severity                 0.000000e+00
Railway                  0.000000e+00
Crossing                 0.000000e+00
Junction                 0.000000e+00
Give_Way                 0.000000e+00
Bump                     0.000000e+00
Amenity                  0.000000e+00
Start_Lat                0.000000e+00
Start_Lng                0.000000e+00
Distance(mi)             0.000000e+00
Source                   0.000000e+00
County                   0.000000e+00
State                    0.000000e+00
ID                       0.000000e+00
dtype: float64
```

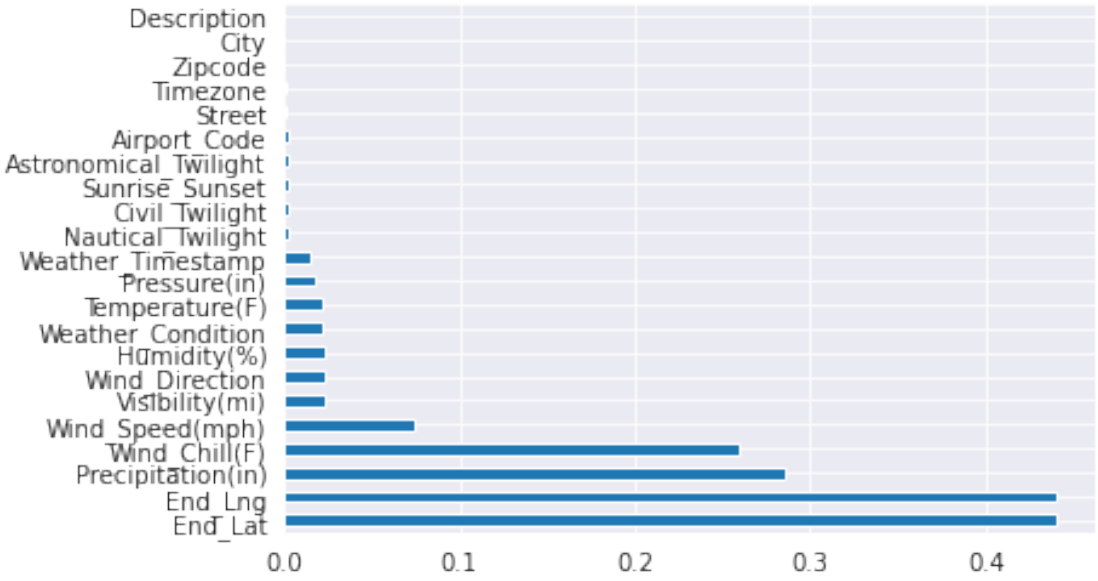
[26]:
```
missing_percent[missing_percent != 0]
```

[26]:
```
End_Lat                  4.402935e-01
End_Lng                  4.402935e-01
```

```
Precipitation(in)        2.851286e-01
Wind_Chill(F)            2.586590e-01
Wind_Speed(mph)          7.391355e-02
Visibility(mi)           2.291524e-02
Wind_Direction           2.267043e-02
Humidity(%)              2.253301e-02
Weather_Condition        2.244438e-02
Temperature(F)           2.120143e-02
Pressure(in)             1.820288e-02
Weather_Timestamp        1.555666e-02
Nautical_Twilight        3.007869e-03
Civil_Twilight           3.007869e-03
Sunrise_Sunset           3.007869e-03
Astronomical_Twilight    3.007869e-03
Airport_Code             2.928810e-03
Street                   1.406372e-03
Timezone                 1.010300e-03
Zipcode                  2.477876e-04
City                     3.273643e-05
Description              6.469649e-07
dtype: float64
```

[27]:
```
missing_percent[missing_percent != 0].plot(kind = 'barh')
```

[27]:
```
<AxesSubplot:>
```



3 Exploratory Analysis and Visualization

3.1 Columns to explore:

- City
- Start Time
- StratLat, StartLong
- Temp
- Wheather Comndition

3.1.1 City

[28]:
```
Cities = df.City.unique()
len(Cities)
```

[28]:
```
13679
```

[29]:
```
cities_by_acc = df['City'].value_counts()
cities_by_acc
```

[29]:
```
Miami                            186917
Houston                          169609
Los Angeles                      156491
Charlotte                        138652
Dallas                           130939
                                  ...
