```
import pandas as pd

data_filepath = "J:\\New folder\\PRODIGY_DS_05-Internship-Hariharan-V-
Jupyter-Lab-TASK-05-main\\us-accidents\\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]

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(mi)             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
```

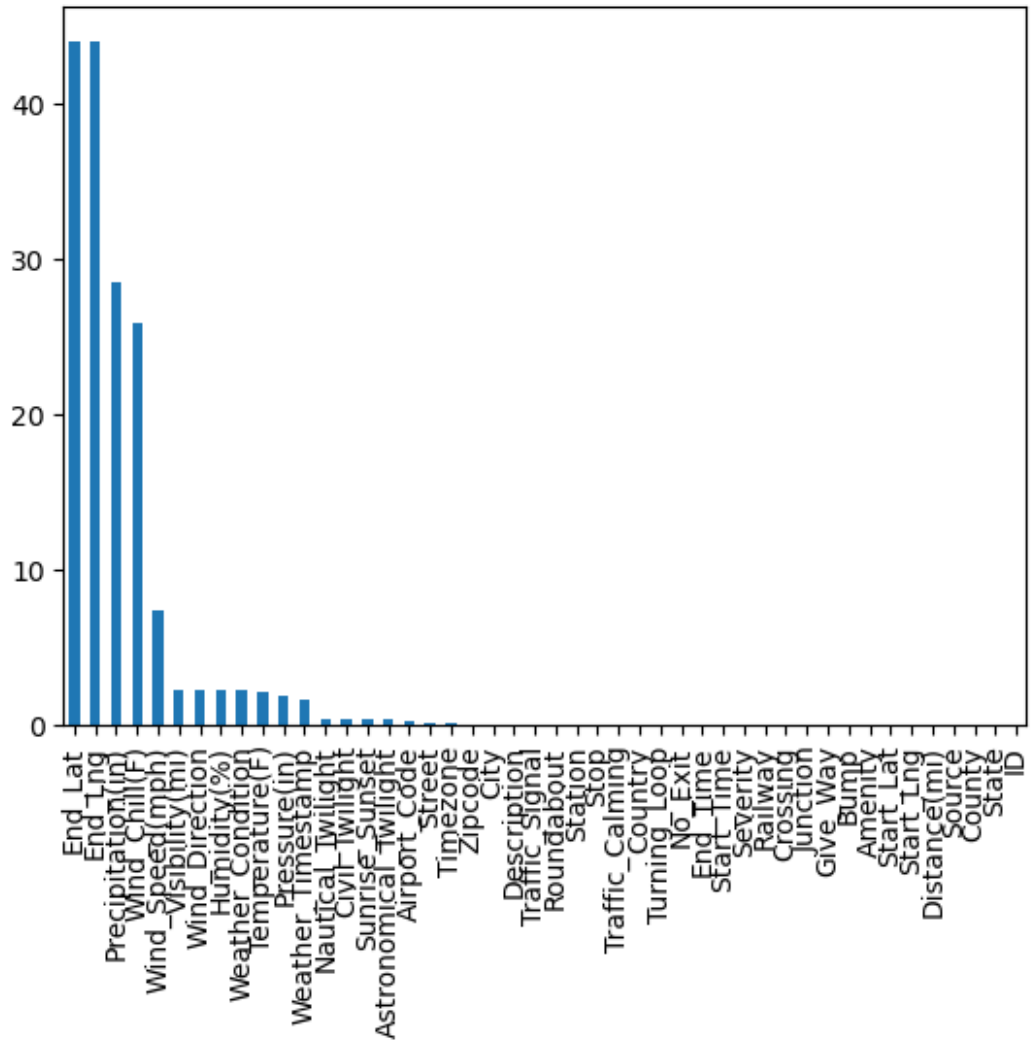
```
Zipcode                 0.024779
City                    0.003274
Description             0.000065
Traffic_Signal          0.000000
Roundabout              0.000000
Station                 0.000000
Stop                    0.000000
Traffic_Calming         0.000000
Country                 0.000000
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
```

```
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
```

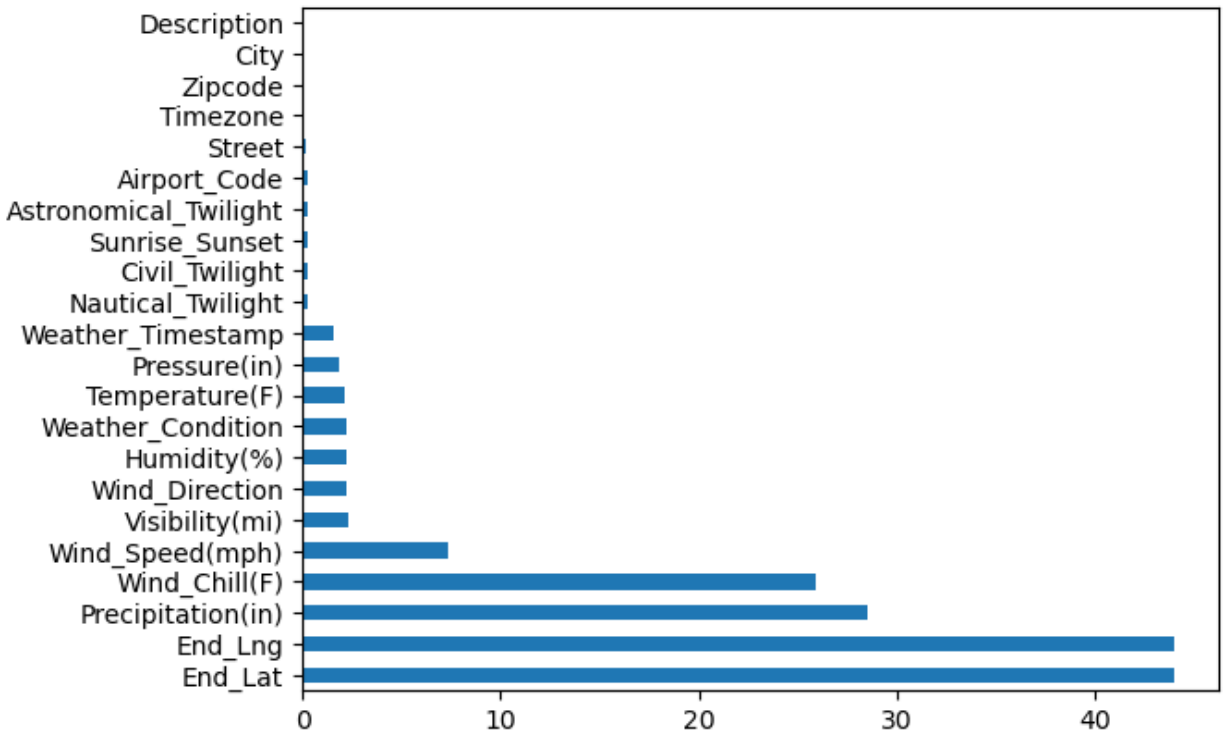
```
missing_data.plot(kind='bar')
```

```
<Axes: >
```




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

```
<Axes: >
```




```
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
```

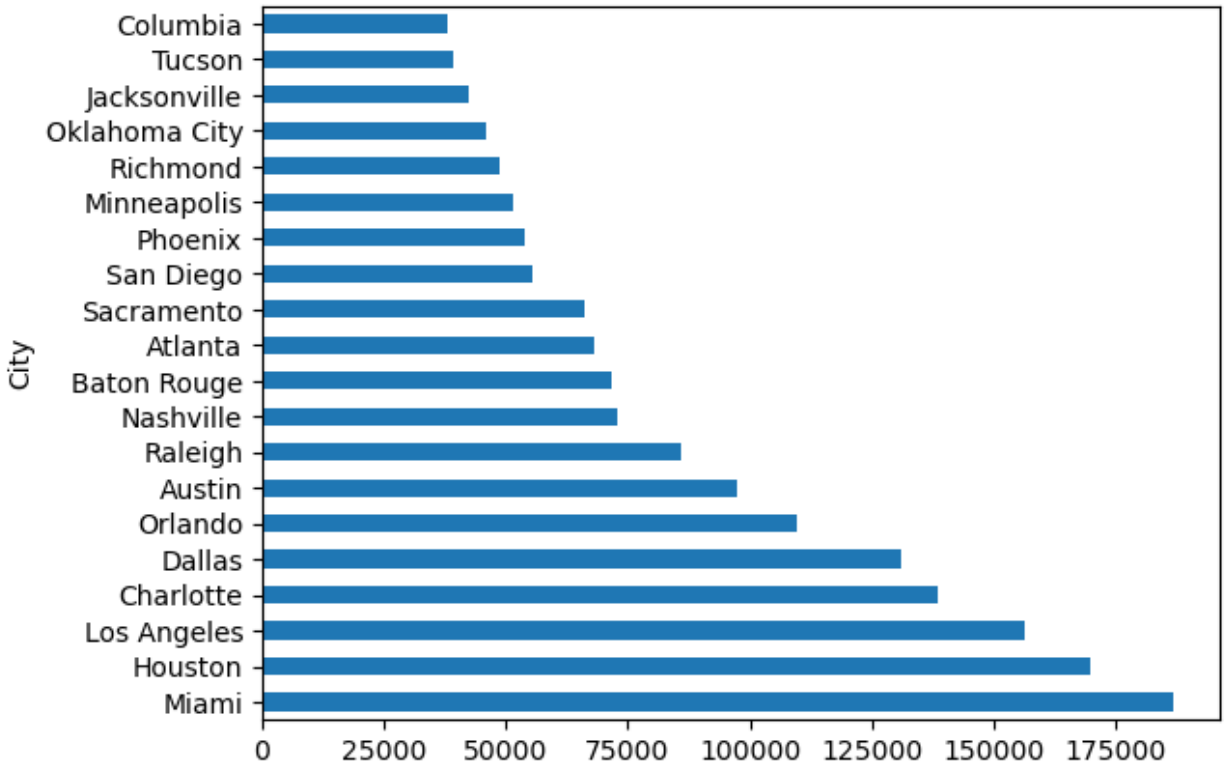
```
True
```

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

```
21699
```

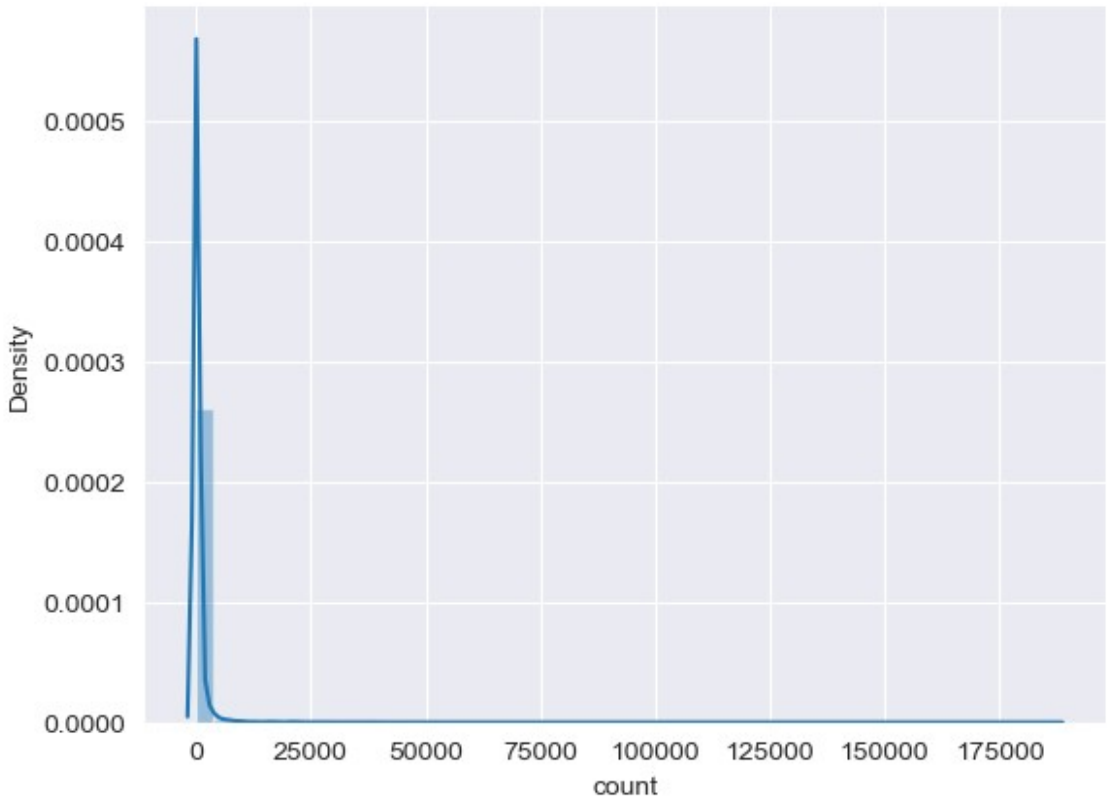
```
cities_by_accident[:20].plot(kind='barh')
```

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




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

sns.distplot(cities_by_accident)
```

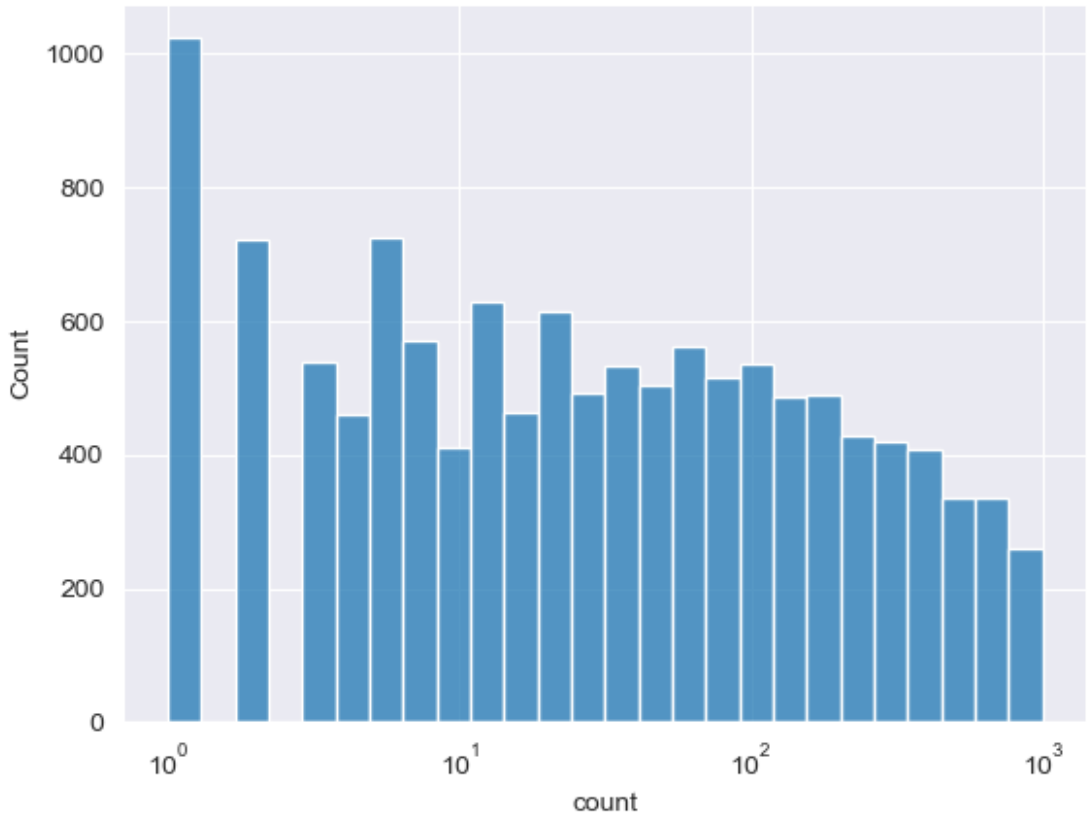
```
C:\Users\Hari\AppData\Local\Temp\ipykernel_16156\3405282844.py:1:
UserWarning:

`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)
J:\Anaconda\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

len(high_accident_cities) / len(cities_by_accident)
sns.histplot(low_accident_cities, log_scale=True)
```

```
J:\Anaconda\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
```

```
df.Start_Time[0]
```

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

```
df.Start_Time = pd.to_datetime(df.Start_Time,format='ISO8601')
```

```
df.Start_Time[0]
```

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

```
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)
```

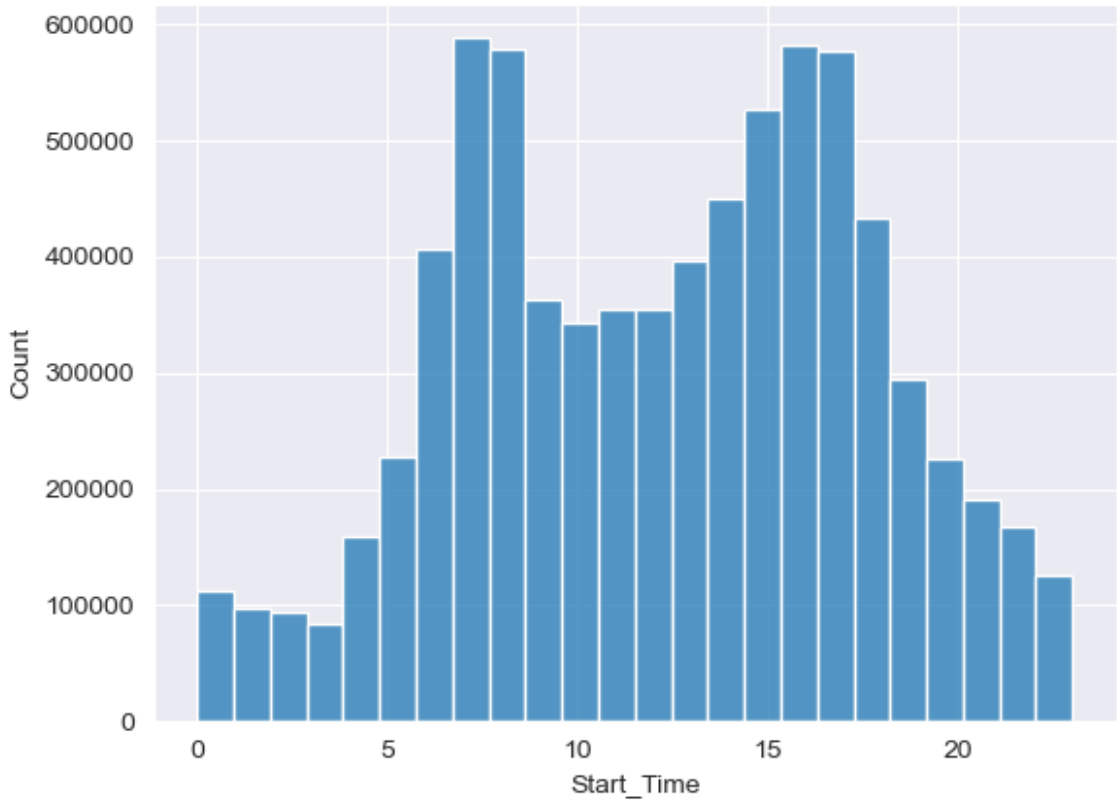
```
df.Start_Time.dt.hour
```

```
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)
```

```
J:\Anaconda\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):
```

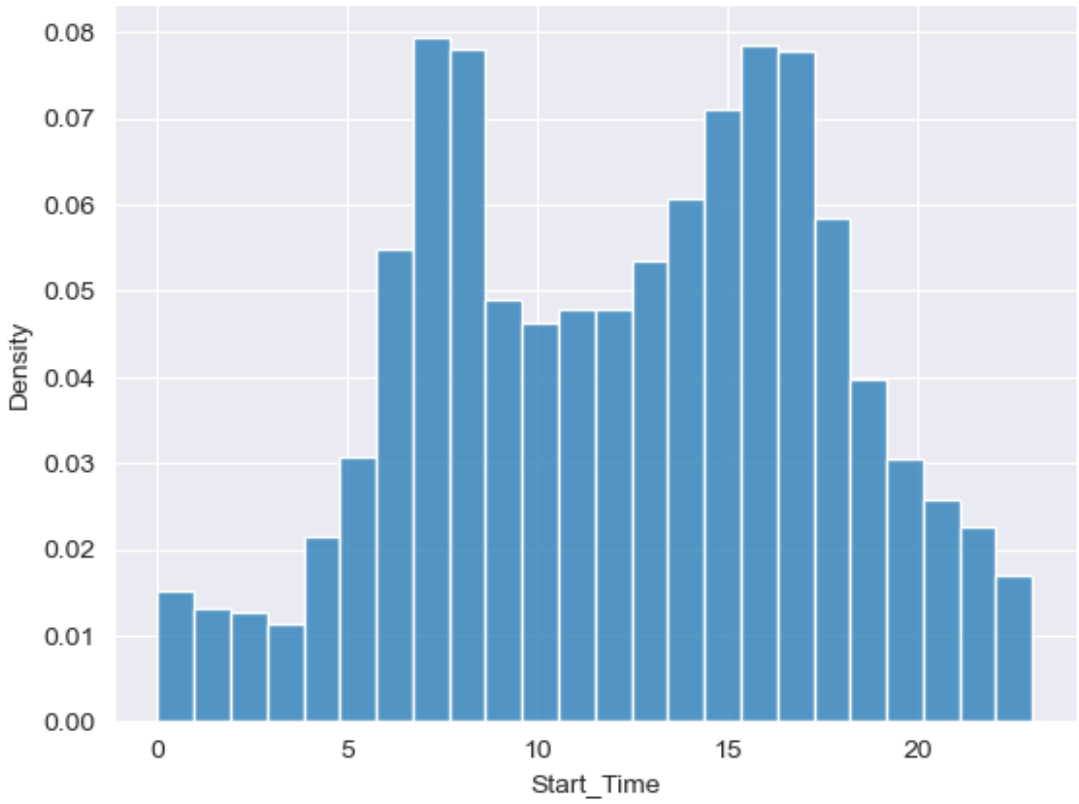
```
<Axes: xlabel='Start_Time', ylabel='Count'>
```




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

```
J:\Anaconda\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):
```

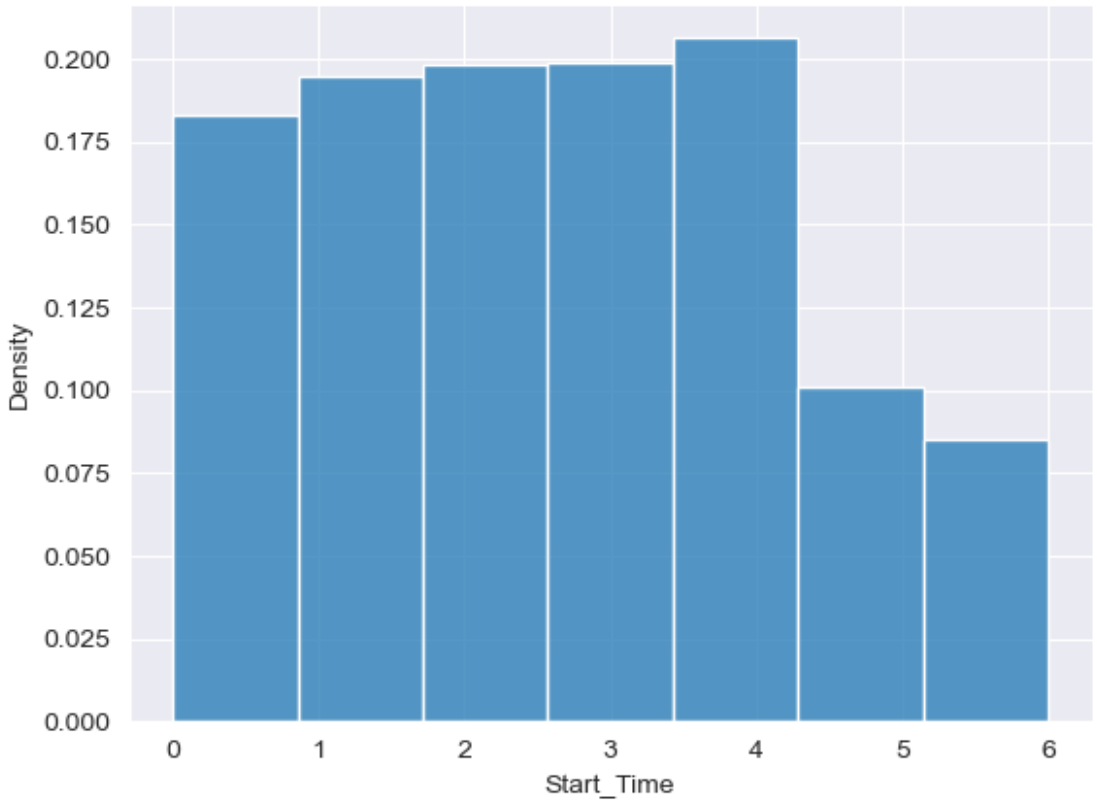
```
<Axes: xlabel='Start_Time', ylabel='Density'>
```




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

```
J:\Anaconda\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):
```

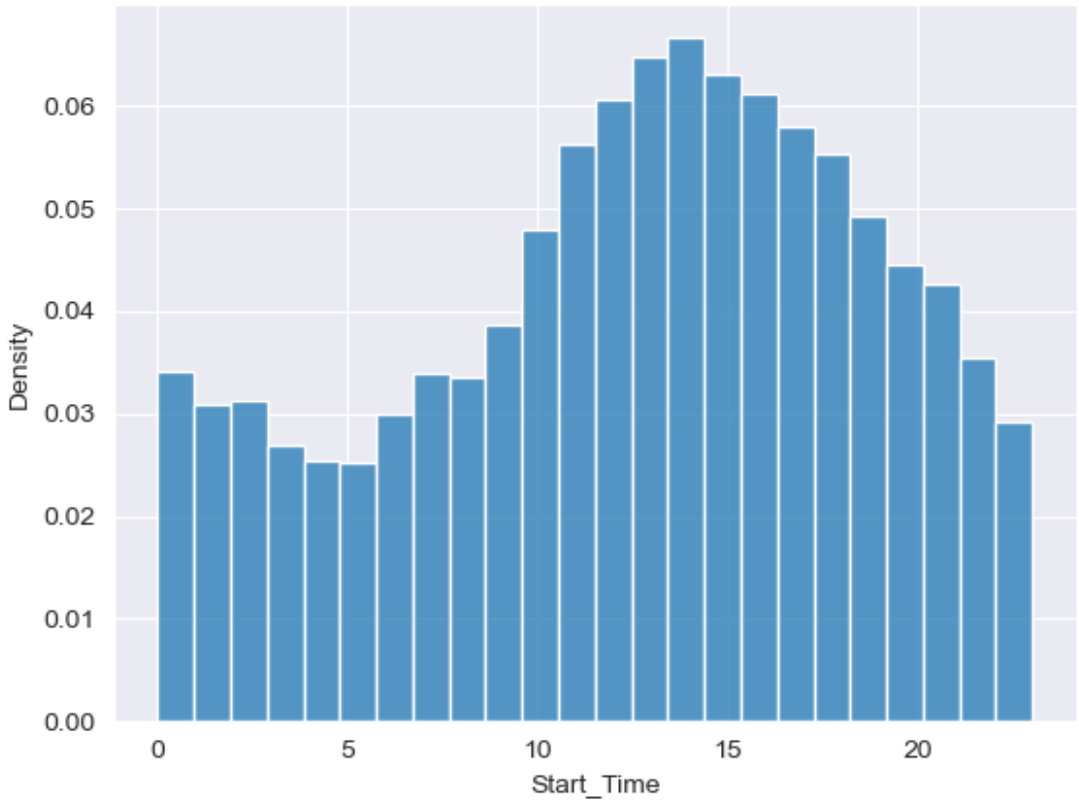
```
<Axes: xlabel='Start_Time', ylabel='Density'>
```




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

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

```
J:\Anaconda\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):
```

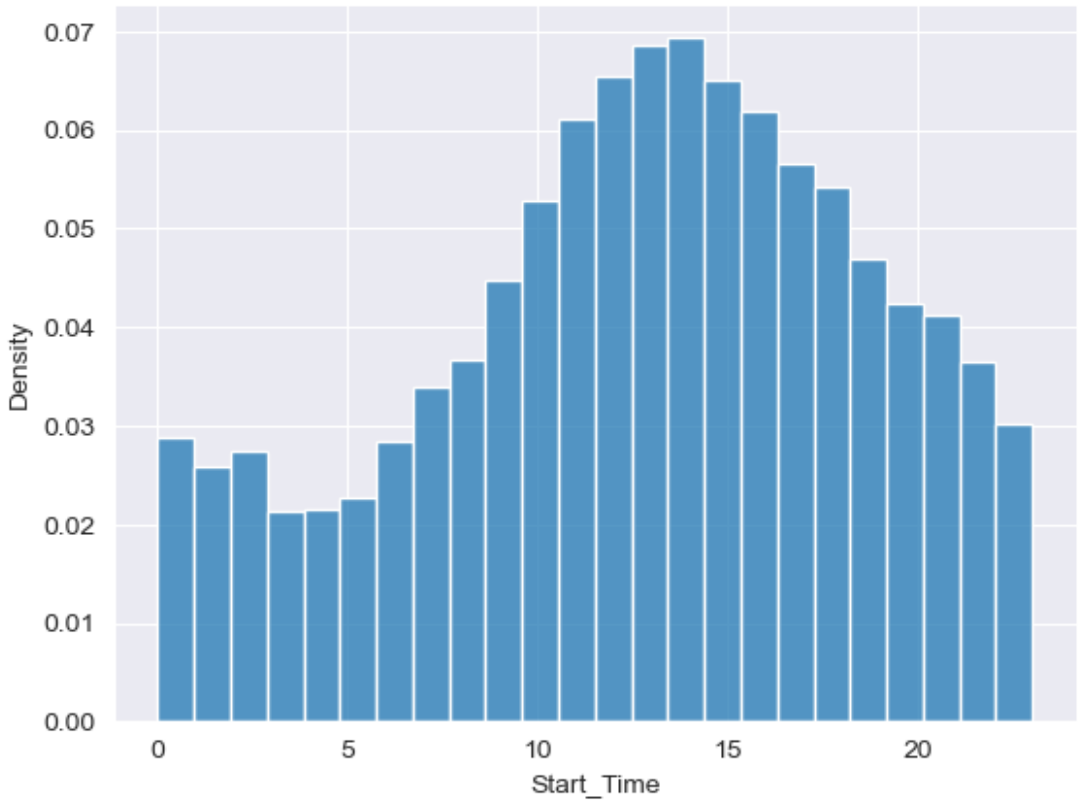
```
<Axes: xlabel='Start_Time', ylabel='Density'>
```




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

```
J:\Anaconda\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):
```

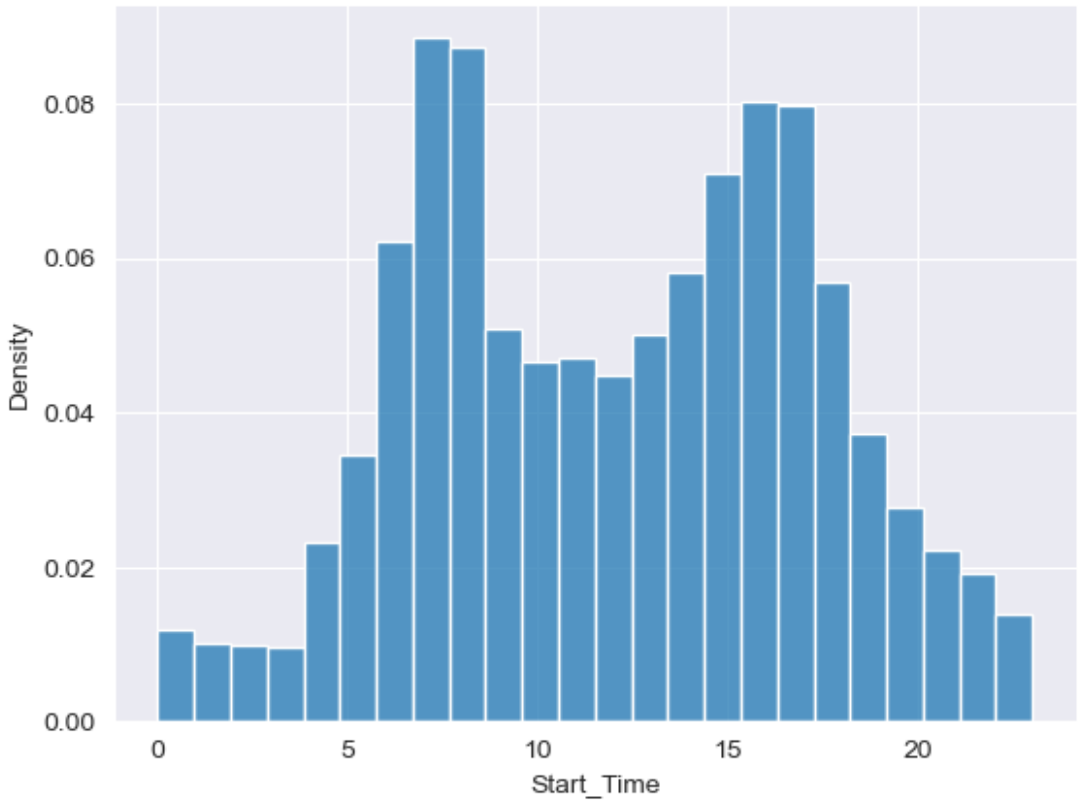
```
<Axes: xlabel='Start_Time', ylabel='Density'>
```




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

```
J:\Anaconda\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):
```

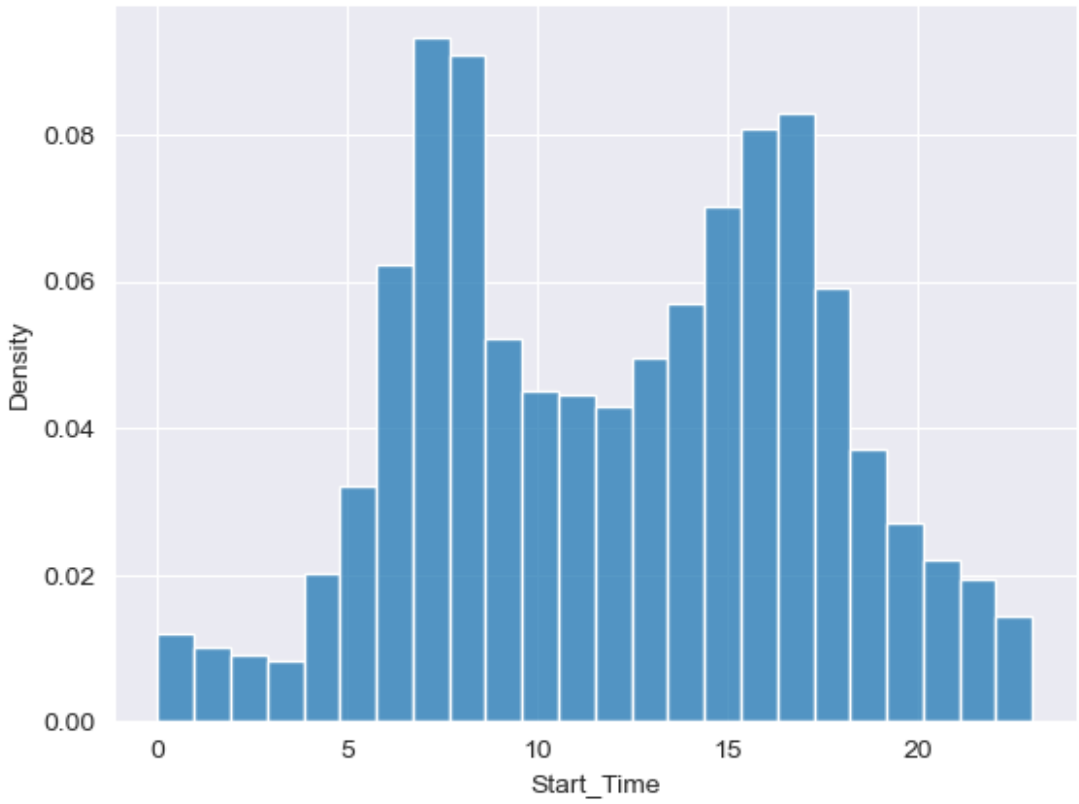
```
<Axes: xlabel='Start_Time', ylabel='Density'>
```




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

```
J:\Anaconda\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):
```

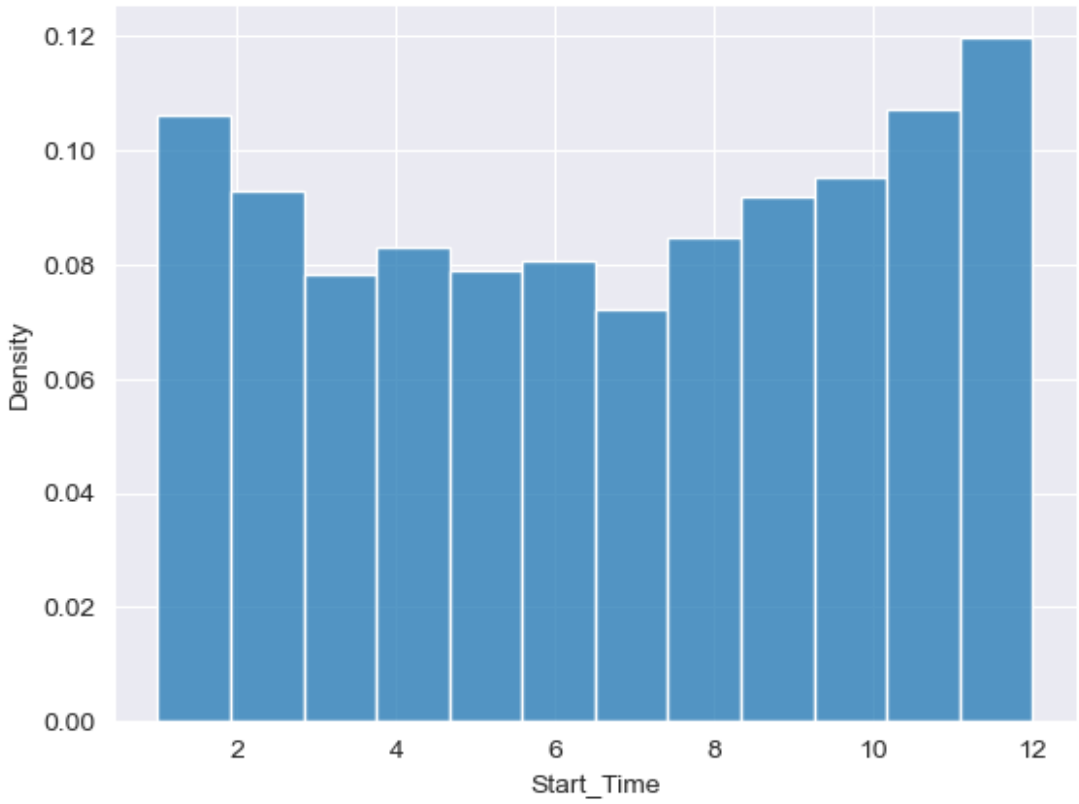
```
<Axes: xlabel='Start_Time', ylabel='Density'>
```




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

```
J:\Anaconda\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):
```

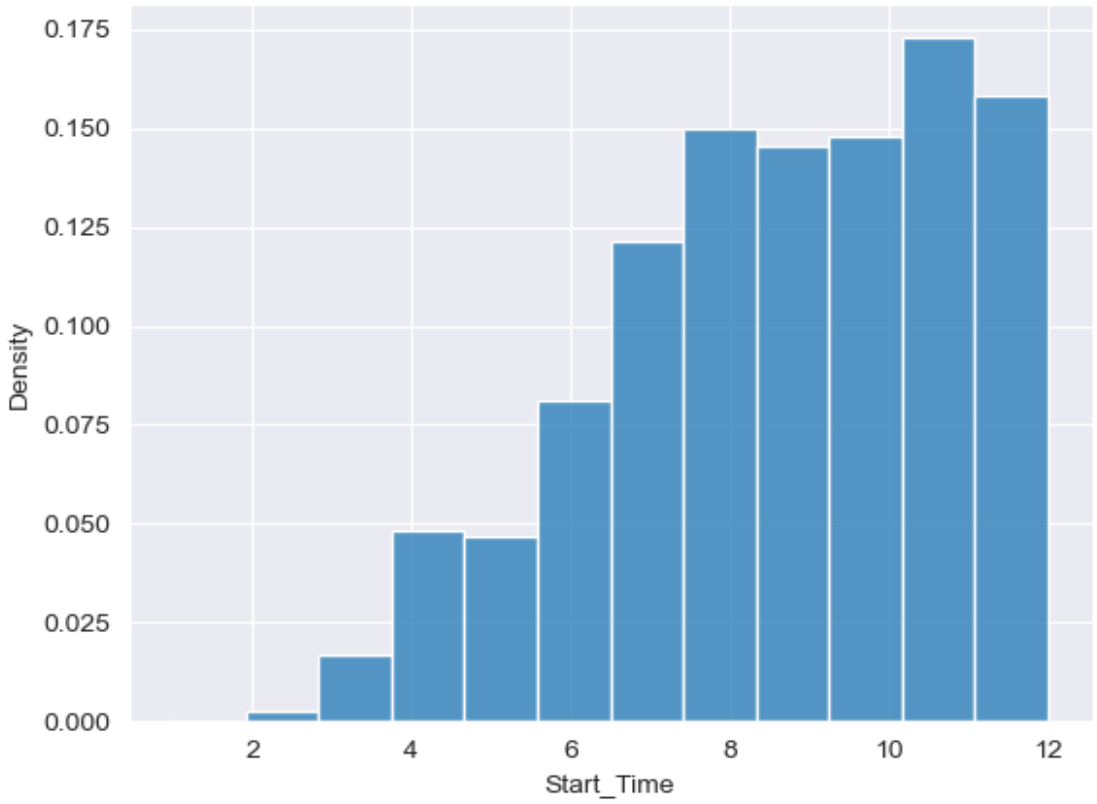
```
<Axes: xlabel='Start_Time', ylabel='Density'>
```




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

```
J:\Anaconda\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):
```

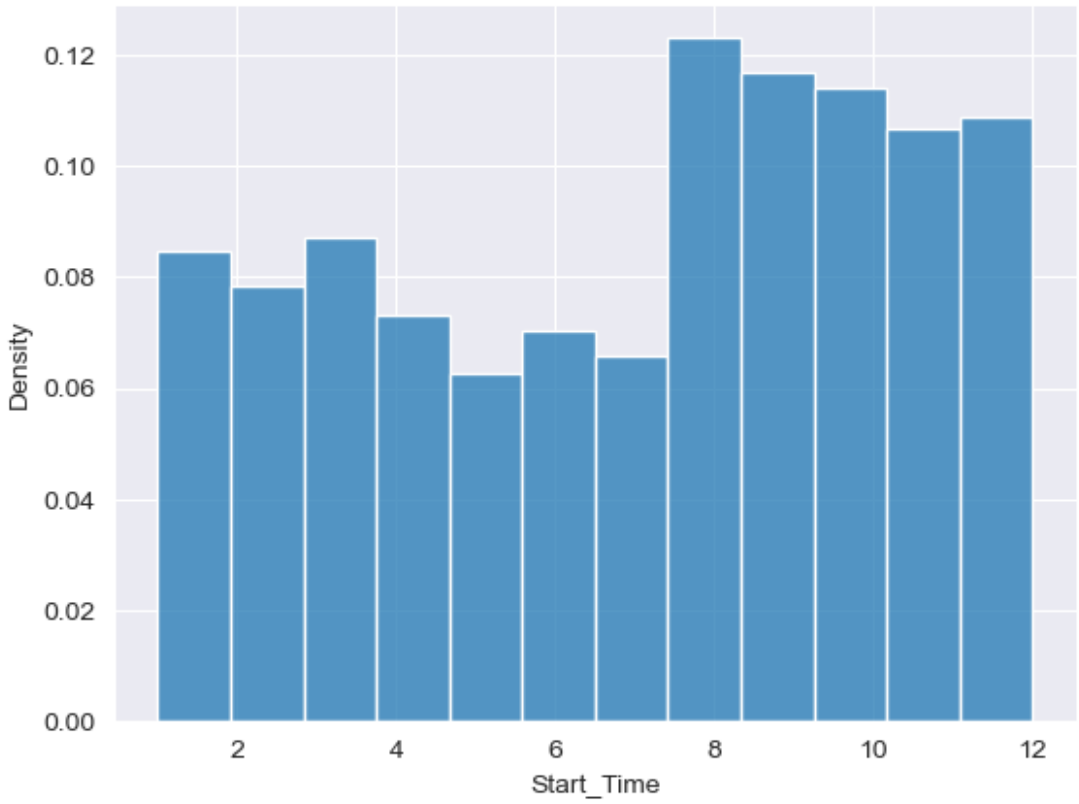
```
<Axes: xlabel='Start_Time', ylabel='Density'>
```




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

```
J:\Anaconda\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):
```

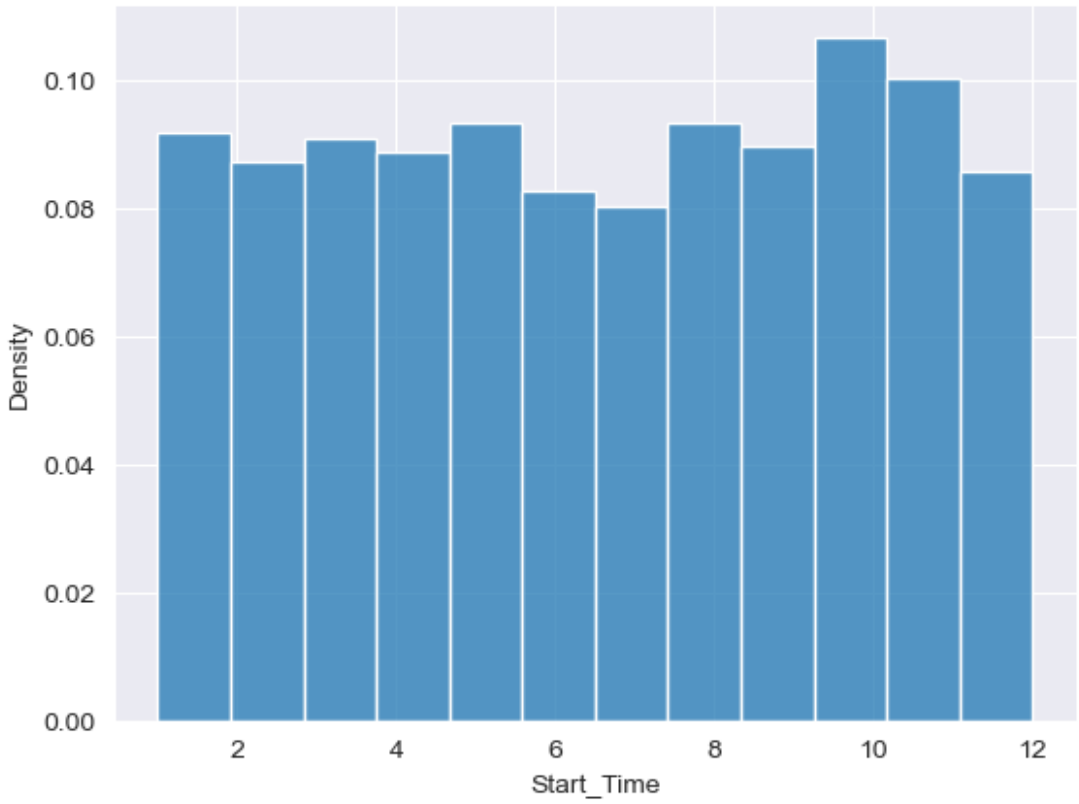
```
<Axes: xlabel='Start_Time', ylabel='Density'>
```




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

```
J:\Anaconda\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):
```

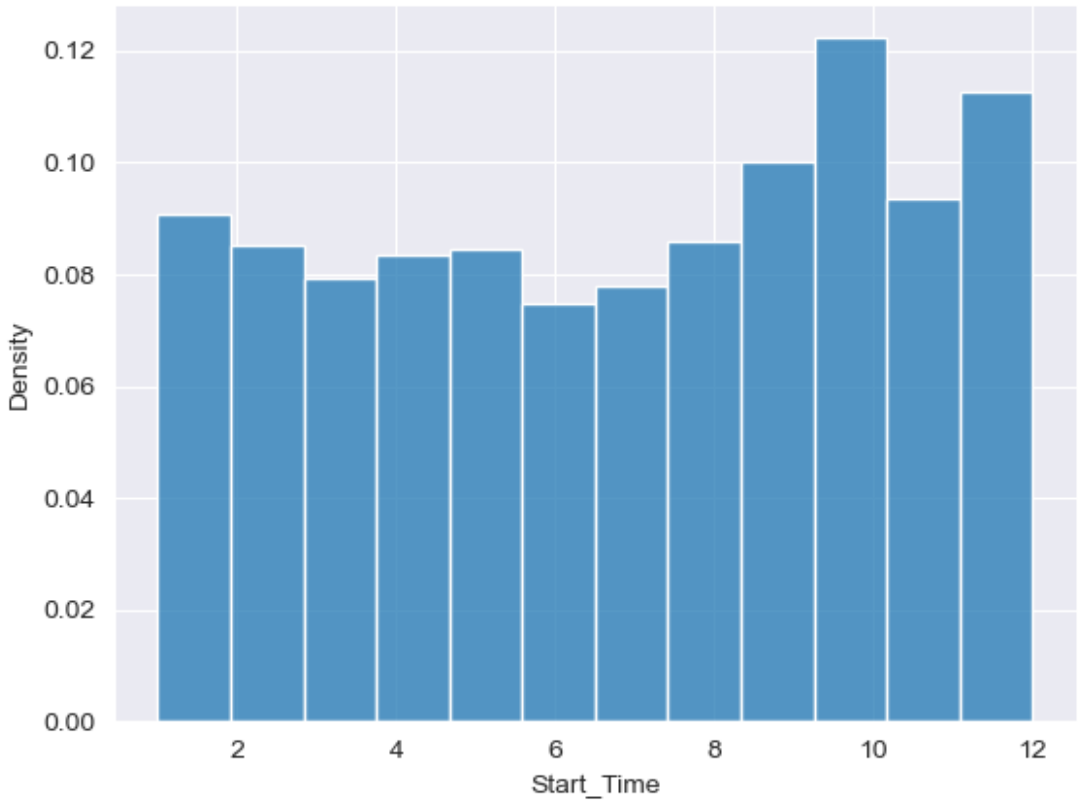
```
<Axes: xlabel='Start_Time', ylabel='Density'>
```




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

```
J:\Anaconda\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):
```

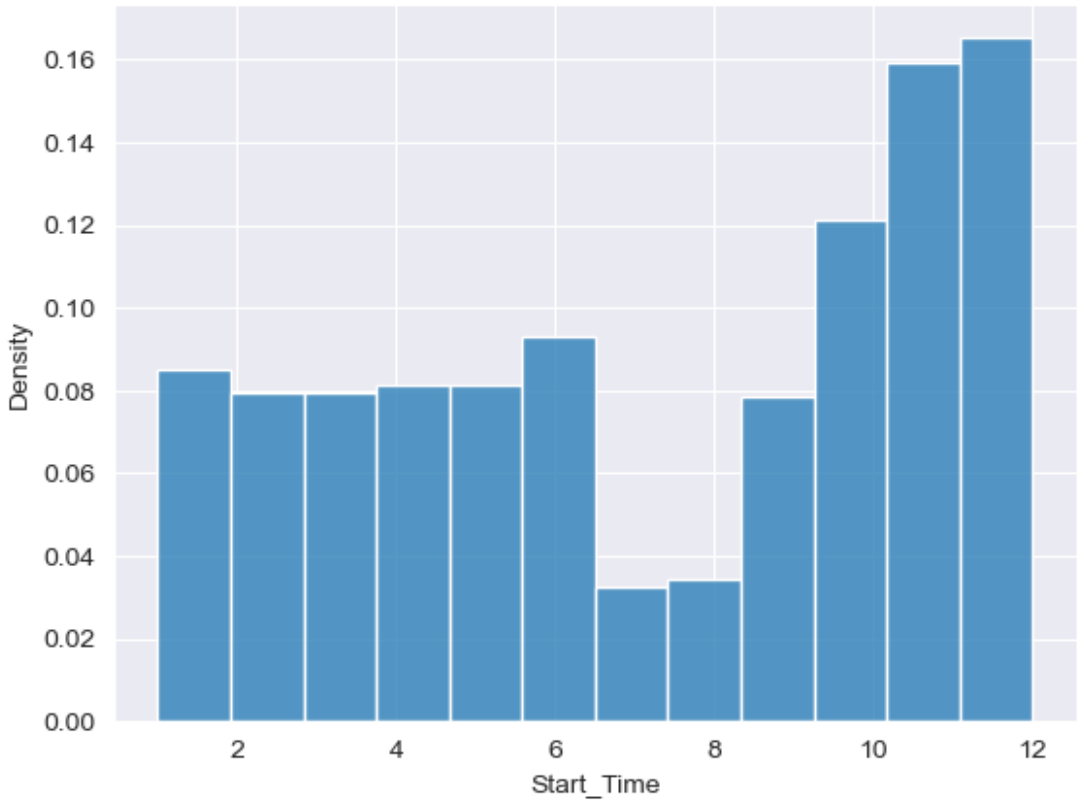
```
<Axes: xlabel='Start_Time', ylabel='Density'>
```




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

```
J:\Anaconda\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
```

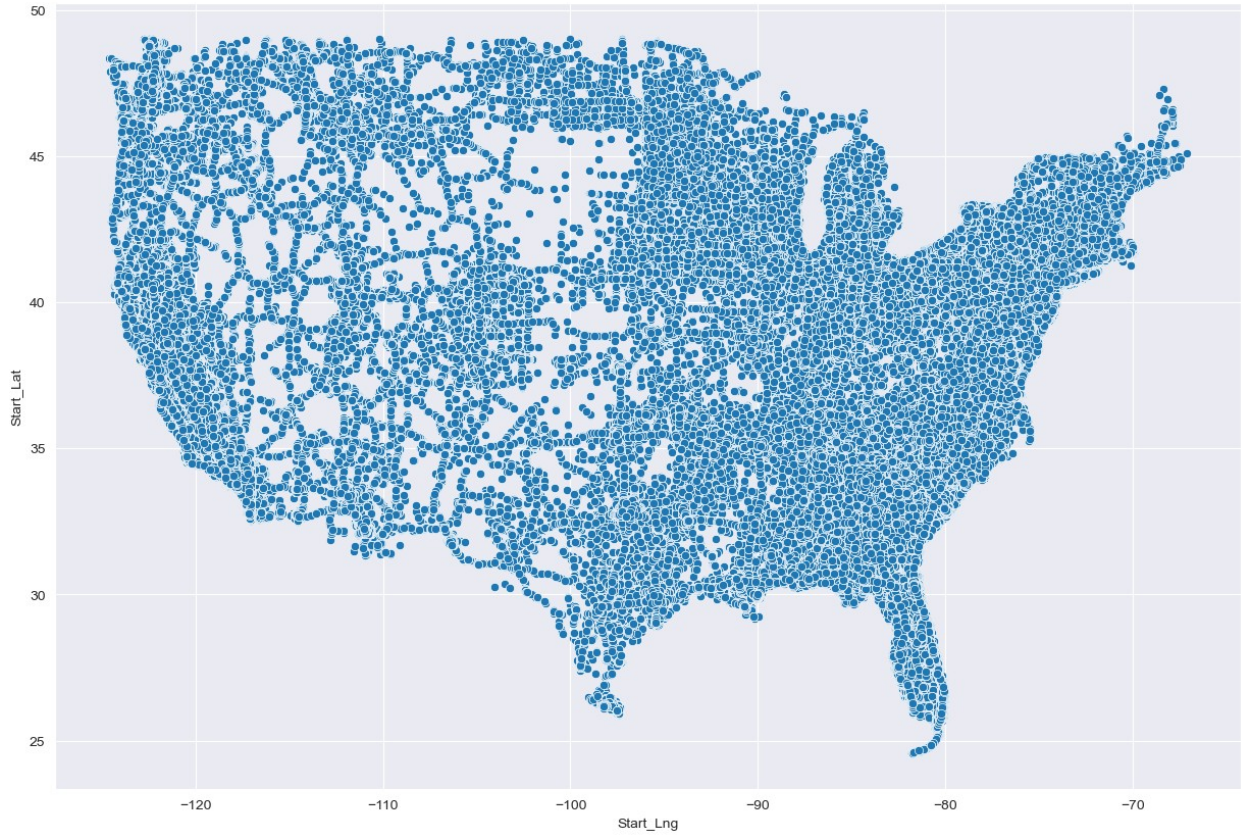
```
df.Start_Lng
```

```
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
```

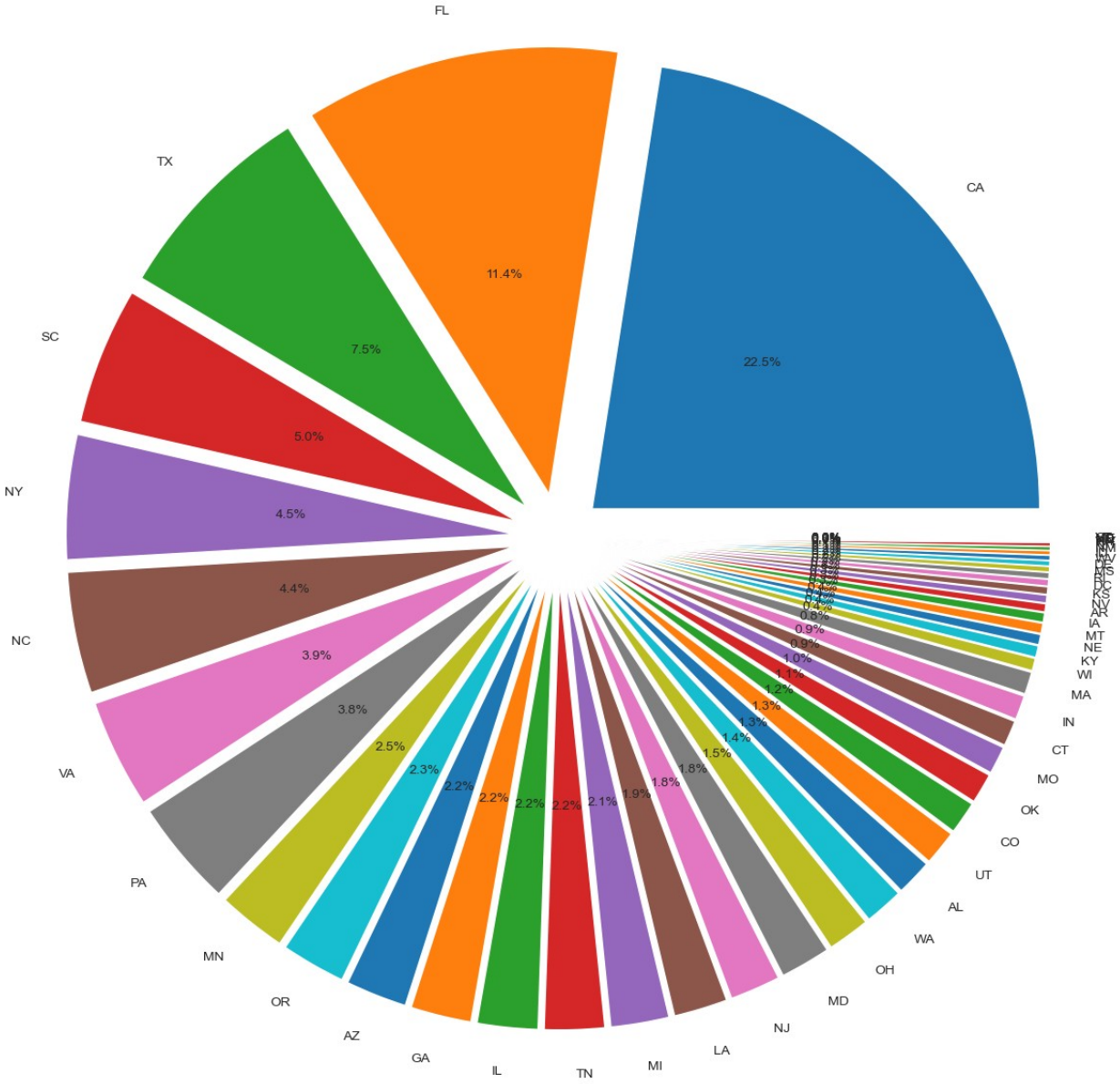
```
 6   Start_Lng              float64
 7   End_Lat                float64
 8   End_Lng                float64
 9   Distance(mi)           float64
 10  Description            object
 11  Street                 object
 12  City                   object
 13  County                 object
 14  State                  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
```

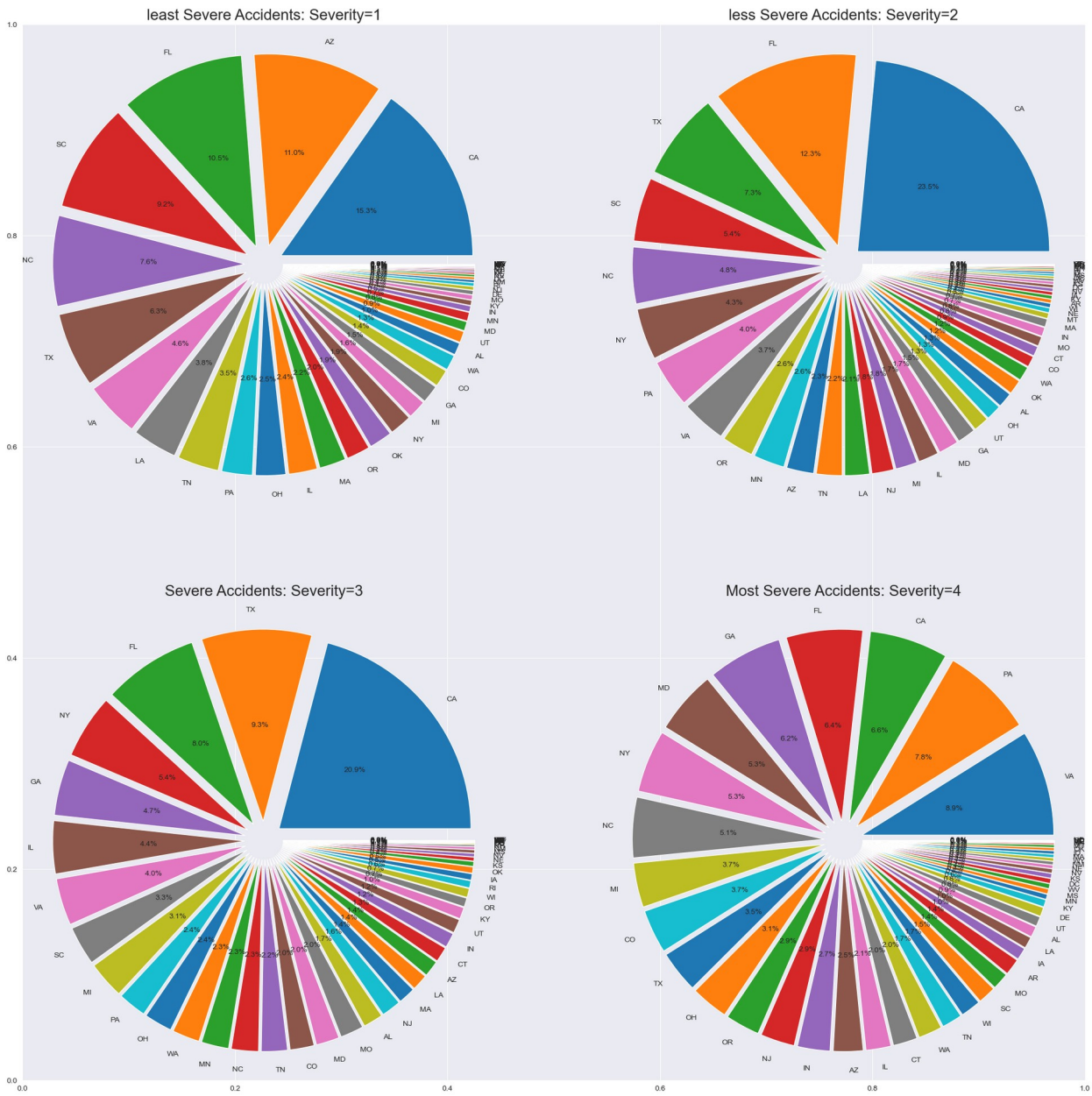
```
pie, ax = plt.subplots(figsize=[15,15])
labels = df.State.value_counts().keys()
plt.pie(x=df.State.value_counts(), autopct="%.1f%%",
explode=[0.1]*len(df.State.value_counts()), labels=labels,
pctdistance=0.5)
plt.show();
```




```
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')
```




```
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)]
```

```
                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.
```

```
4.760109280142808
```

```
(len(df[(df['Visibility(mi)'] <=2) & (df['Severity'] ==4)]) /
len(df) ) * 100.
```

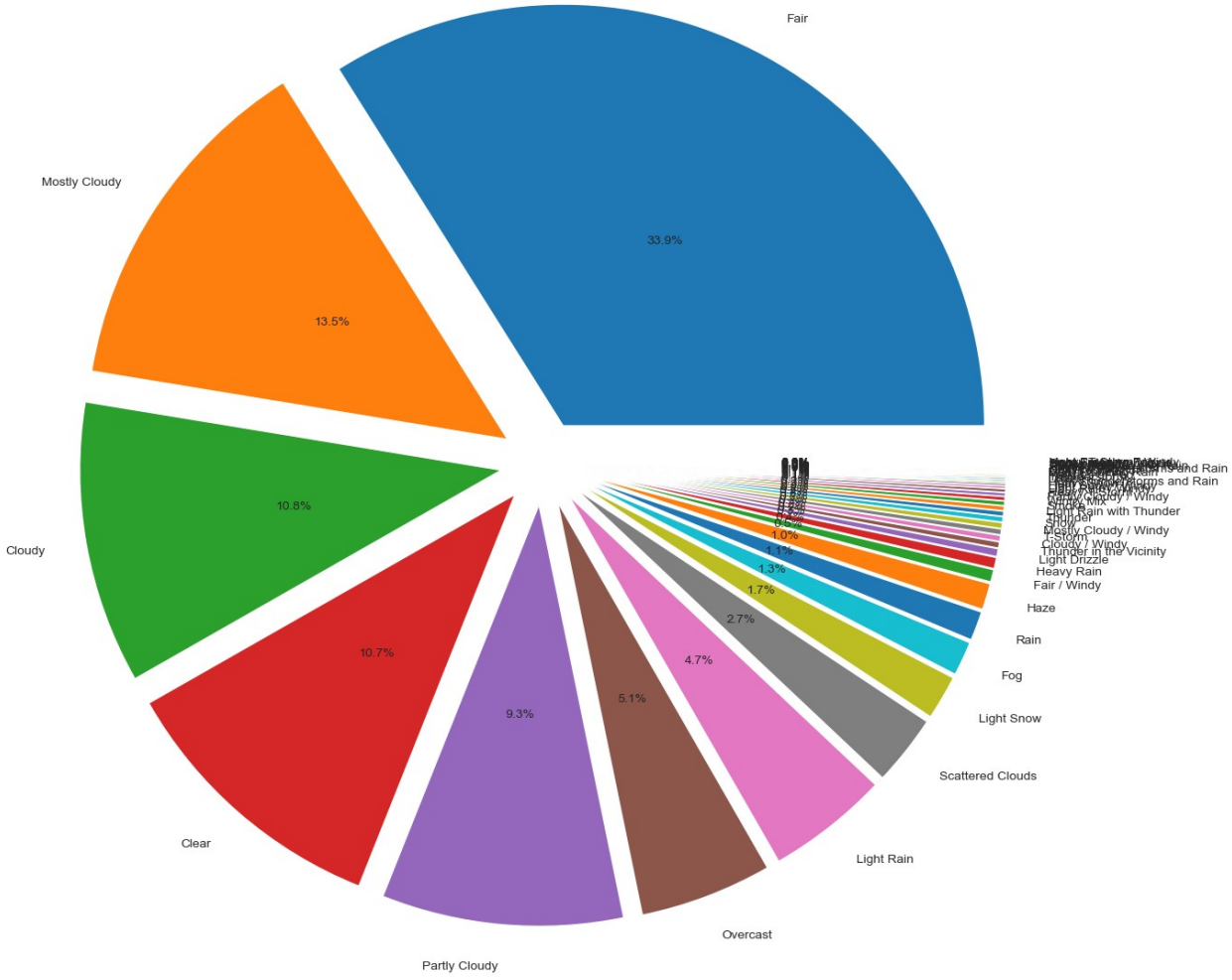
```
0.1428627991792344
```

```
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)
```

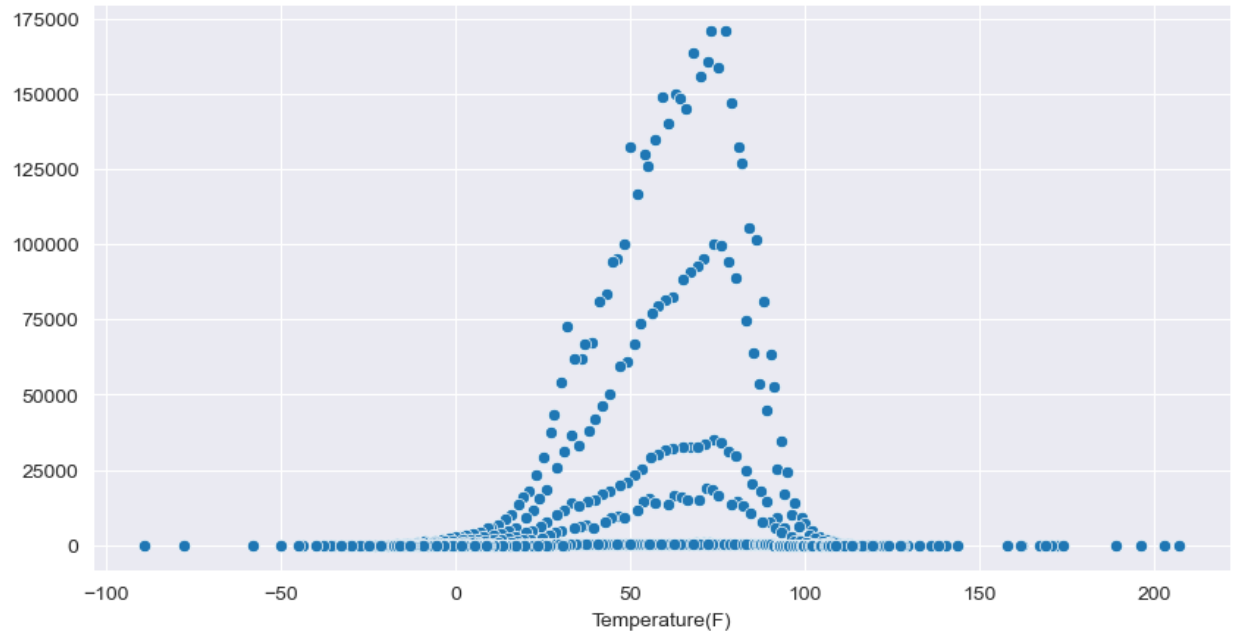
```
temperature.values
```

```
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)
```

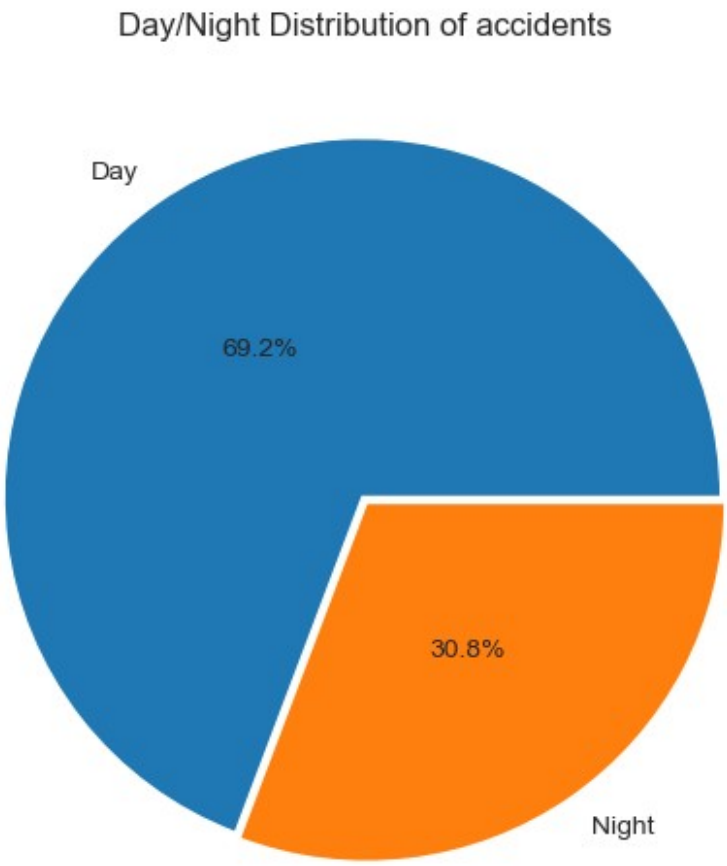
```
import seaborn as sns

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
```

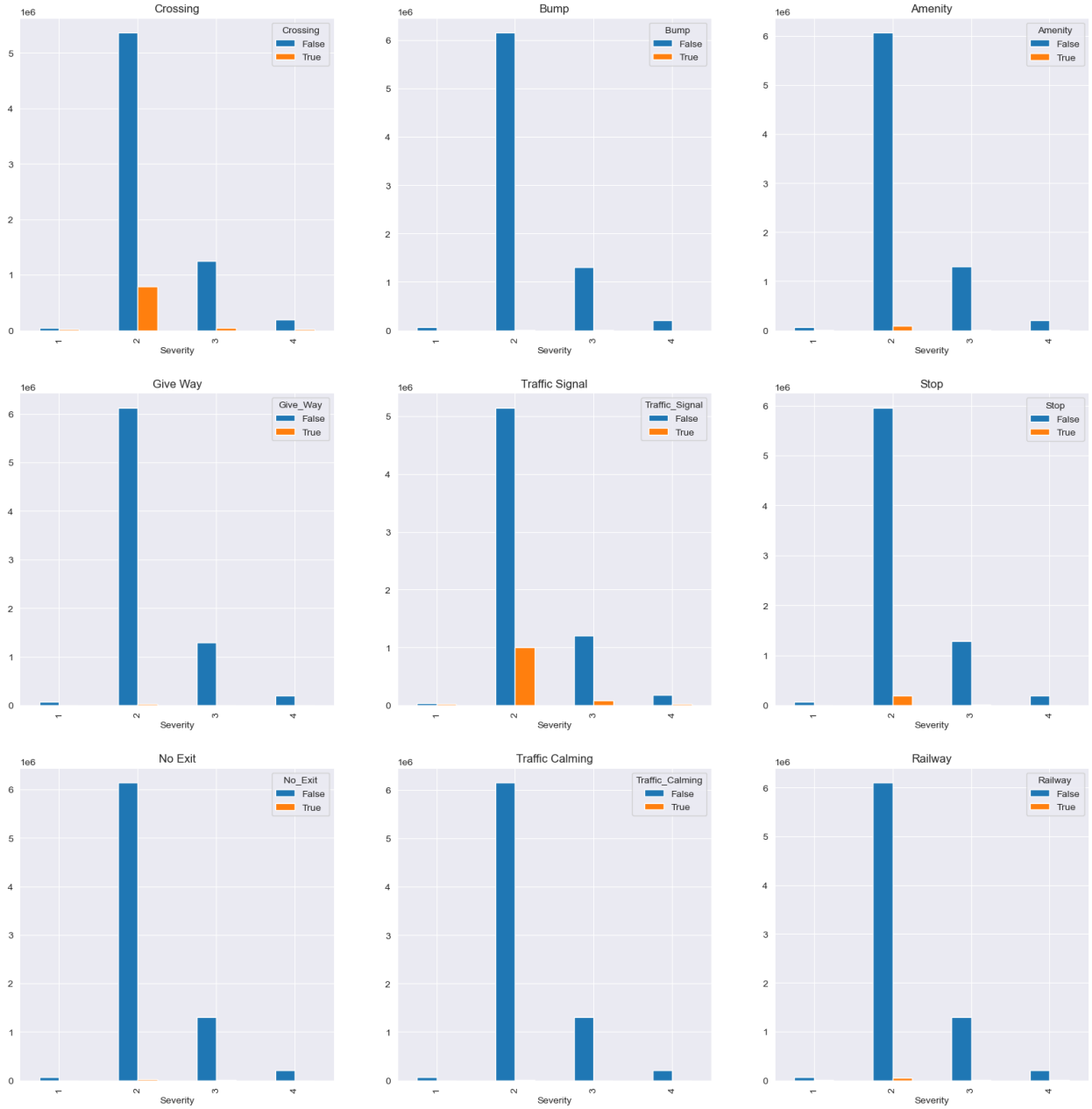
```
df.Turning_Loop.value_counts()
```

```
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'>
```



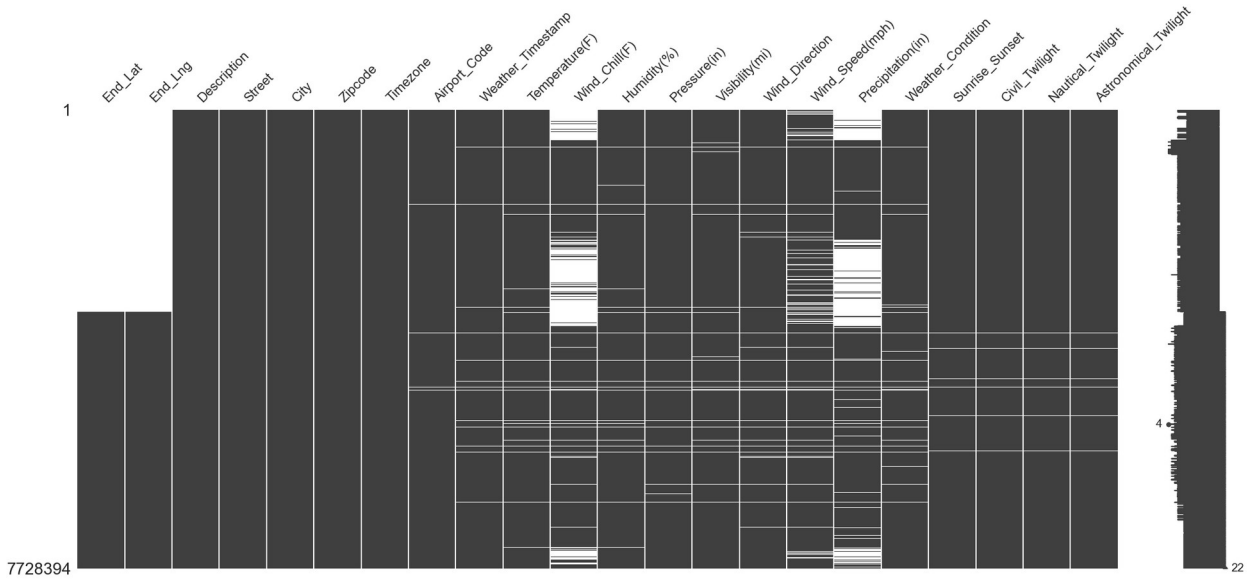
```
import pandas as pd
data = pd.read_csv("J:\\New folder\\PRODIGY_DS_05-Internship-
Hariharan-V-Jupyter-Lab-TASK-05-main\\us-accidents\\
US_Accidents_March23.csv")
null_cols = [i for i in data.columns if data[i].isnull().any()]
print(null_cols)
```

```
['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]);
```

```
Collecting missingno
  Downloading missingno-0.5.2-py3-none-any.whl.metadata (639 bytes)
Requirement already satisfied: numpy in j:\anaconda\lib\site-packages
(from missingno) (1.26.4)
Requirement already satisfied: matplotlib in j:\anaconda\lib\site-
packages (from missingno) (3.8.0)
Requirement already satisfied: scipy in j:\anaconda\lib\site-packages
(from missingno) (1.11.4)
Requirement already satisfied: seaborn in j:\anaconda\lib\site-
packages (from missingno) (0.12.2)
Requirement already satisfied: contourpy>=1.0.1 in j:\anaconda\lib\
site-packages (from matplotlib->missingno) (1.2.0)
Requirement already satisfied: cycler>=0.10 in j:\anaconda\lib\site-
packages (from matplotlib->missingno) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in j:\anaconda\lib\
site-packages (from matplotlib->missingno) (4.25.0)
Requirement already satisfied: kiwisolver>=1.0.1 in j:\anaconda\lib\
site-packages (from matplotlib->missingno) (1.4.4)
Requirement already satisfied: packaging>=20.0 in j:\anaconda\lib\
site-packages (from matplotlib->missingno) (23.1)
Requirement already satisfied: pillow>=6.2.0 in j:\anaconda\lib\site-
packages (from matplotlib->missingno) (10.2.0)
Requirement already satisfied: pyparsing>=2.3.1 in j:\anaconda\lib\
site-packages (from matplotlib->missingno) (3.0.9)
Requirement already satisfied: python-dateutil>=2.7 in j:\anaconda\
lib\site-packages (from matplotlib->missingno) (2.8.2)
Requirement already satisfied: pandas>=0.25 in j:\anaconda\lib\site-
packages (from seaborn->missingno) (2.1.4)
Requirement already satisfied: pytz>=2020.1 in j:\anaconda\lib\site-
packages (from pandas>=0.25->seaborn->missingno) (2023.3.post1)
Requirement already satisfied: tzdata>=2022.1 in j:\anaconda\lib\site-
packages (from pandas>=0.25->seaborn->missingno) (2023.3)
```

```
Requirement already satisfied: six>=1.5 in j:\anaconda\lib\site-
packages (from python-dateutil>=2.7->matplotlib->missingno) (1.16.0)
Downloading missingno-0.5.2-py3-none-any.whl (8.7 kB)
Installing collected packages: missingno
Successfully installed missingno-0.5.2
```




```
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)
```

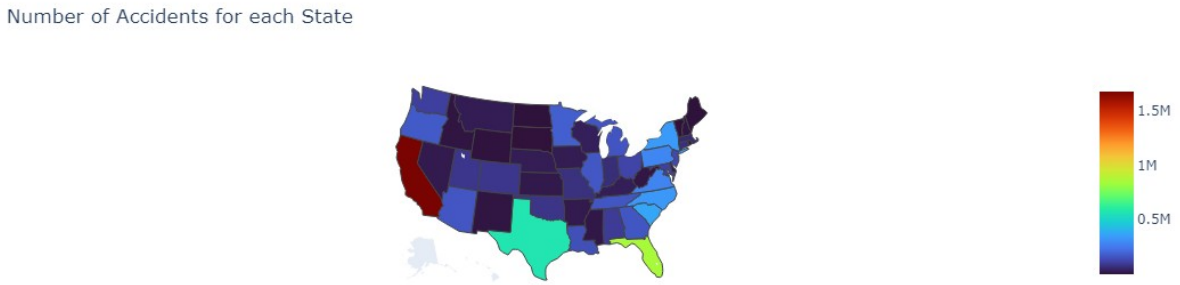
```
final_data.isnull().sum()
```

```
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()
```

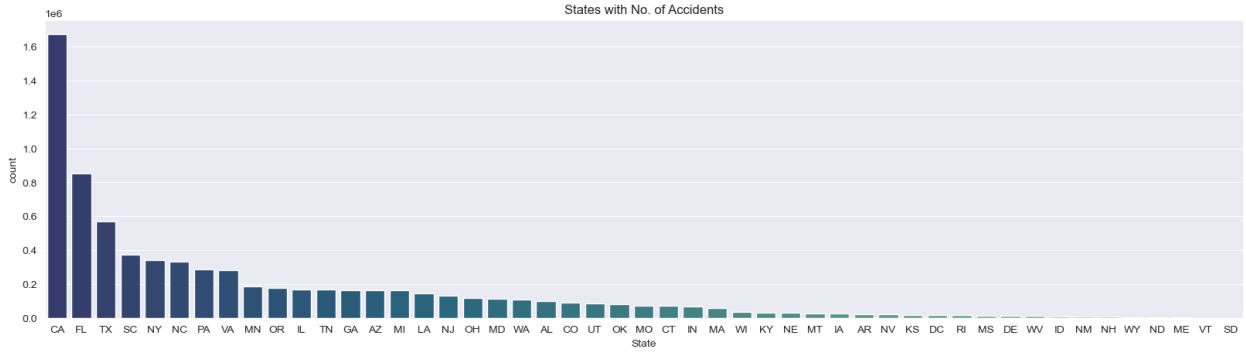
```
Requirement already satisfied: plotly in j:\anaconda\lib\site-packages
(5.9.0)
Requirement already satisfied: tenacity>=6.2.0 in j:\anaconda\lib\
site-packages (from plotly) (8.2.2)
```



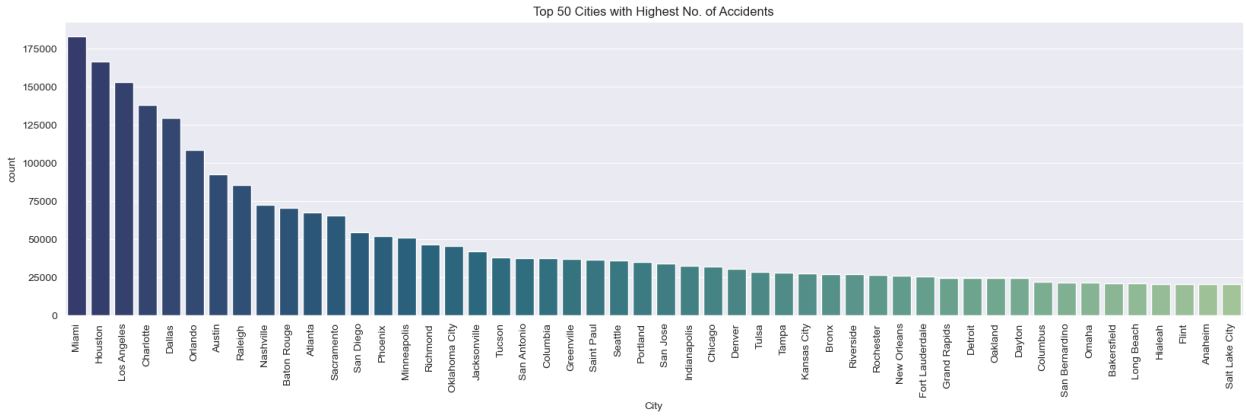

```
print("State Code: ", final_data.State.unique())
print("Total No. of State in Dataset: ",
len(final_data.State.unique()))
```

```
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()
```

```
---------------------------------------------------------------------------
-----
MemoryError                                Traceback (most recent call
last)
Cell In[107], line 11
      9 data_2019 = final_data[final_data.Start_Time.dt.year == 2019]
```

```
     10 data_2020 = final_data[final_data.Start_Time.dt.year == 2020]
---> 11 data_2017_2019 = final_data[(final_data["Year"] >= 2017) &
(final_data["Year"] <= 2019)]
     13 fig, ax = plt.subplots(figsize = (10,5))
     14 c = sns.countplot(x="Year", data=final_data, orient = 'v',
palette = "crest")

File J:\Anaconda\Lib\site-packages\pandas\core\frame.py:3884, in
DataFrame.__getitem__(self, key)
   3882 # Do we have a (boolean) 1d indexer?
   3883 if com.is_bool_indexer(key):
-> 3884     return self._getitem_bool_array(key)
   3886 # We are left with two options: a single key, and a collection
of keys,
   3887 # We interpret tuples as collections only for non-MultiIndex
   3888 is_single_key = isinstance(key, tuple) or not
is_list_like(key)

File J:\Anaconda\Lib\site-packages\pandas\core\frame.py:3946, in
DataFrame._getitem_bool_array(self, key)
   3943     return self.copy(deep=None)
   3945 indexer = key.nonzero()[0]
-> 3946 return self._take_with_is_copy(indexer, axis=0)

File J:\Anaconda\Lib\site-packages\pandas\core\generic.py:4088, in
NDFrame._take_with_is_copy(self, indices, axis)
   4077 @final
   4078 def _take_with_is_copy(self, indices, axis: Axis = 0) -> Self:
   4079     """
   4080     Internal version of the `take` method that sets the
`_is_copy`
   4081     attribute to keep track of the parent dataframe (using in
indexing
   (...)
   4086     See the docstring of `take` for full explanation of the
parameters.
   4087     """
-> 4088     result = self.take(indices=indices, axis=axis)
   4089     # Maybe set copy if we didn't actually change the index.
   4090     if self.ndim == 2 and not
result._get_axis(axis).equals(self._get_axis(axis)):

File J:\Anaconda\Lib\site-packages\pandas\core\generic.py:4068, in
NDFrame.take(self, indices, axis, **kwargs)
   4063     # We can get here with a slice via DataFrame.__getitem__
   4064     indices = np.arange(
   4065         indices.start, indices.stop, indices.step,
dtype=np.intp
   4066     )
-> 4068 new_data = self._mgr.take(
```

```
   4069     indices,
   4070     axis=self._get_block_manager_axis(axis),
   4071     verify=True,
   4072 )
   4073 return self._constructor_from_mgr(new_data,
axes=new_data.axes).__finalize__(
   4074     self, method="take"
   4075 )

File J:\Anaconda\Lib\site-packages\pandas\core\internals\
managers.py:877, in BaseBlockManager.take(self, indexer, axis, verify)
    874 indexer = maybe_convert_indices(indexer, n, verify=verify)
    876 new_labels = self.axes[axis].take(indexer)
--> 877 return self.reindex_indexer(
    878     new_axis=new_labels,
    879     indexer=indexer,
    880     axis=axis,
    881     allow_dups=True,
    882     copy=None,
    883 )

File J:\Anaconda\Lib\site-packages\pandas\core\internals\
managers.py:670, in BaseBlockManager.reindex_indexer(self, new_axis,
indexer, axis, fill_value, allow_dups, copy, only_slice, use_na_proxy)
    663     new_blocks = self._slice_take_blocks_ax0(
    664         indexer,
    665         fill_value=fill_value,
    666         only_slice=only_slice,
    667         use_na_proxy=use_na_proxy,
    668     )
    669 else:
--> 670     new_blocks = [
    671         blk.take_nd(
    672             indexer,
    673             axis=1,
    674             fill_value=(
    675                 fill_value if fill_value is not None else
blk.fill_value
    676             ),
    677         )
    678         for blk in self.blocks
    679     ]
    681 new_axes = list(self.axes)
    682 new_axes[axis] = new_axis

File J:\Anaconda\Lib\site-packages\pandas\core\internals\
managers.py:671, in <listcomp>(.0)
    663     new_blocks = self._slice_take_blocks_ax0(
    664         indexer,
    665         fill_value=fill_value,
```

```
    666         only_slice=only_slice,
    667         use_na_proxy=use_na_proxy,
    668     )
    669 else:
    670     new_blocks = [
--> 671         blk.take_nd(
    672             indexer,
    673             axis=1,
    674             fill_value=(
    675                 fill_value if fill_value is not None else
blk.fill_value
    676             ),
    677         )
    678         for blk in self.blocks
    679     ]
    681 new_axes = list(self.axes)
    682 new_axes[axis] = new_axis

File J:\Anaconda\Lib\site-packages\pandas\core\internals\
blocks.py:1061, in Block.take_nd(self, indexer, axis, new_mgr_locs,
fill_value)
   1058     allow_fill = True
   1060 # Note: algos.take_nd has upcast logic similar to
coerce_to_target_dtype
-> 1061 new_values = algos.take_nd(
   1062     values, indexer, axis=axis, allow_fill=allow_fill,
fill_value=fill_value
   1063 )
   1065 # Called from three places in managers, all of which satisfy
   1066 #  these assertions
   1067 if isinstance(self, ExtensionBlock):
   1068     # NB: in this case, the 'axis' kwarg will be ignored in
the
   1069     #  algos.take_nd call above.

File J:\Anaconda\Lib\site-packages\pandas\core\array_algos\
take.py:118, in take_nd(arr, indexer, axis, fill_value, allow_fill)
    115     return arr.take(indexer, fill_value=fill_value,
allow_fill=allow_fill)
    117 arr = np.asarray(arr)
--> 118 return _take_nd_ndarray(arr, indexer, axis, fill_value,
allow_fill)

File J:\Anaconda\Lib\site-packages\pandas\core\array_algos\
take.py:158, in _take_nd_ndarray(arr, indexer, axis, fill_value,
allow_fill)
    156     out = np.empty(out_shape, dtype=dtype, order="F")
    157 else:
--> 158     out = np.empty(out_shape, dtype=dtype)
    160 func = _get_take_nd_function(
```

```
    161     arr.ndim, arr.dtype, out.dtype, axis=axis,
mask_info=mask_info
    162 )
    163 func(arr, indexer, out, fill_value)

MemoryError: Unable to allocate 190. MiB for an array with shape (10,
2485839) and data type float64
```

```
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()
```