US accidents analysis

URL:https://www.kaggle.com/sobhanmoosavi/us-accidents
(https://www.kaggle.com/sobhanmoosavi/us-accidents)

In [6]:
```
import opendatasets as od
import os
```

Loading dataset from kaggle

In [9]:
```
od.download('https://www.kaggle.com/sobhanmoosavi/us-accidents')
```

```
Please provide your Kaggle credentials to download this dataset. Learn mor
e: http://bit.ly/kaggle-creds (http://bit.ly/kaggle-creds)
Your Kaggle username: swaroopss
Your Kaggle Key: ........

  0%|
| 0.00/117M [00:00<?, ?B/s]

Downloading us-accidents.zip to .\us-accidents

100%|████████████████████████████████████████████████████████████████
██████████| 117M/117M [02:16<00:00, 903kB/s]
```

In [10]:
```
import json

with open('path\\kaggle.json', 'r') as f:
    data=json.load(f)
    print(data)
```

...

In [4]:
```
od.download('https://www.kaggle.com/sobhanmoosavi/us-accidents')
```

...

In [6]:
```
import os
os.listdir("./us-accidents")
```

Out[6]:
```
['US_Accidents_Dec20_updated.csv']
```

Data clensing and preparation

In [7]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.pylab import rcParams
import seaborn as sns
%matplotlib inline
```

In [8]:
```
import matplotlib
```

In [9]:
```
sns.set_style("darkgrid")
matplotlib.rcParams['figure.figsize']=(20,10)
matplotlib.rcParams['font.size']=(10)
```

In [10]:
```
data=pd.read_csv('./us-accidents/US_Accidents_Dec20_updated.csv')
```

In [11]:
```
pd.set_option('display.max_row', None)
pd.set_option('display.max_column', None)
```

In [12]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1516064 entries, 0 to 1516063
Data columns (total 47 columns):
 #   Column                 Non-Null Count    Dtype  
---  ------                 --------------    -----  
 0   ID                     1516064 non-null  object 
 1   Severity               1516064 non-null  int64  
 2   Start_Time             1516064 non-null  object 
 3   End_Time               1516064 non-null  object 
 4   Start_Lat              1516064 non-null  float64
 5   Start_Lng              1516064 non-null  float64
 6   End_Lat                1516064 non-null  float64
 7   End_Lng                1516064 non-null  float64
 8   Distance(mi)           1516064 non-null  float64
 9   Description            1516064 non-null  object 
 10  Number                 469969 non-null   float64
 11  Street                 1516064 non-null  object 
 12  Side                   1516064 non-null  object 
 13  City                   1515981 non-null  object 
 14  County                 1516064 non-null  object 
```

In [13]:
```
data.isna().sum()
```

Out[13]:
```
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           1046095
Street                 0
Side                   0
City                  83
County                 0
State                  0
Zipcode              935
Country                0
Timezone            2302
```

In [14]:
```
## it
(data.shape[0]-data['Number'].isna().sum())/data.shape[0]
```

Out[14]:
```
0.30999284990607257
```

In [15]:
```
use_dict={'per_data':[],'per_null':[],'col_name':[]}
def feature_value(col):
    ## percent of data avilable in column
    d=data.shape[0]
     ## percent of data is null in column
    c=data[col].isna().sum()
    use_dict['per_data'].append(((d-c)/d)*100)
    use_dict['per_null'].append((c/d)*100)
    use_dict['col_name'].append(col)

cols=list(data.columns)
for n in cols:
    feature_value(n)
```

In [16]:
```
df=pd.DataFrame(use_dict)
df
```

Out[16]:

	per_data	per_null	col_name
0	100.000000	0.000000	ID
1	100.000000	0.000000	Severity
2	100.000000	0.000000	Start_Time
3	100.000000	0.000000	End_Time
4	100.000000	0.000000	Start_Lat
5	100.000000	0.000000	Start_Lng
6	100.000000	0.000000	End_Lat
7	100.000000	0.000000	End_Lng
8	100.000000	0.000000	Distance(mi)
9	100.000000	0.000000	Description
10	30.999285	69.000715	Number

In [17]:
```
data.head(20)
```

Out[17]:

	ID	Severity	Start_Time	End_Time	Start_Lat	Start_Lng	End_Lat	End_Lng
0	A-2716600	3	2016-02-08 00:37:08	2016-02-08 06:37:08	40.108910	-83.092860	40.112060	-83.031870
1	A-2716601	2	2016-02-08 05:56:20	2016-02-08 11:56:20	39.865420	-84.062800	39.865010	-84.048730
2	A-2716602	2	2016-02-08 06:15:39	2016-02-08 12:15:39	39.102660	-84.524680	39.102090	-84.523960
3	A-2716603	2	2016-02-08 06:15:39	2016-02-08 12:15:39	39.101480	-84.523410	39.098410	-84.522410
4	A-	2	2016-02-08	2016-02-				

In [18]:
```
numeric=data.select_dtypes(include=np.number).columns
```

In [19]:
```
len(numeric)
```

Out[19]: 14

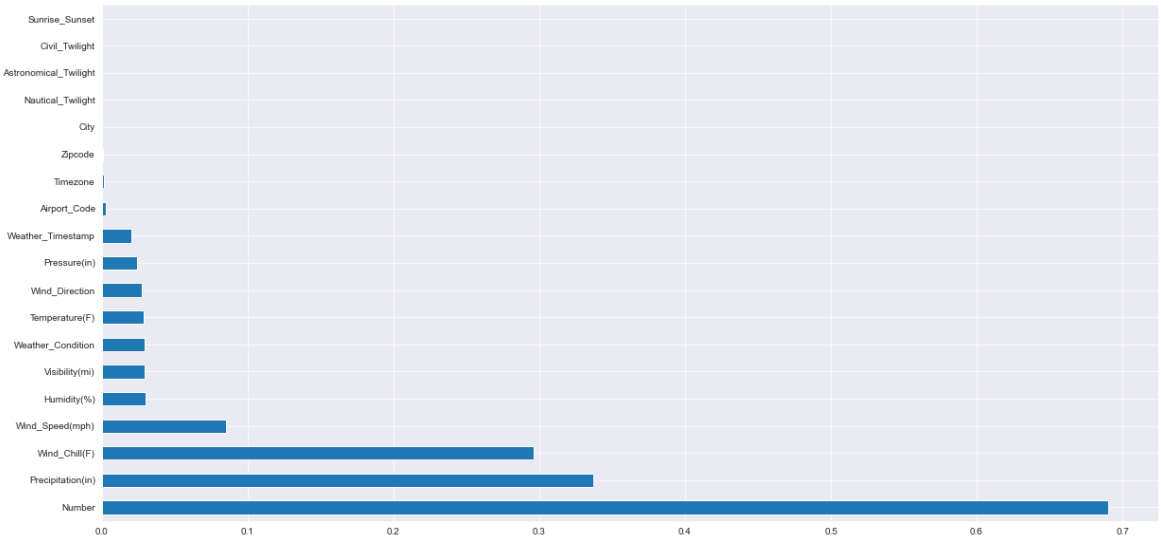
In [20]:
```
data['Street'].nunique()
```

Out[20]: 93048

In [21]:
```
per_missing=(data.isna().sum().sort_values(ascending=False))/len(data)
per_missing
```

Out[21]:
```
Number                 0.690007
Precipitation(in)      0.336760
Wind_Chill(F)          0.296370
Wind_Speed(mph)        0.084998
Humidity(%)            0.030018
Visibility(mi)         0.029162
Weather_Condition      0.029027
Temperature(F)         0.028385
Wind_Direction         0.027610
Pressure(in)           0.023926
Weather_Timestamp      0.019962
Airport_Code           0.002802
Timezone               0.001518
Zipcode                0.000617
City                   0.000055
Nautical_Twilight      0.000055
Astronomical_Twilight  0.000055
Civil_Twilight         0.000055
Sunrise_Sunset         0.000055
Amenity                0.000000
```

In [22]:
```
per_missing[per_missing!=0].plot(kind='barh');

#sns.lineplot(data=per_missing[per_missing!=0])
```



removing columns that are having less than 50% of the data

In [23]:
```
data.drop(['Number'], axis=1)
```

. . .

points to consider

- consider most influential features
- consider facts and make sure whether it is included or not in dataframe, if not included highlight the point
- in each feature look of repetaion of perticular value and try to analyise on that
- pertaining to this data we need to look into the factors to reduce the accidents

lets do some analysis

In [25]:
```
data.columns
```

Out[25]:
```
Index(['ID', 'Severity', 'Start_Time', 'End_Time', 'Start_Lat', 'Start_Ln
g',
       'End_Lat', 'End_Lng', 'Distance(mi)', 'Description', 'Number', 'Stre
et',
       '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', 'Amenit
y',
       'Bump', 'Crossing', 'Give_Way', 'Junction', 'No_Exit', 'Railway',
       'Roundabout', 'Station', 'Stop', 'Traffic_Calming', 'Traffic_Signa
l',
       'Turning_Loop', 'Sunrise_Sunset', 'Civil_Twilight', 'Nautical_Twilig
ht',
       'Astronomical_Twilight'],
      dtype='object')
```

**Consider some columns that seems important and influential to the output

1. City
2. State
3. Weather_Condition
4. Start_Lng
5. Start_Lat
6. Weather_

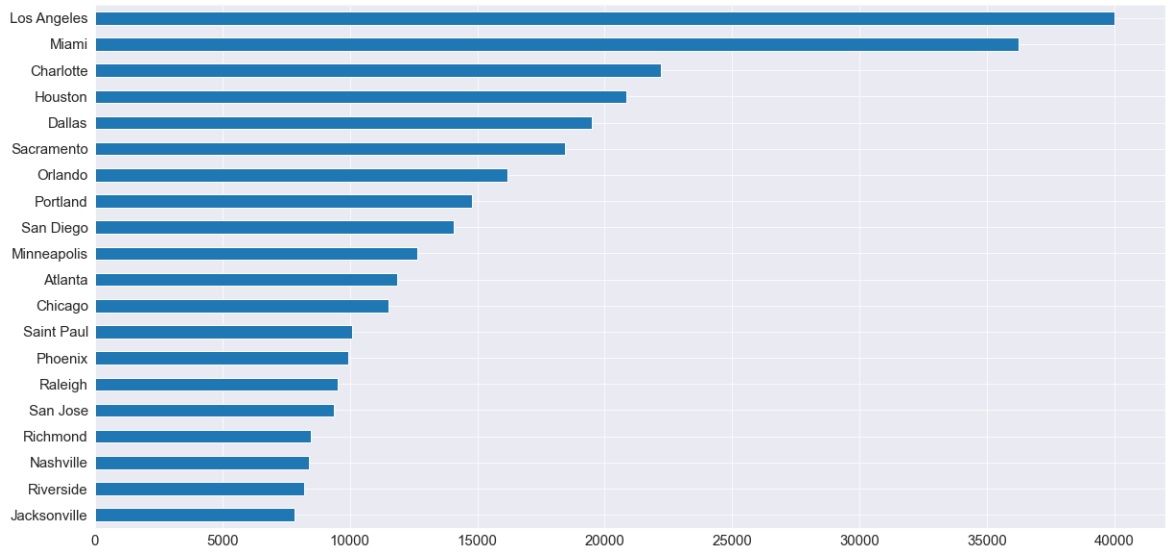
In [26]:
```
top_city=data.City.value_counts().sort_values(ascending=False)
```

New york is the most populated city in US, but why we don't have that in here

In [27]:
```
'New York' in data.City, 'New York' in data.State
```

Out[27]:
```
(False, False)
```

In [28]:
```
f_size=15
plt.rc('font',size=f_size)
top_city[:20].sort_values(ascending=True).plot(kind='barh');
```

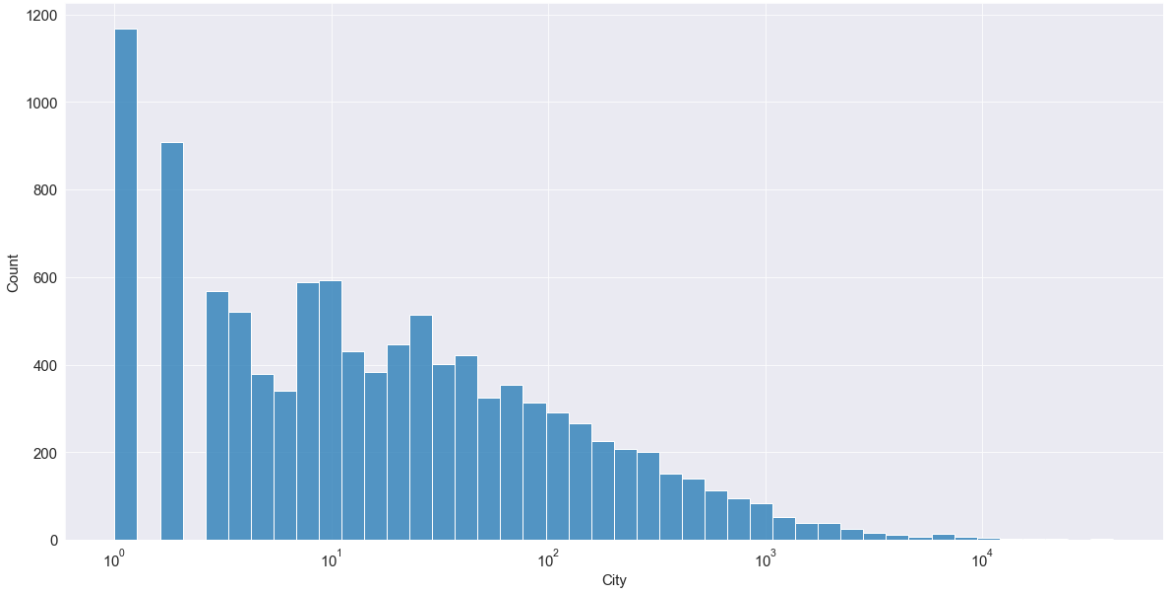


In [29]:
```
df1=pd.DataFrame({'city':top_city.index, 'count':top_city.values})
```

In [30]:

```
import plotly.express as px



fig = px.bar(df1.head(20), x='city', y='count',title='cities with more No on
fig.show()
```

In [31]:
```
import seaborn as sns
sns.set_style('darkgrid')
sns.histplot(top_city, log_scale=True);
```



plot show--->> major chunk of data present in bewtween 100 to 1000

In [32]:
```
top_city[top_city==1].count()
```

Out[32]: 1167

\insight

about 1167 cities have only one yearly accident

IN accordance with above plot we can infer that accidents in most city is not morethan 2000

In [33]:
```
## creating bucket to divide cities with morethan 1000 accidents

high_acc=top_city[top_city>1000]
low_acc=top_city[top_city<=1000]
```

In [34]:
```
len(high_acc)
```

Out[34]: 251

only 251 cities have morethan 1000 accidents

And obviously these cities are more populated than other lesser accidents cities

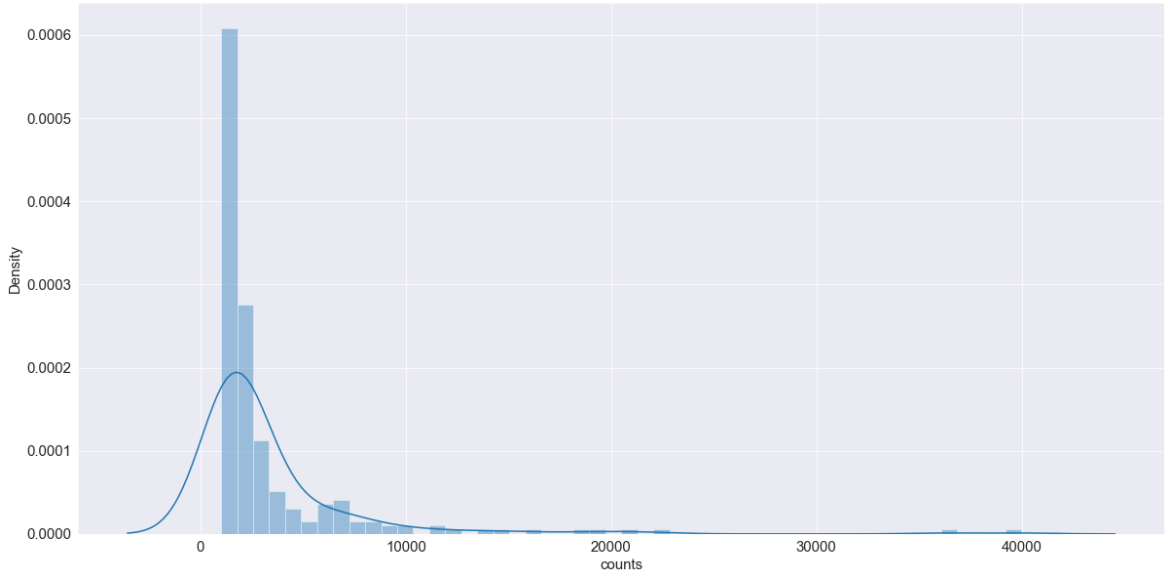
In [35]:
```
## percent of cities with morethan 1k accidents
(len(high_acc)/len(top_city))*100
```

Out[35]: 2.3552594538800786

In [36]:
```
sns.distplot(high_acc);
plt.xlabel('counts')
```

```
c:\users\swaro\appdata\local\programs\python\python37\lib\site-packages\sea
born\distributions.py:2551: FutureWarning:

`distplot` is a deprecated function and will be removed in a future versio
n. Please adapt your code to use either `displot` (a figure-level function
with similar flexibility) or `histplot` (an axes-level function for histogr
ams).
```

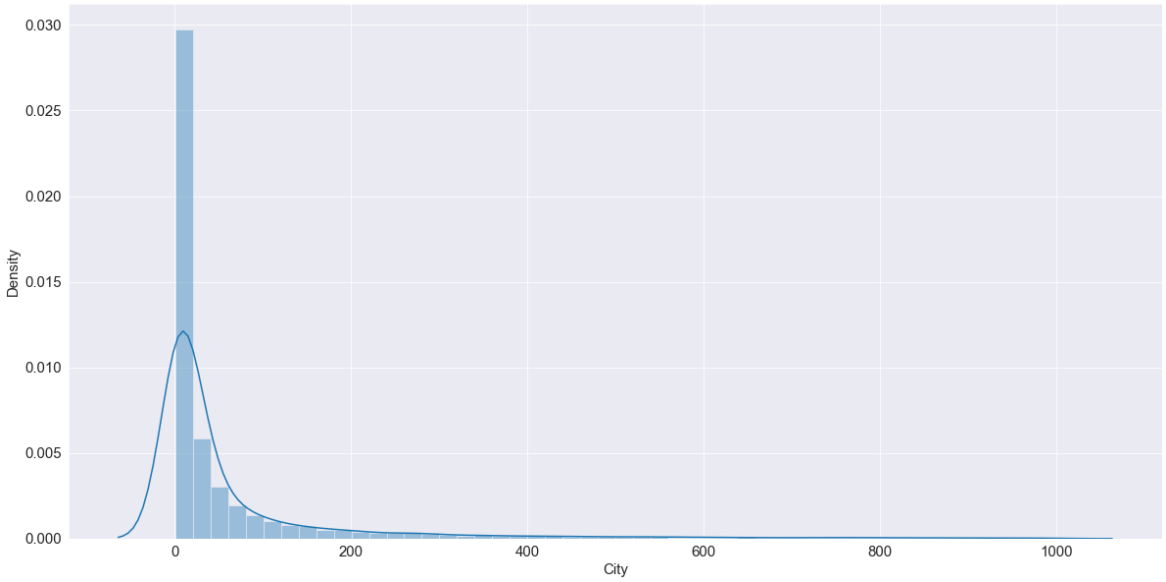
Out[36]: Text(0.5, 0, 'counts')



Out of those more accident ocuured cities there is exponential decrease in the number of cities as the accidents increases

In [37]:
```
sns.distplot(low_acc);
```

```
c:\users\swaro\appdata\local\programs\python\python37\lib\site-packages\seaborn\distributions.py:2551: 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).
```



In [38]:
```
len(low_acc)
```

Out[38]:
```
10406
```

\insight

number of accitents per city exonentially decreases

Start time

In [39]:
```
data.Start_Time.head(5), type(data.Start_Time[0])
```

Out[39]:
```
(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 06:15:39
 4    2016-02-08 06:51:45
 Name: Start_Time, dtype: object,
 str)
```

In [40]:
```
data.Start_Time=pd.to_datetime(data.Start_Time)
```

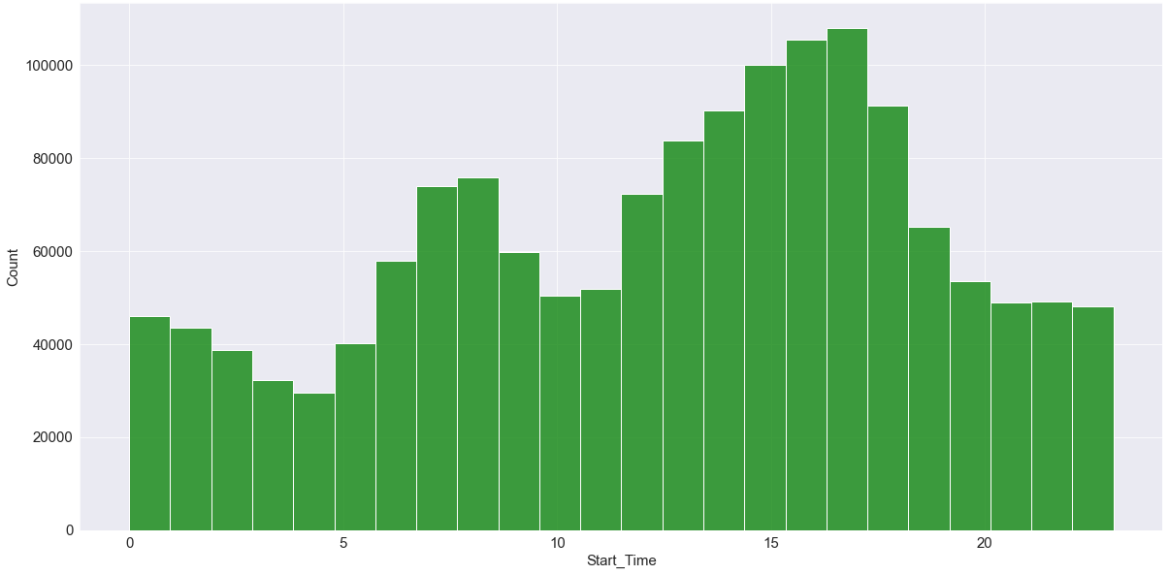
In [41]:
```
type(data.Start_Time[0])
```

Out[41]:
```
pandas._libs.tslibs.timestamps.Timestamp
```

In [42]:
```
hour=data.Start_Time.dt.hour
```

In [43]:
```
sns.histplot(hour, bins=24, kde=False, stat="count", color='green')
```

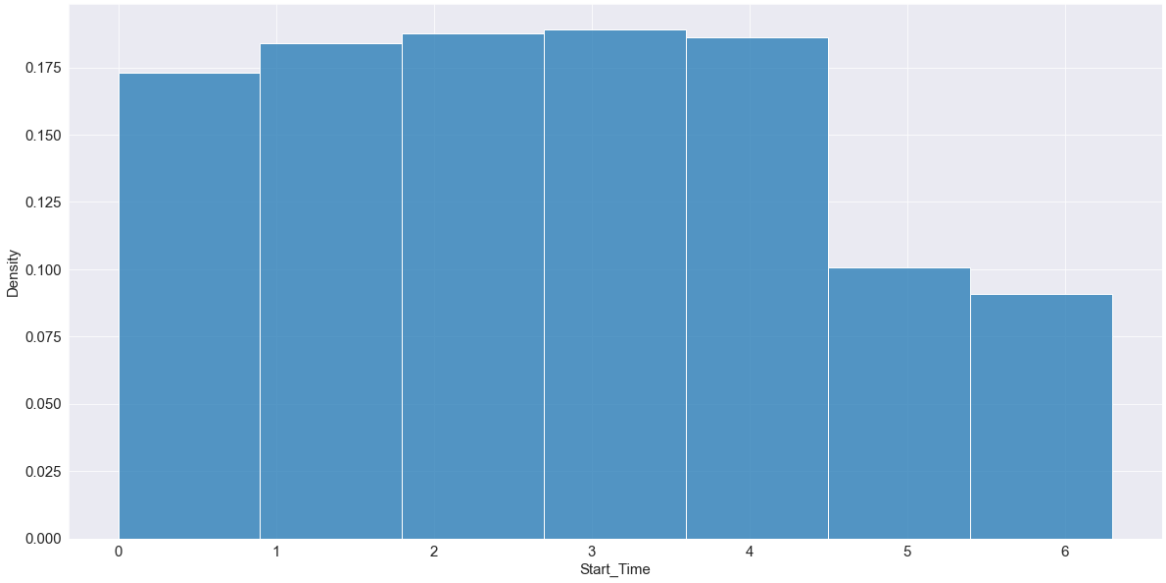
Out[43]:
```
<AxesSubplot:xlabel='Start_Time', ylabel='Count'>
```



Accidents are more between 5am and 10am
adding to this accidents were even more in bewteen the timeline 15pm to 20pm (night)

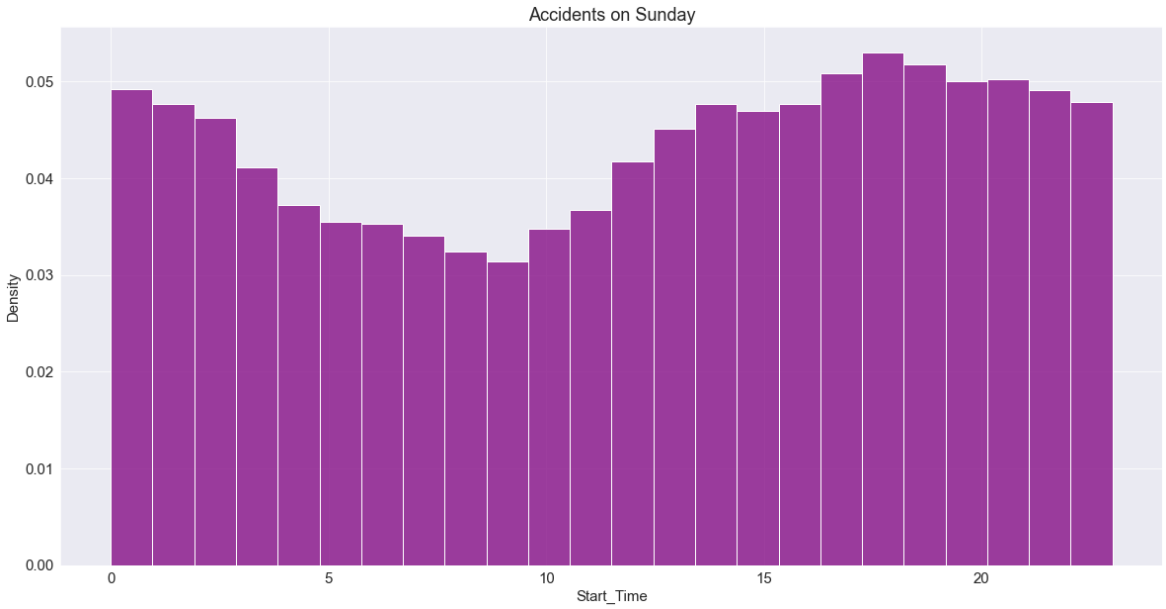
In [44]:
```
#day of the week distribution

sns.histplot(data.Start_Time.dt.dayofweek, bins=7,stat='density', binwidth=0.
```

Out[44]: <AxesSubplot:xlabel='Start_Time', ylabel='Density'>



In [45]:
```
### lets check the hourly distribution of accidents on weekends
sunday=data.Start_Time[data.Start_Time.dt.dayofweek==6]
```

In [46]:
```
plt.title("Accidents on Sunday")
sns.histplot(sunday.dt.hour, color='purple', stat='density', bins=24);
```

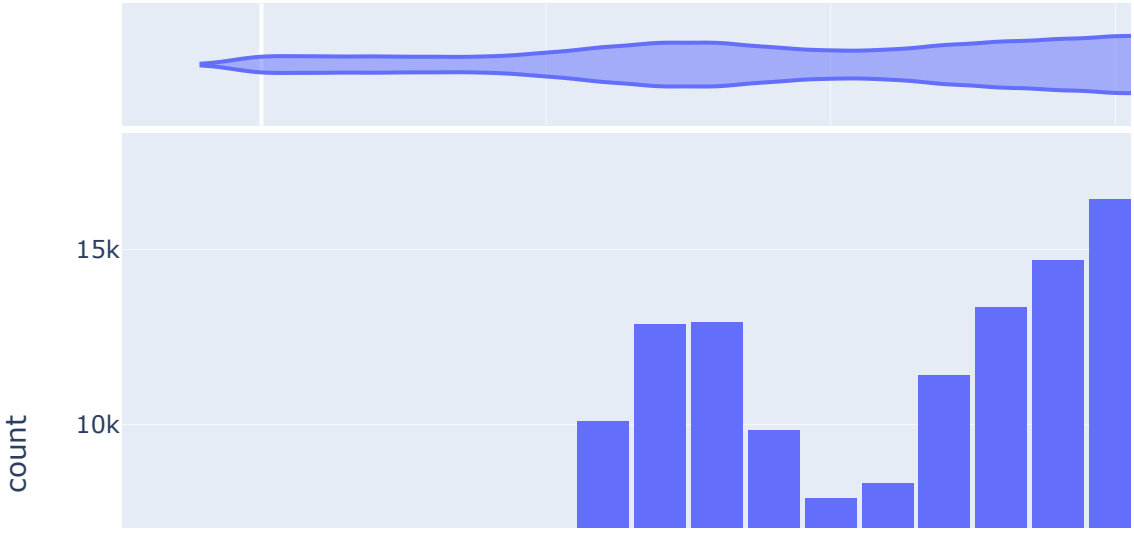


Pretty normally distributed in weekends

In [47]:
```
## lets look at weekdays trend
monday=data.Start_Time[data.Start_Time.dt.dayofweek==0]
```

In [48]:

```
fig = px.histogram(monday, x=monday.dt.hour,
                   marginal="violin"
                  )
fig.update_layout(bargap=0.1)
fig.show()
```



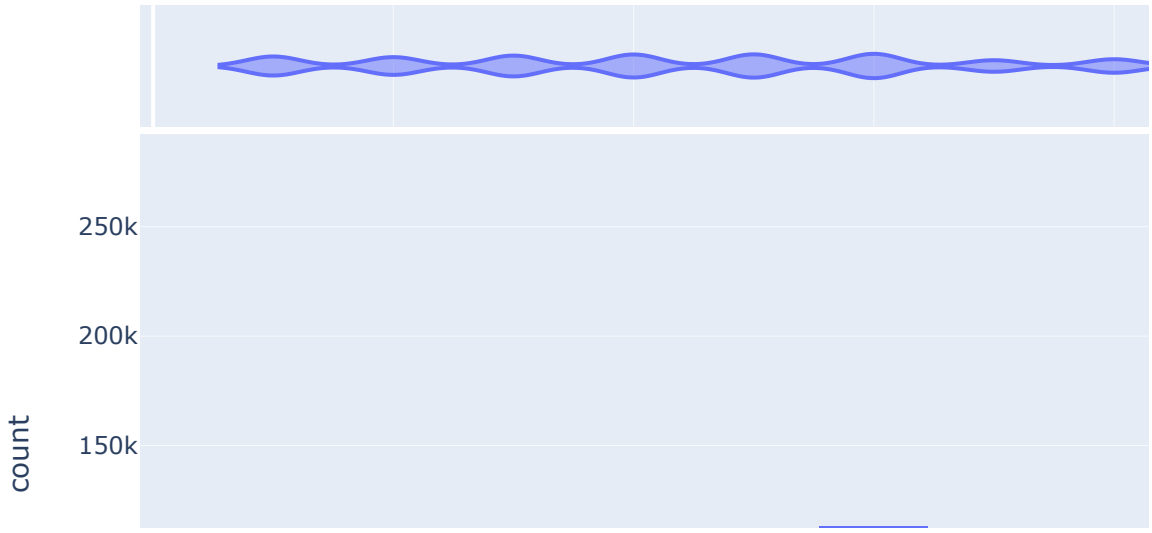
\insoght

During weekends accidents are normally distributed but in all weekdays trend is almost same where in accidents are more during 5am to 10am and again more in between 4pm to 10pm

monthly trends

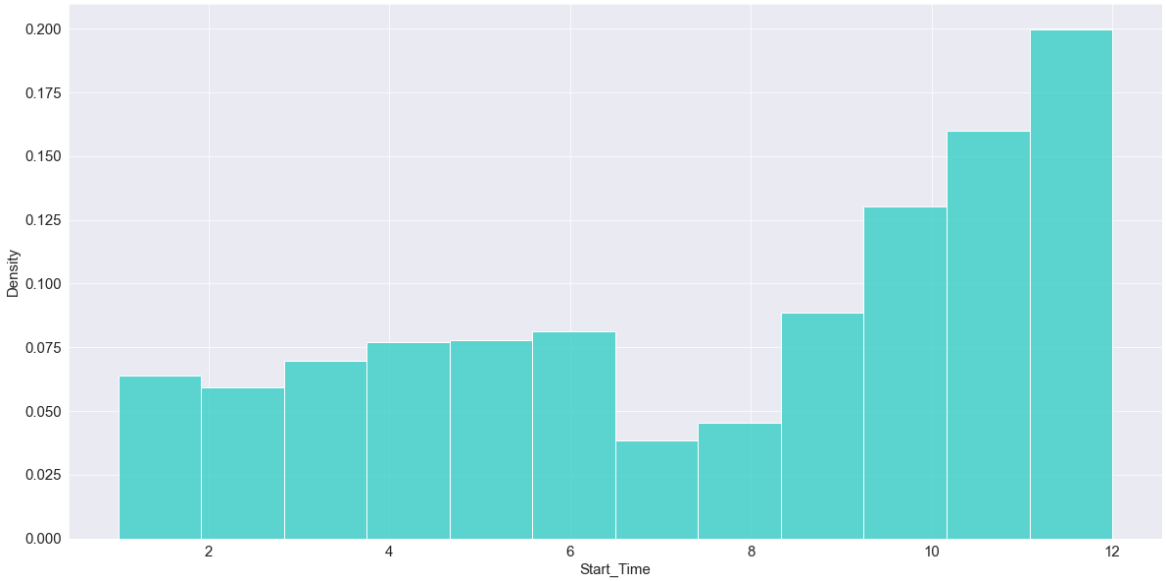
In [49]:

```
## lets look into the monthly data

months=data.Start_Time.dt.month
fig=px.histogram(data.Start_Time,x=months, marginal='violin')
fig.update_layout(bargap=0.1)
```



In [50]:
```
sns.histplot(data=data.Start_Time.dt.month, color='#2BCDC5', stat='density',
```

Out[50]: <AxesSubplot:xlabel='Start_Time', ylabel='Density'>

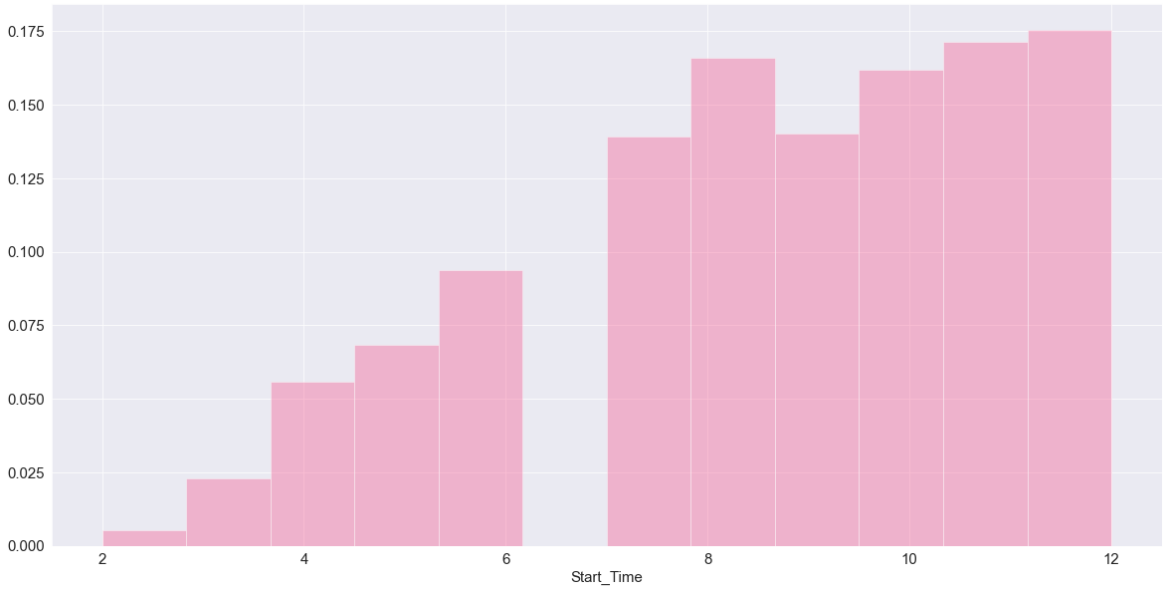


\insight

A/c to above plot--->> during winter accidents are more compare to summer, it does'nt make any sense. There might be some data missing
lets look into the monthly distribution per year

In [51]:
```
yearly=data.Start_Time[data.Start_Time.dt.year==2016]
sns.distplot(yearly.dt.month, color='#F66095', norm_hist=True, bins=12, kde=F
```

```
c:\users\swaro\appdata\local\programs\python\python37\lib\site-packages\sea
born\distributions.py:2551: FutureWarning:

`distplot` is a deprecated function and will be removed in a future versio
n. Please adapt your code to use either `displot` (a figure-level function
with similar flexibility) or `histplot` (an axes-level function for histogr
ams).
```



\insight

collectively, data is missing in 7th month and there are less records collected between the months from jan to june compare to winter months.

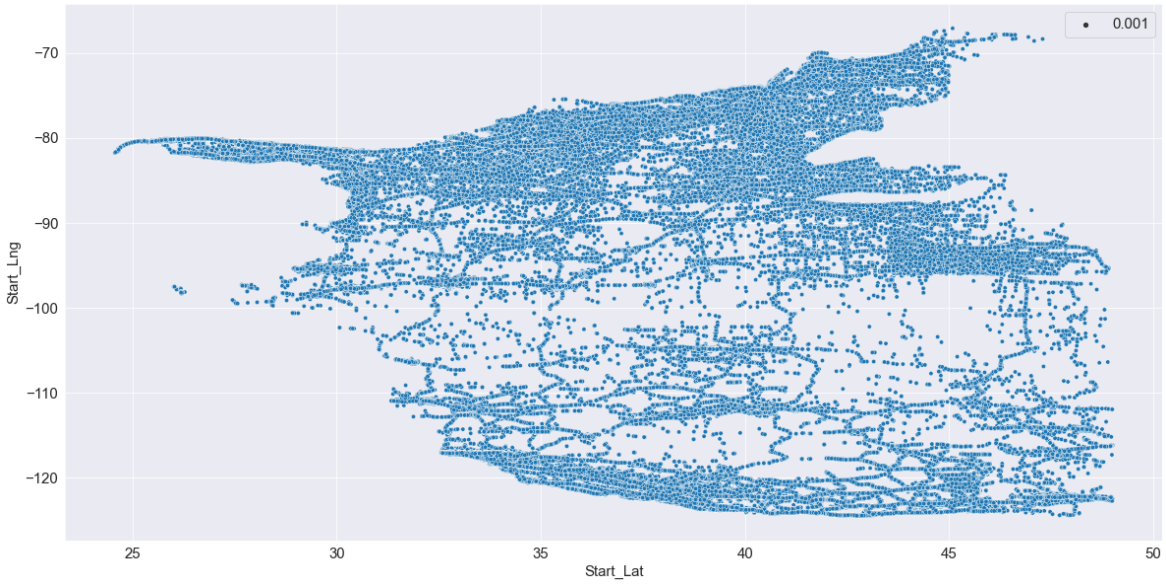
Start Latitude and Start Longitude

In [52]:
```
data.Start_Lat.head(100)#Start_Lng
```

Out[52]:
```
0     40.108910
1     39.865420
2     39.102660
3     39.101480
4     41.062130
5     39.172393
6     39.063240
7     39.067080
8     39.775650
9     41.375310
10    40.702247
11    40.109310
12    39.192880
13    39.138770
14    41.473900
15    39.582242
16    40.151785
17    40.151747
18    39.972410
```

In [53]:
```
## it gives plot wich more look like USA map
sns.scatterplot(x=data.Start_Lat, y=data.Start_Lng, size=0.001)
```

Out[53]:
```
<AxesSubplot:xlabel='Start_Lat', ylabel='Start_Lng'>
```



In [54]:
```
##!pip install folium
```

In [55]:
```
import folium.map
```

In [56]:
```
lat,lag=data.Start_Lat[0], data.Start_Lng[0]
```

In [57]:
```
lat, lag
```

Out[57]:
```
(40.108909999999995, -83.09286)
```

In [69]:
```
map=folium.Map()
marker=folium.Marker((lat,lag))
marker.add_to(map)
map
```

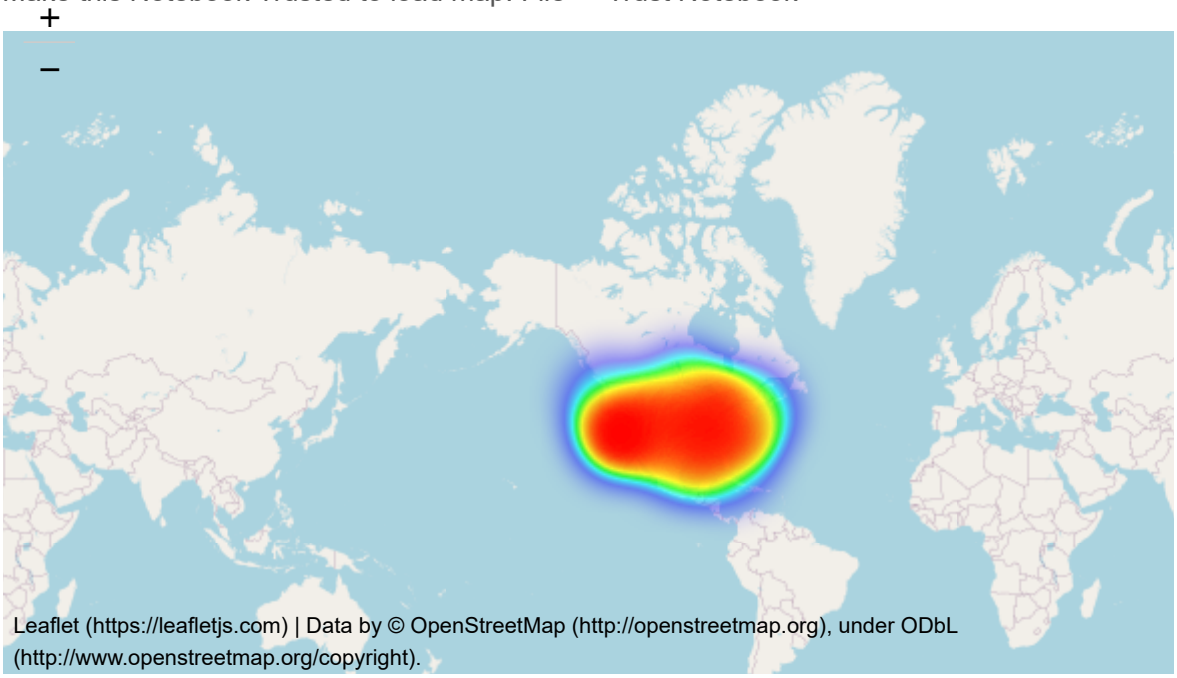
Out[69]:
Make this Notebook Trusted to load map: File -> Trust Notebook

In [59]:
```
from folium import plugins
from folium.plugins import HeatMap
```

In [60]:
```
sample_df=data.sample(n=200)
lat_long=list(zip(sample_df.Start_Lat,sample_df.Start_Lng))
```

In [68]:
```
map=folium.Map()
HeatMap(lat_long).add_to(map)
map

### zoom in to get more insight and clear visualization
```

Out[68]:
Make this Notebook Trusted to load map: File -> Trust Notebook

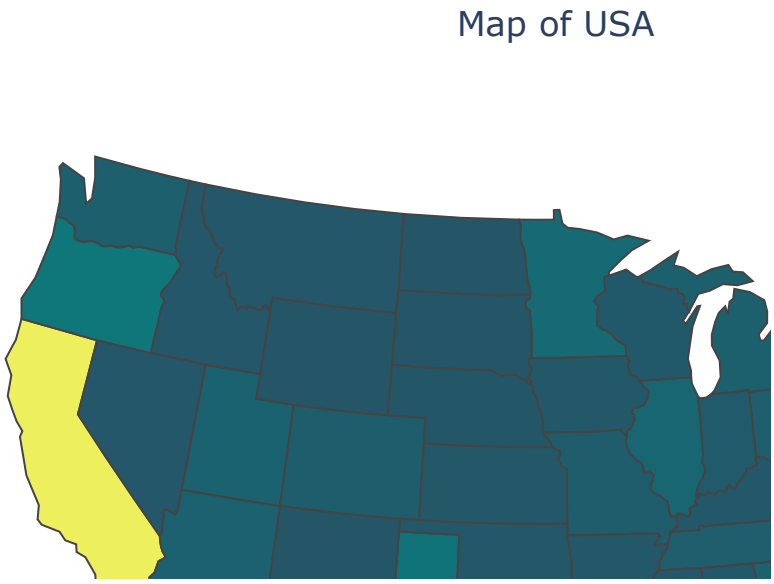


states with more number of accidents

In [62]:
```
top_state={'State':[],'city':[]}
for x in range(0,11,1):
    fltr=data.City==df1['city'][x]
    d=list(data.loc[fltr,'State'].unique())
    top_state['State'].append(d)
    top_state['city'].append(fltr)
```

In [63]:
```
df2=pd.DataFrame(top_state)
```

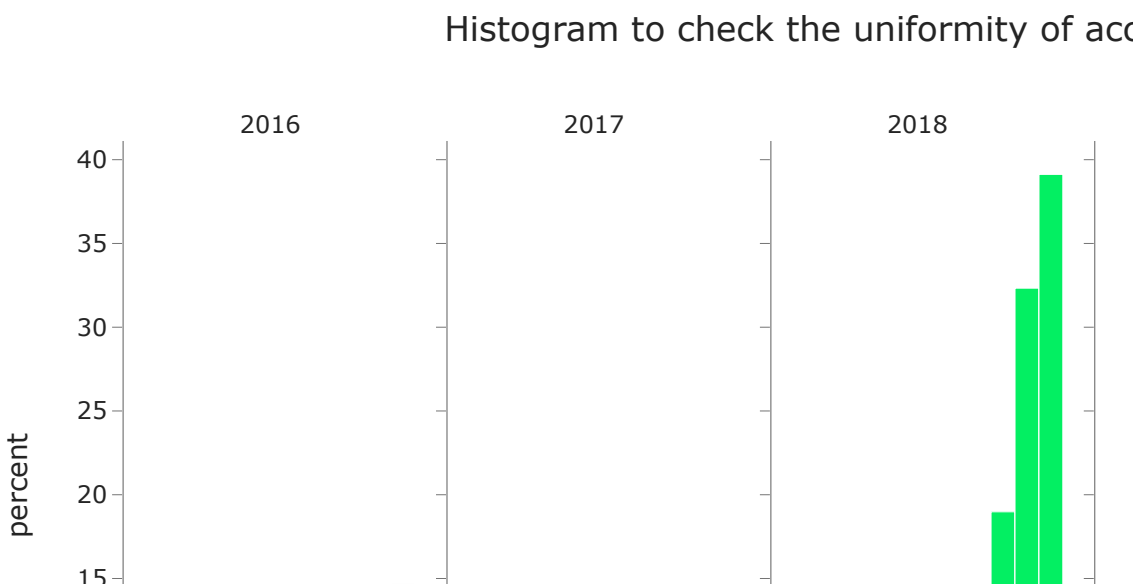
In [67]:
```
states_choropleth=px.choropleth(data_frame=data,locations=data.State.value_cc
states_choropleth.update_layout(paper_bgcolor='#ffffff',showlegend=False, tit
states_choropleth.show()
```



to check accidents occurance every year distributed over months

In [65]:

```
data['Year']=data.Start_Time.dt.year
data['Month']=data.Start_Time.dt.month
year_fig=px.histogram(data.Month,nbins=12,color_discrete_sequence=["#03ef62"]
year_fig.update_layout(paper_bgcolor='#ffffff',showlegend=False, title_x=0.5)
year_fig.for_each_annotation(lambda a: a.update(text=a.text.split("=")[-1]))
year_fig.show()
```



In [73]:
```
data.Weather_Timestamp
```

Out[73]:

	ID	Severity	Start_Time	End_Time	Start_Lat	Start_Lng	End_Lat	End_Lng	Dis
0	A-2716600	3	2016-02-08 00:37:08	2016-02-08 06:37:08	40.108910	-83.092860	40.112060	-83.031870	
1	A-2716601	2	2016-02-08 05:56:20	2016-02-08 11:56:20	39.865420	-84.062800	39.865010	-84.048730	
2	A-2716602	2	2016-02-08 06:15:39	2016-02-08 12:15:39	39.102660	-84.524680	39.102090	-84.523960	
3	A-2716603	2	2016-02-08 06:15:39	2016-02-08 12:15:39	39.101480	-84.523410	39.098410	-84.522410	
4	A-2716604	2	2016-02-08 06:51:45	2016-02-08 12:51:45	41.062130	-81.537840	41.062170	-81.535470	
5	A-2716605	3	2016-02-08 07:53:43	2016-02-08 13:53:43	39.172393	-84.492792	39.170476	-84.501798	
6	A-2716606	2	2016-02-08 08:16:57	2016-02-08 14:16:57	39.063240	-84.032430	39.067310	-84.058510	
7	A-2716607	2	2016-02-08 08:16:57	2016-02-08 14:16:57	39.067080	-84.058550	39.063020	-84.032540	
8	A-2716608	2	2016-02-08 08:15:41	2016-02-08 14:15:41	39.775650	-84.186030	39.772750	-84.188050	
9	A-2716609	2	2016-02-08 11:51:46	2016-02-08 17:51:46	41.375310	-81.820170	41.367860	-81.821740	

In [104]:
```
values=data.Wind_Direction.value_counts()

values
```

Out[104]:
```
CALM        202870
Calm         79192
WNW          77743
NW           75810
W            72059
SSW          69901
WSW          68504
NNW          68014
S            67543
SW           65626
SSE          65058
SE           54770
N            53718
E            52435
ESE          51295
ENE          51257
NE           48355
NNE          46509
West         40611
South        40596
VAR          39670
North        35568
East         24064
Variable     23038
Name: Wind_Direction, dtype: int64
```

In [105]:
```
names=[]
for x in values:
    name=values[values ==x].index[0]
    names.append(name)
```

In [106]:
```
names
```

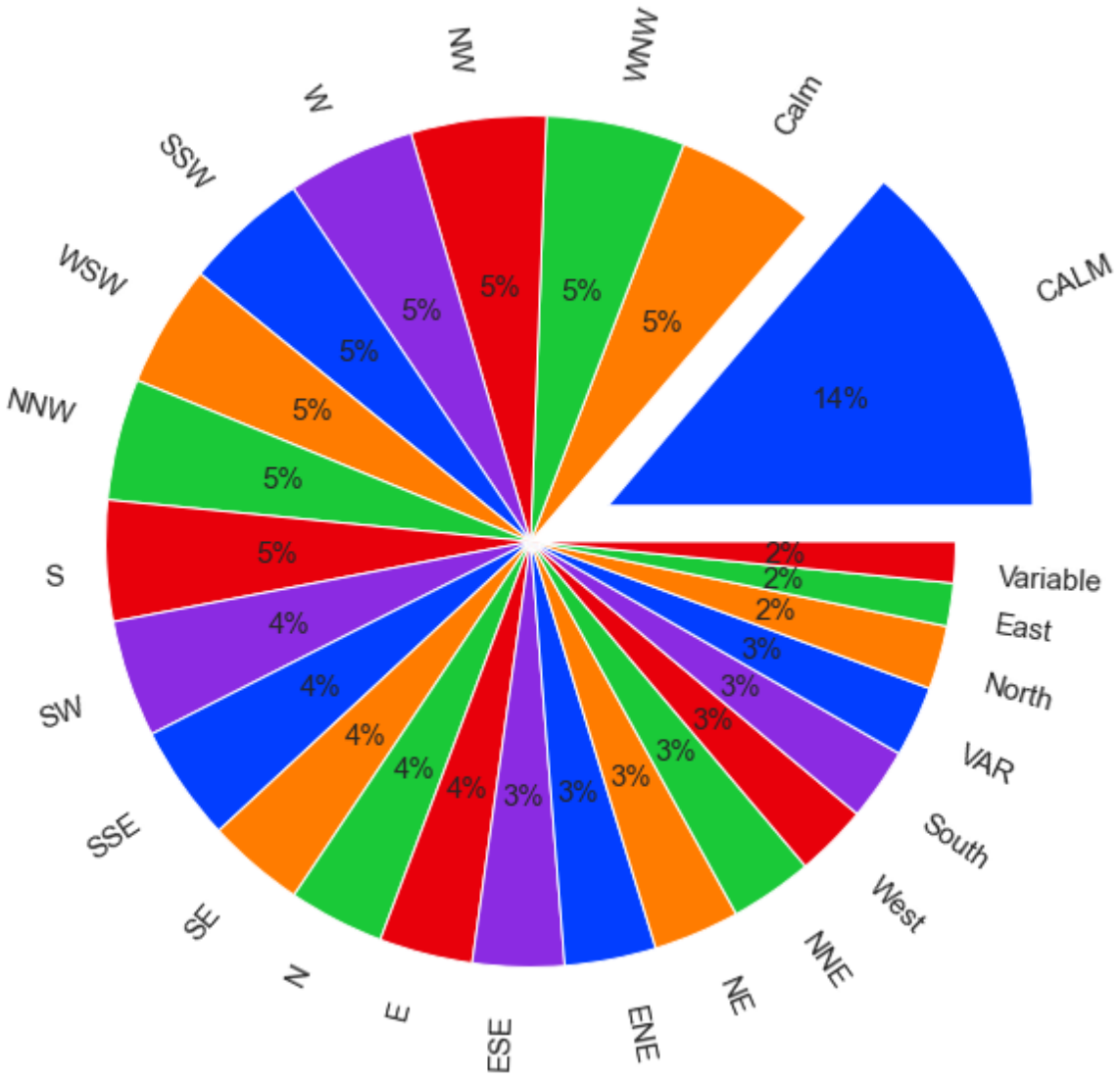
Out[106]:
```
['CALM',
 'Calm',
 'WNW',
 'NW',
 'W',
 'SSW',
 'WSW',
 'NNW',
 'S',
 'SW',
 'SSE',
 'SE',
 'N',
 'E',
 'ESE',
 'ENE',
 'NE',
 'NNE',
 'West',
```

In [126]:
```
len(names)
```

Out[126]:
```
24
```

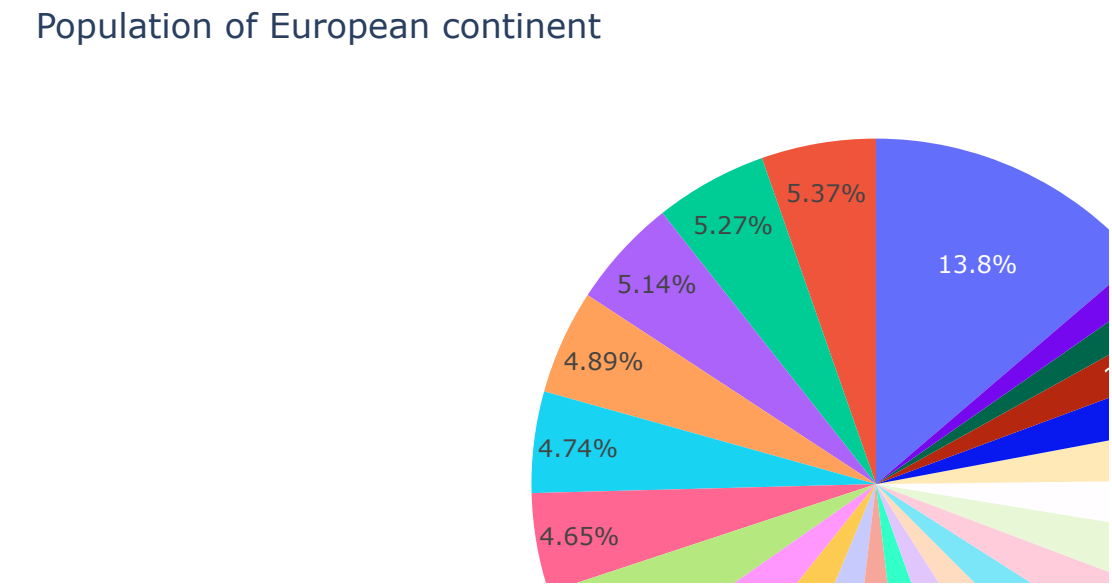
In [128]:

```
colors = sns.color_palette('bright')[0:5]
explode=(0.1,0.5)
#create pie chart
explode=(0.2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0)
plt.pie(values, labels = names, colors = colors, autopct='%.0f%%', rotatelabe
plt.show()
```



In [111]:
```
?plt.pie
```

In [108]:
```
fig = px.pie(data.Wind_Direction, values=values, names=names, title='Populati
fig.show("notebook")
```

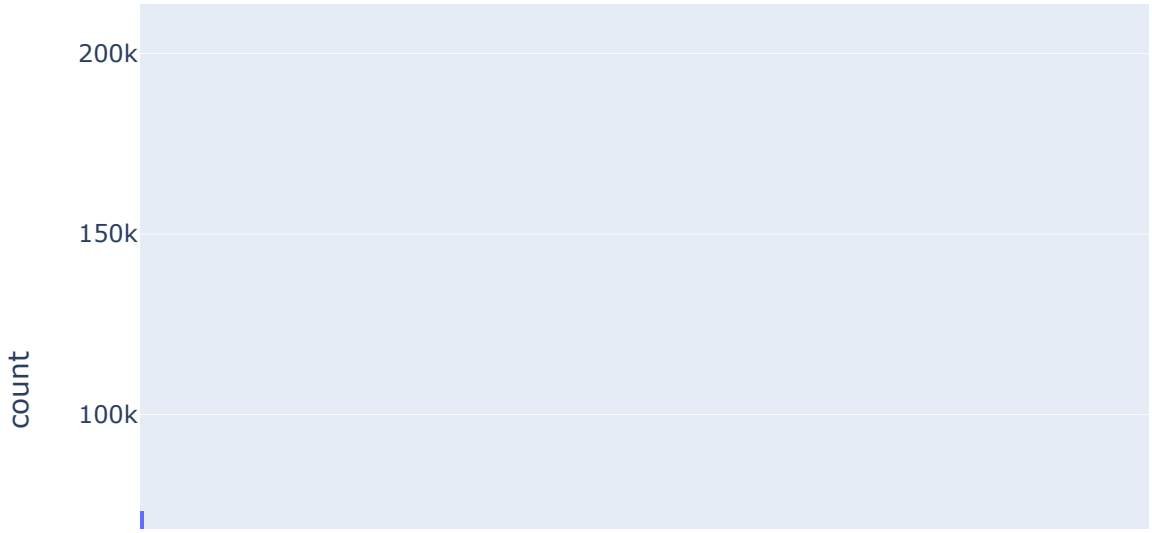


when wind_direction is 'Calm' accidents are more, we can hypothetically infer that people are more cautious during high winds, but during CALM winds there were driving fastly.

**Ps: plotly is more accurate compare to matplotlib pie charts

In [149]:
```
## calling because I kind of messed here

import os
os.listdir("./us-accidents")
data=pd.read_csv('./us-accidents/US_Accidents_Dec20_updated.csv')
```

In [168]:
```
fig=px.histogram(data['Wind_Speed(mph)'],x=data['Wind_Speed(mph)'], range_x=[
fig.show()
```



In [167]:
```
## wind speed is checked in range between 400 to 900
data[data['Wind_Speed(mph)']>=20].count()
```

Out[167]:
```
ID                      40252
Severity                40252
Start_Time              40252
End_Time                40252
Start_Lat               40252
Start_Lng               40252
End_Lat                 40252
End_Lng                 40252
Distance(mi)            40252
Description             40252
Number                  13421
Street                  40252
Side                    40252
City                    40249
County                  40252
State                   40252
Zipcode                 40252
Country                 40252
Timezone                40252
```

when wind speed is in the range between 100 to 900 there are only around 225 accidents , well this concludes that people did'nt went out during high wind speed, and it is obviouse

In [66]:
```
#data cleaning preparation
## ask question
### summary and conclusion
#### areas for feature work
```