EDA and Prediction of US Accidents

Team 5: Jayatha Chandra, Avanti Dorle, Lavina Omprakash Talreja, Mansi Pravin Thanki

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

Importing data from: https://www.kaggle.com/datasets/sobhanmoosavi/us-accidents (https://www.kaggle.com/datasets/sobhanmoosavi/us-accidents)

In [2]:
```
accidents_dataset = pd.read_csv("US_Accidents_Dec21_updated.csv", na_valu
display(accidents_dataset.head())
```

	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

5 rows × 47 columns

Let us check out all the columns of this data.

In [3]:
```
accidents_dataset.columns
```

Out[3]:
```
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', 'Timezo
ne',
       'Airport_Code', 'Weather_Timestamp', 'Temperature(F)', 'Wind_Chi
ll(F)',
       'Humidity(%)', 'Pressure(in)', 'Visibility(mi)', 'Wind_Directio
n',
       'Wind_Speed(mph)', 'Precipitation(in)', 'Weather_Condition', 'Am
enity',
       'Bump', 'Crossing', 'Give_Way', 'Junction', 'No_Exit', 'Railwa
y',
       'Roundabout', 'Station', 'Stop', 'Traffic_Calming', 'Traffic_Sig
nal',
       'Turning_Loop', 'Sunrise_Sunset', 'Civil_Twilight', 'Nautical_Tw
ilight',
       'Astronomical_Twilight'],
      dtype='object')
```

In [4]:
```
accidents_dataset.info()
```

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

We can see there are a lot of object/categorical variables in the data. For now we will clean this data and use it for EDA.

For data cleaning part, we have decided to split equal columns with all 4 members so each one gets to clean the data. Once all the data is clean, we will merge everything and use it for EDA purpose.

Part A contains columns from index 0 to 11

In [5]:
```
# Part A columns 0-10

partA = accidents_dataset.iloc[:, 0 : 11]
partA.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2845342 entries, 0 to 2845341
Data columns (total 11 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
dtypes: float64(6), int64(1), object(4)
memory usage: 238.8+ MB
```

In [6]:
```
partA.isnull().sum()
```

Out[6]:
```
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         1743911
dtype: int64
```

We can see the entire Number column contains all null values and is not useful so we will drop this.

In [7]:
```
partA = partA.drop('Number', axis=1)
```

In [8]:
```
partA["Description"].fillna(value="", inplace=True)
```

PartB contains columns from index 11 to 23. We have kept the primary key ID intact because it will be required during merging.

In [9]:
```
# Columns 11 to 22 being handled
# Select the desired columns
first_col = accidents_dataset.iloc[:, 0]  # First column
middle_cols = accidents_dataset.iloc[:, 11 : 23]  # Middle columns (colum

partB = pd.concat([first_col, middle_cols], axis=1)
partB.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2845342 entries, 0 to 2845341
Data columns (total 13 columns):
 #   Column             Dtype  
---  ------             -----  
 0   ID                 object 
 1   Street             object 
 2   Side               object 
 3   City               object 
 4   County             object 
 5   State              object 
 6   Zipcode            object 
 7   Country            object 
 8   Timezone           object 
 9   Airport_Code       object 
 10  Weather_Timestamp  object 
 11  Temperature(F)     float64
 12  Wind_Chill(F)      float64
dtypes: float64(2), object(11)
memory usage: 282.2+ MB
```

In [10]:
```
partB
```

Out[10]:

	ID	Street	Side	City	County	State	Zipcode	Country	Timezone	A
0	A-1	Outerbelt E	R	Dublin	Franklin	OH	43017	US	US/Eastern	
1	A-2	I-70 E	R	Dayton	Montgomery	OH	45424	US	US/Eastern	
2	A-3	I-75 S	R	Cincinnati	Hamilton	OH	45203	US	US/Eastern	
3	A-4	I-77 N	R	Akron	Summit	OH	44311	US	US/Eastern	
4	A-5	I-75 S	R	Cincinnati	Hamilton	OH	45217	US	US/Eastern	
...	...	...	...	...	...	...	...	...	...	
2845337	A-2845338	Pomona Fwy E	R	Riverside	Riverside	CA	92501	US	US/Pacific	
2845338	A-2845339	I-8 W	R	San Diego	San Diego	CA	92108	US	US/Pacific	
2845339	A-2845340	Garden Grove Fwy	R	Orange	Orange	CA	92866	US	US/Pacific	
2845340	A-2845341	San Diego Fwy S	R	Culver City	Los Angeles	CA	90230	US	US/Pacific	
2845341	A-2845342	CA-210 W	R	Highland	San Bernardino	CA	92346	US	US/Pacific	

2845342 rows × 13 columns

In [11]:
```
partB.describe()
```

Out[11]:

	Temperature(F)	Wind_Chill(F)
count	2.776068e+06	2.375699e+06
mean	6.179356e+01	5.965823e+01
std	1.862263e+01	2.116097e+01
min	-8.900000e+01	-8.900000e+01
25%	5.000000e+01	4.600000e+01
50%	6.400000e+01	6.300000e+01
75%	7.600000e+01	7.600000e+01
max	1.960000e+02	1.960000e+02

In [12]:
```
partB.isna().sum()
```

Out[12]:
```
ID                        0
Street                    2
Side                      0
City                    137
County                    0
State                     0
Zipcode                1319
Country                   0
Timezone               3659
Airport_Code           9549
Weather_Timestamp     50736
Temperature(F)        69274
Wind_Chill(F)        469643
dtype: int64
```

We can observe there are many NA values in these columns. For numerical, we will impute NA values with mean of the column and for other categorical columns, we are replacing it with mode of the column.

In [13]:
```
partB.head()
```

Out[13]:

	ID	Street	Side	City	County	State	Zipcode	Country	Timezone	Airport_Code
0	A-1	Outerbelt E	R	Dublin	Franklin	OH	43017	US	US/Eastern	KOSU
1	A-2	I-70 E	R	Dayton	Montgomery	OH	45424	US	US/Eastern	KFFO
2	A-3	I-75 S	R	Cincinnati	Hamilton	OH	45203	US	US/Eastern	KLUK
3	A-4	I-77 N	R	Akron	Summit	OH	44311	US	US/Eastern	KAKR
4	A-5	I-75 S	R	Cincinnati	Hamilton	OH	45217	US	US/Eastern	KLUK

In [14]:
```
# Imputing numerical columns NA with mean.

partB["Wind_Chill(F)"].fillna(value=partB["Wind_Chill(F)"].mean(), inplac
partB["Temperature(F)"].fillna(value=partB["Temperature(F)"].mean(), inpl
partB.isna().sum()
```

Out[14]:
```
ID                      0
Street                  2
Side                    0
City                  137
County                  0
State                   0
Zipcode              1319
Country                 0
Timezone             3659
Airport_Code         9549
Weather_Timestamp    50736
Temperature(F)           0
Wind_Chill(F)            0
dtype: int64
```

In [15]:
```
# Imputing categorical/object NA data with mode of the column.

partB["Airport_Code"].fillna(value=partB["Airport_Code"].mode()[0], inpla
partB["Timezone"].fillna(value=partB["Timezone"].mode()[0], inplace=True)
partB["City"].fillna(value=partB["City"].mode()[0], inplace=True)
partB["Zipcode"].fillna(value=partB["Zipcode"].mode()[0], inplace=True)
partB["Weather_Timestamp"].fillna(value=partB["Weather_Timestamp"].mode()
partB["Street"].fillna(partB["Street"].mode()[0], inplace=True)
partB["Side"].fillna(partB["Side"].mode()[0], inplace=True)
partB["County"].fillna(partB["County"].mode()[0], inplace=True)
partB["State"].fillna(partB["State"].mode()[0], inplace=True)
partB["Country"].fillna(partB["Country"].mode()[0], inplace=True)
partB.isna().sum()
```

Out[15]:
```
ID                    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
dtype: int64
```

We can see there are no null values anymore in the dataset.

PartC will handle the columns from index 23 to 34.

In [16]:
```
# Part C columns 23-33

first_col = accidents_dataset.iloc[:, 0]  # First column
middle_cols = accidents_dataset.iloc[:, 23 : 34]  # Middle columns (colum

partC = pd.concat([first_col, middle_cols], axis=1)
partC.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2845342 entries, 0 to 2845341
Data columns (total 12 columns):
 #   Column             Dtype  
---  ------             -----  
 0   ID                 object 
 1   Humidity(%)        float64
 2   Pressure(in)       float64
 3   Visibility(mi)     float64
 4   Wind_Direction     object 
 5   Wind_Speed(mph)    float64
 6   Precipitation(in)  float64
 7   Weather_Condition  object 
 8   Amenity            bool   
 9   Bump               bool   
 10  Crossing           bool   
 11  Give_Way           bool   
dtypes: bool(4), float64(5), object(3)
memory usage: 184.5+ MB
```

In [17]:
```
partC.rename(columns={'Humidity(%)': 'Humidity', 'Pressure(in)': 'Pressur
partC
```

Out[17]:

	ID	Humidity	Pressure	Visibility	Wind_Direction	Wind_Speed	Precipitation	Weath
0	A-1	58.0	29.76	10.0	SW	10.4	0.00	
1	A-2	91.0	29.68	10.0	Calm	NaN	0.02	
2	A-3	97.0	29.70	10.0	Calm	NaN	0.02	
3	A-4	55.0	29.65	10.0	Calm	NaN	NaN	
4	A-5	93.0	29.69	10.0	WSW	10.4	0.01	
...	...	...	...	...	...	...	...	
2845337	A-2845338	40.0	28.92	10.0	W	13.0	0.00	
2845338	A-2845339	73.0	29.39	10.0	SW	6.0	0.00	
2845339	A-2845340	64.0	29.74	10.0	SSW	10.0	0.00	
2845340	A-2845341	81.0	29.62	10.0	SW	8.0	0.00	
2845341	A-2845342	47.0	28.63	7.0	SW	7.0	0.00	

2845342 rows × 12 columns

In [18]:
```
partC.describe()
```

Out[18]:

	Humidity	Pressure	Visibility	Wind_Speed	Precipitation
count	2.772250e+06	2.786142e+06	2.774796e+06	2.687398e+06	2.295884e+06
mean	6.436545e+01	2.947234e+01	9.099391e+00	7.395044e+00	7.016940e-03
std	2.287457e+01	1.045286e+00	2.717546e+00	5.527454e+00	9.348831e-02
min	1.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
25%	4.800000e+01	2.931000e+01	1.000000e+01	3.500000e+00	0.000000e+00
50%	6.700000e+01	2.982000e+01	1.000000e+01	7.000000e+00	0.000000e+00
75%	8.300000e+01	3.001000e+01	1.000000e+01	1.000000e+01	0.000000e+00
max	1.000000e+02	5.890000e+01	1.400000e+02	1.087000e+03	2.400000e+01

In [19]:
```
partC.isna().sum()
```

Out[19]:
```
ID                      0
Humidity             73092
Pressure             59200
Visibility           70546
Wind_Direction       73775
Wind_Speed          157944
Precipitation       549458
Weather_Condition    70636
Amenity                 0
Bump                    0
Crossing                0
Give_Way                0
dtype: int64
```

There seem to be many null values in the columns. Let us replace numerical column NA values with the mean and for categorical, with mode.

In [20]:
```
partC.head()
```

Out[20]:

	ID	Humidity	Pressure	Visibility	Wind_Direction	Wind_Speed	Precipitation	Weather_Condition
0	A-1	58.0	29.76	10.0	SW	10.4	0.00	Light Rain
1	A-2	91.0	29.68	10.0	Calm	NaN	0.02	Light Rain
2	A-3	97.0	29.70	10.0	Calm	NaN	0.02	Overcast
3	A-4	55.0	29.65	10.0	Calm	NaN	NaN	Overcast
4	A-5	93.0	29.69	10.0	WSW	10.4	0.01	Light Rain

In [21]:
```
# Imputing numerical columns NA with mean.

partC["Humidity"].fillna(value=partC["Humidity"].mean(), inplace=True)
partC["Pressure"].fillna(value=partC["Pressure"].mean(), inplace=True)
partC["Visibility"].fillna(value=partC["Visibility"].mean(), inplace=True
partC["Wind_Speed"].fillna(value=partC["Wind_Speed"].mean(), inplace=True
partC["Precipitation"].fillna(value=partC["Precipitation"].mean(), inplac
partC.isna().sum()
```

Out[21]:
```
ID                     0
Humidity               0
Pressure               0
Visibility             0
Wind_Direction       73775
Wind_Speed             0
Precipitation          0
Weather_Condition    70636
Amenity                0
Bump                   0
Crossing               0
Give_Way               0
dtype: int64
```

In [22]:
```
# Imputing categorical/object NA data with mode of the column.

partC["Wind_Direction"].fillna(value=partC["Wind_Direction"].mode()[0], i
partC["Weather_Condition"].fillna(value=partC["Weather_Condition"].mode()
partC["Amenity"].fillna(value=partC["Amenity"].mode()[0], inplace=True)
partC["Bump"].fillna(value=partC["Bump"].mode()[0], inplace=True)
partC["Crossing"].fillna(value=partC["Crossing"].mode()[0], inplace=True)
partC["Give_Way"].fillna(value=partC["Give_Way"].mode()[0], inplace=True)

partC.isna().sum()
```

Out[22]:
```
ID                   0
Humidity             0
Pressure             0
Visibility           0
Wind_Direction       0
Wind_Speed           0
Precipitation        0
Weather_Condition    0
Amenity              0
Bump                 0
Crossing             0
Give_Way             0
dtype: int64
```

We will clean up 'Wind Direction' and 'Weather Condition'.

In [23]:
```
partC['Wind_Direction'].unique()
```

Out[23]:
```
array(['SW', 'Calm', 'WSW', 'WNW', 'West', 'NNW', 'South', 'W', 'NW',
       'North', 'SSE', 'SSW', 'ESE', 'SE', 'CALM', 'East', 'Variable',
       'NNE', 'NE', 'ENE', 'S', 'VAR', 'N', 'E'], dtype=object)
```

We can observe there are many values for directions with same meaning, such as E, ESE, ENE as EAST. So we decided to replace all similar valued directions to one single value. This will be easier for modeling later.

In [24]:
```
partC.loc[partC['Wind_Direction']=='Calm','Wind_Direction'] = 'CALM'
partC.loc[partC['Wind_Direction']=='Variable','Wind_Direction'] = 'VAR'
partC.loc[(partC['Wind_Direction']=='East')|(partC['Wind_Direction']=='ES
partC.loc[(partC['Wind_Direction']=='West')|(partC['Wind_Direction']=='WS
partC.loc[(partC['Wind_Direction']=='South')|(partC['Wind_Direction']=='S
partC.loc[(partC['Wind_Direction']=='North')|(partC['Wind_Direction']=='N

partC['Wind_Direction'].unique()
```

Out[24]:
```
array(['SW', 'CALM', 'W', 'N', 'S', 'NW', 'E', 'SE', 'VAR', 'NE'],
      dtype=object)
```

In [25]:
```
partC['Weather_Condition'].unique()
```

Out[25]:
```
array(['Light Rain', 'Overcast', 'Mostly Cloudy', 'Snow', 'Light Snow',
       'Cloudy', 'Fair', 'Scattered Clouds', 'Clear', 'Partly Cloudy',
       'Light Freezing Drizzle', 'Light Drizzle', 'Haze', 'Rain',
       'Heavy Rain', 'Drizzle', 'Fog', 'Thunderstorms and Rain',
       'Patches of Fog', 'Light Thunderstorms and Rain', 'Mist',
       'Rain Showers', 'Light Rain Showers', 'Heavy Drizzle', 'Smoke',
       'Light Freezing Fog', 'Light Freezing Rain', 'Blowing Snow',
       'Heavy Thunderstorms and Rain', 'Heavy Snow', 'Snow Grains',
       'Squalls', 'Light Fog', 'Shallow Fog', 'Thunderstorm',
       'Light Ice Pellets', 'Thunder', 'Thunder in the Vicinity',
       'Fair / Windy', 'Light Rain with Thunder',
       'Heavy Thunderstorms and Snow', 'Light Snow Showers',
       'Cloudy / Windy', 'Ice Pellets', 'N/A Precipitation',
       'Light Thunderstorms and Snow', 'T-Storm', 'Rain / Windy',
       'Wintry Mix', 'Partly Cloudy / Windy', 'Heavy T-Storm', 'Sand',
       'Light Rain / Windy', 'Widespread Dust', 'Mostly Cloudy / Wind
y',
       'Blowing Dust / Windy', 'Blowing Dust', 'Volcanic Ash',
       'Freezing Rain / Windy', 'Small Hail', 'Wintry Mix / Windy',
       'Light Snow / Windy', 'Heavy Ice Pellets', 'Heavy Snow / Windy',
       'Heavy Rain / Windy', 'Heavy T-Storm / Windy', 'Fog / Windy',
       'Dust Whirls', 'Showers in the Vicinity', 'Funnel Cloud',
       'Thunder / Windy', 'Snow / Windy', 'Haze / Windy',
       'Light Snow and Sleet', 'T-Storm / Windy',
       'Sand / Dust Whirlwinds', 'Light Snow with Thunder', 'Rain Showe
r',
       'Blowing Snow / Windy', 'Light Rain Shower', 'Snow and Sleet',
       'Drizzle and Fog', 'Light Sleet', 'Drizzle / Windy',
       'Light Snow Shower', 'Snow and Thunder / Windy',
       'Light Sleet / Windy', 'Smoke / Windy', 'Widespread Dust / Wind
y',
       'Light Drizzle / Windy', 'Tornado', 'Squalls / Windy', 'Hail',
       'Blowing Snow Nearby', 'Partial Fog', 'Sand / Windy',
       'Thunder / Wintry Mix', 'Light Freezing Rain / Windy', 'Duststor
m',
       'Light Snow and Sleet / Windy', 'Heavy Rain Shower / Windy',
       'Sand / Dust Whirlwinds / Windy', 'Light Rain Shower / Windy',
       'Thunder and Hail', 'Freezing Rain', 'Heavy Sleet', 'Sleet',
       'Freezing Drizzle', 'Snow and Sleet / Windy',
       'Heavy Freezing Drizzle', 'Heavy Freezing Rain', 'Blowing Sand',
       'Thunder / Wintry Mix / Windy', 'Mist / Windy', 'Sleet / Windy',
       'Patches of Fog / Windy', 'Sand / Dust Whirls Nearby',
       'Heavy Rain Shower', 'Drifting Snow', 'Heavy Blowing Snow',
       'Low Drifting Snow', 'Light Blowing Snow', 'Heavy Rain Showers',
       'Light Haze', 'Heavy Thunderstorms with Small Hail',
       'Heavy Snow with Thunder', 'Thunder and Hail / Windy'],
      dtype=object)
```

We can observe that there are too many distinct values for weather condition (similar to wind direction). So we will replace similar valued weather conditions to one single value.

In [26]:
```
# Define a dictionary to map weather conditions to categories
weather_map = {
    'Clear': ['Fair', 'Clear', 'Fair / Windy'],
    'Cloud': ['Mostly Cloudy', 'Partly Cloudy / Windy', 'Cloudy', 'Scatte
    'Dusty/Windy' : ['Sand / Dust Whirls Nearby', 'Blowing Sand', 'Sand /
    'Rain': ['Light Rain', 'Thunder and Hail / Windy', 'Heavy Thunderstor
    'Heavy_Rain': ['Heavy Rain', 'Heavy Rain Showers', 'Heavy Rain Shower
    'Snow': ['Snow', 'Thunder / Wintry Mix / Windy', 'Drifting Snow', 'Lo
    'Heavy_Snow': ['Heavy Snow', 'Heavy Blowing Snow', 'Heavy Snow with T
    'Fog': ['Haze', 'Light Haze', 'Patches of Fog / Windy', 'Mist', 'Smok
}

# Iterate through the DataFrame and replace the weather conditions with c
for key in weather_map:
    partC['Weather_Condition'].replace(weather_map[key], key, inplace=Tru
```

In [27]:
```
partC.head()
```

Out[27]:

	ID	Humidity	Pressure	Visibility	Wind_Direction	Wind_Speed	Precipitation	Weather_Condition
0	A-1	58.0	29.76	10.0	SW	10.400000	0.000000	Rain
1	A-2	91.0	29.68	10.0	CALM	7.395044	0.020000	Rain
2	A-3	97.0	29.70	10.0	CALM	7.395044	0.020000	Cloud
3	A-4	55.0	29.65	10.0	CALM	7.395044	0.007017	Cloud
4	A-5	93.0	29.69	10.0	W	10.400000	0.010000	Rain

In [28]:
```
partC.describe()
```

Out[28]:

	Humidity	Pressure	Visibility	Wind_Speed	Precipitation
count	2.845342e+06	2.845342e+06	2.845342e+06	2.845342e+06	2.845342e+06
mean	6.436545e+01	2.947234e+01	9.099391e+00	7.395044e+00	7.016940e-03
std	2.257885e+01	1.034355e+00	2.683646e+00	5.371850e+00	8.397790e-02
min	1.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
25%	4.900000e+01	2.933000e+01	1.000000e+01	4.600000e+00	0.000000e+00
50%	6.600000e+01	2.981000e+01	1.000000e+01	7.000000e+00	0.000000e+00
75%	8.300000e+01	3.001000e+01	1.000000e+01	1.000000e+01	7.016940e-03
max	1.000000e+02	5.890000e+01	1.400000e+02	1.087000e+03	2.400000e+01

PartD contains the columns from index 34 to 47.

In [29]:
```
# Part D columns 34-47
first_col = accidents_dataset.iloc[:, 0]  # First column
middle_cols = accidents_dataset.iloc[:,34 : 47]  # Middle columns (column

partD = pd.concat([first_col, middle_cols], axis=1)
partD.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2845342 entries, 0 to 2845341
Data columns (total 14 columns):
 #   Column                 Dtype 
---  ------                 ----- 
 0   ID                     object
 1   Junction               bool  
 2   No_Exit                bool  
 3   Railway                bool  
 4   Roundabout             bool  
 5   Station                bool  
 6   Stop                   bool  
 7   Traffic_Calming        bool  
 8   Traffic_Signal         bool  
 9   Turning_Loop           bool  
 10  Sunrise_Sunset         object
 11  Civil_Twilight         object
 12  Nautical_Twilight      object
 13  Astronomical_Twilight  object
dtypes: bool(9), object(5)
memory usage: 133.0+ MB
```

In [30]:
```
partD
```

Out[30]:

	ID	Junction	No_Exit	Railway	Roundabout	Station	Stop	Traffic_Calming	Traffic
0	A-1	False	False	False	False	False	False	False	
1	A-2	False	False	False	False	False	False	False	
2	A-3	True	False	False	False	False	False	False	
3	A-4	False	False	False	False	False	False	False	
4	A-5	False	False	False	False	False	False	False	
...	...	...	...	...	...	...	...	...	
2845337	A-2845338	False	False	False	False	False	False	False	
2845338	A-2845339	False	False	False	False	False	False	False	
2845339	A-2845340	True	False	False	False	False	False	False	
2845340	A-2845341	False	False	False	False	False	False	False	
2845341	A-2845342	False	False	False	False	False	False	False	

2845342 rows × 14 columns

In [31]:
```
partD.describe()
```

Out[31]:

	ID	Junction	No_Exit	Railway	Roundabout	Station	Stop	Traffic_Calming	Tra
count	2845342	2845342	2845342	2845342	2845342	2845342	2845342	2845342	
unique	2845342	2	2	2	2	2	2	2	
top	A-1	False	False	False	False	False	False	False	
freq	1	2554837	2841048	2822711	2845219	2777347	2794942	2843630	

In [32]:
```
partD.isna().sum()
```

Out[32]:
```
ID                          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           2867
Civil_Twilight           2867
Nautical_Twilight        2867
Astronomical_Twilight    2867
dtype: int64
```

In [33]:
```
# Imputing categorical/object NA data with mode of the column.

for column in partD.columns:
    partD[column].fillna(partD[column].mode()[0], inplace=True)
partD.isna().sum()
```

Out[33]:
```
ID                       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
```

We replaced the NA values with Mean and Mode for respective data types. Also, we are encoding the Point of Interest and Period of day attributes with numerical data.

In [34]:
```
# Convert binary categorical columns to integers
binary_cols = ['Junction', 'No_Exit', 'Railway', 'Roundabout', 'Station',
               'Turning_Loop']
partD[binary_cols] = partD[binary_cols].replace({'TRUE': 1, 'FALSE': 0})

# Convert sunrise/sunset columns to integers
sunrise_sunset_map = {'Day': 1, 'Night': 0}
partD[['Sunrise_Sunset', 'Civil_Twilight', 'Nautical_Twilight', 'Astronom
    partD[['Sunrise_Sunset', 'Civil_Twilight', 'Nautical_Twilight', 'Astr
```

In [35]:
```
partD
```

Out[35]:

	ID	Junction	No_Exit	Railway	Roundabout	Station	Stop	Traffic_Calming	Traffic
0	A-1	False	False	False	False	False	False	False	
1	A-2	False	False	False	False	False	False	False	
2	A-3	True	False	False	False	False	False	False	
3	A-4	False	False	False	False	False	False	False	
4	A-5	False	False	False	False	False	False	False	
...	...	...	...	...	...	...	...	...	
2845337	A-2845338	False	False	False	False	False	False	False	
2845338	A-2845339	False	False	False	False	False	False	False	
2845339	A-2845340	True	False	False	False	False	False	False	
2845340	A-2845341	False	False	False	False	False	False	False	
2845341	A-2845342	False	False	False	False	False	False	False	

2845342 rows × 14 columns

In [36]:
```
data1 = pd.merge(partA, partB, on='ID')
data2 = pd.merge(partC, partD, on='ID')

clean_data = pd.merge(data1, data2, on='ID')
```

In [37]:
```
clean_data
```

Out[37]:

	ID	Severity	Start_Time	End_Time	Start_Lat	Start_Lng	End_Lat	End_Lng
0	A-1	3	2016-02-08 00:37:08	2016-02-08 06:37:08	40.108910	-83.092860	40.112060	-83.03187C
1	A-2	2	2016-02-08 05:56:20	2016-02-08 11:56:20	39.865420	-84.062800	39.865010	-84.04873C
2	A-3	2	2016-02-08 06:15:39	2016-02-08 12:15:39	39.102660	-84.524680	39.102090	-84.523960
3	A-4	2	2016-02-08 06:51:45	2016-02-08 12:51:45	41.062130	-81.537840	41.062170	-81.53547C
4	A-5	3	2016-02-08 07:53:43	2016-02-08 13:53:43	39.172393	-84.492792	39.170476	-84.501798
...	...	...	...	...	...	...	...	...
2845337	A-2845338	2	2019-08-23 18:03:25	2019-08-23 18:32:01	34.002480	-117.379360	33.998880	-117.37094C
2845338	A-2845339	2	2019-08-23 19:11:30	2019-08-23 19:38:23	32.766960	-117.148060	32.765550	-117.15363C
2845339	A-2845340	2	2019-08-23 19:00:21	2019-08-23 19:28:49	33.775450	-117.847790	33.777400	-117.85727C
2845340	A-2845341	2	2019-08-23 19:00:21	2019-08-23 19:29:42	33.992460	-118.403020	33.983110	-118.39565C
2845341	A-2845342	2	2019-08-23 18:52:06	2019-08-23 19:21:31	34.133930	-117.230920	34.137360	-117.23934C

2845342 rows × 46 columns

Once done, we have exported the clean data as a csv. This data will be imported using R in RStudio for further visualizations and analysis.

In [39]:
```
clean_data.isna().sum()
```

Out[39]:
```
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
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                0
Visibility              0
Wind_Direction          0
Wind_Speed              0
Precipitation           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 [40]:
```
clean_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2845342 entries, 0 to 2845341
Data columns (total 46 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  Street                 object 
 11  Side                   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               float64
 24  Visibility             float64
 25  Wind_Direction         object 
 26  Wind_Speed             float64
 27  Precipitation          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         int64  
 43  Civil_Twilight         int64  
 44  Nautical_Twilight      int64  
 45  Astronomical_Twilight  int64  
dtypes: bool(13), float64(12), int64(5), object(16)
memory usage: 773.4+ MB
```

In [41]:
```
clean_data['Wind_Direction'].unique()
```

Out[41]:
```
array(['SW', 'CALM', 'W', 'N', 'S', 'NW', 'E', 'SE', 'VAR', 'NE'],
      dtype=object)
```

In [42]:
```
clean_data['Weather_Condition'].unique()
```

Out[42]:
```
array(['Rain', 'Cloud', 'Snow', 'Clear', 'Fog', 'Heavy_Rain',
       'Heavy_Snow', 'Dusty/Windy'], dtype=object)
```

In [43]:
```
clean_data.to_csv('clean_data.csv', index=False)
```

Please refer to Project.Rmd for further process. Thank you