US Accidents

Dataset Info

This is a countrywide car accident dataset, which covers 49 states of the USA. The accident data are collected from February 2016 to Dec 2020. Currently there are about 4.2 millions accident records in this dataset.

Source - Kaggle

```
# getting dataset
!pip install kaggle
!mkdir ~/.kaggle
!touch ~/.kaggle/kaggle.json

api_token = {"username":"sachinkumar20","key":"d62f0e4ff97b2bb0c32edf74105574f8"}

import json

with open('/root/.kaggle/kaggle.json', 'w') as file:
    json.dump(api_token, file)

!chmod 600 ~/.kaggle/kaggle.json
```

```
Requirement already satisfied: kaggle in /usr/local/lib/python3.7/dist-packages (1.5.10
Requirement already satisfied: certifi in /usr/local/lib/python3.7/dist-packages (from
Requirement already satisfied: python-dateutil in /usr/local/lib/python3.7/dist-package
Requirement already satisfied: urllib3 in /usr/local/lib/python3.7/dist-packages (from
Requirement already satisfied: requests in /usr/local/lib/python3.7/dist-packages (from
Requirement already satisfied: tqdm in /usr/local/lib/python3.7/dist-packages (from kag
Requirement already satisfied: six>=1.10 in /usr/local/lib/python3.7/dist-packages (fro
Requirement already satisfied: python-slugify in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dist-packa
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-packages (
Requirement already satisfied: text-unidecode>=1.3 in /usr/local/lib/python3.7/dist-pac
```

```
!kaggle datasets download -d sobhanmoosavi/us-accidents
```

```
Downloading us-accidents.zip to /content
 94% 282M/299M [00:02<00:00, 140MB/s]
100% 299M/299M [00:02<00:00, 120MB/s]
```

```
!unzip us-accidents.zip
```

```
Archive:  us-accidents.zip
  inflating: US_Accidents_Dec20.csv
```

Importing and Cleaning Data

```
# importing libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

```
us_acc = pd.read_csv('/content/US_Accidents_Dec20.csv')
us_acc.head()
```

	ID	Source	TMC	Severity	Start_Time	End_Time	Start_Lat	Start_Lng	End_Lat
0	A-1	MapQuest	201.0	3	2016-02-08 05:46:00	2016-02-08 11:00:00	39.865147	-84.058723	NaN
1	A-2	MapQuest	201.0	2	2016-02-08 06:07:59	2016-02-08 06:37:59	39.928059	-82.831184	NaN
2	A-3	MapQuest	201.0	2	2016-02-08 06:49:27	2016-02-08 07:19:27	39.063148	-84.032608	NaN
3	A-4	MapQuest	201.0	3	2016-02-08 07:23:34	2016-02-08 07:53:34	39.747753	-84.205582	NaN
4	A-5	MapQuest	201.0	2	2016-02-08 07:39:07	2016-02-08 08:09:07	39.627781	-84.188354	NaN

Knowing data

```
# Columns
us_acc.columns
```

```
Index(['ID', 'Source', 'TMC', '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')
```

```
# Size of data
len(us_acc)
```

```
4232541
```

```
# Data types of columns
us_acc.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4232541 entries, 0 to 4232540
Data columns (total 49 columns):
 #   Column                 Dtype  
---  ------                 -----  
 0   ID                     object 
 1   Source                 object 
 2   TMC                    float64
 3   Severity               int64  
 4   Start_Time             object 
 5   End_Time               object 
 6   Start_Lat              float64
 7   Start_Lng              float64
 8   End_Lat                float64
 9   End_Lng                float64
 10  Distance(mi)           float64
 11  Description            object 
 12  Number                 float64
 13  Street                 object 
 14  Side                   object 
 15  City                   object 
 16  County                 object 
 17  State                  object 
 18  Zipcode                object 
 19  Country                object 
 20  Timezone               object 
 21  Airport_Code           object 
 22  Weather_Timestamp      object 
 23  Temperature(F)         float64
 24  Wind_Chill(F)          float64
 25  Humidity(%)            float64
 26  Pressure(in)           float64
 27  Visibility(mi)         float64
 28  Wind_Direction         object 
 29  Wind_Speed(mph)        float64
```