Benkelman                             1
Old Appleton                          1
Wildrose                              1
Mc Nabb                               1
American Fork-Pleasant Grove          1
Name: City, Length: 13678, dtype: int64
```

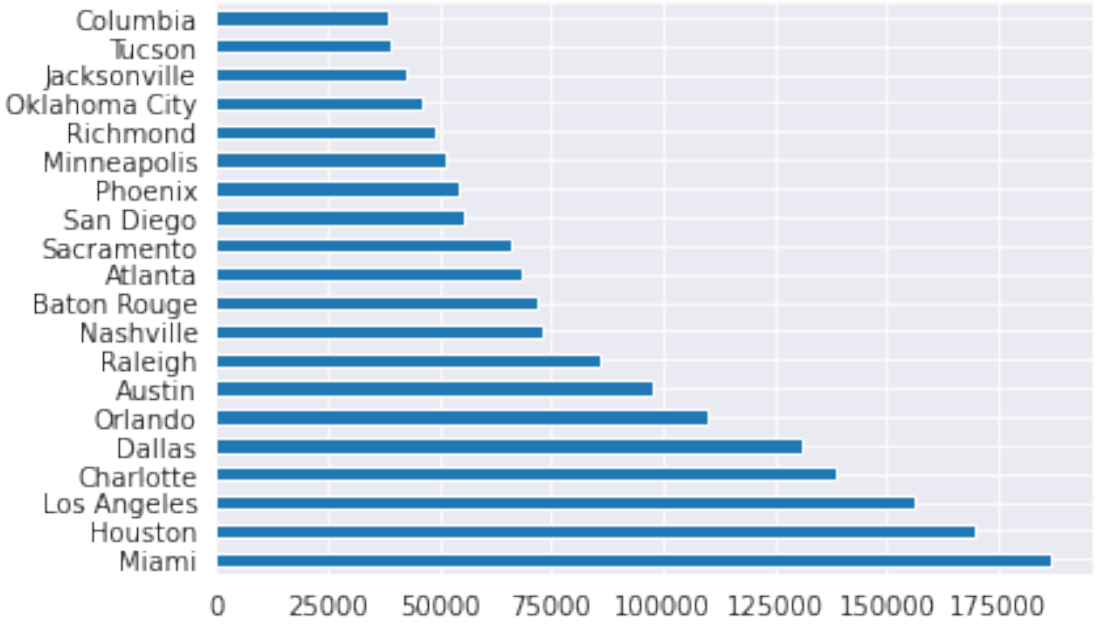
[30]:
```
cities_by_acc[:20]
```

[30]:
```
Miami            186917
Houston          169609
Los Angeles      156491
Charlotte        138652
Dallas           130939
Orlando          109733
Austin            97359
Raleigh           86079
Nashville         72930
Baton Rouge       71588
Atlanta           68186
Sacramento        66264
San Diego         55504
```

```
Phoenix            53974
Minneapolis        51488
Richmond           48845
Oklahoma City      46092
Jacksonville       42447
Tucson             39304
Columbia           38178
Name: City, dtype: int64
```

[31]:
```
cities_by_acc[:20].plot(kind = 'barh')
```

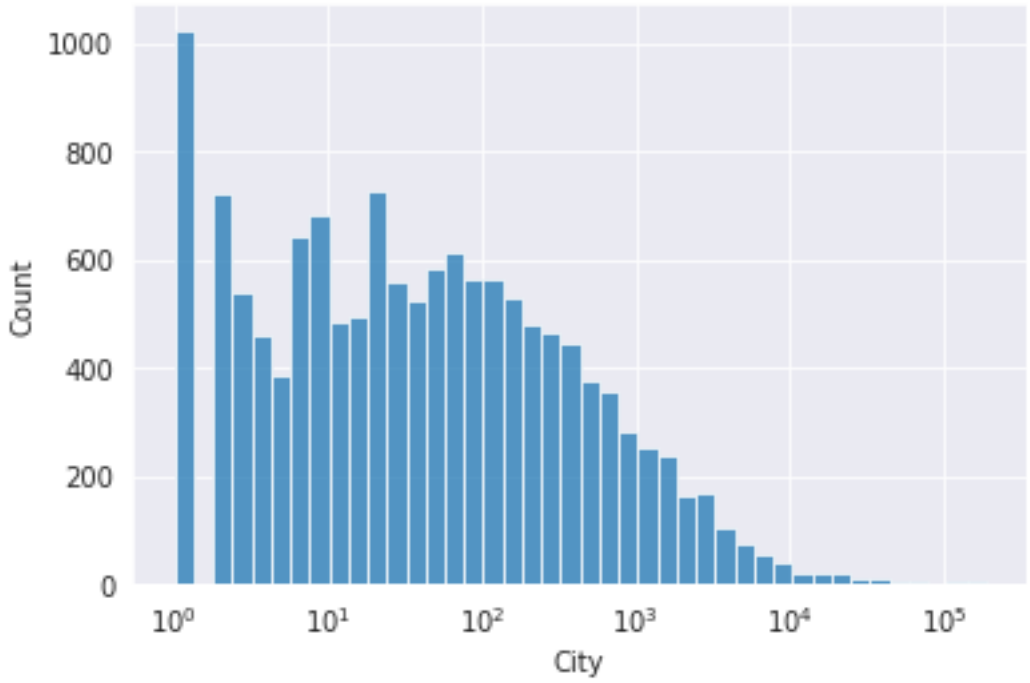
[31]: <AxesSubplot:>



[32]:
```
sns.set_style('darkgrid')
```

[33]:
```
sns.histplot(x=cities_by_acc,log_scale = True)
```

[33]: <AxesSubplot:xlabel='City', ylabel='Count'>



[34]:
```
cities_by_acc[cities_by_acc == 1]
len(cities_by_acc)
```

[34]: 13678

[35]:
```
high_acc_city = cities_by_acc[cities_by_acc >= 1000]
low_acc_city = cities_by_acc[cities_by_acc < 1000]
```

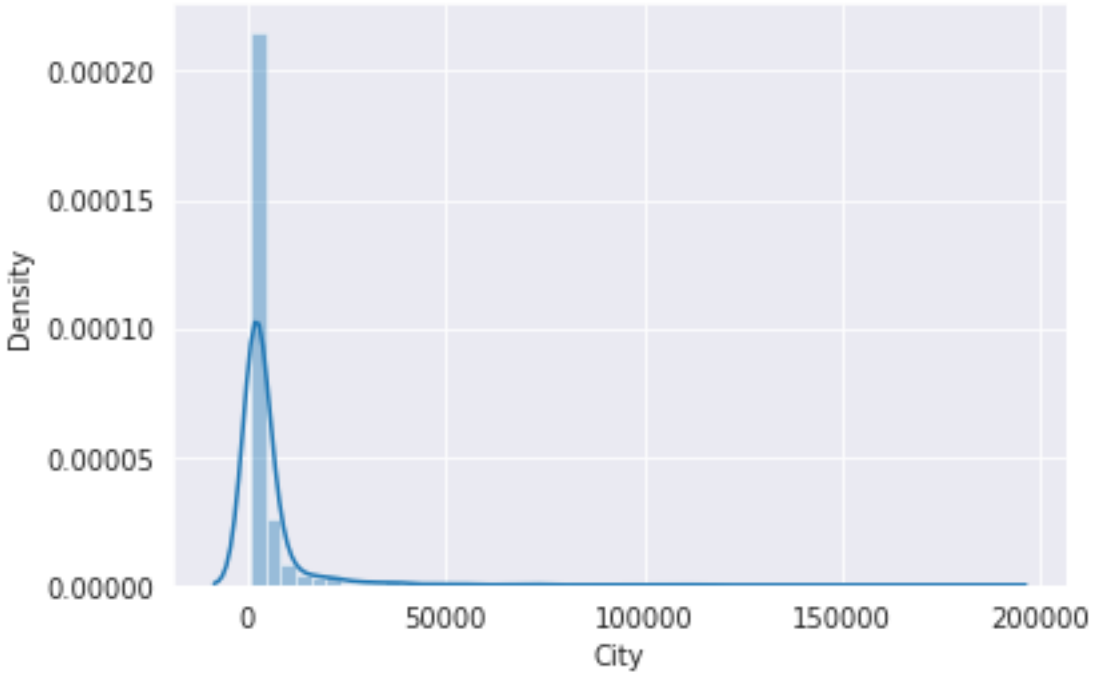
[36]:
```
len(high_acc_city)/len(cities_by_acc)
```

[36]: 0.08904810644831115

[37]:
```
sns.distplot(high_acc_city)
```

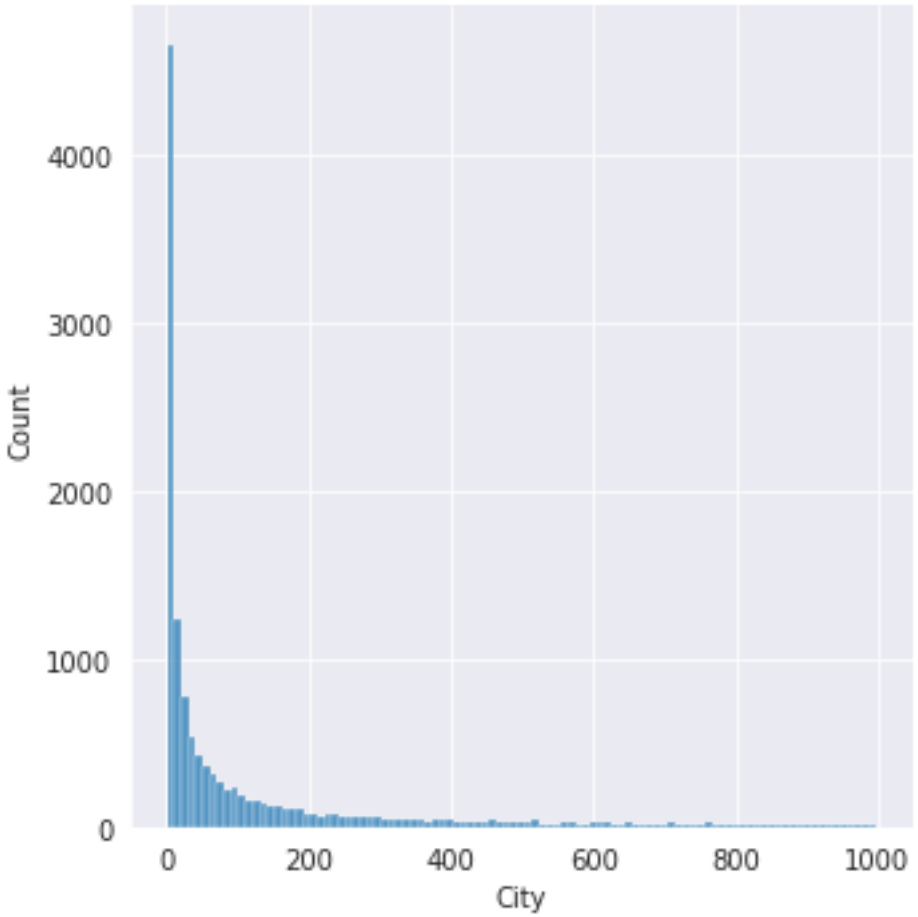
```
/opt/conda/lib/python3.7/site-packages/seaborn/distributions.py:2619:
FutureWarning: `distplot` is a deprecated function and will be removed in a
future version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for
histograms).
  warnings.warn(msg, FutureWarning)
```

[37]: <AxesSubplot:xlabel='City', ylabel='Density'>



[38]:
```
sns.displot(low_acc_city)
```

[38]: <seaborn.axisgrid.FacetGrid at 0x7a1974becf50>



Summary and Conclusion : - No data for New York - Less than 5% cities has more than 1000 accidents - The number of accidents decrease/increases exponentially - Around 12000 cities have reported 1 accident(need to check)

3.1.2 Start time

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

[39]:
```
0          2016-02-08 05:46:00
1          2016-02-08 06:07:59
2          2016-02-08 06:49:27
3          2016-02-08 07:23:34
4          2016-02-08 07:39:07
                  ...
