us-accidents-analysis

July 29, 2024

1 US Accidents Exploratory Data Analysis

TODO - talk about EDA #### TODO - talk about the dataset(source , what it contains , how it will be useful) ### Kaggle ### Information about accidents ### can useful to preventa accidents ### mention that this does not contain data about New York

[ ]:
```
import opendatasets as od
```

[2]:
```
# download_url='https://www.kaggle.com/datasets/sobhanmoosavi/us-accidents'

# od.download(download_url ,  force=True )
```

[7]:
```
data_file = './us-accidents/US_Accidents_March23.csv'
```

2 Data prepration &cleaning

1. Load the file using pandas
2. Look at some info. about the data & the columns
3. Fix any missing or incorrect values

[8]:
```
import pandas as pd
```

[9]:
```
df = pd.read_csv(data_file)
```

[10]:
```
df
```

[10]:

	ID	Source	Severity	Start_Time	\
0	A-1	Source2	3	2016-02-08 05:46:00	
1	A-2	Source2	2	2016-02-08 06:07:59	
2	A-3	Source2	2	2016-02-08 06:49:27	
3	A-4	Source2	3	2016-02-08 07:23:34	
4	A-5	Source2	2	2016-02-08 07:39:07	
…	…	…	…	…	
7728389	A-7777757	Source1	2	2019-08-23 18:03:25	
7728390	A-7777758	Source1	2	2019-08-23 19:11:30	
7728391	A-7777759	Source1	2	2019-08-23 19:00:21	
7728392	A-7777760	Source1	2	2019-08-23 19:00:21	

```
7728393  A-7777761  Source1         2  2019-08-23 18:52:06

                  End_Time  Start_Lat   Start_Lng   End_Lat    End_Lng  \
0        2016-02-08 11:00:00  39.865147  -84.058723       NaN        NaN
1        2016-02-08 06:37:59  39.928059  -82.831184       NaN        NaN
2        2016-02-08 07:19:27  39.063148  -84.032608       NaN        NaN
3        2016-02-08 07:53:34  39.747753  -84.205582       NaN        NaN
4        2016-02-08 08:09:07  39.627781  -84.188354       NaN        NaN
…                        …        …          …       …          …
7728389  2019-08-23 18:32:01  34.002480 -117.379360  33.99888 -117.37094
7728390  2019-08-23 19:38:23  32.766960 -117.148060  32.76555 -117.15363
7728391  2019-08-23 19:28:49  33.775450 -117.847790  33.77740 -117.85727
7728392  2019-08-23 19:29:42  33.992460 -118.403020  33.98311 -118.39565
7728393  2019-08-23 19:21:31  34.133930 -117.230920  34.13736 -117.23934

         Distance(mi)  … Roundabout Station   Stop Traffic_Calming  \
0               0.010  …      False   False  False           False
1               0.010  …      False   False  False           False
2               0.010  …      False   False  False           False
3               0.010  …      False   False  False           False
4               0.010  …      False   False  False           False
…                   …  …          …       …      …             …
7728389         0.543  …      False   False  False           False
7728390         0.338  …      False   False  False           False
7728391         0.561  …      False   False  False           False
7728392         0.772  …      False   False  False           False
7728393         0.537  …      False   False  False           False

        Traffic_Signal Turning_Loop Sunrise_Sunset Civil_Twilight  \
0                False        False          Night          Night
1                False        False          Night          Night
2                 True        False          Night          Night
3                False        False          Night            Day
4                 True        False            Day            Day
…                    …            …              …              …
7728389          False        False            Day            Day
7728390          False        False            Day            Day
7728391          False        False            Day            Day
7728392          False        False            Day            Day
7728393          False        False            Day            Day

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

```
…                    …                    …
7728389              Day                  Day
7728390              Day                  Day
7728391              Day                  Day
7728392              Day                  Day
7728393              Day                  Day

[7728394 rows x 46 columns]
```

[11]:
```
df.columns
```

[11]:
```
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')
```

[12]:
```
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
```

[13]:
```
df.describe()
```

[13]:

	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
```

[14]:
```
numerics = ['int16' , 'int32' , 'int64' , 'float16' , 'float32' , 'float64']

numeric_df = df.select_dtypes(include=numerics)
len(numeric_df.columns)
```

[14]:
```
13
```

Percantage of missing values per columns

[15]:
```
#checking for null values
missing_percantages = df.isna().sum().sort_values(ascending=False) / len(df)
missing_percantages
```

[15]:
```
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
```

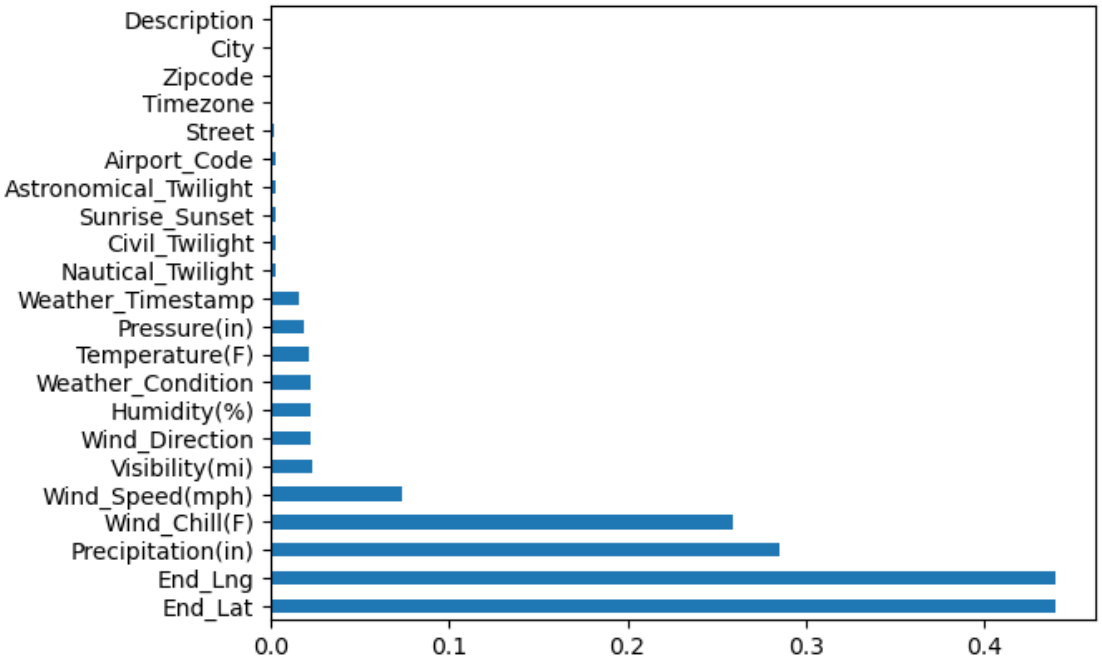
[16]:
```
#removing those columns which doesnt have any null values
missing_percantages[missing_percantages != 0]
```

[16]:
```
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
```

[17]:
```
missing_percantages[missing_percantages != 0].plot(kind='barh')
```

[17]: <Axes: >



Removing columns that doesnt in use

[18]:
```
df.columns
```

[18]:
```
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')
```

3 Exploratory Analysis & Visulization

columns to analyze: 1.City 2.Start Time 3.start lat , start long 4.Temperature 5.weather condition

