lvh1vbvx0

March 12, 2023

1 us-accidents-data-analysis

#US Accidents Exploratory Data Analysis

TODO- Talk about EDA TODO - Talk about the dataset(Sources,what it contains, how it will be useful) * Kaggle * Information about accident * Can be useful to prevent accident * This data don't have New york city data

1.1 Data Prepration And Cleaning¶

- Load the files using Pandas.
- Looks at some information about the data and columns.
- Fix any missing and incorrect value.

[1]:
```
import pandas as pd
```

[2]:
```
df = pd.read_csv("US_Accidents_Dec21_updated.csv")
```

[3]:
```
df
```

[3]:
```
                ID  Severity           Start_Time             End_Time  \
0              A-1         3  2016-02-08 00:37:08  2016-02-08 06:37:08
1              A-2         2  2016-02-08 05:56:20  2016-02-08 11:56:20
2              A-3         2  2016-02-08 06:15:39  2016-02-08 12:15:39
3              A-4         2  2016-02-08 06:51:45  2016-02-08 12:51:45
4              A-5         3  2016-02-08 07:53:43  2016-02-08 13:53:43
…              …        …                  …                  …
2845337  A-2845338         2  2019-08-23 18:03:25  2019-08-23 18:32:01
2845338  A-2845339         2  2019-08-23 19:11:30  2019-08-23 19:38:23
2845339  A-2845340         2  2019-08-23 19:00:21  2019-08-23 19:28:49
2845340  A-2845341         2  2019-08-23 19:00:21  2019-08-23 19:29:42
2845341  A-2845342         2  2019-08-23 18:52:06  2019-08-23 19:21:31

         Start_Lat  Start_Lng    End_Lat    End_Lng  Distance(mi)  \
0        40.108910 -83.092860  40.112060 -83.031870         3.230
1        39.865420 -84.062800  39.865010 -84.048730         0.747
2        39.102660 -84.524680  39.102090 -84.523960         0.055
3        41.062130 -81.537840  41.062170 -81.535470         0.123
4        39.172393 -84.492792  39.170476 -84.501798         0.500
```

```
…              …          …         …         …           …
2845337  34.002480 -117.379360  33.998880 -117.370940         0.543
2845338  32.766960 -117.148060  32.765550 -117.153630         0.338
2845339  33.775450 -117.847790  33.777400 -117.857270         0.561
2845340  33.992460 -118.403020  33.983110 -118.395650         0.772
2845341  34.133930 -117.230920  34.137360 -117.239340         0.537

                                              Description  …  Roundabout  \
0        Between Sawmill Rd/Exit 20 and OH-315/Olentang…  …       False
1                       At OH-4/OH-235/Exit 41 - Accident.  …       False
2                         At I-71/US-50/Exit 1 - Accident.  …       False
3                          At Dart Ave/Exit 21 - Accident.  …       False
4                       At Mitchell Ave/Exit 6 - Accident.  …       False
…                                                      …  …         …
2845337                              At Market St - Accident.  …       False
2845338    At Camino Del Rio/Mission Center Rd - Accident.  …       False
2845339  At Glassell St/Grand Ave - Accident. in the ri…  …       False
2845340     At CA-90/Marina Fwy/Jefferson Blvd - Accident.  …       False
2845341              At Highland Ave/Arden Ave - Accident.  …       False

        Station   Stop Traffic_Calming Traffic_Signal Turning_Loop  \
0         False  False           False          False        False
1         False  False           False          False        False
2         False  False           False          False        False
3         False  False           False          False        False
4         False  False           False          False        False
…           …    …             …            …           …
2845337   False  False           False          False        False
2845338   False  False           False          False        False
2845339   False  False           False          False        False
2845340   False  False           False          False        False
2845341   False  False           False          False        False

        Sunrise_Sunset Civil_Twilight Nautical_Twilight Astronomical_Twilight
0                Night          Night             Night                 Night
1                Night          Night             Night                 Night
2                Night          Night             Night                   Day
3                Night          Night               Day                   Day
4                  Day            Day               Day                   Day
…                  …            …                 …                     …
2845337            Day            Day               Day                   Day
2845338            Day            Day               Day                   Day
2845339            Day            Day               Day                   Day
2845340            Day            Day               Day                   Day
2845341            Day            Day               Day                   Day

[2845342 rows x 47 columns]
```

```
[4]: df.columns
```

```
[4]: Index(['ID', 'Severity', 'Start_Time', 'End_Time', 'Start_Lat', 'Start_Lng',
            'End_Lat', 'End_Lng', 'Distance(mi)', 'Description', 'Number', 'Street',
            'Side', 'City', 'County', 'State', 'Zipcode', 'Country', 'Timezone',
            'Airport_Code', 'Weather_Timestamp', 'Temperature(F)', 'Wind_Chill(F)',
            'Humidity(%)', 'Pressure(in)', 'Visibility(mi)', 'Wind_Direction',
            'Wind_Speed(mph)', 'Precipitation(in)', 'Weather_Condition', 'Amenity',
            'Bump', 'Crossing', 'Give_Way', 'Junction', 'No_Exit', 'Railway',
            'Roundabout', 'Station', 'Stop', 'Traffic_Calming', 'Traffic_Signal',
            'Turning_Loop', 'Sunrise_Sunset', 'Civil_Twilight', 'Nautical_Twilight',
            'Astronomical_Twilight'],
           dtype='object')
```

```
[5]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2845342 entries, 0 to 2845341
Data columns (total 47 columns):
 #   Column                 Dtype
---  ------                 -----
 0   ID                     object
 1   Severity               int64
 2   Start_Time             object
 3   End_Time               object
 4   Start_Lat              float64
 5   Start_Lng              float64
 6   End_Lat                float64
 7   End_Lng                float64
 8   Distance(mi)           float64
 9   Description            object
 10  Number                 float64
 11  Street                 object
 12  Side                   object
 13  City                   object
 14  County                 object
 15  State                  object
 16  Zipcode                object
 17  Country                object
 18  Timezone               object
 19  Airport_Code           object
 20  Weather_Timestamp      object
 21  Temperature(F)         float64
 22  Wind_Chill(F)          float64
 23  Humidity(%)            float64
 24  Pressure(in)           float64
 25  Visibility(mi)         float64
 26  Wind_Direction         object
 27  Wind_Speed(mph)        float64
 28  Precipitation(in)      float64
 29  Weather_Condition      object
 30  Amenity                bool
 31  Bump                   bool
 32  Crossing               bool
 33  Give_Way               bool
 34  Junction               bool
 35  No_Exit                bool
 36  Railway                bool
 37  Roundabout             bool
 38  Station                bool
 39  Stop                   bool
 40  Traffic_Calming        bool
 41  Traffic_Signal         bool
 42  Turning_Loop           bool
 43  Sunrise_Sunset         object
 44  Civil_Twilight         object
 45  Nautical_Twilight      object
 46  Astronomical_Twilight  object
dtypes: bool(13), float64(13), int64(1), object(20)
memory usage: 773.4+ MB
```

[6]:
```
df.describe()
```

[6]:
```
           Severity     Start_Lat     Start_Lng       End_Lat       End_Lng  \
count  2.845342e+06  2.845342e+06  2.845342e+06  2.845342e+06  2.845342e+06
mean   2.137572e+00  3.624520e+01 -9.711463e+01  3.624532e+01 -9.711439e+01
std    4.787216e-01  5.363797e+00  1.831782e+01  5.363873e+00  1.831763e+01
min    1.000000e+00  2.456603e+01 -1.245481e+02  2.456601e+01 -1.245457e+02
25%    2.000000e+00  3.344517e+01 -1.180331e+02  3.344628e+01 -1.180333e+02
50%    2.000000e+00  3.609861e+01 -9.241808e+01  3.609799e+01 -9.241772e+01
75%    2.000000e+00  4.016024e+01 -8.037243e+01  4.016105e+01 -8.037338e+01
max    4.000000e+00  4.900058e+01 -6.711317e+01  4.907500e+01 -6.710924e+01

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

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

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

[7]:
```
numerics = ['int16', 'int32', 'int64', 'float16', 'float32', 'float64']

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

[7]:
```
14
```

Percentage of missing values per column

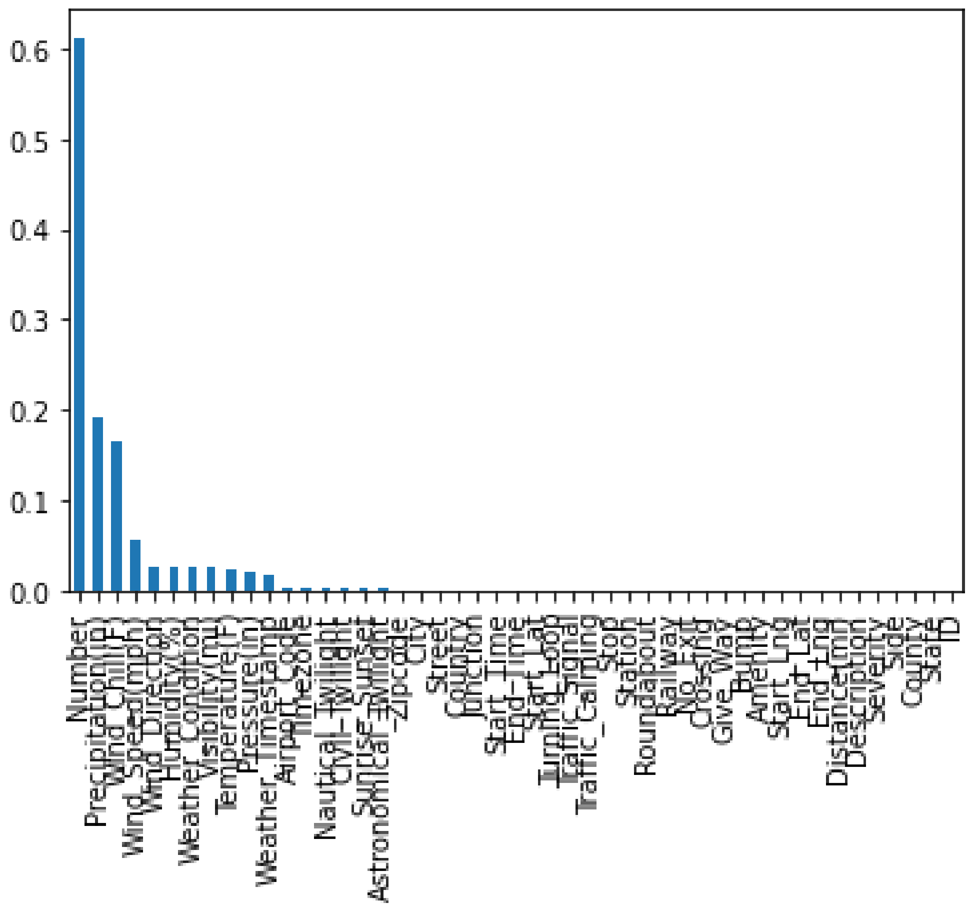
[8]:
```
missing_percentage = df.isna().sum().sort_values(ascending=False)/len(df)
missing_percentage
```

[8]:
```
Number                   6.129003e-01
Precipitation(in)        1.931079e-01
Wind_Chill(F)            1.650568e-01
Wind_Speed(mph)          5.550967e-02
Wind_Direction           2.592834e-02
Humidity(%)              2.568830e-02
Weather_Condition        2.482514e-02
Visibility(mi)           2.479350e-02
Temperature(F)           2.434646e-02
Pressure(in)             2.080593e-02
Weather_Timestamp        1.783125e-02
Airport_Code             3.356011e-03
Timezone                 1.285961e-03
Nautical_Twilight        1.007612e-03
Civil_Twilight           1.007612e-03
Sunrise_Sunset           1.007612e-03
```

```
Astronomical_Twilight    1.007612e-03
Zipcode                  4.635647e-04
City                     4.814887e-05
Street                   7.029032e-07
Country                  0.000000e+00
Junction                 0.000000e+00
Start_Time               0.000000e+00
End_Time                 0.000000e+00
Start_Lat                0.000000e+00
Turning_Loop             0.000000e+00
Traffic_Signal           0.000000e+00
Traffic_Calming          0.000000e+00
Stop                     0.000000e+00
Station                  0.000000e+00
Roundabout               0.000000e+00
Railway                  0.000000e+00
No_Exit                  0.000000e+00
Crossing                 0.000000e+00
Give_Way                 0.000000e+00
Bump                     0.000000e+00
Amenity                  0.000000e+00
Start_Lng                0.000000e+00
End_Lat                  0.000000e+00
End_Lng                  0.000000e+00
Distance(mi)             0.000000e+00
Description              0.000000e+00
Severity                 0.000000e+00
Side                     0.000000e+00
County                   0.000000e+00
State                    0.000000e+00
ID                       0.000000e+00
dtype: float64
```

[9]:
```
missing_percentage.plot(kind="bar")
```

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

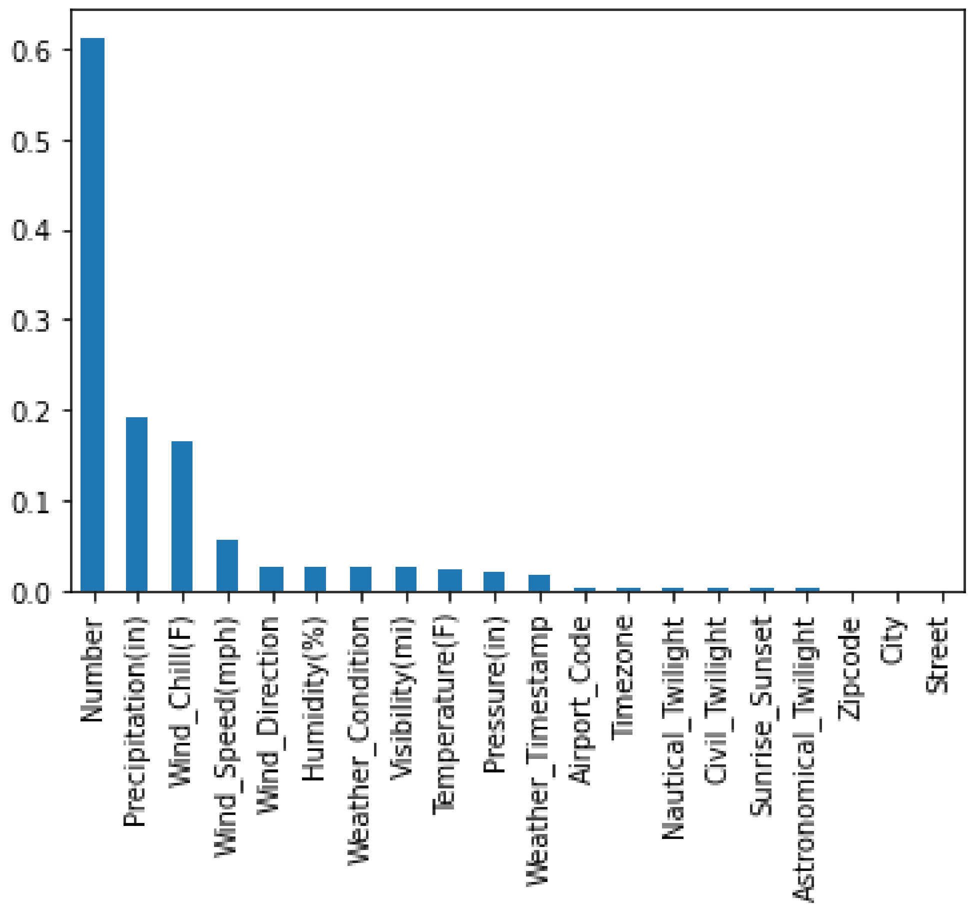


[10]:
```
type(missing_percentage)
```

[10]:
```
pandas.core.series.Series
```

[11]:
```
missing_percentage[missing_percentage!=0].plot(kind="bar")
```

[11]:
```
<AxesSubplot:>
```



Remove Columns that you dont want to use

1.2 Exploratory Analysis and Visualization

Columns we'll analyze: 1. City 2. Start_Time, End_time 3. Start_Lat , End_Lat 4. State 5. Amenity 4. Weather_Condition

```
[12]: df.columns
```

```
[12]: Index(['ID', 'Severity', 'Start_Time', 'End_Time', 'Start_Lat', 'Start_Lng',
             'End_Lat', 'End_Lng', 'Distance(mi)', 'Description', 'Number', 'Street',
             'Side', 'City', 'County', 'State', 'Zipcode', 'Country', 'Timezone',
             'Airport_Code', 'Weather_Timestamp', 'Temperature(F)', 'Wind_Chill(F)',
             'Humidity(%)', 'Pressure(in)', 'Visibility(mi)', 'Wind_Direction',
             'Wind_Speed(mph)', 'Precipitation(in)', 'Weather_Condition', 'Amenity',
             'Bump', 'Crossing', 'Give_Way', 'Junction', 'No_Exit', 'Railway',
             'Roundabout', 'Station', 'Stop', 'Traffic_Calming', 'Traffic_Signal',
             'Turning_Loop', 'Sunrise_Sunset', 'Civil_Twilight', 'Nautical_Twilight',
             'Astronomical_Twilight'],
      dtype='object')
```

1.2.1 City

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

```
[13]: array(['Dublin', 'Dayton', 'Cincinnati', …, 'Clarksdale', 'Bridgeboro',
              'American Fork-Pleasant Grove'], dtype=object)
```

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

```
[14]: Miami                           106966
      Los Angeles                      68956
      Orlando                          54691
      Dallas                           41979
      Houston                          39448
                                       …
      Ridgedale                            1
      Sekiu                                1
      Wooldridge                           1
      Bullock                              1
      American Fork-Pleasant Grove         1
      Name: City, Length: 11681, dtype: int64
```

```
[15]: cities_by_accident.head(20)
```

```
[15]: Miami            106966
      Los Angeles       68956
      Orlando           54691
      Dallas            41979
      Houston           39448
      Charlotte         33152
      Sacramento        32559
      San Diego         26627
      Raleigh           22840
      Minneapolis       22768
      Portland          20944
      Nashville         20267
      Austin            18301
      Baton Rouge       18182
      Phoenix           17143
      Saint Paul        16869
      New Orleans       16251
      Atlanta           15622
      Jacksonville      14967
```

```
Richmond         14349
Name: City, dtype: int64
```

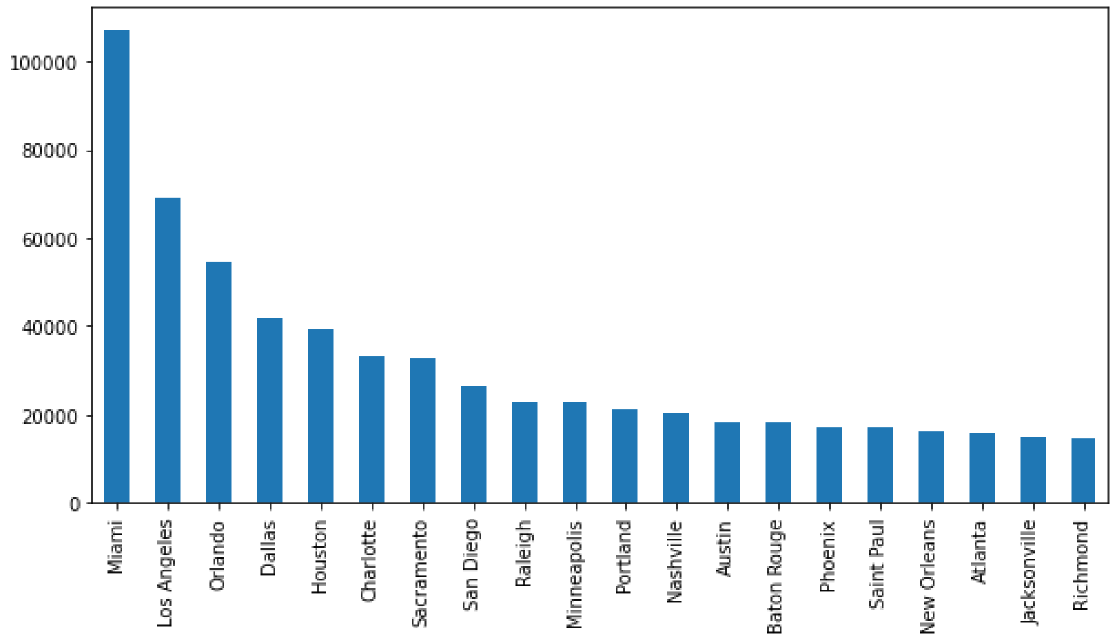
[16]:
```
"New York" in df.City
```

[16]:
```
False
```

[17]:
```
import matplotlib.pyplot as plt
```

[18]:
```
plt.figure(figsize=(10,5))
cities_by_accident.head(20).plot(kind="bar")
```

[18]:
```
<AxesSubplot:>
```

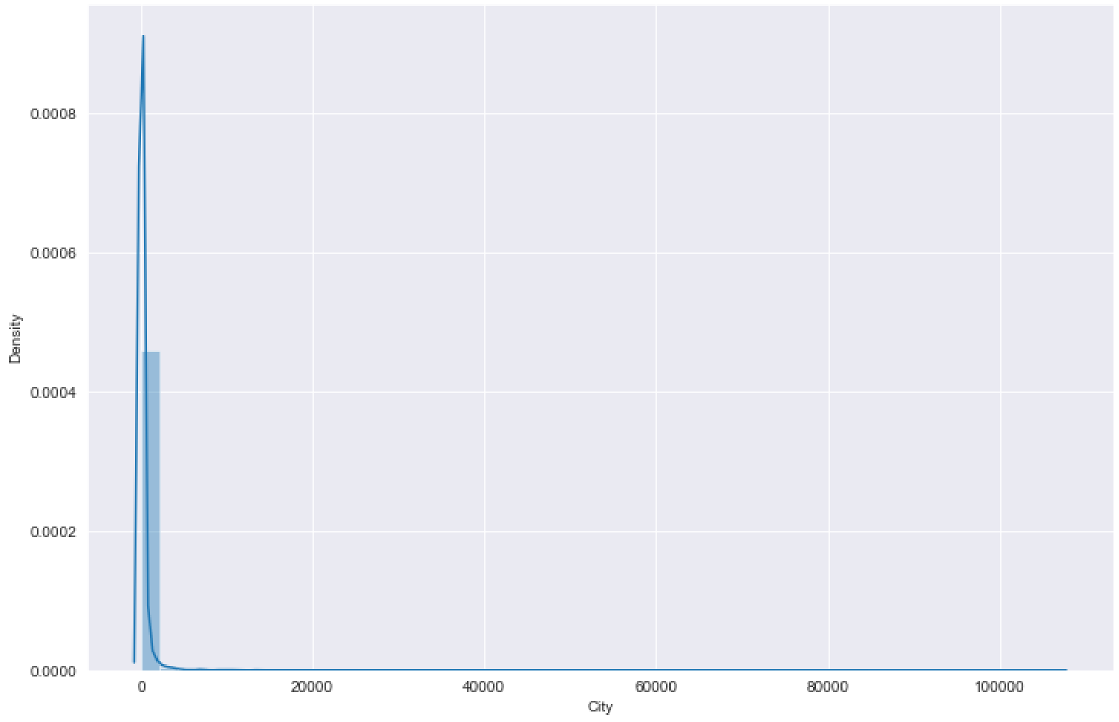


[19]:
```
import seaborn as sns
sns.set_style("darkgrid")
```

[20]:
```
plt.figure(figsize=(12,8))
sns.distplot(cities_by_accident)
```

```
C:\Users\admin\anaconda3\lib\site-packages\seaborn\distributions.py:2619:
FutureWarning: `distplot` is a deprecated function and will be removed in a
future version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for
histograms).
  warnings.warn(msg, FutureWarning)
```

```
[20]: <AxesSubplot:xlabel='City', ylabel='Density'>
```



```
[21]: high_accident_cities = cities_by_accident[cities_by_accident>=1000]
      low_accident_cities = cities_by_accident[cities_by_accident<1000]
```

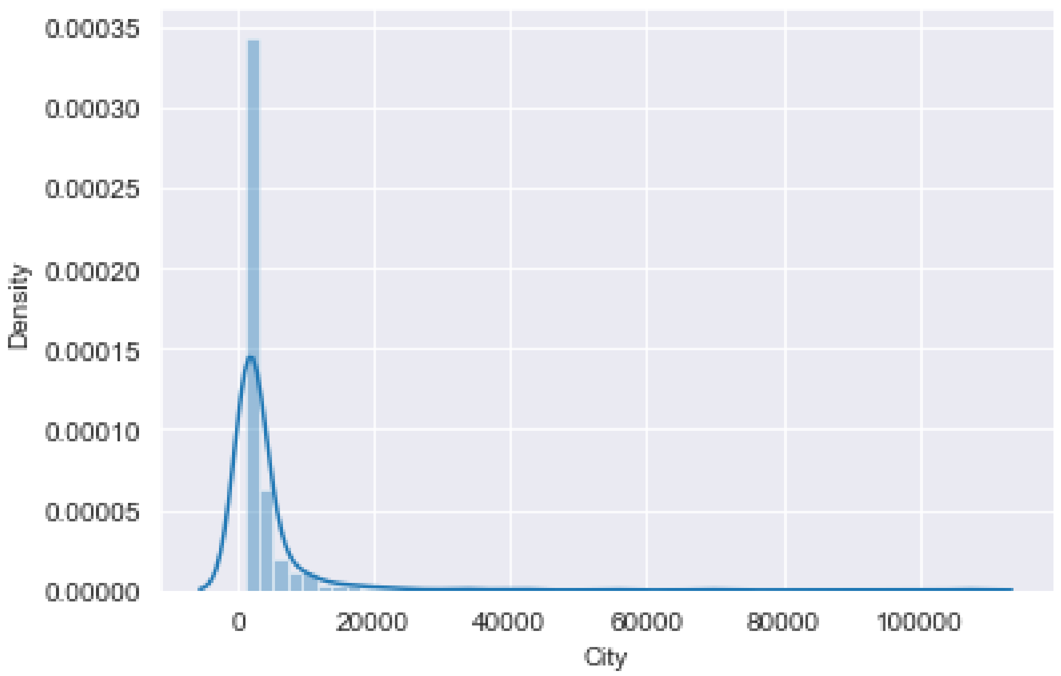
```
[22]: len(high_accident_cities)/len(cities)
```

```
[22]: 0.04245848313644924
```

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

```
C:\Users\admin\anaconda3\lib\site-packages\seaborn\distributions.py:2619:
FutureWarning: `distplot` is a deprecated function and will be removed in a
future version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for
histograms).
  warnings.warn(msg, FutureWarning)
```

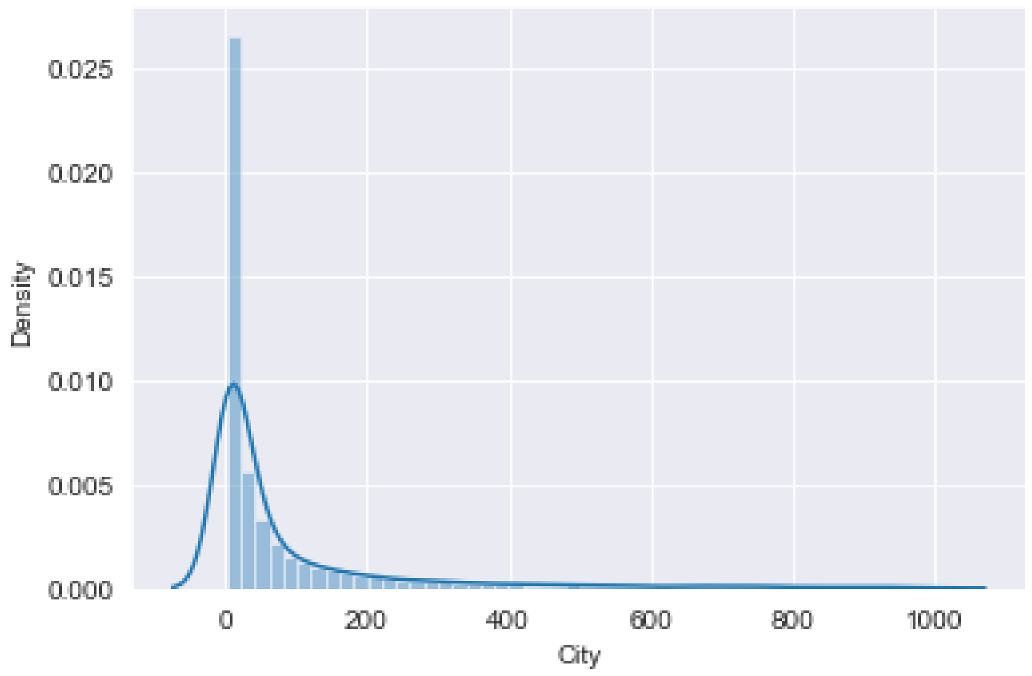
```
[23]: <AxesSubplot:xlabel='City', ylabel='Density'>
```



[24]:
```
sns.distplot(low_accident_cities)
```

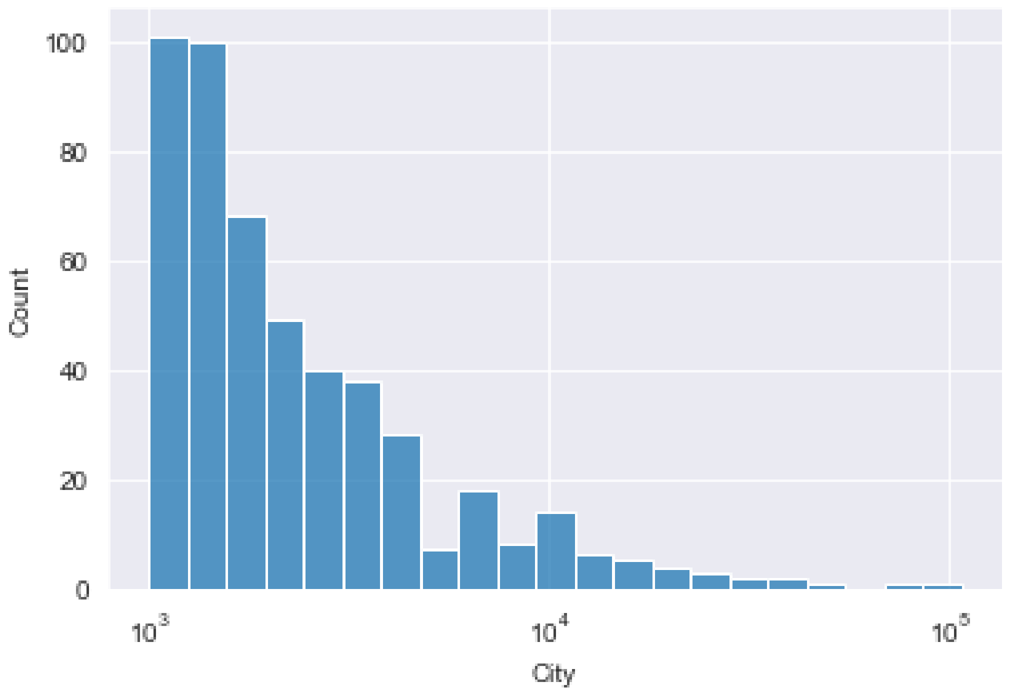
```
C:\Users\admin\anaconda3\lib\site-packages\seaborn\distributions.py:2619:
FutureWarning: `distplot` is a deprecated function and will be removed in a
future version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for
histograms).
  warnings.warn(msg, FutureWarning)
```

[24]:
```
<AxesSubplot:xlabel='City', ylabel='Density'>
```



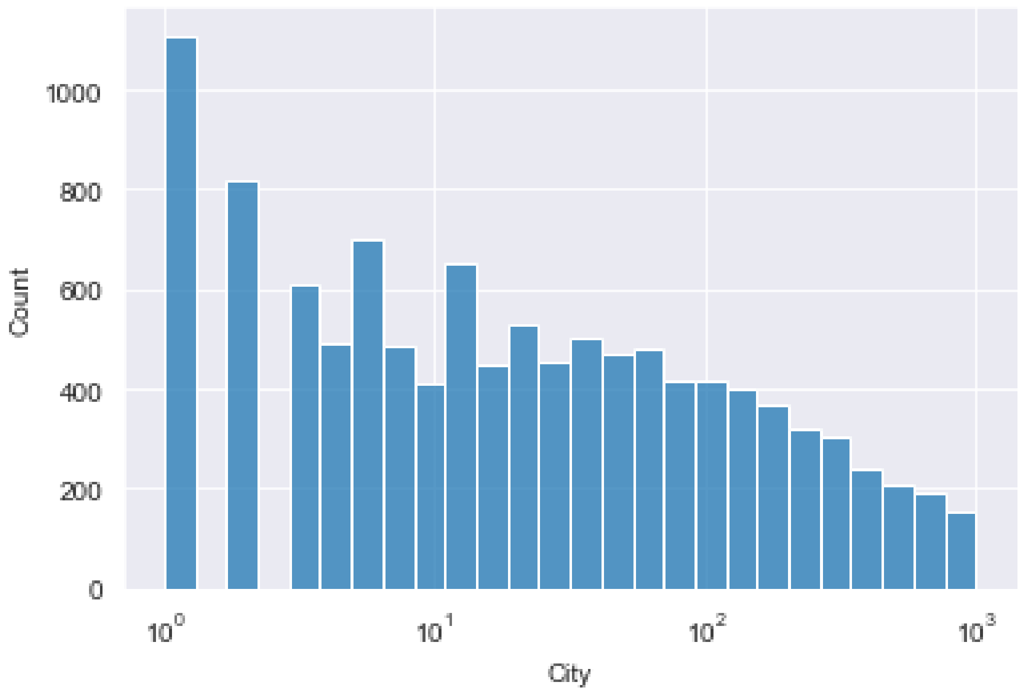
```
[25]: sns.histplot(high_accident_cities, log_scale=True)
```

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



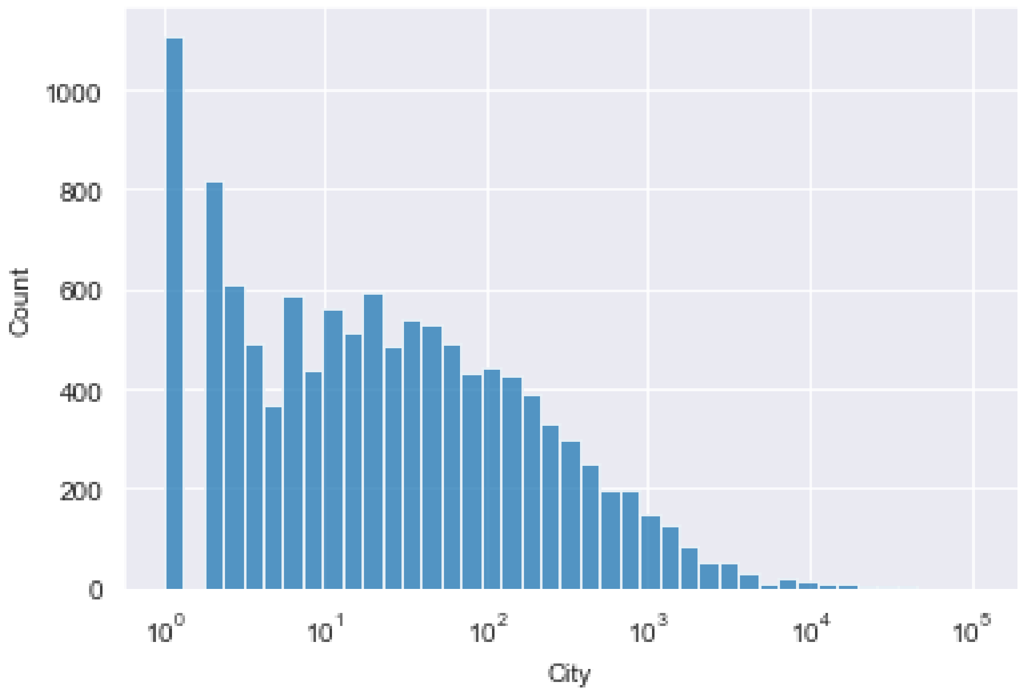
[26]:
```
sns.histplot(low_accident_cities, log_scale=True)
```

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



[27]:
```
sns.histplot(cities_by_accident, log_scale=True)
```

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



[28]:
```
cities_by_accident[cities_by_accident==1]
```

[28]:
```
Carney                          1
Waverly Hall                    1
Center Sandwich                 1
Glen Flora                      1
Sulphur Springs                 1
                               ..
Ridgedale                       1
Sekiu                           1
Wooldridge                      1
Bullock                         1
American Fork-Pleasant Grove    1
Name: City, Length: 1110, dtype: int64
```

1.2.2 Start Time

[29]:
```
df.columns
```

[29]:
```
Index(['ID', 'Severity', 'Start_Time', 'End_Time', 'Start_Lat', 'Start_Lng',
       'End_Lat', 'End_Lng', 'Distance(mi)', 'Description', 'Number', 'Street',
       'Side', 'City', 'County', 'State', 'Zipcode', 'Country', 'Timezone',
       'Airport_Code', 'Weather_Timestamp', 'Temperature(F)', 'Wind_Chill(F)',
       'Humidity(%)', 'Pressure(in)', 'Visibility(mi)', 'Wind_Direction',
       'Wind_Speed(mph)', 'Precipitation(in)', 'Weather_Condition', 'Amenity',
       'Bump', 'Crossing', 'Give_Way', 'Junction', 'No_Exit', 'Railway',
       'Roundabout', 'Station', 'Stop', 'Traffic_Calming', 'Traffic_Signal',
       'Turning_Loop', 'Sunrise_Sunset', 'Civil_Twilight', 'Nautical_Twilight',
       'Astronomical_Twilight'],
      dtype='object')
```

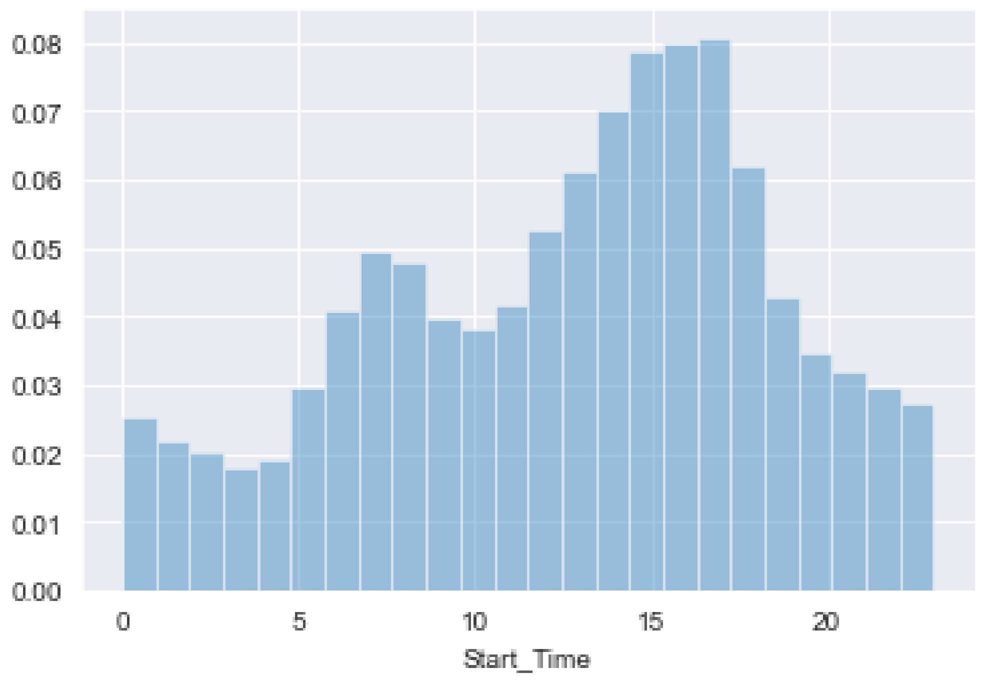
[30]:
```
df.Start_Time = pd.to_datetime(df.Start_Time)
df.Start_Time
```

[30]:
```
0         2016-02-08 00:37:08
1         2016-02-08 05:56:20
2         2016-02-08 06:15:39
3         2016-02-08 06:51:45
4         2016-02-08 07:53:43
                  …
2845337   2019-08-23 18:03:25
2845338   2019-08-23 19:11:30
2845339   2019-08-23 19:00:21
2845340   2019-08-23 19:00:21
2845341   2019-08-23 18:52:06
Name: Start_Time, Length: 2845342, dtype: datetime64[ns]
```

[31]:
```
sns.distplot(df.Start_Time.dt.hour,bins=24,kde=False ,norm_hist=True)
```

```
C:\Users\admin\anaconda3\lib\site-packages\seaborn\distributions.py:2619:
FutureWarning: `distplot` is a deprecated function and will be removed in a
future version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for
histograms).
  warnings.warn(msg, FutureWarning)
```

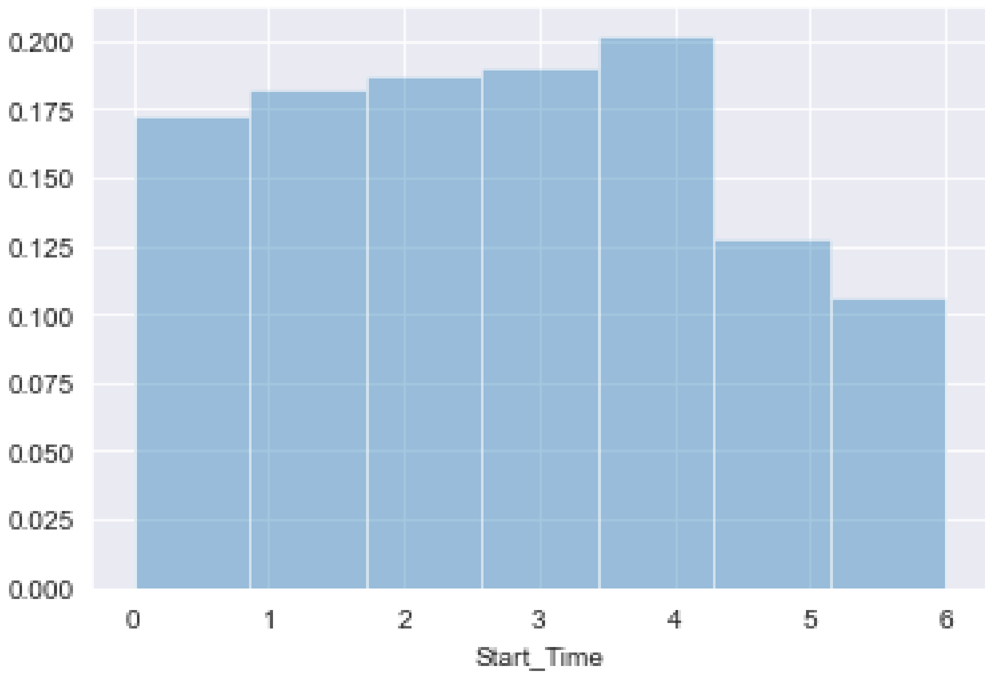
[31]:
```
<AxesSubplot:xlabel='Start_Time'>
```



- A hight percentage of accident happend between 6AM to 10 AM , may be due to work time
- next high percentage of accident happend between 3PM to 6PM.

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

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

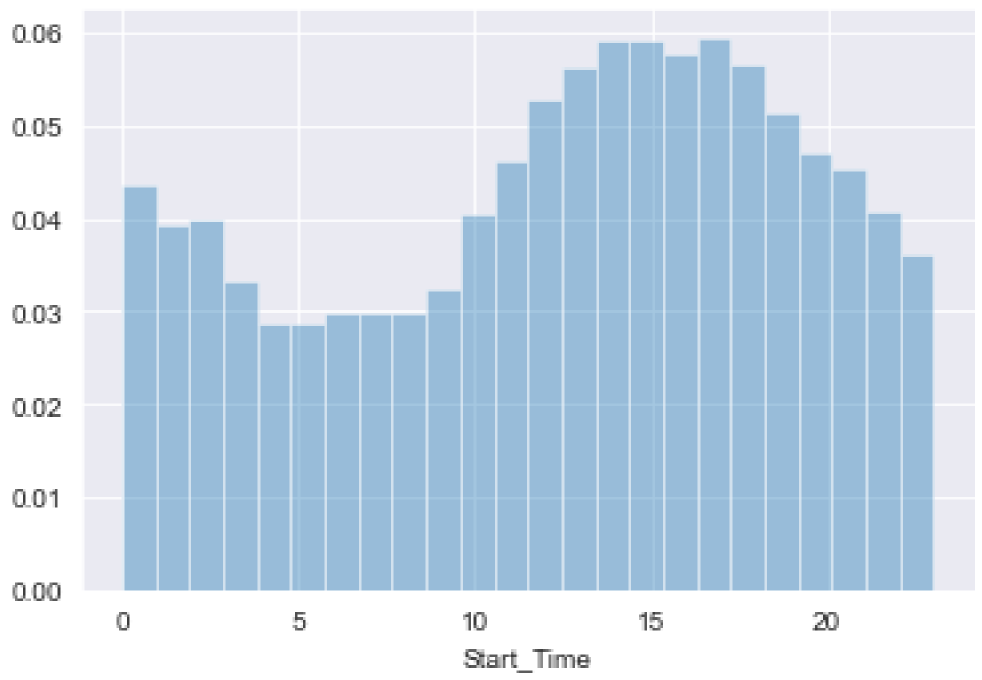


is the distribution of accidents by hour the same on weekends as on weekdays

```
[33]: sunday_start_time=df.Start_Time[df.Start_Time.dt.dayofweek==6]
```

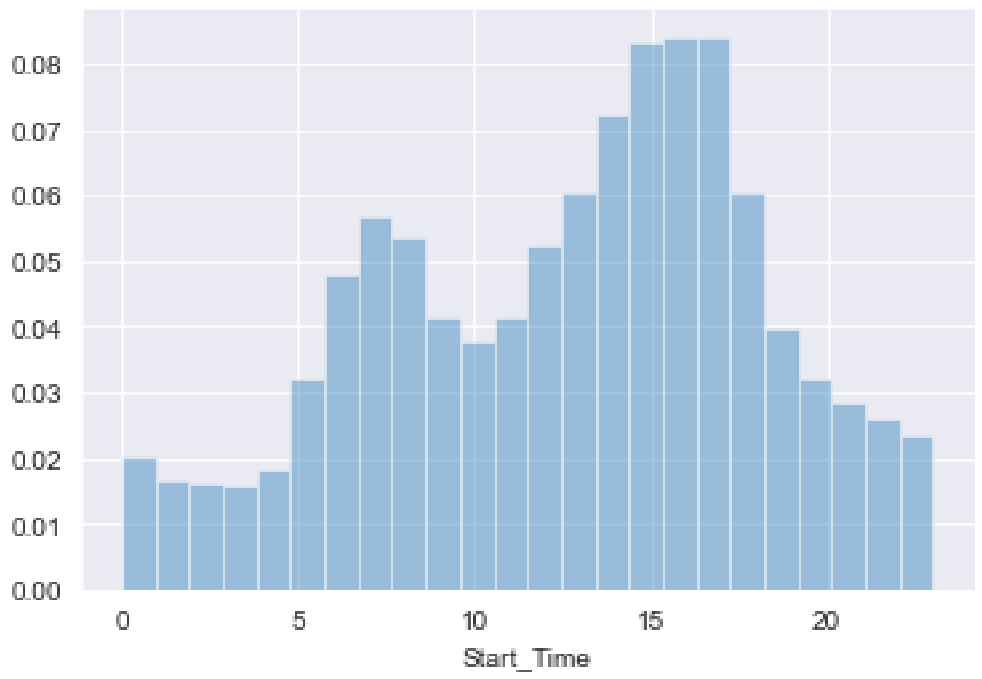
```
[34]: sns.distplot(sunday_start_time.dt.hour,kde=False ,norm_hist=True,bins=24)
```

```
[34]: <AxesSubplot:xlabel='Start_Time'>
```



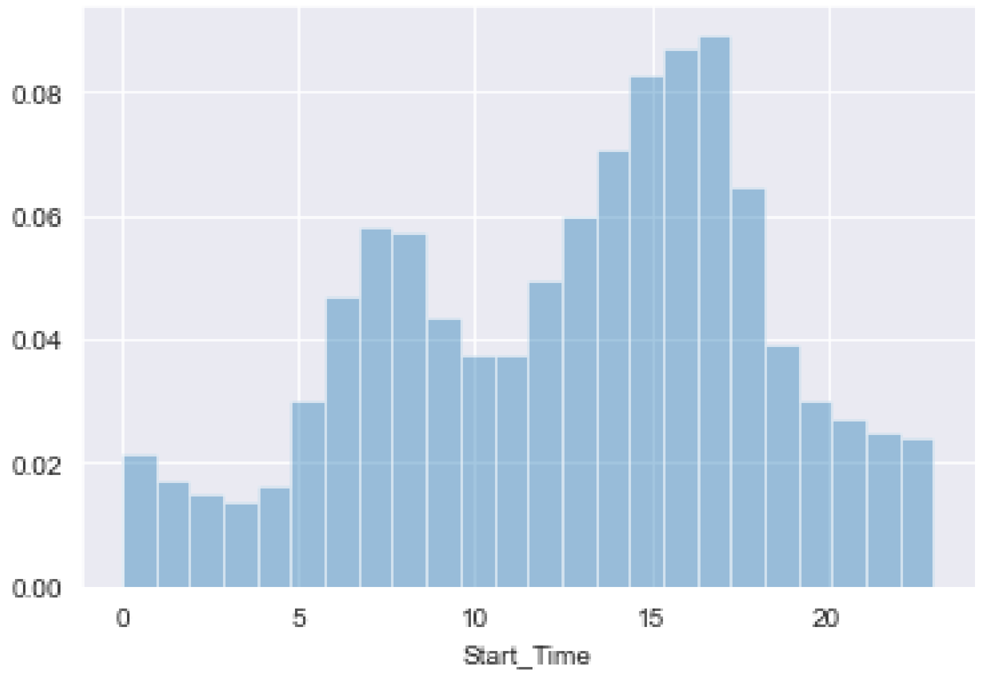
```
[35]: monday_start_time=df.Start_Time[df.Start_Time.dt.dayofweek==0]
      sns.distplot(monday_start_time.dt.hour,kde=False ,norm_hist=True,bins=24)
```

```
[35]: <AxesSubplot:xlabel='Start_Time'>
```



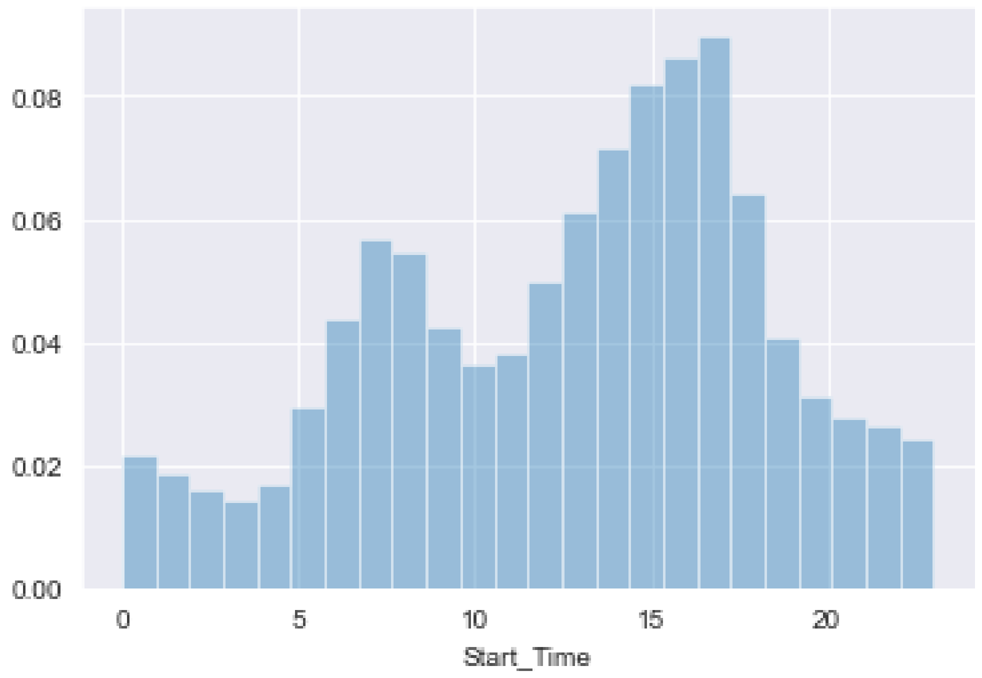
```
[36]: tuesday_start_time=df.Start_Time[df.Start_Time.dt.dayofweek==1]
      sns.distplot(tuesday_start_time.dt.hour,kde=False ,norm_hist=True,bins=24)
```

```
[36]: <AxesSubplot:xlabel='Start_Time'>
```



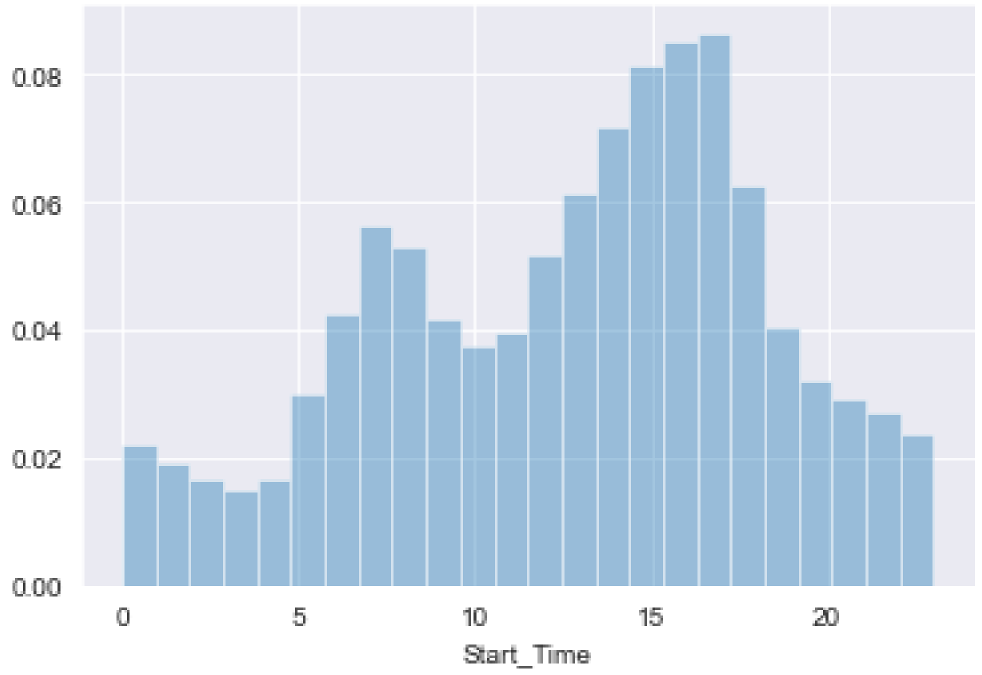
```
[37]: wednesday_start_time=df.Start_Time[df.Start_Time.dt.dayofweek==2]
      sns.distplot(wednesday_start_time.dt.hour,kde=False ,norm_hist=True,bins=24)
```

```
[37]: <AxesSubplot:xlabel='Start_Time'>
```



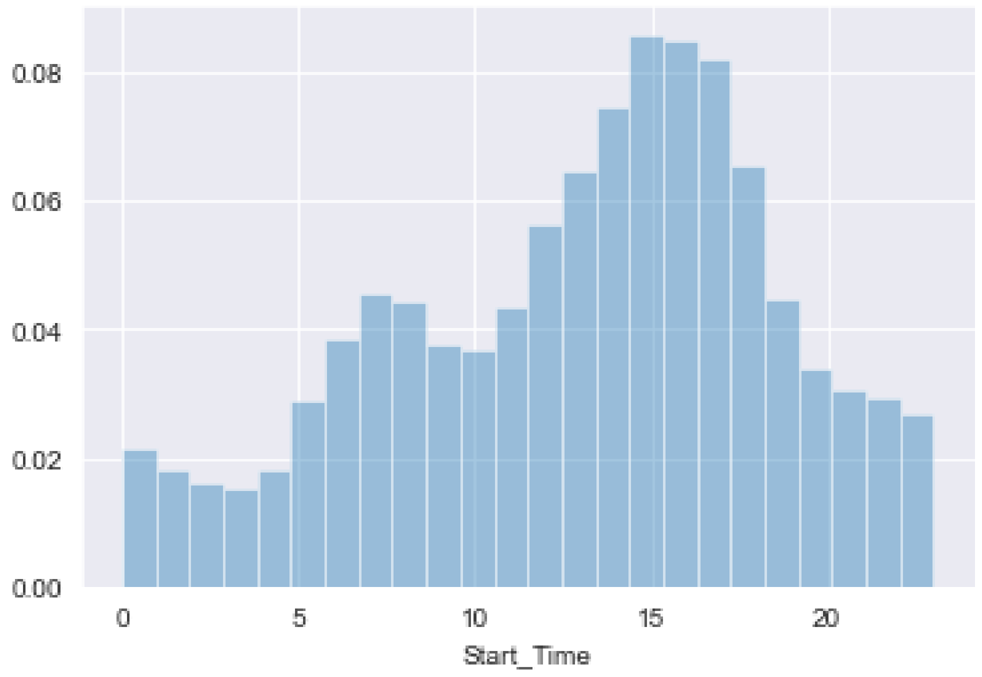
```
[38]: thursday_start_time=df.Start_Time[df.Start_Time.dt.dayofweek==3]
      sns.distplot(thursday_start_time.dt.hour,kde=False ,norm_hist=True,bins=24)
```

```
[38]: <AxesSubplot:xlabel='Start_Time'>
```



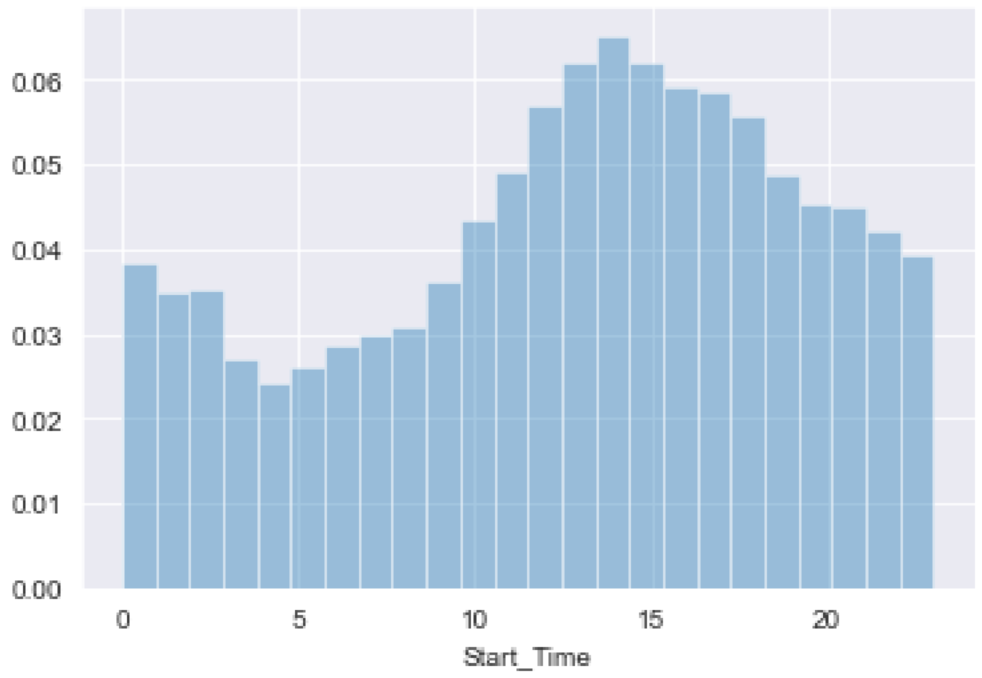
[39]:
```
friday_start_time=df.Start_Time[df.Start_Time.dt.dayofweek==4]
sns.distplot(friday_start_time.dt.hour,kde=False ,norm_hist=True,bins=24)
```

[39]:
```
<AxesSubplot:xlabel='Start_Time'>
```



[40]:
```
saturday_start_time=df.Start_Time[df.Start_Time.dt.dayofweek==5]
sns.distplot(saturday_start_time.dt.hour,kde=False ,norm_hist=True,bins=24)
```

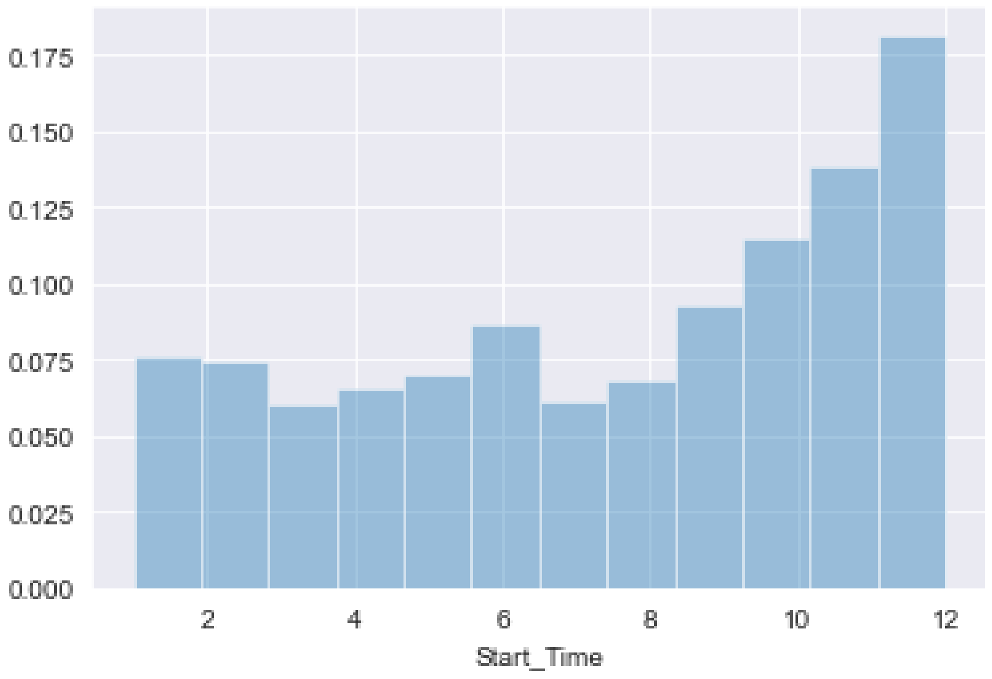
[40]:
```
<AxesSubplot:xlabel='Start_Time'>
```



on sunday the peak occur is between 10AM to 8PM

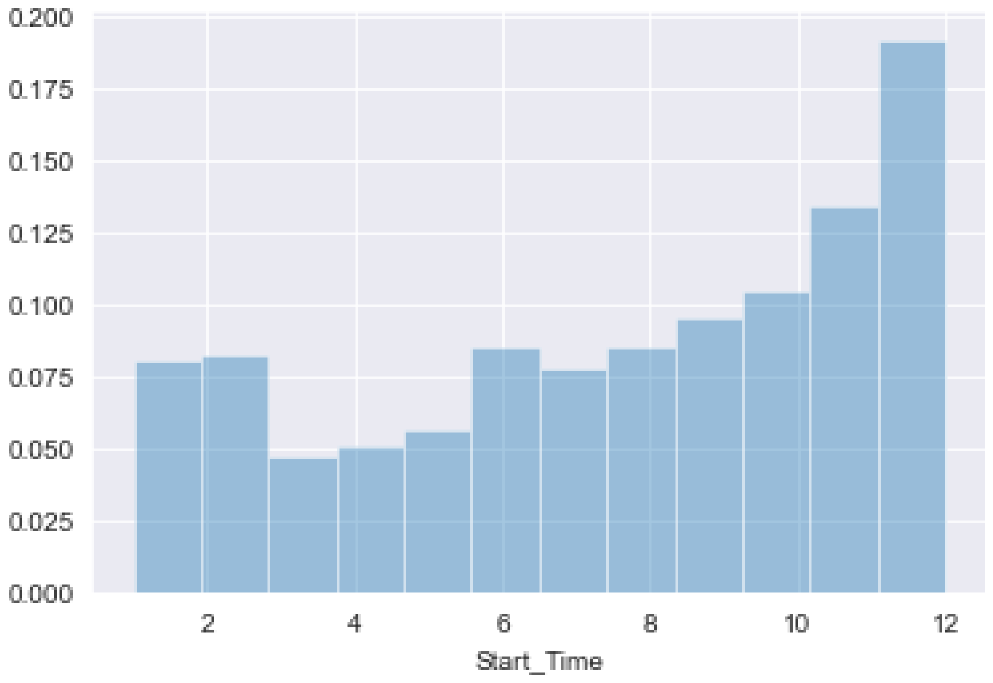
```
[41]: sns.distplot(df.Start_Time.dt.month,kde=False ,norm_hist=True,bins=12)
```

```
[41]: <AxesSubplot:xlabel='Start_Time'>
```



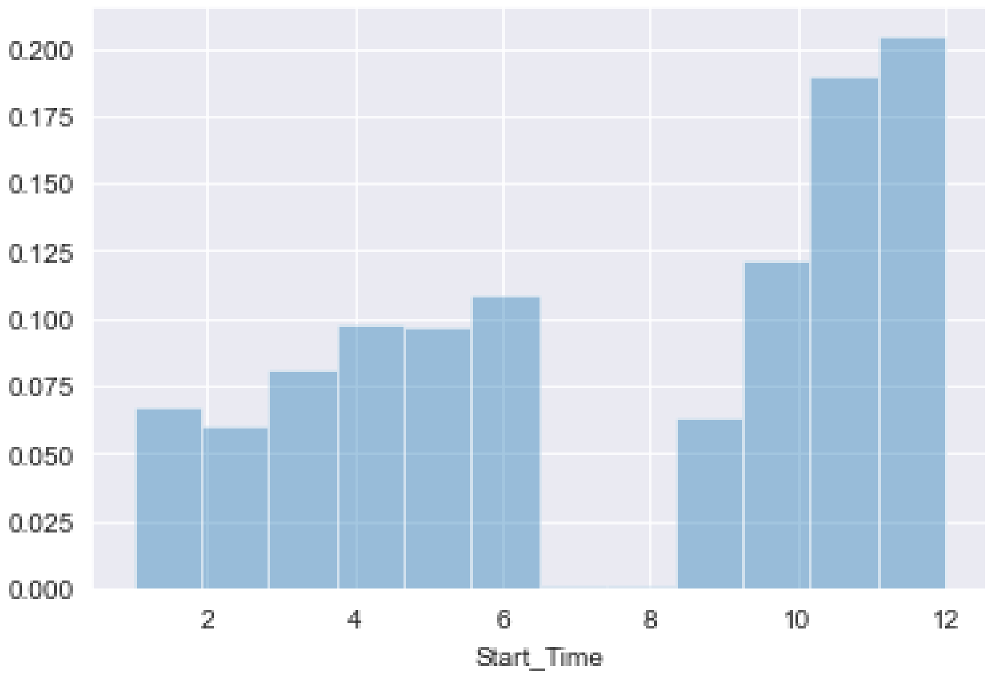
```
[42]: df_2021=df.Start_Time[df.Start_Time.dt.year==2021]
      sns.distplot(df_2021.dt.month,kde=False ,norm_hist=True,bins=12)
```

```
[42]: <AxesSubplot:xlabel='Start_Time'>
```



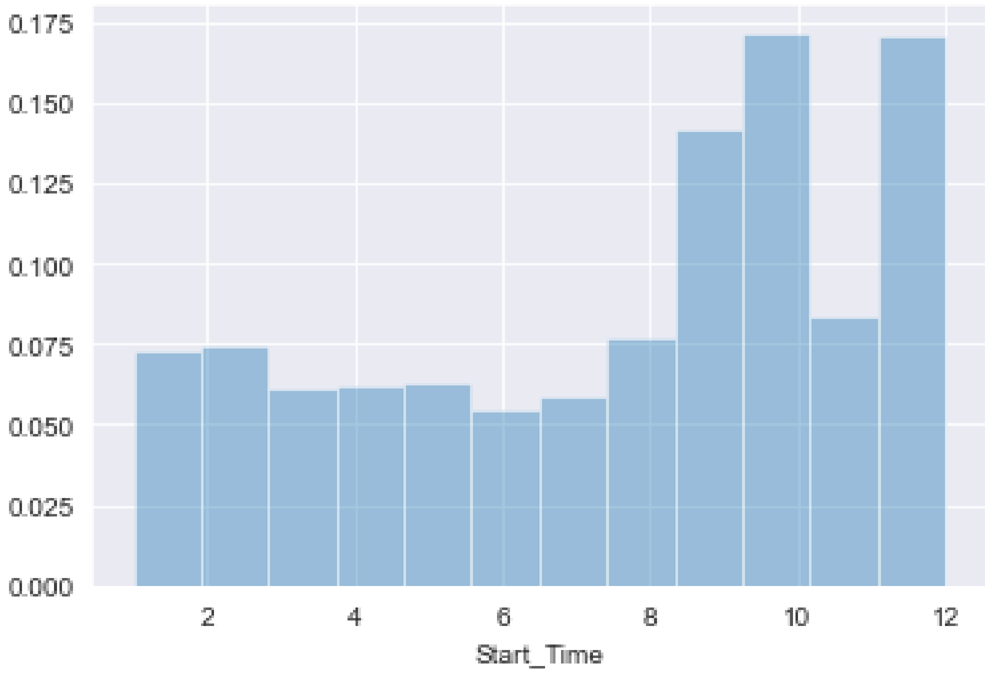
```
[43]: df_2020=df.Start_Time[df.Start_Time.dt.year==2020]
      sns.distplot(df_2020.dt.month,kde=False ,norm_hist=True,bins=12)
```

```
[43]: <AxesSubplot:xlabel='Start_Time'>
```



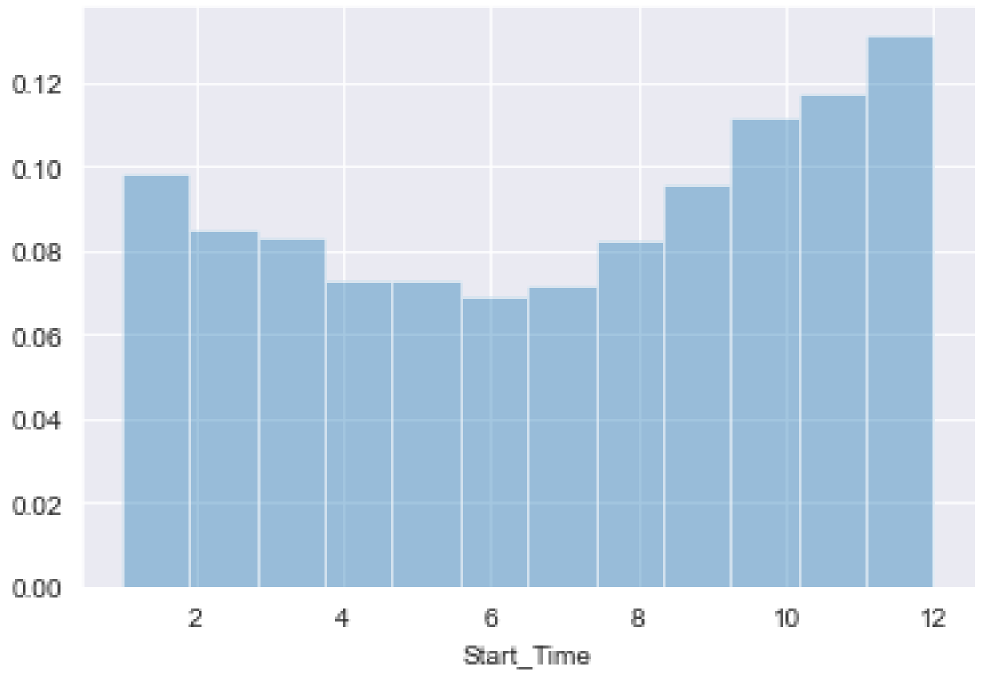
```
[44]: df_2019=df.Start_Time[df.Start_Time.dt.year==2019]
      sns.distplot(df_2019.dt.month,kde=False ,norm_hist=True,bins=12)
```

```
[44]: <AxesSubplot:xlabel='Start_Time'>
```



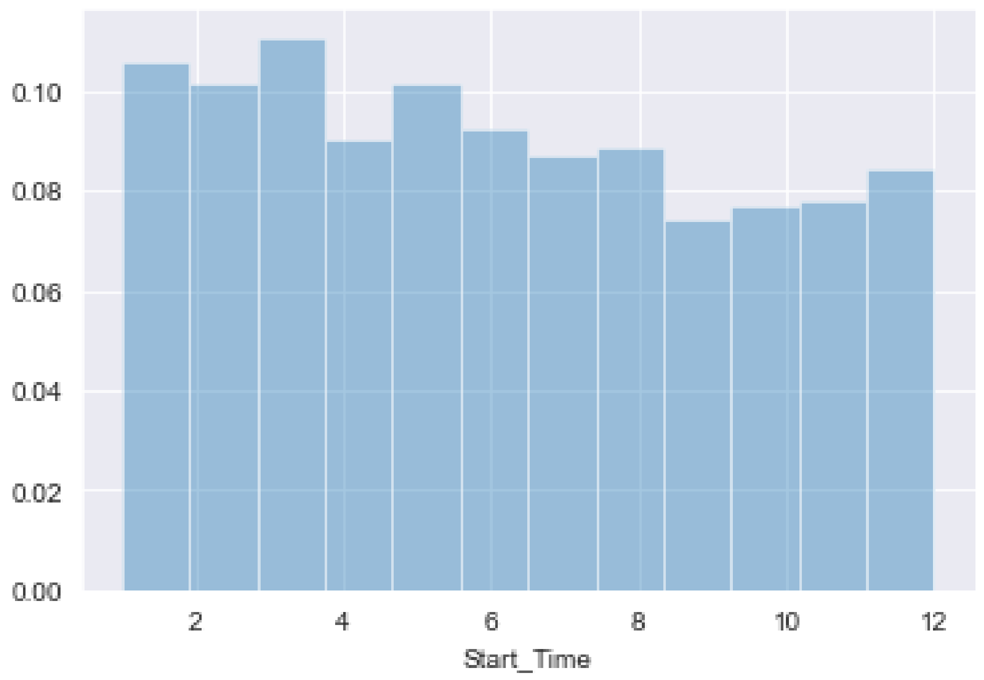
[45]:
```
df_2018=df.Start_Time[df.Start_Time.dt.year==2018]
sns.distplot(df_2018.dt.month,kde=False ,norm_hist=True,bins=12)
```

[45]:
```
<AxesSubplot:xlabel='Start_Time'>
```



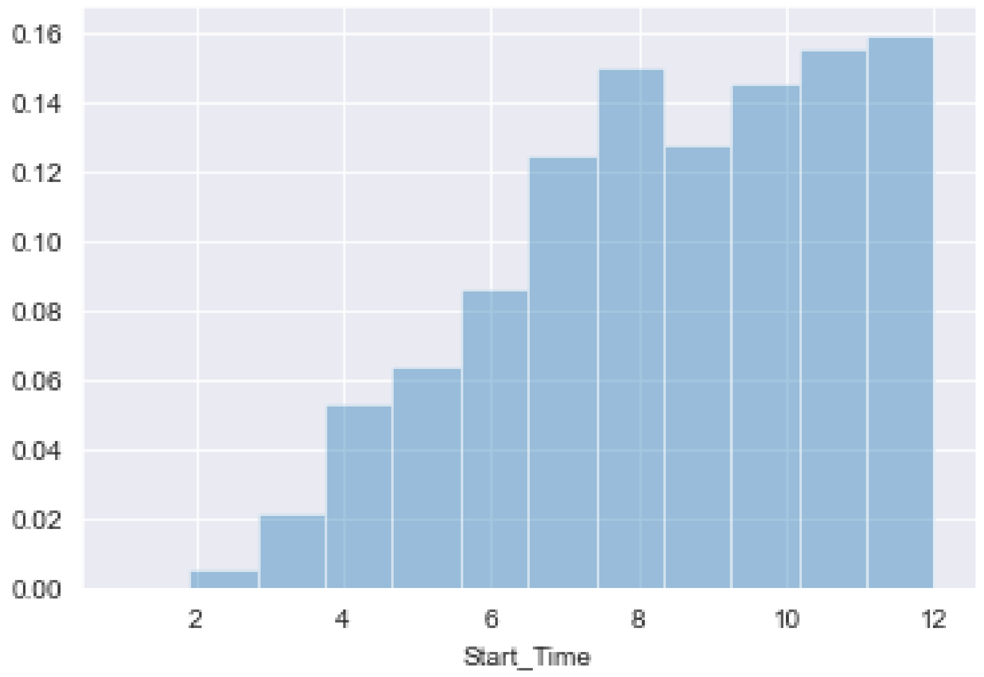
[46]:
```
df_2017=df.Start_Time[df.Start_Time.dt.year==2017]
sns.distplot(df_2017.dt.month,kde=False ,norm_hist=True,bins=12)
```

[46]:
```
<AxesSubplot:xlabel='Start_Time'>
```



```
[47]: df_2016=df.Start_Time[df.Start_Time.dt.year==2016]
      sns.distplot(df_2016.dt.month,kde=False ,norm_hist=True,bins=12)
```

```
[47]: <AxesSubplot:xlabel='Start_Time'>
```



1.2.3 Start Latitude & Longitude

```
[48]: df.Start_Lat
```

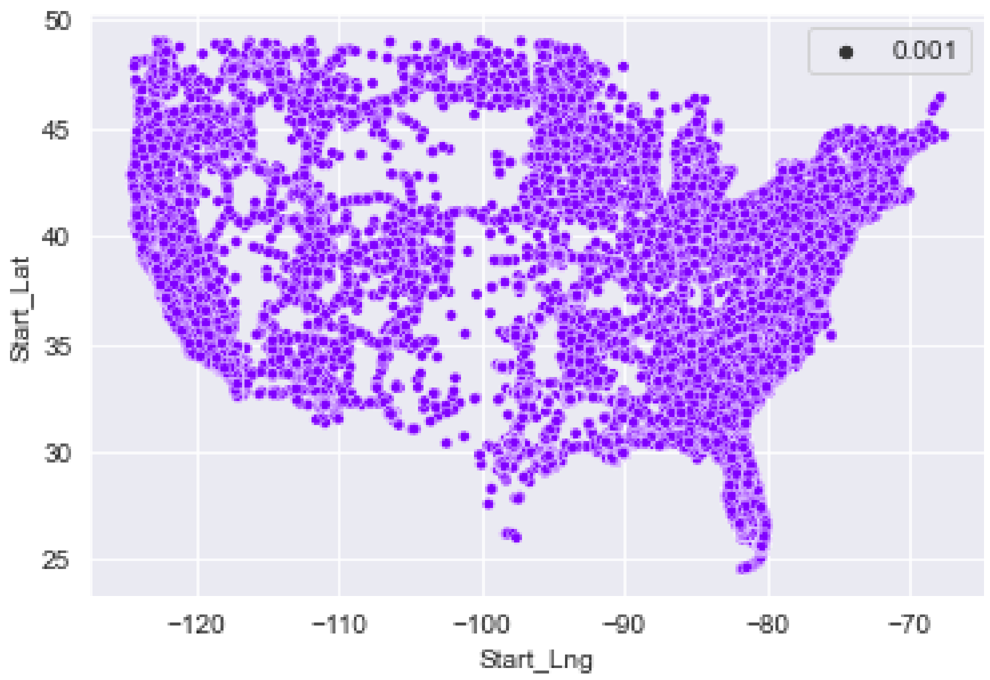
```
[48]: 0          40.108910
      1          39.865420
      2          39.102660
      3          41.062130
      4          39.172393
                   …
      2845337    34.002480
      2845338    32.766960
      2845339    33.775450
      2845340    33.992460
      2845341    34.133930
      Name: Start_Lat, Length: 2845342, dtype: float64
```

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

```
[49]: 0          -83.092860
      1          -84.062800
      2          -84.524680
      3          -81.537840
      4          -84.492792
                    …
      2845337   -117.379360
      2845338   -117.148060
      2845339   -117.847790
      2845340   -118.403020
      2845341   -117.230920
      Name: Start_Lng, Length: 2845342, dtype: float64
```

```
[50]: import matplotlib.cm as cm
      import numpy as np
      sample_df=df.sample(int(0.1 * len(df)))
      colors = iter(cm.rainbow(np.linspace(0, 1, len(sample_df))))
      sns.scatterplot(x=sample_df.Start_Lng,y=sample_df.Start_Lat,size=0.
       ↪001,color=next(colors))
```

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



[52]:

```
pip install folium
```

```
Collecting foliumNote: you may need to restart the kernel to use updated
packages.

  Downloading folium-0.14.0-py2.py3-none-any.whl (102 kB)
Collecting branca>=0.6.0
  Downloading branca-0.6.0-py3-none-any.whl (24 kB)
Requirement already satisfied: jinja2>=2.9 in c:\users\admin\anaconda3\lib\site-
packages (from folium) (2.11.3)
Requirement already satisfied: requests in c:\users\admin\anaconda3\lib\site-
packages (from folium) (2.26.0)
Requirement already satisfied: numpy in c:\users\admin\anaconda3\lib\site-
packages (from folium) (1.20.3)
Requirement already satisfied: MarkupSafe>=0.23 in
c:\users\admin\anaconda3\lib\site-packages (from jinja2>=2.9->folium) (1.1.1)
Requirement already satisfied: charset-normalizer~=2.0.0 in
c:\users\admin\anaconda3\lib\site-packages (from requests->folium) (2.0.4)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in
c:\users\admin\anaconda3\lib\site-packages (from requests->folium) (1.26.7)
Requirement already satisfied: certifi>=2017.4.17 in
c:\users\admin\anaconda3\lib\site-packages (from requests->folium) (2022.9.24)
Requirement already satisfied: idna<4,>=2.5 in
c:\users\admin\anaconda3\lib\site-packages (from requests->folium) (3.2)
Installing collected packages: branca, folium
```

```
Successfully installed branca-0.6.0 folium-0.14.0
```

```
[53]: import folium
```

```
[54]: this_map = folium.Map(prefer_canvas=True)
```

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

```
[55]: (40.10891, -83.09286)
```

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

```
[57]: sample_df= df.sample(int(0.01 * len(df)))
      lat_lon_pairs =list(zip(list(sample_df.Start_Lat),list(sample_df.Start_Lng)))
```

```
[58]: map=folium.Map()
      HeatMap(lat_lon_pairs[:100]).add_to(map)
      map
```

```
[58]: <folium.folium.Map at 0x1ffecb40220>
```

1.2.4 End_Lat & End_Lng

```
[59]: df.End_Lat
```

```
[59]: 0          40.112060
      1          39.865010
      2          39.102090
      3          41.062170
      4          39.170476
                   …
      2845337    33.998880
      2845338    32.765550
      2845339    33.777400
      2845340    33.983110
      2845341    34.137360
      Name: End_Lat, Length: 2845342, dtype: float64
```

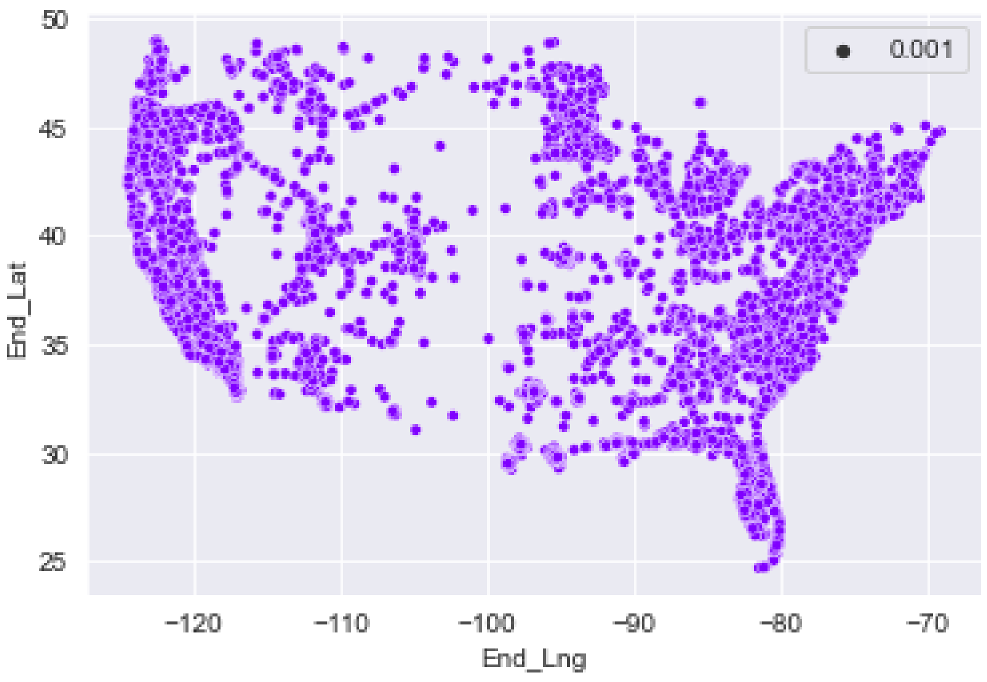
```
[60]: df.End_Lng
```

```
[60]: 0          -83.031870
      1          -84.048730
      2          -84.523960
      3          -81.535470
      4          -84.501798
                    …
```

```
2845337   -117.370940
2845338   -117.153630
2845339   -117.857270
2845340   -118.395650
2845341   -117.239340
Name: End_Lng, Length: 2845342, dtype: float64
```

[61]:
```
sns.scatterplot(x=sample_df.End_Lng,y=sample_df.End_Lat,size=0.
 ↪001,color=next(colors))
```

[61]:
```
<AxesSubplot:xlabel='End_Lng', ylabel='End_Lat'>
```



[63]:
```
lat_lon_end_pairs =list(zip(list(sample_df.End_Lat),list(sample_df.End_Lng)))
```

[64]:
```
HeatMap(lat_lon_end_pairs[:100]).add_to(map)
map
```

[64]:
```
<folium.folium.Map at 0x1ffecb40220>
```

1.2.5 State

[65]:
```
df.columns
```

```
[65]: Index(['ID', 'Severity', 'Start_Time', 'End_Time', 'Start_Lat', 'Start_Lng',
             'End_Lat', 'End_Lng', 'Distance(mi)', 'Description', 'Number', 'Street',
             'Side', 'City', 'County', 'State', 'Zipcode', 'Country', 'Timezone',
             'Airport_Code', 'Weather_Timestamp', 'Temperature(F)', 'Wind_Chill(F)',
             'Humidity(%)', 'Pressure(in)', 'Visibility(mi)', 'Wind_Direction',
             'Wind_Speed(mph)', 'Precipitation(in)', 'Weather_Condition', 'Amenity',
             'Bump', 'Crossing', 'Give_Way', 'Junction', 'No_Exit', 'Railway',
             'Roundabout', 'Station', 'Stop', 'Traffic_Calming', 'Traffic_Signal',
             'Turning_Loop', 'Sunrise_Sunset', 'Civil_Twilight', 'Nautical_Twilight',
             'Astronomical_Twilight'],
            dtype='object')
```

```
[66]: df.State
```

```
[66]: 0          OH
      1          OH
      2          OH
      3          OH
      4          OH
                 ..
      2845337    CA
      2845338    CA
      2845339    CA
      2845340    CA
      2845341    CA
      Name: State, Length: 2845342, dtype: object
```

```
[67]: states_by_accident = df.State.value_counts()
      states_by_accident
```

```
[67]: CA    795868
      FL    401388
      TX    149037
      OR    126341
      VA    113535
      NY    108049
      PA     99975
      MN     97185
      NC     91362
      SC     89216
      MD     65085
      AZ     56504
      NJ     52902
      TN     52613
      UT     49193
      LA     47232
      IL     47105
```

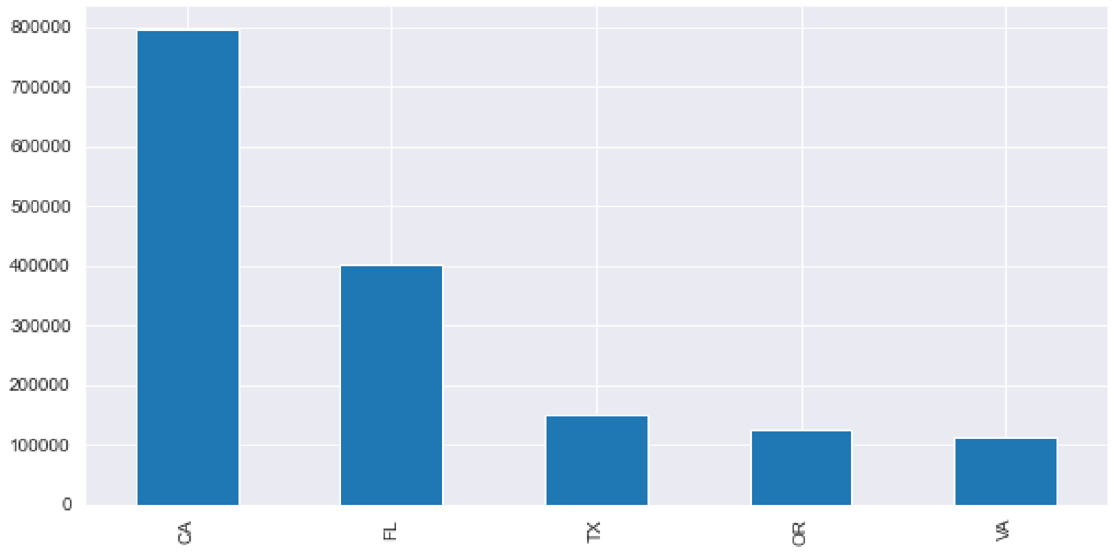
```
MI     43843
GA     40086
WA     32554
CT     29762
MO     29633
CO     25340
OH     24409
IN     20850
AL     19322
MT     15964
AR     10935
IA      9607
DC      9133
KS      9033
OK      8806
ID      8544
WI      7896
WV      7632
KY      6638
MA      6392
NV      6197
MS      5320
DE      4842
RI      4451
NH      3866
NE      3320
NM      2370
ND      2258
ME      2193
WY       990
VT       365
SD       201
Name: State, dtype: int64
```

[68]:
```
states_by_accident.head()
```

[68]:
```
CA    795868
FL    401388
TX    149037
OR    126341
VA    113535
Name: State, dtype: int64
```

[69]:
```
plt.figure(figsize=(10,5))
states_by_accident.head().plot(kind="bar")
```

[69]:
```
<AxesSubplot:>
```



1.2.6 Amenity

```
[70]: list(df.Amenity)== True
```

```
[70]: False
```

1.2.7 Weather_Condition

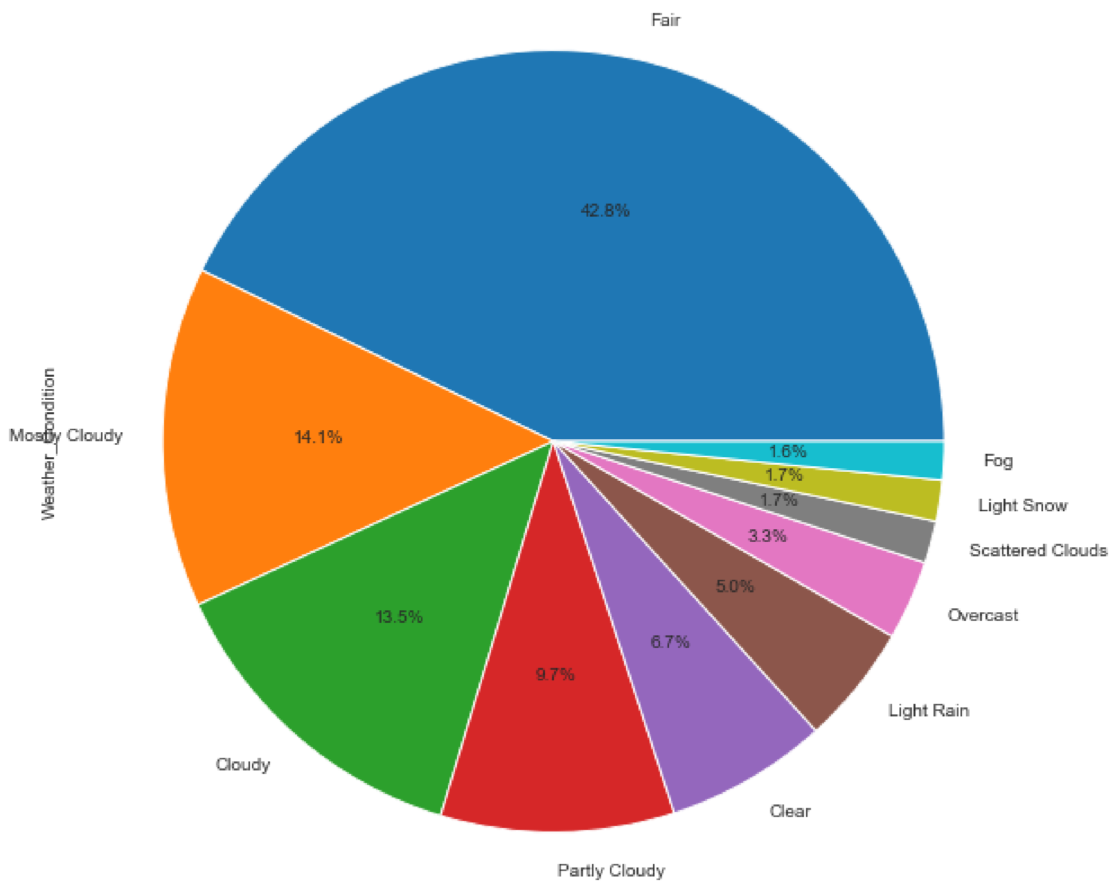
```
[71]: accident_by_weather_condition=df.Weather_Condition.value_counts()
      accident_by_weather_condition.head(50)
```

```
[71]: Fair                            1107194
      Mostly Cloudy                    363959
      Cloudy                           348767
      Partly Cloudy                    249939
      Clear                            173823
      Light Rain                       128403
      Overcast                          84882
      Scattered Clouds                  45132
      Light Snow                        43752
      Fog                               41226
      Haze                              36354
      Rain                              31044
      Fair / Windy                      15195
      Heavy Rain                        11824
      Smoke                              7200
      Light Drizzle                      7041
      Thunder in the Vicinity            6944
```

```
Cloudy / Windy                     6839
T-Storm                            6546
Mostly Cloudy / Windy              6297
Thunder                            6018
Snow                               5289
Light Rain with Thunder            5287
Partly Cloudy / Windy              3876
Wintry Mix                         3843
Heavy T-Storm                      3598
Light Rain / Windy                 3412
Light Snow / Windy                 2153
Drizzle                            1705
Heavy Snow                         1441
Rain / Windy                       1093
Light Thunderstorms and Rain       1089
N/A Precipitation                  1079
Patches of Fog                     1046
Mist                               1011
Thunderstorm                        985
Shallow Fog                         952
Light Freezing Rain                 900
Heavy Rain / Windy                  725
Showers in the Vicinity             650
Haze / Windy                        563
Heavy Thunderstorms and Rain        510
Thunderstorms and Rain              506
Snow / Windy                        401
Light Freezing Fog                  385
Heavy T-Storm / Windy               382
Light Freezing Drizzle              369
Fog / Windy                         270
T-Storm / Windy                     260
Thunder / Windy                     224
Name: Weather_Condition, dtype: int64
```

[72]:
```
accident_by_weather_condition.head(10).plot(kind="pie",autopct='%1.
 ↪1f%%',figsize=(10, 10))
```

[72]:
```
<AxesSubplot:ylabel='Weather_Condition'>
```



1.3 Ask & answer questions

1. Are there more accidents in which weather condition?

- Fair

2. which 5 states have the highest number of accidents?

- CA 795868
- FL 401388
- TX 149037
- OR 126341
- VA 113535

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

- Answered

4. what time of the day are accidents most frequent in ?

- mostly office time i.e. 6AM to 10AM and 3PM to 9PM

5. which day of week have most accidents?

- workdays

6. which months have the most accidents ?

- its varry but most accidents ocuurs in december .

7. what is the trend of accidnets year over year(decresing/incresing)?

- overall its constant for every year

1.4 Summary and Conclusion

Insights: - No data from New York - Less than 4% of cities have more than 1000 accident yearly - over 1110 cities have reported just 1 accident (need to check) - accident rate decreases exponentially - no Amenity present nearby any accident