7728389    2019-08-23 18:03:25
7728390    2019-08-23 19:11:30
7728391    2019-08-23 19:00:21
7728392    2019-08-23 19:00:21
```

```
7728393    2019-08-23 18:52:06
Name: Start_Time, Length: 7728394, dtype: object
```

[40]:
```
### Currently the timme data is in object type, converting it to date-time
```

[41]:
```
df.Start_Time=pd.to_datetime(df.Start_Time)
```

[42]:
```
df.Start_Time[0]
```

[42]:
```
Timestamp('2016-02-08 05:46:00')
```

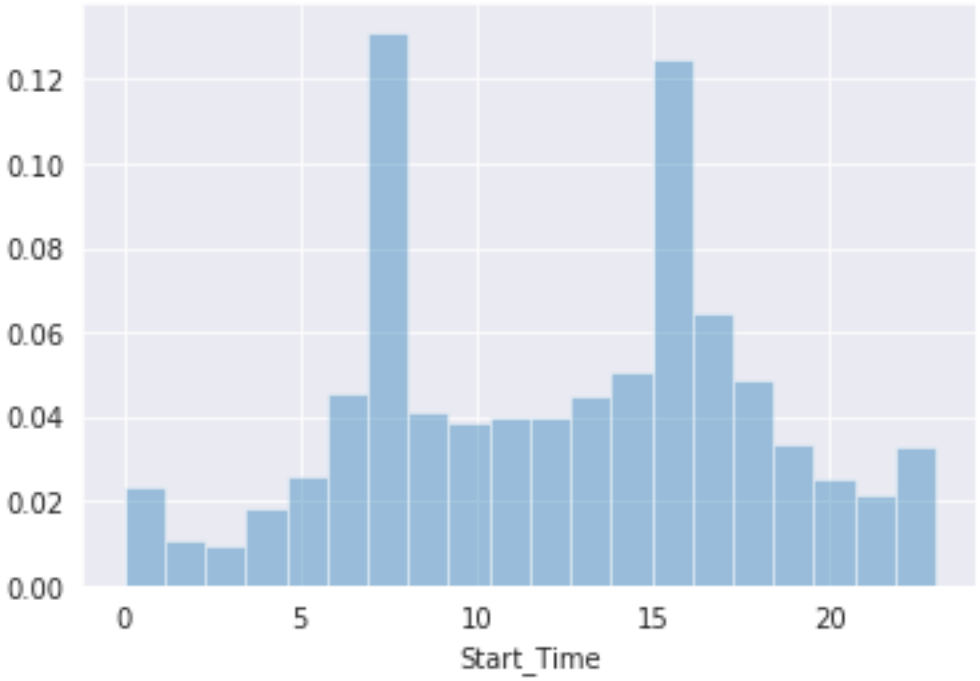
[43]:
```
df.Start_Time.dt.hour
```

[43]:
```
0           5
1           6
2           6
3           7
4           7
           ..
7728389    18
7728390    19
7728391    19
7728392    19
7728393    18
Name: Start_Time, Length: 7728394, dtype: int64
```

[44]:
```
sns.distplot(df.Start_Time.dt.hour,bins=20,norm_hist=True,kde=False)
```

```
/opt/conda/lib/python3.7/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)
```

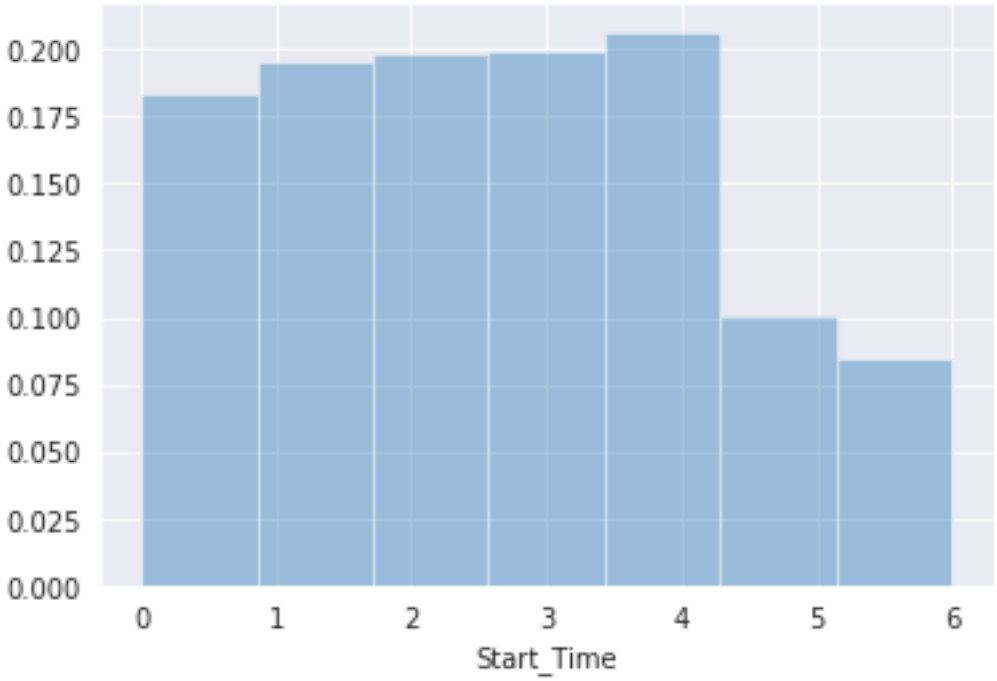
[44]:
```
<AxesSubplot:xlabel='Start_Time'>
```



Observations: 1. A high percentage of accients happen between 6 am to 10 am (probably people leaving for work in a hurry). 2. Next high percentage of accidents happen betwwen 3 pm to 6 pm.