3.1 Cities

[19]:
```
df.City
```

[19]:
```
0               Dayton
1          Reynoldsburg
2          Williamsburg
3               Dayton
4               Dayton
              ...
7728389       Riverside
7728390       San Diego
7728391          Orange
7728392     Culver City
7728393        Highland
Name: City, Length: 7728394, dtype: object
```

[20]:
```
cities=df.City.unique()
len(cities)
```

[20]:
```
13679
```

[21]:
```
cities__by_accident = df.City.value_counts()
cities__by_accident
```

[21]:
```
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
```

[22]:
```
cities__by_accident[:20]
```

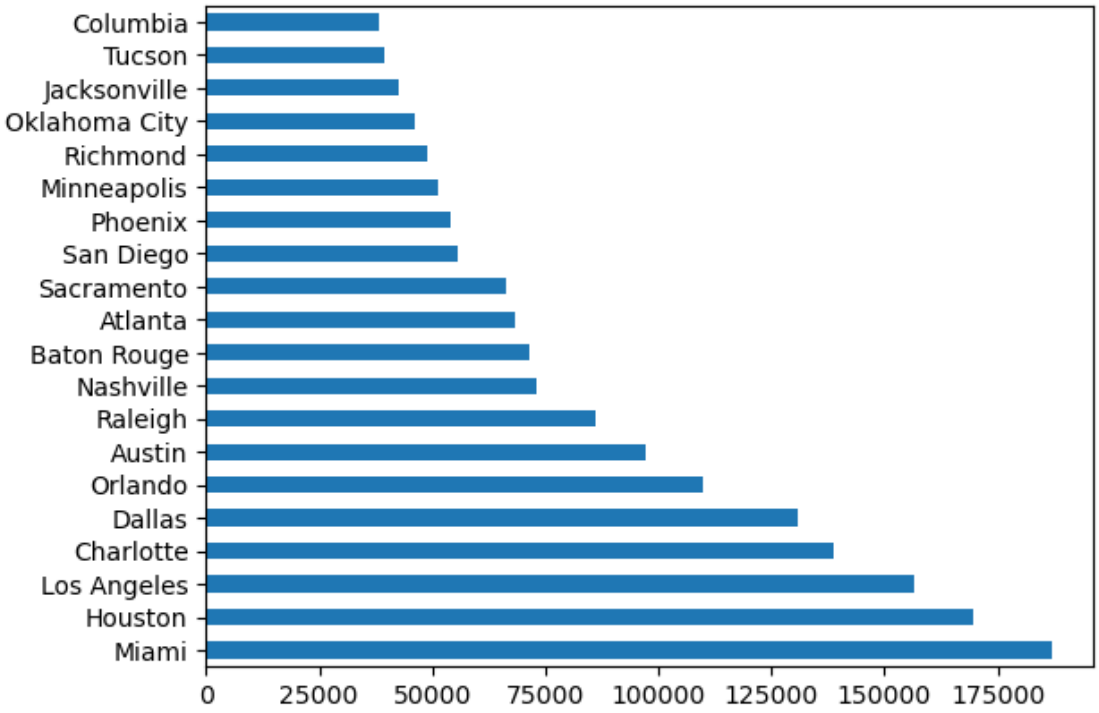
[22]:
```
Miami            186917
Houston          169609
Los Angeles      156491
Charlotte        138652
Dallas           130939
Orlando          109733
Austin            97359
Raleigh           86079
Nashville         72930
Baton Rouge       71588
Atlanta           68186
Sacramento        66264
San Diego         55504
Phoenix           53974
Minneapolis       51488
Richmond          48845
Oklahoma City     46092
Jacksonville      42447
Tucson            39304
Columbia          38178
Name: City, dtype: int64
```

[23]:
```
'NY' in df.State
```

[23]:
```
False
```

[24]:
```
cities__by_accident[:20].plot(kind='barh')
```

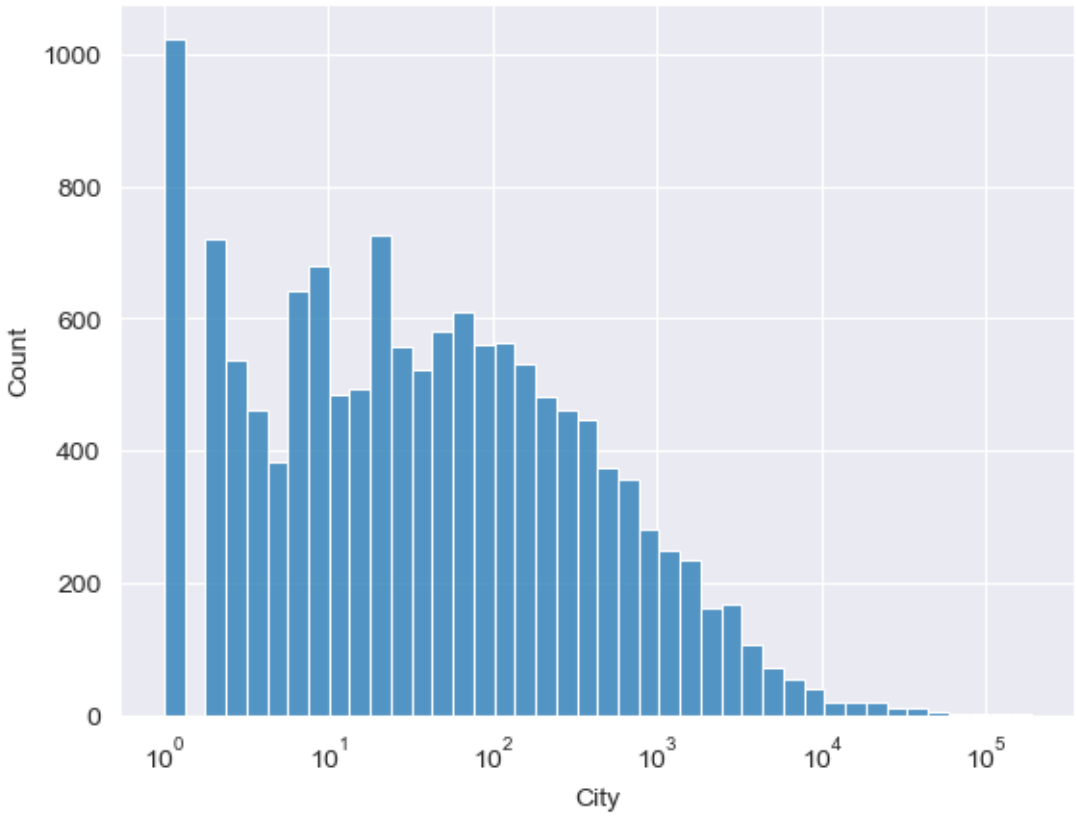
[24]:
```
<Axes: >
```



[25]:
```
import seaborn as sns
sns.set_style("darkgrid")
```

[26]:
```
sns.histplot(cities__by_accident , log_scale = True)
```

[26]: <Axes: xlabel='City', ylabel='Count'>



[27]:
```
cities__by_accident[cities__by_accident == 1]
```

[27]:
```
Lake Andes                      1
Catoctin                        1
Duck Hill                       1
Westbrookville                  1
Saint Croix                     1
                               ..
Benkelman                       1
Old Appleton                    1
Wildrose                        1
Mc Nabb                         1
American Fork-Pleasant Grove    1
Name: City, Length: 1023, dtype: int64
```

[28]:
```
high_accident_cities = cities__by_accident[cities__by_accident > 1000]
low_accident_cities = cities__by_accident[cities__by_accident < 1000]
```

