US Accident analysis dataset

In [1]:

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

Download the dataset

In [2]:

```
data=pd.read_csv('US_Accidents_Dec21_updated.csv')
```

In [4]:

```
data
```

Loading [MathJax]/extensions/Safe.js

Out[4]:

	ID	Severity	Start_Time	End_Time	Start_Lat	Start_Lng	End_Lat	End_Lng	Distance(mi
0	A-1	3	2016-02-08 00:37:08	2016-02-08 06:37:08	40.108910	-83.092860	40.112060	-83.031870	3.230
1	A-2	2	2016-02-08 05:56:20	2016-02-08 11:56:20	39.865420	-84.062800	39.865010	-84.048730	0.747
2	A-3	2	2016-02-08 06:15:39	2016-02-08 12:15:39	39.102660	-84.524680	39.102090	-84.523960	0.055
3	A-4	2	2016-02-08 06:51:45	2016-02-08 12:51:45	41.062130	-81.537840	41.062170	-81.535470	0.123
4	A-5	3	2016-02-08 07:53:43	2016-02-08 13:53:43	39.172393	-84.492792	39.170476	-84.501798	0.500
...	...	...	...	...	...	...	...	...	..
2845337	A-2845338	2	2019-08-23 18:03:25	2019-08-23 18:32:01	34.002480	-117.379360	33.998880	-117.370940	0.543
2845338	A-2845339	2	2019-08-23 19:11:30	2019-08-23 19:38:23	32.766960	-117.148060	32.765550	-117.153630	0.338
2845339	A-2845340	2	2019-08-23 19:00:21	2019-08-23 19:28:49	33.775450	-117.847790	33.777400	-117.857270	0.561
2845340	A-2845341	2	2019-08-23 19:00:21	2019-08-23 19:29:42	33.992460	-118.403020	33.983110	-118.395650	0.772
2845341	A-2845342	2	2019-08-23 18:52:06	2019-08-23 19:21:31	34.133930	-117.230920	34.137360	-117.239340	0.537

2845342 rows × 47 columns

In [20]:

```
# Data preparation and cleaning-Look at some information about the data and the columns
#                             -Fix any missing or incorrect values
```

In [7]:

```
data.columns
```

Out[7]:

```
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')
```

In [50]:

```
#No of columns
umns)
```

Loading [MathJax]/extensions/Safe.js

Out[50]:
```
47
```

In [51]:
```python
#No of rows
len(data)
```

Out[51]:
```
2845342
```

In [15]:
```python
data.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)
             773.4+ MB
```

Loading [MathJax]/extensions/Safe.js

In [16]:

```
data.describe()
```

Out[16]:

	Severity	Start_Lat	Start_Lng	End_Lat	End_Lng	Distance(mi)	Number	T
count	2.845342e+06	2.845342e+06	2.845342e+06	2.845342e+06	2.845342e+06	2.845342e+06	1.101431e+06	
mean	2.137572e+00	3.624520e+01	-9.711463e+01	3.624532e+01	-9.711439e+01	7.026779e-01	8.089408e+03	
std	4.787216e-01	5.363797e+00	1.831782e+01	5.363873e+00	1.831763e+01	1.560361e+00	1.836009e+04	
min	1.000000e+00	2.456603e+01	-1.245481e+02	2.456601e+01	-1.245457e+02	0.000000e+00	0.000000e+00	
25%	2.000000e+00	3.344517e+01	-1.180331e+02	3.344628e+01	-1.180333e+02	5.200000e-02	1.270000e+03	
50%	2.000000e+00	3.609861e+01	-9.241808e+01	3.609799e+01	-9.241772e+01	2.440000e-01	4.007000e+03	
75%	2.000000e+00	4.016024e+01	-8.037243e+01	4.016105e+01	-8.037338e+01	7.640000e-01	9.567000e+03	
max	4.000000e+00	4.900058e+01	-6.711317e+01	4.907500e+01	-6.710924e+01	1.551860e+02	9.999997e+06	

Questions

1) Are there more accidents in warmer or colder areas? 2) Which 5 states have highest number of accidents? 3) Among the top 100 cities of accidents,which states do they belong to most frequently? 4) What time of the day have accidents occured more frequently? 5) Which days of the week has the most accidents? 6) Which month has the most accidents? 7) What is the trend of accidents year over year(increasing/decreasing) 8) Is the distribution of accidents by hour the same on weekends as on weekdays. 9) List out the top 10 cities having highest accidents? 10) Find the number of cases for each timezone. 11) Which year has the highest number of cases?

In [42]:

```
#No of numeric datas
numerics=['int16','int32','int64','float16','float32','float64']
numeric_data=data.select_dtypes(include=numerics)
len(numeric_data.columns)
```

Out[42]:

```
14
```

In [22]:

```
# Missing values
data.isna().sum().sort_values(ascending=False)
```

Loading [MathJax]/extensions/Safe.js

Out[22]:

```
Number                    1743911
Precipitation(in)          549458
Wind_Chill(F)              469643
Wind_Speed(mph)            157944
Wind_Direction              73775
Humidity(%)                 73092
Weather_Condition           70636
Visibility(mi)              70546
Temperature(F)              69274
Pressure(in)                59200
Weather_Timestamp           50736
Airport_Code                 9549
Timezone                     3659
Nautical_Twilight            2867
Civil_Twilight               2867
Sunrise_Sunset               2867
Astronomical_Twilight        2867
Zipcode                      1319
City                          137
Street                          2
Country                         0
Junction                        0
Start_Time                      0
End_Time                        0
Start_Lat                       0
Turning_Loop                    0
Traffic_Signal                  0
Traffic_Calming                 0
Stop                            0
Station                         0
Roundabout                      0
Railway                         0
No_Exit                         0
Crossing                        0
Give_Way                        0
Bump                            0
Amenity                         0
Start_Lng                       0
End_Lat                         0
End_Lng                         0
Distance(mi)                    0
Description                     0
Severity                        0
Side                            0
County                          0
State                           0
ID                              0
dtype: int64
```

In [45]:

```
#Find the % of missing values
missing_percentages=data.isna().sum().sort_values(ascending=False)/len(data)
missing_percentages
```

Out[45]:

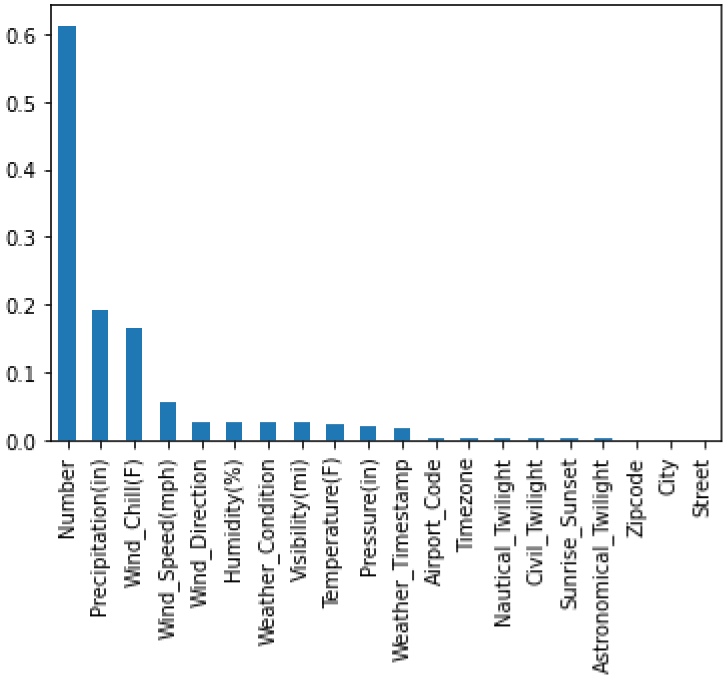
```
Wind_Direction           2.592834e-02
Weather_Condition        2.482514e-02
Weather_Timestamp        1.783125e-02
Airport_Code             3.356011e-03
Timezone                 1.285961e-03
Astronomical_Twilight    1.007612e-03
Sunrise_Sunset           1.007612e-03
Nautical_Twilight        1.007612e-03
Civil_Twilight           1.007612e-03
Zipcode                  4.635647e-04
City                     4.814887e-05
Street                   7.029032e-07
Railway                  0.000000e+00
Wind_Speed(mph)          0.000000e+00
No_Exit                  0.000000e+00
Junction                 0.000000e+00
Traffic_Calming          0.000000e+00
Give_Way                 0.000000e+00
Turning_Loop             0.000000e+00
Roundabout               0.000000e+00
Station                  0.000000e+00
Crossing                 0.000000e+00
Bump                     0.000000e+00
Amenity                  0.000000e+00
Stop                     0.000000e+00
Precipitation(in)        0.000000e+00
Traffic_Signal           0.000000e+00
ID                       0.000000e+00
Visibility(mi)           0.000000e+00
Description              0.000000e+00
Start_Time               0.000000e+00
End_Time                 0.000000e+00
Start_Lat                0.000000e+00
Start_Lng                0.000000e+00
End_Lat                  0.000000e+00
End_Lng                  0.000000e+00
Distance(mi)             0.000000e+00
Number                   0.000000e+00
Pressure(in)             0.000000e+00
Side                     0.000000e+00
County                   0.000000e+00
State                    0.000000e+00
Country                  0.000000e+00
Temperature(F)           0.000000e+00
Wind_Chill(F)            0.000000e+00
Severity                 0.000000e+00
Humidity(%)              0.000000e+00
dtype: float64
```

In [32]:

```python
#plotting bar chart for missing percentage
missing_percentages[missing_percentages!=0].plot(kind='bar')
```

Out[32]:

```
<AxesSubplot:>
```



In [52]:

```
#handling the missing values
data= data.fillna(missing_percentages.median())
data
```

Out[52]:

	ID	Severity	Start_Time	End_Time	Start_Lat	Start_Lng	End_Lat	End_Lng	Distance(mi
0	A-1	3	2016-02-08 00:37:08	2016-02-08 06:37:08	40.108910	-83.092860	40.112060	-83.031870	3.23
1	A-2	2	2016-02-08 05:56:20	2016-02-08 11:56:20	39.865420	-84.062800	39.865010	-84.048730	0.74
2	A-3	2	2016-02-08 06:15:39	2016-02-08 12:15:39	39.102660	-84.524680	39.102090	-84.523960	0.05
3	A-4	2	2016-02-08 06:51:45	2016-02-08 12:51:45	41.062130	-81.537840	41.062170	-81.535470	0.12
4	A-5	3	2016-02-08 07:53:43	2016-02-08 13:53:43	39.172393	-84.492792	39.170476	-84.501798	0.50
...	...	...	...	...	...	...	...	...	..
2845337	A-2845338	2	2019-08-23 18:03:25	2019-08-23 18:32:01	34.002480	-117.379360	33.998880	-117.370940	0.54
2845338	A-2845339	2	2019-08-23 19:11:30	2019-08-23 19:38:23	32.766960	-117.148060	32.765550	-117.153630	0.33
2845339	A-2845340	2	2019-08-23 19:00:21	2019-08-23 19:28:49	33.775450	-117.847790	33.777400	-117.857270	0.56
2845340	A-2845341	2	2019-08-23 19:00:21	2019-08-23 19:29:42	33.992460	-118.403020	33.983110	-118.395650	0.77
2845341	A-2845342	2	2019-08-23 18:52:06	2019-08-23 19:21:31	34.133930	-117.230920	34.137360	-117.239340	0.53

2845342 rows × 47 columns

In [49]:

```
data.isnull().sum()
```

Out[49]:

```
ID                       0
Severity                 0
Start_Time               0
End_Time                 0
Start_Lat                0
Start_Lng                0
End_Lat                  0
End_Lng                  0
Distance(mi)             0
Description              0
Number                   0
Street                   0
Side                     0
City                     0
County                   0
State                    0
Zipcode                  0
Country                  0
Timezone                 0
Airport_Code             0
Weather_Timestamp        0
Temperature(F)           0
Wind_Chill(F)            0
Humidity(%)              0
Pressure(in)             0
Visibility(mi)           0
Wind_Direction           0
Wind_Speed(mph)          0
Precipitation(in)        0
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
```

In [ ]:

```
# Since there is a whole lot of columns we will be focussing majorly on the below mentio
columns we will analyse
1)city
2)start time
3)start lat,start lng
4)temperature
5)weather condition
6)timezone
7)street
```

In [42]:

```
data.City.nunique()
```

Out[42]:

```
11681
```

Loading [MathJax]/extensions/Safe.js

```
ique()
```

Out[43]:
```
array(['Dublin', 'Dayton', 'Cincinnati', ..., 'Clarksdale', 'Bridgeboro',
       'American Fork-Pleasant Grove'], dtype=object)
```

In [53]:
```
#accidents according to time zone
timezone_df = pd.DataFrame(data['Timezone'].value_counts()).reset_index().rename(columns
timezone_df
```

Out[53]:

	Timezone	Cases
0	US/Eastern	1221927
1	US/Pacific	967094
2	US/Central	488065
3	US/Mountain	164597
4	0.0	3659

In [ ]:
```
#observation: US/Eastern has the highest number of accidents
```

In [54]:
```
#count of accident for each street
street_df = pd.DataFrame(data['Street'].value_counts()).reset_index().rename(columns={'i
street_df
```

Out[54]:

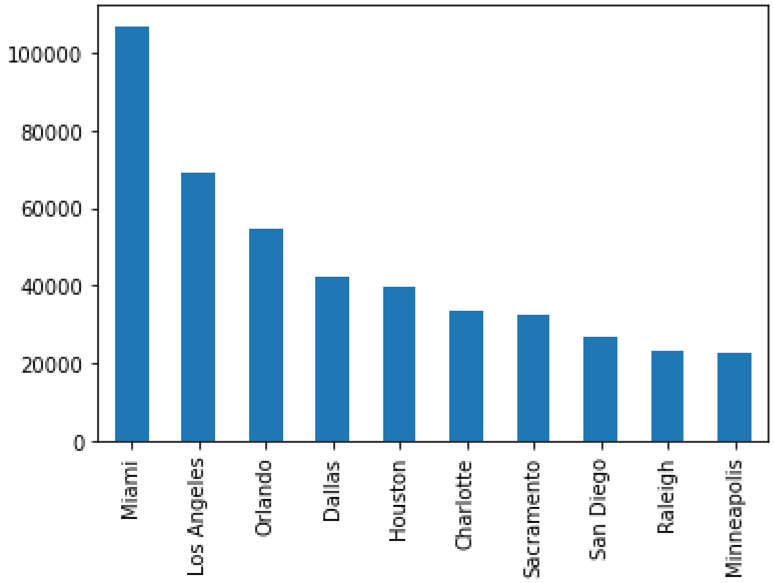
	Street No.	Cases
0	I-95 N	39853
1	I-5 N	39402
2	I-95 S	36425
3	I-5 S	30229
4	I-10 E	26164
...	...	...
159647	Villareal Dr	1
159648	La Jacaranda	1
159649	SW 273rd Ln	1
159650	Silent Brook Dr	1
159651	473-401 Cutoff Rd	1

159652 rows × 2 columns

In [ ]:
```
#Observation: In  5 years (2016-2021) Street No. I-95 N is having the highest road accid
```

In [55]:
```
cities_by_accident=data.City.value_counts()
cities_by_accident[:10].plot.bar()
```

Out[55]:
```
<AxesSubplot:>
```



In [ ]:

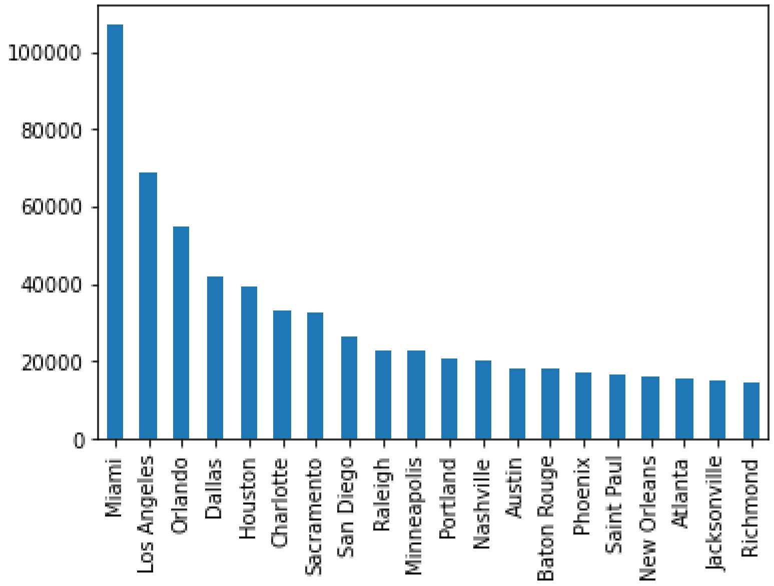
```
#Observation: Miami is the city with the highest no. of road accidents in US (2016-2021)
```

In [56]:

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

Out[56]:

```
<AxesSubplot:>
```



In [57]:

```
#accidents vs year
year_df = pd.DataFrame(data.Start_Time.dt.year.value_counts()).reset_index().rename(colu
year_df
```

Out[57]:

	Year	Cases
5	2016	122024
4	2018	163176
3	2017	163918
2	2019	258615
1	2020	625864
0	2021	1511745

In [ ]:

```
#observation: Highest number of accidents happened in 2021
```

Loading [MathJax]/extensions/Safe.js

In [58]:

```
weather_condition_df = pd.DataFrame(data.Weather_Condition.value_counts().head(10)).rese
weather_condition_df
```

Out[58]:

	Weather_Condition	Cases
0	Fair	1107194
1	Mostly Cloudy	363959
2	Cloudy	348767
3	Partly Cloudy	249939
4	Clear	173823
5	Light Rain	128403
6	Overcast	84882
7	0.0	70636
8	Scattered Clouds	45132
9	Light Snow	43752

In [77]:

```
high_accident_cities=cities_by_accident[cities_by_accident>=10000]
low_accident_cities=cities_by_accident[cities_by_accident<1000]
highest_accident=len(high_accident_cities)/len(data)
lowest_accident=len(low_accident_cities)/len(data)
highest_accident,lowest_accident
```

Out[77]:

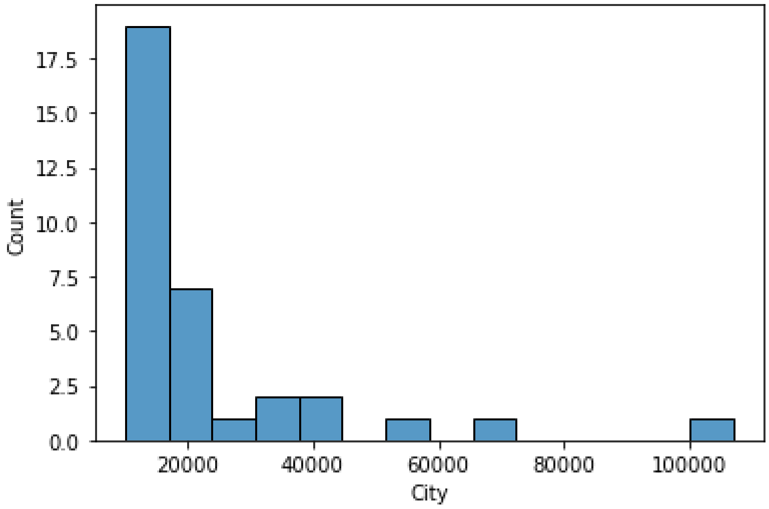
```
(1.1949354418554958e-05, 0.003931337603704581)
```

In [75]:

```
sns.histplot(high_accident_cities)
```

Out[75]:

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



In [ ]:

```
#Observation: Most cities have less than 20000 accidents
```

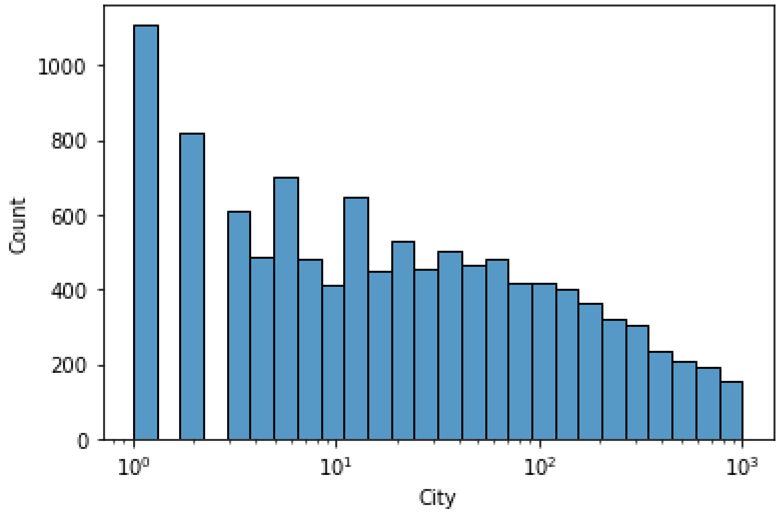
In [62]:

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

Out[62]:

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

Loading [MathJax]/extensions/Safe.js



In [63]:

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

Out[63]:

```
Avera                               1
Clifford Township                   1
White Castle                        1
Russian River-Sonoma Coast          1
Underhill                           1
                                   ..
Horntown                            1
Crooksville                         1
Novinger                            1
Trementina                          1
American Fork-Pleasant Grove        1
Name: City, Length: 1110, dtype: int64
```

In [64]:

```
#start time
data.Start_Time
```

Out[64]:

```
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]
```

In [65]:

```
#Convert datatype
data.Start_Time=pd.to_datetime(data.Start_Time)
```

In [66]:

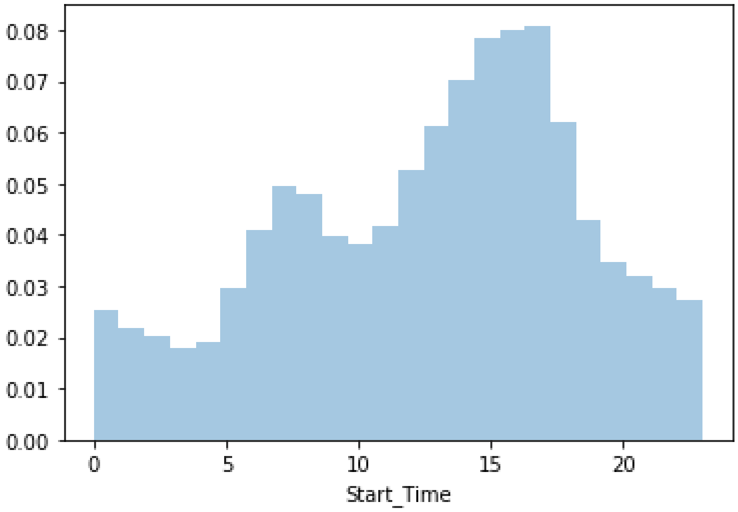
```
sns.distplot(data.Start_Time.dt.hour,bins=24,norm_hist=True,kde=False)
```

```
C:\anaconda\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` i
s 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)
```

Out[66]:

```
<AxesSubplot:xlabel='Start_Time'>
```

Loading [MathJax]/extensions/Safe.js



In [ ]:

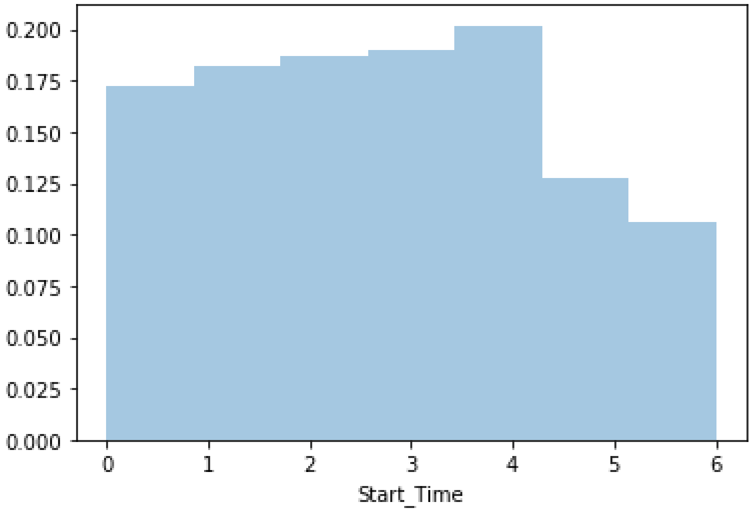
```
#Observation:Most accidents happen between 2-7 pm
```

In [71]:

```
sns.distplot(data.Start_Time.dt.dayofweek,bins=7,kde=False,norm_hist=True)
```

Out[71]:

```
<AxesSubplot:xlabel='Start_Time'>
```



In [ ]:

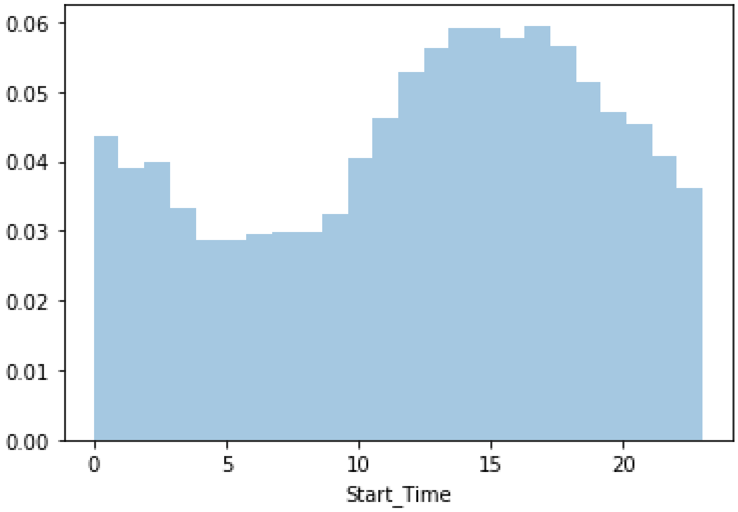
```
#observation: On weekends the number of accidents is low
```

In [70]:

```
sundays_start_time=data.Start_Time[data.Start_Time.dt.dayofweek==6]
sns.distplot(sundays_start_time.dt.hour,bins=24,kde=False,norm_hist=True)
```

Out[70]:

```
<AxesSubplot:xlabel='Start_Time'>
```



In [ ]:

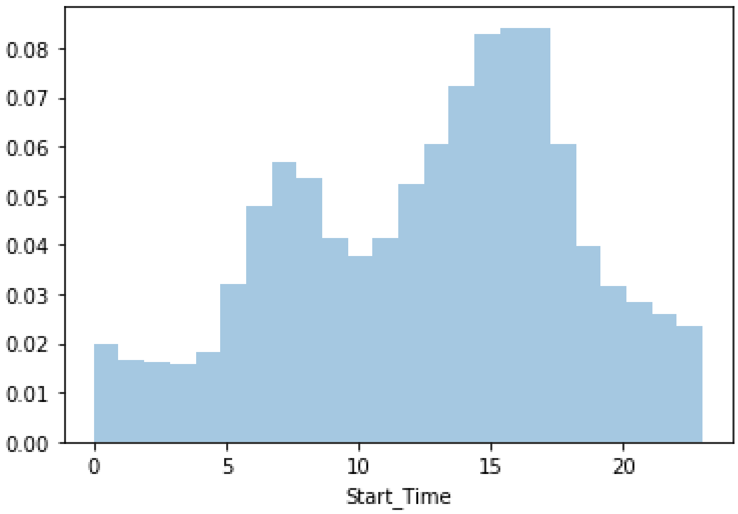
```
#observation: During weekend max accidents occur during 11am-12pm
```

In [68]:

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

Out[68]:

```
<AxesSubplot:xlabel='Start_Time'>
```



In [ ]:

```
#observation: During weekend max accidents occur during 2pm-7pm
```

In [ ]:

```
#Summary:
-Less than 2% of the cities have accident count more than 10000
-Over 1100 cities had just 1 accident.
-The number of accidents per city decreases/increases exponentially
-Most accidents happen between 2-7 pm(genereal obs)
-second highest % of accidents lie between 6am-2pm
-On weekends the number of accidents is low
-weekdays has different trend when compared to weekends
-More accidents occur during the end of the year
-During weekend max accidents occur during 2pm-7pm
-During weekend max accidents occur during 11am-12pm
```

END

Loading [MathJax]/extensions/Safe.js