[45]:
```
sns.distplot(df.Start_Time.dt.dayofweek,bins=7,norm_hist=True,kde=False)
```

[45]:
```
<AxesSubplot:xlabel='Start_Time'>
```

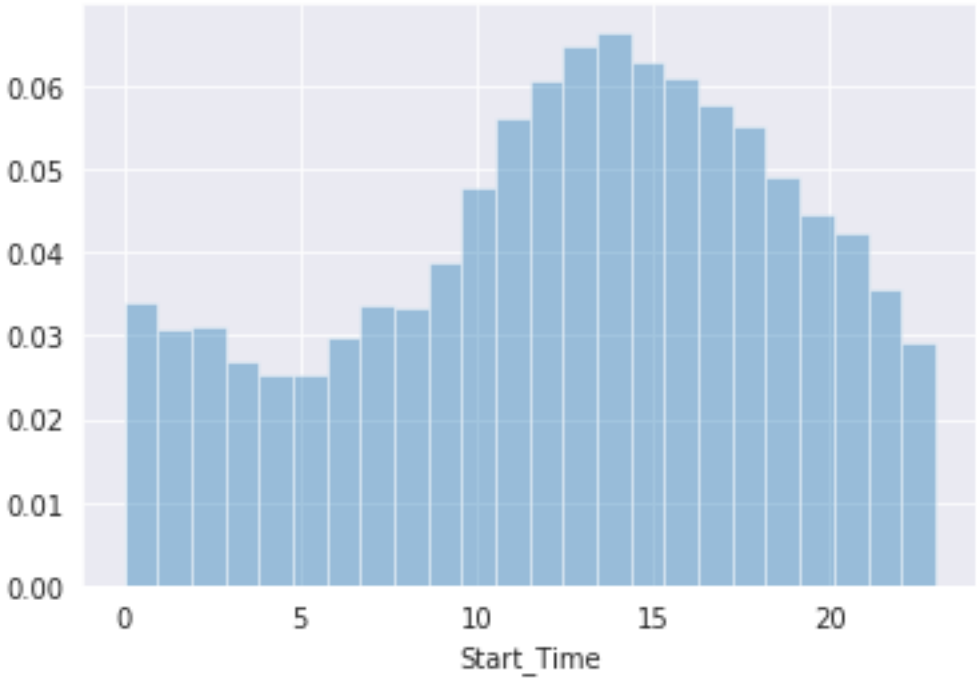