[80]:
```
high_accident_cities
```

[80]:
```
Miami             186917
Houston           169609
Los Angeles       156491
Charlotte         138652
Dallas            130939
                   …
Coalinga            1006
Gulf Breeze         1003
West Hempstead      1003
Fairview            1002
Sedona              1001
Name: City, Length: 1215, dtype: int64
```

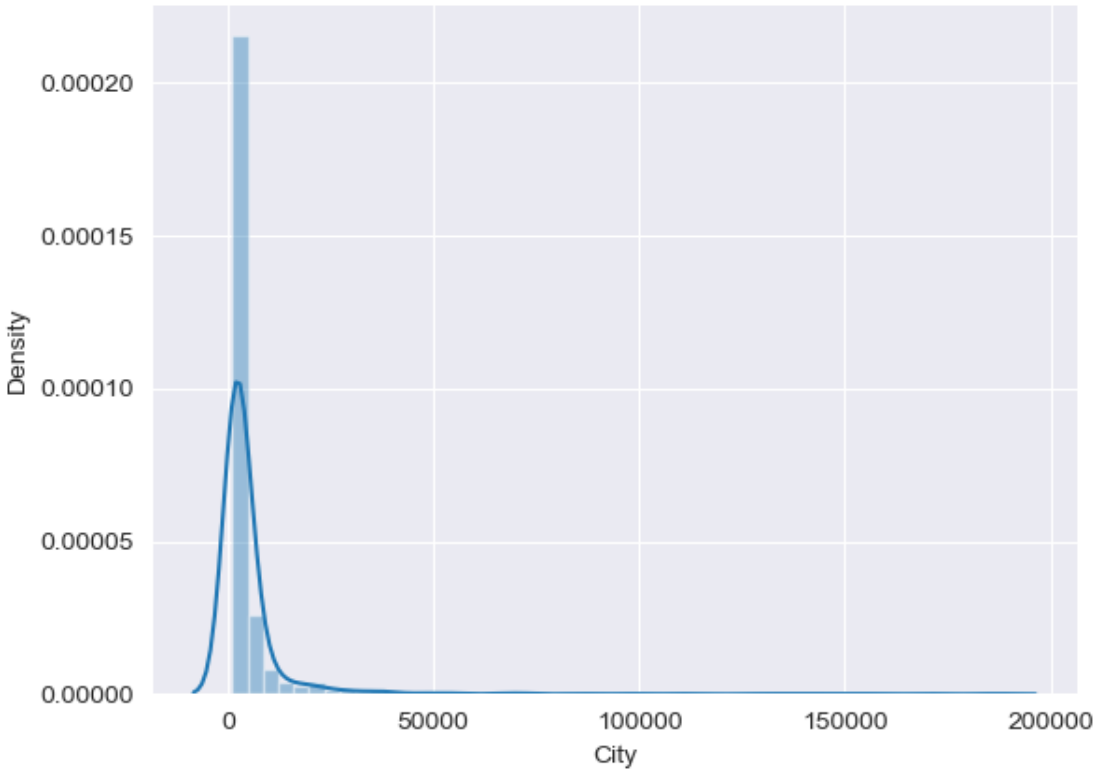
[29]:
```
len(high_accident_cities) / len(cities)
```

[29]:
```
0.08882228233057972
```

[30]:
```
sns.distplot(high_accident_cities)
```

```
C:\Users\Acer\AppData\Local\Temp\ipykernel_12632\2843252471.py:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(high_accident_cities)
```

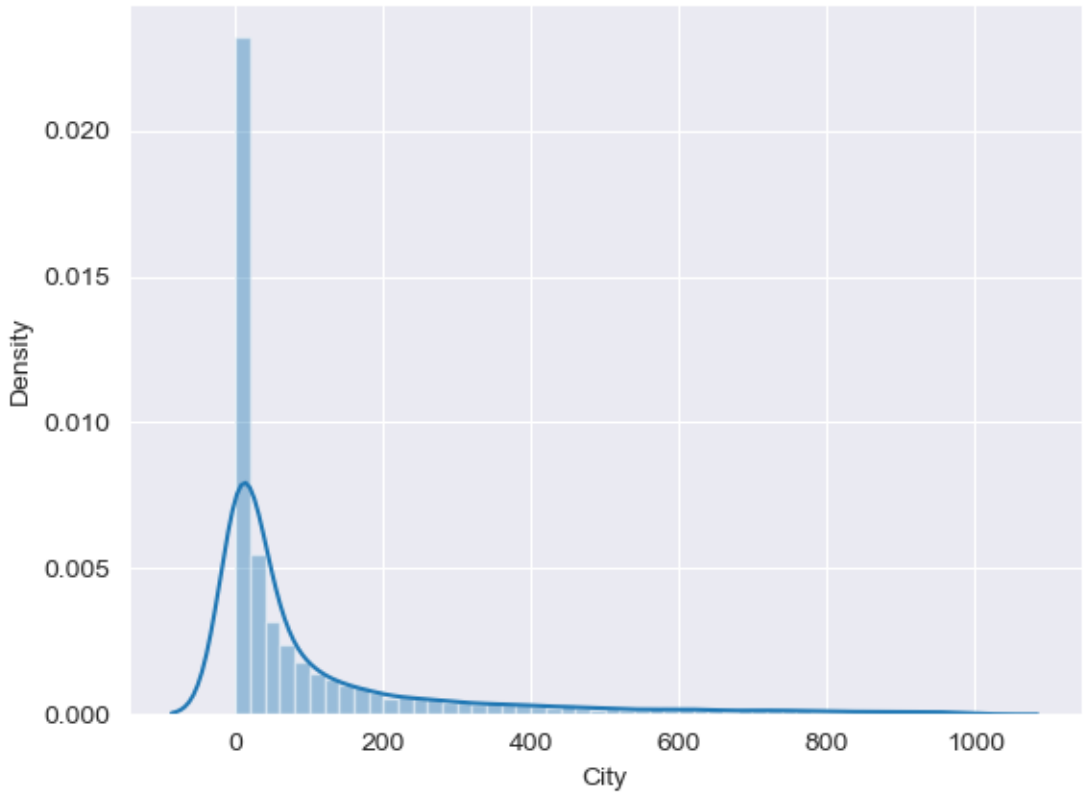
[30]:
```
<Axes: xlabel='City', ylabel='Density'>
```



[31]:
```
sns.distplot(low_accident_cities)
```

```
C:\Users\Acer\AppData\Local\Temp\ipykernel_12632\469555131.py:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(low_accident_cities)
```

[31]:
```
<Axes: xlabel='City', ylabel='Density'>
```



3.1.1 Start Time

[32]:
```
df.columns
```

[32]:
```
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')
```

[33]:
```
df.Start_Time
```

[33]:
```
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
```

[34]:
```
df.Start_Time[0]
```

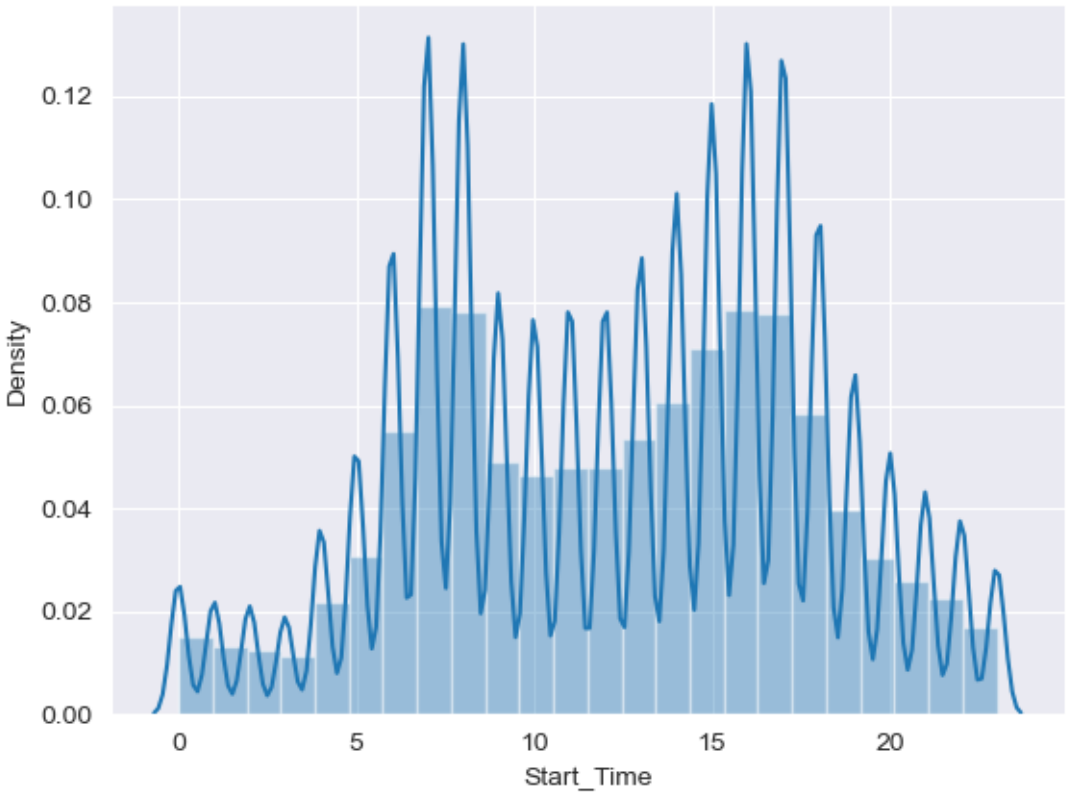
[34]:
```
'2016-02-08 05:46:00'
```

[35]:
```
df.Start_Time = pd.to_datetime(df.Start_Time)
```

[36]:
```
sns.distplot(df.Start_Time.dt.hour , bins=24 , norm_hist=True)

# -- A high percentage of accident occur between 6 am to 10 am(probably people␣
 ↪in a hurry to get to work)
# -- And also the number of accidents is high in between 3 pm to 6 pm(probably␣
 ↪traffic is high or maybe overspeeding or maybe people going back to home␣
 ↪from their work)
# -- But here is the strange thing is that after 6 pm the number of accidents␣
 ↪is reducing , because there is always high chances of accidents in␣
 ↪night(probably the num most of the people doesnt drive after 6pm)
```

```
C:\Users\Acer\AppData\Local\Temp\ipykernel_12632\4258993182.py:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df.Start_Time.dt.hour , bins=24 , norm_hist=True)
```

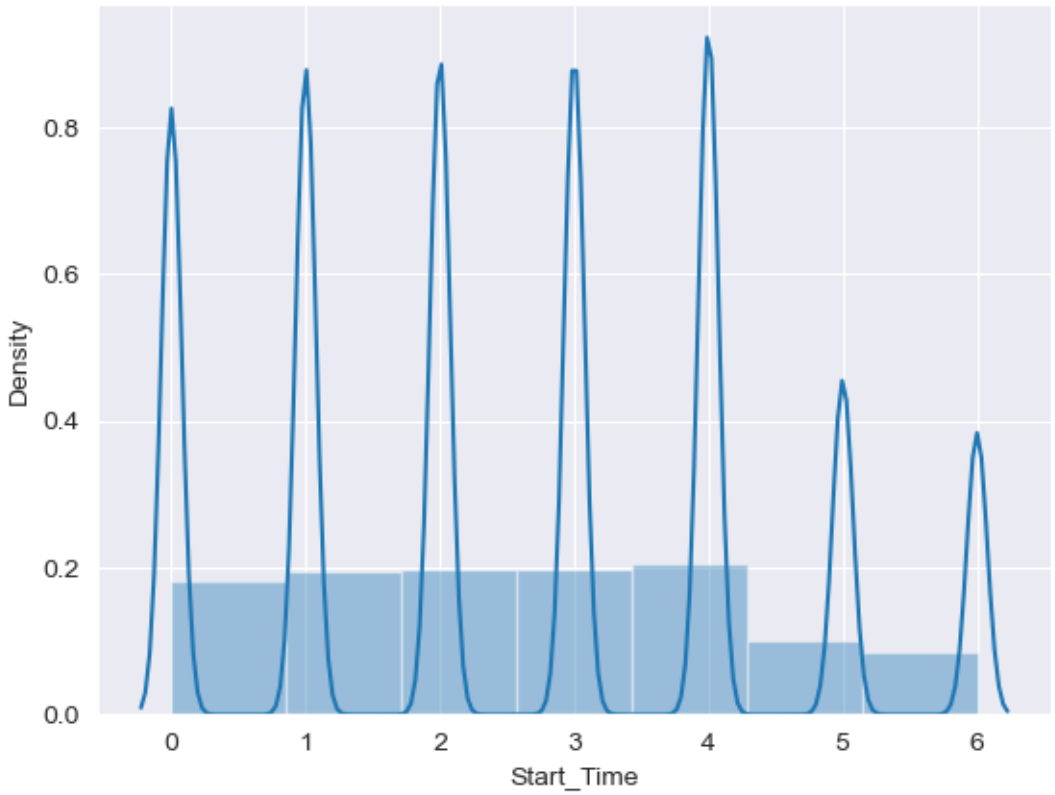
[36]:
```
<Axes: xlabel='Start_Time', ylabel='Density'>
```



[37]:
```
sns.distplot(df.Start_Time.dt.dayofweek , bins=7 , norm_hist=True)
```

```
C:\Users\Acer\AppData\Local\Temp\ipykernel_12632\2993767461.py:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df.Start_Time.dt.dayofweek , bins=7 , norm_hist=True)
```

[37]:
```
<Axes: xlabel='Start_Time', ylabel='Density'>
```

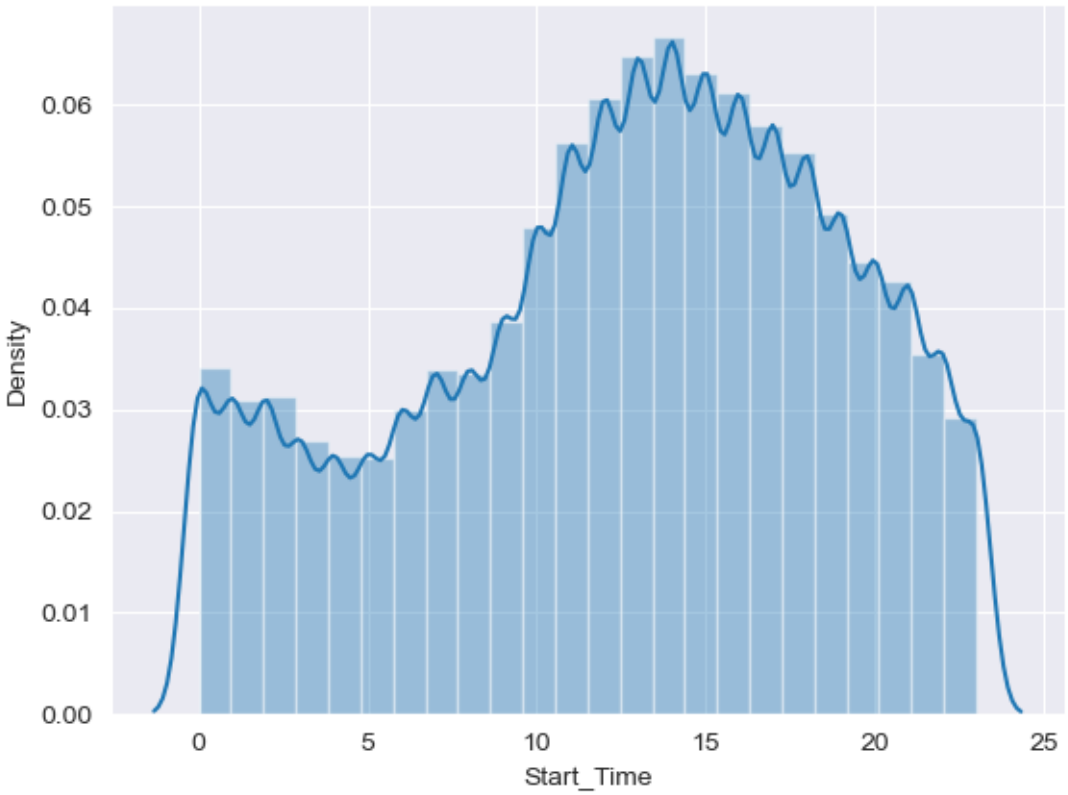


Is the distribution of accident by hour the same on weekends as on weekdays.

[38]:
```
sundays_start_time = df.Start_Time[df.Start_Time.dt.dayofweek == 6]
sns.distplot(sundays_start_time.dt.hour , bins=24 , norm_hist=True)
```

```
C:\Users\Acer\AppData\Local\Temp\ipykernel_12632\1036660686.py:2: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(sundays_start_time.dt.hour , bins=24 , norm_hist=True)
```

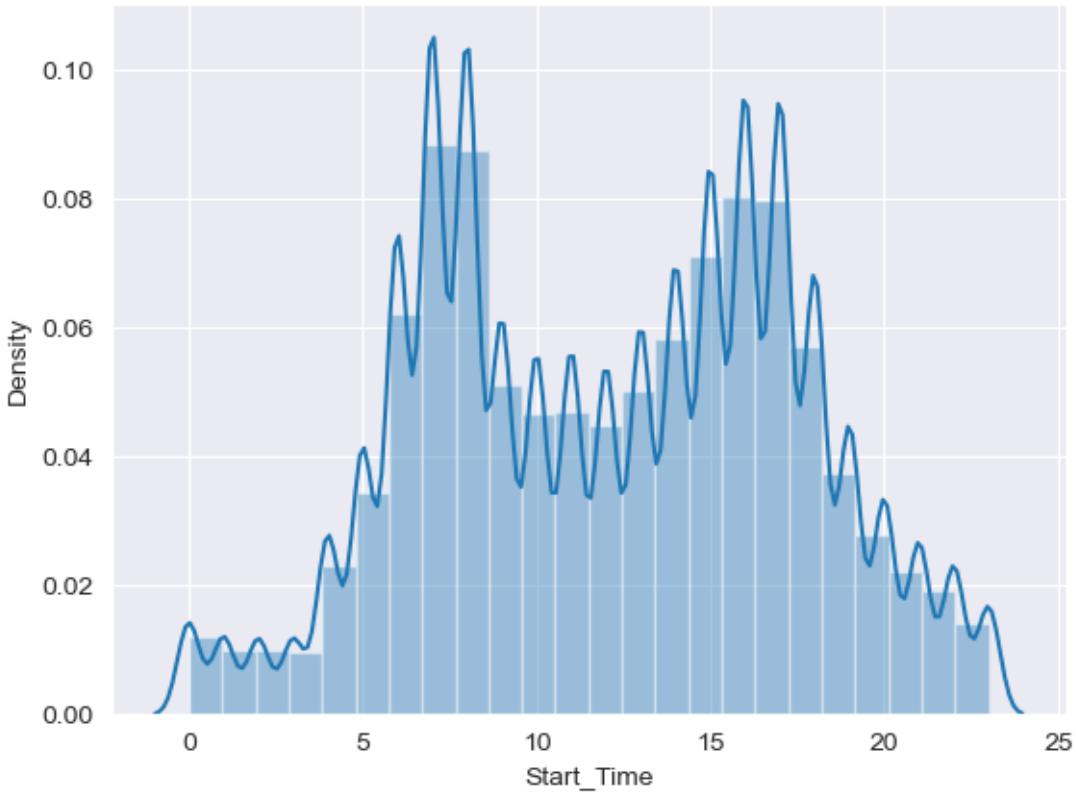
[38]:
```
<Axes: xlabel='Start_Time', ylabel='Density'>
```



[39]:
```
mondays_start_time = df.Start_Time[df.Start_Time.dt.dayofweek == 0]
sns.distplot(mondays_start_time.dt.hour , bins=24 , norm_hist=True)
```

```
C:\Users\Acer\AppData\Local\Temp\ipykernel_12632\1662964316.py:2: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(mondays_start_time.dt.hour , bins=24 , norm_hist=True)
```

[39]: <Axes: xlabel='Start_Time', ylabel='Density'>



On Sundays, the peak occurs between 10 am & 3 pm , unlike week days

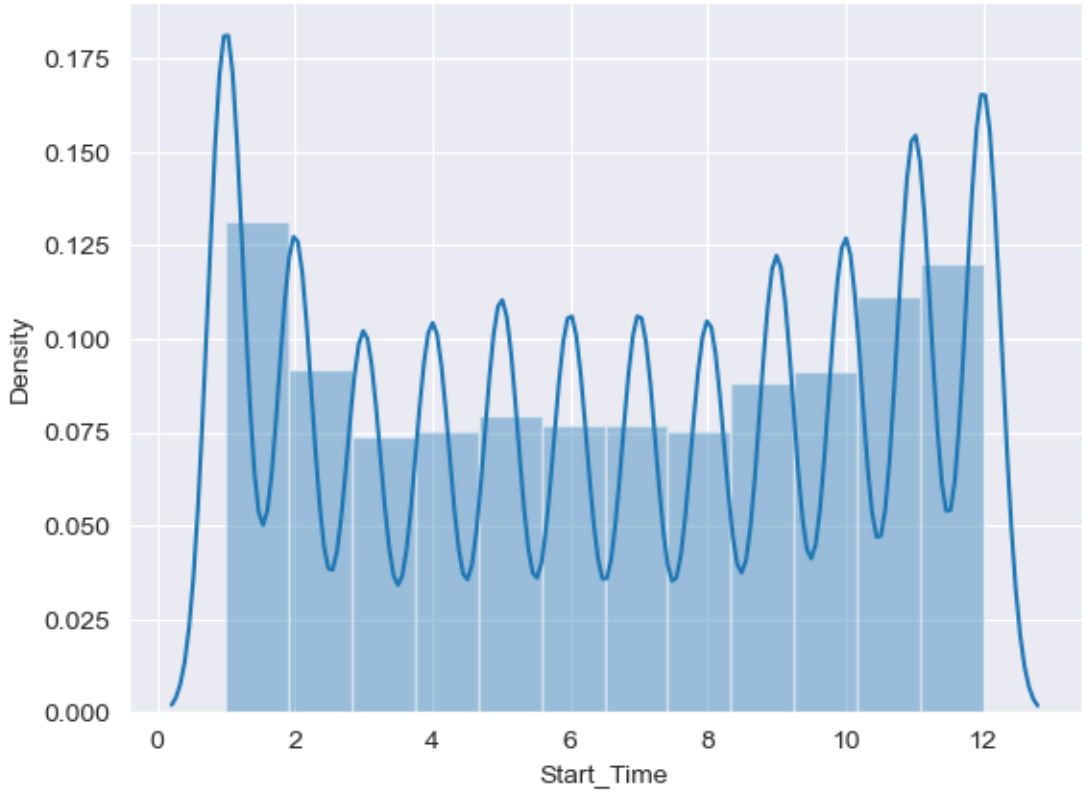
[51]:
```
df.Start_Time.dt.year
```

[51]:
```
0          2016
1          2016
2          2016
3          2016
4          2016
           ...
7728389    2019
7728390    2019
7728391    2019
7728392    2019
7728393    2019
Name: Start_Time, Length: 7728394, dtype: int64
```

[40]:
```
sns.distplot(sundays_start_time.dt.month , bins=12 , norm_hist=True)
# -- The number of accidents is most happening in between the winter season␣
 ↪possibly due to less visibility on roads.
# -- But on the other hand the number of accidents are lower during summer.
```

```
C:\Users\Acer\AppData\Local\Temp\ipykernel_12632\651180735.py:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(sundays_start_time.dt.month , bins=12 , norm_hist=True)
```

[40]: <Axes: xlabel='Start_Time', ylabel='Density'>

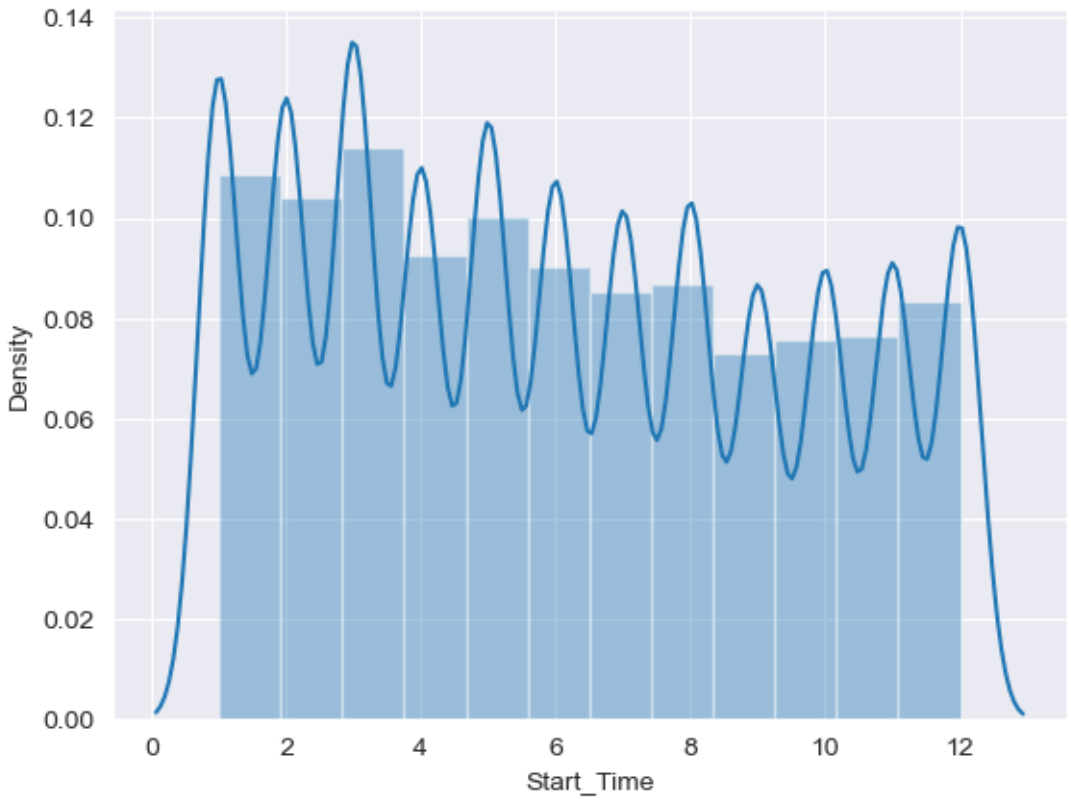


[46]:
```
df_2019 = df[df.Start_Time.dt.year == 2017]
df_2019_Source1 = df_2019[df_2019.Source == 'Source1']
sns.distplot(df_2019_Source1.Start_Time.dt.month , bins=12 , norm_hist=True)
```

```
C:\Users\Acer\AppData\Local\Temp\ipykernel_12632\978620205.py:3: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.
```

```
Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df_2019_Source1.Start_Time.dt.month , bins=12 , norm_hist=True)
```

[46]: <Axes: xlabel='Start_Time', ylabel='Density'>



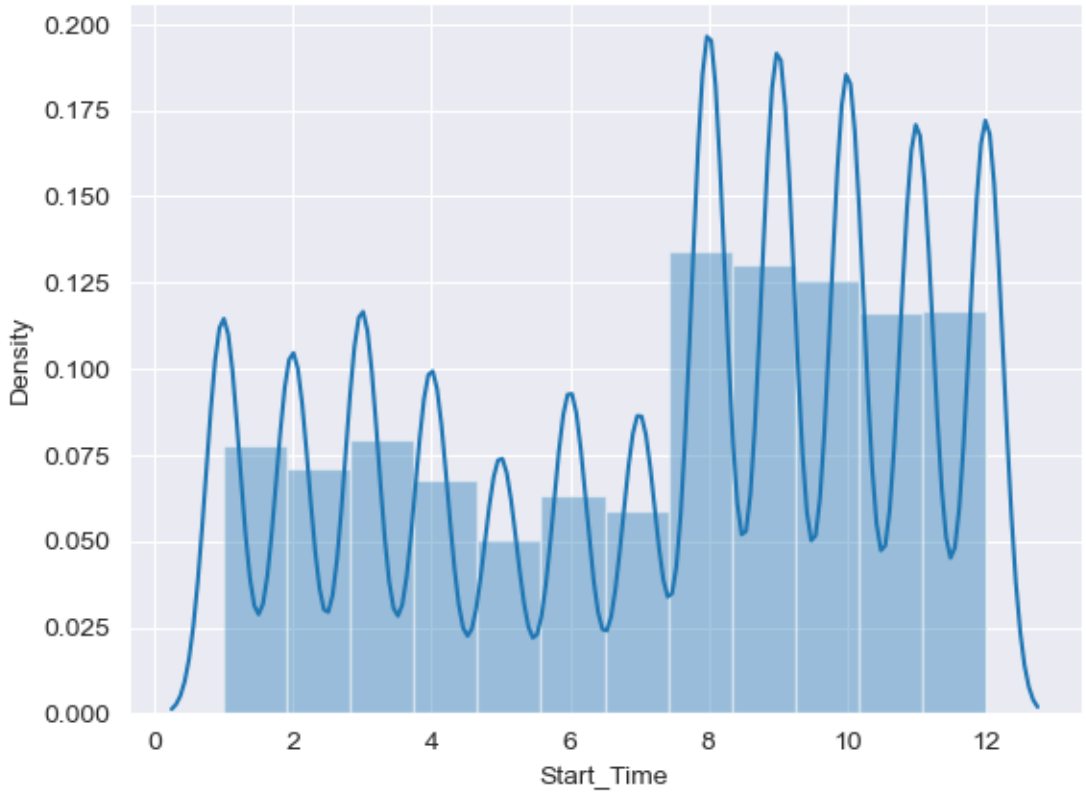
[45]:
```
df_2019 = df[df.Start_Time.dt.year == 2017]
df_2019_Source2 = df_2019[df_2019.Source == 'Source2']
sns.distplot(df_2019_Source2.Start_Time.dt.month , bins=12 , norm_hist=True)
```

```
C:\Users\Acer\AppData\Local\Temp\ipykernel_12632\2083113810.py:3: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
```

https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

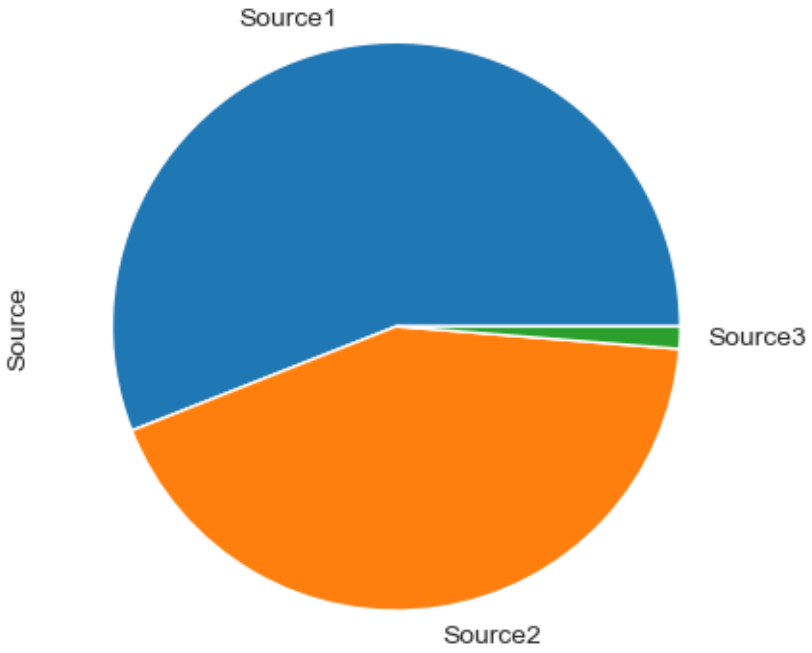
```
sns.distplot(df_2019_Source2.Start_Time.dt.month , bins=12 , norm_hist=True)
```

[45]: <Axes: xlabel='Start_Time', ylabel='Density'>



[43]:
```
df.Source.value_counts().plot(kind='pie')
```

[43]: <Axes: ylabel='Source'>



Consider excluding source2 data , seems to have issues.

4 # Start Latitude & Start Longitude

[47]:
```
df.Start_Lat
```

[47]:
```
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
```

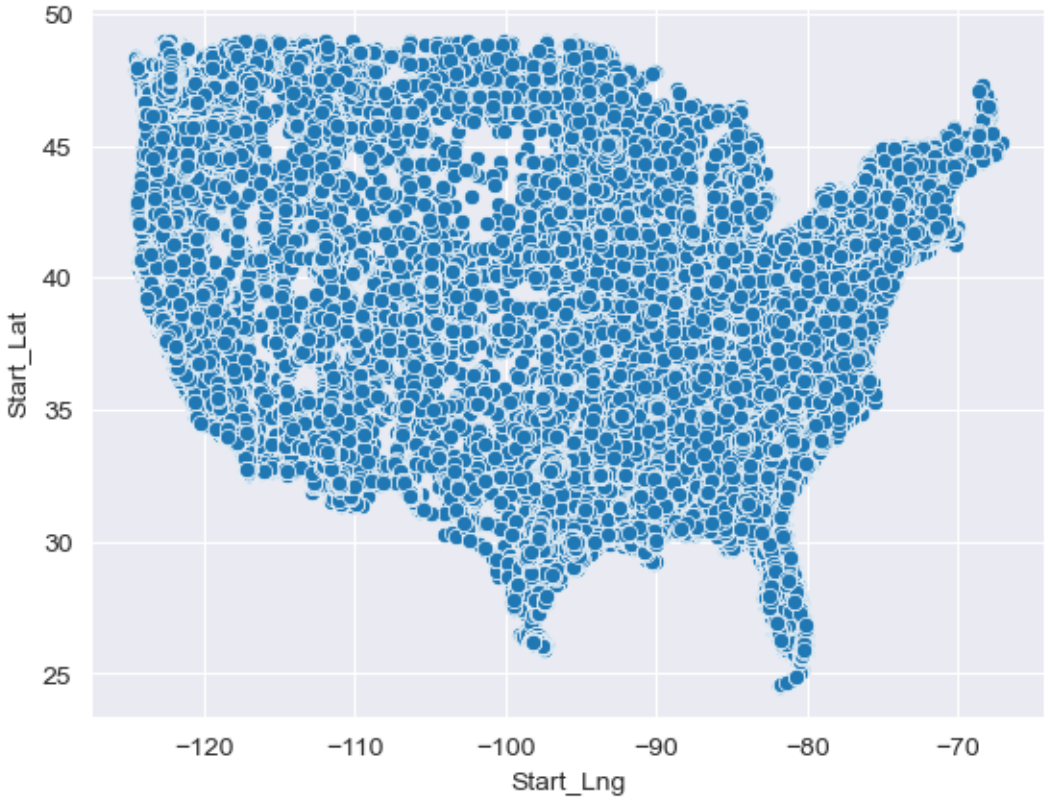
[49]:
```
df.Start_Lng
```

[49]:
```
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
```

[53]:
```
sns.scatterplot(x=df.Start_Lng , y=df.Start_Lat)
```

[53]:
```
<Axes: xlabel='Start_Lng', ylabel='Start_Lat'>
```



[55]:
```
#scatterplot with only 10% samples

sample_df = df.sample(int(0.1 * len(df)))
sample_df
```

[55]:

	ID	Source	Severity	Start_Time	\
171077	A-171084	Source2	2	2016-11-04 20:33:23	
1384817	A-1394611	Source2	2	2020-07-21 17:02:12	
11456	A-11457	Source2	2	2017-01-17 18:33:42	
3027375	A-3037254	Source2	3	2018-02-27 17:23:36	
5162394	A-5201538	Source1	2	2022-10-12 22:13:03	
…	…	…	…	…	
7528612	A-7577980	Source1	2	2018-05-01 15:44:54	
2192063	A-2201929	Source2	3	2019-03-05 06:13:02	
5251867	A-5291718	Source1	2	2022-03-23 06:20:00	
4801366	A-4837798	Source1	2	2022-11-23 16:27:00	
6109550	A-6154841	Source1	2	2021-10-07 12:36:00	

	End_Time	Start_Lat	Start_Lng	End_Lat	\
171077	2016-11-04 21:33:23	42.269711	-88.237640	NaN	
1384817	2020-07-21 18:02:48	37.189724	-121.992638	NaN	
11456	2017-01-17 19:03:28	37.314762	-121.913216	NaN	
3027375	2018-02-27 18:23:01	34.975117	-81.984497	NaN	
5162394	2022-10-12 23:31:04.000000000	37.227486	-77.396902	37.227574	
…	…	…	…	…	
7528612	2018-05-01 21:44:54	37.511460	-121.937720	37.514630	
2192063	2019-03-05 06:42:30	41.818680	-87.914902	NaN	
5251867	2022-03-23 09:20:00	40.799822	-73.929888	40.797542	
4801366	2022-11-23 17:44:49	33.748307	-118.008983	33.755376	
6109550	2021-10-07 13:55:14	36.703879	-121.707772	36.697380	

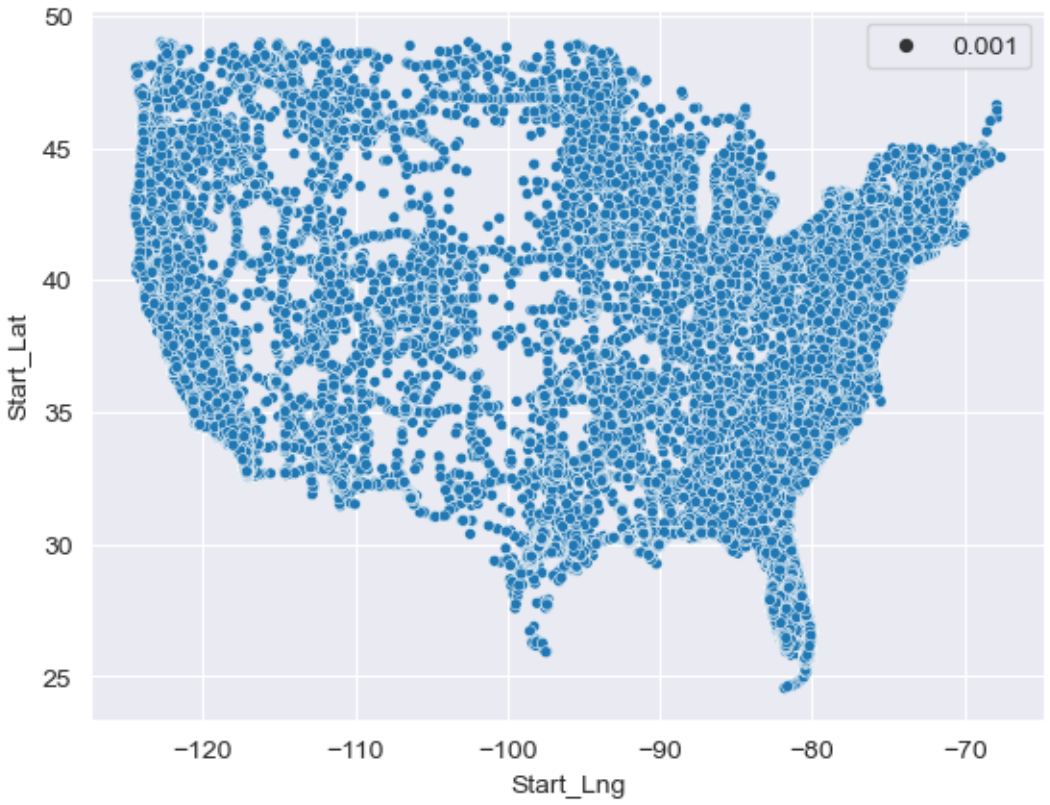
	End_Lng	Distance(mi)	…	Roundabout	Station	Stop	\
171077	NaN	0.010	…	False	False	True	
1384817	NaN	0.000	…	False	False	False	
11456	NaN	0.010	…	False	False	False	
3027375	NaN	0.000	…	False	False	False	
5162394	-77.396454	0.025	…	False	False	False	
…	…	…	…	…	…	…	
7528612	-121.939570	0.241	…	False	False	False	
2192063	NaN	5.820	…	False	False	False	
5251867	-73.929348	0.160	…	False	False	False	
4801366	-118.017440	0.689	…	False	False	False	
6109550	-121.701242	0.577	…	False	False	False	

	Traffic_Calming	Traffic_Signal	Turning_Loop	Sunrise_Sunset	\
171077	False	False	False	Night	
1384817	False	False	False	Day	
11456	False	False	False	Night	
3027375	False	False	False	Day	
5162394	False	False	False	Night	
…	…	…	…	…	
7528612	False	False	False	Day	

```
2192063          False          False        False          Night
5251867          False          False        False          Night
4801366          False          False        False            Day
6109550          False          False        False            Day

        Civil_Twilight Nautical_Twilight Astronomical_Twilight
171077          Night             Night                 Night
1384817           Day               Day                   Day
11456           Night             Night                   Day
3027375           Day               Day                   Day
5162394         Night             Night                 Night
…                 …                 …                     …
7528612           Day               Day                   Day
2192063           Day               Day                   Day
5251867         Night               Day                   Day
4801366           Day               Day                   Day
6109550           Day               Day                   Day

[772839 rows x 46 columns]
```

[57]:
```
sns.scatterplot(x=sample_df.Start_Lng , y=sample_df.Start_Lat , size=0.001)
```

[57]:
```
<Axes: xlabel='Start_Lng', ylabel='Start_Lat'>
```



[68]:
```
#--Plotting Lat , Long cordinates on a map or in a heatmap using folium

from folium.plugins import HeatMap
```

[64]:
```
lat , lon = df.Start_Lat[0] , df.Start_Lng[0]
lat , lon
```

[64]:
```
(39.865147, -84.058723)
```

[71]:
```
#--Creating heatmap data

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

[71]:
```
<zip at 0x1dadccb7e80>
```

[78]:
```
sample_df = df.sample(int(0.001 * len(df)))
lat_lon_pairs = zip(list(sample_df.Start_Lat) , list(sample_df.Start_Lng))
```

[79]:
```
map=folium.Map()

#--HeatMap(heat_data).add_to(map)
HeatMap(lat_lon_pairs).add_to(map)

map
```

[79]:
```
<folium.folium.Map at 0x1d852d93df0>
```

5 Ask question & Answers

1.Are there more accidents in warmer or colder areas? –Answered(In colder areas)

2.Which 5 states have the highest number of accidents ? How about per capita? –Answered(Miami
Houston
Los Angeles
Charlotte
Dallas)

3.Does New york show up in the data? if yes , why is the count lower if this themost populated city? –Answered(There is no data related to New york)

4.Among the top 100 cities in number of accidents,which states do they belong to most frequently?

5.What time of the day are accidents most frequent in ? – Answered(A high percentage of accident occur between 6 am to 10 am(probably people in a hurry to get to work)) And also the number of accidents is high in between 3 pm to 6 pm(probably traffic is high or maybe overspeeding or maybe people going back to home from their work)

6.Which day of the week have the most accidents? – Answered(on the week Days)

7.Which months have the most accidets? – Answered(winter season)

8.What is the trend of accidents year over year(decreasing/increasing)? –Answered(The trend is high during the winter season probably less visibility on roads)

9.When is accidents per unit of traffic the highest?

6 Summary & conclusion

Insights: –No Data from New York –The number of accidents per city descreses/increases exponentially. –Less than 5–Over 1200 cities have reported just 1 accident(need to investigate).