DataAnalysis-US_accidents

January 30, 2023

1 US Accidents Exploratory Data Analysis

by Shivam Kumar

TODO - talk about the EDA

TODO - talk about the dataset (source, what it contains, how it will be useful)

- Kaggle
- information about accidents
- can be useful to prevent accidents

(This does not contain data about New York)

1.1 Dataset

[1]:
```
data_filename  = './Dataset/US_Accidents_Dec21_updated.csv'
```

1.2 Data Preparation and Cleaning

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

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

[3]:
```
dataFrame = pd.read_csv(data_filename)
```

[4]:
```
dataFrame.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2845342 entries, 0 to 2845341
Data columns (total 47 columns):
 #   Column                 Dtype
---  ------                 -----
 0   ID                     object
 1   Severity               int64
 2   Start_Time             object
 3   End_Time               object
 4   Start_Lat              float64
```

```
 5   Start_Lng              float64
 6   End_Lat                float64
 7   End_Lng                float64
 8   Distance(mi)           float64
 9   Description            object
 10  Number                 float64
 11  Street                 object
 12  Side                   object
 13  City                   object
 14  County                 object
 15  State                  object
 16  Zipcode                object
 17  Country                object
 18  Timezone               object
 19  Airport_Code           object
 20  Weather_Timestamp      object
 21  Temperature(F)         float64
 22  Wind_Chill(F)          float64
 23  Humidity(%)            float64
 24  Pressure(in)           float64
 25  Visibility(mi)         float64
 26  Wind_Direction         object
 27  Wind_Speed(mph)        float64
 28  Precipitation(in)      float64
 29  Weather_Condition      object
 30  Amenity                bool
 31  Bump                   bool
 32  Crossing               bool
 33  Give_Way               bool
 34  Junction               bool
 35  No_Exit                bool
 36  Railway                bool
 37  Roundabout             bool
 38  Station                bool
 39  Stop                   bool
 40  Traffic_Calming        bool
 41  Traffic_Signal         bool
 42  Turning_Loop           bool
 43  Sunrise_Sunset         object
 44  Civil_Twilight         object
 45  Nautical_Twilight      object
 46  Astronomical_Twilight  object
dtypes: bool(13), float64(13), int64(1), object(20)
memory usage: 773.4+ MB
```

[5]:
```
dataFrame.describe()
```

[5]:

```
            Severity     Start_Lat     Start_Lng       End_Lat       End_Lng  \
count  2.845342e+06  2.845342e+06  2.845342e+06  2.845342e+06  2.845342e+06
mean   2.137572e+00  3.624520e+01 -9.711463e+01  3.624532e+01 -9.711439e+01
std    4.787216e-01  5.363797e+00  1.831782e+01  5.363873e+00  1.831763e+01
min    1.000000e+00  2.456603e+01 -1.245481e+02  2.456601e+01 -1.245457e+02
25%    2.000000e+00  3.344517e+01 -1.180331e+02  3.344628e+01 -1.180333e+02
50%    2.000000e+00  3.609861e+01 -9.241808e+01  3.609799e+01 -9.241772e+01
75%    2.000000e+00  4.016024e+01 -8.037243e+01  4.016105e+01 -8.037338e+01
max    4.000000e+00  4.900058e+01 -6.711317e+01  4.907500e+01 -6.710924e+01

       Distance(mi)        Number  Temperature(F)  Wind_Chill(F)  \
count  2.845342e+06  1.101431e+06    2.776068e+06   2.375699e+06
mean   7.026779e-01  8.089408e+03    6.179356e+01   5.965823e+01
std    1.560361e+00  1.836009e+04    1.862263e+01   2.116097e+01
min    0.000000e+00  0.000000e+00   -8.900000e+01  -8.900000e+01
25%    5.200000e-02  1.270000e+03    5.000000e+01   4.600000e+01
50%    2.440000e-01  4.007000e+03    6.400000e+01   6.300000e+01
75%    7.640000e-01  9.567000e+03    7.600000e+01   7.600000e+01
max    1.551860e+02  9.999997e+06    1.960000e+02   1.960000e+02

        Humidity(%)  Pressure(in)  Visibility(mi)  Wind_Speed(mph)  \
count  2.772250e+06  2.786142e+06    2.774796e+06     2.687398e+06
mean   6.436545e+01  2.947234e+01    9.099391e+00     7.395044e+00
std    2.287457e+01  1.045286e+00    2.717546e+00     5.527454e+00
min    1.000000e+00  0.000000e+00    0.000000e+00     0.000000e+00
25%    4.800000e+01  2.931000e+01    1.000000e+01     3.500000e+00
50%    6.700000e+01  2.982000e+01    1.000000e+01     7.000000e+00
75%    8.300000e+01  3.001000e+01    1.000000e+01     1.000000e+01
max    1.000000e+02  5.890000e+01    1.400000e+02     1.087000e+03

       Precipitation(in)
count       2.295884e+06
mean        7.016940e-03
std         9.348831e-02
min         0.000000e+00
25%         0.000000e+00
50%         0.000000e+00
75%         0.000000e+00
max         2.400000e+01
```

[6]:

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

numeric_df = dataFrame.select_dtypes(include = numerics)

len(numeric_df.columns)
```

[6]: 14

1.2.1 Percentage of missing values per column

[7]:
```
missing_percentages = (dataFrame.isna().sum().sort_values(ascending = False) /␣
↪len(dataFrame)) * 100

missing_percentages
```

[7]:
```
Number                   61.290031
Precipitation(in)        19.310789
Wind_Chill(F)            16.505678
Wind_Speed(mph)           5.550967
Wind_Direction            2.592834
Humidity(%)               2.568830
Weather_Condition         2.482514
Visibility(mi)            2.479350
Temperature(F)            2.434646
Pressure(in)              2.080593
Weather_Timestamp         1.783125
Airport_Code              0.335601
Timezone                  0.128596
Nautical_Twilight         0.100761
Civil_Twilight            0.100761
Sunrise_Sunset            0.100761
Astronomical_Twilight     0.100761
Zipcode                   0.046356
City                      0.004815
Street                    0.000070
Country                   0.000000
Junction                  0.000000
Start_Time                0.000000
End_Time                  0.000000
Start_Lat                 0.000000
Turning_Loop              0.000000
Traffic_Signal            0.000000
Traffic_Calming           0.000000
Stop                      0.000000
Station                   0.000000
Roundabout                0.000000
Railway                   0.000000
No_Exit                   0.000000
Crossing                  0.000000
Give_Way                  0.000000
Bump                      0.000000
Amenity                   0.000000
Start_Lng                 0.000000
End_Lat                   0.000000
End_Lng                   0.000000
```

```
Distance(mi)              0.000000
Description               0.000000
Severity                  0.000000
Side                      0.000000
County                    0.000000
State                     0.000000
ID                        0.000000
dtype: float64
```

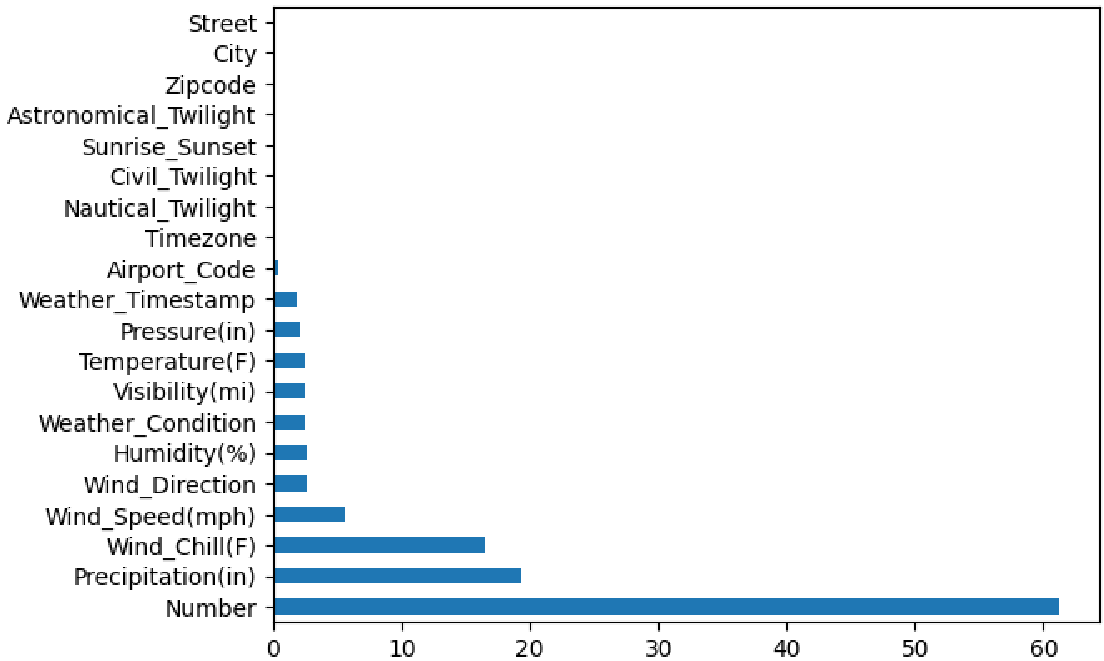
[8]:
```
filtered_missing_percentages = missing_percentages[missing_percentages != 0]

filtered_missing_percentages
```

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

1.2.2 Plotting Graph

[9]:
```
filtered_missing_percentages.plot(kind = "barh")
```

[9]:
```
<AxesSubplot: >
```



[10]:
```
dataFrame.columns
```

[10]:
```
Index(['ID', 'Severity', 'Start_Time', 'End_Time', 'Start_Lat', 'Start_Lng',
       'End_Lat', 'End_Lng', 'Distance(mi)', 'Description', 'Number', 'Street',
       'Side', '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')
```

1.3 Exploratory Analysis and Visualization

1.3.1 Columns to be analyzed:

1. City
2. Start_Time
3. Start_Lat, Start_Lng
4. Temperature(F)
5. Visibility
6. Weather_Condition

1.3.2 1. City Analysis

[11]:
```
dataFrame.City
```

[11]:
```
0                 Dublin
1                 Dayton
2             Cincinnati
3                  Akron
4             Cincinnati
                 …
2845337        Riverside
2845338        San Diego
2845339           Orange
2845340      Culver City
2845341         Highland
Name: City, Length: 2845342, dtype: object
```

[12]:
```
cities = dataFrame.City.unique()
len(cities)
```

[12]:
```
11682
```

[13]:
```
top_accident_cities = dataFrame.City.value_counts()

top_accident_cities
```

[13]:
```
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
```

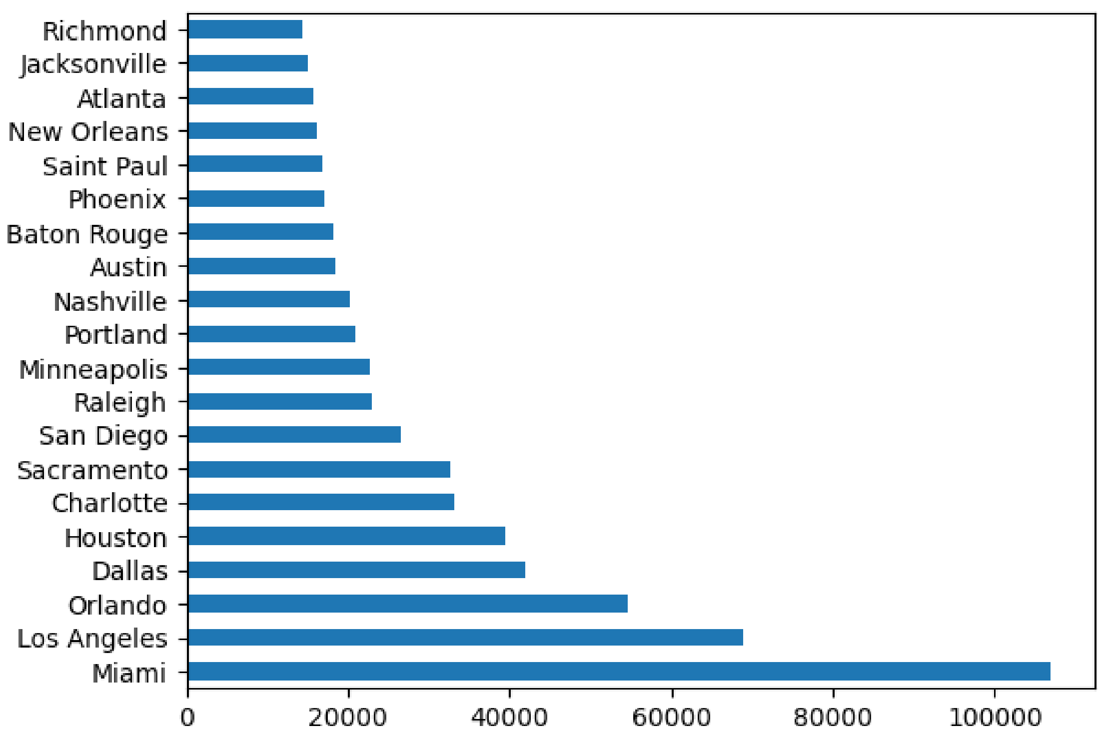
[14]:
```
top_accident_cities[:20]
```

[14]:
```
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
```

[15]:
```
top_accident_cities[:20].plot(kind = "barh")
```

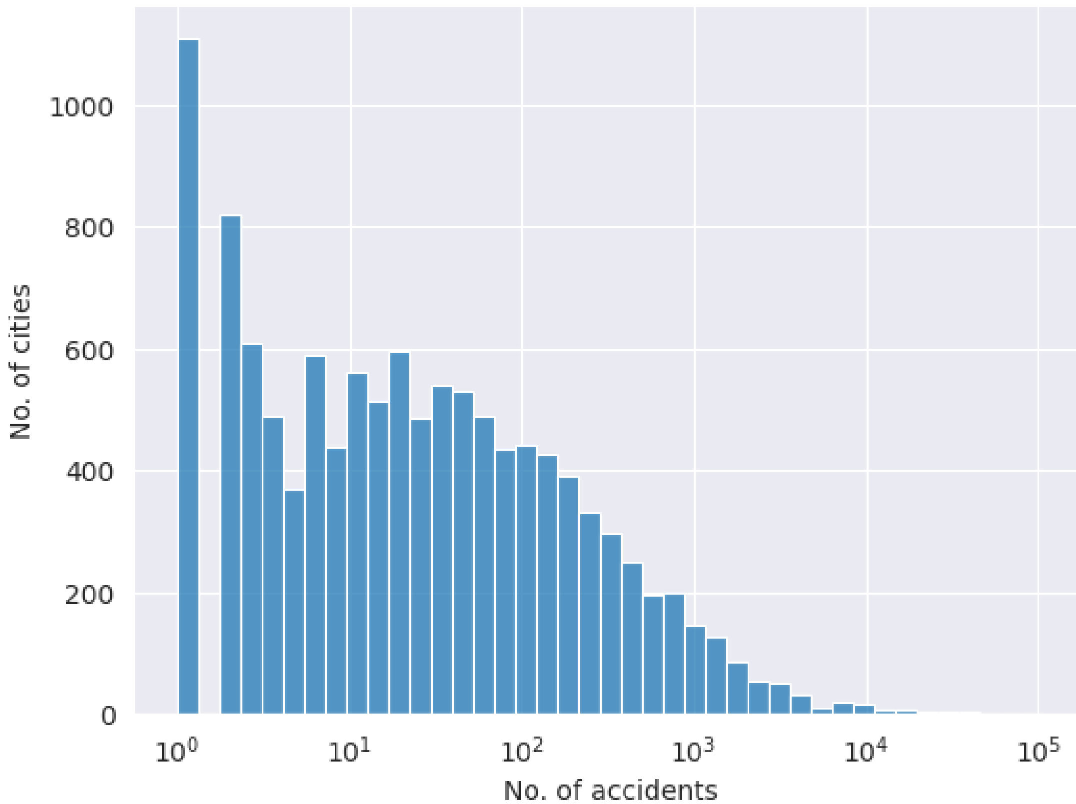
[15]: <AxesSubplot: >



[16]:
```
import seaborn as sns
sns.set_style("darkgrid")
```

[17]:
```
cities_accidents_plot = sns.histplot(top_accident_cities, log_scale = True,␣
  ↪stat="count")

cities_accidents_plot.set(xlabel = "No. of accidents", ylabel="No. of cities")
```

[17]: [Text(0.5, 0, 'No. of accidents'), Text(0, 0.5, 'No. of cities')]



1.3.3 No. of cities with only 1 yearly accident

[18]:
```
len(top_accident_cities[top_accident_cities == 1])
```

[18]: 1110

Observations from city-wise data:

- Miami has the most number of accidents.
- Over 1100 cities out of 11682 cities have atleast one accident.

1.3.4 2. Start_Time Analysis

[19]:
```
dataFrame.Start_Time
```

[19]:
```
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
```

[20]:
```
dataFrame.Start_Time.value_counts()
```

[20]:
```
2021-01-26 16:16:13    214
2021-01-26 16:17:33    150
2021-02-16 06:42:43    130
2021-05-03 06:29:42     92
2021-04-26 08:58:47     87
                       …
2021-10-08 03:58:30      1
2021-12-16 23:53:00      1
2021-07-27 18:46:31      1
2021-10-26 17:37:30      1
2019-08-23 18:52:06      1
Name: Start_Time, Length: 1959333, dtype: int64
```

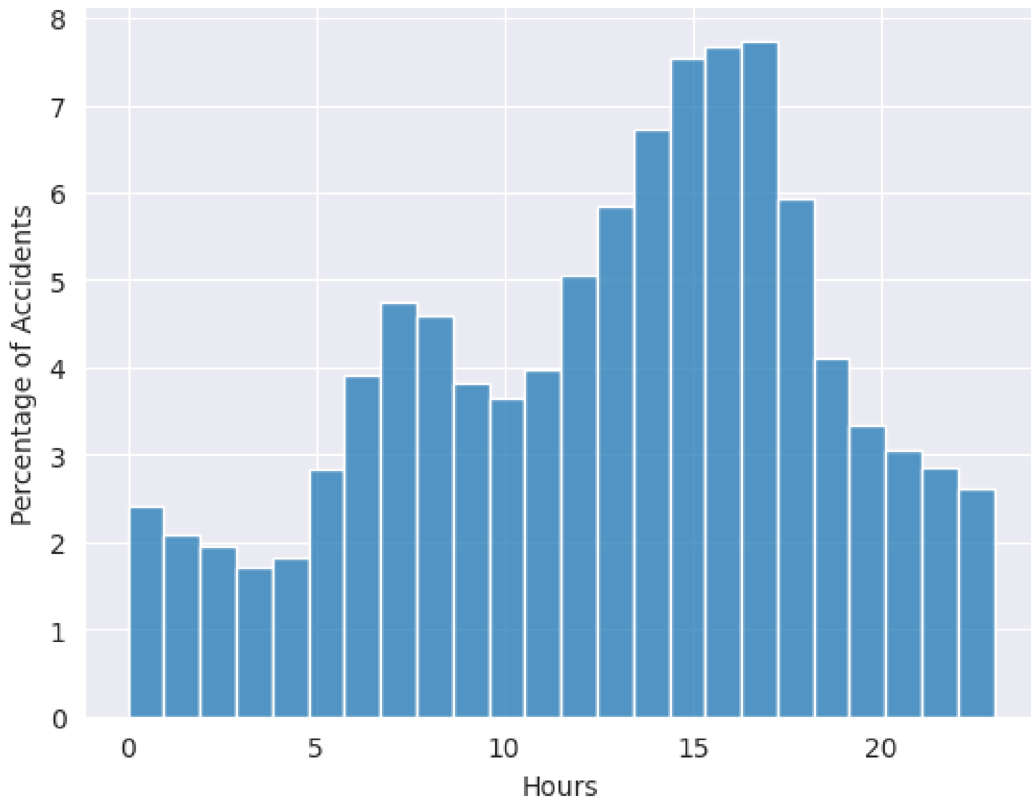
[21]:
```
# converting strings of dates into datetime type

dataFrame.Start_Time = pd.to_datetime(dataFrame.Start_Time)

dataFrame.Start_Time[0]
```

[21]:
```
Timestamp('2016-02-08 00:37:08')
```

1.3.5 Time of the day

[22]:
```
start_time_plot = sns.histplot(dataFrame.Start_Time.dt.hour, stat = "percent",␣
  ↪bins = 24)

start_time_plot.set(xlabel = "Hours", ylabel="Percentage of Accidents")
```

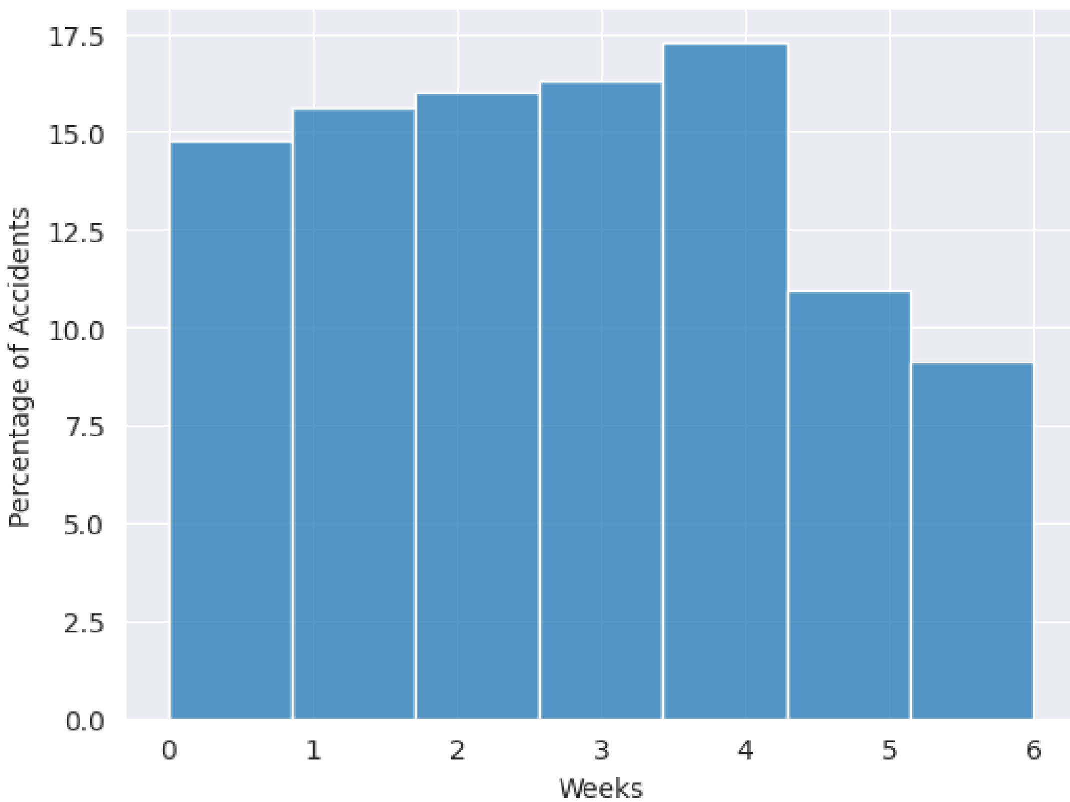
[22]: [Text(0.5, 0, 'Hours'), Text(0, 0.5, 'Percentage of Accidents')]



1.3.6 Days of the week

[23]:
```
start_time_plot = sns.histplot(dataFrame.Start_Time.dt.dayofweek, stat =␣
↪"percent", bins = 7)

start_time_plot.set(xlabel = "Weeks", ylabel="Percentage of Accidents")
```

[23]: [Text(0.5, 0, 'Weeks'), Text(0, 0.5, 'Percentage of Accidents')]



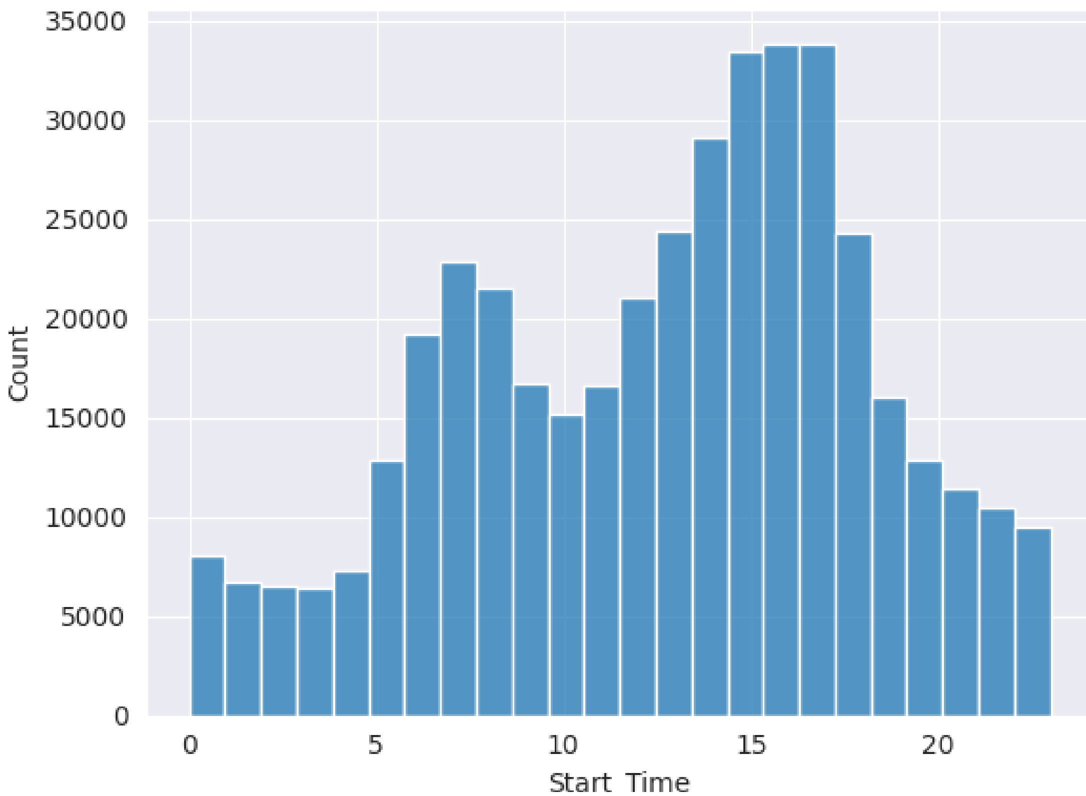
- `Weekdays` have more higher percentage of accidents than the weeekends.

`Does the time of accidents matches on weekends and weekdays?`

1. Monday

[24]:
```
monday_start_time = dataFrame.Start_Time[dataFrame.Start_Time.dt.dayofweek == 0]

sns.histplot(monday_start_time.dt.hour, stat = "count", bins = 24)
```

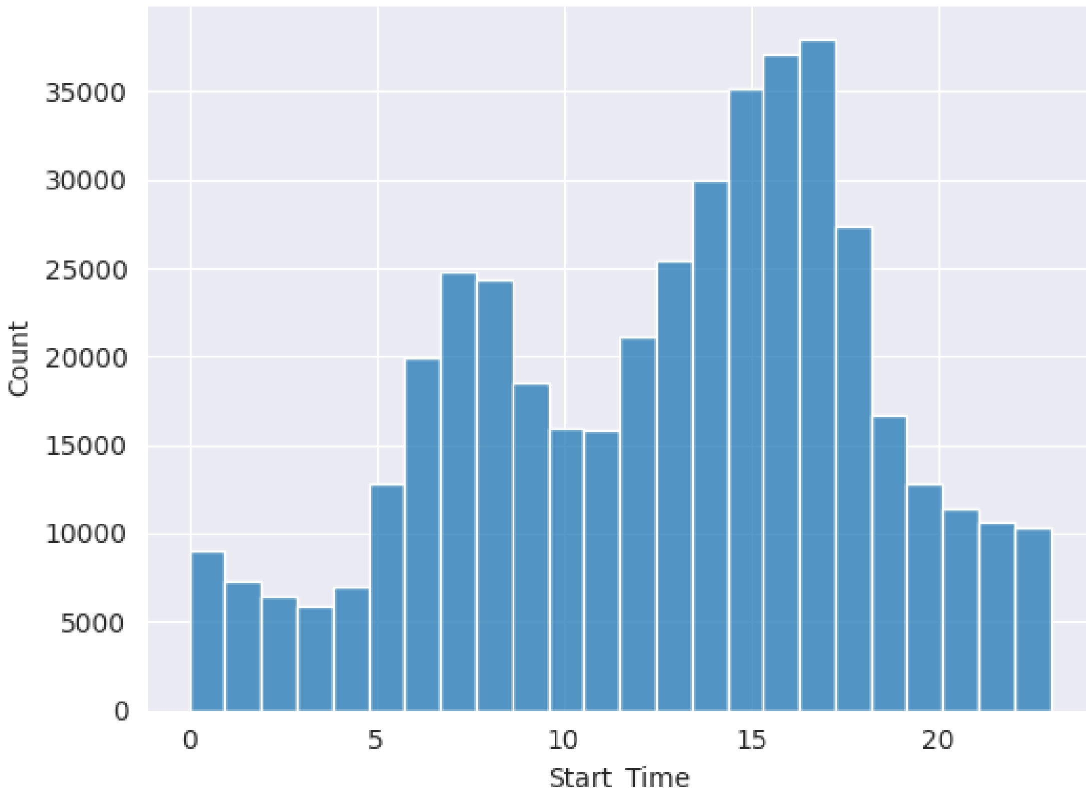
[24]:
```
<AxesSubplot: xlabel='Start_Time', ylabel='Count'>
```



2. Tuesday

[25]:
```
tuesday_start_time = dataFrame.Start_Time[dataFrame.Start_Time.dt.dayofweek ==␣
↪1]

sns.histplot(tuesday_start_time.dt.hour, stat = "count", bins = 24)
```

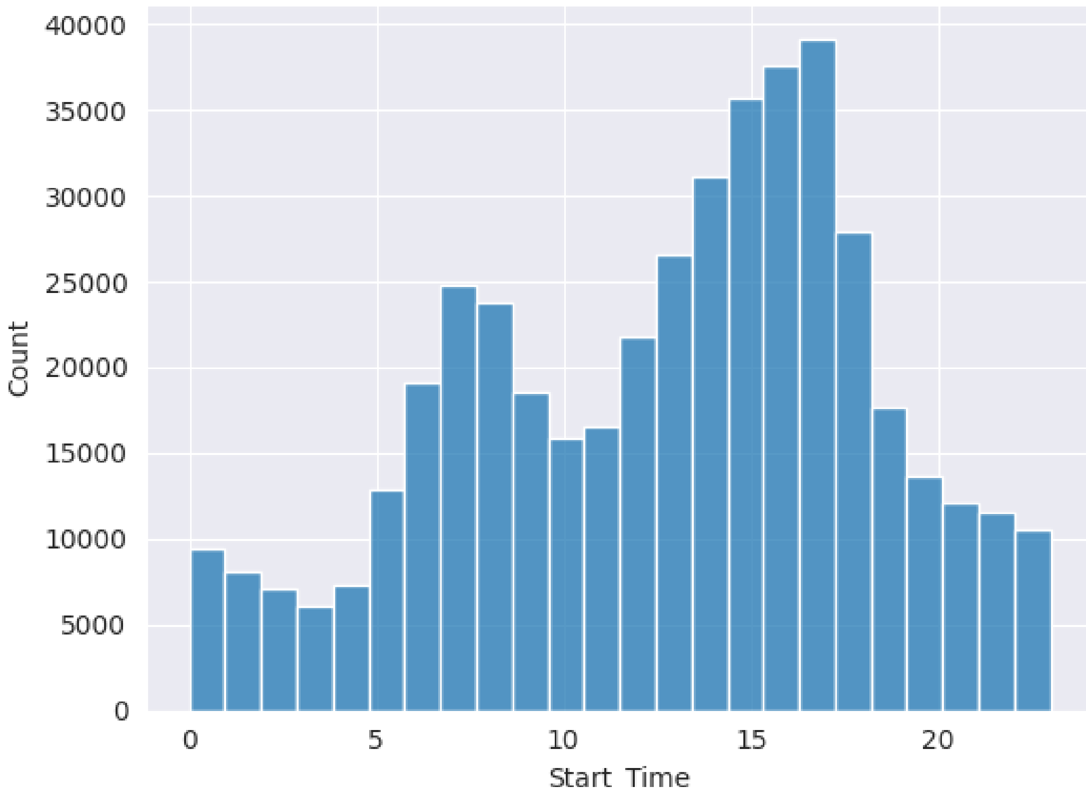
[25]: <AxesSubplot: xlabel='Start_Time', ylabel='Count'>



3. Wednesday

[26]:
```
wednesday_start_time = dataFrame.Start_Time[dataFrame.Start_Time.dt.dayofweek␣
↪== 2]

sns.histplot(wednesday_start_time.dt.hour, stat = "count", bins = 24)
```

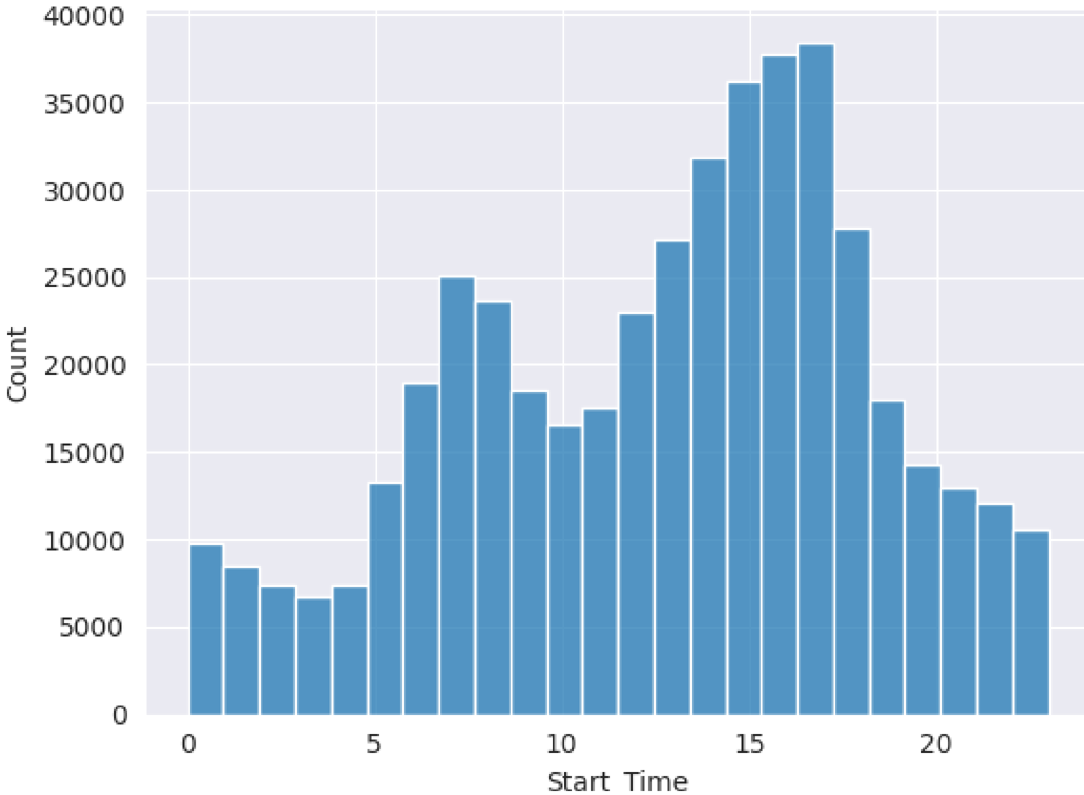
[26]: <AxesSubplot: xlabel='Start_Time', ylabel='Count'>



4. Thursday

```
[27]: thursday_start_time = dataFrame.Start_Time[dataFrame.Start_Time.dt.dayofweek ==␣
      ↪3]

      sns.histplot(thursday_start_time.dt.hour, stat = "count", bins = 24)
```

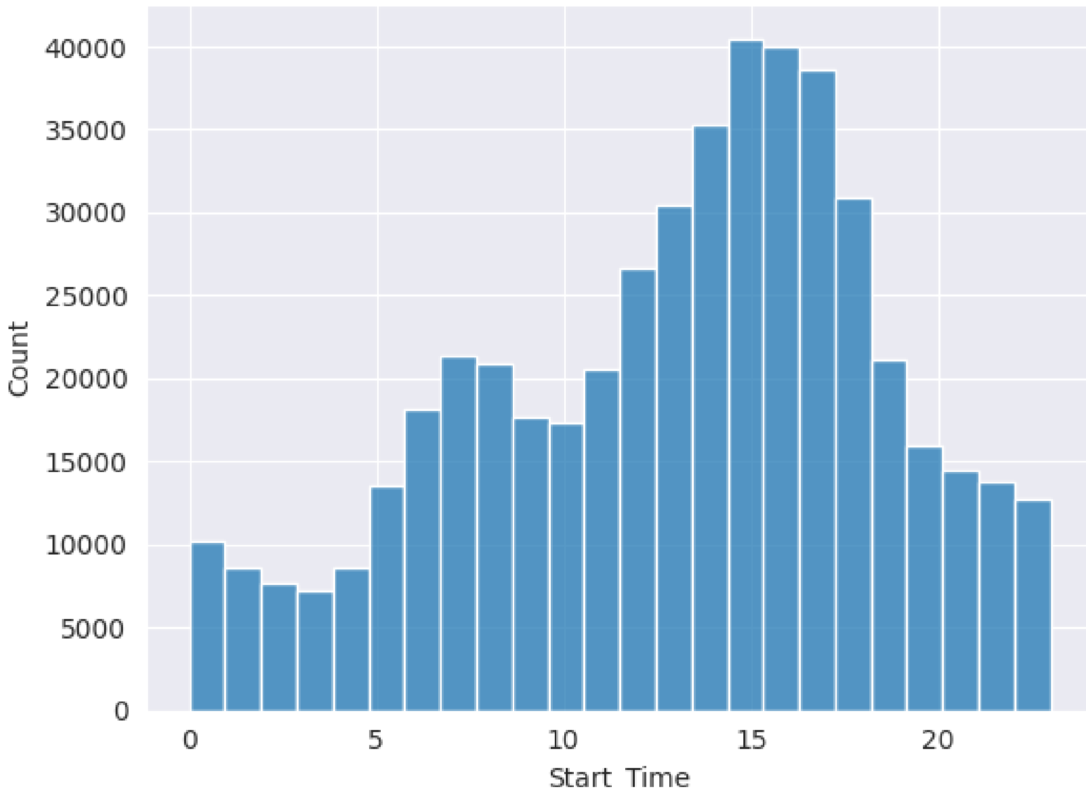
```
[27]: <AxesSubplot: xlabel='Start_Time', ylabel='Count'>
```



5. Friday

[28]:
```
friday_start_time = dataFrame.Start_Time[dataFrame.Start_Time.dt.dayofweek == 4]

sns.histplot(friday_start_time.dt.hour, stat = "count", bins = 24)
```

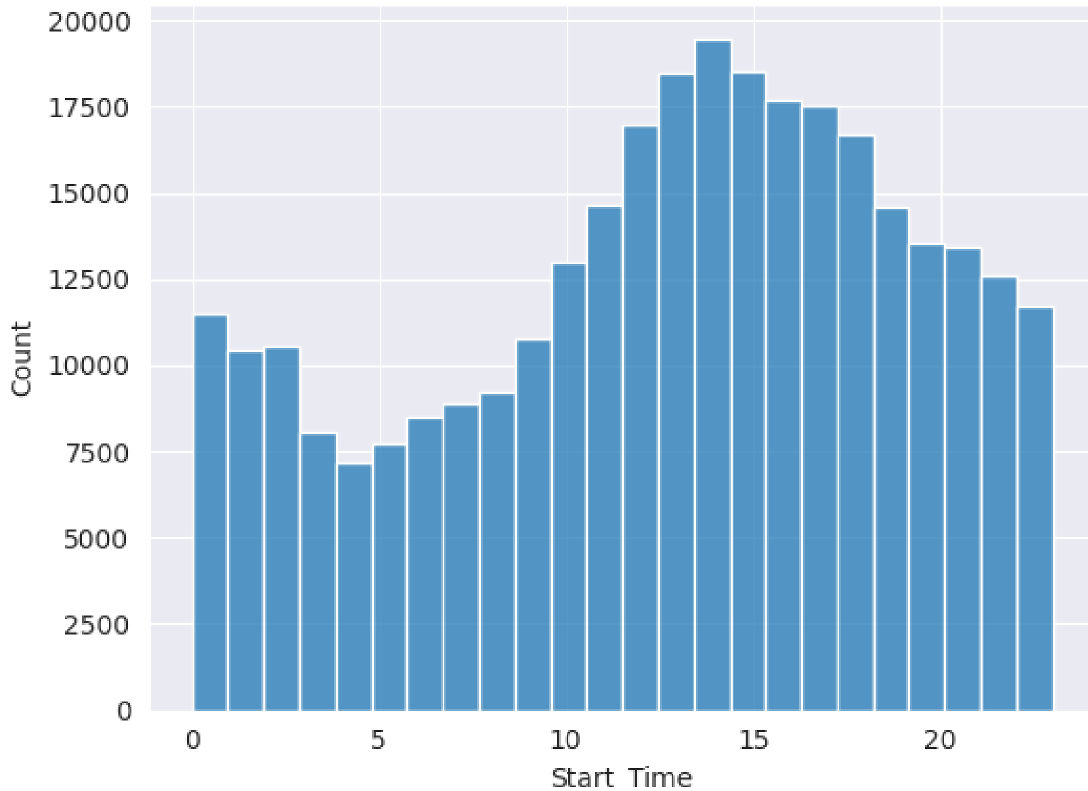
[28]: <AxesSubplot: xlabel='Start_Time', ylabel='Count'>



6. Saturday

[29]:
```
saturday_start_time = dataFrame.Start_Time[dataFrame.Start_Time.dt.dayofweek ==␣
↪5]

sns.histplot(saturday_start_time.dt.hour, stat = "count", bins = 24)
```

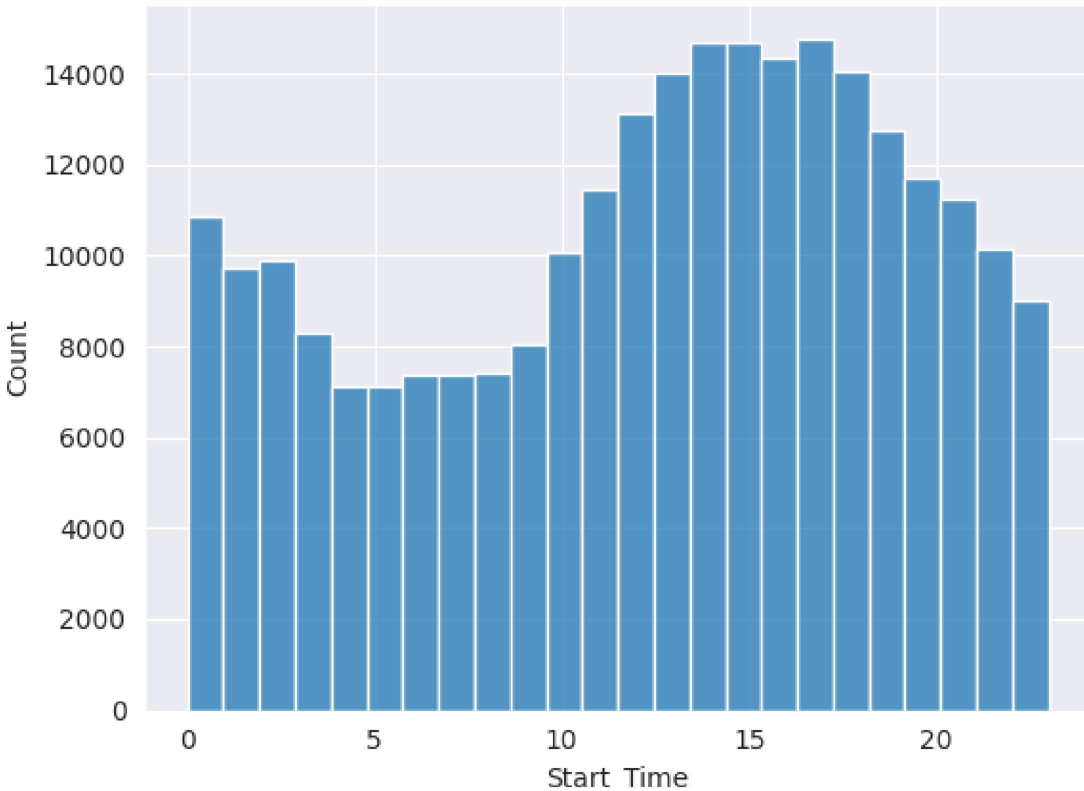
[29]: <AxesSubplot: xlabel='Start_Time', ylabel='Count'>



7. Sunday

[30]:
```
sunday_start_time = dataFrame.Start_Time[dataFrame.Start_Time.dt.dayofweek == 6]

sns.histplot(sunday_start_time.dt.hour, stat = "count", bins = 24)
```

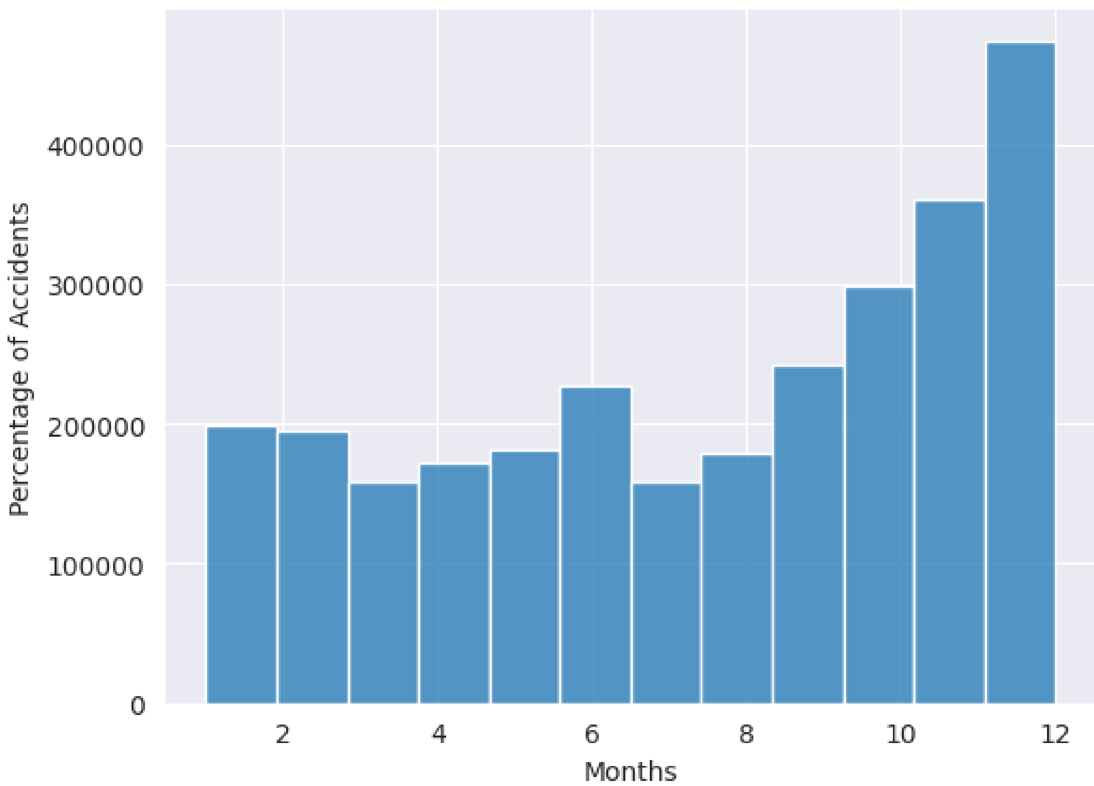
[30]: <AxesSubplot: xlabel='Start_Time', ylabel='Count'>



1.3.7 Months

[31]:
```
month_plot = sns.histplot(dataFrame.Start_Time.dt.month, stat = "count", bins =
↪12)

month_plot.set(xlabel = "Months", ylabel="Percentage of Accidents")
```

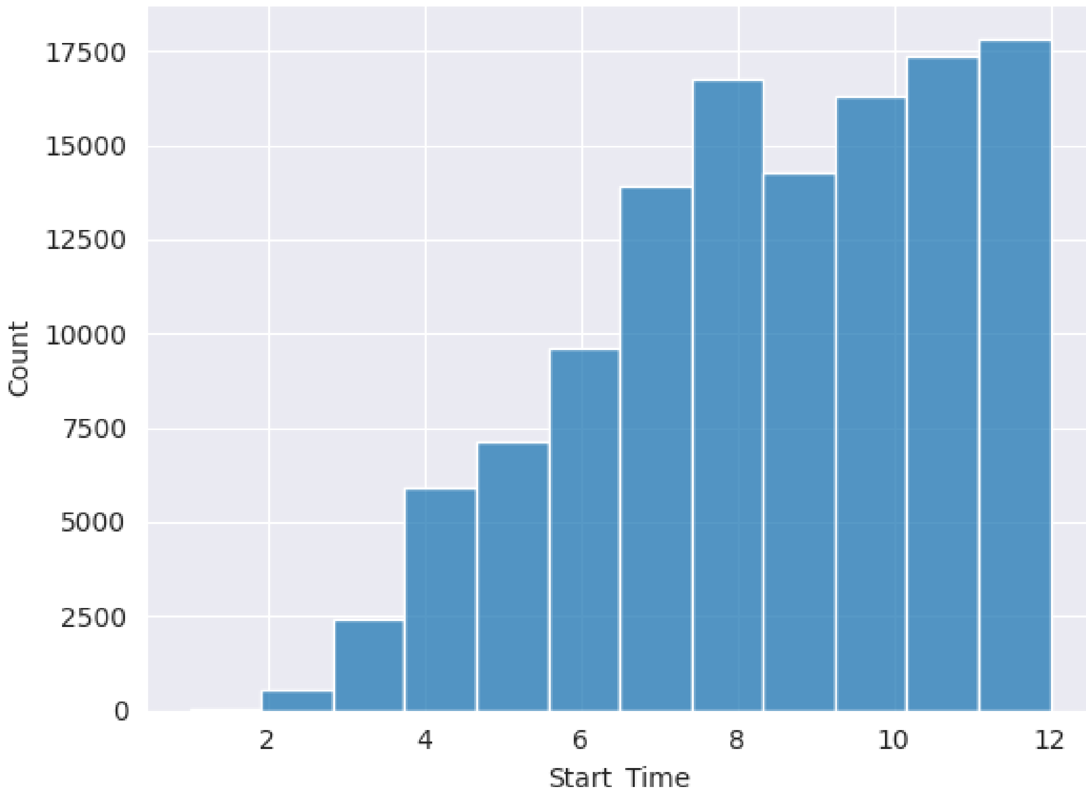
[31]: [Text(0.5, 0, 'Months'), Text(0, 0.5, 'Percentage of Accidents')]



1.3.8 Year 2016

[32]:
```
df_2016 = dataFrame.Start_Time[dataFrame.Start_Time.dt.year == 2016]

sns.histplot(df_2016.dt.month, bins = 12, stat = 'count')
```

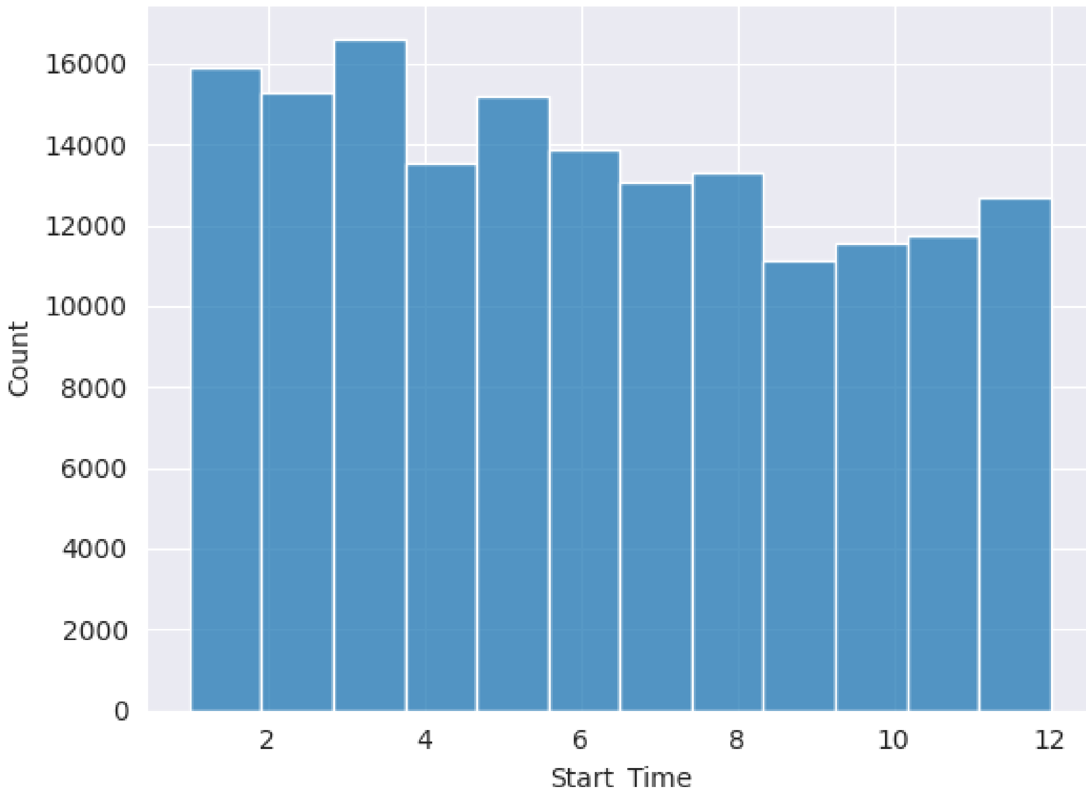
[32]:
```
<AxesSubplot: xlabel='Start_Time', ylabel='Count'>
```



1.3.9 Year 2017

[33]:
```
df_2017 = dataFrame.Start_Time[dataFrame.Start_Time.dt.year == 2017]

sns.histplot(df_2017.dt.month, bins = 12, stat = 'count')
```

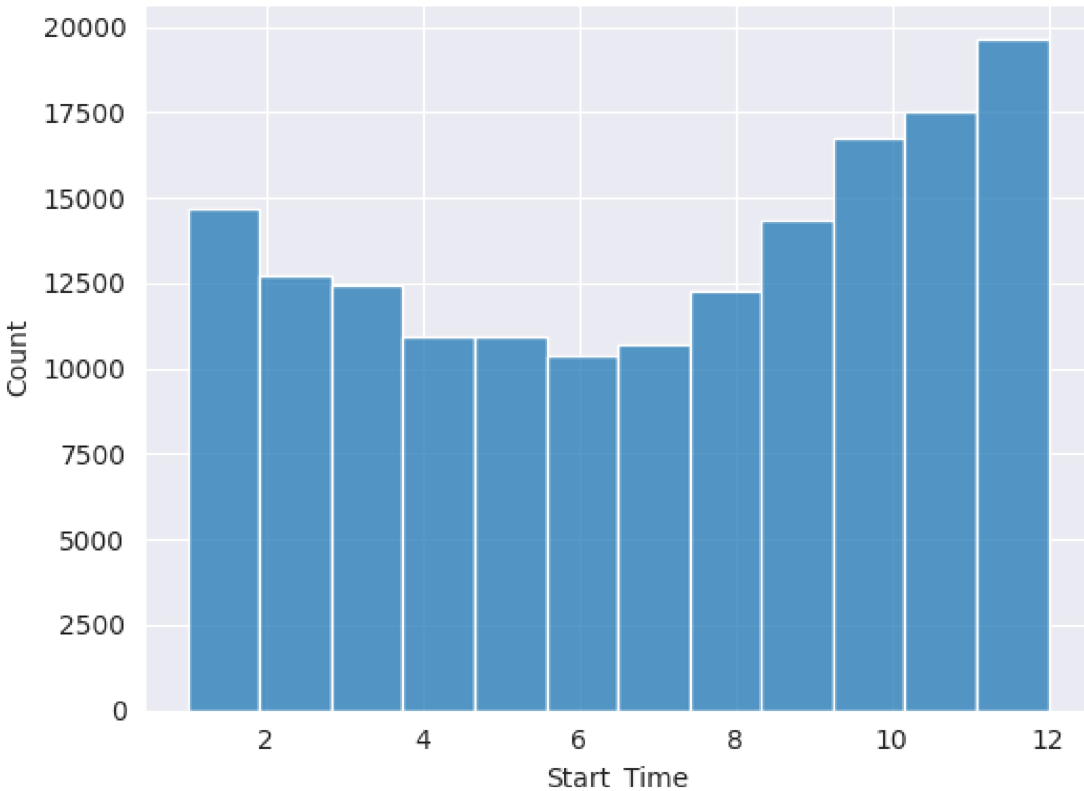
[33]: <AxesSubplot: xlabel='Start_Time', ylabel='Count'>



1.3.10 Year 2018

[34]:
```
df_2018 = dataFrame.Start_Time[dataFrame.Start_Time.dt.year == 2018]

sns.histplot(df_2018.dt.month, bins = 12, stat = 'count')
```

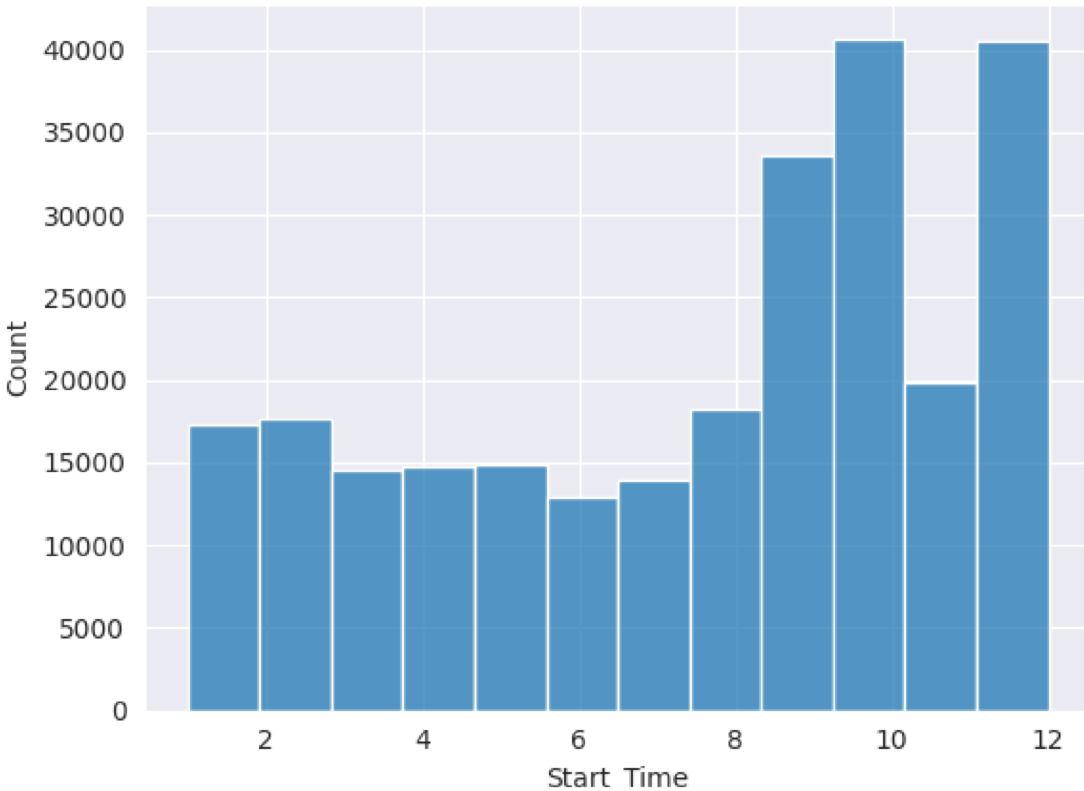
[34]: <AxesSubplot: xlabel='Start_Time', ylabel='Count'>



1.3.11 Year 2019

[35]:
```
df_2019 = dataFrame.Start_Time[dataFrame.Start_Time.dt.year == 2019]

sns.histplot(df_2019.dt.month, bins = 12, stat = 'count')
```

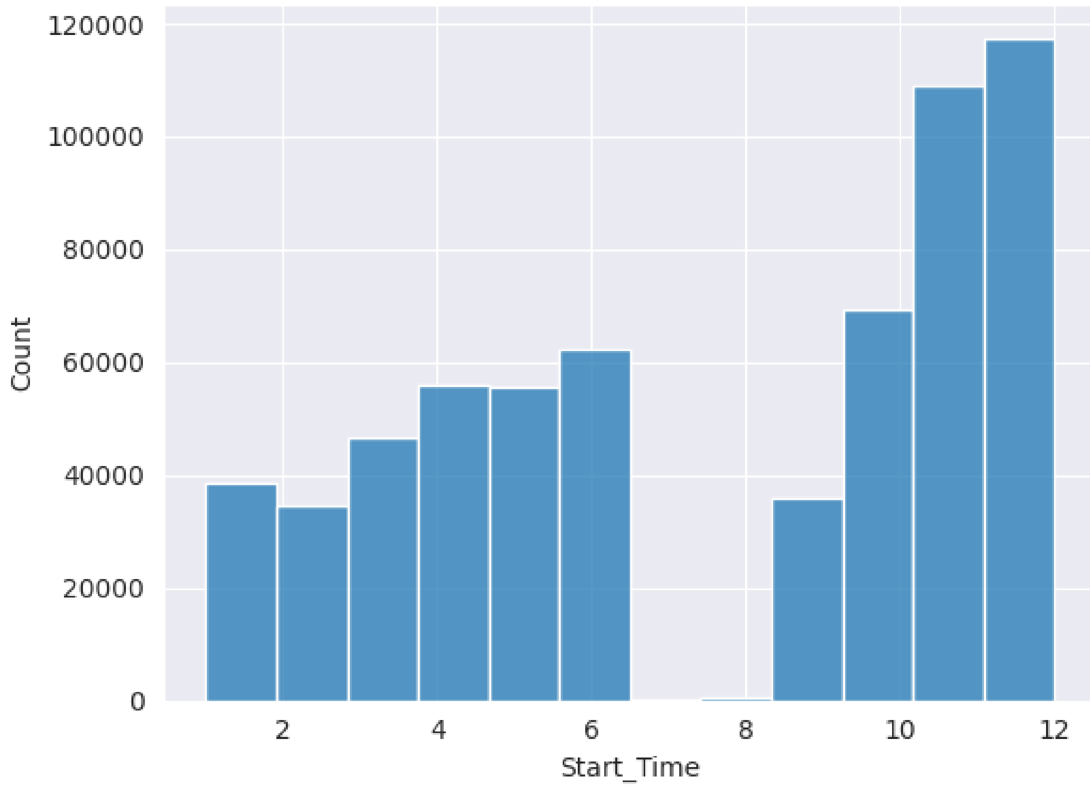
[35]: <AxesSubplot: xlabel='Start_Time', ylabel='Count'>



1.3.12 Year 2020

[36]:
```
df_2020 = dataFrame.Start_Time[dataFrame.Start_Time.dt.year == 2020]

sns.histplot(df_2020.dt.month, bins = 12, stat = 'count')
```

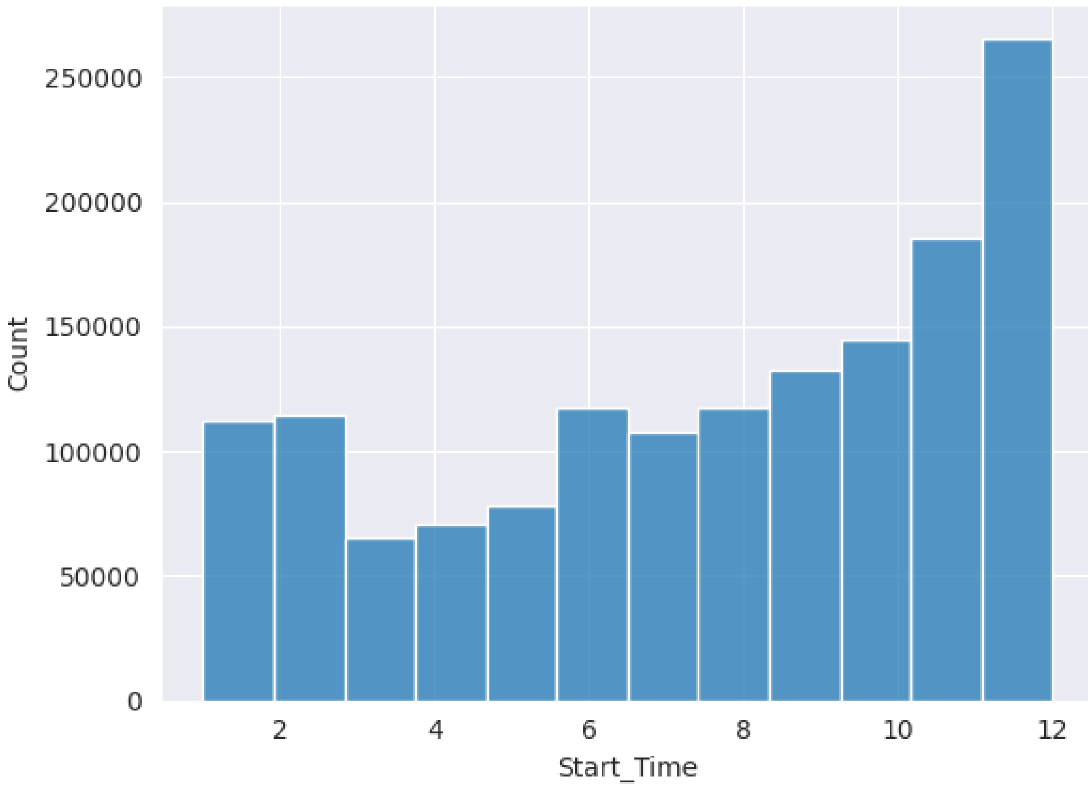
[36]: <AxesSubplot: xlabel='Start_Time', ylabel='Count'>



1.3.13 Year 2021

[37]:
```
df_2021 = dataFrame.Start_Time[dataFrame.Start_Time.dt.year == 2021]

sns.histplot(df_2021.dt.month, bins = 12, stat = 'count')
```

[37]: <AxesSubplot: xlabel='Start_Time', ylabel='Count'>



1.3.14 3. Start_Lat and Start_Lng Analysis

[38]:
```
dataFrame.Start_Lat
```

[38]:
```
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
```

[39]:
```
dataFrame.Start_Lng
```

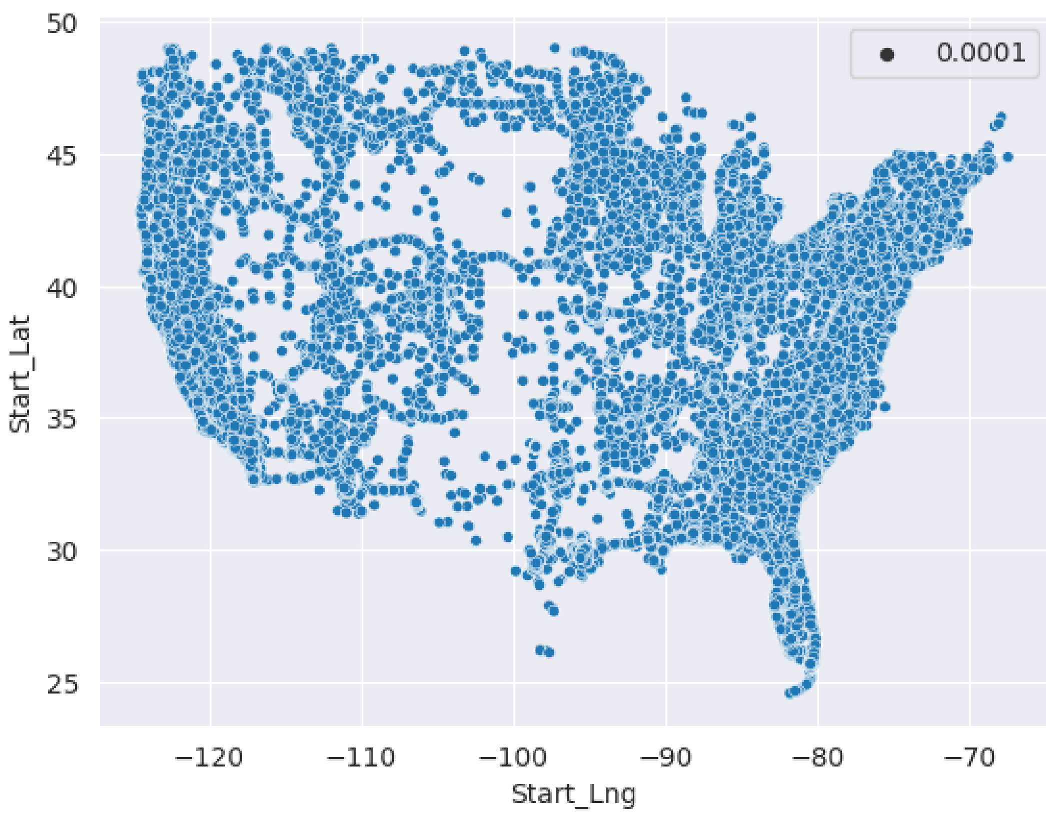
[39]:
```
0          -83.092860
1          -84.062800
```

```
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
```

[40]:
```
sample_df = dataFrame.sample(int(0.1 * len(dataFrame)))
```

[41]:
```
sns.scatterplot(x = sample_df.Start_Lng, y = sample_df.Start_Lat, size = 0.0001)
```

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



[42]:
```
import folium
from folium import plugins
from folium.plugins import HeatMap
```

[61]:
```python
lat, lon = sample_df.Start_Lat, sample_df.Start_Lng

lat, lon

zipped_lat_lon = list(zip(lat, lon))

zipped_lat_lon[:20]
```

[61]:
```
[(39.398677, -77.431839),
 (39.76324, -105.00279),
 (40.075447, -75.35700600000001),
 (38.628134, -120.208293),
 (35.206744, -89.767424),
 (38.404768, -121.48273799999998),
 (25.858694, -80.207995),
 (35.79196500000001, -78.543523),
 (43.0077, -76.13209),
 (35.37717, -97.524),
 (33.480193, -112.220653),
 (34.025824, -117.781091),
 (42.2315, -83.58454),
 (35.17897, -80.89097),
 (35.097383, -81.677012),
 (40.75777, -73.73823),
 (38.985442, -94.705378),
 (40.749672, -73.61585600000002),
 (28.567167, -81.286008),
 (34.016901000000004, -117.467225)]
```

[60]:
```python
map = folium.Map()

HeatMap(zipped_lat_lon).add_to(map)

map
```

[60]:
```
<folium.folium.Map at 0x7f0404ad4fa0>
```

1.3.15 4. Temperature

[45]:
```python
dataFrame['Temperature(F)']
```

[45]:
```
0          42.1
1          36.9
2          36.0
3          39.0
4          37.0
           …
```

```
2845337    86.0
2845338    70.0
2845339    73.0
2845340    71.0
2845341    79.0
Name: Temperature(F), Length: 2845342, dtype: float64
```

[46]:
```
top_accident_temp = dataFrame['Temperature(F)'].value_counts()

top_accident_temp
```

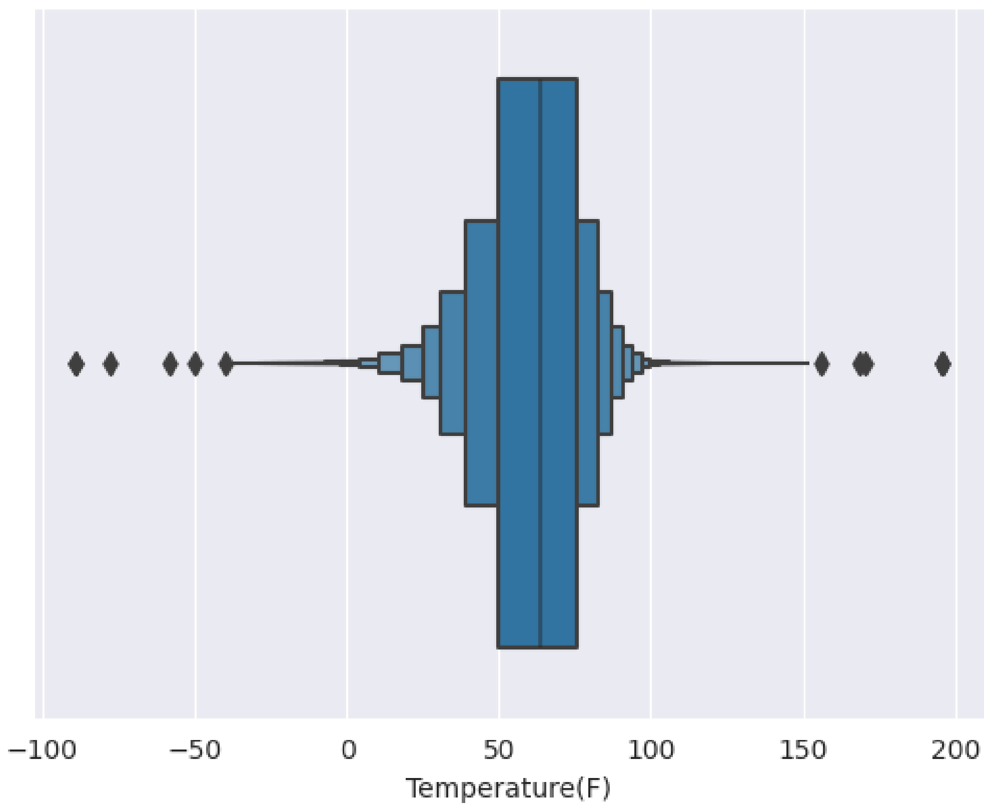
[46]:
```
 73.0     64505
 77.0     63575
 75.0     60534
 72.0     59681
 68.0     58557
          …
 109.8        1
-9.8          1
 170.6        1
 107.2        1
 99.1         1
Name: Temperature(F), Length: 788, dtype: int64
```

[47]:
```
top_accident_temp[:3]
```

[47]:
```
73.0     64505
77.0     63575
75.0     60534
72.0     59681
68.0     58557
         …
30.2      1258
104.0     1217
28.4      1157
93.9      1128
3.0       1110
Name: Temperature(F), Length: 159, dtype: int64
```

[48]:
```
sns.boxenplot(x = dataFrame["Temperature(F)"])
```

[48]:
```
<AxesSubplot: xlabel='Temperature(F)'>
```



1.3.16 5. Visibility

[49]:
```
dataFrame["Visibility(mi)"]
```

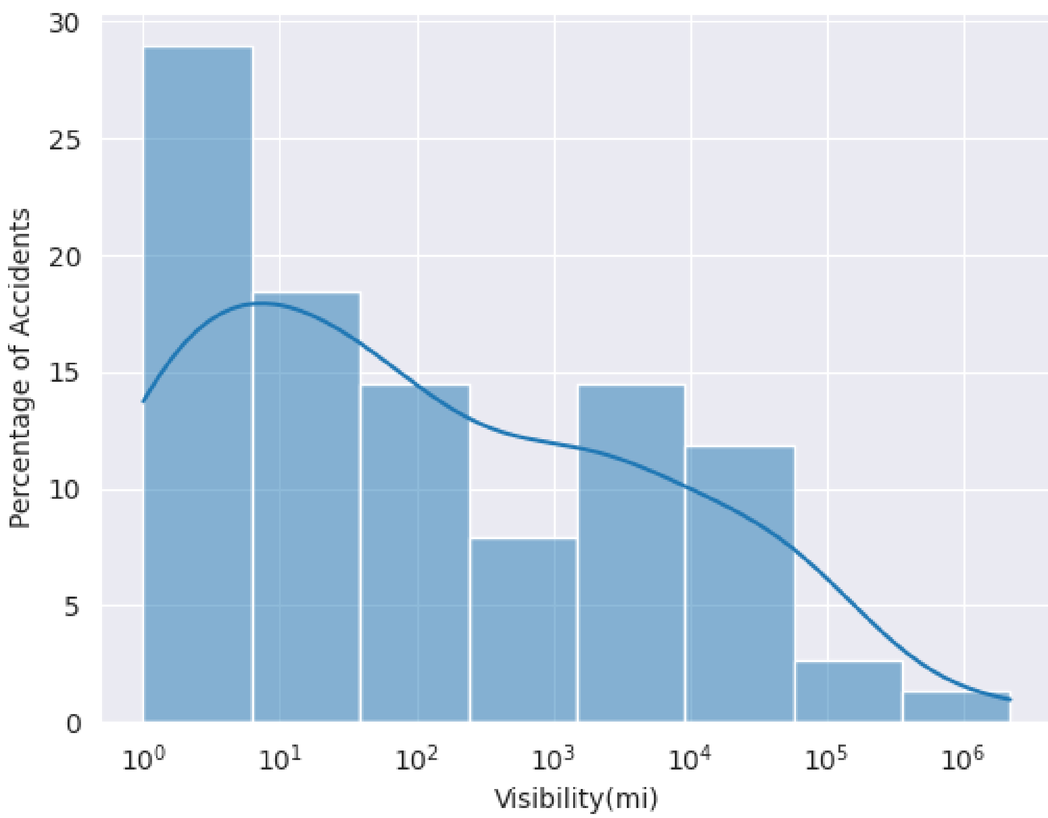
[49]:
```
0          10.0
1          10.0
2          10.0
3          10.0
4          10.0
           …
2845337    10.0
2845338    10.0
2845339    10.0
2845340    10.0
2845341     7.0
Name: Visibility(mi), Length: 2845342, dtype: float64
```

[50]:
```
accident_visibility = dataFrame["Visibility(mi)"].value_counts()

accident_visibility
```

[50]:
```
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
```

[51]:
```
visibility_plot = sns.histplot(accident_visibility, kde = True, stat =␣
 ↪'percent', log_scale = True)

visibility_plot.set(ylabel = "Percentage of Accidents")
```

[51]:
```
[Text(0, 0.5, 'Percentage of Accidents')]
```



1.3.17 6. Weather Conditions

[52]:
```
dataFrame.columns
```

[52]:
```
Index(['ID', 'Severity', 'Start_Time', 'End_Time', 'Start_Lat', 'Start_Lng',
       'End_Lat', 'End_Lng', 'Distance(mi)', 'Description', 'Number', 'Street',
       'Side', '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')
```

[53]:
```
dataFrame.Weather_Condition
```

[53]:
```
0              Light Rain
1              Light Rain
2                Overcast
3                Overcast
4              Light Rain
                  …
2845337              Fair
2845338              Fair
2845339     Partly Cloudy
2845340              Fair
2845341              Fair
Name: Weather_Condition, Length: 2845342, dtype: object
```

[54]:
```
accident_weather_condition = dataFrame.Weather_Condition.value_counts()

accident_weather_condition
```

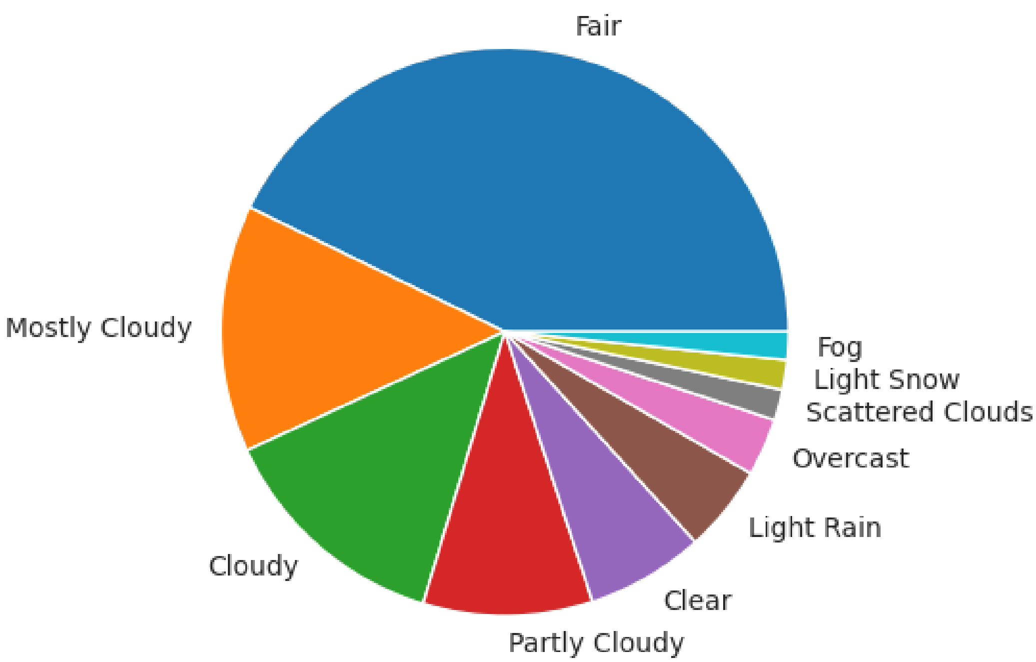
[54]:
```
Fair                            1107194
Mostly Cloudy                    363959
Cloudy                           348767
Partly Cloudy                    249939
Clear                            173823
                                  …
Sleet / Windy                         1
Mist / Windy                          1
Blowing Sand                          1
Heavy Freezing Rain                   1
Thunder and Hail / Windy              1
Name: Weather_Condition, Length: 127, dtype: int64
```

[55]:
```
top_accident_weathers = accident_weather_condition[:10]

top_accident_weathers
```

[55]:
```
Fair                   1107194
Mostly Cloudy           363959
Cloudy                  348767
Partly Cloudy           249939
Clear                   173823
Light Rain              128403
Overcast                 84882
Scattered Clouds         45132
Light Snow               43752
Fog                      41226
Name: Weather_Condition, dtype: int64
```

[56]:
```
top_accident_weathers.plot.pie().set(xlabel="", ylabel="")
```

[56]:
```
[Text(0.5, 0, ''), Text(0, 0.5, '')]
```



1.4 Summary and Conclusion

1. No data from New York.
2. No. of accidents decreases exponentially for almost every city.

3. Less than 5% cities have more than 1000 yearly accidents.
4. Over 1100 cities have reported just one accident.
5. A high percentage of accidents are occuring between 15:00 to 17:00. Reason could be people `returning from` their work.
6. Next highest percentage of accidents are occurring between 08:00 to 09:00. Reason could be people leaving their homes `for` work.
7. Highest frequency of accidents are all happening around the `same hour(15:00 to 17:00)` in both cases.
8. Weekends have 2nd highest frequency of accidents during `01:00 to 03:00` which is different from the weekdays.
9. More accidents are occuring during the end of the year. Reasons could be low visibility in winters or people partying during festivals like new year, christmas, etc.
10. Data seems to be inconsistent and inaccurate in the year 2016. Reason could be that data was being collected from various sources at that time.
11. There are almost no accidents in the month of July and August. Reason could be the lockdown due to corona. Still the data seems inaccurate and the reason could be irregularity in keeping data during covid.
12. Accidents are consistently more around the end of all years.
13. HeatMap shows that many accidents are happening all over US.
14. Some regions have less amount of accidents reported. Reason could be less population.
15. Many accidents happen in the cold weather.
16. Accidents have a higher frequency around 50F to 70F.
17. Higher percentage of accodents happened when the visibility was very low. Fog ,Smog, Heavy Rain and other visibility declining factors could be the reason.
18. Seems more numbers of accidents happen in a Fair weather condition.
19. Other than that, Mostly Cloudy and Cloudy weather conditions have more rate of accidents.