```
 30  Precipitation(in)      float64
 31  Weather_Condition      object
 32  Amenity                bool
 33  Bump                   bool
 34  Crossing               bool
 35  Give_Way               bool
 36  Junction               bool
 37  No_Exit                bool
 38  Railway                bool
 39  Roundabout             bool
 40  Station                bool
 41  Stop                   bool
 42  Traffic_Calming        bool
 43  Traffic_Signal         bool
 44  Turning_Loop           bool
 45  Sunrise_Sunset         object
 46  Civil_Twilight         object
 47  Nautical_Twilight      object
 48  Astronomical_Twilight  object
dtypes: bool(13), float64(14), int64(1), object(21)
memory usage: 1.2+ GB
```

```
# getting info about numerical columns
us_acc.describe()
```

	TMC	Severity	Start_Lat	Start_Lng	End_Lat	End_Lr
count	2.716477e+06	4.232541e+06	4.232541e+06	4.232541e+06	1.516064e+06	1.516064e+0
mean	2.083517e+02	2.305035e+00	3.639782e+01	-9.546420e+01	3.690061e+01	-9.859901e+0
std	2.124413e+01	5.332261e-01	4.964404e+00	1.735319e+01	5.165629e+00	1.849590e+0
min	2.000000e+02	1.000000e+00	2.455527e+01	-1.246238e+02	2.457011e+01	-1.244978e+0
25%	2.010000e+02	2.000000e+00	3.352058e+01	-1.173570e+02	3.385420e+01	-1.182077e+0
50%	2.010000e+02	2.000000e+00	3.582542e+01	-9.002078e+01	3.735134e+01	-9.437987e+0
75%	2.010000e+02	3.000000e+00	4.018313e+01	-8.084682e+01	4.072593e+01	-8.087449e+0
max	4.060000e+02	4.000000e+00	4.900220e+01	-6.711317e+01	4.907500e+01	-6.710924e+0

Cleaning data

```
# getting count of null values in columns
null_counts = us_acc.isna().sum().sort_values(ascending=False)
null_counts[null_counts > 0]
```

```
End_Lat                 2716477
End_Lng                 2716477
Number                  2687949
Precipitation(in)       2065589
Wind_Chill(F)           1896001
TMC                     1516064
Wind_Speed(mph)          479326
Visibility(mi)            98668
Weather_Condition         98383
Humidity(%)               95467
Temperature(F)            89900
Wind_Direction            83611
Pressure(in)              76384
Weather_Timestamp         62644
Airport_Code               8973
Timezone                   4615
Zipcode                    1292
Nautical_Twilight           141
Astronomical_Twilight       141
Civil_Twilight              141
Sunrise_Sunset              141
City                        137
Description                   2
dtype: int64
```

Data Analysis and Visualization

Columns we will be using for analysis

1. State
2. City
3. Start time
4. Weather condition
5. Temperature
6. Severity
7. Other Factors such as Bump, Crossing, No_Exit, Roundabout etc.

State

```
# State
accidents_by_state = us_acc['State'].value_counts()
accidents_by_state.head()
```

```
CA    972585
```

```
TX    376445
FL    370131
SC    212712
NC    193457
Name: State, dtype: int64
```

```
# adding ranks to state
def get_rank(state):
  value = accidents_by_state[state]
  accidents_by_state_rank = pd.Index(accidents_by_state)
  rank = accidents_by_state_rank.get_loc(value)
  print(rank)
```

```
get_rank('NY')
```

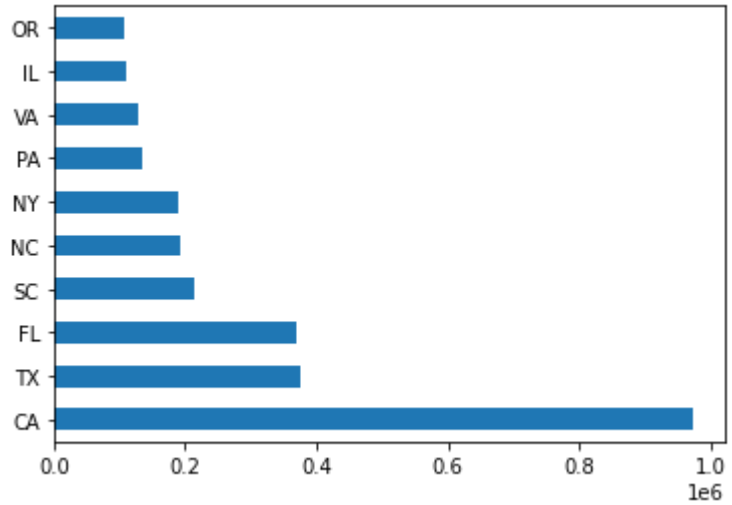
```
5
```

```
accidents_by_state[:10]
```

```
CA    972585
TX    376445
FL    370131
SC    212712
NC    193457
NY    189513
PA    136049
VA    127949
IL    111712
OR    108352
Name: State, dtype: int64
```

```
accidents_by_state[:10].plot(kind='barh')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f497811da50>
```



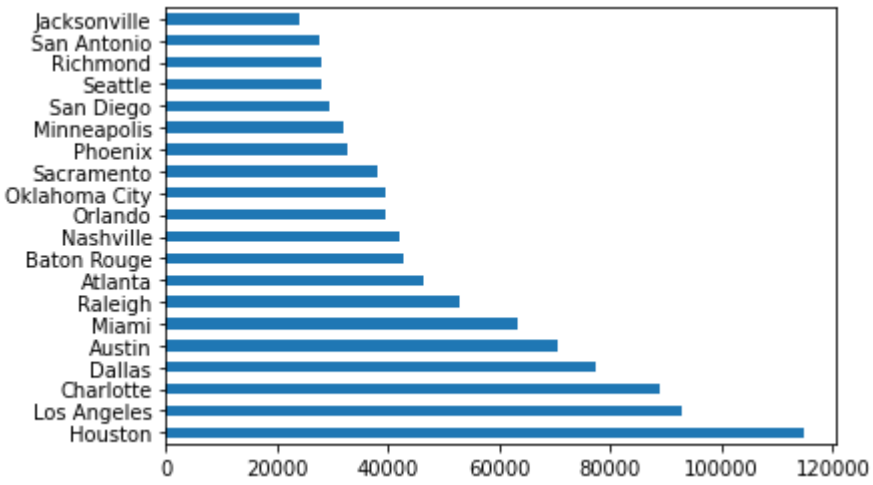
City

```
# City
accidents_by_city = us_acc['City'].value_counts()
accidents_by_city[:20]
```

```
Houston          114905
Los Angeles       92701
Charlotte         88887
Dallas            77303
Austin            70538
Miami             63162
Raleigh           52876
Atlanta           46328
Baton Rouge       42814
Nashville         41850
Orlando           39561
Oklahoma City     39484
Sacramento        38061
Phoenix           32805
Minneapolis       31781
San Diego         29416
Seattle           28004
Richmond          27907
San Antonio       27516
Jacksonville      24009
Name: City, dtype: int64
```

```
# getting a graph
accidents_by_city[:20].plot(kind='barh')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f48fc9094d0>
```



NY Details

```
# Check if NY in States:
'NY' in list(us_acc['State'])
```

```
True
```

```
us_acc['State'].unique()
```

```
array(['OH', 'WV', 'CA', 'FL', 'GA', 'SC', 'NE', 'IA', 'IL', 'MO', 'WI',
       'IN', 'MI', 'NJ', 'NY', 'CT', 'MA', 'RI', 'NH', 'PA', 'KY', 'MD',
       'VA', 'DC', 'DE', 'TX', 'WA', 'OR', 'AL', 'ME', 'VT', 'TN', 'NC',
       'KS', 'MN', 'LA', 'OK', 'AR', 'UT', 'CO', 'AZ', 'NM', 'NV', 'MS',
       'ID', 'WY', 'MT', 'ND', 'SD'], dtype=object)
```

```
# Check if New York in cities
'New York' in list(us_acc['City'])
```

```
True
```

```
# Check where New York ranks in cities
x = list(accidents_by_city)
# Check again
```

Weather

```
# Weather
weather_values = us_acc['Weather_Condition'].value_counts()
weather_values[:10]
```

```
Fair                900872
Clear               808181
Mostly Cloudy       571743
Partly Cloudy       397415
Overcast            382485
Cloudy              323340
Light Rain          206389
Scattered Clouds    204661
Light Snow           57148
Rain                 48641
Name: Weather_Condition, dtype: int64
```

Temperature(F)

```
# Temperature
temperature_values = us_acc['Temperature(F)'].value_counts()
```

```
temperature_values[:10]
```

```
68.0    91545
77.0    90041
59.0    86347
73.0    84212
```

```
63.0    80072
        ...
96.1     3041
17.1     2993
12.0     2883
20.0     2625
10.0     2558
Name: Temperature(F), Length: 153, dtype: int64
```

```
# getting numerical extracts from temperature
us_acc['Temperature(F)'].describe()
```

```
count    4.142641e+06
mean     6.147799e+01
std      1.852586e+01
min     -8.900000e+01
25%      4.900000e+01
50%      6.300000e+01
75%      7.520000e+01
max      2.030000e+02
Name: Temperature(F), dtype: float64
```

75 % accidents showing below 75 F temperature

```
max_temp = us_acc['Temperature(F)'].max()
us_acc[us_acc['Temperature(F)'] == max_temp]
```

	ID	Source	TMC	Severity	Start_Time	End_Time	Start_Lat	Start_Lng
694282	A-694311	MapQuest	201.0	3	2020-07-17 11:41:59	2020-07-17 12:26:37	32.670929	-97.062508

```
min_temp = us_acc['Temperature(F)'].min()
us_acc[us_acc['Temperature(F)'] == min_temp]
```

	ID	Source	TMC	Severity	Start_Time	End_Time	Start_Lat	Start_L
849453	A-849533	MapQuest	343.0	3	2020-04-03 07:37:17	2020-04-03 08:10:00	39.779041	-104.9900
849455	A-849535	MapQuest	343.0	3	2020-04-03 07:44:19	2020-04-03 08:34:00	39.778301	-104.8992
849457	A-849537	MapQuest-Bing	201.0	2	2020-04-03 07:50:12	2020-04-03 09:36:00	39.741428	-104.9988
849459	A-849539	MapQuest	201.0	2	2020-04-03 07:58:54	2020-04-03 09:00:00	39.725639	-105.0817
849460	A-849540	MapQuest	201.0	2	2020-04-03 07:59:49	2020-04-03 09:01:00	39.785191	-105.0817
849462	A-849542	MapQuest	201.0	3	2020-04-03 08:05:15	2020-04-03 08:51:00	39.767784	-104.9954
849463	A-849543	MapQuest	201.0	2	2020-04-03 08:10:07	2020-04-03 09:11:00	39.779591	-104.9882
849464	A-849544	MapQuest	201.0	2	2020-04-03 08:15:53	2020-04-03 09:17:00	39.733109	-105.0532

Start Time

3610682	3617506	Bing	NaN	2	[illegible]	03	39.729290	-105.0252

```
type(us_acc['Start_Time'][0])
```

```
str
```

3610688	3617512	Bing	NaN	3	07:56:58	03	39.778300	-104.9060

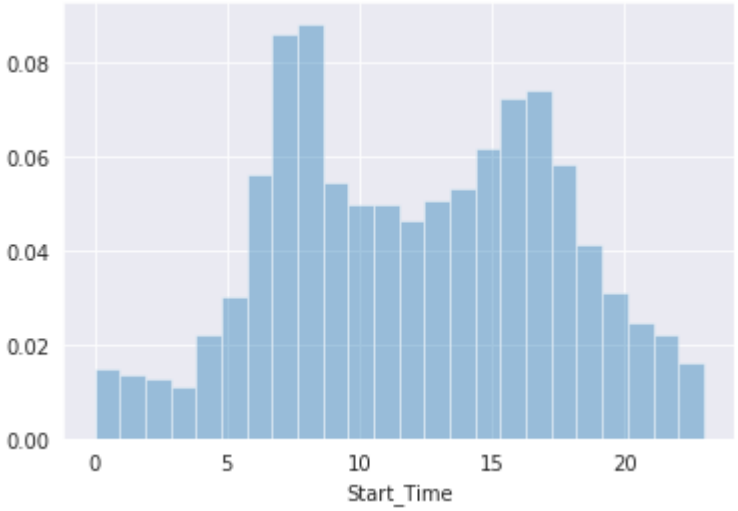
```
us_acc['Start_Time'] = pd.to_datetime(us_acc['Start_Time'])
```

```
type(us_acc['Start_Time'][0])
```

```
pandas._libs.tslibs.timestamps.Timestamp
```

```
# get graph for accidents at what time of day
sns.set_style('darkgrid')
sns.distplot(us_acc['Start_Time'].dt.hour, bins=24, kde=False, norm_hist=True)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2557: FutureWarning: `d
  warnings.warn(msg, FutureWarning)
<matplotlib.axes._subplots.AxesSubplot at 0x7f48fc84aad0>
```

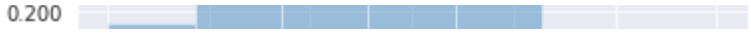


Most accidents seems to be between 8AM to 10AM, then 3PM to 6PM

```
# get graph for accidents on week of the day
sns.distplot(us_acc['Start_Time'].dt.weekday, bins=7, kde=False, norm_hist=True)
```

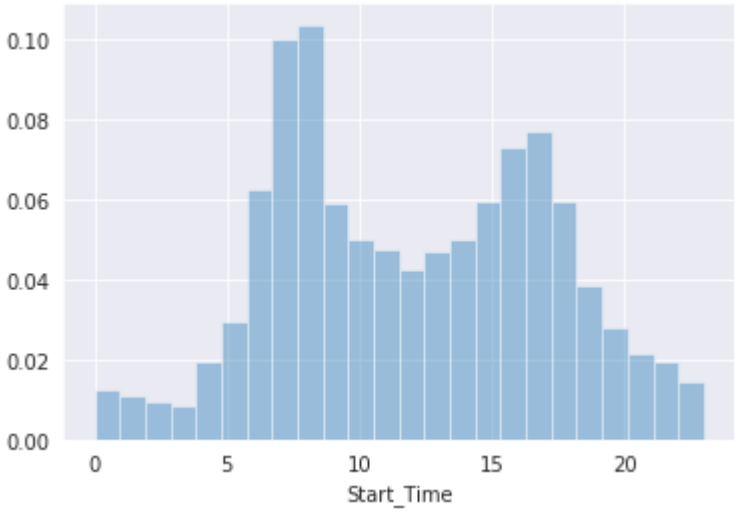
```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2557: FutureWarning: `d
  warnings.warn(msg, FutureWarning)
```

Seems less accidents on weekend



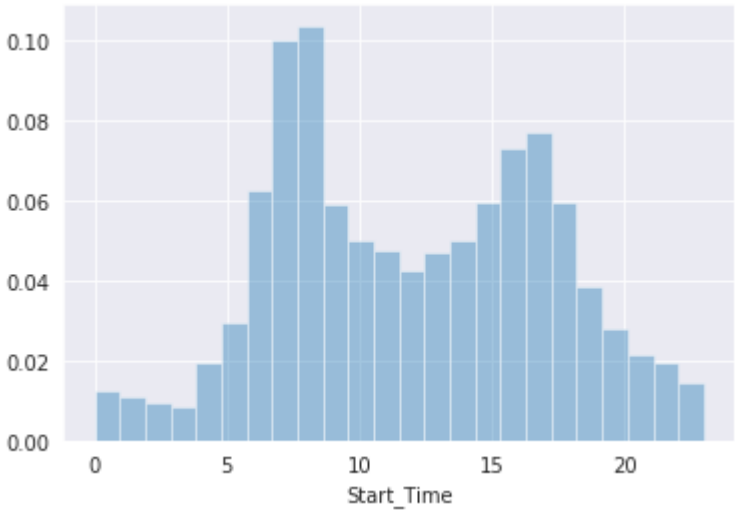

```
# Check day time of accidents on weekdays comparitively to weekend
monday_data = us_acc[us_acc['Start_Time'].dt.dayofweek == 1 ]
sns.distplot(monday_data['Start_Time'].dt.hour, bins=24, kde=False, norm_hist=True)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2557: FutureWarning: `d
  warnings.warn(msg, FutureWarning)
<matplotlib.axes._subplots.AxesSubplot at 0x7f48fc7d6910>
```




```
monday_data = us_acc[us_acc['Start_Time'].dt.dayofweek == 1 ]
sns.distplot(monday_data['Start_Time'].dt.hour, bins=24, kde=False, norm_hist=True)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2557: FutureWarning: `d
  warnings.warn(msg, FutureWarning)
<matplotlib.axes._subplots.AxesSubplot at 0x7f48fc78cc10>
```

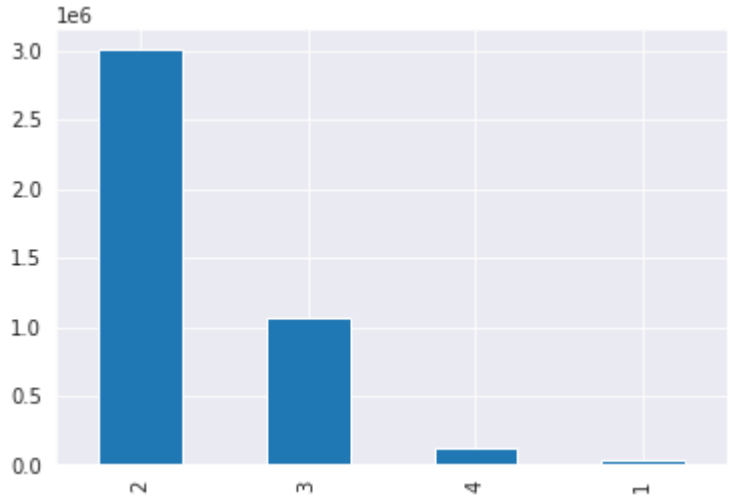


Change in time of accidents on weekdays and weekend

Severity

```
# How severe were accidents
severity_types = us_acc['Severity'].value_counts()
severity_types.plot(kind='bar')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f48fc6ea050>
```



Other Factors

```
us_acc.columns
```

```
Index(['ID', 'Source', 'TMC', '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')
```

```
# Getting other factors that could relate to severity
other_factors = ['Severity', 'Bump', 'Crossing', 'Give_Way', 'Junction', 'No_Exit', 'Roundabc
other_factors_data = us_acc[other_factors]
other_factors_data.head()
```

	Severity	Bump	Crossing	Give_Way	Junction	No_Exit	Roundabout	Stop	Traffic_Si
0	3	False	False	False	False	False	False	False	F
1	2	False	False	False	False	False	False	False	F
2	2	False	False	False	False	False	False	False	

```
other_factors_data.groupby('Severity').sum()
```

Severity	Bump	Crossing	Give_Way	Junction	No_Exit	Roundabout	Stop	Traffic_Signa
1	10	9155	96	2537	124	1	526	1266
2	723	309628	10348	212039	4664	228	65723	63732
3	116	22601	2309	111055	1033	3	4126	6603
4	12	6033	442	14011	148	5	1808	1209

```
# checking for turning loop values
us_acc['Turning_Loop'].value_counts()
```

```
False    4232541
Name: Turning_Loop, dtype: int64
```

```
# importing geopandas
!pip install geopandas
import geopandas as gpd
```

```
Collecting geopandas
  Downloading https://files.pythonhosted.org/packages/d7/bf/e9cefb69d39155d122b6ddca538
     |████████████████████████████████| 1.0MB 8.3MB/s
Collecting pyproj>=2.2.0
  Downloading https://files.pythonhosted.org/packages/b1/72/d52e9ca81caef056062d71991b0
     |████████████████████████████████| 6.5MB 22.3MB/s
Requirement already satisfied: pandas>=0.24.0 in /usr/local/lib/python3.7/dist-packages
Collecting fiona>=1.8
  Downloading https://files.pythonhosted.org/packages/47/c2/67d1d0acbaaee3b03e5e22e3b96
     |████████████████████████████████| 14.8MB 309kB/s
Requirement already satisfied: shapely>=1.6 in /usr/local/lib/python3.7/dist-packages (
Requirement already satisfied: certifi in /usr/local/lib/python3.7/dist-packages (from
Requirement already satisfied: python-dateutil>=2.7.3 in /usr/local/lib/python3.7/dist-
Requirement already satisfied: numpy>=1.15.4 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: pytz>=2017.2 in /usr/local/lib/python3.7/dist-packages (
Requirement already satisfied: attrs>=17 in /usr/local/lib/python3.7/dist-packages (fro
Requirement already satisfied: six>=1.7 in /usr/local/lib/python3.7/dist-packages (from
Collecting munch
  Downloading https://files.pythonhosted.org/packages/cc/ab/85d8da5c9a45e072301beb37ad7
Collecting click-plugins>=1.0
  Downloading https://files.pythonhosted.org/packages/e9/da/824b92d9942f4e4727024888579
Collecting cligj>=0.5
```

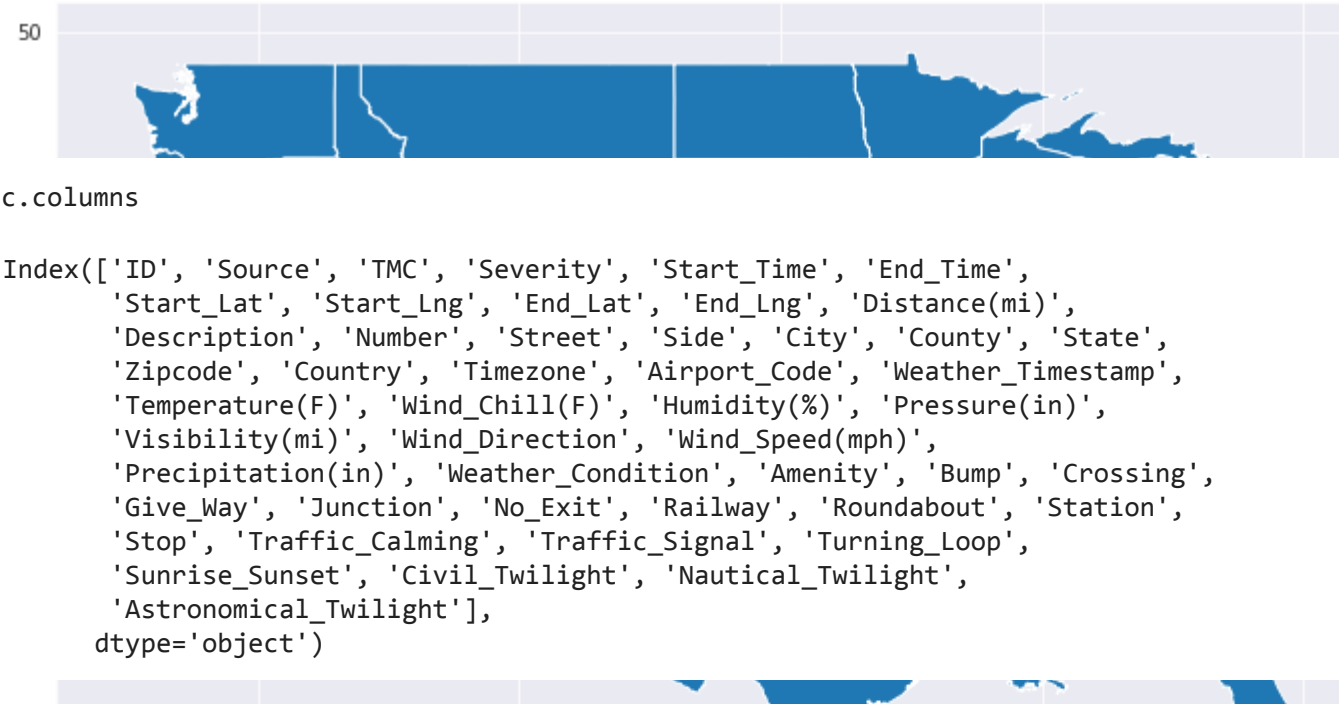
```
  Downloading https://files.pythonhosted.org/packages/42/1e/947eadf10d6804bf276eb8a038b
Requirement already satisfied: click<8,>=4.0 in /usr/local/lib/python3.7/dist-packages
Installing collected packages: pyproj, munch, click-plugins, cligj, fiona, geopandas
Successfully installed click-plugins-1.1.1 cligj-0.7.1 fiona-1.8.18 geopandas-0.9.0 mun
```

```
us_shape = gpd.read_file('gz_2010_us_040_00_500k.shp')
us_shape.head()
```

	GEO_ID	STATE	NAME	LSAD	CENSUSAREA	geometry
0	0400000US23	23	Maine	None	30842.923	MULTIPOLYGON (((-67.61976 44.51975, -67.61541 ...
1	0400000US25	25	Massachusetts	None	7800.058	MULTIPOLYGON (((-70.83204 41.60650, -70.82373 ...
2	0400000US26	26	Michigan	None	56538.901	MULTIPOLYGON (((-88.68443 48.11578, -88.67563 ...
3	0400000US30	30	Montana	None	145545.801	POLYGON ((-104.05770 44.99743,

```
us_shape[us_shape['NAME'].isin(['Alaska','Hawaii']) == False].plot(figsize=(30,10))
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f48ea02c5d0>
```




```
us_acc.columns
```

```
Index(['ID', 'Source', 'TMC', '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')
```

```
us_acc[['Start_Lat','Start_Lng']].head()
```

	Start_Lat	Start_Lng
0	39.865147	-84.058723
1	39.928059	-82.831184
2	39.063148	-84.032608
3	39.747753	-84.205582
4	39.627781	-84.188354

```
us_acc['coordinates'] = us_acc[['Start_Lng','Start_Lat']].values.tolist()
us_acc['coordinates'].head()
```

```
/usr/local/lib/python3.7/dist-packages/geopandas/geodataframe.py:1321: UserWarning: Geo
  warnings.warn("Geometry column does not contain geometry.")
0             [-84.058723, 39.865147]
1    [-82.831184, 39.928059000000005]
2             [-84.032608, 39.063148]
3     [-84.20558199999999, 39.747753]
4             [-84.188354, 39.627781]
Name: coordinates, dtype: object
```

```
from shapely.geometry import Point
us_acc['coordinates'] = us_acc['coordinates'].apply(Point)
```

```
us_acc = gpd.GeoDataFrame(us_acc, geometry='coordinates')
```

```
us_acc.head()
```

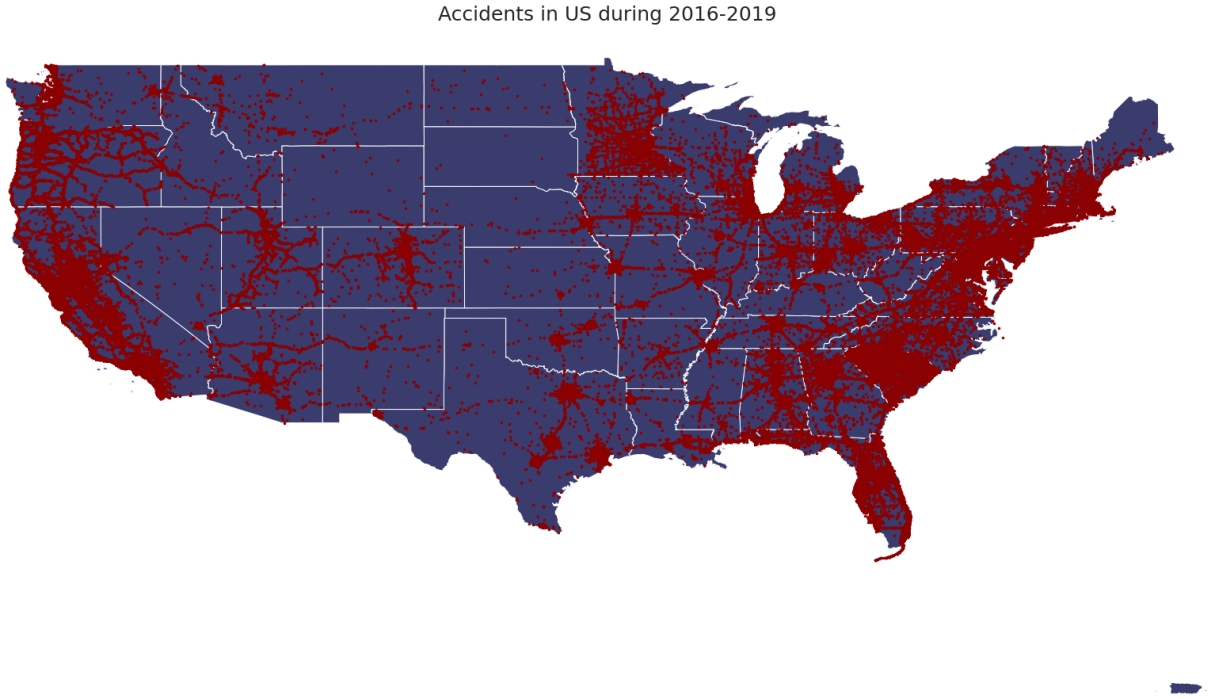
	ID	Source	TMC	Severity	Start_Time	End_Time	Start_Lat	Start_Lng	End_Lat
0	A-1	MapQuest	201.0	3	2016-02-08 05:46:00	2016-02-08 11:00:00	39.865147	-84.058723	NaN
1	A-2	MapQuest	201.0	2	2016-02-08 06:07:59	2016-02-08 06:37:59	39.928059	-82.831184	NaN
2	A-3	MapQuest	201.0	2	2016-02-08 06:49:27	2016-02-08 07:19:27	39.063148	-84.032608	NaN
3	A-4	MapQuest	201.0	3	2016-02-08 07:23:34	2016-02-08 07:53:34	39.747753	-84.205582	NaN
4	A-5	MapQuest	201.0	2	2016-02-08 07:39:07	2016-02-08 08:09:07	39.627781	-84.188354	NaN

```
us_acc.sample(100000).plot(figsize=(20,10));
```


```
fig, ax = plt.subplots(1, figsize=(30,20))
base = us_shape[us_shape['NAME'].isin(['Alaska','Hawaii']) == False].plot(ax=ax, color='#3B3C

us_acc.sample(500000).plot(ax=base, color='darkred', marker="*", markersize=10);
_ = ax.axis('off')
ax.set_title("Accidents in US during 2016-2019", fontsize=25)
plt.savefig('US_accidents.png',bbox_inches='tight');
```

```
No handles with labels found to put in legend.
```



Questions to Answer

1. Are there more accidents in colder or warmer areas?
2. Which 5 states/cities have highest accidents?
3. What is the rank of New York in accidents?
4. What are the weather condition fro most of accidents?
5. At what time of day most accidents happen?
6. On which day of week most accidents happen?
7. What are other factors common in accidents?
8. Factors affecting accident severity?