US Accidents Exploratory Data Analysis

TODO - talk about EDA

TODO - talk about the dataset (source, what it contains, how it will be useful)

- Kaggle
- informaiton about accidents
- can use useful to prevent accidents
- mention that this does not contain data about New York

In [2]:

```
import opendatasets as od

download_url = 'https://www.kaggle.com/datasets/sobhanmoosavi/us-accidents'

od.download(download_url)
```

```
Skipping, found downloaded files in ".\us-accidents" (use force=True to force down
load)
```

In [3]:

```
data_filename = './us-accidents/US_Accidents_March23.csv'
```

Data Preparation and Cleaning

1.Load the file using Pandas

2.Look at some information about the data & the columns

3.Fix any missing or incorrect values

In [4]:

```
import pandas as pd
```

In [5]:

```
df = pd.read_csv(data_filename)
```

In [6]:

```
df
```

Out[6]:

	ID	Source	Severity	Start_Time	End_Time	Start_Lat	Start_Lng	End_Lat	E
0	A-1	Source2	3	2016-02-08 05:46:00	2016-02-08 11:00:00	39.865147	-84.058723	NaN	
1	A-2	Source2	2	2016-02-08 06:07:59	2016-02-08 06:37:59	39.928059	-82.831184	NaN	
2	A-3	Source2	2	2016-02-08 06:49:27	2016-02-08 07:19:27	39.063148	-84.032608	NaN	
3	A-4	Source2	3	2016-02-08 07:23:34	2016-02-08 07:53:34	39.747753	-84.205582	NaN	
4	A-5	Source2	2	2016-02-08 07:39:07	2016-02-08 08:09:07	39.627781	-84.188354	NaN	
...	...	...	...	...	...	...	...	...	
7728389	A-7777757	Source1	2	2019-08-23 18:03:25	2019-08-23 18:32:01	34.002480	-117.379360	33.99888	-11
7728390	A-7777758	Source1	2	2019-08-23 19:11:30	2019-08-23 19:38:23	32.766960	-117.148060	32.76555	-11
7728391	A-7777759	Source1	2	2019-08-23 19:00:21	2019-08-23 19:28:49	33.775450	-117.847790	33.77740	-11
7728392	A-7777760	Source1	2	2019-08-23 19:00:21	2019-08-23 19:29:42	33.992460	-118.403020	33.98311	-11
7728393	A-7777761	Source1	2	2019-08-23 18:52:06	2019-08-23 19:21:31	34.133930	-117.230920	34.13736	-11

7728394 rows × 46 columns

In [7]:

```
df.columns
```

Out[7]:

```
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')
```

In [8]:

```
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
```

In [9]:

```
df.describe()
```

Out[9]:

	Severity	Start_Lat	Start_Lng	End_Lat	End_Lng	Distance(mi)	Te
count	7.728394e+06	7.728394e+06	7.728394e+06	4.325632e+06	4.325632e+06	7.728394e+06	
mean	2.212384e+00	3.620119e+01	-9.470255e+01	3.626183e+01	-9.572557e+01	5.618423e-01	
std	4.875313e-01	5.076079e+00	1.739176e+01	5.272905e+00	1.810793e+01	1.776811e+00	
min	1.000000e+00	2.455480e+01	-1.246238e+02	2.456601e+01	-1.245457e+02	0.000000e+00	-8
25%	2.000000e+00	3.339963e+01	-1.172194e+02	3.346207e+01	-1.177543e+02	0.000000e+00	
50%	2.000000e+00	3.582397e+01	-8.776662e+01	3.618349e+01	-8.802789e+01	3.000000e-02	
75%	2.000000e+00	4.008496e+01	-8.035368e+01	4.017892e+01	-8.024709e+01	4.640000e-01	
max	4.000000e+00	4.900220e+01	-6.711317e+01	4.907500e+01	-6.710924e+01	4.417500e+02	

In [10]:
```
df.describe(include ='object')
```

Out[10]:

	ID	Source	Start_Time	End_Time	Description	Street	City	County	Stat
count	7728394	7728394	7728394	7728394	7728389	7717525	7728141	7728394	772839
unique	7728394	3	6131796	6705355	3761578	336306	13678	1871	4
top	A-1	Source1	2021-01-26 16:16:13	2021-11-22 08:00:00	A crash has occurred causing no to minimum del...	I-95 N	Miami	Los Angeles	C
freq	1	4325632	225	112	9593	78430	186917	526851	174143

In [11]:
```
df.shape
```

Out[11]:
```
(7728394, 46)
```

In [12]:
```
numerics = ['int16', 'int32', 'int64', 'float16', 'float32', 'float64']

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

Out[12]:
```
13
```

In [13]:
```
missing_percentage = df.isna().sum().sort_values(ascending= False)/len(df)
missing_percentage
```

Out[13]:

```
End_Lat                  4.402935e-01
End_Lng                  4.402935e-01
Precipitation(in)        2.851286e-01
Wind_Chill(F)            2.586590e-01
Wind_Speed(mph)          7.391355e-02
Visibility(mi)           2.291524e-02
Wind_Direction           2.267043e-02
Humidity(%)              2.253301e-02
Weather_Condition        2.244438e-02
Temperature(F)           2.120143e-02
Pressure(in)             1.820288e-02
Weather_Timestamp        1.555666e-02
Nautical_Twilight        3.007869e-03
Civil_Twilight           3.007869e-03
Sunrise_Sunset           3.007869e-03
Astronomical_Twilight    3.007869e-03
Airport_Code             2.928810e-03
Street                   1.406372e-03
Timezone                 1.010300e-03
Zipcode                  2.477876e-04
City                     3.273643e-05
Description              6.469649e-07
Traffic_Signal           0.000000e+00
Roundabout               0.000000e+00
Station                  0.000000e+00
Stop                     0.000000e+00
Traffic_Calming          0.000000e+00
Country                  0.000000e+00
Turning_Loop             0.000000e+00
No_Exit                  0.000000e+00
End_Time                 0.000000e+00
Start_Time               0.000000e+00
Severity                 0.000000e+00
Railway                  0.000000e+00
Crossing                 0.000000e+00
Junction                 0.000000e+00
Give_Way                 0.000000e+00
Bump                     0.000000e+00
Amenity                  0.000000e+00
Start_Lat                0.000000e+00
Start_Lng                0.000000e+00
Distance(mi)             0.000000e+00
Source                   0.000000e+00
County                   0.000000e+00
State                    0.000000e+00
ID                       0.000000e+00
dtype: float64
```

In [14]:

```
missing_percentage[missing_percentage != 0]
```

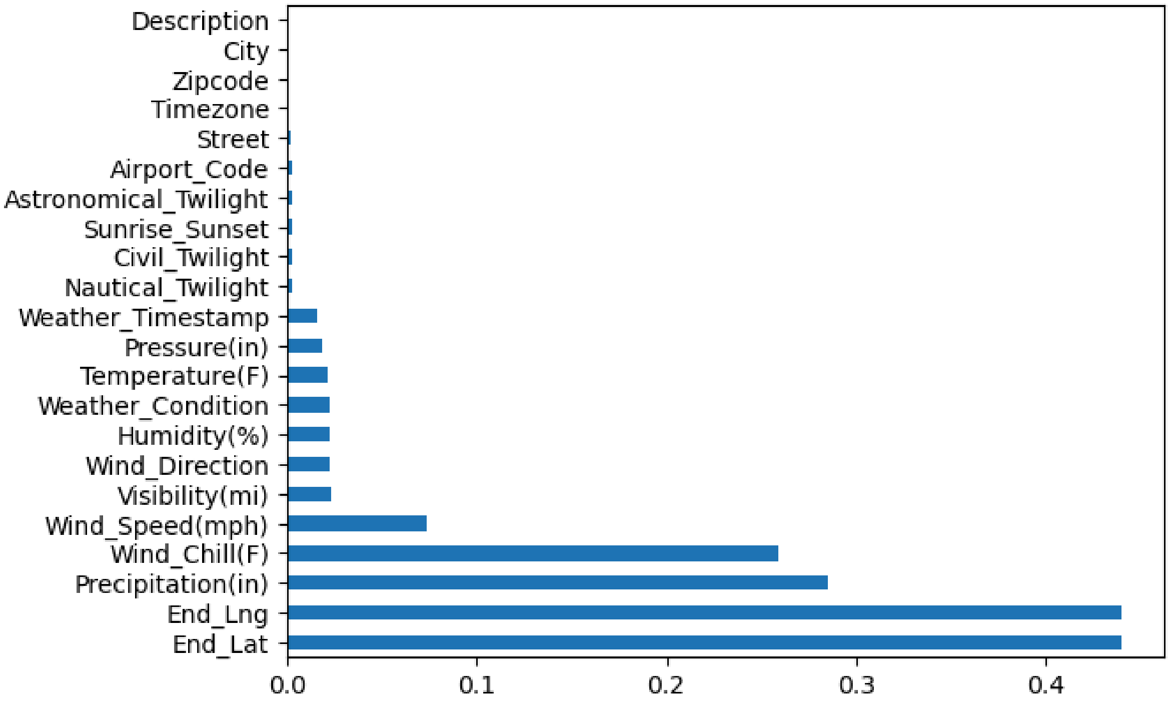
Out[14]:
```
End_Lat                  4.402935e-01
End_Lng                  4.402935e-01
Precipitation(in)        2.851286e-01
Wind_Chill(F)            2.586590e-01
Wind_Speed(mph)          7.391355e-02
Visibility(mi)           2.291524e-02
Wind_Direction           2.267043e-02
Humidity(%)              2.253301e-02
Weather_Condition        2.244438e-02
Temperature(F)           2.120143e-02
Pressure(in)             1.820288e-02
Weather_Timestamp        1.555666e-02
Nautical_Twilight        3.007869e-03
Civil_Twilight           3.007869e-03
Sunrise_Sunset           3.007869e-03
Astronomical_Twilight    3.007869e-03
Airport_Code             2.928810e-03
Street                   1.406372e-03
Timezone                 1.010300e-03
Zipcode                  2.477876e-04
City                     3.273643e-05
Description              6.469649e-07
dtype: float64
```

In [15]:
```
type(missing_percentage)
```

Out[15]:
```
pandas.core.series.Series
```

In [16]:
```
missing_percentage[missing_percentage != 0].plot(kind='barh')
```

Out[16]:
```
<Axes: >
```



- Remove columns that you don't want to use

Exploratory Data Analysis and Visulization

Columns we'll analyze

- City
- State
- Start Time
- Start Lat
- Start Lag
- Temprature
- Weather Condition

In [17]:
```
df.columns
```

Out[17]:
```
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')
```

Analysis for City

In [18]:
```
df.City
```

Out[18]:
```
0                 Dayton
1           Reynoldsburg
2           Williamsburg
3                 Dayton
4                 Dayton
               ...
7728389        Riverside
7728390        San Diego
7728391           Orange
7728392      Culver City
7728393         Highland
Name: City, Length: 7728394, dtype: object
```

In [19]:
```
df.City.unique()
```

Out[19]:
```
array(['Dayton', 'Reynoldsburg', 'Williamsburg', ..., 'Ness City',
       'Clarksdale', 'American Fork-Pleasant Grove'], dtype=object)
```

In [20]:
```
Unique_cities = df.City.unique()
len(Unique_cities)
```

Out[20]:
```
13679
```

In [21]:
```
cities_by_accident = df.City.value_counts()
cities_by_accident
```

```
Out[21]: Miami                               186917
         Houston                             169609
         Los Angeles                         156491
         Charlotte                           138652
         Dallas                              130939
                                              ...
         Benkelman                                1
         Old Appleton                             1
         Wildrose                                 1
         Mc Nabb                                  1
         American Fork-Pleasant Grove             1
         Name: City, Length: 13678, dtype: int64
```

```
In [22]: 'New York' in df.City
```

```
Out[22]: False
```

```
In [23]: cities_by_accident[:20]
```

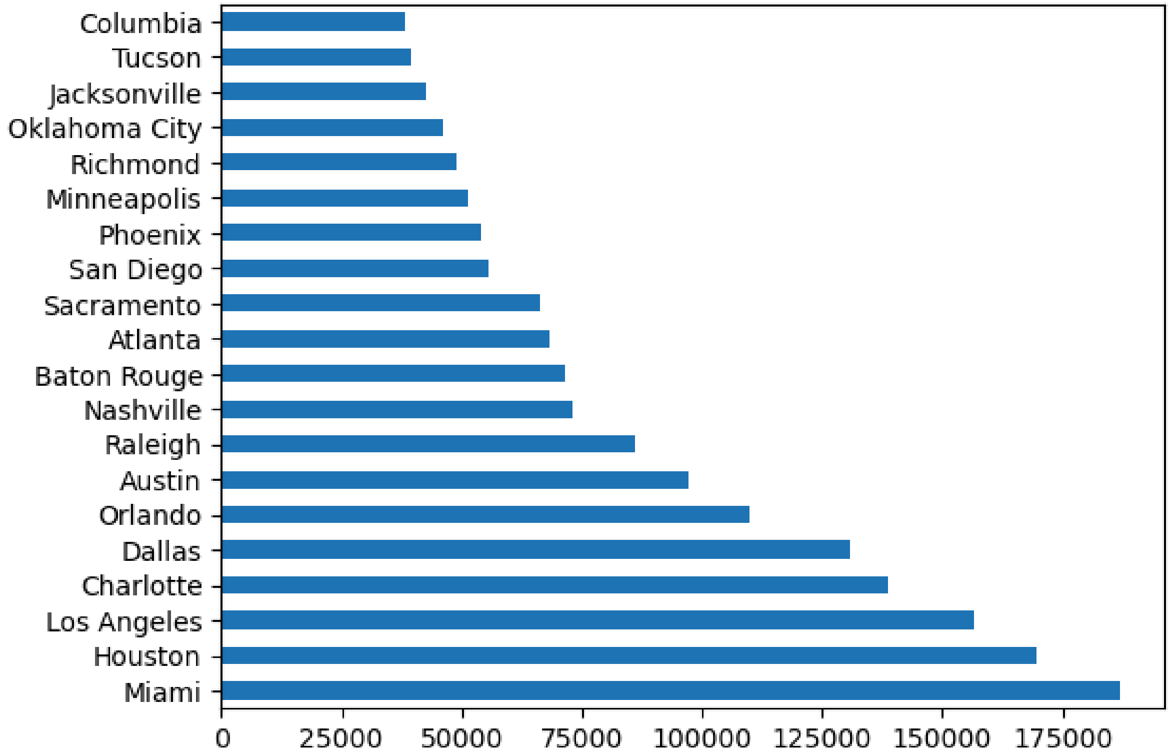
```
Out[23]: Miami             186917
         Houston           169609
         Los Angeles       156491
         Charlotte         138652
         Dallas            130939
         Orlando           109733
         Austin             97359
         Raleigh            86079
         Nashville          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: City, dtype: int64
```

```
In [24]: type(cities_by_accident)
```

```
Out[24]: pandas.core.series.Series
```

```
In [25]: cities_by_accident[:20].plot(kind = 'barh')
```

```
Out[25]: <Axes: >
```



In [26]:

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

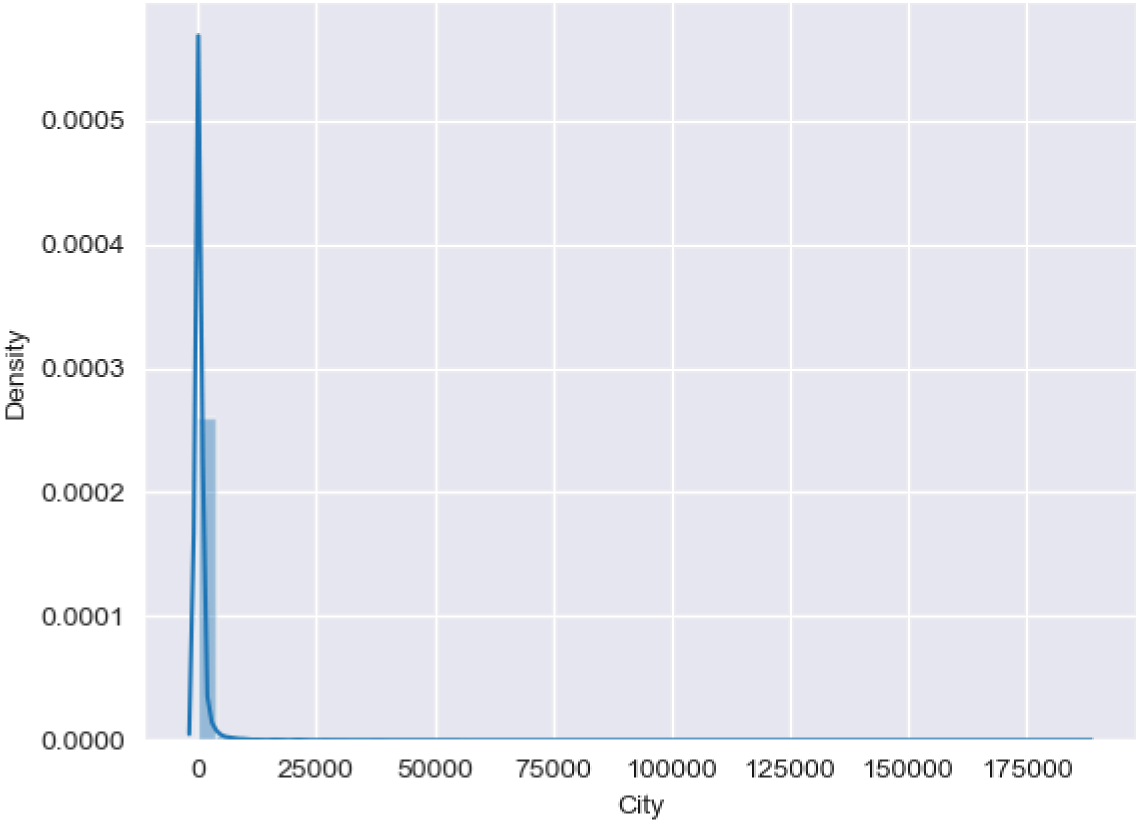
In [27]:

```
sns.distplot(cities_by_accident)
```

```
C:\Users\Atul Gupta\AppData\Local\Temp\ipykernel_11296\3405282844.py:1: UserWarnin
g:

`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)
```

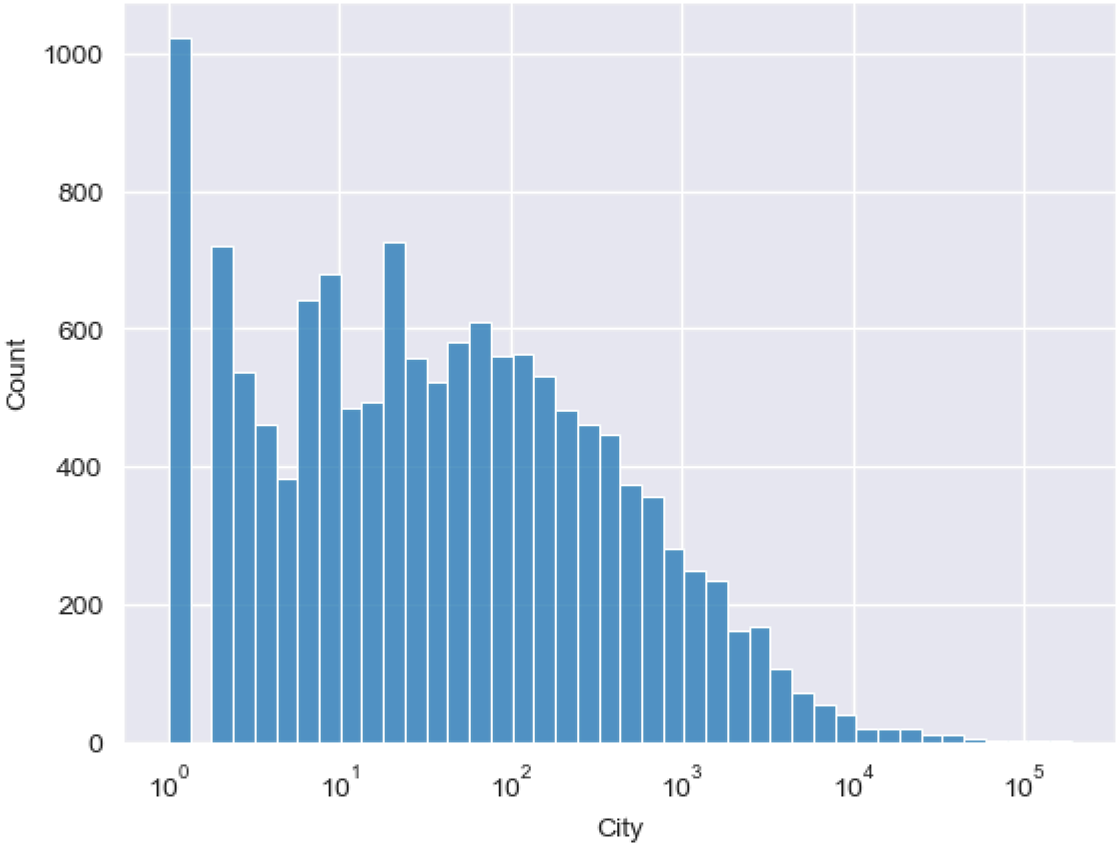
Out[27]:

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



In [28]:
```
# log_scale = True plot is more clear visible no of accident in cities
sns.histplot(cities_by_accident, log_scale = True)
```

Out[28]:
```
<Axes: xlabel='City', ylabel='Count'>
```



In [29]:
```
cities_by_accident[cities_by_accident == 1]
```

Out[29]:

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

In [30]:

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

In [31]:

```
len(high_accident_cities)/ len(cities_by_accident)
```

Out[31]:

```
0.08904810644831115
```

- 8.9% cities high accidents

In [32]:

```
len(low_accident_cities)/len(cities_by_accident)
```

Out[32]:

```
0.9109518935516888
```

- 91% cities low accidents

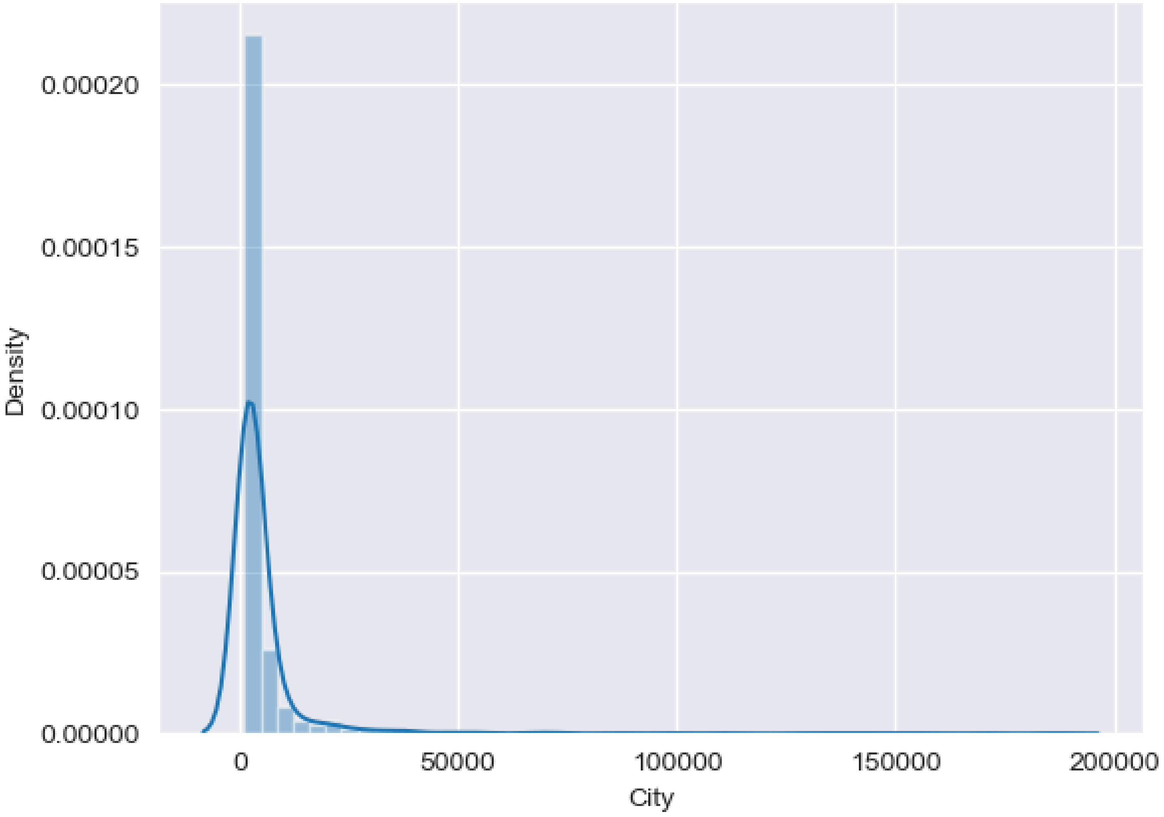
In [33]:

```
sns.distplot(high_accident_cities)
```

```
C:\Users\Atul Gupta\AppData\Local\Temp\ipykernel_11296\2843252471.py:1: UserWarnin
g:

`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(high_accident_cities)
```

Out[33]:

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



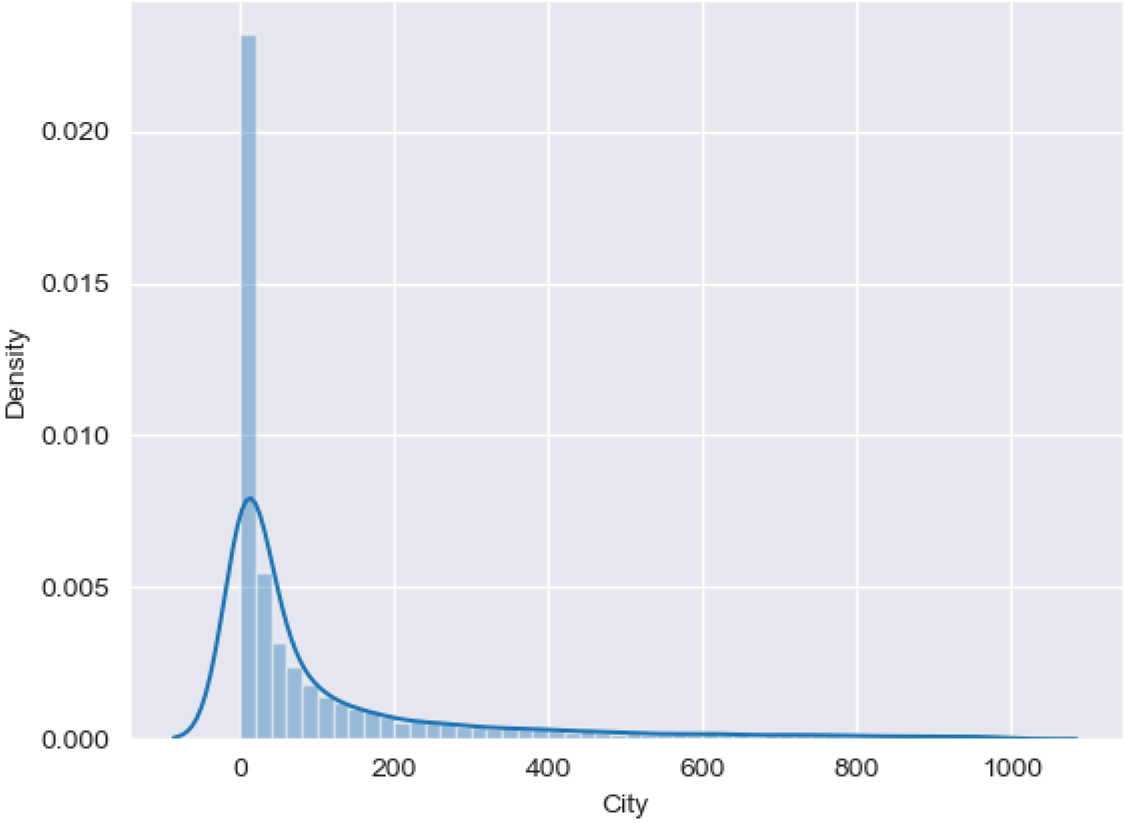
In [34]:

```
sns.distplot(low_accident_cities)
```

```
C:\Users\Atul Gupta\AppData\Local\Temp\ipykernel_11296\469555131.py:1: UserWarnin
g: 

`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(low_accident_cities)
```

Out[34]:

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



Analysis for Start Time

In [35]:
```
df.Start_Time
```

Out[35]:
```
0          2016-02-08 05:46:00
1          2016-02-08 06:07:59
2          2016-02-08 06:49:27
3          2016-02-08 07:23:34
4          2016-02-08 07:39:07
                  ...
7728389    2019-08-23 18:03:25
7728390    2019-08-23 19:11:30
7728391    2019-08-23 19:00:21
7728392    2019-08-23 19:00:21
7728393    2019-08-23 18:52:06
Name: Start_Time, Length: 7728394, dtype: object
```

In [36]:
```
df.Start_Time[0]
```

Out[36]:
```
'2016-02-08 05:46:00'
```

In [37]:
```
df['Start_Time'] = pd.to_datetime(df['Start_Time'])
```

In [38]:
```
df.dtypes
```

Out[38]:

```
ID                                object
Source                            object
Severity                           int64
Start_Time                datetime64[ns]
End_Time                          object
Start_Lat                        float64
Start_Lng                        float64
End_Lat                          float64
End_Lng                          float64
Distance(mi)                     float64
Description                       object
Street                            object
City                              object
County                            object
State                             object
Zipcode                           object
Country                           object
Timezone                          object
Airport_Code                      object
Weather_Timestamp                 object
Temperature(F)                   float64
Wind_Chill(F)                    float64
Humidity(%)                      float64
Pressure(in)                     float64
Visibility(mi)                   float64
Wind_Direction                    object
Wind_Speed(mph)                  float64
Precipitation(in)                float64
Weather_Condition                 object
Amenity                             bool
Bump                                bool
Crossing                            bool
Give_Way                            bool
Junction                            bool
No_Exit                             bool
Railway                             bool
Roundabout                          bool
Station                             bool
Stop                                bool
Traffic_Calming                     bool
Traffic_Signal                      bool
Turning_Loop                        bool
Sunrise_Sunset                    object
Civil_Twilight                    object
Nautical_Twilight                 object
Astronomical_Twilight             object
dtype: object
```

In [39]:

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

Out[39]:

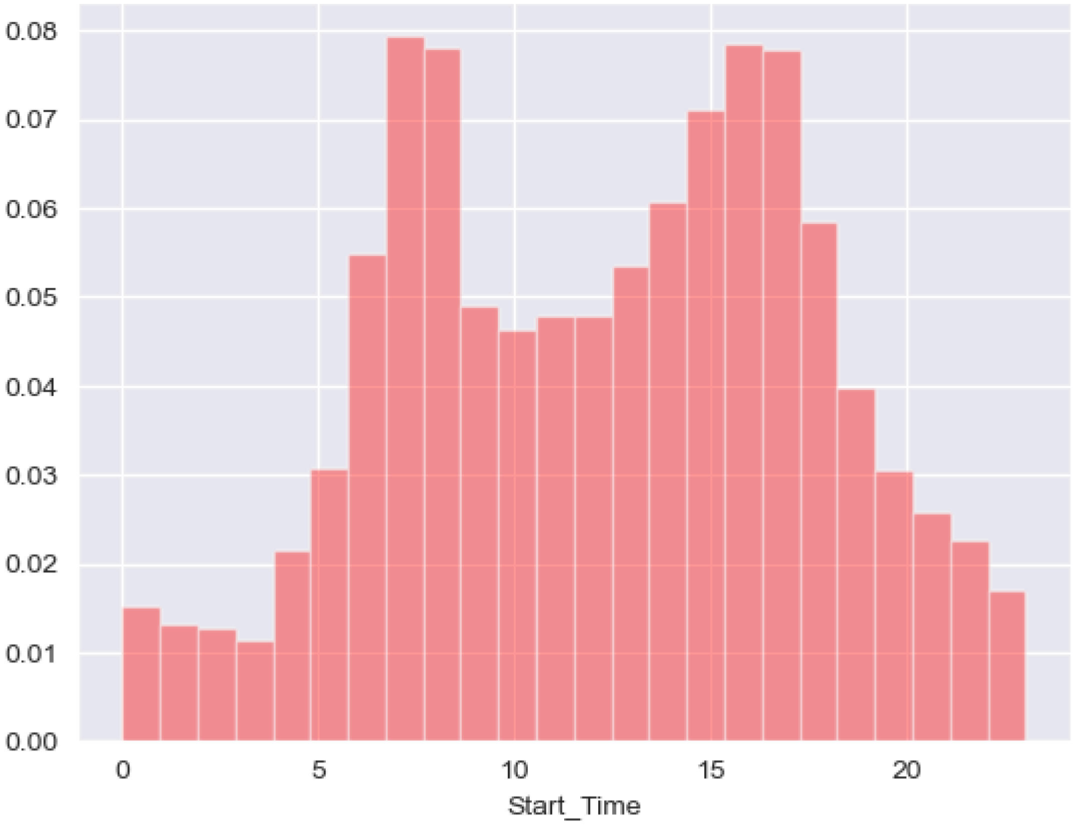
```
Timestamp('2016-02-08 05:46:00')
```

In [40]:

```
#(norm_hist = True use to convert into percentage)
sns.distplot(df.Start_Time.dt.hour, bins = 24, kde = False,norm_hist = True , color
C:\Users\Atul Gupta\AppData\Local\Temp\ipykernel_11296\823610032.py:2: UserWarnin
g:

`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(df.Start_Time.dt.hour, bins = 24, kde = False,norm_hist = True , co
lor = 'red')
```

Out[40]:

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



What time of the day are accidents most frequent in?

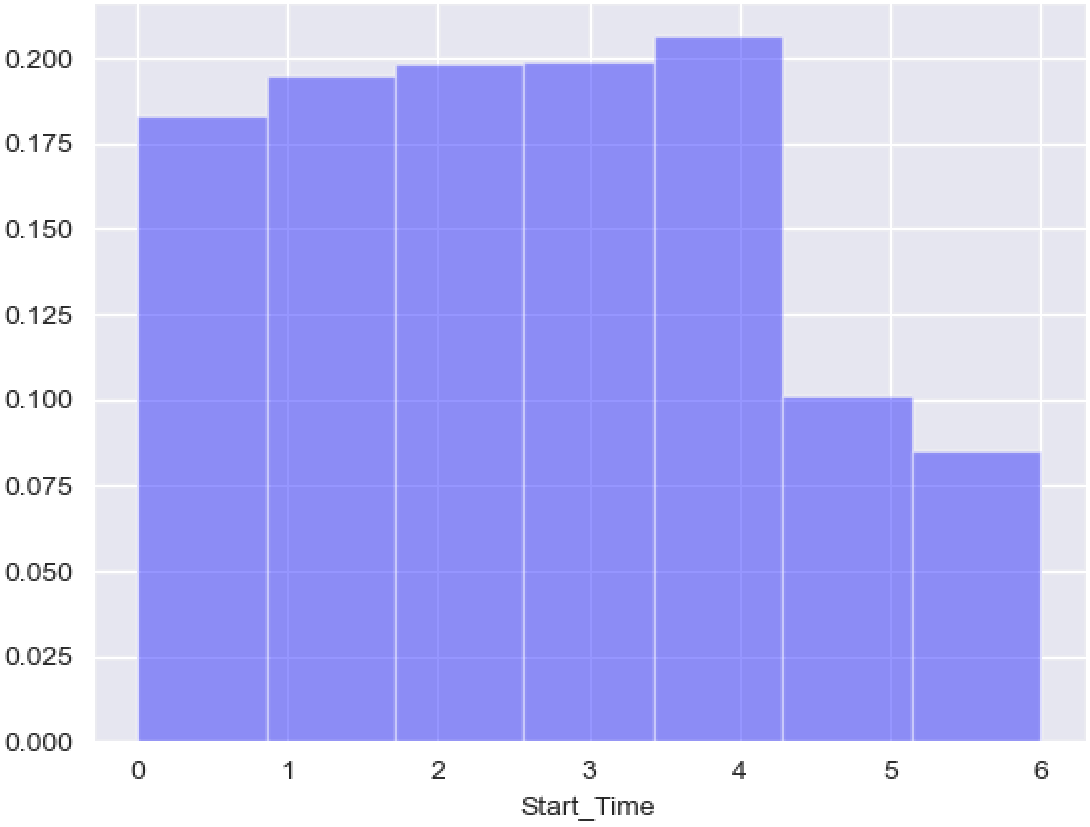
- A high percentage of accidents occur between 7 am to 8 am (probably people in a hurry to get to work)
- Next higest percentage is 3 pm to 5 pm.

In [41]:

```
sns.distplot(df.Start_Time.dt.dayofweek, bins = 7, kde = False,norm_hist = True, c
C:\Users\Atul Gupta\AppData\Local\Temp\ipykernel_11296\530816872.py:1: UserWarnin
g:

`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(df.Start_Time.dt.dayofweek, bins = 7, kde = False,norm_hist = True,
color = 'blue')
```

Out[41]:

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



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

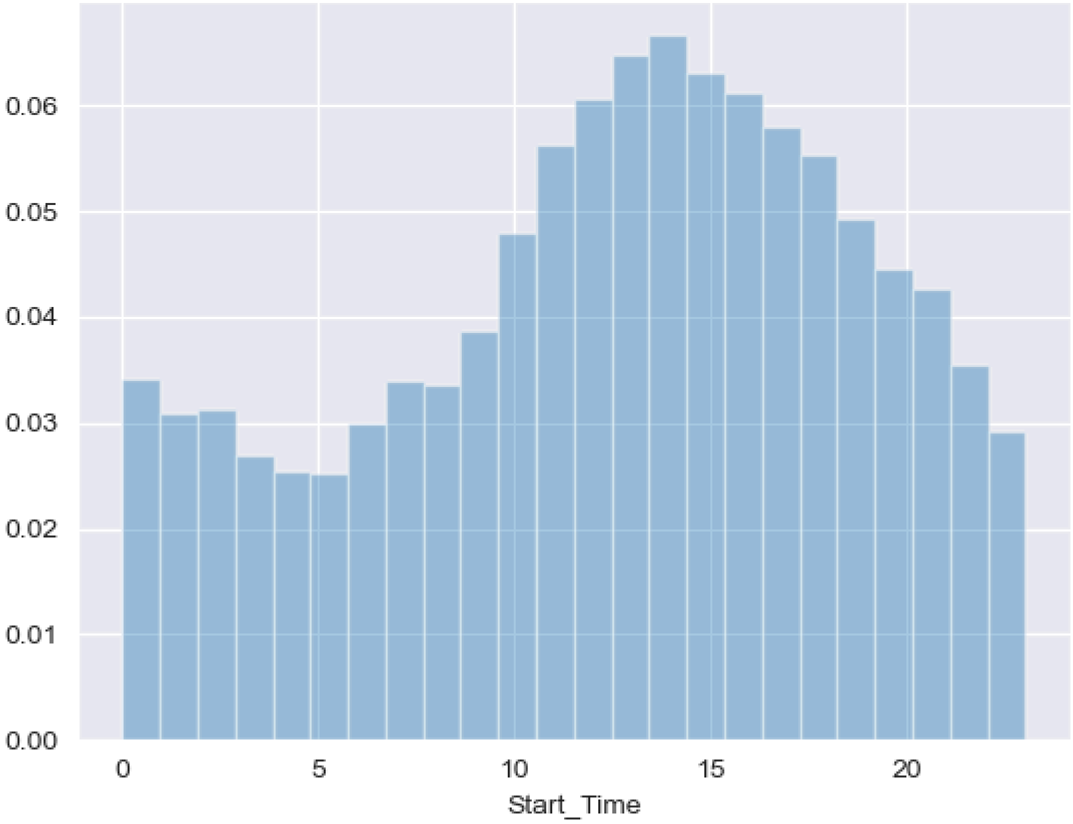
In [42]:

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

```
C:\Users\Atul Gupta\AppData\Local\Temp\ipykernel_11296\2255068480.py:3: UserWarnin
g:

`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(sundays_start_time.dt.hour, bins = 24, kde = False, norm_hist = Tru
e)
```

Out[42]:

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



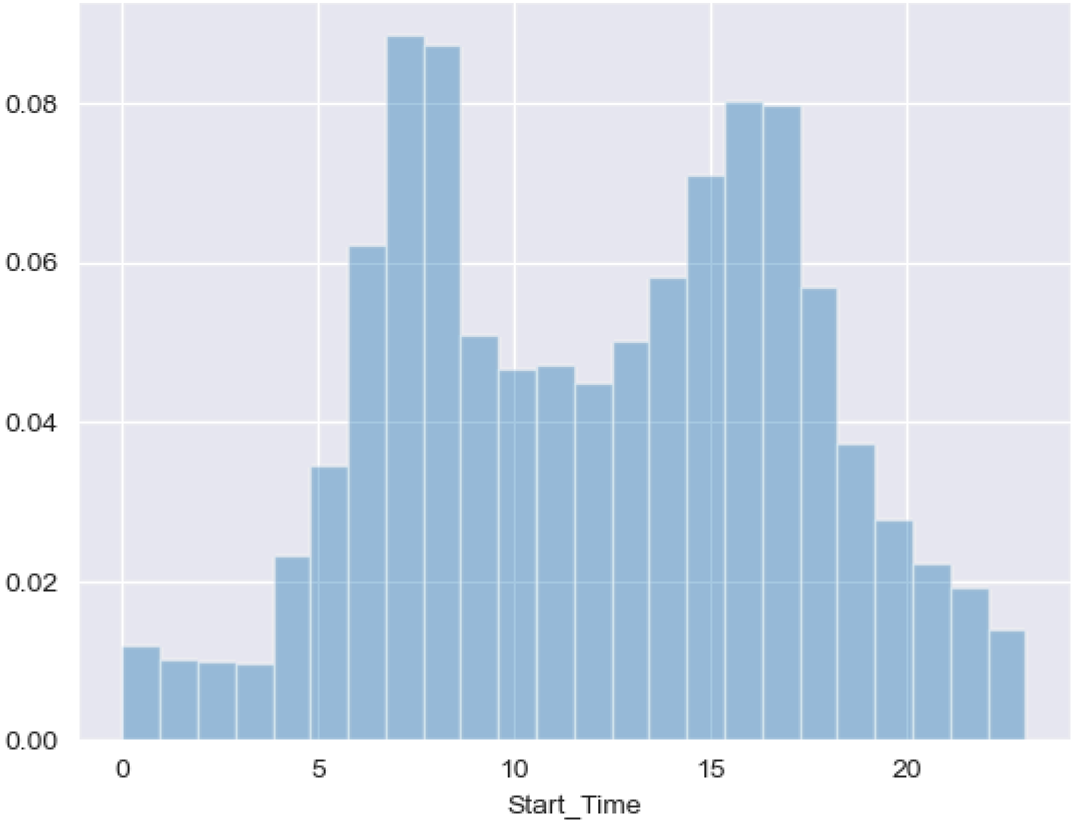
In [43]:

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

```
C:\Users\Atul Gupta\AppData\Local\Temp\ipykernel_11296\1425569934.py:2: UserWarnin
g:

`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(mondays_start_time.dt.hour, bins = 24, kde = False, norm_hist = Tru
e)
```

Out[43]:

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



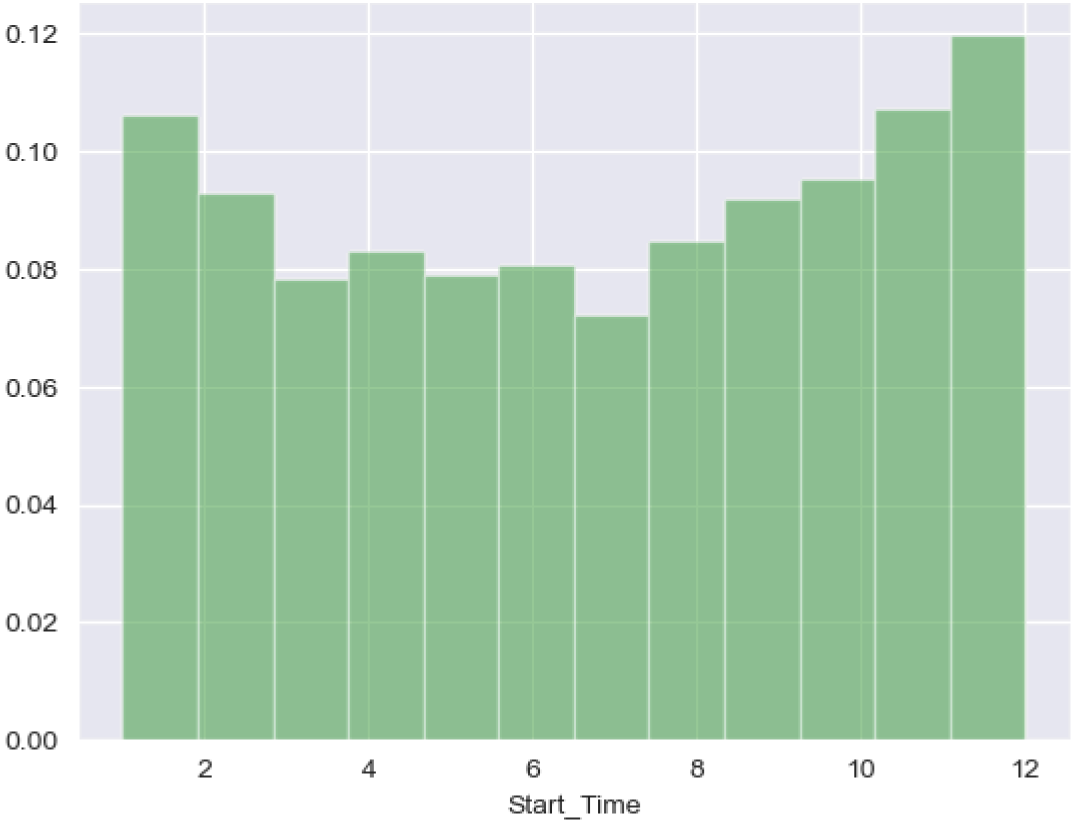
- **On workings i.e. monday, tuesday, wednesday, thurday, friday you'll find almost the same trend in accidents time.
- While on saturday and sunday the is a different trend i.e. from 10 am to 7 pm the frequency of accident is more.**

Analysis for Month distribution

In [44]:
```
sns.distplot(df.Start_Time.dt.month, bins = 12, kde = False, norm_hist = True, colo
```

```
C:\Users\Atul Gupta\AppData\Local\Temp\ipykernel_11296\1242160542.py:1: UserWarnin
g:

`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(df.Start_Time.dt.month, bins = 12, kde = False, norm_hist = True, c
olor = 'green')
```

Out[44]:
```
<Axes: xlabel='Start_Time'>
```



- The accidents are high from December and it is lowest at july. The rise continues to increase from the month of July.
- It's seems during summer there are less accidents but as the winter starts the is a increasing trend in accidents.

Analysis for Year

In [45]:

```
df.Start_Time.dt.year
```

Out[45]:

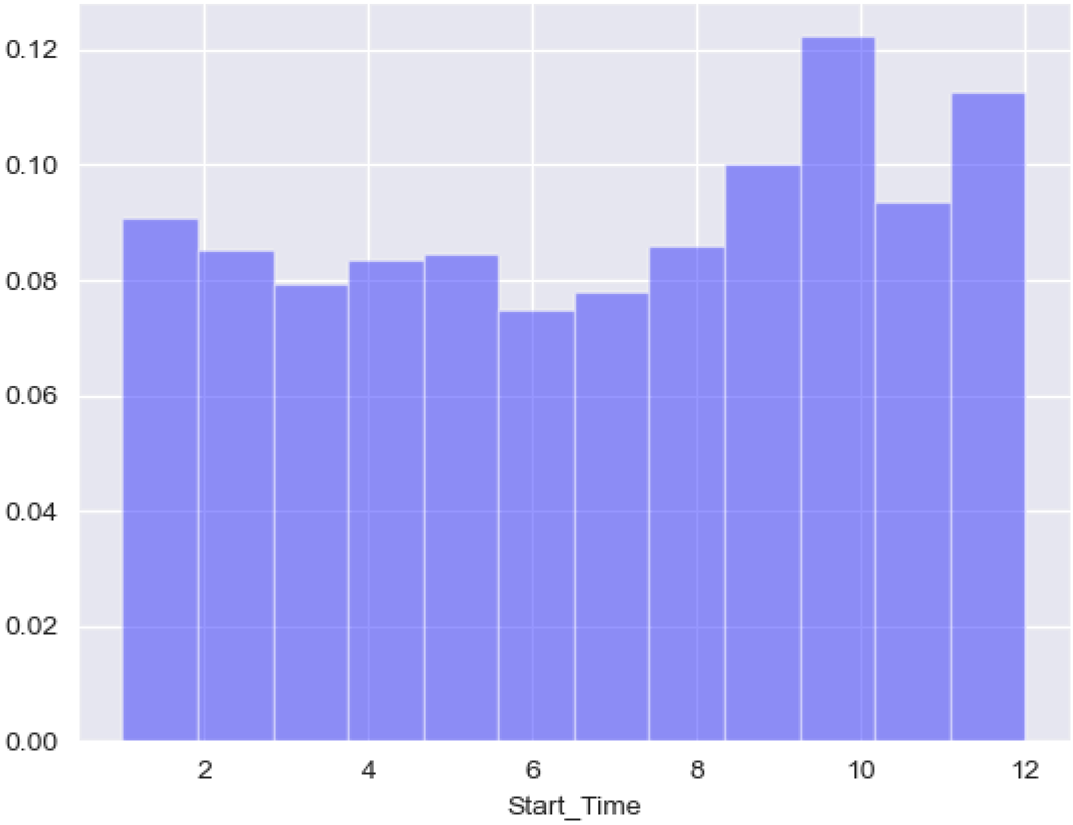
```
0          2016
1          2016
2          2016
3          2016
4          2016
           ... 
7728389    2019
7728390    2019
7728391    2019
7728392    2019
7728393    2019
Name: Start_Time, Length: 7728394, dtype: int64
```

In [46]:

```
df_2019 = df[df.Start_Time.dt.year == 2019]
sns.distplot(df_2019.Start_Time.dt.month, bins = 12, kde = False, norm_hist = True,
C:\Users\Atul Gupta\AppData\Local\Temp\ipykernel_11296\3765400231.py:2: UserWarnin
g:

`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(df_2019.Start_Time.dt.month, bins = 12, kde = False, norm_hist = Tr
ue, color = 'Blue')
```

Out[46]:

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



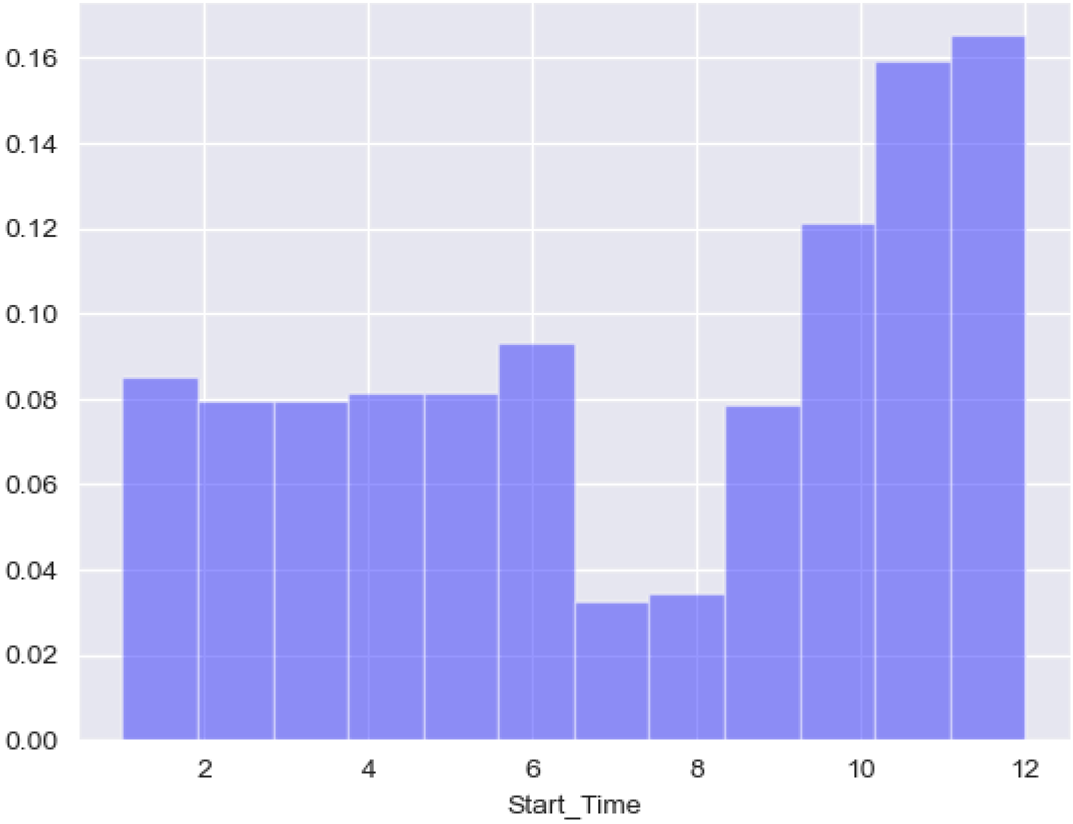
In [47]:

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

```
C:\Users\Atul Gupta\AppData\Local\Temp\ipykernel_11296\2660523566.py:2: UserWarnin
g:

`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(df_2020.Start_Time.dt.month, bins = 12, kde = False, norm_hist = Tr
ue, color = 'Blue')
```

Out[47]:

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



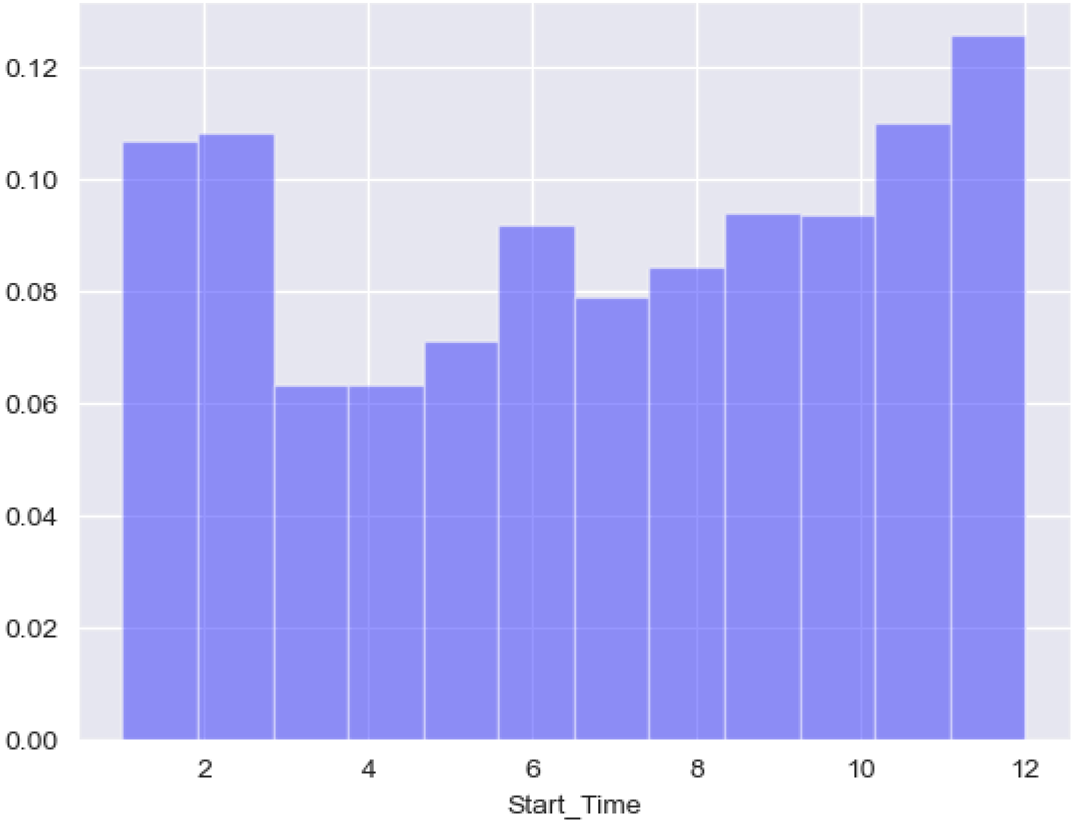
In [48]:

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

```
C:\Users\Atul Gupta\AppData\Local\Temp\ipykernel_11296\607408822.py:2: UserWarnin
g:

`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(df_2021.Start_Time.dt.month, bins = 12, kde = False, norm_hist = Tr
ue, color = 'Blue')
```

Out[48]:

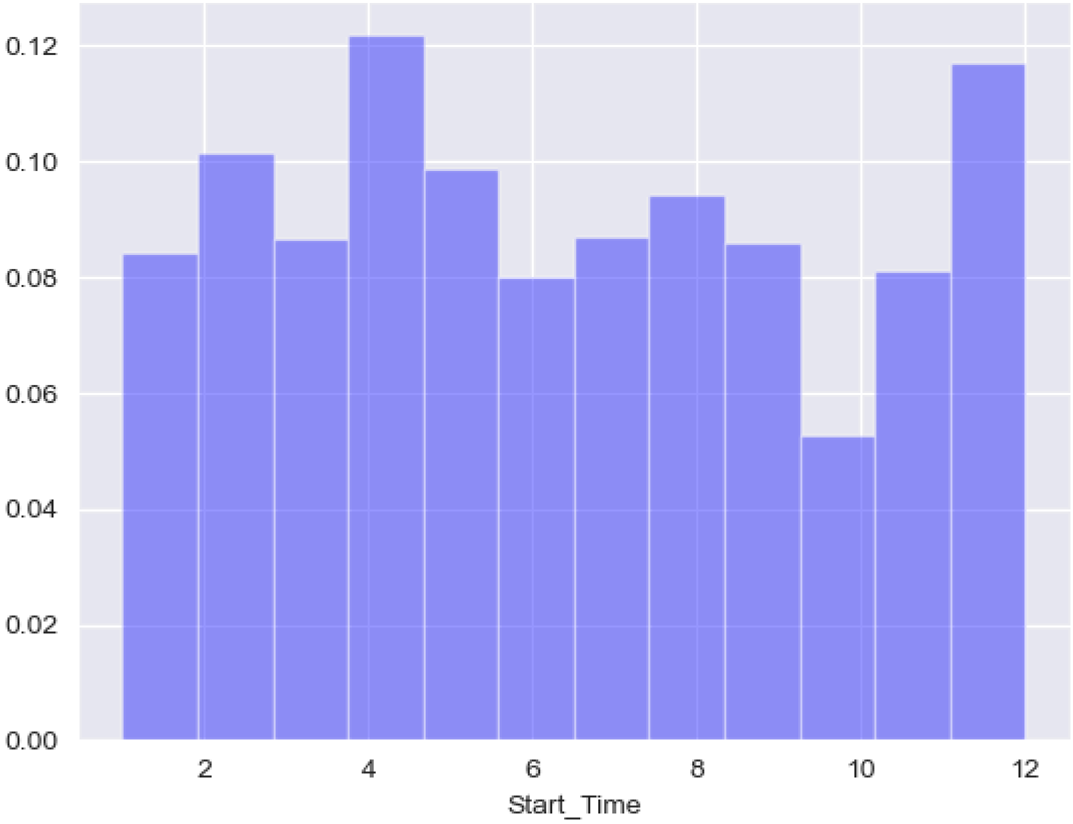
```
<Axes: xlabel='Start_Time'>
```



In [49]:
```
df_2022 = df[df.Start_Time.dt.year == 2022]
sns.distplot(df_2022.Start_Time.dt.month, bins = 12, kde = False, norm_hist = True,
```

```
C:\Users\Atul Gupta\AppData\Local\Temp\ipykernel_11296\1463311549.py:2: UserWarnin
g:

`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(df_2022.Start_Time.dt.month, bins = 12, kde = False, norm_hist = Tr
ue, color = 'Blue')
```

Out[49]:
```
<Axes: xlabel='Start_Time'>
```



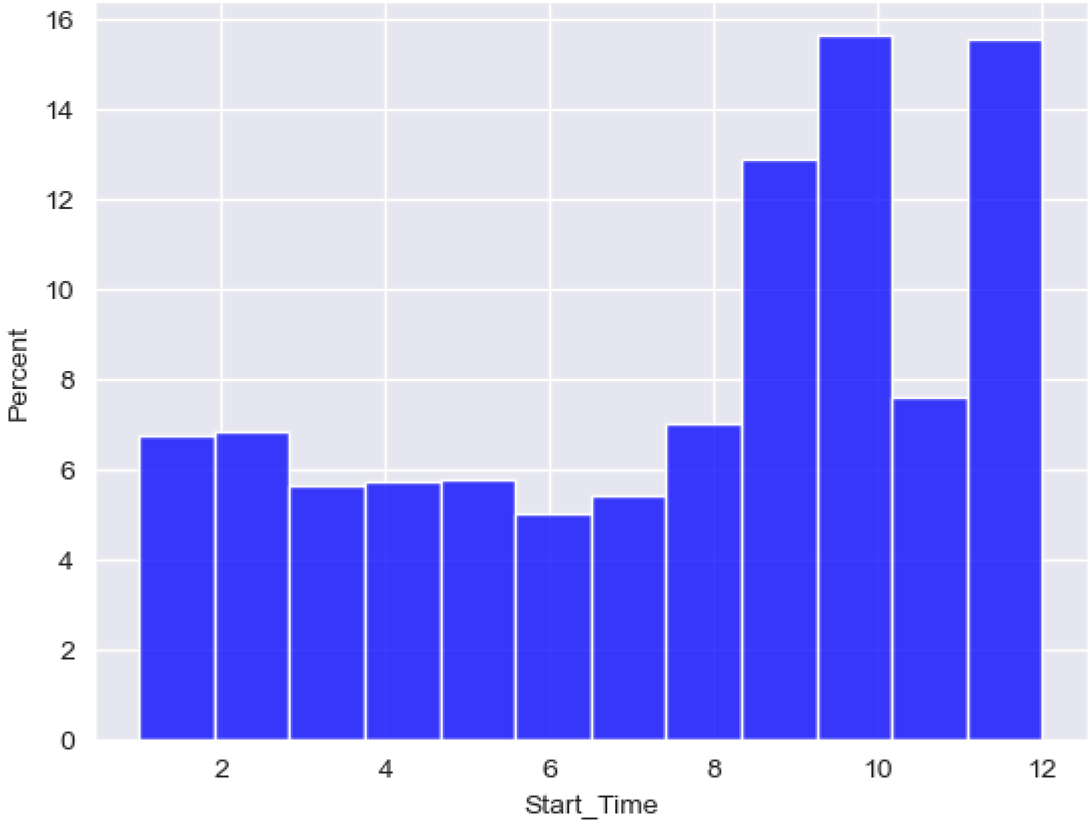
- Much data is missing for yearly analysis
- so,need to check some other colunm affected by our study, we can analysis source dataset

In [50]:
```
df.Source
```

Out[50]:
```
0          Source2
1          Source2
2          Source2
3          Source2
4          Source2
            ...   
7728389    Source1
7728390    Source1
7728391    Source1
7728392    Source1
7728393    Source1
Name: Source, Length: 7728394, dtype: object
```

In [51]:
```
df_2019 = df[df.Start_Time.dt.year == 2019]
df_2019_Source1=df_2019[df_2019.Source == 'Source1']
#sns.distplot(df_2019_Source1.Start_Time.dt.month, bins=12, kde=False,norm_hist = T
sns.histplot(df_2019_Source1['Start_Time'].dt.month, color='blue', bins=12, stat='p
```

Out[51]:
```
<Axes: xlabel='Start_Time', ylabel='Percent'>
```

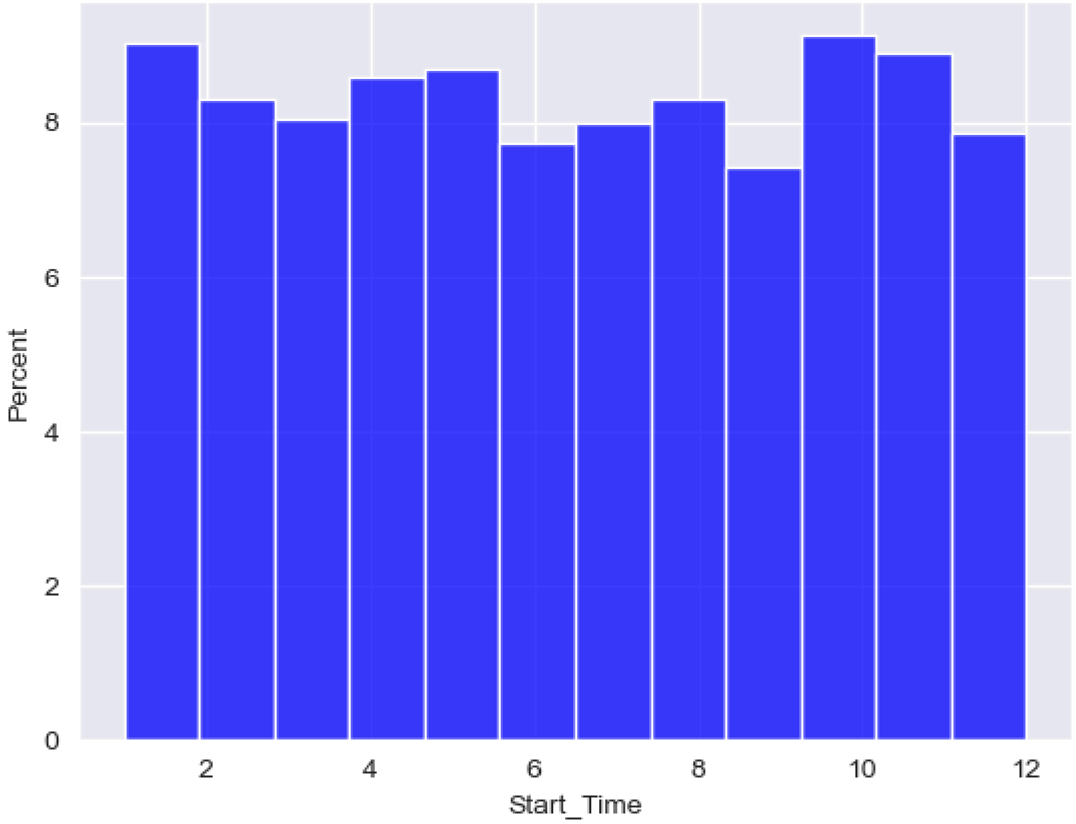


In [52]:

```
df_2019 = df[df.Start_Time.dt.year == 2019]
df_2019_Source2=df_2019[df_2019.Source == 'Source2']
#sns.distplot(df_2019_Source2.Start_Time.dt.month, bins=12, kde=False,norm_hist=Tru
sns.histplot(df_2019_Source2['Start_Time'].dt.month, color='blue', bins=12, stat='p
```

Out[52]:

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



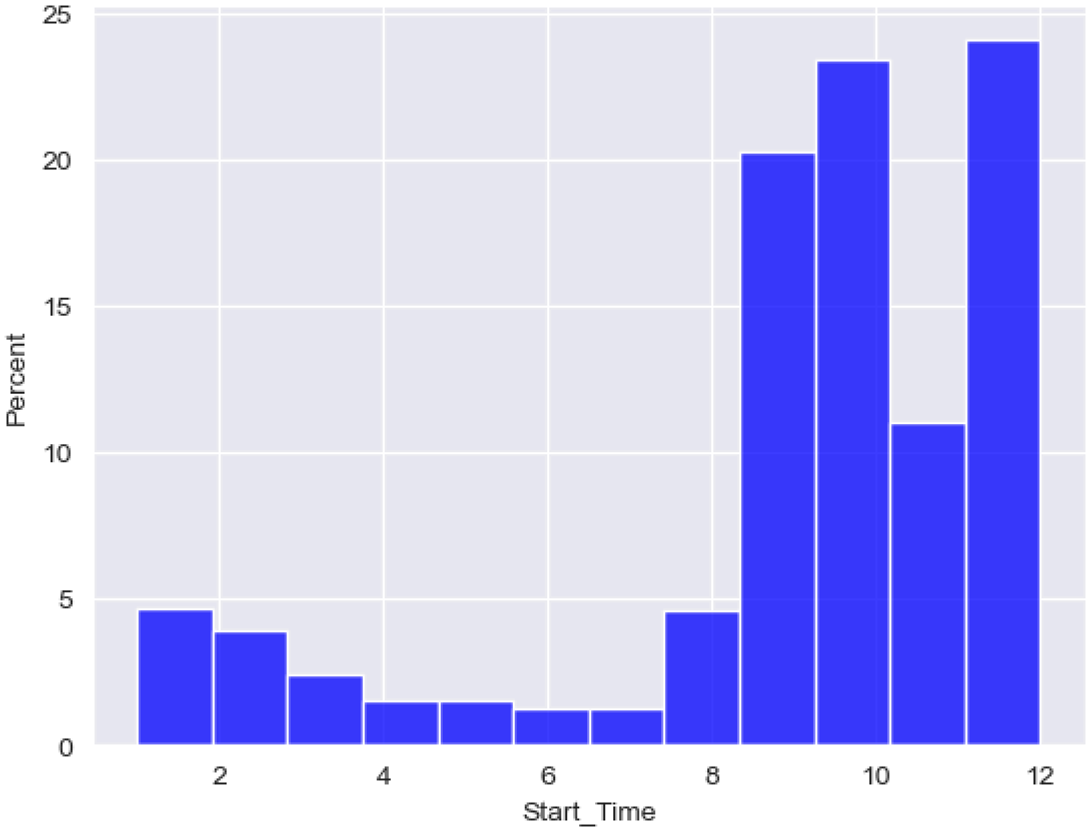
In [53]:

```
df_2019 = df[df.Start_Time.dt.year == 2019]
df_2019_Source3=df_2019[df_2019.Source == 'Source3']
```

```
#sns.distplot(df_2019_Source3.Start_Time.dt.month, bins=12, kde=False, stat='percer
sns.histplot(df_2019_Source3['Start_Time'].dt.month, color='blue', bins=12, stat='p
```

Out[53]:

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

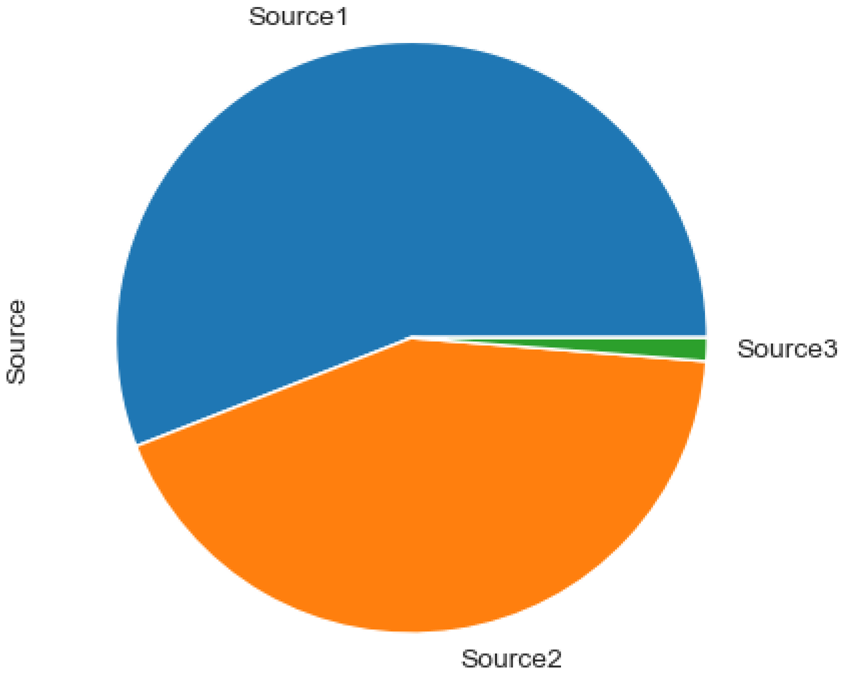


In [54]:

```
df.Source.value_counts().plot(kind = 'pie')
```

Out[54]:

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



- There seems to be some issue with the Source2 and Source3 data so consider excluding Source2 and Source3 **

Start Latitude & Longitude

In [55]:
```
df.Start_Lat
```

Out[55]:
```
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
```

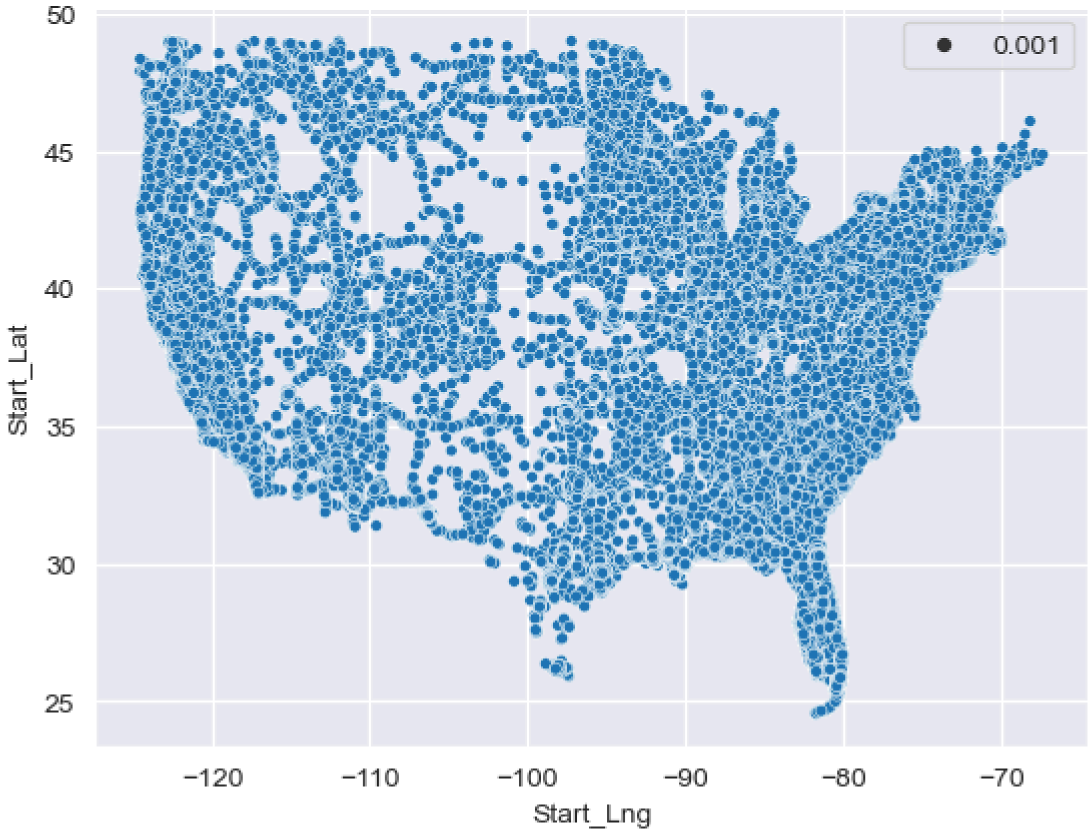
In [56]:
```
df.Start_Lng
```

Out[56]:
```
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
```

In [57]:
```
# use sample function to extract 10% Data
sample_df = df.sample(int(0.1 * len(df)))
```

In [58]:
```
sns.scatterplot(y = sample_df.Start_Lat, x = sample_df.Start_Lng,size = 0.001)
```

Out[58]:
```
<Axes: xlabel='Start_Lng', ylabel='Start_Lat'>
```



In [59]:

```
# show the above Lat & Lng scatter plot in Map (use libraries folium)
import folium
```

In [60]:

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

Out[60]:

```
(-84.058723, 39.865147)
```

In [61]:

```
# sample().iteritems() used to show only 100 results
for x in df[['Start_Lat','Start_Lng']].sample(100).iteritems():
    print(x[1])
```

```
7010944    40.012440
1321647    30.212299
821239     36.052238
4066706    39.872330
104620     33.862186
             ...
1477305    39.890221
3986384    34.947461
3432428    39.105720
6666424    36.200835
3630864    45.101930
Name: Start_Lat, Length: 100, dtype: float64
7010944    -77.537700
1321647    -82.639572
821239     -86.743103
4066706    -75.348974
104620    -118.041985
              ...
1477305    -76.237991
3986384    -89.833700
3432428    -94.840850
6666424    -86.769689
3630864    -93.456630
Name: Start_Lng, Length: 100, dtype: float64
```

```
C:\Users\Atul Gupta\AppData\Local\Temp\ipykernel_11296\3450368729.py:2: FutureWarn
ing: iteritems is deprecated and will be removed in a future version. Use .items i
nstead.
  for x in df[['Start_Lat','Start_Lng']].sample(100).iteritems():
```

In [62]:

```
# creat heatmap
from folium.plugins import HeatMap
```

In [63]:

```
# zip used to pair the both list, it is nescessary to convert list of lat & lng pai
zip(list(df.Start_Lat),list(df.Start_Lng))
```

Out[63]:

```
<zip at 0x1c3e1ea7e00>
```

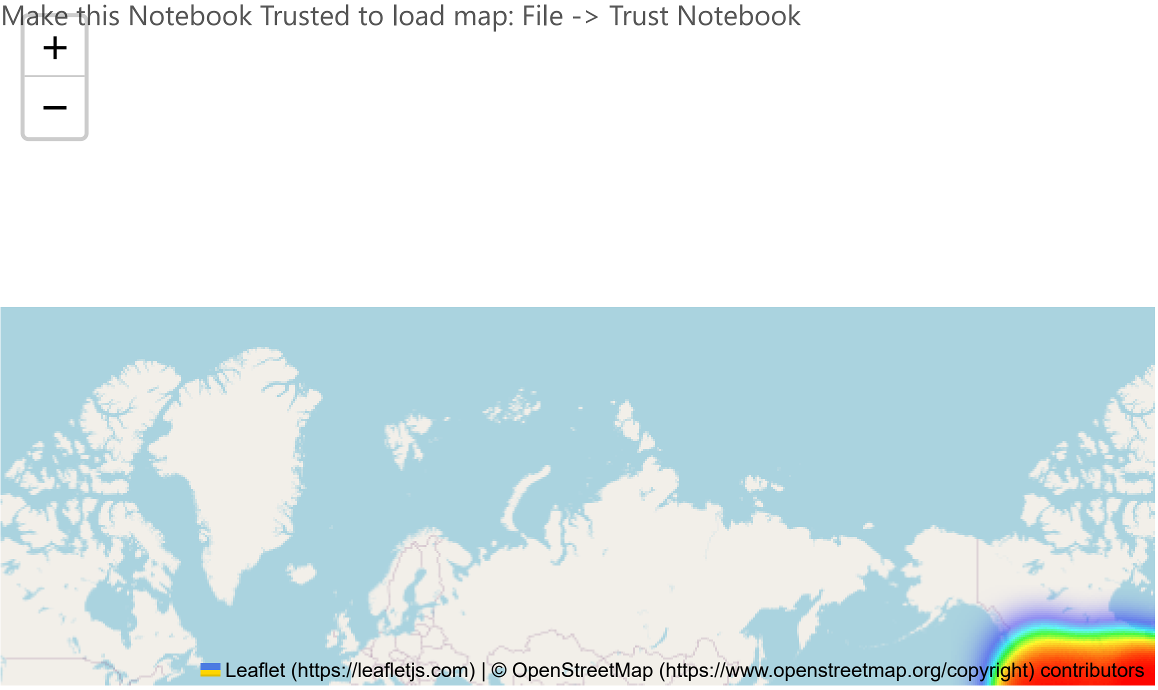
In [64]:

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

In [65]:

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

Out[65]: Make this Notebook Trusted to load map: File -> Trust Notebook


+
−
Leaflet (https://leafletjs.com) | © OpenStreetMap (https://www.openstreetmap.org/copyright) contributors

Are there more accidents in warmer or colder areas

In [66]:

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

Out[66]:

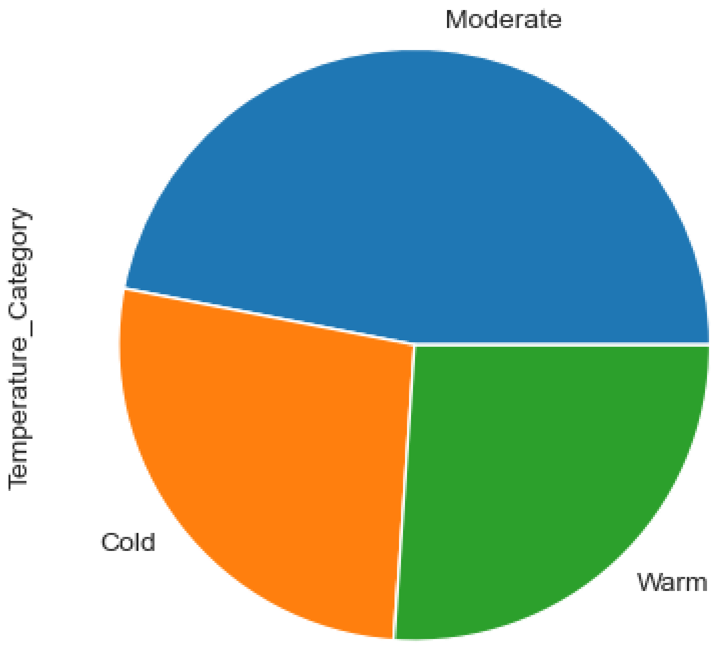
```
0          36.9
1          37.9
2          36.0
3          35.1
4          36.0
           ...
7728389    86.0
7728390    70.0
7728391    73.0
7728392    71.0
7728393    79.0
Name: Temperature(F), Length: 7728394, dtype: float64
```

In [67]:
```
# Create temperature bins (customize according to your data)
bins = [0, 50, 75, 100]
labels = ['Cold', 'Moderate', 'Warm']
# Assign temperature ranges to each row
df['Temperature_Category'] = pd.cut(df['Temperature(F)'], bins=bins, labels=labels,
df['Temperature_Category']
```

Out[67]:
```
0              Cold
1              Cold
2              Cold
3              Cold
4              Cold
             ...   
7728389        Warm
7728390    Moderate
7728391    Moderate
7728392    Moderate
7728393        Warm
Name: Temperature_Category, Length: 7728394, dtype: category
Categories (3, object): ['Cold' < 'Moderate' < 'Warm']
```

In [68]:
```
# Group by temperature category and calculate the number of accidents
#accidents_by_temperature = df.groupby('Temperature_Category', observed=False).size

accidents_by_temperature = df['Temperature_Category'].value_counts()
df['Temperature_Category'].value_counts().plot(kind='pie')
```

Out[68]:
```
<Axes: ylabel='Temperature_Category'>
```



- Moderate temperature days more accident happens

Analyzing the data by state

In [69]:
```
states = df['State'].value_counts().head(5)
states
# The data indicates california is the highest accident state
```

Out[69]:

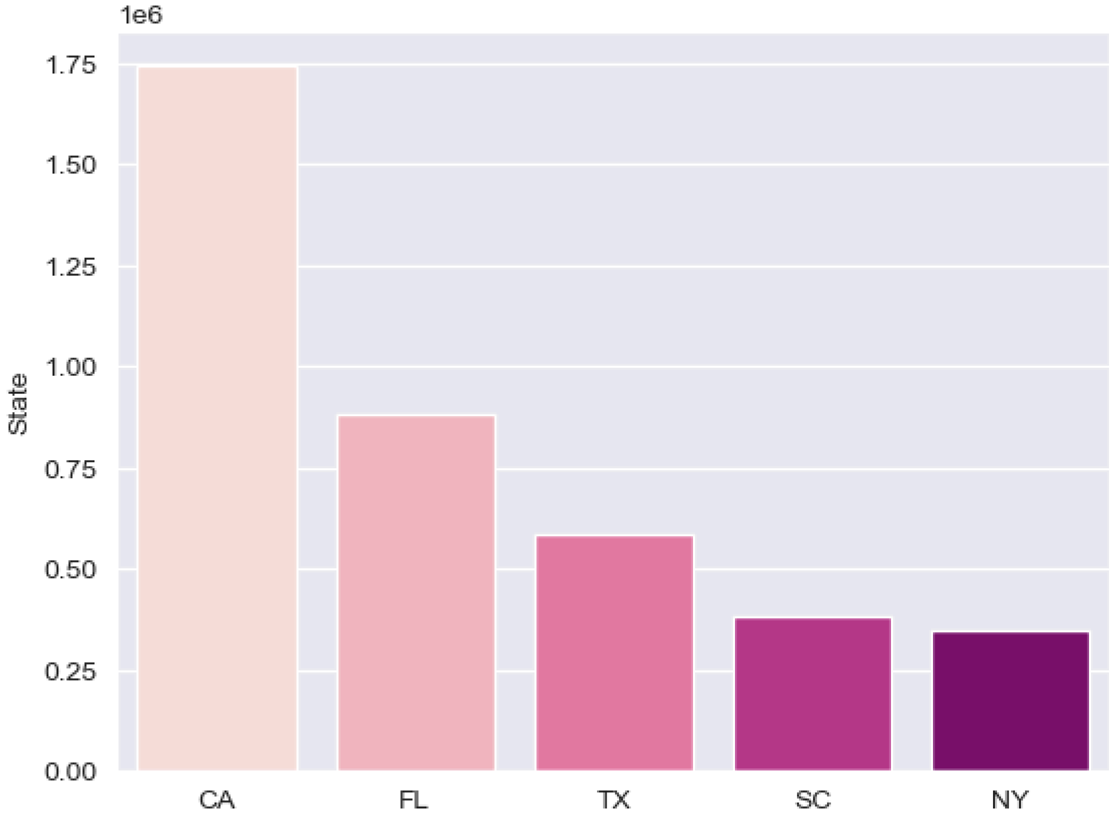
```
CA    1741433
FL     880192
TX     582837
SC     382557
NY     347960
Name: State, dtype: int64
```

In [70]:

```
sns.barplot(y=states , x = states.index, palette="RdPu")
```

Out[70]:

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



Summary and Conclusion

Insights:

- The cities with the highest reported accidents are Miami, Houston, Los Angeles, Charlotte, Dallas, Orlando, Austin, Raleigh, Nashville, Baton Rouge, Atlanta, Sacramento, San Diego, Phoenix, Minneapolis, Richmond, Oklahoma City, Jacksonville, Tucson, and Columbia.
- About 8.9% of cities experience a high number of accidents.
- The majority of cities (91%) have a low number of accidents.
- Over 1023 cities reported just 1 accident, suggesting the presence of potential outliers that may need to be addressed.
- There is a notable spike in accidents around 7-8 AM, possibly correlated with morning rush hours and commuting to work or school.
- Another spike occurs around 4-5 PM, likely associated with evening rush hours and the return home from work or recreational activities.
- On weekdays (Monday to Friday), the trend in accident times is consistent.
- On weekends (Saturday and Sunday), there is a different trend, with a higher frequency of accidents between 10 AM and 7 PM.

- There is a seasonal variation in accidents, with fewer incidents during the summer and an increasing trend as winter approaches.
- The use of Folium indicates that many people live near bay areas.
- No data from New York
- California, Florida, Texas, South Carolina, and New York emerge as the top 5 states with the highest number of accidents
- Moderate temperature days more accident happens

In [ ]: