US Accident Exploratory Data Analysis

Importing Liberaries

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
```

Reading the datasets using Pandas

[2]:
```
df=pd.read_csv(r'C:\Users\ANKITA UPADHAYAY\Downloads\archive (8)\US_Accidents_March23.csv')
```

*[4]:
```
df
```

[4]:

	ID	Source	Severity	Start_Time	End_Time	Start_Lat	Start_Lng	End_Lat	End_Lng	Distance(mi)	...	Roundabout	Station	Stop	Traffic_Calming
0	A-1	Source2	3	2016-02-08 05:46:00	2016-02-08 11:00:00	39.865147	-84.058723	NaN	NaN	0.010	...	False	False	False	False
1	A-2	Source2	2	2016-02-08 06:07:59	2016-02-08 06:37:59	39.928059	-82.831184	NaN	NaN	0.010	...	False	False	False	False
2	A-3	Source2	2	2016-02-08 06:49:27	2016-02-08 07:19:27	39.063148	-84.032608	NaN	NaN	0.010	...	False	False	False	False
3	A-4	Source2	3	2016-02-08 07:23:34	2016-02-08 07:53:34	39.747753	-84.205582	NaN	NaN	0.010	...	False	False	False	False
4	A-5	Source2	2	2016-02-08 07:39:07	2016-02-08 08:09:07	39.627781	-84.188354	NaN	NaN	0.010	...	False	False	False	False
7728393	A-7777761	Source1	2	2019-08-23 18:52:06	2019-08-23 19:21:31	34.133930	-117.230920	34.13736	-117.23934	0.537	...	False	False	False	False

7728394 rows × 46 columns

[182]:
```
#Listing the columns of datasets
df.columns
```

[182]:
```
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')
```

[183]:
```
# Shape of the datasets
df.shape
```

[183]:
```
(7728394, 46)
```

[184]:
```
# Displaying the first four rows.
df.head()
```

[184]:

	ID	Source	Severity	Start_Time	End_Time	Start_Lat	Start_Lng	End_Lat	End_Lng	Distance(mi)	...	Roundabout	Station	Stop	Traffic_Calming	Traffic_Signal
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	False	False	False	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	False	False	False	True

5 rows × 46 columns

[185]:
```
# tail is used to display last four rows
df.tail()
```

[185]:

	ID	Source	Severity	Start_Time	End_Time	Start_Lat	Start_Lng	End_Lat	End_Lng	Distance(mi)	...	Roundabout	Station	Stop	Traffic_Calming
7728389	A-7777757	Source1	2	2019-08-23 18:03:25	2019-08-23 18:32:01	34.00248	-117.37936	33.99888	-117.37094	0.543	...	False	False	False	False
7728390	A-7777758	Source1	2	2019-08-23 19:11:30	2019-08-23 19:38:23	32.76696	-117.14806	32.76555	-117.15363	0.338	...	False	False	False	False
7728391	A-7777759	Source1	2	2019-08-23 19:00:21	2019-08-23 19:28:49	33.77545	-117.84779	33.77740	-117.85727	0.561	...	False	False	False	False
7728392	A-7777760	Source1	2	2019-08-23 19:00:21	2019-08-23 19:29:42	33.99246	-118.40302	33.98311	-118.39565	0.772	...	False	False	False	False
7728393	A-7777761	Source1	2	2019-08-23 18:52:06	2019-08-23 19:21:31	34.13393	-117.23092	34.13736	-117.23934	0.537	...	False	False	False	False

5 rows × 46 columns

[186]:
```
# It is used to display some detail information about datasets.
df.info()
```

Collapse Output

```
<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             datetime64[ns]
 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), datetime64[ns](1), float64(12), int64(1), object(19)
memory usage: 2.0+ GB
```

There is no any column which is empty.

[187]:
```
# describe - It is used to display statistical data of datasets.
df.describe()
```

[187]:

	Severity	Start_Time	Start_Lat	Start_Lng	End_Lat	End_Lng	Distance(mi)	Temperature(F)	Wind_Chill(F)	Humidity(%)	Pressu
count	7.728394e+06	6985228	7.728394e+06	7.728394e+06	4.325632e+06	4.325632e+06	7.728394e+06	7.564541e+06	5.729375e+06	7.554250e+06	7.58771
mean	2.212384e+00	2020-03-21 21:49:59.056443136	3.620119e+01	-9.470255e+01	3.626183e+01	-9.572557e+01	5.618423e-01	6.166329e+01	5.825105e+01	6.483104e+01	2.9538
min	1.000000e+00	2016-01-14 20:18:33	2.455480e+01	-1.246238e+02	2.456601e+01	-1.245457e+02	0.000000e+00	-8.900000e+01	-8.900000e+01	1.000000e+00	0.0000
25%	2.000000e+00	2018-09-17 02:20:31.750000128	3.339963e+01	-1.172194e+02	3.346207e+01	-1.177543e+02	0.000000e+00	4.900000e+01	4.300000e+01	4.800000e+01	2.9370
50%	2.000000e+00	2020-06-23 13:37:23.500000	3.582397e+01	-8.776662e+01	3.618349e+01	-8.802789e+01	3.000000e-02	6.400000e+01	6.200000e+01	6.700000e+01	2.9860
75%	2.000000e+00	2021-10-28 15:44:07	4.008496e+01	-8.035368e+01	4.017892e+01	-8.024709e+01	4.640000e-01	7.600000e+01	7.500000e+01	8.400000e+01	3.0030
max	4.000000e+00	2023-03-31 23:30:00	4.900220e+01	-6.711317e+01	4.907500e+01	-6.710924e+01	4.417500e+02	2.070000e+02	2.070000e+02	1.000000e+02	5.8630
std	4.875313e-01	NaN	5.076079e+00	1.739176e+01	5.272905e+00	1.810793e+01	1.776811e+00	1.901365e+01	2.238983e+01	2.282097e+01	1.0061

[188]:
```
# No of numeric data into datasets
numerics = ['int16','int32','int64','float16','float32','float64']
numeric_df = df.select_dtypes(include=numerics)
len(numeric_df.columns)
```

[188]:
```
13
```

[195]:
```
# To count null values of each column and sorting them in descending order
null_count=pd.isnull(df).sum().sort_values(ascending=False )
```

[196]:
```
null_count
```

[196]:
```
End_Lng                  3402762
End_Lat                  3402762
Precipitation(in)        2203586
Wind_Chill(F)            1999019
Start_Time                743166
Wind_Speed(mph)           571233
Visibility(mi)            177098
Wind_Direction            175206
Humidity(%)               174144
Weather_Condition         173459
Temperature(F)            163853
Pressure(in)              140679
Weather_Timestamp         120228
Civil_Twilight             23246
Astronomical_Twilight      23246
Nautical_Twilight          23246
Sunrise_Sunset             23246
Airport_Code               22635
Street                     10869
Timezone                    7808
Zipcode                     1915
City                         253
Description                    5
Distance(mi)                   0
Start_Lng                      0
ID                             0
Source                         0
Severity                       0
End_Time                       0
Start_Lat                      0
County                         0
Amenity                        0
```

[11]:
```
# Filter out columns with zero null values
non_zero_null_counts = null_count[null_count != 0]
```

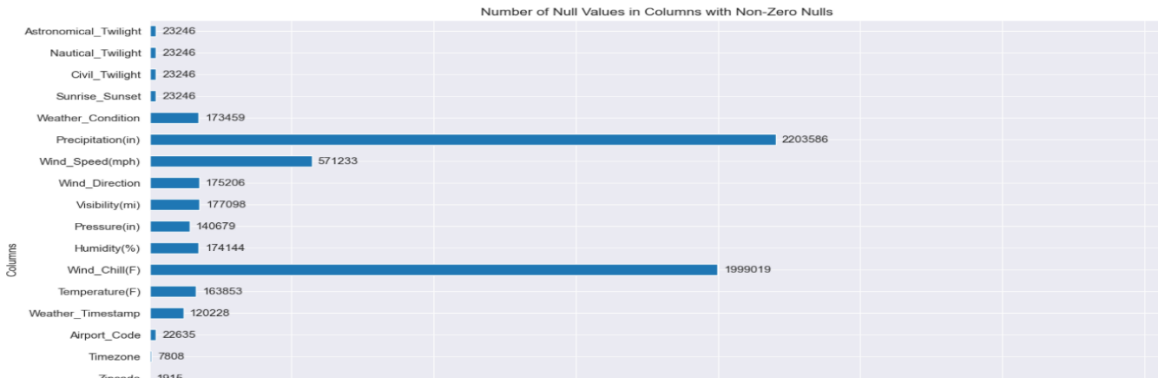
[12]:
```
non_zero_null_counts
```

[12]:
```
End_Lat                  3402762
End_Lng                  3402762
Description                    5
Street                     10869
City                         253
Zipcode                     1915
Timezone                    7808
Airport_Code               22635
Weather_Timestamp         120228
Temperature(F)            163853
Wind_Chill(F)            1999019
Humidity(%)               174144
Pressure(in)              140679
Visibility(mi)            177098
Wind_Direction            175206
Wind_Speed(mph)           571233
Precipitation(in)        2203586
Weather_Condition         173459
Sunrise_Sunset             23246
Civil_Twilight             23246
Nautical_Twilight          23246
Astronomical_Twilight      23246
dtype: int64
```

[197]:
```
# Applying dark grid in the background for graph
sns.set_style("darkgrid")
```

[198]:
```
# Plot the horizontal bar chart
plt.figure(figsize=(15, 10))  # Increase the figure size for better visibility
ax = non_zero_null_counts.plot(kind='barh')
plt.title('Number of Null Values in Columns with Non-Zero Nulls')
```

```
plt.title('Number of Null Values in Columns with Non-Zero Nulls')
plt.ylabel('Columns')
plt.xlabel('Number of Null Values')
# Add annotations to the bars
for i in ax.patches:
    ax.annotate(str(i.get_width()),
                (i.get_width(), i.get_y() + i.get_height() / 2), ha='left', va='center', xytext=(5, 0), textcoords='offset points')
```



Exploratory Data Analysis

Analyzing the column values

1. State
2. Cities
3. Start Time
4. Start Lat , Start Lng

[199]:
```
df.columns
```

[199]:
```
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')
```

1. State

[200]:
```
df.State
```

[200]:
```
0          OH
1          OH
2          OH
3          OH
4          OH
           ..
7728389    CA
7728390    CA
7728391    CA
7728392    CA
7728393    CA
Name: State, Length: 7728394, dtype: object
```

[220]:
```
# Displaying the unique state in the US
state = df.State.unique()
```

[170]:
```
len(state)
```

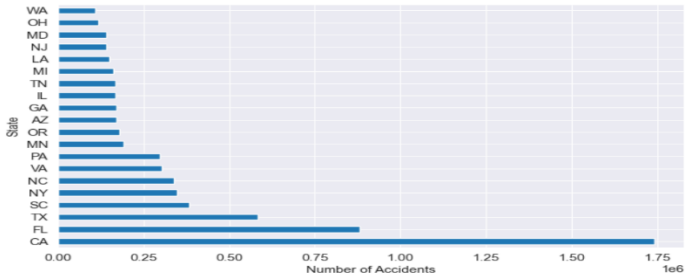
[170]:
```
49
```

[172]:
```
# No of accidents in the state
state_by_Accidents = df.State.value_counts()
state_by_Accidents
```

[172]:
```
State
CA    1741433
FL     880192
TX     582837
SC     382557
NY     347960
NC     338199
VA     303301
PA     296620
MN     192084
OR     179660
AZ     170609
GA     169234
IL     168958
TN     167388
MI     162191
LA     149701
NJ     140719
MD     140417
```

[202]:
```
#top 20 state where maximum takes place
ss=state_by_Accidents[:20]
```

[203]:
```
# plotting the graph state vs no of accidents
import matplotlib
ss.plot(kind='barh')
plt.ylabel('State')
plt.xlabel('Number of Accidents')
matplotlib.rcParams['figure.figsize']=(8,5)
```



2. City

[16]:
```
df.City
```

[16]:
```
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
```

[150]:
```
cities=df.City.unique()
```

[151]:
```
len(cities)
```

[151]:
```
13679
```

[19]:
```
# No of accidents in the cities
cities_by_Accidents = df.City.value_counts()
cities_by_Accidents
```

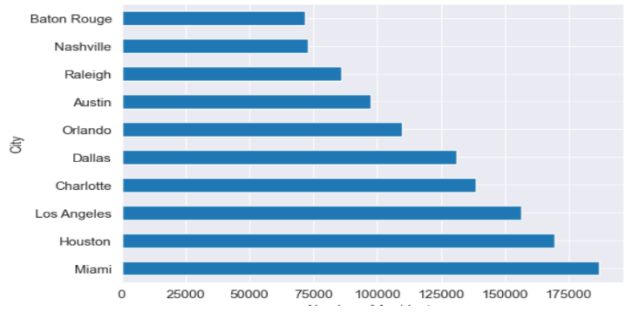
[19]:
```
City
Miami            186917
Houston          169609
Los Angeles      156491
Charlotte        138652
Dallas           130939
                  ...
Rapid River           1
Cat Spring            1

Marfa                 1
Name: count, Length: 13678, dtype: int64
```

[204]:
```
#top 10 cities where maximum accidents takes place
cc=cities_by_Accidents[:10]
```

[22]:
```
cc.plot(kind='barh')
plt.ylabel('City')
plt.xlabel('Number of Accidents')
```

[22]:
```
Text(0.5, 0, 'Number of Accidents')
```



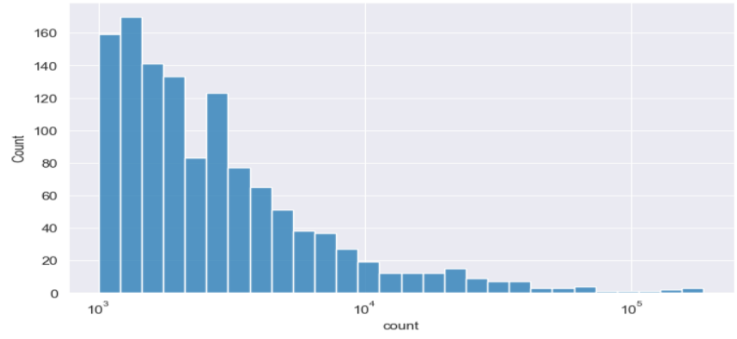
:
```
high_accident_cities = cities_by_Accidents[cities_by_Accidents>1000]
```

:
```
len(high_accident_cities)/len(cities)
```

:
```
0.08882228233057972
```

:
```
sns.histplot(high_accident_cities ,log_scale=True)
```

:
```
<Axes: xlabel='count', ylabel='Count'>
```



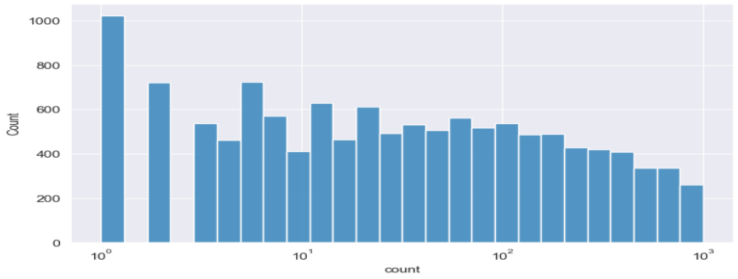
```
low_accident_cities = cities_by_Accidents[cities_by_Accidents<1000]
```

```
len(low_accident_cities)
```

```
12460
```

```
sns.histplot(low_accident_cities,log_scale=True )
```

```
<Axes: xlabel='count', ylabel='Count'>
```



 # Calculating the no of cities which have just 1 accident case.

```
a=cities_by_Accidents[cities_by_Accidents ==1]
```

```
len(a)
```

```
1023
```

3. Start Time

```
df.Start_Time
```

```
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
```

```
df['Start_Time'] = pd.to_datetime(df['Start_Time'], errors='coerce')
```

```
df['Start_Time']
```

```
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: datetime64[ns]
```

Hour wise Accident report

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

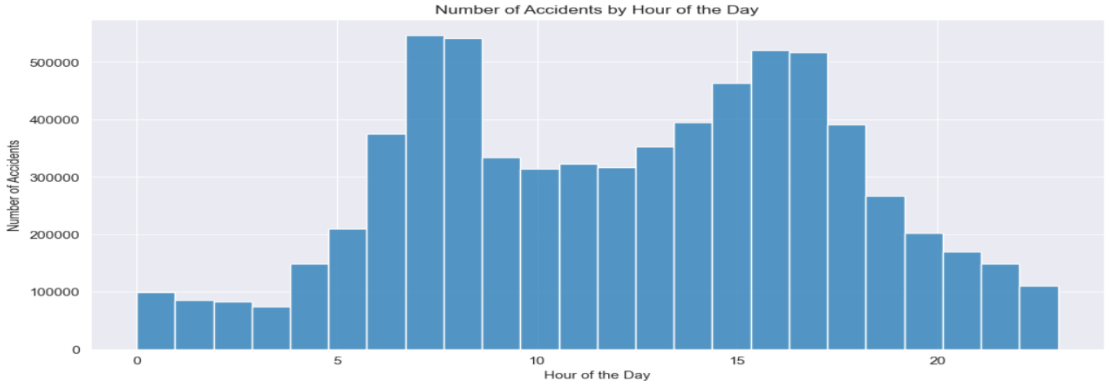
```
[36]: 0           5.0
      1           6.0
      2           6.0
      3           7.0
      4           7.0
                 ...
      7728389    18.0
      7728390    19.0
      7728391    19.0
      7728392    19.0
      7728393    18.0
      Name: Start_Time, Length: 7728394, dtype: float64
```

```
[247]: # Extract the hour from 'Start_Time'
       df['Hour'] = df['Start_Time'].dt.hour

       plt.figure(figsize=(12, 6))
       # Create a histogram for the hours of the day when accidents occur
       sns.histplot(df['Hour'], bins=24, kde=False)

       # Adding title and labels
       plt.title('Number of Accidents by Hour of the Day')
       plt.xlabel('Hour of the Day')
       plt.ylabel('Number of Accidents')
```

```
[247]: Text(0, 0.5, 'Number of Accidents')
```



Day of the week wise report

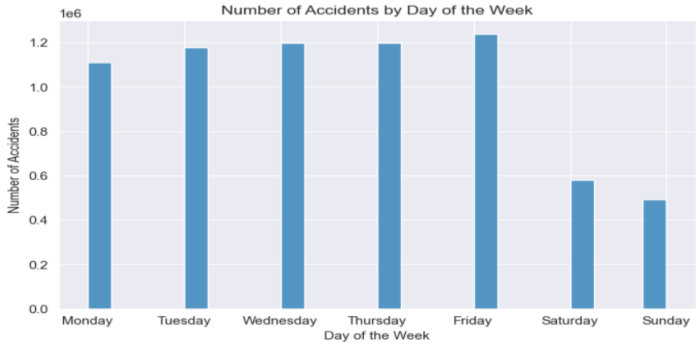
```
# Mostly accidents takes place during day of week
df.Start_Time.dt.dayofweek
```

```
# Mostly accidents takes place during day of week
df.Start_Time.dt.dayofweek
```

```
0          0.0
1          0.0
2          0.0
3          0.0
4          0.0
          ...
7728389    4.0
7728390    4.0
7728391    4.0
7728392    4.0
7728393    4.0
Name: Start_Time, Length: 7728394, dtype: float64
```

```
sns.histplot(df.Start_Time.dt.dayofweek,bins=24)
# Set the color for each bar
for i, bar in enumerate(ax.patches):
    bar.set_color(palette[i % 7])

# Adding title and labels
plt.title('Number of Accidents by Day of the Week')
plt.xlabel('Day of the Week')
plt.ylabel('Number of Accidents')
plt.xticks(range(7), ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday']);
```

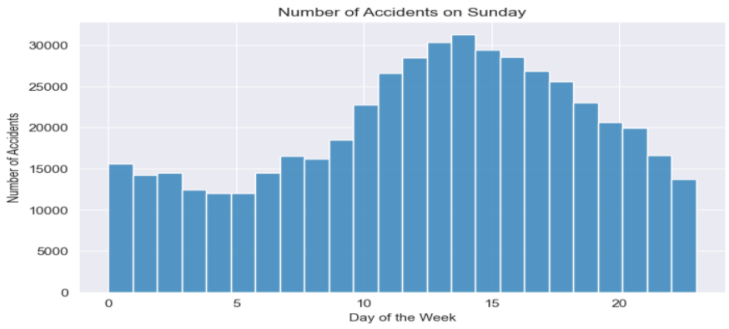


Is the distribution by hour the same as on weekends as on weekdays?

```
[241]: sunday=df.Start_Time[df.Start_Time.dt.dayofweek==6]
sns.histplot(sunday.dt.hour,bins=24)
# Adding title and labels
plt.title('Number of Accidents on Sunday')
plt.xlabel('Day of the Week')
plt.ylabel('Number of Accidents')
```

```
# Adding title and labels
plt.title('Number of Accidents on Sunday')
plt.xlabel('Day of the Week')
plt.ylabel('Number of Accidents')
```

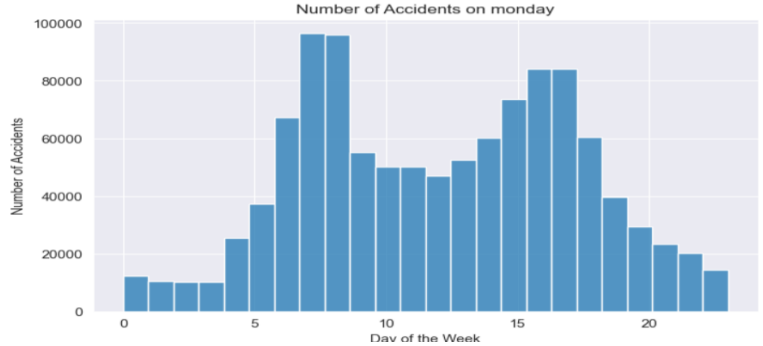
```
41]: Text(0, 0.5, 'Number of Accidents')
```



--> On sunday during afternoon the most accidents occured between 10 am to 3 am

```
# checking for mondays
monday=df.Start_Time[df.Start_Time.dt.dayofweek==0]
sns.histplot(monday.dt.hour,bins=24)
# Adding title and labels
plt.title('Number of Accidents on monday')
plt.xlabel('Day of the Week')
plt.ylabel('Number of Accidents')
```

```
Text(0, 0.5, 'Number of Accidents')
```



On monday, the accidents cases were between morning hours and evening hours.

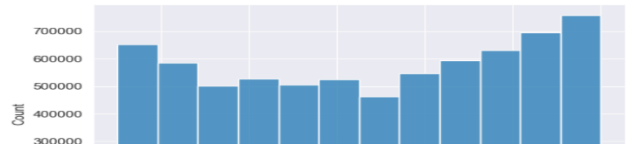
month wise Accident report

```
df.Start_Time.dt.month
```

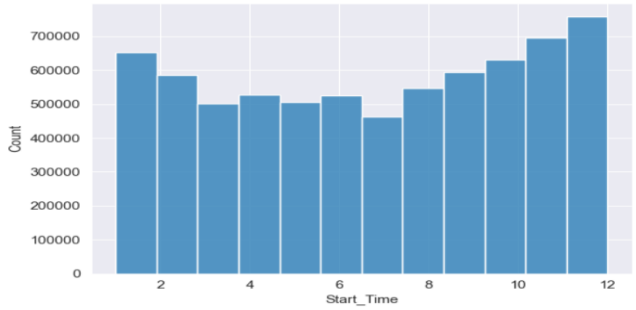
```
0           2.0
1           2.0
2           2.0
3           2.0
4           2.0
           ...
7728389     8.0
7728390     8.0
7728391     8.0
7728392     8.0
7728393     8.0
Name: Start_Time, Length: 7728394, dtype: float64
```

```
sns.histplot(df.Start_Time.dt.month,bins=12)
```

```
<Axes: xlabel='Start_Time', ylabel='Count'>
```



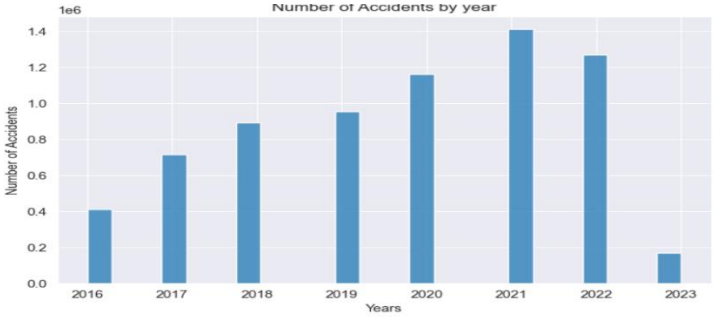

```
<Axes: xlabel='Start_Time', ylabel='Count'>
```



Year wise accidents report

```
years=df.Start_Time.dt.year
```

```
sns.histplot(df.Start_Time.dt.year,bins=24)
plt.title('Number of Accidents by year')
plt.xlabel('Years')
plt.ylabel('Number of Accidents')
```



4. Start Latitude and Longitude

```
77]: df.Start_Lat
```

```
77]: 0           39.865147
     1           39.928059
     2           39.063148
     3           39.747753
     4           39.627781
```


4. Start Latitude and Longitude


```
df.Start_Lat
```

```
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
```

```
df.Start_Lng
```

```
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
```

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

scatterplot graph where most cases from

```
sns.scatterplot(x=sample_df.Start_Lng,y=sample_df.Start_Lat,size = 0.001)
```

scatterplot graph where most cases from

```
sns.scatterplot(x=sample_df.Start_Lng,y=sample_df.Start_Lat,size = 0.001)
```

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

```
C:\Users\ANKITA UPADHAYAY\AppData\Local\Programs\Python\Python312\Lib\site-packages\IPython\core\events.py:82: UserWarning: Creating legend with loc="best" can be slow with large amounts of data.
  func(*args, **kwargs)
C:\Users\ANKITA UPADHAYAY\AppData\Local\Programs\Python\Python312\Lib\site-packages\IPython\core\pylabtools.py:170: UserWarning: Creating legend with loc="best" can be slow with large amounts of data.
  fig.canvas.print_figure(bytes_io, **kw)
```


```
import folium
```

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

```
(np.float64(39.865147), np.float64(-84.058723))
```

```
for x in df[['Start_Lat', 'Start_Lng']].sample(100).iterrows():
    print(x[1])
```

```
Start_Lat    29.486731
Start_Lng   -98.587097
Name: 1663325, dtype: float64
Start_Lat     34.015930
Start_Lng   -118.173027
Name: 497708, dtype: float64
Start_Lat    42.909562
Start_Lng   -85.676858
Name: 5140966, dtype: float64
Start_Lat     34.066069
Start_Lng   -117.484438
Name: 5525786, dtype: float64
Start_Lat     33.480438
Start_Lng   -112.860414
Name: 4692204, dtype: float64
Start_Lat    33.873170
Start_Lng   -78.836035
Name: 4027148, dtype: float64
Start_Lat    40.833054
Start_Lng   -73.860069
Name: 220014, dtype: float64
Start_Lat     34.031220
Start_Lng   -118.274582
Name: 6226910, dtype: float64
Start_Lat    39.172031
Start_Lng   -94.558289
```

```
list(df.Start_Lat)
```

```
[39.865147,
 39.928059000000001,
 39.063148,
 39.747753,
 39.627781,
 40.10059,
 39.758274,
 39.770382,
 39.778061,
 40.10059,
 39.952812,
 39.932709,
 39.737633,
 39.79076,
 39.972038,
 39.745888,
 39.748329,
 39.752174,
 39.740669,
 39.790703,
 40.052509,
 39.773346,
 39.628288,
 40.023487,
 39.761379,
 40.158024,
 39.733219,
 39.775303,
 39.789322,
 39.75872,
 40.081459,
 39.83321,
 40.042725,
 39.974415,
 39.994766,
```

```
from folium.plugins import HeatMap
```

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

```
map=folium.Map()
HeatMap(lat_lon_pairs).add_to(map)
map
```

```
# Displaying single record of accident with lattitude and longitude in map.
map=folium.Map()
marker=folium.Marker((lat,lon))
marker.add_to(map)
map
```

Questions

1. Are there more accidents in warmer or colder months?
2. Which states have the highest number of accidents?
3. Which cities have the highest number of accident reports?
4. Which day of the week has the most accidents?
5. Which month has the most accidents?
6. What is the yearly trend of accidents?
7. At what time do most accidents occur?
8. Is the hourly distribution of accidents the same on weekends as on weekdays?
9. How many cities are there where no of accidents are 1 ?

Answers-

1. More accidents take place in the winter months.
2. States with the highest number of accidents include California, Florida, and Texas.
3. Cities with the highest number of accidents are Miami and Houston.
4. More accidents occur on Thursdays and Wednesdays compared to other weekdays.
5. The number of accident cases decreases from January to June and then increases from July to December.
6. From 2016 to 2021, the number of accidents increased, but after 2021, it started to decrease.
7. The highest number of accidents occur between 6 AM to 10 AM and around 3 PM to 6 PM.
8. On weekends, the number of accidents is lower, whereas more accidents occur on Tuesdays, Wednesdays, and Thursdays.
9. There are total 1023 cities where no of accidents are 1.

[ ]:

[ ]: