```
import pandas as pd

data_filepath = "C:\\Users\\jayaraman\\Downloads\\archive (2)\\
US_Accidents_March23.csv"

df = pd.read_csv(data_filepath)
df.head(10)
```

```
     ID   Source  Severity           Start_Time
End_Time  \
0   A-1  Source2         3  2016-02-08 05:46:00  2016-02-08 11:00:00

1   A-2  Source2         2  2016-02-08 06:07:59  2016-02-08 06:37:59

2   A-3  Source2         2  2016-02-08 06:49:27  2016-02-08 07:19:27

3   A-4  Source2         3  2016-02-08 07:23:34  2016-02-08 07:53:34

4   A-5  Source2         2  2016-02-08 07:39:07  2016-02-08 08:09:07

5   A-6  Source2         3  2016-02-08 07:44:26  2016-02-08 08:14:26

6   A-7  Source2         2  2016-02-08 07:59:35  2016-02-08 08:29:35

7   A-8  Source2         3  2016-02-08 07:59:58  2016-02-08 08:29:58

8   A-9  Source2         2  2016-02-08 08:00:40  2016-02-08 08:30:40

9  A-10  Source2         3  2016-02-08 08:10:04  2016-02-08 08:40:04


   Start_Lat  Start_Lng  End_Lat  End_Lng  Distance(mi)  ...
Roundabout  \
0  39.865147 -84.058723      NaN      NaN          0.01  ...
False
1  39.928059 -82.831184      NaN      NaN          0.01  ...
False
2  39.063148 -84.032608      NaN      NaN          0.01  ...
False
3  39.747753 -84.205582      NaN      NaN          0.01  ...
False
4  39.627781 -84.188354      NaN      NaN          0.01  ...
False
5  40.100590 -82.925194      NaN      NaN          0.01  ...
False
6  39.758274 -84.230507      NaN      NaN          0.00  ...
False
7  39.770382 -84.194901      NaN      NaN          0.01  ...
False
8  39.778061 -84.172005      NaN      NaN          0.00  ...
False
```

```
9  40.100590 -82.925194      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
5   False  False           False          False        False
Day
6   False  False           False          False        False
Day
7   False  False           False          False        False
Day
8   False  False           False          False        False
Day
9   False  False           False          False        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            Day               Day                   Day
6            Day               Day                   Day
7            Day               Day                   Day
8            Day               Day                   Day
9            Day               Day                   Day

[10 rows x 46 columns]
```

```
# Checking the columns in the data
df.columns
```

```
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')
```

```
print("Number of columns: ",len(df.columns))
print("Number of rows: ",len(df))
```

```
Number of columns:  46
Number of rows:  7728394
```

```
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
```

```
df.describe()
```

```
           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
```

```
len(df.select_dtypes(['int64', 'float64']).columns)
```

```
13
```

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

```
ID                            0
Source                        0
Severity                      0
Start_Time                    0
End_Time                      0
Start_Lat                     0
Start_Lng                     0
End_Lat                 3402762
End_Lng                 3402762
Distance(mi)                  0
Description                   5
Street                    10869
City                        253
County                        0
State                         0
Zipcode                    1915
Country                       0
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
Amenity                         0
Bump                            0
Crossing                        0
Give_Way                        0
Junction                        0
No_Exit                         0
Railway                         0
Roundabout                      0
Station                         0
Stop                            0
Traffic_Calming                 0
Traffic_Signal                  0
Turning_Loop                    0
Sunrise_Sunset             23246
Civil_Twilight             23246
Nautical_Twilight          23246
Astronomical_Twilight      23246
dtype: int64
```

```
df.isna().sum()
```

```
ID                              0
Source                          0
Severity                        0
Start_Time                      0
End_Time                        0
Start_Lat                       0
Start_Lng                       0
End_Lat                   3402762
End_Lng                   3402762
Distance(mi)                    0
Description                     5
Street                      10869
City                          253
County                          0
State                           0
Zipcode                      1915
Country                         0
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
Amenity                         0
Bump                            0
Crossing                        0
Give_Way                        0
Junction                        0
No_Exit                         0
Railway                         0
Roundabout                      0
Station                         0
Stop                            0
Traffic_Calming                 0
Traffic_Signal                  0
Turning_Loop                    0
Sunrise_Sunset             23246
Civil_Twilight             23246
Nautical_Twilight          23246
Astronomical_Twilight      23246
dtype: int64
```

```
df.isna().sum().sort_values(ascending=False) * 100. / len(df)
```

```
End_Lat                   44.029355
End_Lng                   44.029355
Precipitation(in)         28.512858
Wind_Chill(F)             25.865904
Wind_Speed(mph)            7.391355
Visibility(mi)             2.291524
Wind_Direction             2.267043
Humidity(%)                2.253301
Weather_Condition          2.244438
Temperature(F)             2.120143
Pressure(in)               1.820288
Weather_Timestamp          1.555666
Nautical_Twilight          0.300787
Civil_Twilight             0.300787
Sunrise_Sunset             0.300787
Astronomical_Twilight      0.300787
Airport_Code               0.292881
Street                     0.140637
```

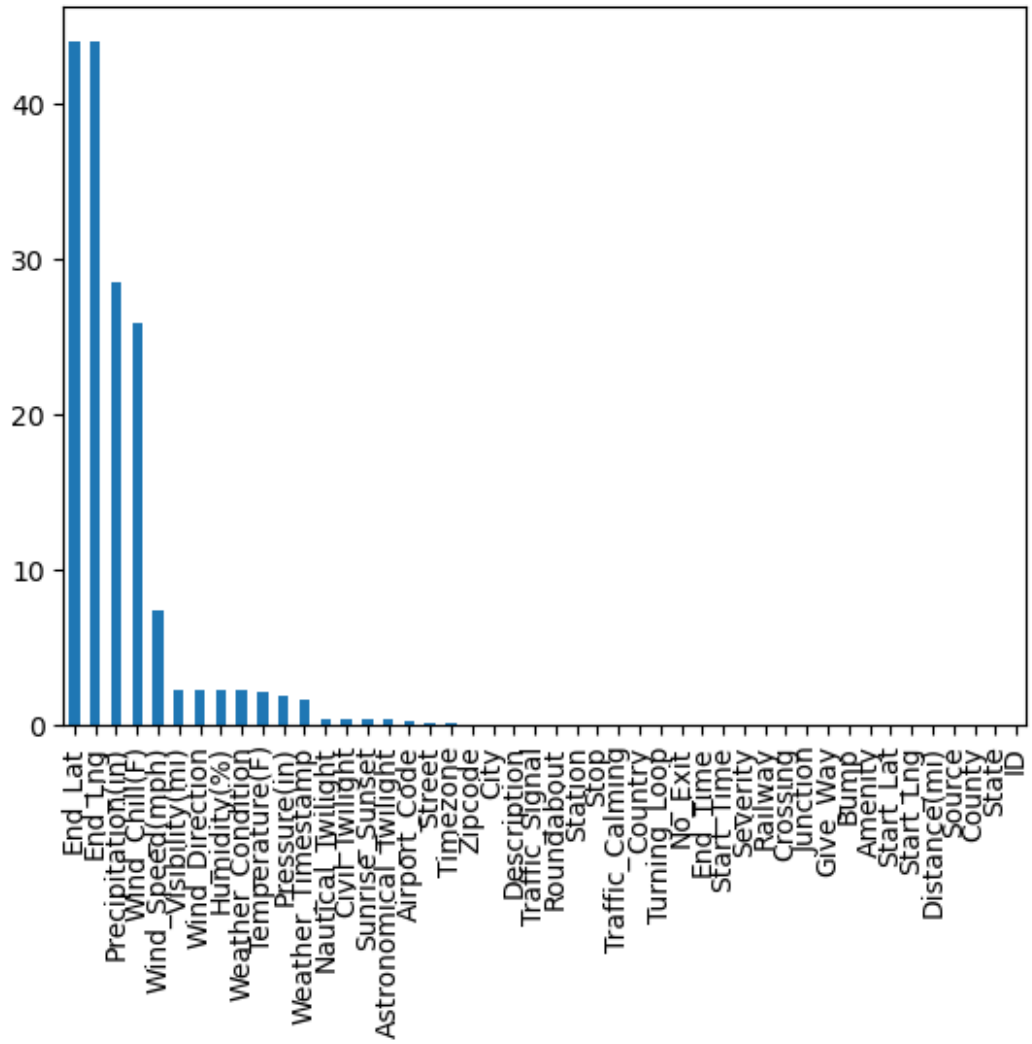
```
Timezone                 0.101030
Zipcode                  0.024779
City                     0.003274
Description              0.000065
Traffic_Signal           0.000000
Roundabout               0.000000
Station                  0.000000
Stop                     0.000000
Traffic_Calming          0.000000
Country                  0.000000
Turning_Loop             0.000000
No_Exit                  0.000000
End_Time                 0.000000
Start_Time               0.000000
Severity                 0.000000
Railway                  0.000000
Crossing                 0.000000
Junction                 0.000000
Give_Way                 0.000000
Bump                     0.000000
Amenity                  0.000000
Start_Lat                0.000000
Start_Lng                0.000000
Distance(mi)             0.000000
Source                   0.000000
County                   0.000000
State                    0.000000
ID                       0.000000
dtype: float64
```

```
# Plotting a Pandas.Series data
missing_data = df.isna().sum().sort_values(ascending=False) * 100. /
len(df)

type(missing_data) # we can directly plot the Pandas.Series using
plot()
```

```
pandas.core.series.Series
```

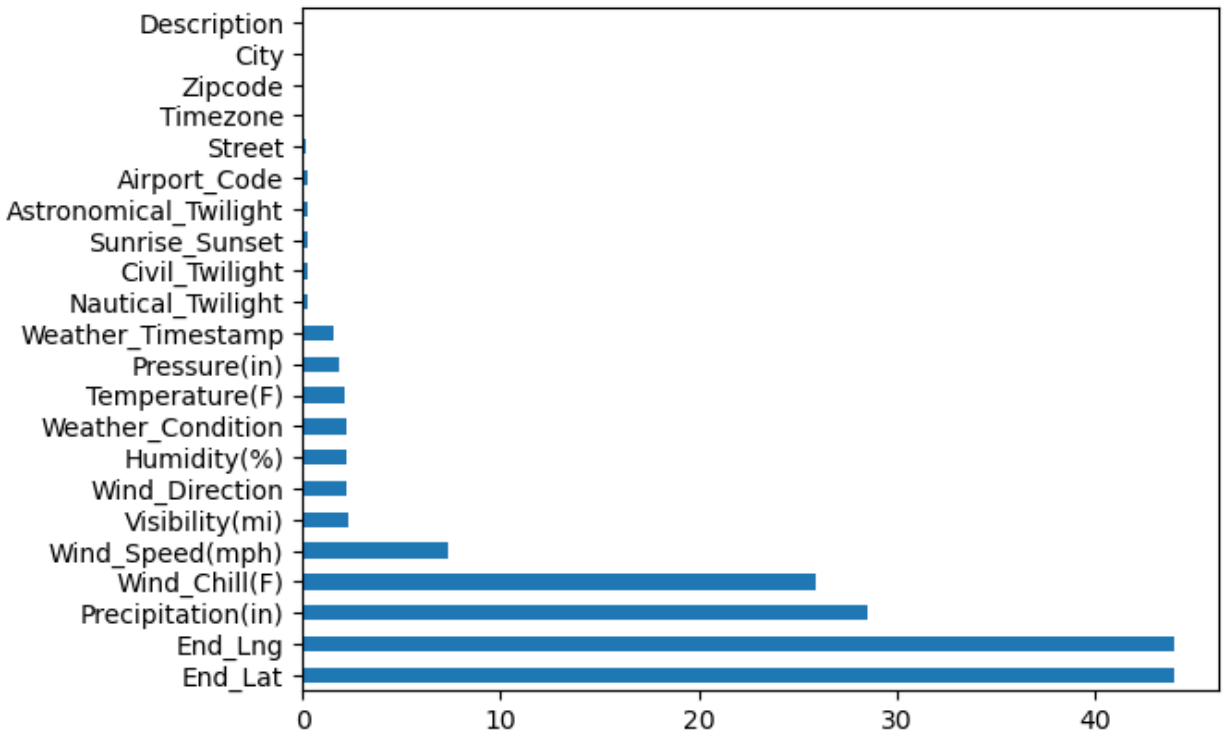
```
missing_data.plot(kind='bar')
```

```
<Axes: >
```




```
missing_data[missing_data!=0].plot(kind='barh')
```

```
<Axes: >
```




```
# Printing all the columns
df.columns
```

```
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')
```

```
df.City.unique()
```

```
array(['Dayton', 'Reynoldsburg', 'Williamsburg', ..., 'Ness City',
       'Clarksdale', 'American Fork-Pleasant Grove'], dtype=object)
```

```
cities = df.City.unique()
len(cities)
```

```
13679
```

```
cities_by_accident = df.City.value_counts()
cities_by_accident[:20]
```

```
City
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: count, dtype: int64
```

```
'New York' in cities
```

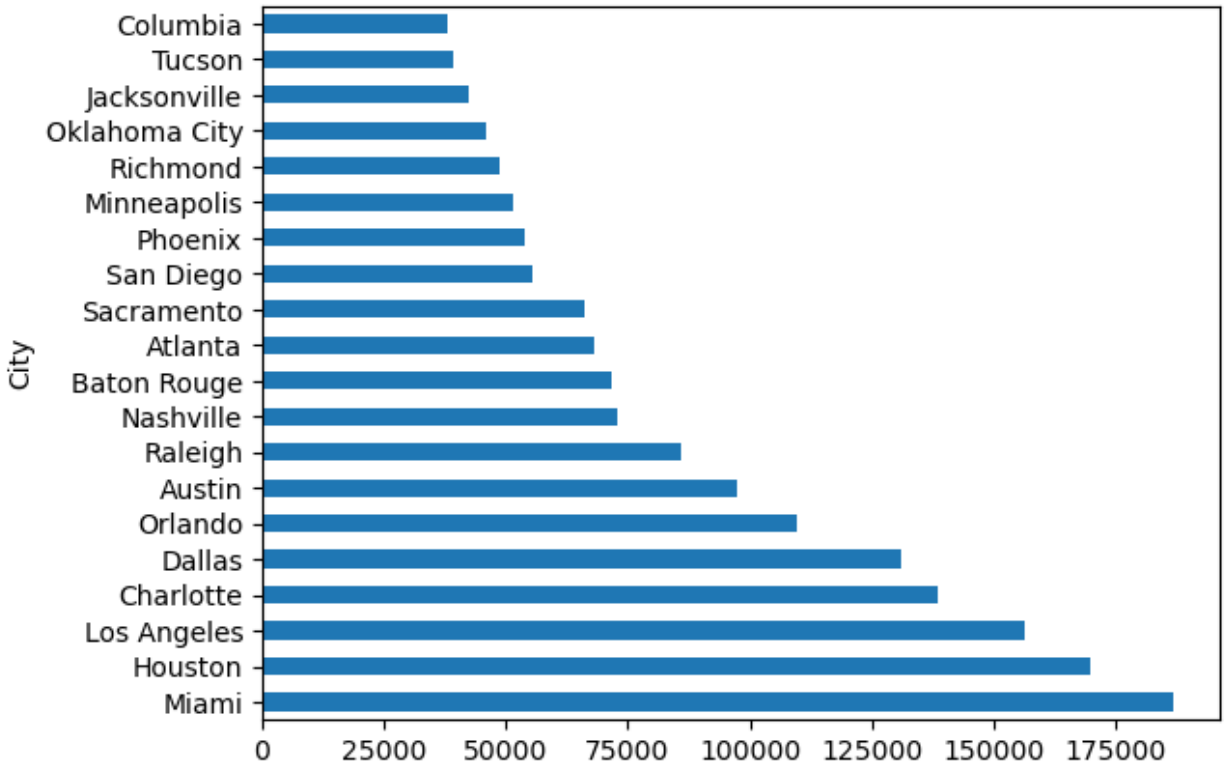
```
True
```

```
cities_by_accident["New York"]
```

```
21699
```

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

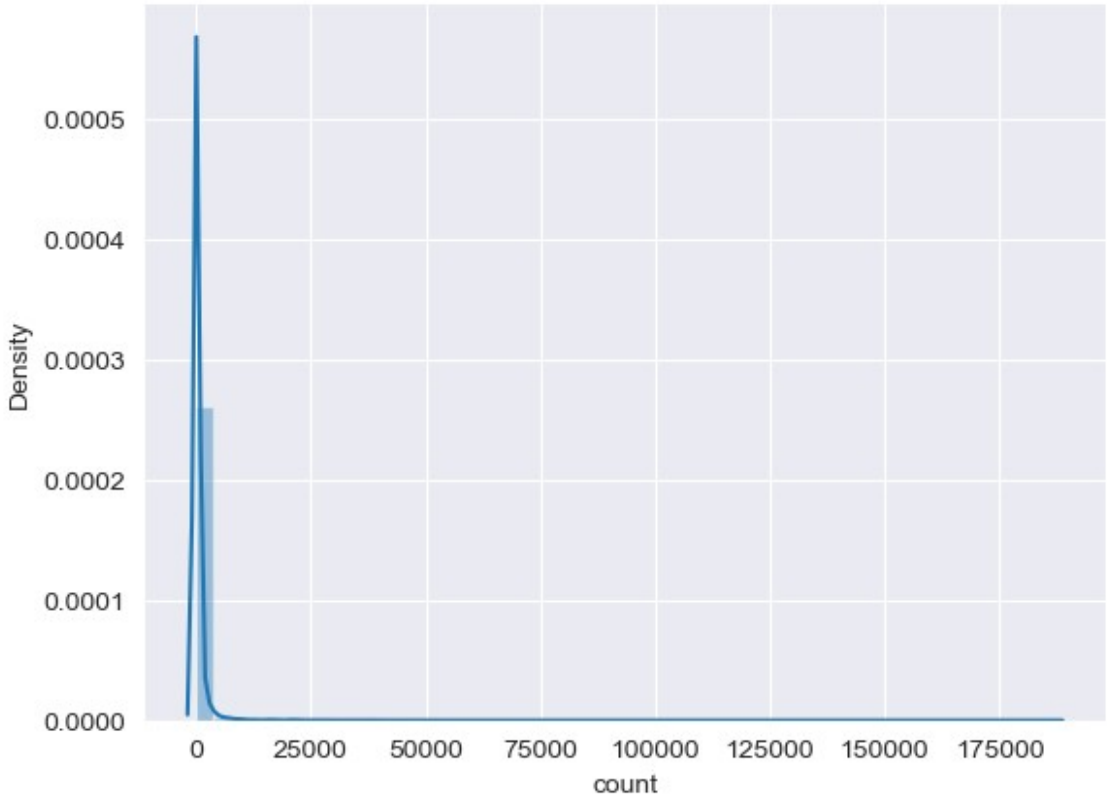
```
<Axes: ylabel='City'>
```




```
import seaborn as sns
sns.set_style("darkgrid")

sns.distplot(cities_by_accident)
```

```
C:\Users\jayaraman\AppData\Local\Temp\ipykernel_18940\3405282844.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(cities_by_accident)
C:\Users\jayaraman\anaconda3\Lib\site-packages\seaborn\
_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated
and will be removed in a future version. Convert inf values to NaN
before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

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

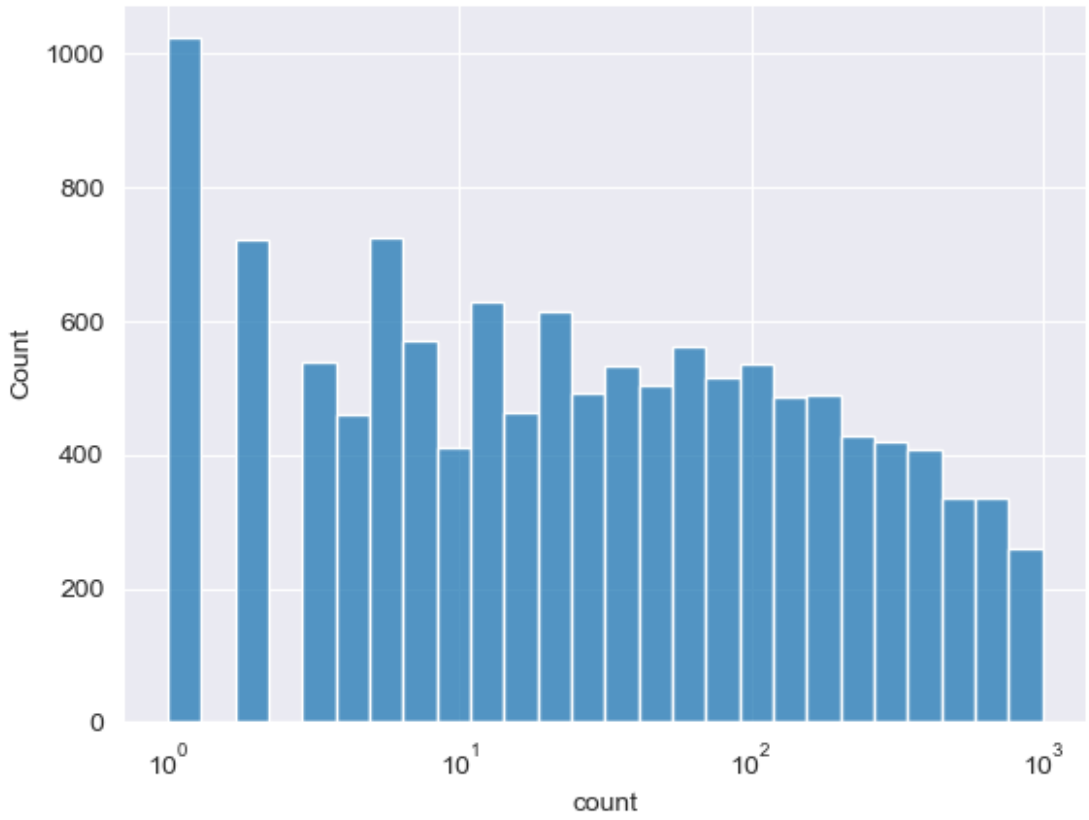



```
high_accident_cities = cities_by_accident[cities_by_accident >=1000] # having over 1000 accidents
low_accident_cities = cities_by_accident[cities_by_accident < 1000]  # having less than 1000 accidents

# Percentage of high accident cities
len(high_accident_cities) / len(cities_by_accident)
```

```
0.08904810644831115
```

```
# Distribution of low accident cities
sns.histplot(low_accident_cities, log_scale=True)
```

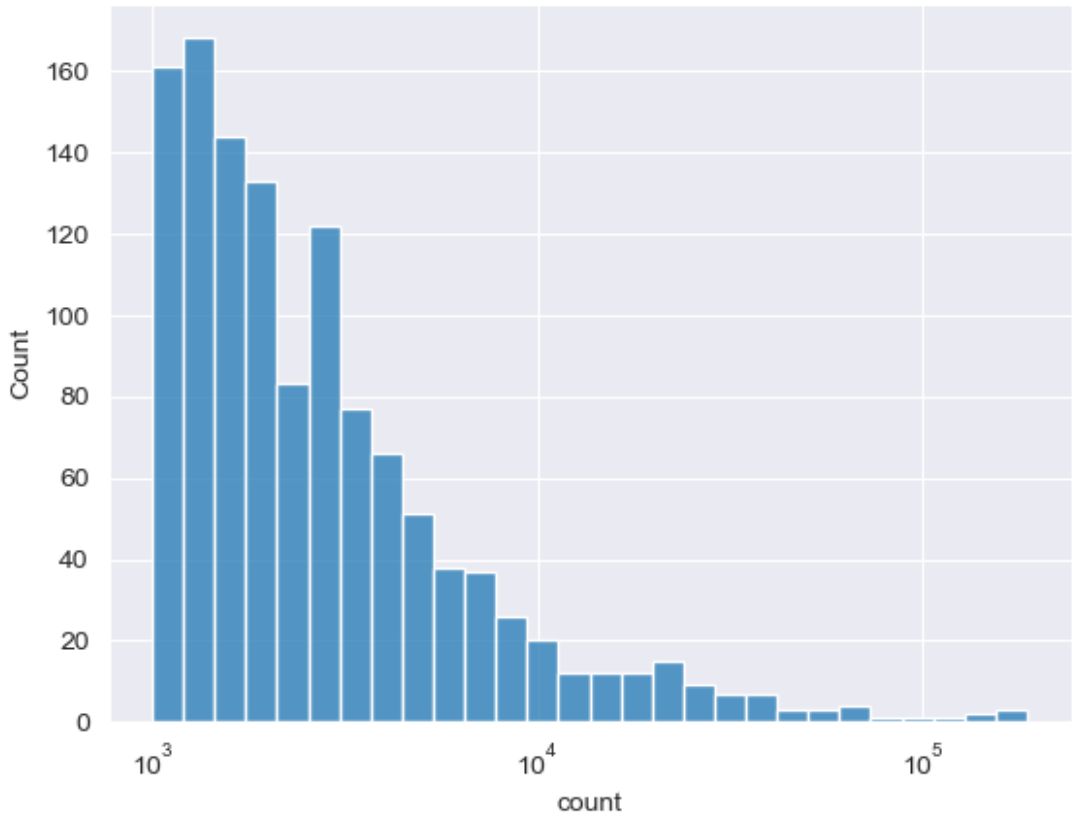
```
C:\Users\jayaraman\anaconda3\Lib\site-packages\seaborn\
_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated
and will be removed in a future version. Convert inf values to NaN
before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):

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




```
# Distribution of high accident cities
sns.histplot(high_accident_cities, log_scale=True)
```

```
C:\Users\jayaraman\anaconda3\Lib\site-packages\seaborn\
_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated
and will be removed in a future version. Convert inf values to NaN
before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

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




```
cities_by_accident[cities_by_accident == 1]
```

```
City
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: count, Length: 1023, dtype: int64
```

```
#checking out an entry
df.Start_Time[0]
```

```
'2016-02-08 05:46:00'
```

```
# converting date time to correct format
df.Start_Time = pd.to_datetime(df.Start_Time,format='ISO8601')

df.Start_Time[0]
```

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

```
# Segregating the different aspects of date-time
df.Start_Time[0].day, df.Start_Time[0].month, df.Start_Time[0].year,
df.Start_Time[0].hour, df.Start_Time[0].minute,
df.Start_Time[0].second
```

```
(8, 2, 2016, 5, 46, 0)
```

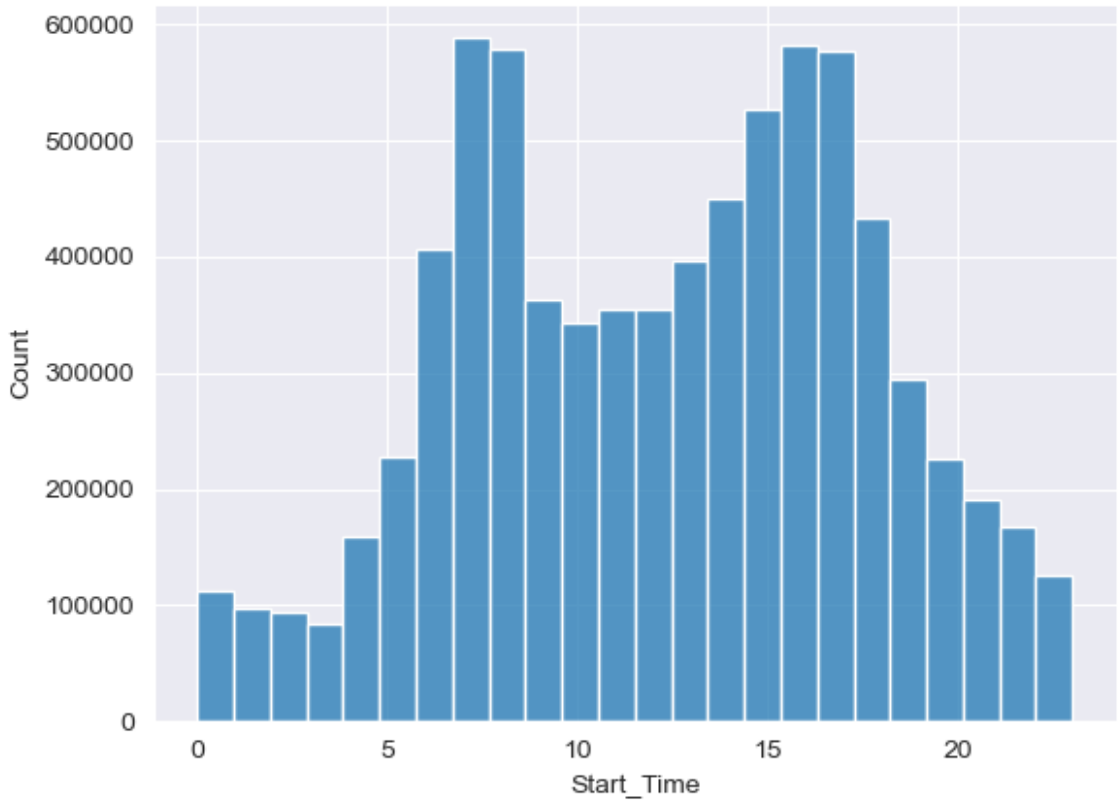
```
df.Start_Time.dt.hour
```

```
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: int32
```

```
sns.histplot(df.Start_Time.dt.hour, bins=24)
```

```
C:\Users\jayaraman\anaconda3\Lib\site-packages\seaborn\
_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated
and will be removed in a future version. Convert inf values to NaN
before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

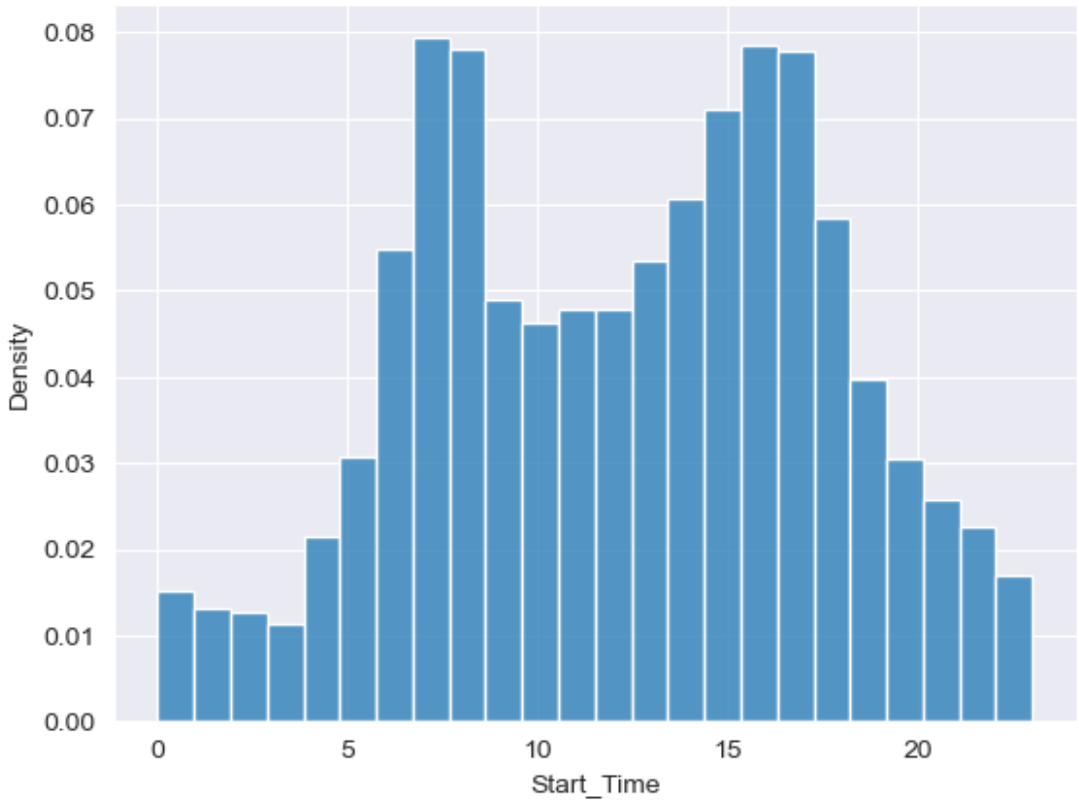
```
<Axes: xlabel='Start_Time', ylabel='Count'>
```




```
sns.histplot(df.Start_Time.dt.hour, bins=24, stat='density')
```

```
C:\Users\jayaraman\anaconda3\Lib\site-packages\seaborn\
_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated
and will be removed in a future version. Convert inf values to NaN
before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

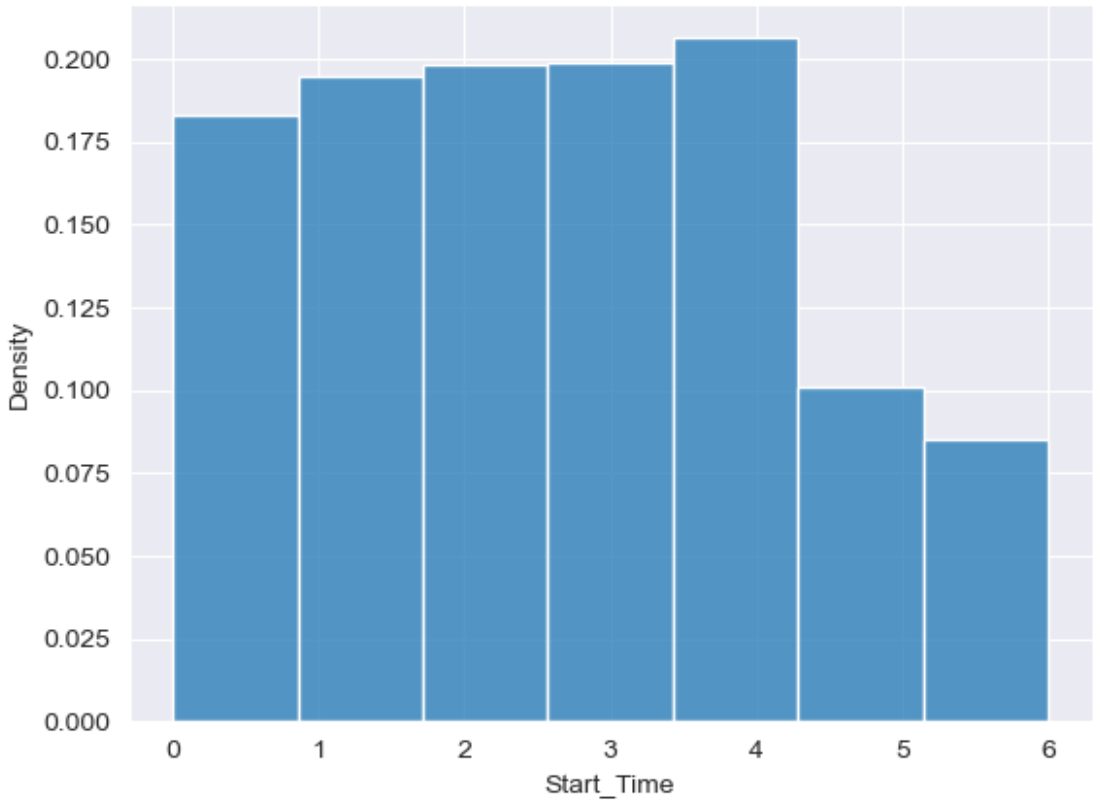
```
<Axes: xlabel='Start_Time', ylabel='Density'>
```




```
sns.histplot(df.Start_Time.dt.dayofweek, bins=7, stat='density')
```

```
C:\Users\jayaraman\anaconda3\Lib\site-packages\seaborn\
_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated
and will be removed in a future version. Convert inf values to NaN
before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

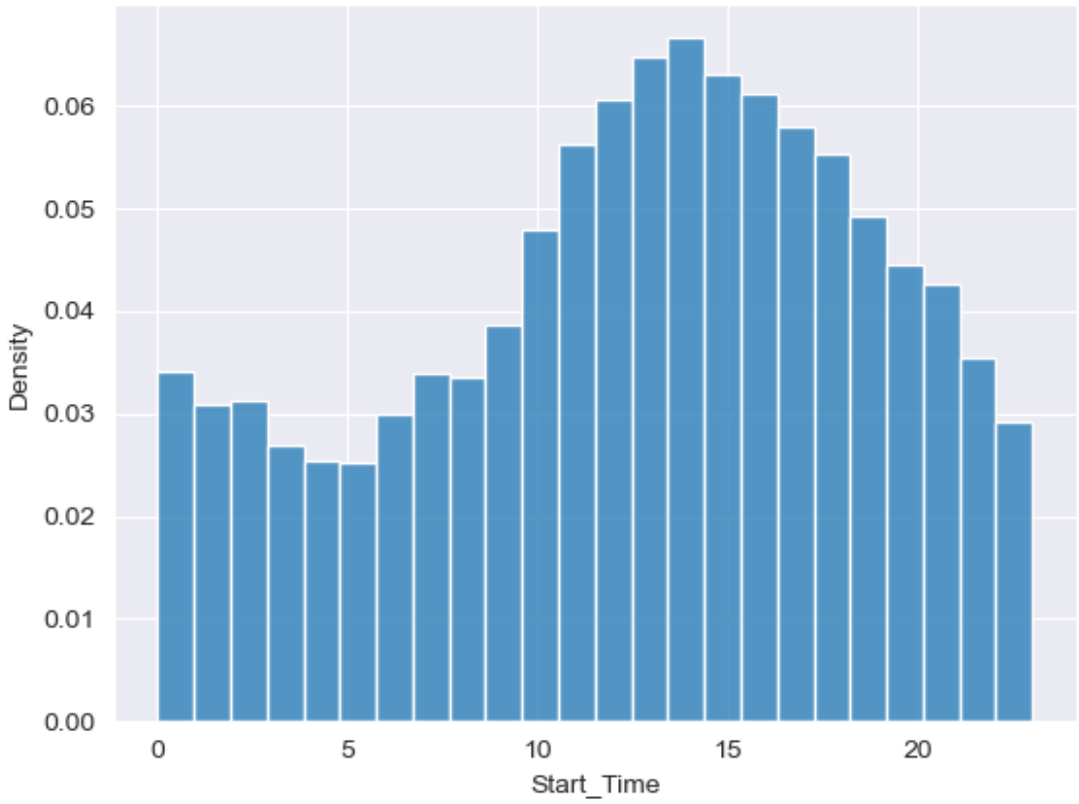
```
<Axes: xlabel='Start_Time', ylabel='Density'>
```




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

sns.histplot(sundays_start_time.dt.hour, bins=24, stat='density')
```

```
C:\Users\jayaraman\anaconda3\Lib\site-packages\seaborn\
_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated
and will be removed in a future version. Convert inf values to NaN
before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

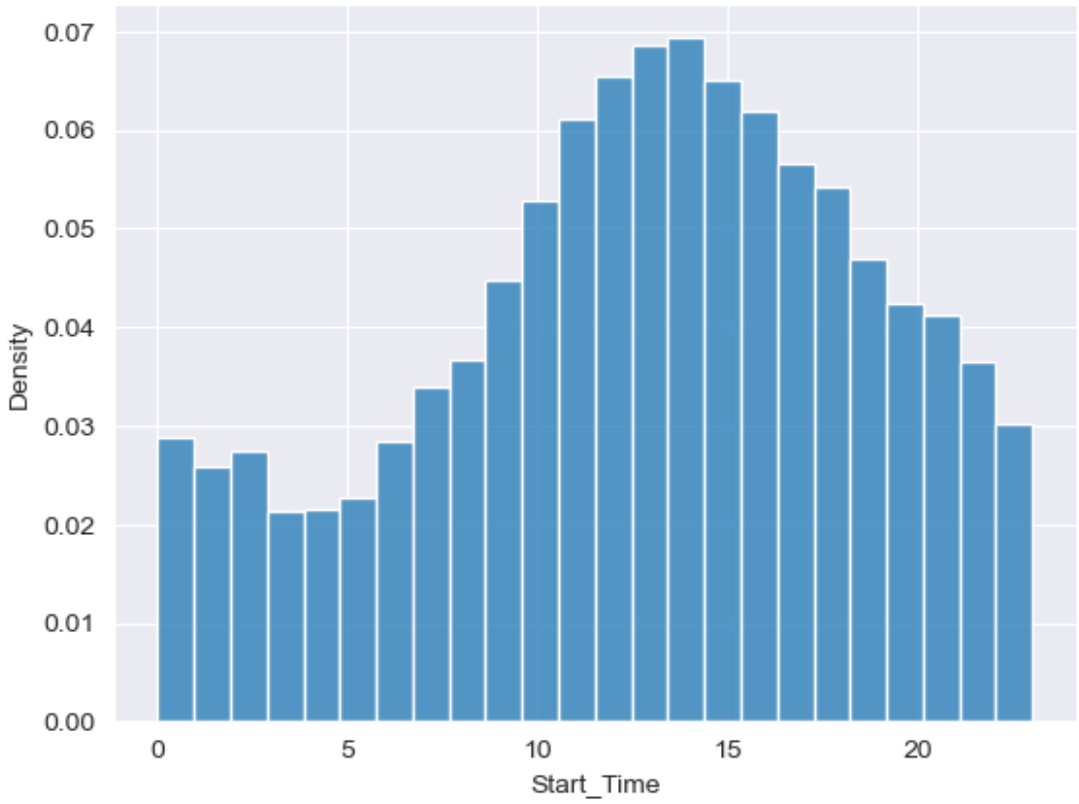
```
<Axes: xlabel='Start_Time', ylabel='Density'>
```




```
saturdays_start_time = df.Start_Time[df.Start_Time.dt.dayofweek == 5]
sns.histplot(saturdays_start_time.dt.hour, bins=24, stat='density')
```

```
C:\Users\jayaraman\anaconda3\Lib\site-packages\seaborn\
_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated
and will be removed in a future version. Convert inf values to NaN
before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

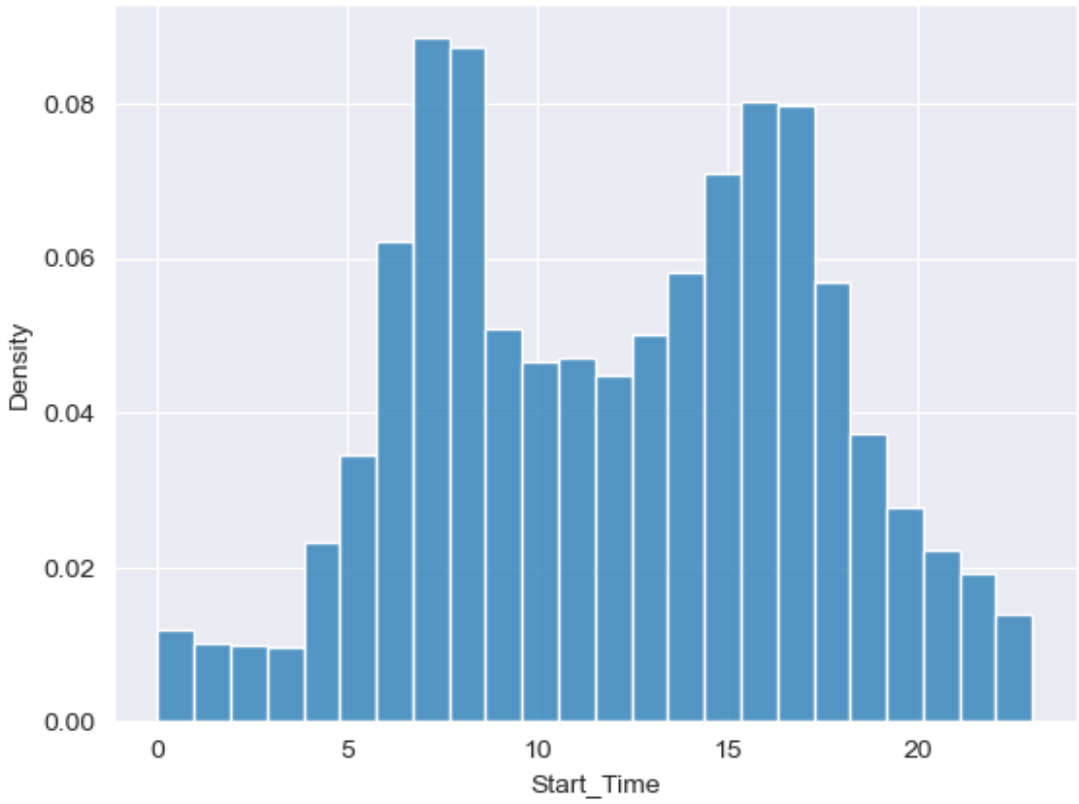
```
<Axes: xlabel='Start_Time', ylabel='Density'>
```




```
mondays_start_time = df.Start_Time[df.Start_Time.dt.dayofweek == 0]
sns.histplot(mondays_start_time.dt.hour, bins=24, stat='density')
```

```
C:\Users\jayaraman\anaconda3\Lib\site-packages\seaborn\
_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated
and will be removed in a future version. Convert inf values to NaN
before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

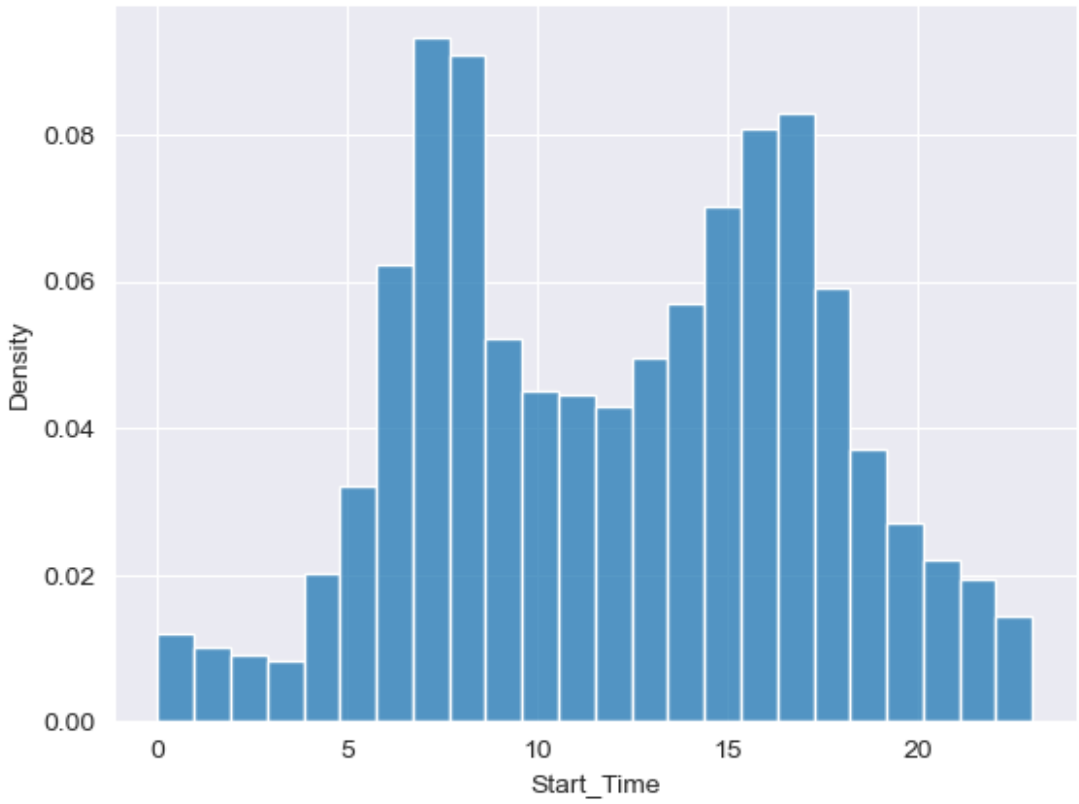
```
<Axes: xlabel='Start_Time', ylabel='Density'>
```




```
wednesdays_start_time = df.Start_Time[df.Start_Time.dt.dayofweek == 2]
sns.histplot(wednesdays_start_time.dt.hour, bins=24, stat='density')
```

```
C:\Users\jayaraman\anaconda3\Lib\site-packages\seaborn\
_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated
and will be removed in a future version. Convert inf values to NaN
before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

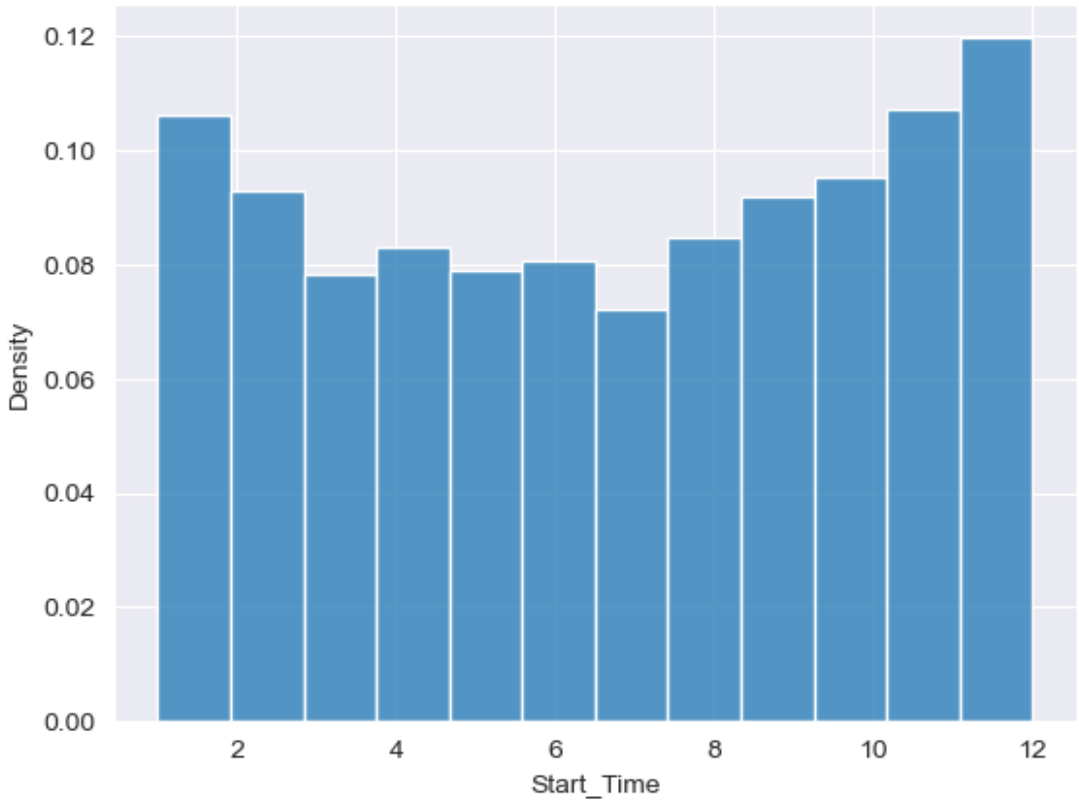
```
<Axes: xlabel='Start_Time', ylabel='Density'>
```




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

```
C:\Users\jayaraman\anaconda3\Lib\site-packages\seaborn\
_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated
and will be removed in a future version. Convert inf values to NaN
before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

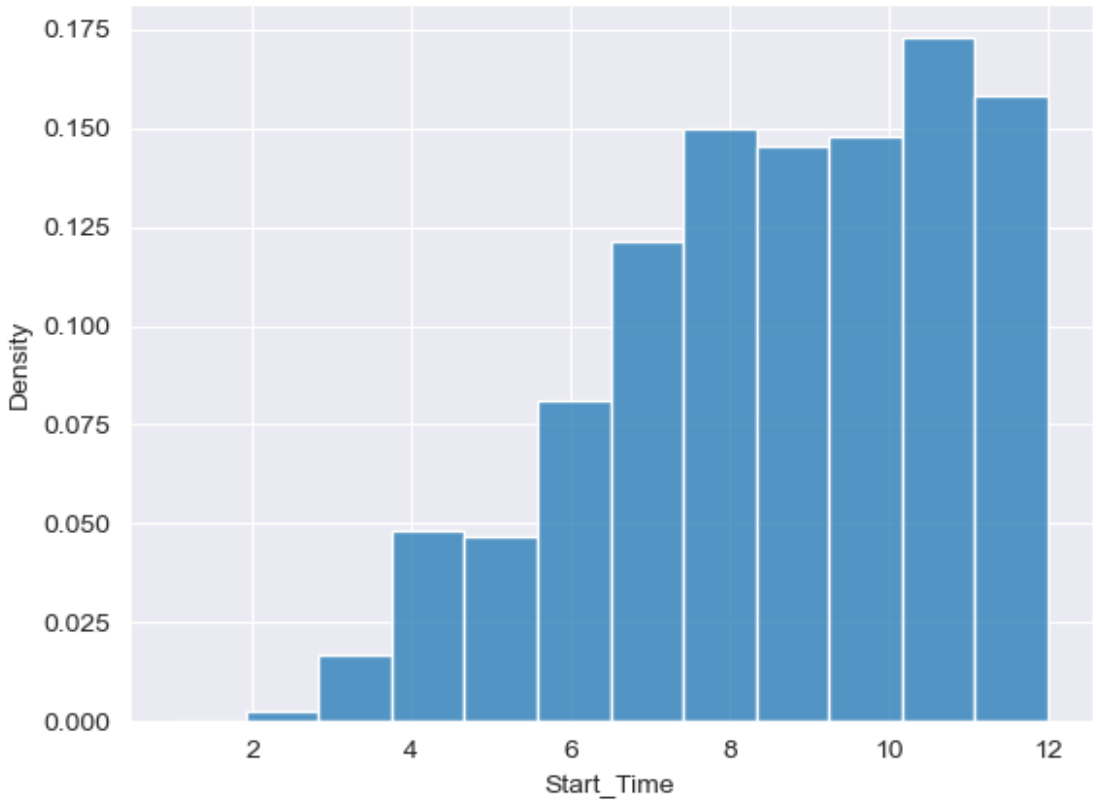
```
<Axes: xlabel='Start_Time', ylabel='Density'>
```




```
df_particular_year = df[df.Start_Time.dt.year == 2016]
sns.histplot(df_particular_year.Start_Time.dt.month, bins=12,
stat='density')
```

```
C:\Users\jayaraman\anaconda3\Lib\site-packages\seaborn\
_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated
and will be removed in a future version. Convert inf values to NaN
before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

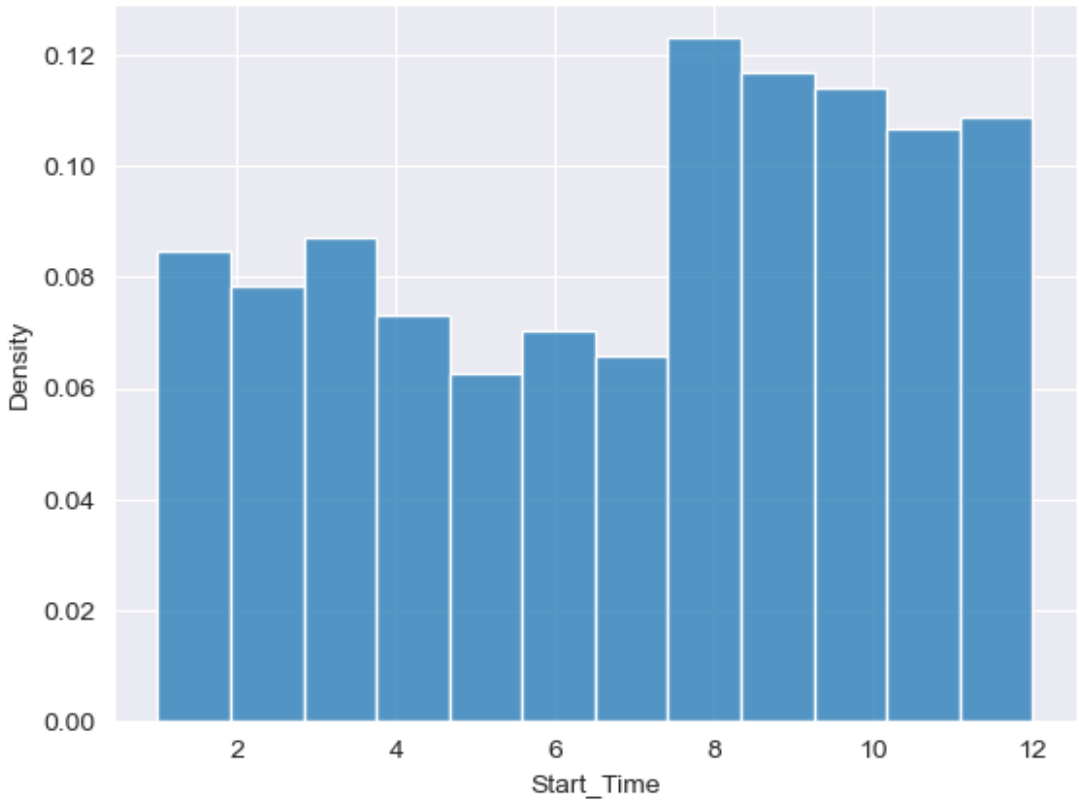
```
<Axes: xlabel='Start_Time', ylabel='Density'>
```




```
df_particular_year = df[df.Start_Time.dt.year == 2017]
sns.histplot(df_particular_year.Start_Time.dt.month, bins=12,
stat='density')
```

```
C:\Users\jayaraman\anaconda3\Lib\site-packages\seaborn\
_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated
and will be removed in a future version. Convert inf values to NaN
before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

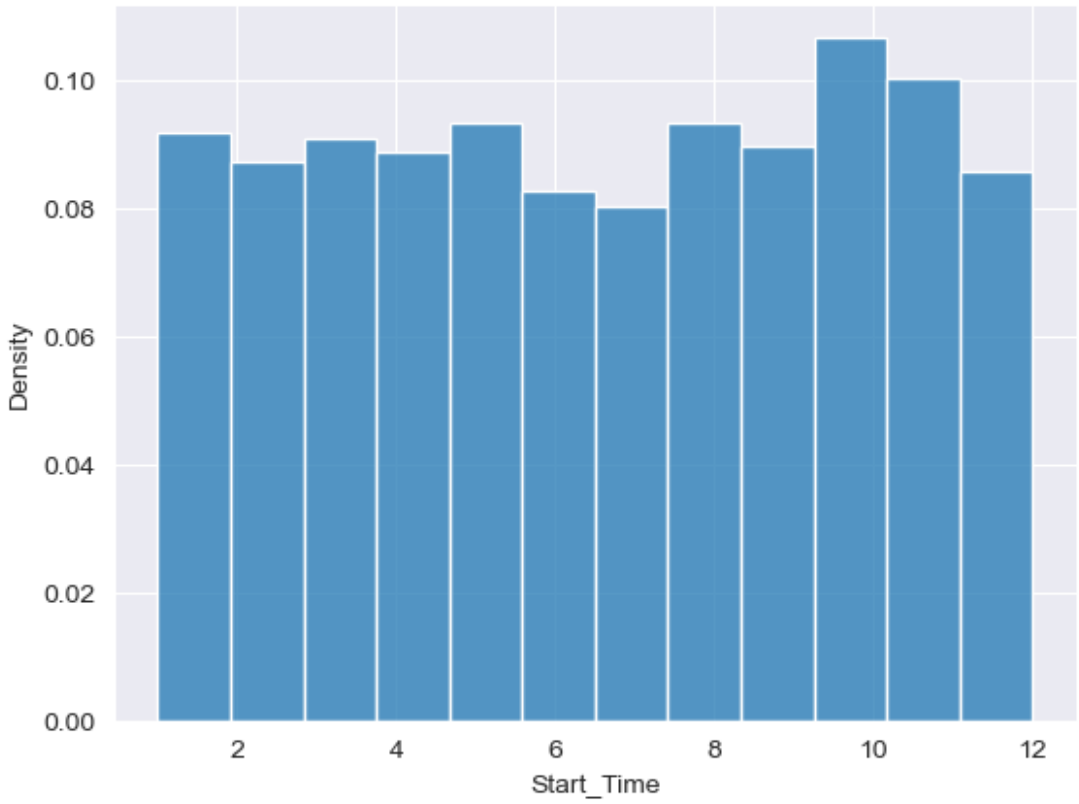
```
<Axes: xlabel='Start_Time', ylabel='Density'>
```




```
df_particular_year = df[df.Start_Time.dt.year == 2018]
sns.histplot(df_particular_year.Start_Time.dt.month, bins=12,
stat='density')
```

```
C:\Users\jayaraman\anaconda3\Lib\site-packages\seaborn\
_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated
and will be removed in a future version. Convert inf values to NaN
before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

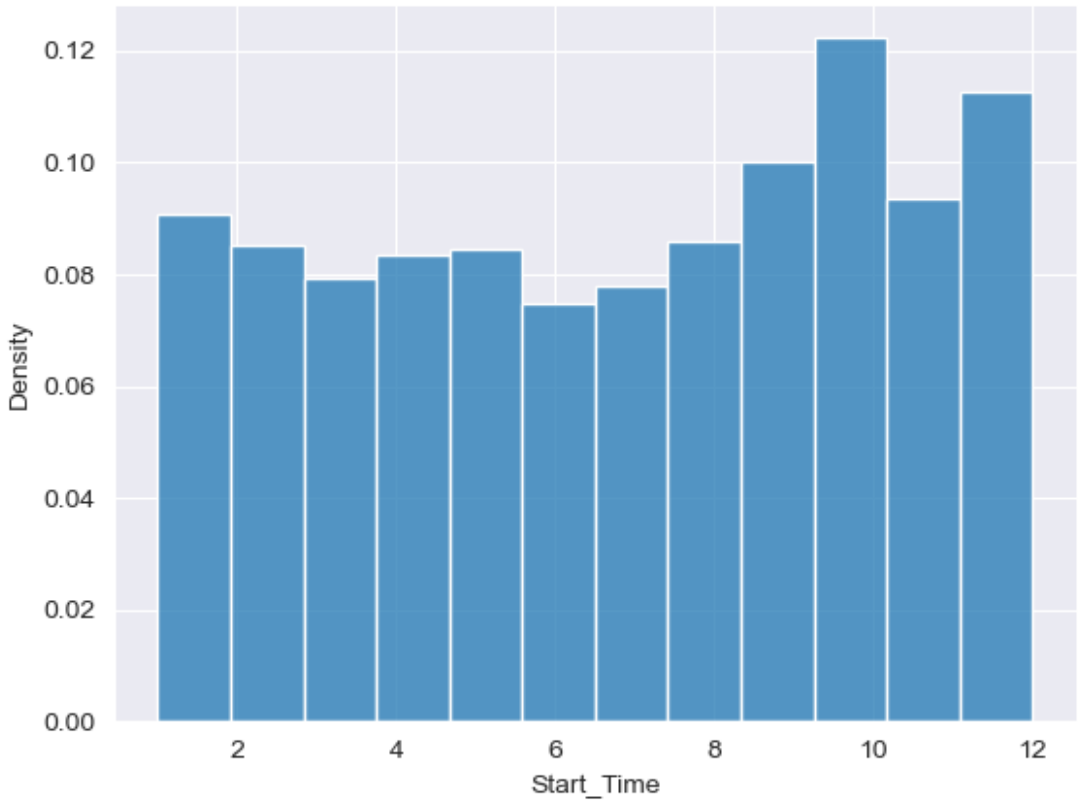
```
<Axes: xlabel='Start_Time', ylabel='Density'>
```




```
df_particular_year = df[df.Start_Time.dt.year == 2019]
sns.histplot(df_particular_year.Start_Time.dt.month, bins=12,
stat='density')
```

```
C:\Users\jayaraman\anaconda3\Lib\site-packages\seaborn\
_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated
and will be removed in a future version. Convert inf values to NaN
before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

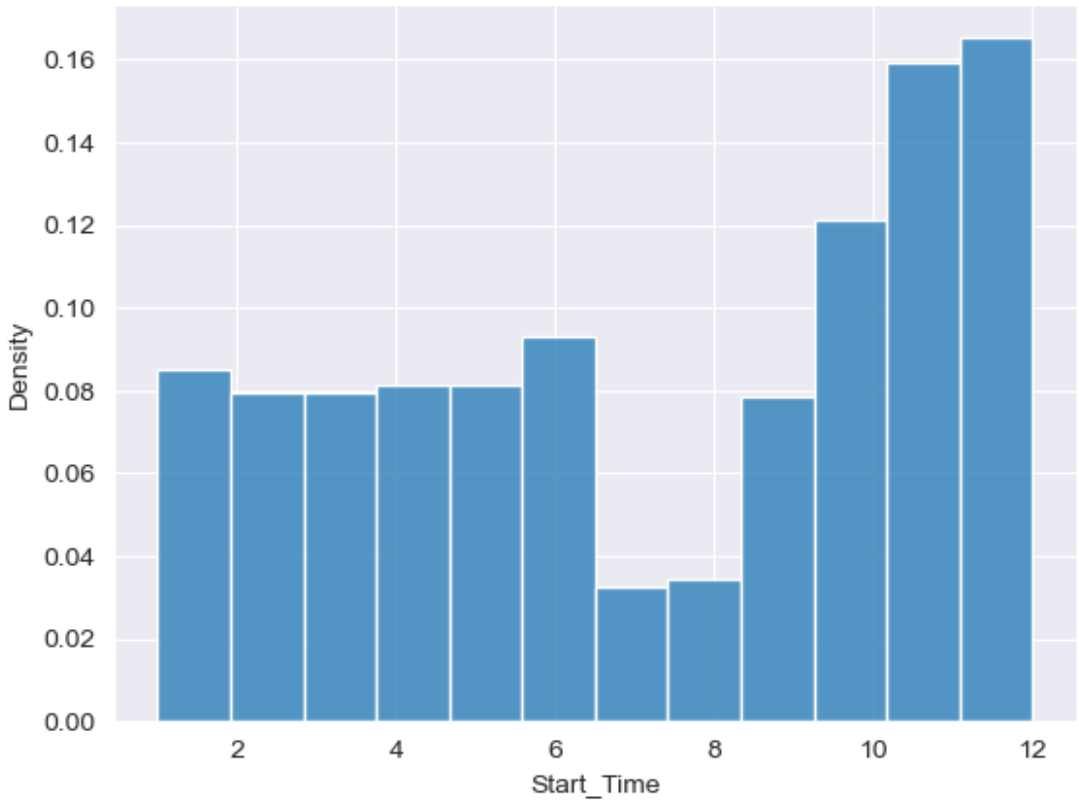
```
<Axes: xlabel='Start_Time', ylabel='Density'>
```




```
df_particular_year = df[df.Start_Time.dt.year == 2020]
sns.histplot(df_particular_year.Start_Time.dt.month, bins=12,
stat='density')
```

```
C:\Users\jayaraman\anaconda3\Lib\site-packages\seaborn\
_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated
and will be removed in a future version. Convert inf values to NaN
before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

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




```
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
```

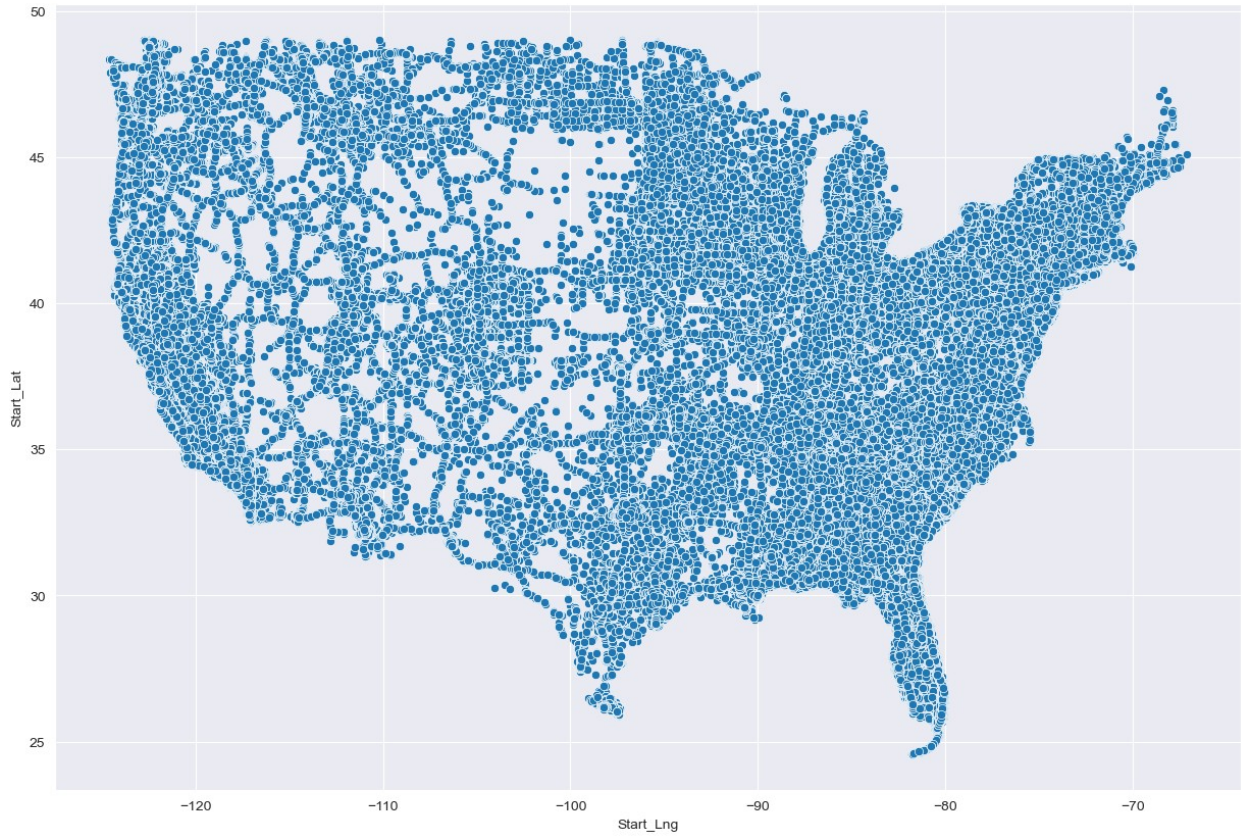
```
df.Start_Lng
```

```
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
```

```
import matplotlib.pyplot as plt

plt.figure(figsize=(15,10))
sns.scatterplot(y=df.Start_Lat, x=df.Start_Lng)
```

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



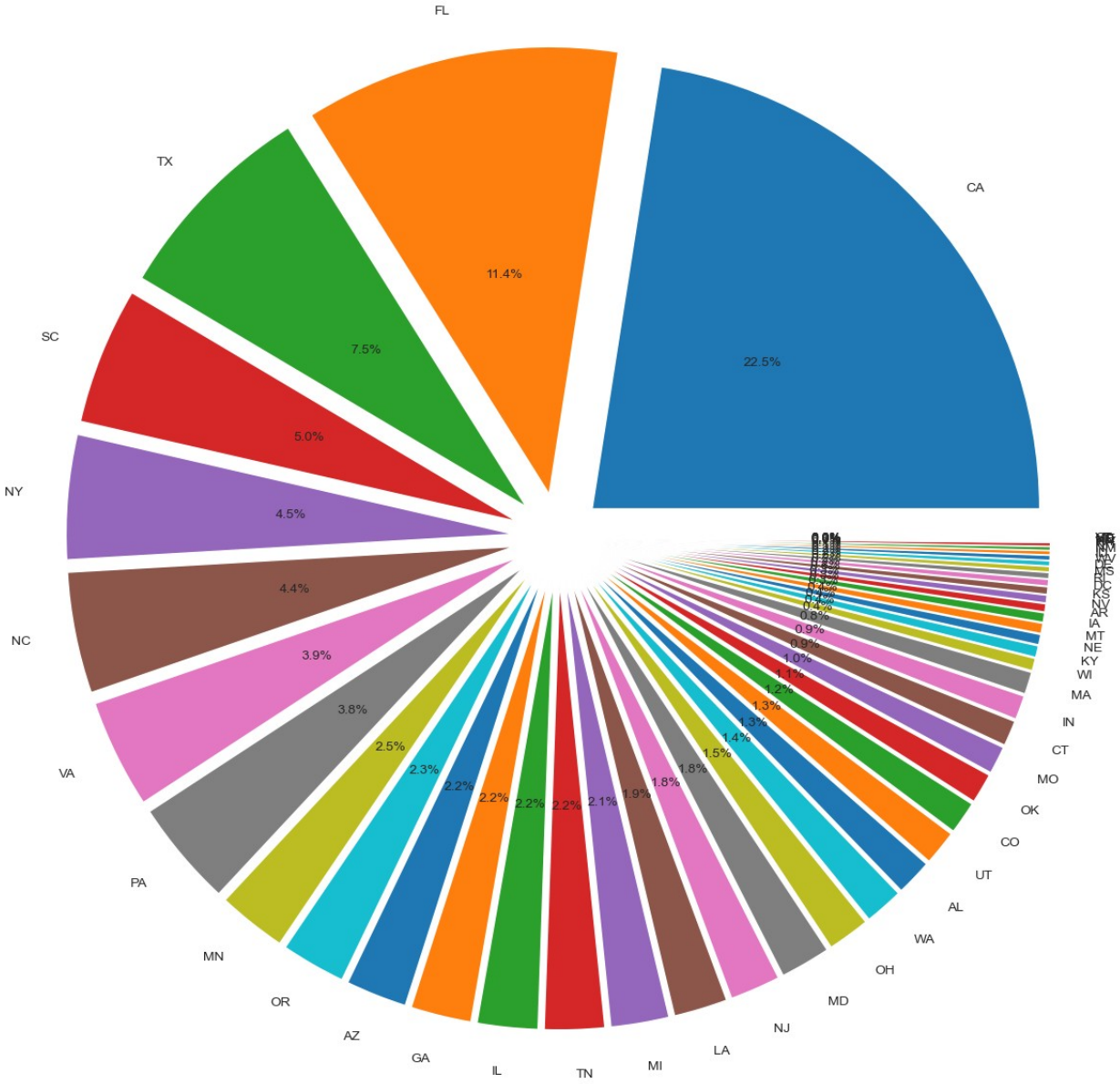

```
df.info()
```

```
<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             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

df.State.value_counts()[:25]

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
OH     118115
WA     108221
AL     101044
UT      97079
CO      90885
OK      83647
MO      77323
Name: count, dtype: int64
```

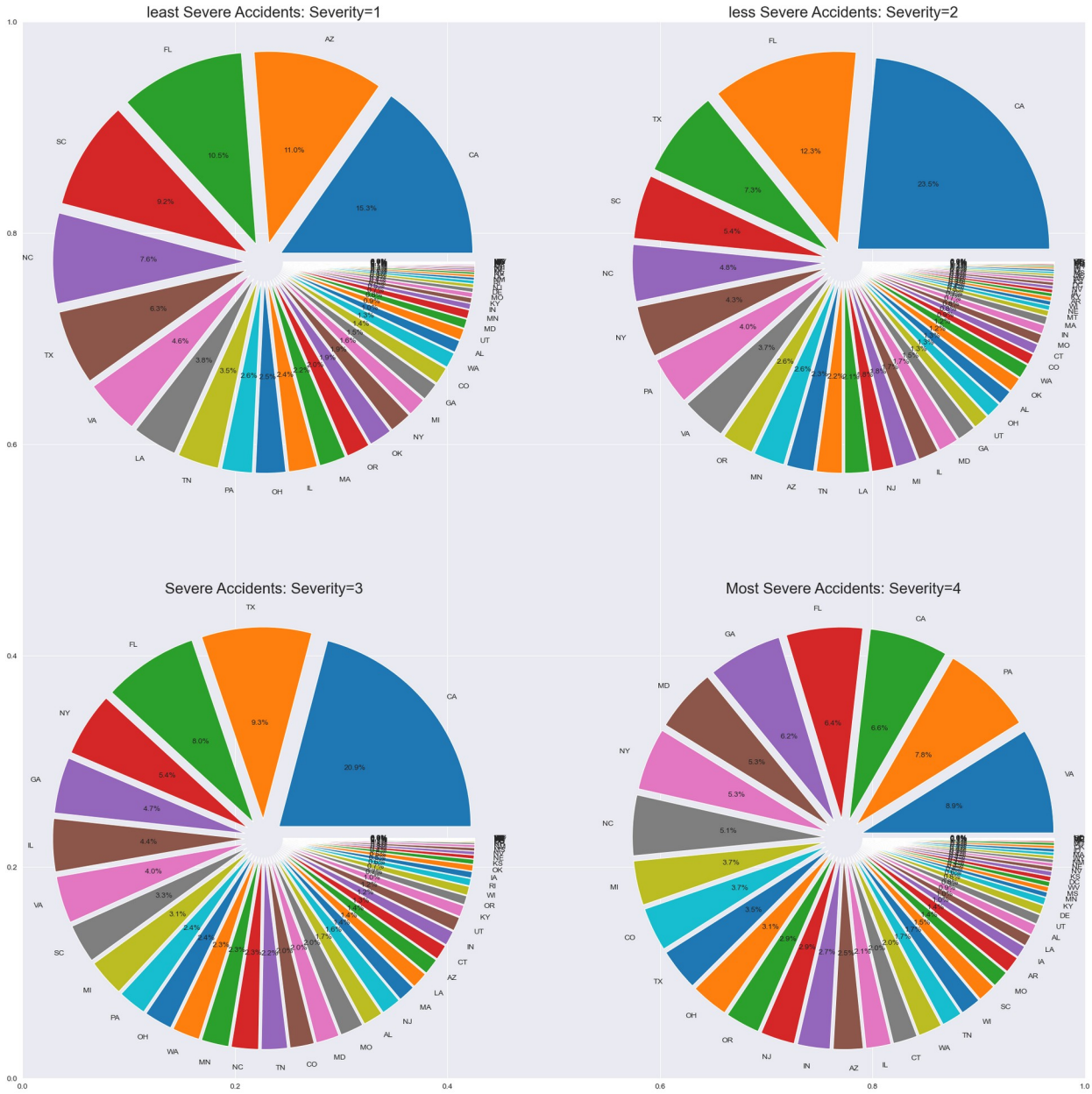
```
pie, ax = plt.subplots(figsize=[15,15])
labels = df.State.value_counts().keys()
plt.pie(x=df.State.value_counts(), autopct="%.1f%%",
explode=[0.1]*len(df.State.value_counts()), labels=labels,
pctdistance=0.5)
plt.show();
```




```
# Segregating accidents on the basis of severity
severe_accidents_4 = df[df.Severity==4].State.value_counts()
severe_accidents_3 = df[df.Severity==3].State.value_counts()
severe_accidents_2 = df[df.Severity==2].State.value_counts()
severe_accidents_1 = df[df.Severity==1].State.value_counts()

fig, ax1 = plt.subplots(figsize=[25,25])
ax1 = plt.subplot2grid((2,2),(0,0))
labels = severe_accidents_1.keys()
plt.pie(x=severe_accidents_1, autopct="%.1f%%",
explode=[0.1]*len(severe_accidents_1), labels=labels, pctdistance=0.5)
plt.title("least Severe Accidents: Severity=1", fontsize=20)
```

```
ax1 = plt.subplot2grid((2,2),(0,1))
labels = severe_accidents_2.keys()
plt.pie(x=severe_accidents_2, autopct="%.1f%%",
explode=[0.1]*len(severe_accidents_2), labels=labels, pctdistance=0.5)
plt.title("less Severe Accidents: Severity=2", fontsize=20)

ax1 = plt.subplot2grid((2,2),(1,0))
labels = severe_accidents_3.keys()
plt.pie(x=severe_accidents_3, autopct="%.1f%%",
explode=[0.1]*len(severe_accidents_3), labels=labels, pctdistance=0.5)
plt.title("Severe Accidents: Severity=3", fontsize=20)

ax1 = plt.subplot2grid((2,2),(1,1))
labels = severe_accidents_4.keys()
plt.pie(x=severe_accidents_4, autopct="%.1f%%",
explode=[0.1]*len(severe_accidents_4), labels=labels, pctdistance=0.5)
plt.title("Most Severe Accidents: Severity=4", fontsize=20)
```

```
Text(0.5, 1.0, 'Most Severe Accidents: Severity=4')
```




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

```
[(39.865147, -84.058723),
 (39.92805900000001, -82.831184),
 (39.063148, -84.032608),
 (39.747753, -84.20558199999998),
 (39.627781, -84.188354),
 (40.10059, -82.92519399999998),
 (39.758274, -84.23050699999997),
 (39.770382, -84.194901),
 (39.778061, -84.172005),
 (40.10059, -82.92519399999998),
```

```
(39.952812, -83.119293),
(39.932709, -82.83091),
(39.737633, -84.14993299999998),
(39.79076, -84.241547),
(39.972038, -82.913521),
(39.745888, -84.17041),
(39.748329, -84.224007),
(39.752174, -84.239952),
(39.740669, -84.184135),
(39.790703, -84.244461),
(40.052509, -82.88233199999998),
(39.773346, -84.224686),
(39.628288, -84.226151),
(40.023487, -82.994888),
(39.761379, -84.25921600000002),
(40.158024, -82.641762),
(39.733219, -84.159653),
(39.775303, -84.200523),
(39.789322, -84.23910500000002),
(39.75872, -84.183762),
(40.081459, -83.122398),
(39.83321, -84.112946),
(40.042725, -82.99730699999998),
(39.974415, -82.848854),
(39.994766, -83.02449),
(40.006477, -83.030991),
(39.742062, -84.186996),
(39.843521, -82.78157),
(39.782578, -84.178688),
(40.007626, -82.912155),
(39.447502, -84.19120799999997),
(39.787731, -84.173439),
(39.426277, -83.624611),
(39.978306, -82.852554),
(39.84182, -84.18960600000003),
(39.953625, -82.958954),
(40.053082, -83.049644),
(39.936874, -82.878723),
(39.780643, -84.16349),
(39.773193, -84.187454),
(39.77858, -84.20468100000002),
(39.792988, -84.20607),
(39.770382, -84.194901),
(39.976398, -83.119225),
(39.6395, -84.23136099999998),
(39.232578, -84.159286),
(39.746593, -84.23912),
(39.742783, -84.182472),
(39.75452, -84.159279),
```

```
(39.749916, -84.139359),
(40.100903, -82.98410799999998),
(39.77280800000001, -84.201851),
(39.492485, -84.131798),
(39.570904, -84.251724),
(39.834282, -82.74382),
(39.622787, -84.229858),
(39.429871, -84.079666),
(41.407391, -81.64676700000003),
(41.424404, -81.57867399999998),
(39.89217, -83.039169),
(41.022358, -83.650345),
(39.206806, -84.053253),
(39.973797, -83.022285),
(40.048008, -82.86421999999997),
(39.814526, -84.021744),
(40.151196, -84.22003199999997),
(39.954617, -82.844238),
(38.990551, -84.207497),
(40.008995, -83.002678),
(39.077332, -84.225014),
(39.955452, -82.795845),
(40.224586, -82.929085),
(39.976398, -83.119225),
(39.98560300000001, -82.939247),
(41.040714, -81.61314399999998),
(41.083679, -81.579002),
(40.766891, -82.42411),
(41.422199, -81.843018),
(41.063995, -81.57294499999998),
(39.747753, -84.20558199999998),
(38.99408, -84.142021),
(41.420818, -81.694008),
(40.030342, -83.030197),
(39.858089, -82.827904),
(41.112324, -81.60992399999998),
(39.994061, -82.729416),
(41.355396, -81.81926700000002),
(39.687389, -84.23796800000002),
(39.751038, -84.214325),
(39.749916, -84.139359),
(39.97509, -82.88460500000002),
(40.11195, -83.016663),
(41.424313, -81.57914),
(39.691864, -84.211166),
(39.769173, -84.20728299999998),
(39.75452, -84.159279),
(39.874947, -83.003937),
(39.769596, -84.176964),
```

```
(39.776417, -84.23026999999998),
(39.64185, -84.227455),
(39.76028400000001, -84.196434),
(39.802528, -84.220093),
(39.828781, -84.19839499999998),
(39.733337, -84.180359),
(39.759426, -84.138367),
(39.78361500000001, -84.227531),
(39.74004, -84.194923),
(39.735199, -84.204948),
(39.749035, -84.20474200000002),
(39.969223, -82.984131),
(39.995369, -82.985085),
(39.744873, -84.205025),
(39.749786, -84.199493),
(39.780331, -84.195076),
(40.109928, -82.97820300000002),
(40.11195, -83.016663),
(40.112156, -83.035072),
(39.871761, -82.933449),
(39.934517, -82.86065699999997),
(40.02404, -82.99479699999998),
(41.014015, -81.932777),
(40.575069, -82.528145),
(39.996578, -84.21680500000002),
(39.986324, -82.985458),
(40.013062, -82.90213),
(39.981709, -82.98436),
(39.953022, -82.99889399999998),
(39.973812, -82.983162),
(39.978306, -82.852554),
(39.917412, -83.014236),
(39.469582, -83.712799),
(39.63958, -84.234818),
(39.975986, -82.996307),
(39.951401, -83.040161),
(39.96888, -83.022263),
(39.804295, -84.178917),
(39.831444, -84.097267),
(39.890587, -83.03538499999998),
(39.770977, -84.183914),
(39.808121, -84.173477),
(41.410461, -81.725853),
(39.747753, -84.20558199999998),
(39.6395, -84.23136099999998),
(39.94537, -82.88744399999999),
(39.79813, -84.24479699999998),
(40.081715, -82.89576),
(41.111031, -81.550728),
```

```
(40.082016, -82.937119),
(39.763363, -84.15535),
(39.740063, -84.191833),
(39.99876, -82.625221),
(39.740063, -84.191833),
(39.99519, -82.86859100000002),
(41.395805, -81.935562),
(39.887058, -82.883263),
(39.824703, -83.141228),
(40.977516, -82.790749),
(40.11195, -83.016663),
(41.150993, -81.631393),
(39.824703, -83.141228),
(39.885872, -82.806198),
(40.11195, -83.016663),
(40.09917100000001, -83.134499),
(40.012829, -83.109131),
(40.048183, -82.945183),
(39.77792, -84.246872),
(39.917412, -83.014236),
(41.313236, -81.810959),
(39.691998, -84.043655),
(39.852417, -82.93860600000002),
(39.956909, -83.044914),
(39.602947, -82.951836),
(39.89188400000001, -83.087547),
(39.78508400000001, -84.210564),
(40.1105, -82.977524),
(40.757088, -82.50975),
(41.063995, -81.57294499999998),
(40.004749, -82.867638),
(40.033955, -83.000862),
(38.929512, -83.495262),
(39.973419, -83.081558),
(39.797958, -84.255135),
(40.057369, -82.903717),
(39.847927, -82.848274),
(40.051949, -83.03280600000002),
(40.051949, -83.03280600000002),
(39.772823, -84.135612),
(39.673203, -84.221886),
(39.76244000000001, -84.20506999999998),
(39.79166, -84.169342),
(39.72757, -84.13549),
(40.09917100000001, -83.134499),
(38.455898, -81.93562299999998),
(39.764778, -84.177193),
(40.016014, -82.903442),
(39.686901, -83.911812),
```

```
(39.74403, -84.214638),
(39.955738, -83.002953),
(38.98587, -84.052551),
(41.12336, -81.652626),
(41.403664, -82.13942),
(39.719009, -84.21461500000002),
(41.063995, -81.57294499999998),
(39.741764, -84.209923),
(39.933804, -82.78944399999997),
(39.673203, -84.221886),
(38.997852, -84.043739),
(40.080402, -82.87745699999998),
(41.04023, -81.608711),
(39.749664, -84.201012),
(39.749916, -84.139359),
(41.039204, -81.631744),
(40.993484, -82.03600300000002),
(41.407391, -81.64676700000003),
(39.732441, -84.224243),
(39.773193, -84.187454),
(39.792931, -84.215195),
(39.948299, -82.047081),
(39.67263, -84.216522),
(41.172302, -83.64917),
(39.615208, -84.22731800000003),
(39.790604, -84.25140400000002),
(39.968788, -83.003754),
(39.905334, -82.968689),
(39.855103, -84.183678),
(41.414131, -81.632141),
(39.933804, -82.78944399999997),
(41.063995, -81.57294499999998),
(39.75782, -83.303154),
(41.42181, -81.734482),
(41.422199, -81.843018),
(39.819839, -84.189087),
(39.745735, -84.120644),
(40.004749, -82.867638),
(39.626396, -84.20768699999998),
(39.761005, -84.15855400000002),
(39.765377, -84.155914),
(39.936871, -82.936317),
(41.420666, -81.81644399999998),
(39.607605, -84.15598299999998),
(39.819809, -84.18710300000002),
(41.371895, -81.820786),
(39.999187, -83.075813),
(39.741695, -84.163101),
(39.917412, -83.014236),
```

```
(41.423275, -81.76001),
(39.993843, -82.985054),
(41.36974, -81.821457),
(41.420506, -81.816078),
(41.407391, -81.64676700000003),
(39.869961, -84.137642),
(39.75782, -83.303154),
(39.84182, -84.18960600000003),
(39.695045, -84.10419499999998),
(39.754223, -84.224075),
(39.759544, -84.191811),
(39.755566, -84.180534),
(39.960938, -84.189583),
(39.609039, -84.155869),
(39.953472, -82.960274),
(40.106297, -83.036079),
(39.812687, -84.22734799999998),
(39.724258, -84.213142),
(39.929554, -82.995773),
(39.976398, -83.119225),
(39.915928, -82.98246800000003),
(39.009884, -84.173073),
(39.737701, -84.134987),
(39.974285, -82.961174),
(40.152611, -82.686996),
(39.786629, -84.18869000000002),
(40.016937, -83.156601),
(40.003746, -83.15389300000002),
(39.736061, -84.093002),
(41.375961, -83.647453),
(39.953648, -82.960373),
(39.767757, -84.231712),
(39.800312, -84.218811),
(39.754055, -84.17588),
(39.712257, -84.201782),
(39.759201, -84.149101),
(39.759472, -84.243698),
(39.797668, -84.217506),
(39.763664, -84.193298),
(39.758121, -84.191414),
(40.266499, -82.927544),
(40.099384, -83.155396),
(39.981926, -83.046265),
(39.786633, -84.170837),
(39.765259, -84.13780200000002),
(40.10564, -82.949692),
(39.762531, -84.1427),
(41.428753, -83.61036700000002),
(39.985336, -82.78984100000002),
```

```
(39.955517, -82.87687700000002),
(39.858089, -82.827904),
(39.763615, -84.142586),
(39.933804, -82.78944399999997),
(40.136345, -83.00573),
(39.76516, -84.157234),
(39.772438, -84.14218100000002),
(39.981968, -83.008698),
(39.232101, -84.16014100000002),
(40.061661, -82.99807),
(40.082397, -82.906921),
(39.317173, -84.220894),
(39.690781, -84.205147),
(39.803726, -84.18514300000002),
(39.838779, -84.25831600000002),
(39.76144, -84.167038),
(39.740906, -84.182121),
(39.81963, -84.198273),
(39.79969000000001, -84.186996),
(39.72364, -84.1418),
(39.754055, -84.13889300000002),
(39.831997, -84.223694),
(41.403744, -82.282776),
(39.754055, -84.17588),
(39.630444, -84.19626600000002),
(39.801601, -84.194542),
(39.752647, -84.189774),
(39.933804, -82.78944399999997),
(39.571163, -84.202606),
(39.17505300000001, -84.236313),
(39.977322, -83.02109499999997),
(39.175728, -84.180763),
(39.68865200000001, -84.127998),
(39.628288, -84.226151),
(39.96764, -83.06081400000002),
(39.797958, -84.255135),
(39.783669, -84.155045),
(39.989044, -83.049088),
(39.819721, -84.191528),
(39.783089, -84.188911),
(39.590527, -84.17076899999998),
(41.40274, -81.817856),
(40.039909, -82.99683399999998),
(39.972336, -82.983917),
(40.00164, -82.923073),
(39.902088, -82.898178),
(39.833824, -83.100853),
(39.713467, -84.216095),
(39.640202, -84.235283),
```

```
(39.733723, -84.204971),
(41.351006, -83.622826),
(39.845287, -83.993217),
(39.607788, -84.228088),
(39.873238, -84.13720699999998),
(40.058132, -82.922752),
(41.420975, -81.689758),
(39.788414, -84.183693),
(39.836643, -84.060753),
(41.039829, -81.662872),
(38.581139, -82.00456199999998),
(39.754055, -84.17588),
(39.929844, -83.116486),
(39.962147, -83.092667),
(39.697266, -84.218781),
(40.993092, -81.662689),
(39.845287, -83.993217),
(41.041447, -81.565163),
(41.407391, -81.64676700000003),
(41.037266, -81.583786),
(39.59872100000001, -84.161491),
(40.018669, -81.565704),
(39.724251, -84.21315799999998),
(39.758556, -84.177246),
(39.790565, -84.25914),
(39.753105, -84.15158100000002),
(40.301411, -83.06575),
(39.732567, -84.15900400000002),
(39.800358, -82.708252),
(41.351006, -83.622826),
(39.790565, -84.25914),
(39.753105, -84.15158100000002),
(40.11195, -83.016663),
(38.892097, -84.208702),
(39.981602, -82.885704),
(41.337879, -82.216049),
(39.747112, -84.163826),
(40.087536, -82.98741899999997),
(39.782368, -84.203339),
(39.87857800000001, -83.049774),
(39.669388, -84.22225999999998),
(39.626953, -84.212288),
(39.812687, -84.22734799999998),
(39.819839, -84.189087),
(39.953262, -83.01795200000002),
(39.739319, -84.157219),
(39.91468, -83.01690699999997),
(41.037647, -81.584183),
(39.798611, -84.207466),
```

```
(40.083206, -82.90679899999998),
(39.749916, -84.139359),
(39.752735, -84.145386),
(39.689022, -84.220665),
(39.778011, -84.20049300000002),
(41.399147, -81.565674),
(39.59670300000001, -84.238388),
(39.973015, -83.014801),
(39.777355, -84.19957),
(41.139114, -81.597305),
(39.64724, -84.224319),
(39.745487, -84.16544300000002),
(39.732552, -84.12197900000002),
(39.778011, -84.20049300000002),
(40.002552, -83.11835500000002),
(39.780014, -84.162888),
(39.643101, -84.209923),
(39.767769, -84.17716999999998),
(39.812687, -84.22734799999998),
(39.986324, -82.985458),
(40.109913, -84.223648),
(39.83847, -84.246307),
(39.627991, -84.17021899999997),
(39.945835, -83.061295),
(39.591389, -84.229485),
(39.790302, -84.21390500000003),
(39.927441, -83.056198),
(39.946892, -82.91548900000002),
(40.002552, -83.11835500000002),
(39.79031, -84.029114),
(39.087887, -84.236076),
(39.188583, -84.258705),
(39.012833, -83.797585),
(40.151196, -84.22003199999997),
(40.033405, -82.910225),
(39.591389, -84.229485),
(40.080338, -82.880074),
(40.200333, -83.027435),
(40.02504, -82.904129),
(40.013062, -82.90213),
(40.065559, -82.906418),
(39.588413, -84.229774),
(39.902195, -83.084763),
(41.370506, -83.616997),
(39.945427, -82.846558),
(39.945751, -82.846252),
(41.316475, -81.649101),
(39.786633, -84.170837),
(39.768337, -84.194244),
```

```
(39.797005, -84.154556),
(40.005947, -82.858887),
(39.355782, -83.268311),
(40.030914, -82.994598),
(39.746147, -84.183167),
(39.771233, -84.1763),
(39.976398, -83.119225),
(39.948391, -82.943558),
(39.321606, -84.070908),
(39.085529, -84.178314),
(39.758556, -84.177246),
(39.939571, -83.009911),
(39.747753, -84.20558199999998),
(39.815628, -84.093338),
(39.635357, -84.216339),
(39.757317, -84.151299),
(39.768837, -84.20156899999998),
(39.749916, -84.139359),
(39.685413, -84.220871),
(39.919437, -82.775238),
(39.972076, -83.09818299999998),
(39.791206, -84.206223),
(39.778645, -84.20476500000002),
(39.749916, -84.139359),
(39.934155, -82.852524),
(39.755344, -84.203598),
(40.045822, -83.033424),
(40.051949, -83.03280600000002),
(39.943745, -83.073151),
(39.89217, -83.039169),
(41.113785, -81.611633),
(39.948158, -83.036201),
(39.819839, -84.189087),
(39.788414, -84.183693),
(39.747406, -84.239487),
(39.747753, -84.20558199999998),
(39.788414, -84.183693),
(41.351006, -83.622826),
(39.756203, -84.25322),
(39.94365300000001, -83.060631),
(39.985336, -82.78984100000002),
(40.077419, -82.907898),
(39.739037, -84.152107),
(40.083298, -84.117058),
(40.089439, -83.036194),
(39.735844, -84.121658),
(39.764576, -84.187706),
(39.747974, -84.215065),
(39.747753, -84.20558199999998),
(39.95232, -83.03761300000002),
```

```
(40.01757, -82.928978),
(39.95652800000001, -83.018555),
(39.939571, -83.009911),
(39.749916, -84.139359),
(41.137856, -83.65884399999999),
(41.112324, -81.60992399999998),
(40.109077, -83.091064),
(39.779266, -84.23465),
(41.420975, -81.689758),
(39.59670300000001, -84.238388),
(39.772823, -84.135612),
(40.109497, -83.090813),
(40.080402, -82.87745699999998),
(39.741417, -84.18055),
(39.760406, -84.195686),
(39.627781, -84.188354),
(39.706184, -84.217506),
(39.812687, -84.22734799999998),
(39.767471, -84.15786700000002),
(39.902088, -82.898178),
(40.699066, -84.089935),
(39.811291, -84.235176),
(39.811291, -84.235176),
(39.872375, -84.055153),
(39.916485, -82.897026),
(39.019779, -84.20581800000002),
(39.096836, -84.253273),
(40.006477, -83.030991),
(40.080402, -82.87745699999998),
(39.748081, -84.13945799999998),
(40.003059, -82.862289),
(41.36795, -81.821655),
(39.75777100000001, -84.142036),
(39.749916, -84.139359),
(39.106014, -84.147224),
(39.603619, -84.228577),
(39.636219, -84.21775799999998),
(39.790596, -84.253807),
(39.786629, -84.18869000000002),
(40.112133, -83.016998),
(39.753288, -84.191978),
(39.770718, -84.211449),
(39.995968, -83.024574),
(40.131283, -83.089745),
(40.103794, -83.13369),
(40.047691, -82.997086),
(39.93639, -82.881676),
(40.827763, -83.975716),
(39.67155800000001, -84.252678),
```

```
(40.028641, -83.01509899999998),
(40.02404, -82.99479699999998),
(40.11182, -82.901276),
(40.09917100000001, -83.134499),
(40.059711, -82.419037),
(39.801685, -84.253998),
(39.798016, -84.25018299999998),
(39.995106, -83.025337),
(39.740906, -84.182121),
(39.757221, -84.205894),
(39.745384, -84.183006),
(39.658932, -84.157677),
(39.780712, -84.18723299999998),
(39.754055, -84.17588),
(39.828781, -84.19839499999998),
(39.805809, -84.222435),
(39.896893, -83.723129),
(39.715988, -84.219238),
(39.896893, -83.723129),
(39.715988, -84.219238),
(39.933804, -82.78944399999997),
(39.668198, -84.147636),
(39.781204, -84.20558199999998),
(39.901142, -84.18789699999998),
(39.903053, -83.162949),
(39.783173, -84.117981),
(39.085529, -84.178314),
(40.111134, -82.992203),
(39.948391, -82.943558),
(39.786411, -84.25376899999998),
(39.782368, -84.203339),
(39.74688, -84.15986600000002),
(39.596725, -84.231667),
(39.784584, -84.15226700000002),
(39.769596, -84.176964),
(39.696461, -83.456673),
(39.97847700000001, -83.119835),
(39.773762, -84.14045),
(39.956909, -83.044914),
(39.788414, -84.183693),
(38.843636, -82.216187),
(39.741505, -84.16555),
(39.978668, -82.974152),
(39.59670300000001, -84.238388),
(39.919376, -82.936417),
(39.788647, -84.18390699999998),
(39.858292, -83.003548),
(39.832066, -83.000374),
(39.965687, -83.009842),
```

```
(39.974941, -83.10006),
(41.403664, -82.13942),
(41.323063, -82.616463),
(41.420448, -81.828468),
(39.816311, -84.130341),
(39.798302, -84.235229),
(39.767513, -84.194016),
(39.671932, -84.210815),
(39.781044, -84.242554),
(39.774738, -84.174683),
(39.767204, -84.162796),
(39.95166, -83.017601),
(39.950317, -83.01724200000002),
(39.732437, -84.20874),
(39.253136, -84.135544),
(39.790283, -84.190063),
(39.946922, -83.031151),
(39.956467, -83.045815),
(39.641197, -84.157799),
(41.425308, -81.897865),
(39.917412, -83.014236),
(39.981674, -82.984566),
(39.994205, -83.024689),
(40.004406, -82.918137),
(40.084126, -83.137772),
(39.959778, -82.997467),
(40.008327, -82.986023),
(39.977394, -83.150467),
(40.025917, -82.994576),
(41.40752, -81.64746099999998),
(41.060379, -81.554153),
(40.10564, -82.949692),
(39.732307, -84.240646),
(38.892097, -84.208702),
(39.536186, -84.235069),
(39.805882, -84.21578199999998),
(39.944817, -82.877861),
(39.370232, -84.249039),
(39.734035, -84.14502),
(39.808914, -83.97833299999998),
(39.735283, -84.161659),
(39.820873, -84.237328),
(40.105621, -83.091125),
(39.050854, -84.07486),
(39.751579, -84.239983),
(41.404495, -82.116928),
(39.625694, -84.200005),
(41.383537, -83.641235),
(39.76926, -84.236465),
```

```
(39.592564, -84.155518),
(39.788414, -84.183693),
(39.800083, -84.21869699999998),
(39.733723, -84.204971),
(39.738392, -84.140419),
(39.645313, -84.232414),
(39.651726, -84.233627),
(39.945255, -82.600075),
(39.783661, -84.166382),
(40.082298, -82.896858),
(39.75125900000001, -84.167778),
(39.748081, -84.13945799999998),
(39.943745, -83.073151),
(41.407391, -81.64676700000003),
(39.802471, -84.235107),
(39.787209, -84.205185),
(39.731812, -84.15831),
(39.733723, -84.204971),
(39.812775, -84.220001),
(39.794266, -84.215851),
(39.764576, -84.187706),
(39.819523, -84.20468100000002),
(39.788414, -84.183693),
(39.980274, -82.83625),
(39.773193, -84.187454),
(39.941711, -83.978874),
(39.737404, -84.185776),
(40.082397, -82.906921),
(39.184589, -84.235176),
(39.923038, -82.784538),
(39.745384, -84.183006),
(39.870964, -82.855354),
(39.981812, -83.118767),
(39.997391, -83.118156),
(39.632637, -84.20303299999998),
(39.9767, -83.045517),
(39.72419, -84.152336),
(39.770382, -84.194901),
(40.266563, -82.928535),
(41.42202800000001, -81.877464),
(39.820084, -84.16755699999999),
(39.754055, -84.17588),
(39.122822, -84.135796),
(39.803738, -84.22461700000002),
(39.596882, -84.117653),
(39.615669, -84.135452),
(39.973015, -83.014801),
(39.949711, -82.970123),
(39.949711, -82.970123),
```

```
(39.733723, -84.204971),
(39.74773, -84.172325),
(39.740417, -84.186539),
(39.625694, -84.200005),
(39.785046, -84.236137),
(39.749626, -84.178291),
(39.783344, -84.253738),
(39.820084, -84.16755699999999),
(39.739201, -84.230324),
(39.790916, -84.235085),
(39.80933, -84.235146),
(39.904541, -83.02684),
(39.982506, -82.98451999999997),
(39.75777100000001, -84.142036),
(39.736004, -84.14344799999998),
(39.59872100000001, -84.161491),
(39.74004, -84.194923),
(39.77626, -84.243752),
(39.62394000000001, -84.194153),
(39.73232700000001, -84.163055),
(39.772064, -84.253517),
(39.772064, -84.253517),
(40.263664, -82.927452),
(39.658939, -84.223244),
(39.67889, -84.221413),
(39.749786, -84.199493),
(39.745384, -84.183006),
(39.625694, -84.200005),
(40.120651, -82.973045),
(41.128685, -81.652153),
(39.959805, -83.370483),
(38.0853, -122.233017),
(37.631813, -122.084167),
(37.896564, -122.070717),
(37.334255, -122.032471),
(37.250729, -121.910713),
(37.701584, -121.906929),
(37.328312, -121.871811),
(37.719162, -122.448273),
(37.868114, -122.19593),
(37.700951, -121.80175),
(38.321342, -122.713707),
(38.005135, -122.037445),
(37.70071400000001, -121.773895),
(37.395771, -122.012772),
(37.701576, -121.872177),
(38.481682, -121.408768),
(38.682713, -121.33628799999998),
(37.391258, -121.995895),
```

```
(37.775215, -122.220863),
(37.418728, -121.970436),
(37.38237, -121.904358),
(37.701363, -121.849533),
(37.131519, -121.633308),
(38.303768, -122.708183),
(38.026699, -122.113785),
(38.220692, -122.607536),
(37.613815, -122.396652),
(38.646683, -121.376694),
(37.66637, -122.107491),
(37.70071400000001, -121.773895),
(38.54702, -122.808311),
(37.856171, -122.029907),
(37.119289, -121.626114),
(37.088932, -121.59922),
(38.601768, -122.864983),
(37.701061, -122.393188),
(37.448879, -122.12313799999998),
(37.732746, -122.404938),
(37.749485, -122.403236),
(37.454247, -121.923073),
(37.401531, -121.908859),
(38.563538, -121.637466),
(37.825397, -122.304802),
(37.8009, -122.228767),
(37.31665, -121.945389),
(38.36479600000001, -121.350449),
(38.551689, -121.699516),
(38.618011, -121.401459),
(37.49221, -122.386093),
(37.46876500000001, -122.155205),
(38.452553, -121.37175),
(37.741356, -121.580826),
(38.668053, -121.31208),
(37.78735, -122.247955),
(38.781418, -121.145889),
(37.608707, -122.068192),
(38.547523, -121.717918),
(37.977779, -122.06575),
(37.339455, -121.852043),
(37.553684, -122.296257),
(37.731983, -122.415413),
(37.854485, -122.219765),
(37.992588, -122.039635),
(38.58997, -121.26548),
(39.328636, -121.11116),
(38.481689, -121.396133),
(37.462418, -122.144402),
```

```
(38.404911, -121.912468),
(38.744423, -121.247093),
(38.203735, -121.556808),
(38.671062, -121.591881),
(38.566139, -121.336296),
(38.846264, -121.299644),
(38.564201, -121.513878),
(38.682323, -120.847435),
(37.923492, -122.513168),
(37.692444, -122.140167),
(37.804386, -121.989738),
(37.252171, -121.95821399999998),
(37.701649, -121.911278),
(37.807072, -122.475616),
(37.690559, -122.074669),
(37.766987, -121.335495),
(37.700043, -121.779655),
(37.933292, -121.296669),
(39.411289, -123.354477),
(38.556145, -121.443192),
(38.74008900000001, -121.273468),
(38.74008900000001, -121.273468),
(38.74398400000001, -121.226074),
(38.477348, -122.73214),
(37.364136, -122.124481),
(37.314922, -121.910576),
(37.955566, -122.370888),
(38.501938, -121.789146),
(38.714298, -121.540237),
(37.69278, -122.101685),
(39.267353, -121.02877),
(39.098984, -122.804314),
(37.785709, -122.244514),
(37.996063, -121.785194),
(38.558533, -122.447746),
(37.926357, -122.060768),
(37.352432, -121.837746),
(37.301517, -121.876823),
(37.808498, -122.366852),
(37.674477, -122.100662),
(37.549904, -122.025429),
(38.176636, -122.594177),
(37.590557, -122.36186200000002),
(38.606236, -121.508064),
(37.57436, -121.88523899999998),
(38.692123, -120.886917),
(38.242306, -122.08654),
(37.036915, -122.042366),
(37.708939, -121.723686),
```

```
(38.094425, -120.944992),
(37.359665, -121.995987),
(37.986759, -122.31488),
(37.997124, -122.286018),
(37.488846, -122.212852),
(38.578747, -121.307381),
(37.598789, -122.060852),
(38.005135, -122.037445),
(37.701225, -121.92247),
(38.686363, -121.33287),
(37.506924, -122.341293),
(36.666836, -121.684853),
(37.798454, -122.274803),
(38.522472, -121.521584),
(38.069008, -122.53876499999998),
(38.149895, -122.451241),
(37.910786, -122.31713899999998),
(37.861332, -122.210243),
(36.992775, -121.957268),
(36.609657, -121.898407),
(37.700855, -121.81778),
(36.676228, -121.701744),
(38.100746, -122.229858),
(37.495384, -121.923294),
(37.434715, -121.888161),
(37.588905, -121.870926),
(38.022526, -122.262299),
(37.740906, -122.196472),
(38.02545900000001, -122.112236),
(37.605614, -121.872658),
(36.624298, -121.841774),
(37.318287, -121.831551),
(38.510151, -121.459702),
(38.644184, -121.38831299999998),
(38.559532, -121.373085),
(37.74055900000001, -121.58578500000002),
(37.997124, -122.286018),
(37.337513, -122.059784),
(37.276817, -122.007233),
(38.494873, -121.45028700000002),
(38.409306, -121.35331),
(37.89658, -122.071281),
(37.955479, -121.324814),
(38.022526, -122.262299),
(37.687485, -122.133484),
(37.334255, -122.032471),
(37.381866, -121.963829),
(37.704407, -122.394241),
(38.169338, -122.199928),
```

```
(37.795124, -122.266594),
(37.276817, -122.007233),
(37.756107, -122.392464),
(37.793484, -122.263992),
(37.877388, -122.184158),
(38.478897, -121.421066),
(37.54932, -122.292496),
(38.588772, -121.404259),
(37.76329000000001, -122.216171),
(37.994179, -122.074287),
(38.641266, -121.473755),
(38.631653, -121.397972),
(37.560719, -122.035637),
(38.343533, -121.335594),
(38.546047, -121.509056),
(38.026489, -121.960167),
(39.202377, -121.062012),
(36.988647, -121.982933),
(37.539642, -121.923859),
(38.116135, -121.395554),
(38.23616, -122.461472),
(38.676525, -121.359612),
(37.784885, -122.39267),
(37.351471, -121.91433700000002),
(38.937412, -121.095352),
(38.772587, -121.253479),
(39.300751, -121.029884),
(37.410072, -121.879219),
(38.675251, -121.352432),
(38.667656, -121.33844),
(38.678547, -121.235008),
(39.143547, -121.061333),
(38.552616, -121.69381),
(37.570854, -122.032478),
(38.945255, -121.046089),
(38.929756, -121.08866100000002),
(37.31652800000001, -121.911858),
(37.315025, -121.909073),
(37.830669, -122.293343),
(37.641117, -122.406021),
(37.693237, -122.052116),
(37.852394, -122.029083),
(38.373032, -121.358482),
(38.64341, -121.434265),
(38.537041, -121.739067),
(38.559505, -121.467201),
(38.546028, -121.509315),
(38.942398, -121.09426100000002),
(38.618366, -121.529411),
```

```
(37.779552, -121.314514),
(37.851658, -121.282875),
(38.64315, -121.434677),
(38.658169, -120.969086),
(37.638012, -121.324409),
(37.31638, -121.951538),
(38.767025, -121.253128),
(39.233013, -121.03894),
(38.651775, -121.382942),
(37.634174, -122.414825),
(37.581238, -122.049042),
(38.601833, -121.382904),
(38.236176, -122.461472),
(37.785542, -122.39138),
(37.62746, -120.99706299999998),
(38.761791, -120.915413),
(38.229401, -122.116844),
(37.327499, -121.878082),
(37.750488, -121.379982),
(37.37682, -121.941536),
(37.38965200000001, -122.163582),
(37.70306, -122.07856),
(37.323593, -121.940826),
(37.690514, -121.918709),
(37.656654, -121.901588),
(37.753693, -122.151398),
(38.227737, -122.120537),
(37.769375, -122.405533),
(37.696819, -122.071869),
(37.8745, -122.306038),
(37.752502, -122.403008),
(37.7883, -121.300407),
(38.860706, -121.300247),
(37.86365900000001, -121.218956),
(37.54723, -122.37265),
(37.33442700000001, -121.936485),
(36.950603, -121.523094),
(38.660831, -121.337288),
(38.222748, -122.12883),
(37.343258, -121.846283),
(37.777836, -121.317451),
(38.722092, -121.22583799999998),
(38.246796, -122.627808),
(38.661053, -121.359779),
(37.813274, -121.034325),
(37.731277, -122.435219),
(37.315239, -121.914902),
(37.328579, -121.871277),
(37.882004, -121.641479),
(38.835583, -121.168533),
```

```
 (37.933201, -122.046196),
 (37.657974, -121.902954),
 (38.562939, -121.641922),
 (38.022778, -121.965698),
 (37.656654, -121.901588),
 (38.690273, -121.392136),
 (38.68111, -121.333244),
 (38.653061, -121.070541),
 ...]
```

```
import random

df_sample = df.sample(10000)

df.columns
```

```
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')
```

```
df['Visibility(mi)']
```

```
0          10.0
1          10.0
2          10.0
3           9.0
4           6.0
           ...
7728389    10.0
7728390    10.0
7728391    10.0
7728392    10.0
7728393     7.0
Name: Visibility(mi), Length: 7728394, dtype: float64
```

```
df['Visibility(mi)'].value_counts()
```

```
Visibility(mi)
10.0     6070231
7.0       217027
9.0       188529
8.0       149975
5.0       144153
           ...
78.0           1
101.0          1
72.0           1
67.0           1
43.0           1
Name: count, Length: 92, dtype: int64
```

```
df[(df.Severity == 4) & (df['Visibility(mi)'] <=10)]  # data when
severity is high and visibility is moderate
```

```
                ID   Source  Severity          Start_Time  \
619          A-620  Source2         4 2016-03-11 13:18:48
1197        A-1198  Source2         4 2016-06-24 22:28:49
1901        A-1902  Source2         4 2016-07-01 14:09:13
4143        A-4144  Source2         4 2016-07-25 14:23:33
4964        A-4965  Source2         4 2016-08-01 07:44:37
...            ...      ...       ...                  ...
7728354  A-7777722  Source1         4 2019-08-23 17:25:12
7728355  A-7777723  Source1         4 2019-08-23 17:25:12
7728366  A-7777734  Source1         4 2019-08-23 13:39:48
7728367  A-7777735  Source1         4 2019-08-23 13:39:48
7728380  A-7777748  Source1         4 2019-08-23 16:51:29

                    End_Time  Start_Lat   Start_Lng   End_Lat
End_Lng  \
619      2016-03-11 13:48:48  39.917412  -83.014236       NaN
NaN
1197     2016-06-24 22:58:49  37.321117 -121.899887       NaN
NaN
1901     2016-07-01 14:39:13  37.630623 -122.435043       NaN
NaN
4143     2016-07-25 15:11:13  37.339115 -121.851807       NaN
NaN
4964     2016-08-01 08:29:37  37.710648 -122.166687       NaN
NaN
...                      ...        ...         ...       ...
...
7728354  2019-08-23 17:54:00  38.995930 -121.672020  39.00317 -
121.662679
7728355  2019-08-23 17:54:00  39.003170 -121.662679  38.99593 -
121.672020
```

```
7728366  2019-08-23 14:05:33  33.685990 -117.886260  33.68537 -
117.885720
7728367  2019-08-23 14:05:33  33.687300 -117.890190  33.68599 -
117.886260
7728380  2019-08-23 17:21:02  33.779130 -117.887980  33.77991 -
117.890860

         Distance(mi)  ... Roundabout Station   Stop
Traffic_Calming  \
619             0.010  ...      False   False  False           False

1197            0.000  ...      False   False  False           False

1901            0.000  ...      False   False  False           False

4143            0.000  ...      False   False  False           False

4964            0.000  ...      False   False  False           False

...               ...  ...        ...     ...    ...             ...

7728354         0.708  ...      False   False  False           False

7728355         0.708  ...      False   False  False           False

7728366         0.053  ...      False   False  False           False

7728367         0.243  ...      False   False  False           False

7728380         0.174  ...      False   False  False           False


        Traffic_Signal Turning_Loop Sunrise_Sunset Civil_Twilight  \
619              False        False            Day            Day
1197             False        False          Night          Night
1901             False        False            Day            Day
4143             False        False            Day            Day
4964             False        False            Day            Day
...                ...          ...            ...            ...
7728354          False        False            Day            Day
7728355          False        False            Day            Day
7728366          False        False            Day            Day
7728367          False        False            Day            Day
7728380          False        False            Day            Day

        Nautical_Twilight Astronomical_Twilight
619                   Day                  Day
1197                Night                Night
1901                  Day                  Day
4143                  Day                  Day
4964                  Day                  Day
```

```
...                     ...                     ...
7728354                 Day                     Day
7728355                 Day                     Day
7728366                 Day                     Day
7728367                 Day                     Day
7728380                 Day                     Day

[196205 rows x 46 columns]
```

```
(len(df[df['Visibility(mi)'] <=2]) / len(df) )* 100. # total percentage of accidents in which visibility was less than 2 miles
```

```
4.760109280142808
```

```
(len(df[(df['Visibility(mi)'] <=2) & (df['Severity'] ==4)]) / len(df) ) * 100. # total percentage of accidents in which visibility was less than 2 miles and severity was very high
```

```
0.1428627991792344
```

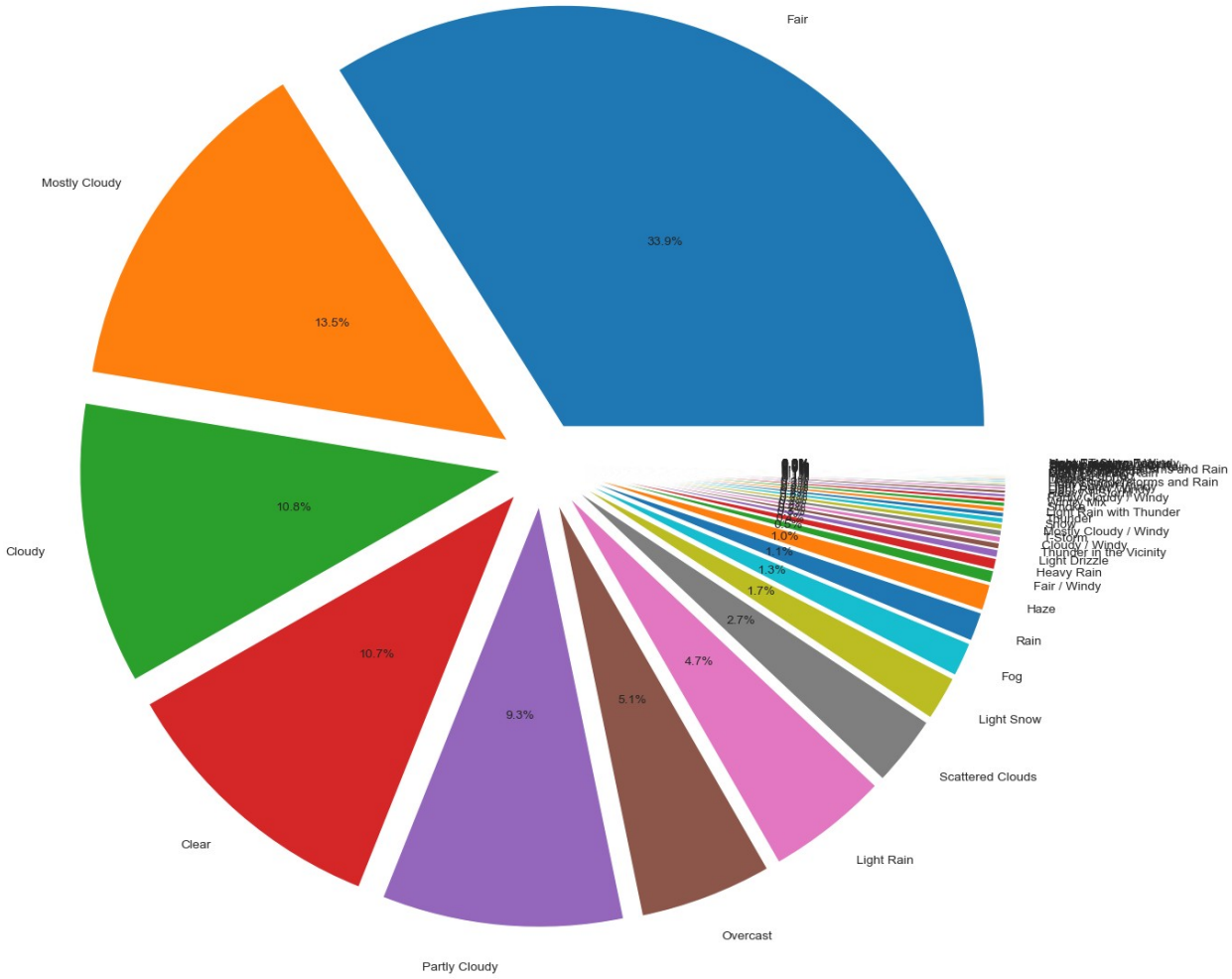
```
weather = df.Weather_Condition.value_counts()
```

```
weather[weather > 1000]  # Kind of weather when no. of accidents were greater than 1000
```

```
Weather_Condition
Fair                            2560802
Mostly Cloudy                   1016195
Cloudy                           817082
Clear                            808743
Partly Cloudy                    698972
Overcast                         382866
Light Rain                       352957
Scattered Clouds                 204829
Light Snow                       128680
Fog                               99238
Rain                              84331
Haze                              76223
Fair / Windy                      35671
Heavy Rain                        32309
Light Drizzle                     22684
Thunder in the Vicinity           17611
Cloudy / Windy                    17035
T-Storm                           16810
Mostly Cloudy / Windy             16508
Snow                              15537
Thunder                           14202
Light Rain with Thunder           13597
Smoke                             12668
Wintry Mix                        11703
Partly Cloudy / Windy             10241
```

```
Heavy T-Storm                          9671
Light Rain / Windy                     7946
Light Snow / Windy                     6826
Heavy Snow                             5003
Light Thunderstorms and Rain           4931
Drizzle                                4726
Thunderstorm                           4438
Patches of Fog                         4144
Mist                                   3539
Light Freezing Rain                    3465
N/A Precipitation                      3252
Shallow Fog                            3068
Heavy Thunderstorms and Rain           2485
Rain / Windy                           2372
Thunderstorms and Rain                 2217
Haze / Windy                           1595
Heavy Rain / Windy                     1523
Showers in the Vicinity                1514
Snow / Windy                           1285
Light Freezing Drizzle                 1240
Heavy T-Storm / Windy                  1096
Light Freezing Fog                     1001
Name: count, dtype: int64
```

```
import matplotlib.pyplot as plt

pie, ax = plt.subplots(figsize=[15,15])
labels = weather[weather > 1000].keys()
plt.pie(x=weather[weather > 1000], autopct="%.1f%%",
explode=[0.1]*len(weather[weather > 1000]), labels=labels,
pctdistance=0.5)
plt.show();
```




```
df['Temperature(F)']
```

```
0          36.9
1          37.9
2          36.0
3          35.1
4          36.0
           ...
7728389    86.0
7728390    70.0
7728391    73.0
7728392    71.0
7728393    79.0
Name: Temperature(F), Length: 7728394, dtype: float64
```

```
df['Temperature(F)'].value_counts()
```

```
Temperature(F)
 77.0     170991
 73.0     170898
 68.0     163767
 72.0     160498
 75.0     158448
           ...
 1.6           1
-21.5          1
 127.0         1
 158.0         1
 132.6         1
Name: count, Length: 860, dtype: int64
```

```
temperature = df['Temperature(F)'].value_counts()
```

```
temperature.index
```

```
Index([ 77.0,  73.0,  68.0,  72.0,  75.0,  70.0,  63.0,  59.0,  64.0,
79.0,
       ...
       113.4, 108.7, -32.8, -16.2, -13.2,   1.6, -21.5, 127.0, 158.0,
132.6],
      dtype='float64', name='Temperature(F)', length=860)
```

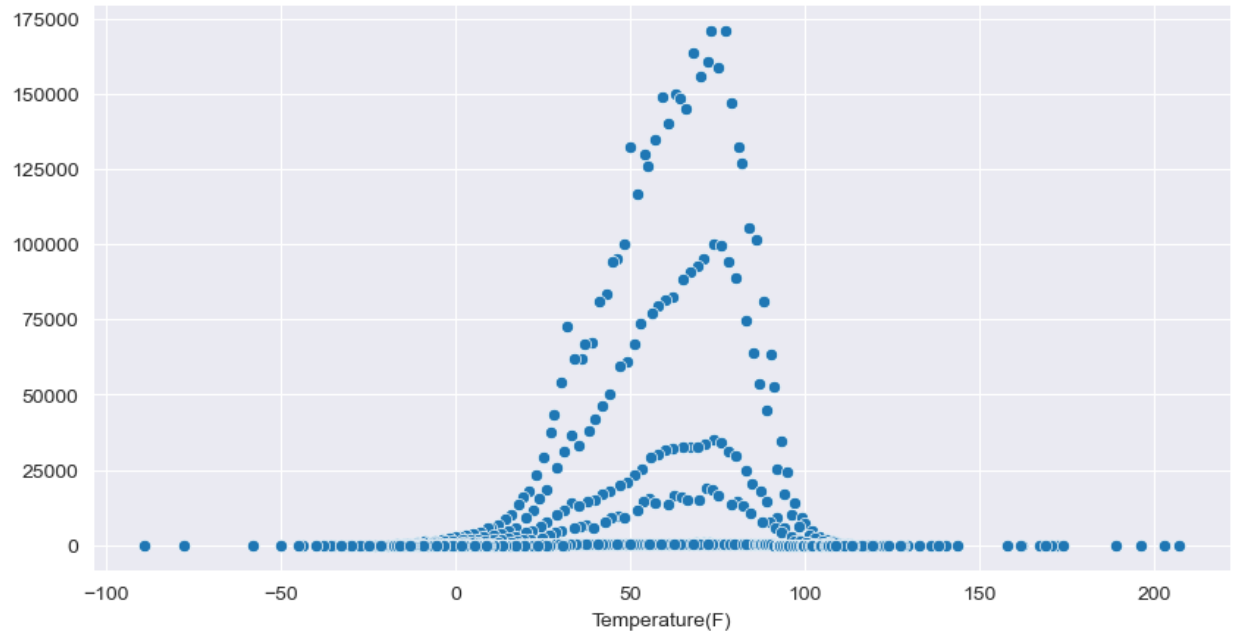
```
temperature.values
```

```
array([170991, 170898, 163767, 160498, 158448, 155568, 149787, 149017,
       148466, 147140, 144854, 140366, 134818, 132517, 132335, 129882,
       126838, 125909, 116664, 105573, 101392, 100146,  99828,  99380,
        95131,  95006,  94044,  93998,  92772,  90551,  88678,  88196,
        83239,  82609,  81313,  80997,  80877,  79400,  76945,  74755,
        73827,  72796,  67168,  66892,  66673,  63754,  63191,  62154,
        61907,  61152,  59488,  53974,  53605,  52905,  50464,  46184,
        44781,  43555,  41912,  37787,  37591,  36699,  35306,  34490,
        34079,  33764,  33145,  32887,  32703,  32666,  32069,  31882,
        31414,  31115,  30261,  29707,  29102,  28983,  25846,  25494,
        25174,  24592,  24228,  23428,  23165,  21019,  20620,  20101,
        19046,  18683,  18460,  18184,  18062,  17763,  17105,  17020,
        16598,  16350,  16115,  16106,  15360,  15359,  15252,  15075,
        14912,  14719,  14677,  14585,  14514,  14279,  14039,  13846,
        13833,  13780,  13716,  13262,  13062,  11753,  11516,  11404,
        10719,  10304,  10193,   9980,   9726,   9434,   9281,   9263,
         9050,   8978,   8751,   7890,   7762,   7709,   7588,   7448,
         6843,   6824,   6274,   6222,   6125,   5987,   5948,   5939,
         5777,   5738,   5636,   5592,   5039,   5009,   4791,   4790,
         4485,   4414,   4180,   4115,   3793,   3675,   3376,   3347,
         3301,   3187,   3041,   2996,   2977,   2954,   2775,   2718,
         2587,   2447,   2326,   2183,   2022,   2011,   2000,   1899,
         1845,   1804,   1717,   1691,   1685,   1674,   1619,   1605,
```

```
        1432,   1371,   1343,   1342,   1275,   1262,   1229,   1171,
        1054,   1045,   1008,    967,    944,    933,    905,    838,
         826,    812,    756,    705,    692,    687,    674,    674,
         630,    628,    627,    609,    605,    591,    586,    563,
         562,    554,    551,    548,    544,    540,    539,    536,
         536,    535,    533,    531,    528,    525,    522,    521,
         521,    515,    514,    513,    511,    503,    503,    498,
         495,    493,    491,    489,    488,    488,    487,    485,
         485,    483,    482,    481,    480,    479,    474,    474,
         468,    468,    466,    463,    462,    461,    459,    457,
         457,    456,    456,    456,    456,    455,    453,    450,
         449,    447,    447,    447,    446,    445,    443,    442,
         441,    437,    437,    436,    435,    433,    431,    430,
         428,    427,    426,    425,    422,    421,    418,    417,
         417,    417,    417,    416,    414,    414,    413,    412,
         411,    410,    410,    409,    407,    405,    404,    403,
         401,    401,    401,    400,    395,    394,    392,    391,
         391,    388,    386,    385,    385,    384,    384,    383,
         383,    382,    380,    380,    378,    376,    376,    376,
         375,    375,    374,    373,    373,    372,    371,    368,
         367,    367,    367,    367,    367,    366,    364,    363,
         363,    362,    362,    360,    359,    358,    356,    355,
         355,    354,    354,    353,    351,    351,    349,    349,
         349,    348,    346,    343,    343,    343,    343,    342,
         341,    340,    337,    336,    336,    335,    334,    333,
         333,    332,    331,    331,    330,    329,    327,    324,
         324,    324,    321,    320,    319,    318,    318,    317,
         317,    315,    314,    313,    312,    312,    311,    310,
         309,    309,    309,    308,    307,    307,    307,    305,
         305,    304,    302,    300,    300,    299,    298,    298,
         295,    295,    293,    293,    293,    289,    289,    284,
         284,    284,    283,    282,    281,    272,    268,    267,
         267,    266,    264,    263,    262,    261,    261,    260,
         259,    258,    256,    255,    255,    255,    255,    253,
         250,    248,    247,    246,    245,    245,    239,    239,
         236,    235,    234,    232,    231,    230,    228,    226,
         226,    225,    224,    224,    223,    214,    214,    212,
         211,    211,    206,    201,    198,    197,    192,    190,
         190,    190,    189,    188,    186,    185,    185,    183,
         179,    178,    177,    176,    174,    174,    173,    172,
         166,    165,    164,    163,    159,    158,    158,    155,
         154,    152,    151,    151,    150,    147,    144,    135,
         131,    131,    128,    128,    122,    120,    120,    117,
         117,    116,    115,    114,    113,    113,    112,    112,
         110,    109,    109,    109,    108,    108,    105,    104,
         102,    101,     99,     97,     97,     95,     90,     90,
          89,     89,     84,     84,     83,     83,     82,     81,
          80,     80,     79,     79,     75,     74,     74,     72,
          71,     69,     69,     68,     67,     67,     66,     66,
```

```
           66,     65,     64,     63,     63,     63,     62,     62,
           60,     60,     59,     57,     57,     56,     55,     55,
           54,     54,     53,     52,     51,     51,     50,     50,
           48,     47,     47,     47,     46,     46,     46,     45,
           44,     42,     42,     42,     41,     40,     39,     39,
           38,     38,     37,     37,     37,     37,     36,     35,
           35,     34,     33,     32,     32,     32,     32,     32,
           31,     31,     30,     29,     27,     27,     27,     27,
           27,     27,     26,     26,     26,     26,     25,     25,
           25,     25,     24,     24,     24,     24,     24,     24,
           23,     23,     22,     22,     22,     22,     21,     21,
           20,     20,     20,     20,     20,     19,     19,     18,
           17,     16,     16,     16,     16,     15,     15,     15,
           15,     14,     14,     13,     13,     13,     13,     13,
           13,     13,     13,     13,     13,     13,     13,     12,
           12,     12,     12,     12,     12,     12,     12,     11,
           11,     11,     11,     11,     11,     11,     10,     10,
           10,     10,     10,     10,     10,     10,     10,     10,
            9,      9,      8,      8,      8,      8,      8,      7,
            7,      7,      7,      7,      7,      7,      7,      7,
            6,      6,      6,      6,      6,      6,      5,      5,
            5,      5,      5,      5,      5,      5,      5,      5,
            5,      5,      5,      4,      4,      4,      4,      4,
            4,      4,      4,      4,      4,      4,      4,      4,
            4,      4,      4,      4,      4,      4,      4,      4,
            3,      3,      3,      3,      3,      3,      3,      3,
            3,      3,      3,      3,      3,      3,      3,      2,
            2,      2,      2,      2,      2,      2,      2,      2,
            2,      2,      2,      2,      2,      2,      2,      2,
            2,      2,      2,      2,      2,      2,      2,      2,
            2,      1,      1,      1,      1,      1,      1,      1,
            1,      1,      1,      1,      1,      1,      1,      1,
            1,      1,      1,      1,      1,      1,      1,      1,
            1,      1,      1,      1,      1,      1,      1,      1,
            1,      1,      1,      1,      1,      1,      1,      1,
            1,      1,      1,      1], dtype=int64)
```

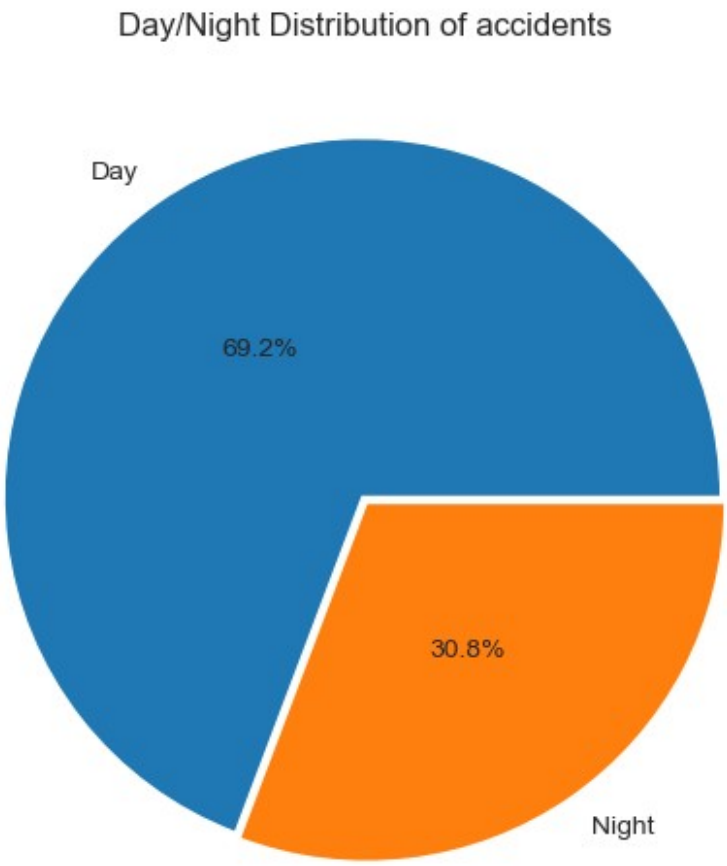
```
import seaborn as sns

plt.figure(figsize=(10,5))
sns.scatterplot(x=temperature.index, y=temperature.values)
plt.show();
```




```
df.Sunrise_Sunset.value_counts()
```

```
Sunrise_Sunset
Day      5334553
Night    2370595
Name: count, dtype: int64
```

```
pie, ax = plt.subplots(figsize=[6,6])
labels = df.Sunrise_Sunset.value_counts().keys()
plt.pie(x=df.Sunrise_Sunset.value_counts(), autopct="%.1f%%",
explode=[0.01]*len(df.Sunrise_Sunset.value_counts()), labels=labels,
pctdistance=0.5)
plt.title("Day/Night Distribution of accidents")
plt.show();
```




```
df.columns

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')
```

```
amenity = df.Amenity.groupby(df.Severity).value_counts()
amenity
```

```
Severity  Amenity
1         False        65987
          True          1379
2         False      6068089
          True         88892
3         False      1295307
          True          4030
4         False       202677
          True          2033
Name: count, dtype: int64
```

```
amenity.index
```

```
MultiIndex([(1, False),
            (1,  True),
            (2, False),
            (2,  True),
            (3, False),
            (3,  True),
            (4, False),
            (4,  True)],
           names=['Severity', 'Amenity'])
```

```
no_exit = df.No_Exit.groupby(df.Severity).value_counts()
no_exit
```

```
Severity  No_Exit
1         False        66985
          True           381
2         False      6140021
          True         16960
3         False      1297502
          True          1835
4         False       204341
          True           369
Name: count, dtype: int64
```

```
railway = df.Railway.groupby(df.Severity).value_counts()
railway
```

```
Severity  Railway
1         False        66302
          True          1064
2         False      6101200
          True         55781
```

```
3         False      1290760
          True          8577
4         False       203153
          True          1557
Name: count, dtype: int64
```

```
traffic_calming =
df.Traffic_Calming.groupby(df.Severity).value_counts()
traffic_calming
```

```
Severity  Traffic_Calming
1         False                67280
          True                    86
2         False              6150420
          True                  6561
3         False              1298485
          True                   852
4         False               204611
          True                    99
Name: count, dtype: int64
```

```
stop = df.Stop.groupby(df.Severity).value_counts()
stop
```

```
Severity  Stop
1         False      64723
          True        2643
2         False    5958591
          True      198390
3         False    1291686
          True        7651
4         False     199023
          True        5687
Name: count, dtype: int64
```

```
traffic_signal = df.Traffic_Signal.groupby(df.Severity).value_counts()
traffic_signal
```

```
Severity  Traffic_Signal
1         False               41025
          True                26341
2         False             5148309
          True              1008672
3         False             1210728
          True                88609
4         False              184560
          True                20150
Name: count, dtype: int64
```

```
give_way = df.Give_Way.groupby(df.Severity).value_counts()
give_way
```

```
Severity  Give_Way
1         False         66817
          True            549
2         False       6126858
          True          30123
3         False       1294598
          True           4739
4         False        203539
          True           1171
Name: count, dtype: int64
```

```
bump = df.Bump.groupby(df.Severity).value_counts()
bump
```

```
Severity  Bump
1         False      67332
          True          34
2         False    6153837
          True        3144
3         False    1299031
          True         306
4         False     204680
          True          30
Name: count, dtype: int64
```

```
crossing = df.Crossing.groupby(df.Severity).value_counts()
crossing
```

```
Severity  Crossing
1         False         48675
          True          18691
2         False       5363435
          True         793546
3         False       1251305
          True          48032
4         False        191216
          True          13494
Name: count, dtype: int64
```

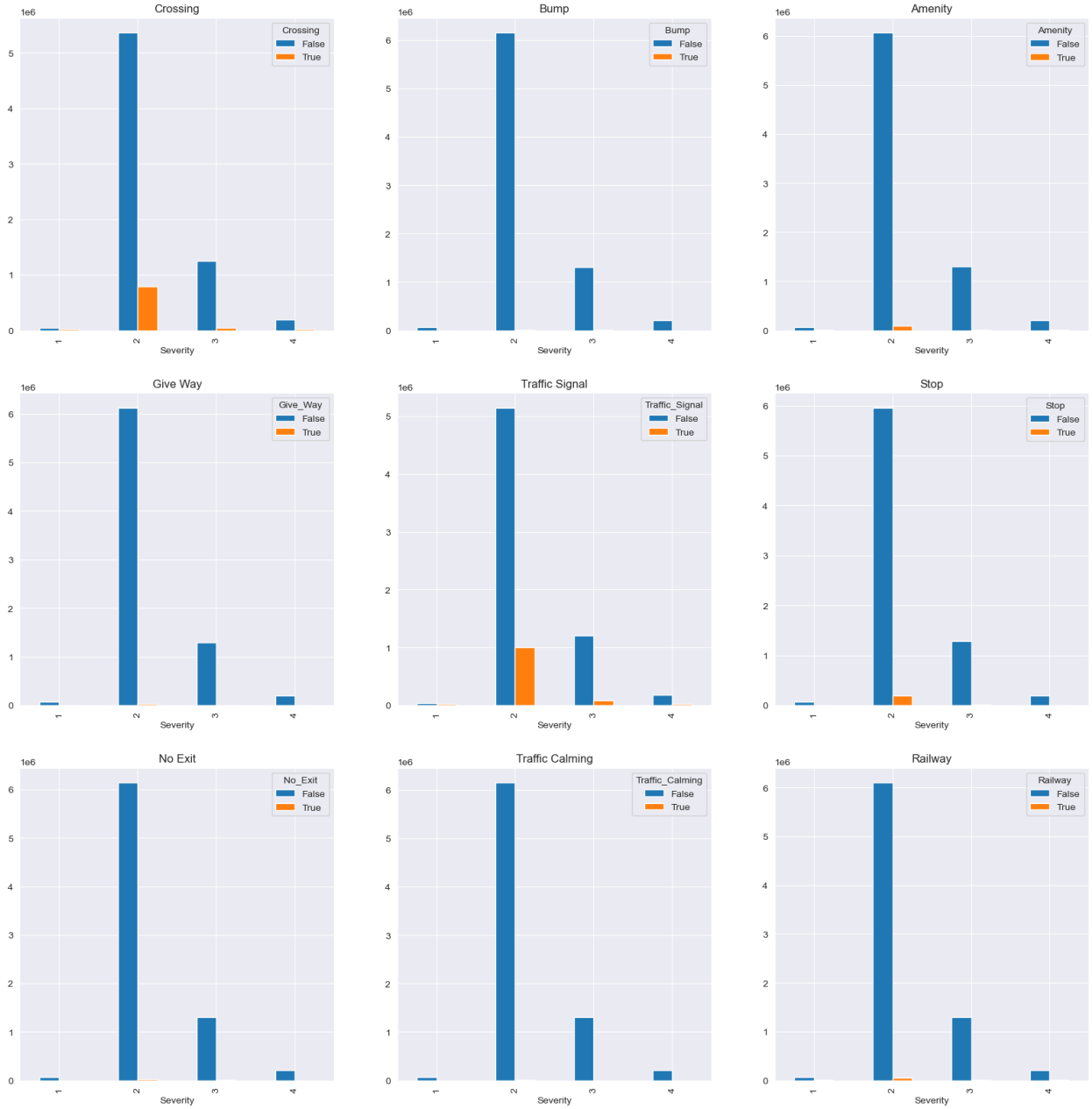
```
df.Turning_Loop.value_counts()
```

```
Turning_Loop
False    7728394
Name: count, dtype: int64
```

```
fig, ax = plt.subplots(3,3, figsize=(20, 20))

crossing.unstack().plot(kind='bar', ax=ax[0,0], title="Crossing")
bump.unstack().plot(kind='bar', ax=ax[0,1], title="Bump")
amenity.unstack().plot(kind='bar', ax=ax[0,2], title="Amenity")
give_way.unstack().plot(kind='bar', ax=ax[1,0], title="Give Way")
```

```
traffic_signal.unstack().plot(kind='bar', ax=ax[1,1], title="Traffic Signal")
stop.unstack().plot(kind='bar', ax=ax[1,2], title="Stop")
no_exit.unstack().plot(kind='bar', ax=ax[2,0], title="No Exit")
traffic_calming.unstack().plot(kind='bar', ax=ax[2,1], title="Traffic Calming")
railway.unstack().plot(kind='bar', ax=ax[2,2], title="Railway")
```

```
<Axes: title={'center': 'Railway'}, xlabel='Severity'>
```



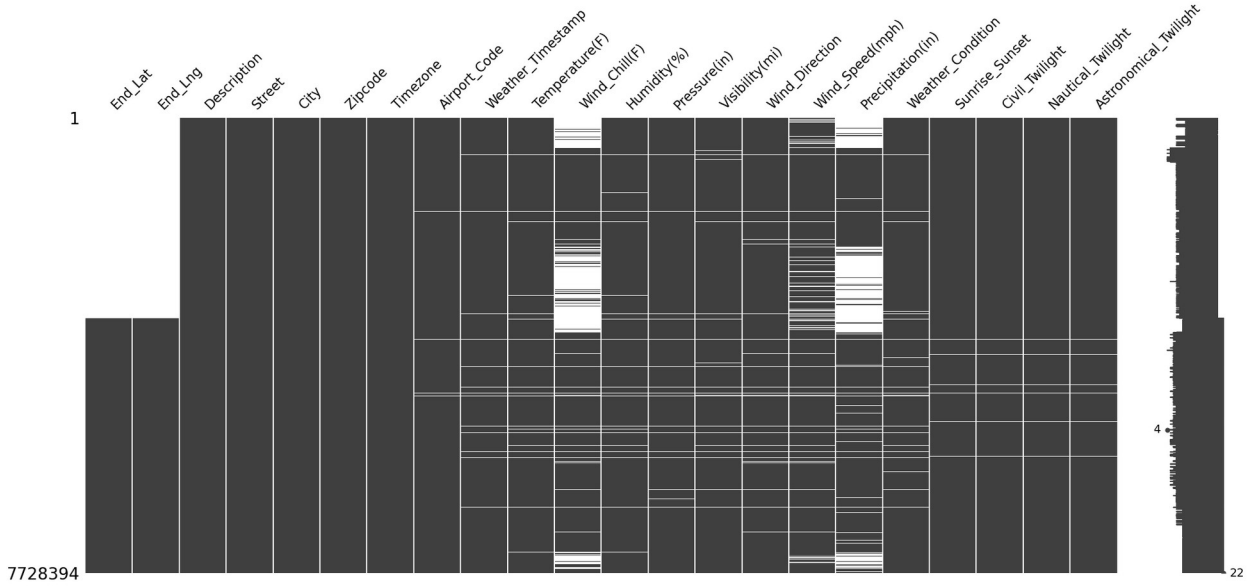
```
import pandas as pd
data = pd.read_csv("C:\\Users\\jayaraman\\Downloads\\archive (2)\\
US_Accidents_March23.csv")
null_cols = [i for i in data.columns if data[i].isnull().any()]
print(null_cols)
```

```
['End_Lat', 'End_Lng', 'Description', 'Street', 'City', 'Zipcode',
'Timezone', 'Airport_Code', 'Weather_Timestamp', 'Temperature(F)',
'Wind_Chill(F)', 'Humidity(%)', 'Pressure(in)', 'Visibility(mi)',
'Wind_Direction', 'Wind_Speed(mph)', 'Precipitation(in)',
'Weather_Condition', 'Sunrise_Sunset', 'Civil_Twilight',
'Nautical_Twilight', 'Astronomical_Twilight']
```

```
!pip install missingno
import missingno as mn
mn.matrix(data[null_cols]);
```

```
Requirement already satisfied: missingno in c:\users\jayaraman\
anaconda3\lib\site-packages (0.5.2)
Requirement already satisfied: numpy in c:\users\jayaraman\anaconda3\
lib\site-packages (from missingno) (1.26.4)
Requirement already satisfied: matplotlib in c:\users\jayaraman\
anaconda3\lib\site-packages (from missingno) (3.8.0)
Requirement already satisfied: scipy in c:\users\jayaraman\anaconda3\
lib\site-packages (from missingno) (1.11.4)
Requirement already satisfied: seaborn in c:\users\jayaraman\
anaconda3\lib\site-packages (from missingno) (0.12.2)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\jayaraman\
anaconda3\lib\site-packages (from matplotlib->missingno) (1.2.0)
Requirement already satisfied: cycler>=0.10 in c:\users\jayaraman\
anaconda3\lib\site-packages (from matplotlib->missingno) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\
jayaraman\anaconda3\lib\site-packages (from matplotlib->missingno)
(4.25.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\
jayaraman\anaconda3\lib\site-packages (from matplotlib->missingno)
(1.4.4)
Requirement already satisfied: packaging>=20.0 in c:\users\jayaraman\
anaconda3\lib\site-packages (from matplotlib->missingno) (23.1)
Requirement already satisfied: pillow>=6.2.0 in c:\users\jayaraman\
anaconda3\lib\site-packages (from matplotlib->missingno) (10.2.0)
Requirement already satisfied: pyparsing>=2.3.1 in c:\users\jayaraman\
anaconda3\lib\site-packages (from matplotlib->missingno) (3.0.9)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\
jayaraman\anaconda3\lib\site-packages (from matplotlib->missingno)
(2.8.2)
Requirement already satisfied: pandas>=0.25 in c:\users\jayaraman\
anaconda3\lib\site-packages (from seaborn->missingno) (2.1.4)
Requirement already satisfied: pytz>=2020.1 in c:\users\jayaraman\
anaconda3\lib\site-packages (from pandas>=0.25->seaborn->missingno)
```

```
(2023.3.post1)
Requirement already satisfied: tzdata>=2022.1 in c:\users\jayaraman\
anaconda3\lib\site-packages (from pandas>=0.25->seaborn->missingno)
(2023.3)
Requirement already satisfied: six>=1.5 in c:\users\jayaraman\
anaconda3\lib\site-packages (from python-dateutil>=2.7->matplotlib-
>missingno) (1.16.0)
```




```
new_data_a = data.drop(columns=["End_Lng", "End_Lat"], axis=0)

new_data_b = new_data_a.dropna(subset =
['Visibility(mi)','Weather_Condition','Humidity(%)','Temperature(F)','
Wind_Direction','Pressure(in)','Weather_Timestamp','Airport_Code','Tim
ezone','Zipcode','Sunrise_Sunset','Civil_Twilight','Nautical_Twilight'
,'Astronomical_Twilight','City','Description'])

new_data_b.isnull().sum()
```

```
ID                          0
Source                      0
Severity                    0
Start_Time                  0
End_Time                    0
Start_Lat                   0
Start_Lng                   0
Distance(mi)                0
Description                 0
Street                  10214
City                        0
County                      0
State                       0
```

```
Zipcode                          0
Country                          0
Timezone                         0
Airport_Code                     0
Weather_Timestamp                0
Temperature(F)                   0
Wind_Chill(F)              1769892
Humidity(%)                      0
Pressure(in)                     0
Visibility(mi)                   0
Wind_Direction                   0
Wind_Speed(mph)             375174
Precipitation(in)          2039619
Weather_Condition                0
Amenity                          0
Bump                             0
Crossing                         0
Give_Way                         0
Junction                         0
No_Exit                          0
Railway                          0
Roundabout                       0
Station                          0
Stop                             0
Traffic_Calming                  0
Traffic_Signal                   0
Turning_Loop                     0
Sunrise_Sunset                   0
Civil_Twilight                   0
Nautical_Twilight                0
Astronomical_Twilight            0
dtype: int64
```

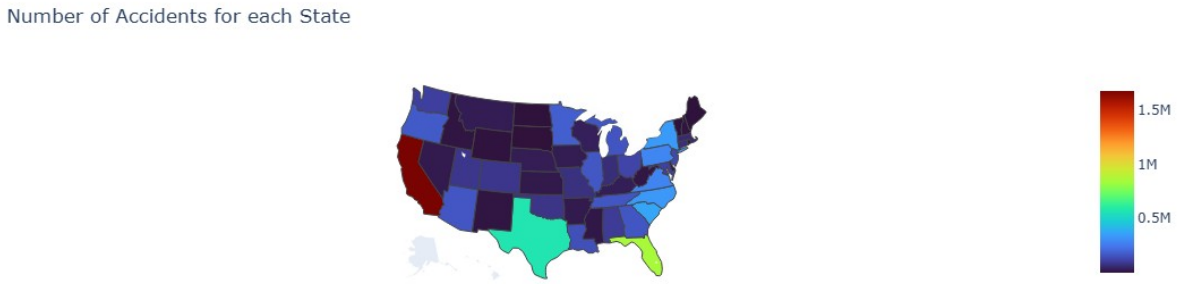
```
final_data = new_data_b.drop(columns = 'ID', axis=0)

final_data.isnull().sum()
```

```
Source                           0
Severity                         0
Start_Time                       0
End_Time                         0
Start_Lat                        0
Start_Lng                        0
Distance(mi)                     0
Description                      0
Street                       10214
City                             0
County                           0
State                            0
Zipcode                          0
```

```
Country                          0
Timezone                         0
Airport_Code                     0
Weather_Timestamp                0
Temperature(F)                   0
Wind_Chill(F)              1769892
Humidity(%)                      0
Pressure(in)                     0
Visibility(mi)                   0
Wind_Direction                   0
Wind_Speed(mph)             375174
Precipitation(in)          2039619
Weather_Condition                0
Amenity                          0
Bump                             0
Crossing                         0
Give_Way                         0
Junction                         0
No_Exit                          0
Railway                          0
Roundabout                       0
Station                          0
Stop                             0
Traffic_Calming                  0
Traffic_Signal                   0
Turning_Loop                     0
Sunrise_Sunset                   0
Civil_Twilight                   0
Nautical_Twilight                0
Astronomical_Twilight            0
dtype: int64
```

```
!pip install plotly
import plotly.graph_objects as go
state_counts = final_data["State"].value_counts()
fig = go.Figure(data=go.Choropleth(locations=state_counts.index,
z=state_counts.values.astype(float), locationmode="USA-states",
colorscale="turbo"))
fig.update_layout(title_text="Number of Accidents for each State",
geo_scope="usa")
fig.show()
```

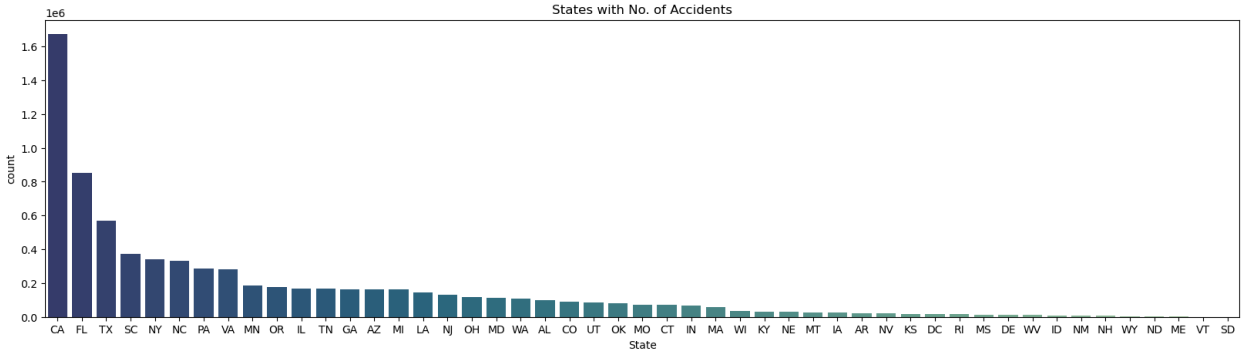
```
Requirement already satisfied: plotly in c:\users\jayaraman\anaconda3\
lib\site-packages (5.9.0)
Requirement already satisfied: tenacity>=6.2.0 in c:\users\jayaraman\
anaconda3\lib\site-packages (from plotly) (8.2.2)
```



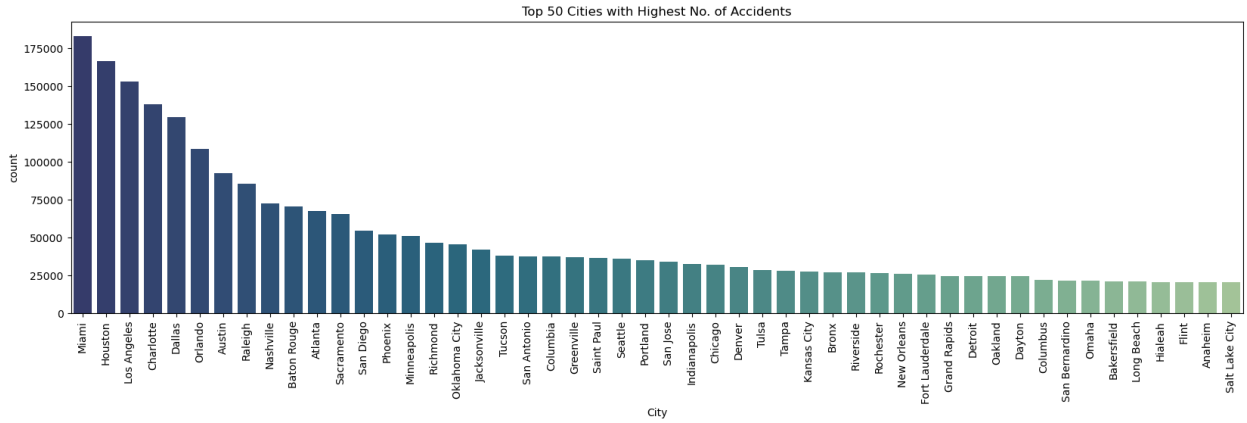

```
print("State Code: ", final_data.State.unique())
print("Total No. of State in Dataset: ",
len(final_data.State.unique()))
```

```
State Code:  ['OH' 'WV' 'CA' 'FL' 'GA' 'SC' 'NE' 'IA' 'IL' 'MO' 'WI'
'IN' 'MI' 'NJ'
 'NY' 'CT' 'MA' 'RI' 'NH' 'PA' 'KY' 'MD' 'VA' 'DC' 'DE' 'TX' 'WA' 'OR'
 'AL' 'NC' 'AZ' 'TN' 'LA' 'MN' 'CO' 'OK' 'NV' 'UT' 'KS' 'NM' 'AR' 'MS'
 'ME' 'VT' 'WY' 'ID' 'ND' 'MT' 'SD']
Total No. of State in Dataset:  49
```

```
import matplotlib.pyplot as plt
import seaborn as sns
fig, ax = plt.subplots(figsize = (20,5))
c = sns.countplot(x="State", data=final_data, orient = 'v', palette =
"crest_r", order = final_data['State'].value_counts().index)
c.set_title("States with No. of Accidents");
```



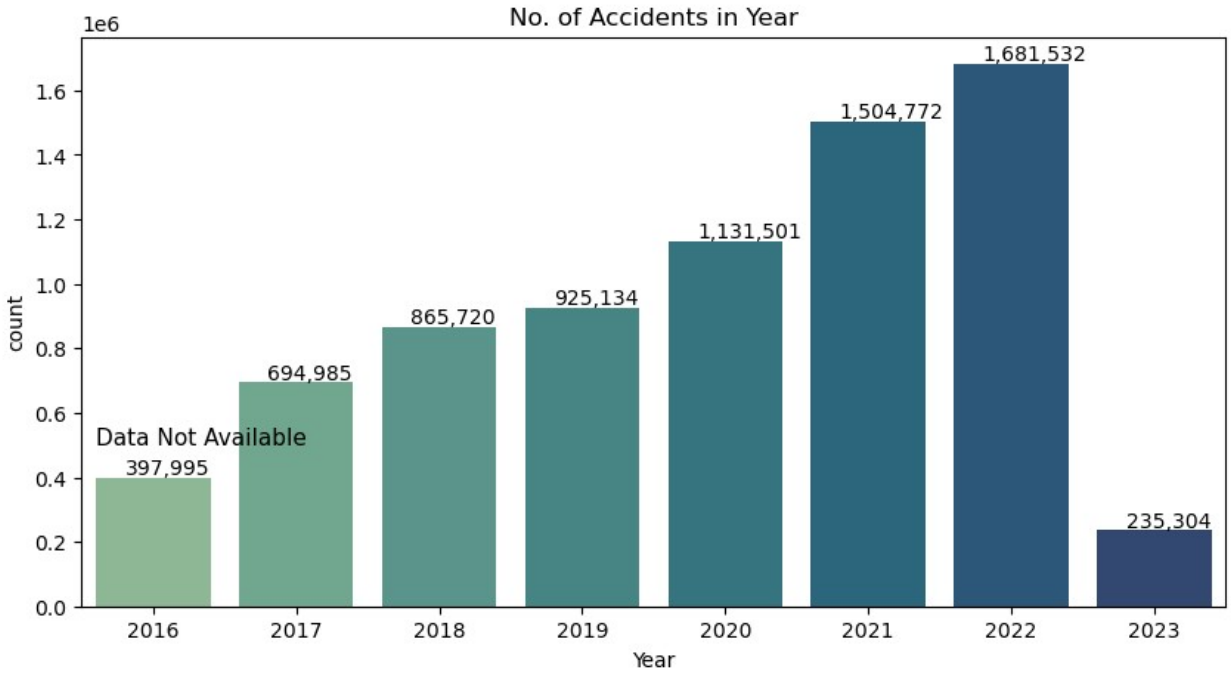

```
fig, ax = plt.subplots(figsize = (20,5))
c = sns.countplot(x="City", data=final_data,
order=final_data.City.value_counts().iloc[:50].index, orient = 'v',
palette = "crest_r")
c.set_title("Top 50 Cities with Highest No. of Accidents")
c.set_xticklabels(c.get_xticklabels(), rotation=90)
plt.show()
```



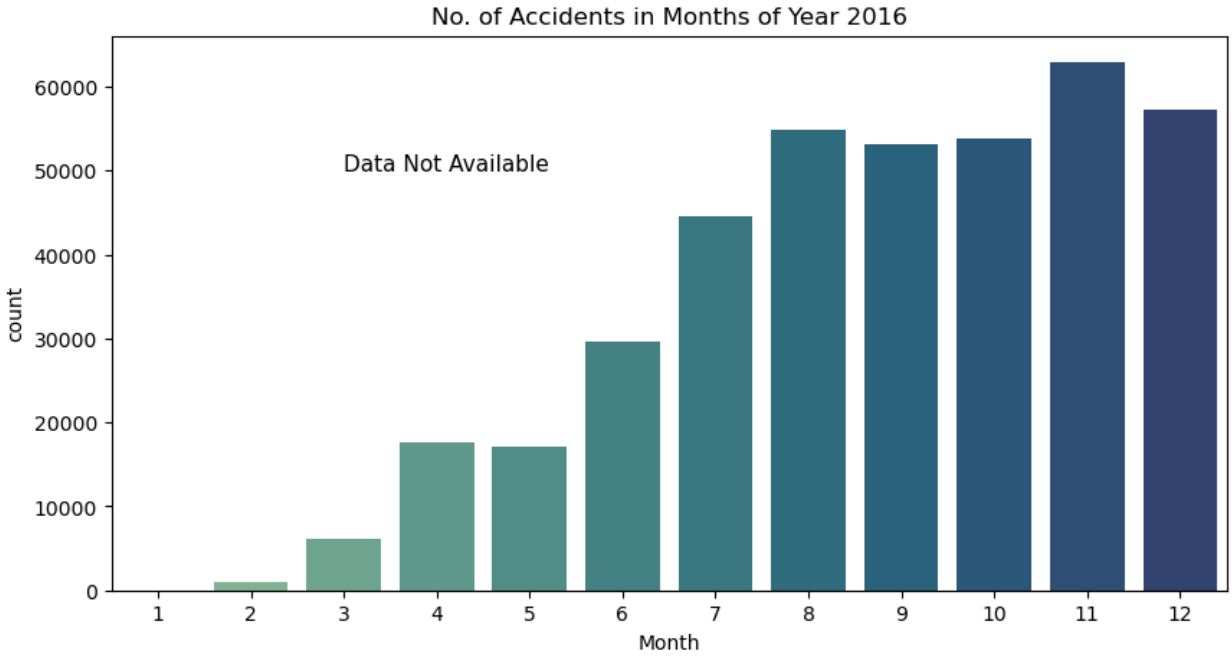

```
final_data.Start_Time =
pd.to_datetime(final_data.Start_Time,format="ISO8601")
final_data.Start_Time[0]
```

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

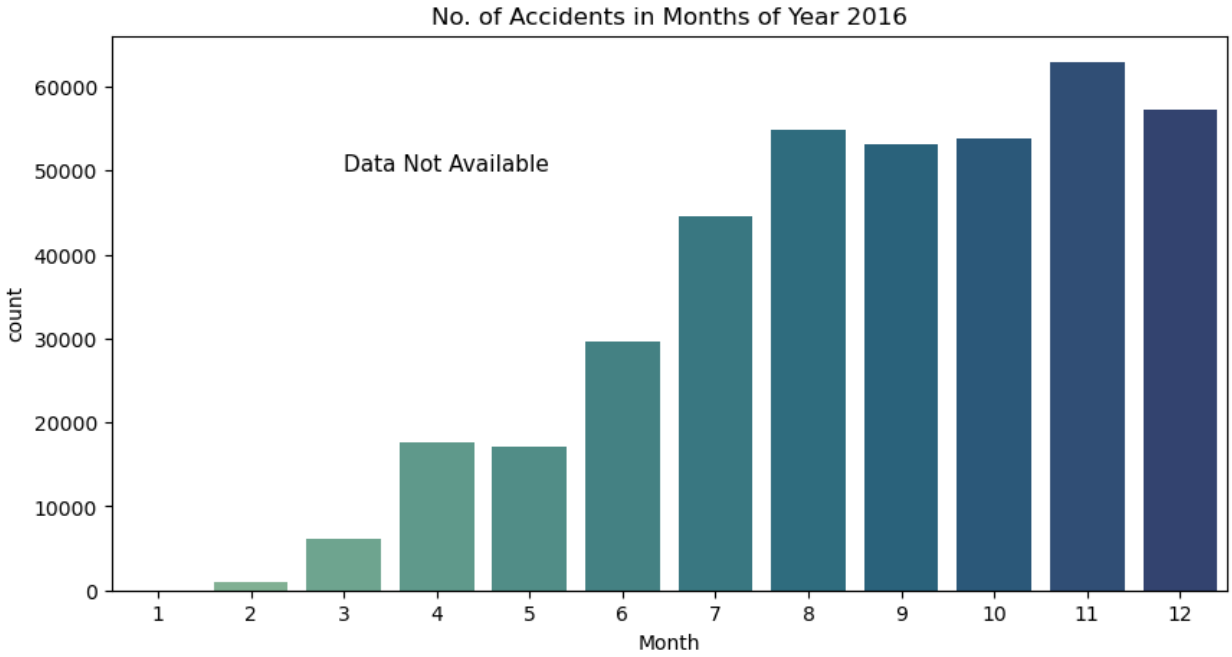
```
final_data['Month'] = final_data['Start_Time'].dt.month
final_data['Year'] = final_data['Start_Time'].dt.year
final_data['Hour'] = final_data['Start_Time'].dt.hour
final_data['Weekday'] = final_data['Start_Time'].dt.weekday
#yearly data subset
data_2016 = final_data[final_data.Start_Time.dt.year == 2016]
data_2017 = final_data[final_data.Start_Time.dt.year == 2017]
data_2018 = final_data[final_data.Start_Time.dt.year == 2018]
data_2019 = final_data[final_data.Start_Time.dt.year == 2019]
data_2020 = final_data[final_data.Start_Time.dt.year == 2020]
data_2017_2019 = final_data[(final_data["Year"] >= 2017) &
(final_data["Year"] <= 2019)]

fig, ax = plt.subplots(figsize = (10,5))
c = sns.countplot(x="Year", data=final_data, orient = 'v', palette =
"crest")
plt.annotate('Data Not Available',xy=(-0.4,500000), fontsize=11)
c.set_title("No. of Accidents in Year")
for i in ax.patches:
    count = '{:,.0f}'.format(i.get_height())
    x = i.get_x()+i.get_width()-0.60
    y = i.get_height()+10000
    ax.annotate(count, (x, y))
plt.show()
```



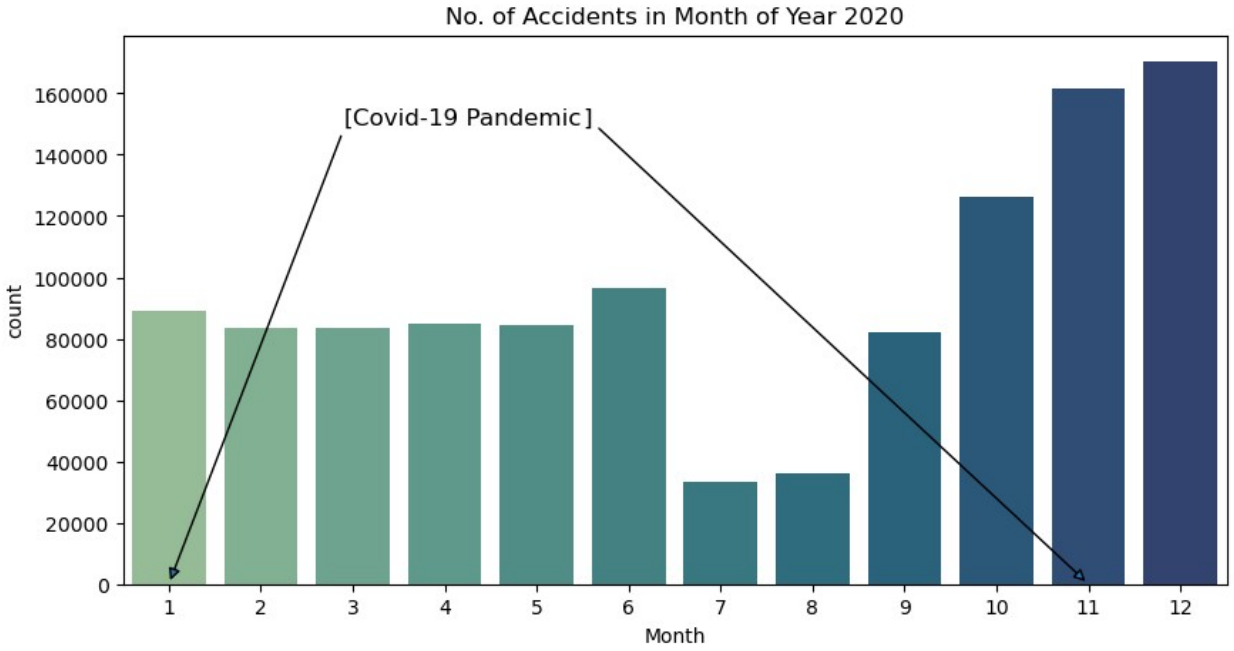

```
fig, ax = plt.subplots(figsize = (10,5))
c = sns.countplot(x="Month", data=data_2016, orient = 'v', palette =
"crest")
plt.annotate('Data Not Available',xy=(2,50000), fontsize=11)
c.set_title("No. of Accidents in Months of Year 2016")
plt.show()
```



```
fig, ax = plt.subplots(figsize = (10,5))
c = sns.countplot(x="Month", data=data_2016, orient = 'v', palette =
"crest")
plt.annotate('Data Not Available',xy=(2,50000), fontsize=11)
c.set_title("No. of Accidents in Months of Year 2016")
plt.show()
```




```
fig, ax = plt.subplots(figsize = (10,5))
c = sns.countplot(x="Month", data=data_2020, orient = 'v', palette =
"crest")
plt.annotate('Covid-19 Pandemic',xy=(2,150000), fontsize=12)
plt.annotate("[",xy=(0,0),xytext=(1.9,150000),arrowprops={'arrowstyle'
:'-|>'}, fontsize=12)
plt.annotate("]",xy=(10,0),xytext=(4.5,150000),arrowprops={'arrowstyle
':'-|>'}, fontsize=12)
c.set_title("No. of Accidents in Month of Year 2020")
plt.show()
```


No. of Accidents in Month of Year 2020
[Covid-19 Pandemic]
count
Month
0
20000
40000
60000
80000
100000
120000
140000
160000
1
2
3
4
5
6
7
8
9
10
11
12