3.1.3 Is the distribution of accidents same in weekdays and weekends?

[46]:
```
sundays_start_time=df.Start_Time[df.Start_Time.dt.dayofweek == 6]
```

[47]:
```
sns.distplot(sundays_start_time.dt.hour,bins=24,norm_hist=True,kde=False)
```

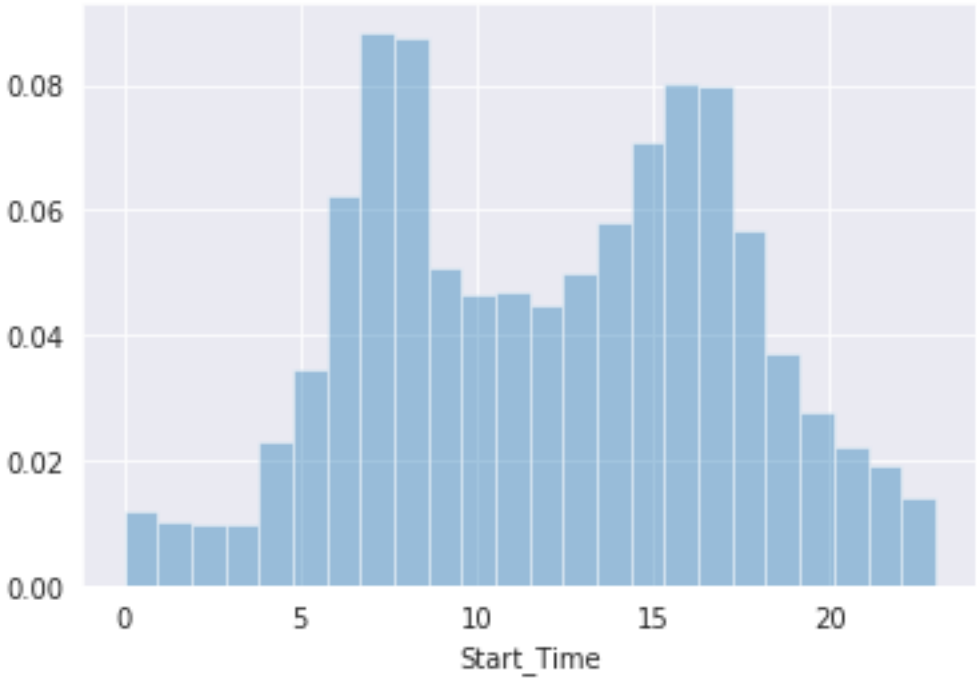
[47]: <AxesSubplot:xlabel='Start_Time'>



[48]:
```
mondays_start_time=df.Start_Time[df.Start_Time.dt.dayofweek == 0]
```

[49]:
```
sns.distplot(mondays_start_time.dt.hour,bins=24,norm_hist=True,kde=False)
```

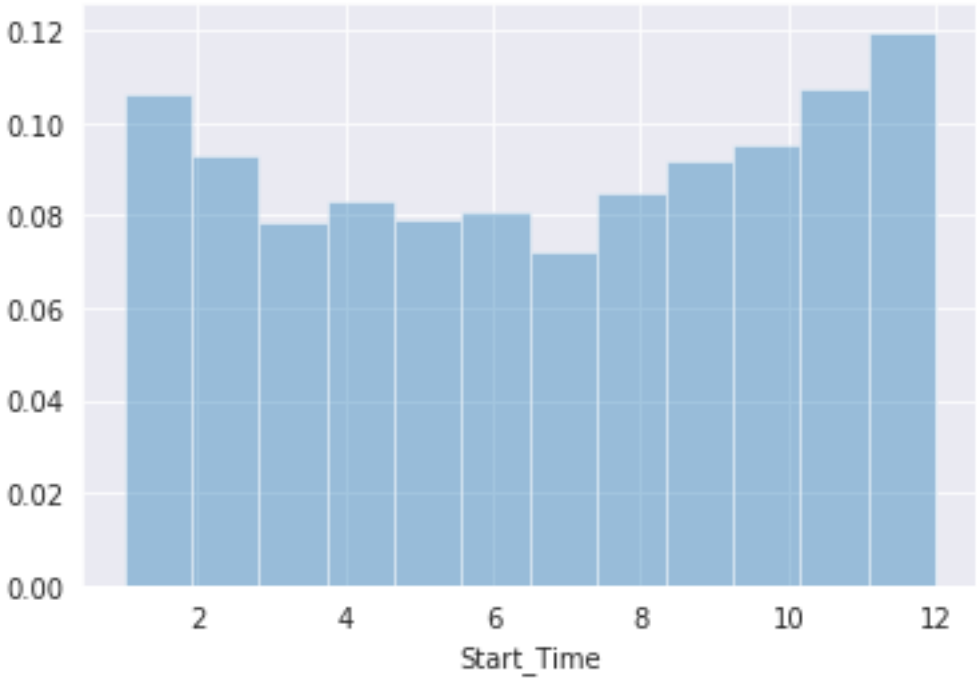
[49]: <AxesSubplot:xlabel='Start_Time'>



Observations: The distribution of the time for most of the acidents during weekdays and weekends vary differently. 1. In weekdays most frequent acidents occur between 6 am to 10 am and 6 pm and 8 pm. 2. In weekends most frequent acidents occur between 10 am to 8 pm.

[50]:
```
sns.distplot(df.Start_Time.dt.month,bins=12,norm_hist=True,kde=False)
```

[50]:
```
<AxesSubplot:xlabel='Start_Time'>
```



Observation : The frequency of the accidents seem to increase towards the end of the year.

3.2 StartLat And StartLong

[51]:
```
df.Start_Lat
```

[51]:
```
0          39.865147
1          39.928059
2          39.063148
3          39.747753
4          39.627781
               ...
7728389    34.002480
7728390    32.766960
7728391    33.775450
7728392    33.992460
7728393    34.133930
Name: Start_Lat, Length: 7728394, dtype: float64
```

[52]:
```
df.Start_Lng
```

[52]:
```
0          -84.058723
1          -82.831184
2          -84.032608
3          -84.205582
```

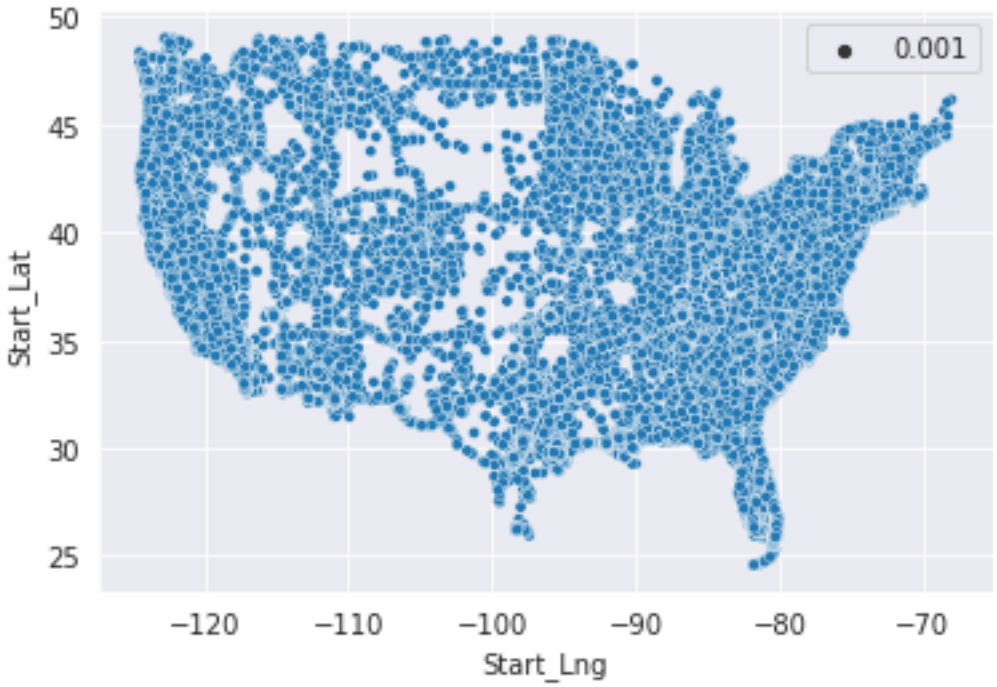
```
4          -84.188354
              …
7728389   -117.379360
7728390   -117.148060
7728391   -117.847790
7728392   -118.403020
7728393   -117.230920
Name: Start_Lng, Length: 7728394, dtype: float64
```

Since the datframe is very big, so we will use a prtion of the sample.

[53]:
```
df_10percent=df.sample(int(0.1 * len(df)))
```

[54]:
```
sns.scatterplot(x=df_10percent.Start_Lng, y=df_10percent.Start_Lat, size=0.001)
```

[54]:
```
<AxesSubplot:xlabel='Start_Lng', ylabel='Start_Lat'>
```



Questions : 1. Are there more accidents in warmer or colder areas? 2. Which 5 states has most number of accidents? 3. Why New York data is not present although it is the most populated city? 4. Out of top 100 cities, which state they belong to? 5. What time of the day are accidents more frequent? 6. Which days of the week has the most no of accidents? 7. Which month has the most no of accidents? 8. What is the trend of accidents over the years?

[ ]: