US Accidents Exploratory Data Analysis

```
pip install --upgrade matplotlib
```

```
Collecting matplotlib
  Using cached matplotlib-3.9.3-cp312-cp312-win_amd64.whl.metadata (11
kB)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\user\
appdata\local\programs\python\python312\lib\site-packages (from
matplotlib) (1.3.0)
Requirement already satisfied: cycler>=0.10 in c:\users\user\appdata\
local\programs\python\python312\lib\site-packages (from matplotlib)
(0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\user\
appdata\local\programs\python\python312\lib\site-packages (from
matplotlib) (4.53.1)
Requirement already satisfied: kiwisolver>=1.3.1 in c:\users\user\
appdata\local\programs\python\python312\lib\site-packages (from
matplotlib) (1.4.7)
Requirement already satisfied: numpy>=1.23 in c:\users\user\appdata\
local\programs\python\python312\lib\site-packages (from matplotlib)
(2.1.1)
Requirement already satisfied: packaging>=20.0 in c:\users\user\
appdata\local\programs\python\python312\lib\site-packages (from
matplotlib) (24.1)
Requirement already satisfied: pillow>=8 in c:\users\user\appdata\
local\programs\python\python312\lib\site-packages (from matplotlib)
(10.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in c:\users\user\
appdata\local\programs\python\python312\lib\site-packages (from
matplotlib) (3.1.4)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\user\
appdata\local\programs\python\python312\lib\site-packages (from
matplotlib) (2.9.0.post0)
Requirement already satisfied: six>=1.5 in c:\users\user\appdata\
local\programs\python\python312\lib\site-packages (from python-
dateutil>=2.7->matplotlib) (1.16.0)
Using cached matplotlib-3.9.3-cp312-cp312-win_amd64.whl (7.8 MB)
Installing collected packages: matplotlib
Successfully installed matplotlib-3.9.3
Note: you may need to restart the kernel to use updated packages.

WARNING: Ignoring invalid distribution ~atplotlib (C:\Users\user\
AppData\Local\Programs\Python\Python312\Lib\site-packages)
WARNING: Ignoring invalid distribution ~atplotlib (C:\Users\user\
AppData\Local\Programs\Python\Python312\Lib\site-packages)
WARNING: Ignoring invalid distribution ~atplotlib (C:\Users\user\
AppData\Local\Programs\Python\Python312\Lib\site-packages)
```

```
pip install opendatasets --upgrade
```

```
Requirement already satisfied: opendatasets in c:\users\user\appdata\local\programs\python\python312\lib\site-packages (0.1.22)
Requirement already satisfied: tqdm in c:\users\user\appdata\local\programs\python\python312\lib\site-packages (from opendatasets) (4.67.1)
Requirement already satisfied: kaggle in c:\users\user\appdata\local\programs\python\python312\lib\site-packages (from opendatasets) (1.6.17)
Requirement already satisfied: click in c:\users\user\appdata\local\programs\python\python312\lib\site-packages (from opendatasets) (8.1.7)
Requirement already satisfied: colorama in c:\users\user\appdata\local\programs\python\python312\lib\site-packages (from click->opendatasets) (0.4.6)
Requirement already satisfied: six>=1.10 in c:\users\user\appdata\local\programs\python\python312\lib\site-packages (from kaggle->opendatasets) (1.16.0)
Requirement already satisfied: certifi>=2023.7.22 in c:\users\user\appdata\local\programs\python\python312\lib\site-packages (from kaggle->opendatasets) (2024.8.30)
Requirement already satisfied: python-dateutil in c:\users\user\appdata\local\programs\python\python312\lib\site-packages (from kaggle->opendatasets) (2.9.0.post0)
Requirement already satisfied: requests in c:\users\user\appdata\local\programs\python\python312\lib\site-packages (from kaggle->opendatasets) (2.32.3)
Requirement already satisfied: python-slugify in c:\users\user\appdata\local\programs\python\python312\lib\site-packages (from kaggle->opendatasets) (8.0.4)
Requirement already satisfied: urllib3 in c:\users\user\appdata\local\programs\python\python312\lib\site-packages (from kaggle->opendatasets) (2.2.2)
Requirement already satisfied: bleach in c:\users\user\appdata\local\programs\python\python312\lib\site-packages (from kaggle->opendatasets) (6.1.0)
Requirement already satisfied: webencodings in c:\users\user\appdata\local\programs\python\python312\lib\site-packages (from bleach->kaggle->opendatasets) (0.5.1)
Requirement already satisfied: text-unidecode>=1.3 in c:\users\user\appdata\local\programs\python\python312\lib\site-packages (from python-slugify->kaggle->opendatasets) (1.3)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\users\user\appdata\local\programs\python\python312\lib\site-packages (from requests->kaggle->opendatasets) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in c:\users\user\appdata\local\programs\python\python312\lib\site-packages (from requests->kaggle->opendatasets) (3.8)
Note: you may need to restart the kernel to use updated packages.
```

```
WARNING: Ignoring invalid distribution ~atplotlib (C:\Users\user\
AppData\Local\Programs\Python\Python312\Lib\site-packages)
WARNING: Retrying (Retry(total=4, connect=None, read=None,
redirect=None, status=None)) after connection broken by
'NewConnectionError('<pip._vendor.urllib3.connection.HTTPSConnection
object at 0x0000022CE4A677D0>: Failed to establish a new connection:
[Errno 11001] getaddrinfo failed')': /simple/opendatasets/
WARNING: Retrying (Retry(total=3, connect=None, read=None,
redirect=None, status=None)) after connection broken by
'NewConnectionError('<pip._vendor.urllib3.connection.HTTPSConnection
object at 0x0000022CE4A67A40>: Failed to establish a new connection:
[Errno 11001] getaddrinfo failed')': /simple/opendatasets/
WARNING: Retrying (Retry(total=2, connect=None, read=None,
redirect=None, status=None)) after connection broken by
'NewConnectionError('<pip._vendor.urllib3.connection.HTTPSConnection
object at 0x0000022CE4A67C80>: Failed to establish a new connection:
[Errno 11001] getaddrinfo failed')': /simple/opendatasets/
WARNING: Retrying (Retry(total=1, connect=None, read=None,
redirect=None, status=None)) after connection broken by
'NewConnectionError('<pip._vendor.urllib3.connection.HTTPSConnection
object at 0x0000022CE4A67EF0>: Failed to establish a new connection:
[Errno 11001] getaddrinfo failed')': /simple/opendatasets/
WARNING: Retrying (Retry(total=0, connect=None, read=None,
redirect=None, status=None)) after connection broken by
'NewConnectionError('<pip._vendor.urllib3.connection.HTTPSConnection
object at 0x0000022CE4A8C110>: Failed to establish a new connection:
[Errno 11001] getaddrinfo failed')': /simple/opendatasets/
WARNING: Ignoring invalid distribution ~atplotlib (C:\Users\user\
AppData\Local\Programs\Python\Python312\Lib\site-packages)
WARNING: Ignoring invalid distribution ~atplotlib (C:\Users\user\
AppData\Local\Programs\Python\Python312\Lib\site-packages)
```

```python
import opendatasets as od

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

od.download(download_url)
```

```
Please provide your Kaggle credentials to download this dataset. Learn
more: http://bit.ly/kaggle-creds
Your Kaggle username:

  rohitdas2002

Your Kaggle Key:

  ········
```

```
Dataset URL: https://www.kaggle.com/datasets/sobhanmoosavi/us-accidents
Downloading us-accidents.zip to .\us-accidents

100%|██████████| 653M/653M [02:22<00:00, 4.80MB/s]


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 and the columns
3. Fix any missing or incorrect values

```
import pandas as pd

df = pd.read_csv(data_filename)

df_population = pd.read_csv("US_population_data.csv")

df.head(2)

    ID   Source  Severity           Start_Time             End_Time  \
0  A-1  Source2         3  2016-02-08 05:46:00  2016-02-08 11:00:00
1  A-2  Source2         2  2016-02-08 06:07:59  2016-02-08 06:37:59

   Start_Lat  Start_Lng  End_Lat  End_Lng  Distance(mi)  ...
Roundabout  \
0  39.865147 -84.058723      NaN      NaN          0.01  ...
False
1  39.928059 -82.831184      NaN      NaN          0.01  ...
False

  Station   Stop Traffic_Calming Traffic_Signal Turning_Loop
Sunrise_Sunset  \
0   False  False           False          False        False
Night
1   False  False           False          False        False
Night

  Civil_Twilight Nautical_Twilight Astronomical_Twilight
0          Night             Night                 Night
1          Night             Night                   Day

[2 rows x 46 columns]
```

```
df_population.head(5)
```

```
         Name State  Year Population
0     Alabama    AL  2016    4863300
1      Alaska    AK  2016     741894
2     Arizona    AZ  2016    6931071
3    Arkansas    AR  2016    2988248
4  California    CA  2016   39250017
```

```
df_population.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 416 entries, 0 to 415
Data columns (total 4 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   Name        416 non-null    object
 1   State       416 non-null    object
 2   Year        416 non-null    int64 
 3   Population  416 non-null    object
dtypes: int64(1), object(3)
memory usage: 13.1+ KB
```

```
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
```

```
df.describe()
```

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

```
       Distance(mi)  Temperature(F)  Wind_Chill(F)   Humidity(%)  \
count  7.728394e+06    7.564541e+06   5.729375e+06  7.554250e+06
mean   5.618423e-01    6.166329e+01   5.825105e+01  6.483104e+01
std    1.776811e+00    1.901365e+01   2.238983e+01  2.282097e+01
min    0.000000e+00   -8.900000e+01  -8.900000e+01  1.000000e+00
25%    0.000000e+00    4.900000e+01   4.300000e+01  4.800000e+01
50%    3.000000e-02    6.400000e+01   6.200000e+01  6.700000e+01
75%    4.640000e-01    7.600000e+01   7.500000e+01  8.400000e+01
max    4.417500e+02    2.070000e+02   2.070000e+02  1.000000e+02

       Pressure(in)  Visibility(mi)  Wind_Speed(mph)
Precipitation(in)
count  7.587715e+06    7.551296e+06     7.157161e+06
5.524808e+06
mean   2.953899e+01    9.090376e+00     7.685490e+00       8.407210e-
03
std    1.006190e+00    2.688316e+00     5.424983e+00       1.102246e-
01
min    0.000000e+00    0.000000e+00     0.000000e+00
0.000000e+00
25%    2.937000e+01    1.000000e+01     4.600000e+00
0.000000e+00
50%    2.986000e+01    1.000000e+01     7.000000e+00
0.000000e+00
75%    3.003000e+01    1.000000e+01     1.040000e+01
0.000000e+00
max    5.863000e+01    1.400000e+02     1.087000e+03
3.647000e+01
```

```python
numerics = ['int16', 'int32', 'int64', 'float16', 'float32',
'float64']
numeric_df = df.select_dtypes(include=numerics)
len(numeric_df.columns)
```

```
13
```

```python
missing_percentages = df.isna().sum().sort_values(ascending = False) /
len(df)
missing_percentages
```

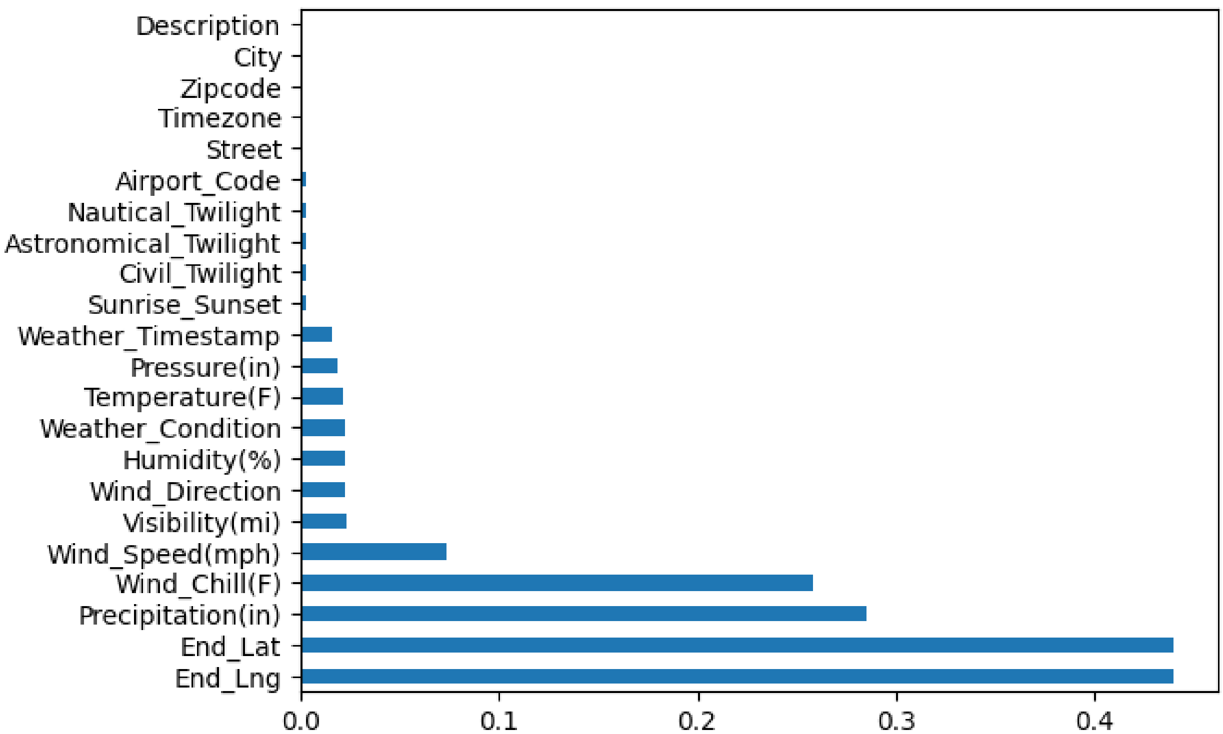
```
End_Lng                   4.402935e-01
End_Lat                   4.402935e-01
Precipitation(in)         2.851286e-01
Wind_Chill(F)             2.586590e-01
Wind_Speed(mph)           7.391355e-02
Visibility(mi)            2.291524e-02
Wind_Direction            2.267043e-02
Humidity(%)               2.253301e-02
Weather_Condition         2.244438e-02
Temperature(F)            2.120143e-02
```

```
Pressure(in)               1.820288e-02
Weather_Timestamp          1.555666e-02
Sunrise_Sunset             3.007869e-03
Civil_Twilight             3.007869e-03
Astronomical_Twilight      3.007869e-03
Nautical_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
ID                         0.000000e+00
Distance(mi)               0.000000e+00
Start_Lng                  0.000000e+00
Source                     0.000000e+00
Severity                   0.000000e+00
Start_Time                 0.000000e+00
End_Time                   0.000000e+00
Start_Lat                  0.000000e+00
County                     0.000000e+00
Amenity                    0.000000e+00
Country                    0.000000e+00
State                      0.000000e+00
Bump                       0.000000e+00
Crossing                   0.000000e+00
Give_Way                   0.000000e+00
Junction                   0.000000e+00
Station                    0.000000e+00
Roundabout                 0.000000e+00
Railway                    0.000000e+00
No_Exit                    0.000000e+00
Turning_Loop               0.000000e+00
Traffic_Signal             0.000000e+00
Traffic_Calming            0.000000e+00
Stop                       0.000000e+00
dtype: float64
```

```
missing_percentages[missing_percentages !=0]
```

```
missing_percentages[missing_percentages !=0].plot(kind='barh')
```

```
<Axes: >
```



Plotting the Missing values to check the distribution of missing values per column

Exploratory Data Analysis and Visualization

columns we'll analyze:

1. City
2. Start Time
3. Start Lat, Start Lng
4. Temperature
5. Weather Condition

City Column Analysis

```
cities = df.City.unique()
len(cities)
```

```
13679
```

```
cities_by_accidents = df.City.value_counts()
cities_by_accidents
```

```
City
Miami           186917
Houston         169609
```

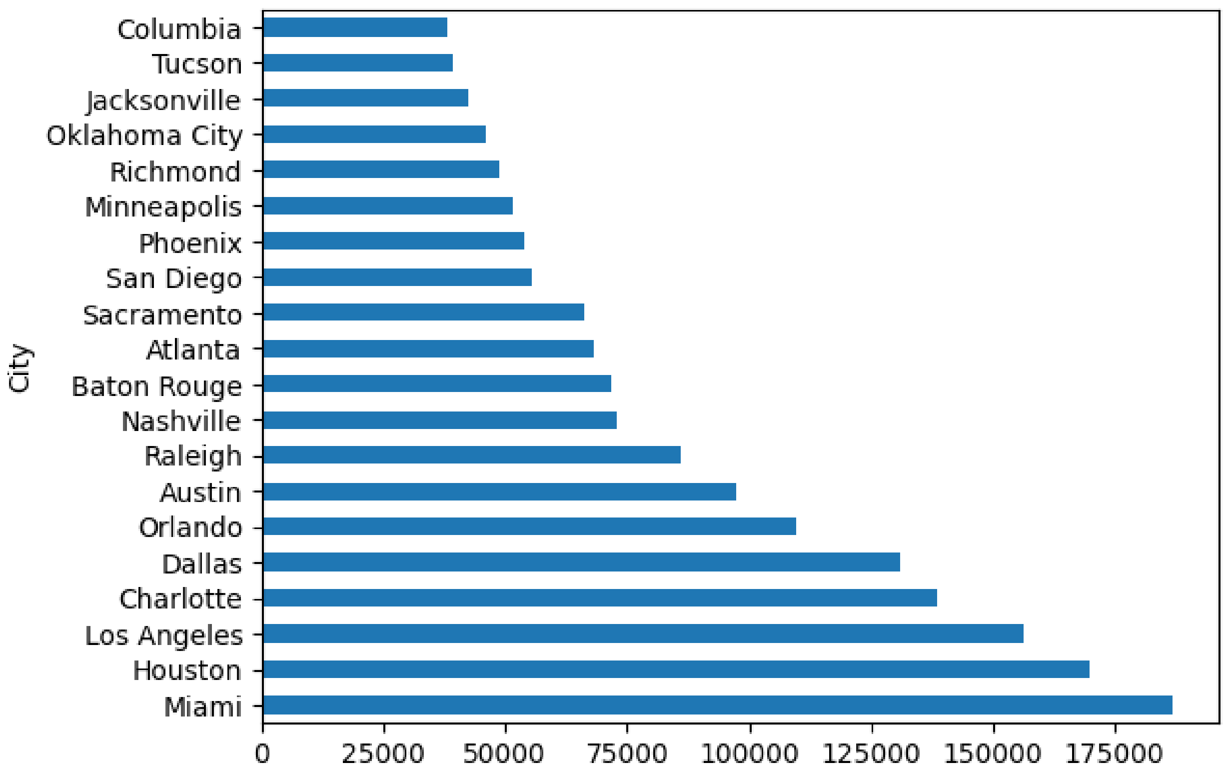
```
Los Angeles    156491
Charlotte      138652
Dallas         130939
                ...
Saint Croix         1
Masardis            1
Okaton              1
Wasta               1
Adell               1
Name: count, Length: 13678, dtype: int64
```

```
cities_by_accidents[:20]
```

```
City
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: count, dtype: int64
```

```
cities_by_accidents[:20].plot(kind = 'barh')
```

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



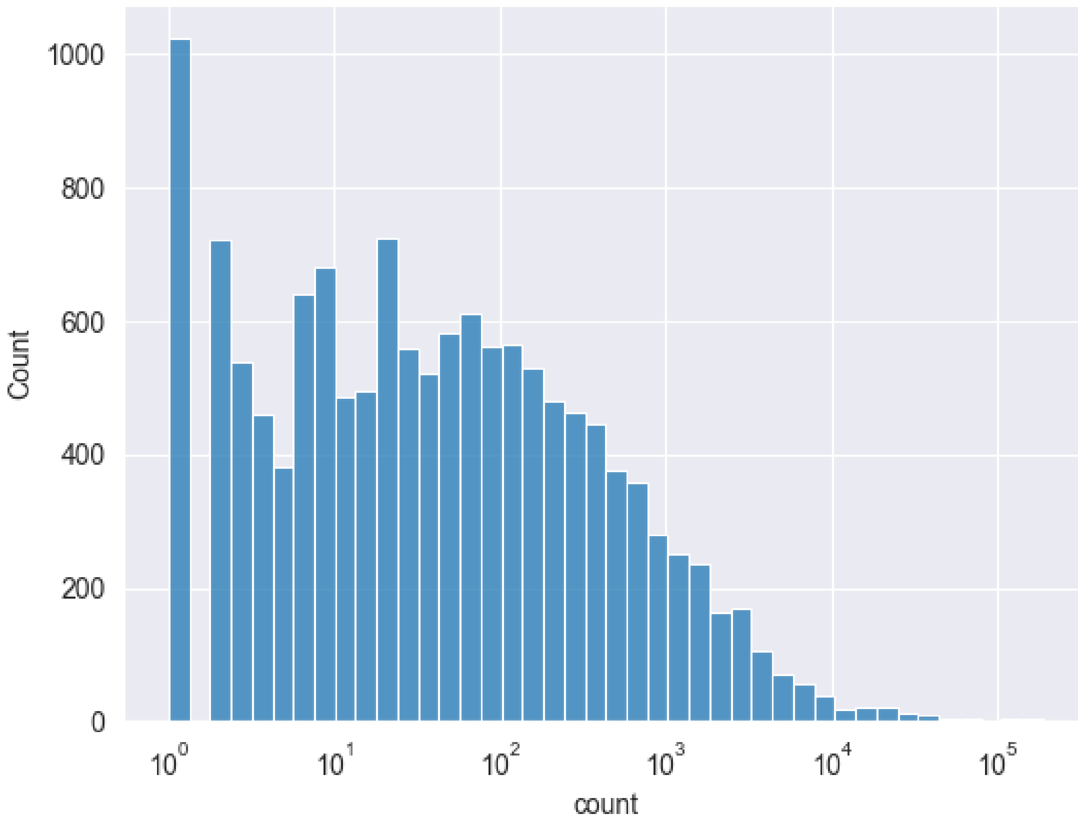
Visualizing Total number of Accidents by city in which we found that Miami, Houston, Los Angeles,

Charlotte and Dallas are the Top 5 Cities with most number of Accidents

```
import seaborn as sns
sns.set_style("darkgrid")
import matplotlib.pyplot as plt
```

Distribution for the no of Accidents

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

<Axes: xlabel='count', ylabel='Count'>
```



By this graph we can analyze that the major chunk of Accidents are between 0 to 100 Accidents and we can also see that

there are 1023 cities with only 1 Accident which seems suspicious because its the data for almost 7 years so it needs some investigation

or else we can skip those cities with less than 10 accidents because we can't really make any useful insights with such small no of rows

```
cities_by_accidents[cities_by_accidents == 1]

City
American Fork-Pleasant Grove    1
Berlin township                 1
District 1 Abingdon             1
Selby                           1
Smackover                       1
                               ..
Saint Croix                     1
Masardis                        1
Okaton                          1
Wasta                           1
```

```
Adell                                  1
Name: count, Length: 1023, dtype: int64
```

These are the cities with only 1 Accident

```
cities_by_accidents[cities_by_accidents == 10]
```

```
City
Orlinda                    10
Lecompton                  10
Hume                       10
Springer                   10
Setauket                   10
                           ..
Pierceton                  10
Benton City                10
Keansburg                  10
Eunice                     10
Luke Air Force Base        10
Name: count, Length: 209, dtype: int64
```

These are the cities with only 10 Accident

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

len(high_accident_cities) / len(cities)
```

```
0.0894578313253012
```

Start Time Column Analysis

```
df.Start_Time
```

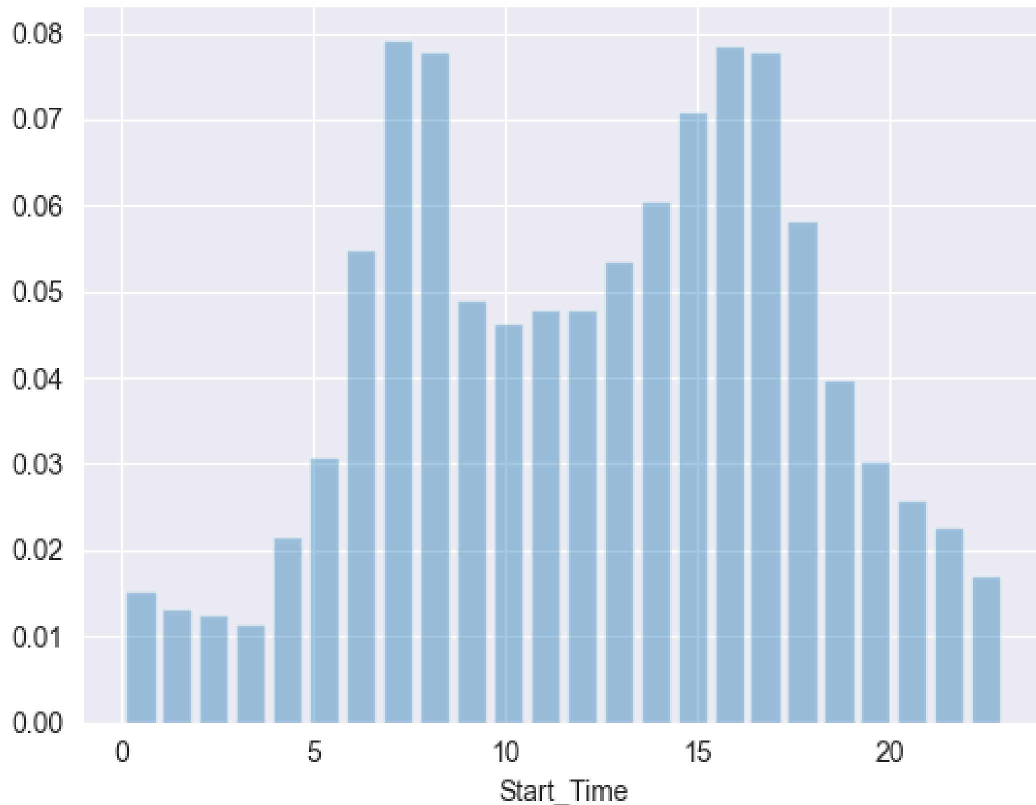
```
0          2016-02-08 05:46:00
1          2016-02-08 06:07:59
2          2016-02-08 06:49:27
3          2016-02-08 07:23:34
4          2016-02-08 07:39:07
                  ...
7728389    2019-08-23 18:03:25
7728390    2019-08-23 19:11:30
7728391    2019-08-23 19:00:21
7728392    2019-08-23 19:00:21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

```
df['Start_Time'] = pd.to_datetime(df['Start_Time'], format='mixed')
## converted it to datetime format
```

```
0          5
1          6
2          6
3          7
4          7
          ..
7728389   18
7728390   19
7728391   19
7728392   19
7728393   18
Name: Start_Time, Length: 7728394, dtype: int32
```

```
sns.distplot(df.Start_Time.dt.hour, bins=24, kde=False, norm_hist =
True, hist_kws={'rwidth': 0.8})
```

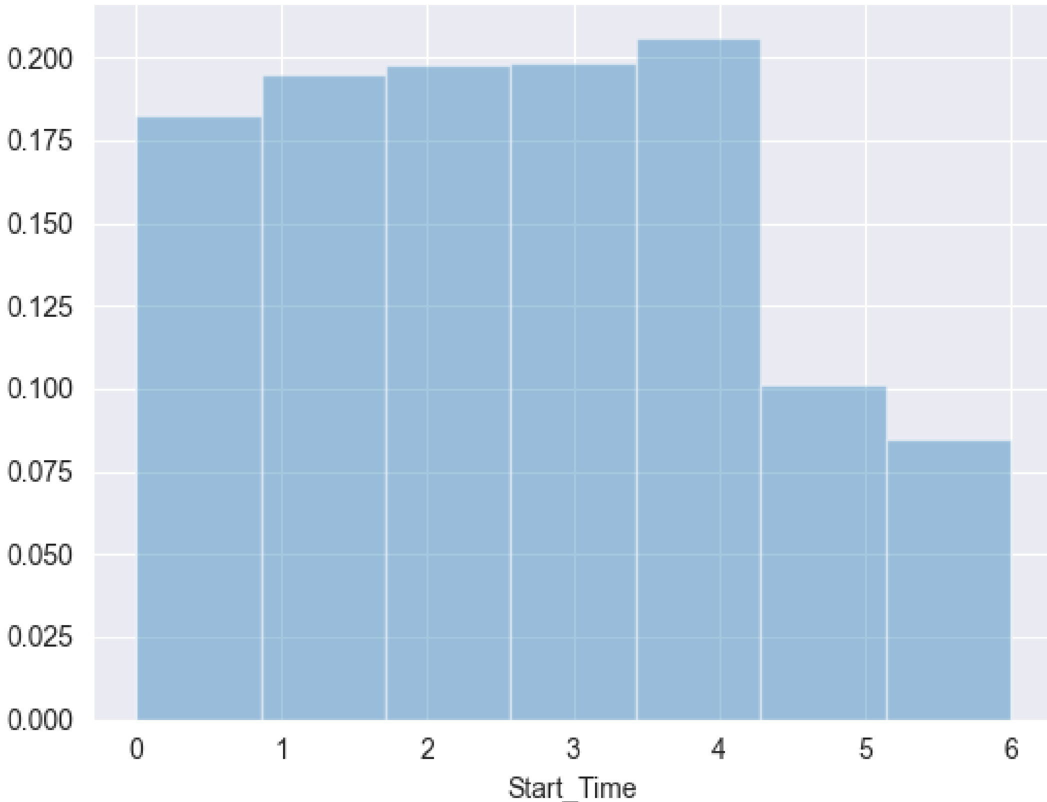
```
C:\Users\user\AppData\Local\Temp\ipykernel_13476\2767187012.py:1:
UserWarning:

`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, hist_kws={'rwidth': 0.8})

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



- A high percentage of accidents occur between 6 am to 10 am (probably people are in a hurry to get to work)
- Next highest percentage is 3 pm to 6pm (Probably people are returning back from work in this time period)

```
sns.distplot(df.Start_Time.dt.dayofweek, bins=7, kde=False, norm_hist
= True, hist_kws={'rwidth': 1.5})

C:\Users\user\AppData\Local\Temp\ipykernel_13476\498906560.py:1:
UserWarning:

`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, hist_kws={'rwidth': 1.5})
```

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



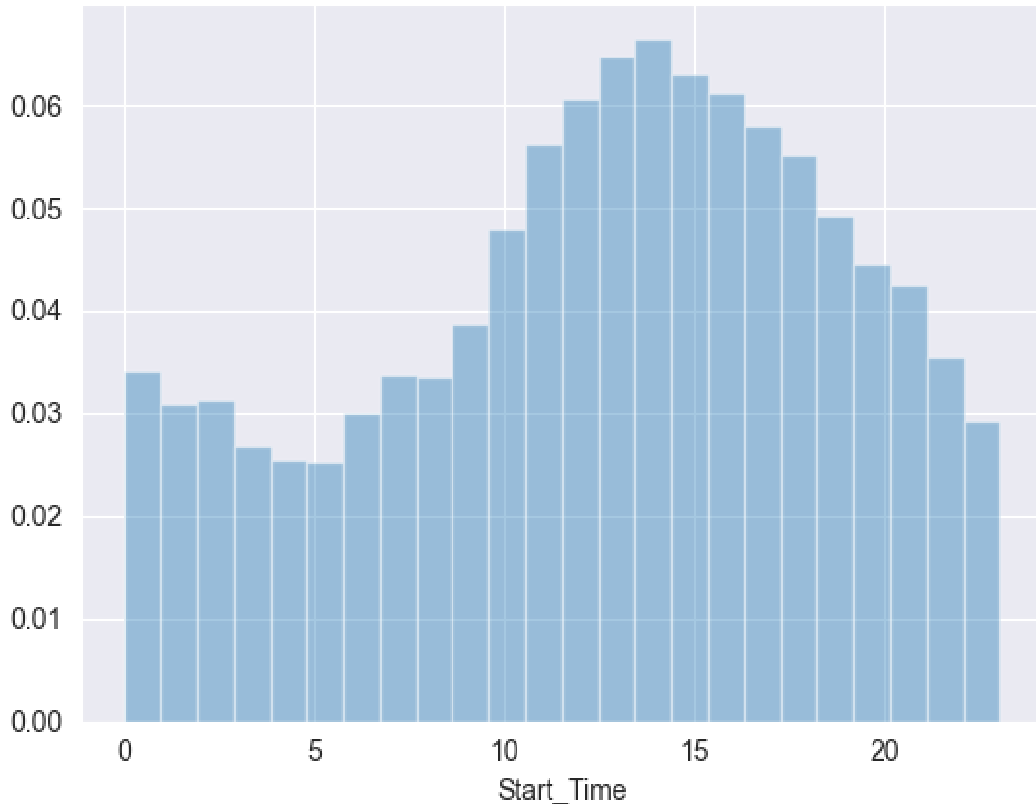
- we can notice that in the weekdays accidents occur more comparatively from Weekends
- Is the distribution of accidents by hours the same on weekends as on weekdays?

```
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, hist_kws={'rwidth': 1})
```

```
C:\Users\user\AppData\Local\Temp\ipykernel_13476\1186104827.py:2:
UserWarning:

`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 = True, hist_kws={'rwidth': 1})
```

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



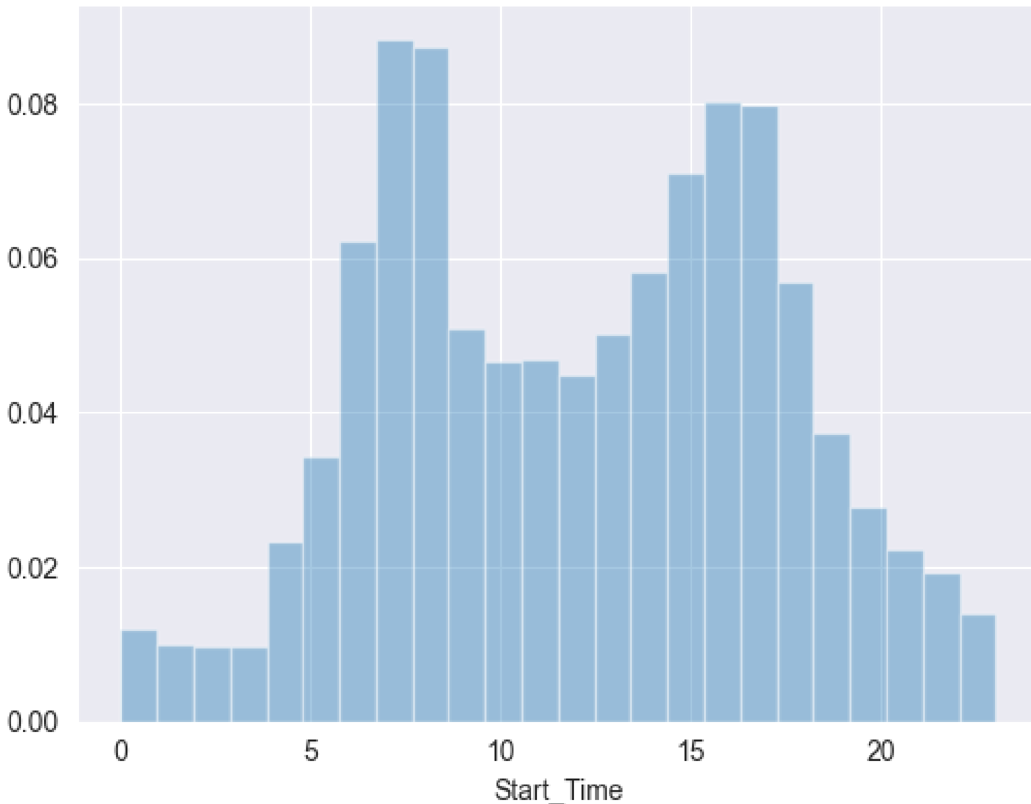
- On Sundays, the peak occurs between 10 am and 3 pm, unlike weekdays

```
monday_start_time = df.Start_Time[df.Start_Time.dt.dayofweek == 0]
sns.distplot(monday_start_time.dt.hour, bins=24, kde=False, norm_hist
= True, hist_kws={'rwidth': 1})
```

```
C:\Users\user\AppData\Local\Temp\ipykernel_13476\3804390815.py:2:
UserWarning:

`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(monday_start_time.dt.hour, bins=24, kde=False,
norm_hist = True, hist_kws={'rwidth': 1})
```

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



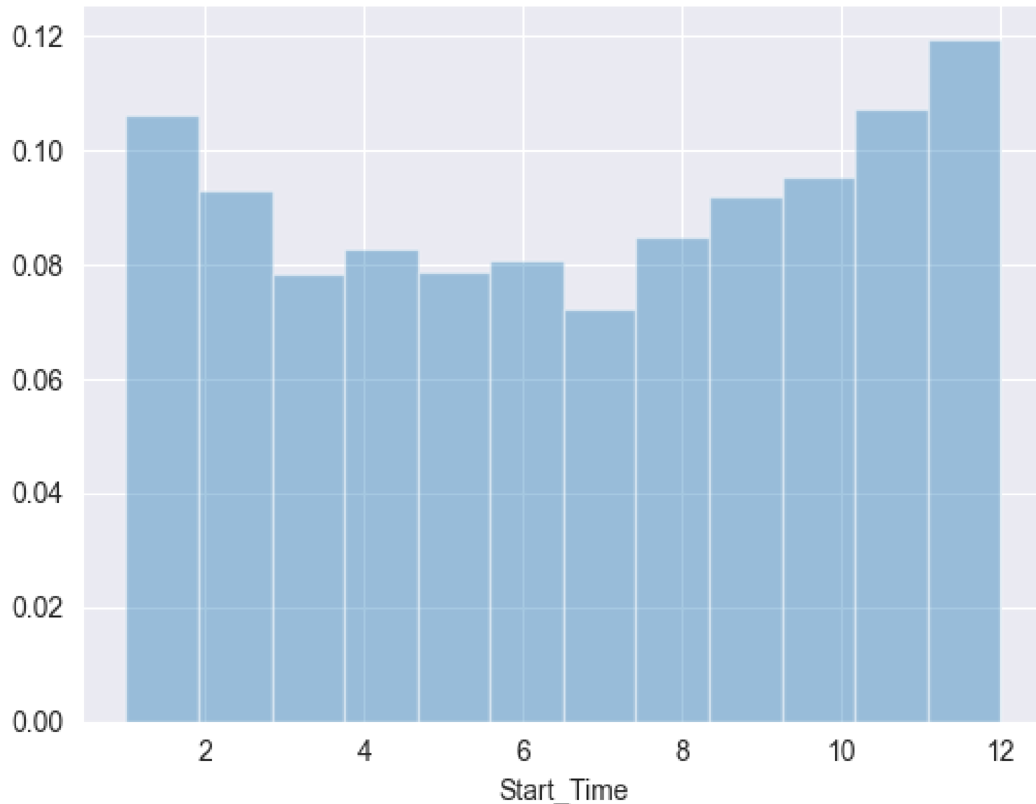
- On Mondays, the peak occurs between 5 am and 10 pm, unlike weekdays

```
sns.distplot(df.Start_Time.dt.month, bins=12, kde=False, norm_hist =
True, hist_kws={'rwidth': 1})
```

```
C:\Users\user\AppData\Local\Temp\ipykernel_13476\3382391673.py:1:
UserWarning:

`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, hist_kws={'rwidth': 1})
```

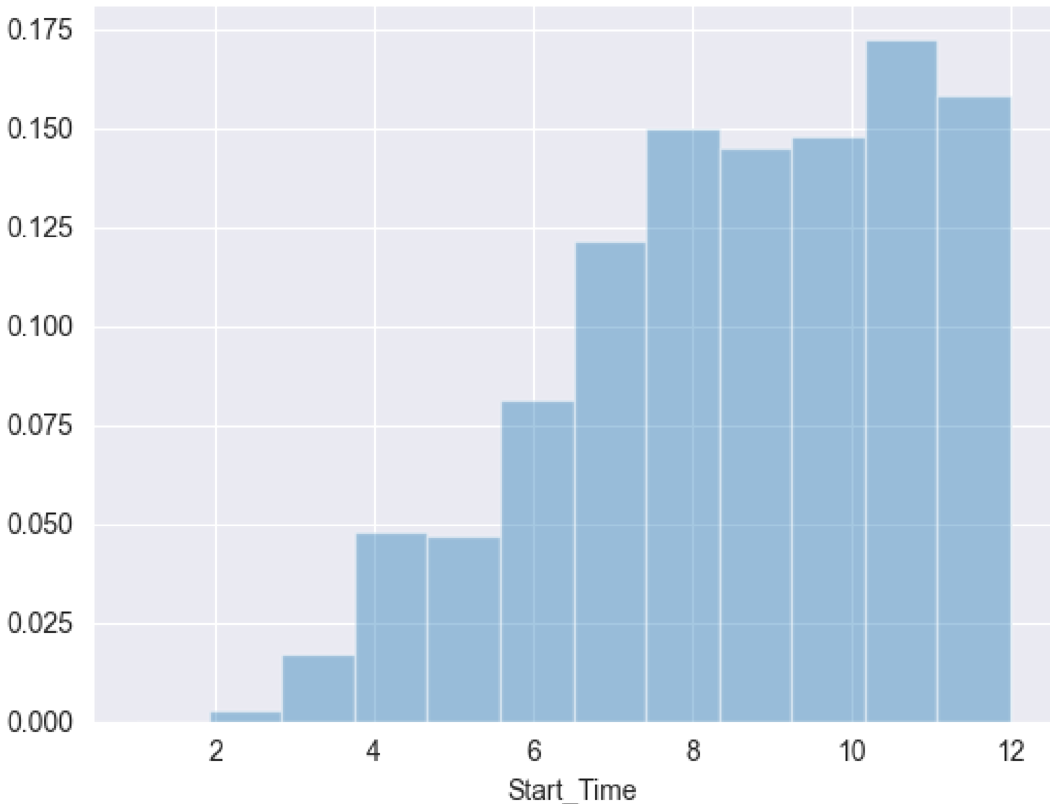
```
<Axes: xlabel='Start_Time'>
```




```
df_2019 = df[df.Start_Time.dt.year == 2016]

sns.distplot(df_2019.Start_Time.dt.month, bins=12, kde=False,
norm_hist = True, hist_kws={'rwidth': 1})
```

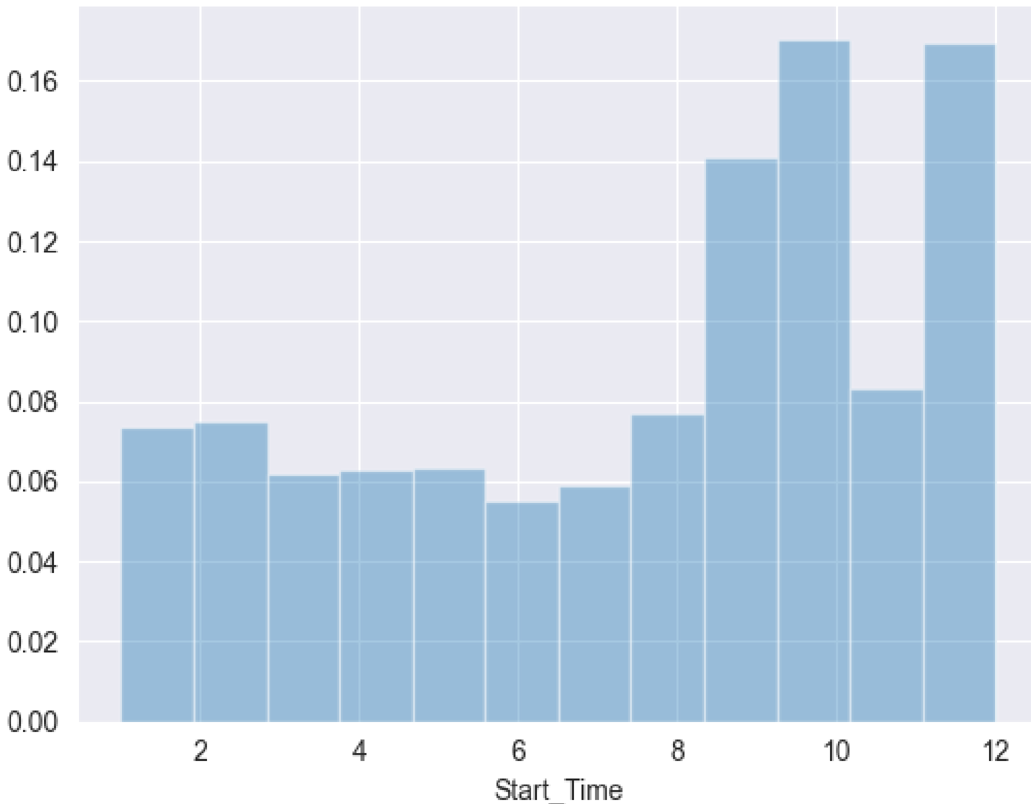
```
C:\Users\user\AppData\Local\Temp\ipykernel_18192\1101022183.py:3:
UserWarning:

`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 = True, hist_kws={'rwidth': 1})

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




```
df_2019 = df[df.Start_Time.dt.year == 2019]
df_2019_s1 = df_2019[df_2019.Source == 'Source1']
sns.distplot(df_2019_s1.Start_Time.dt.month, bins=12, kde=False,
norm_hist = True, hist_kws={'rwidth': 1})

C:\Users\user\AppData\Local\Temp\ipykernel_14092\3959586177.py:3:
UserWarning:

`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
```

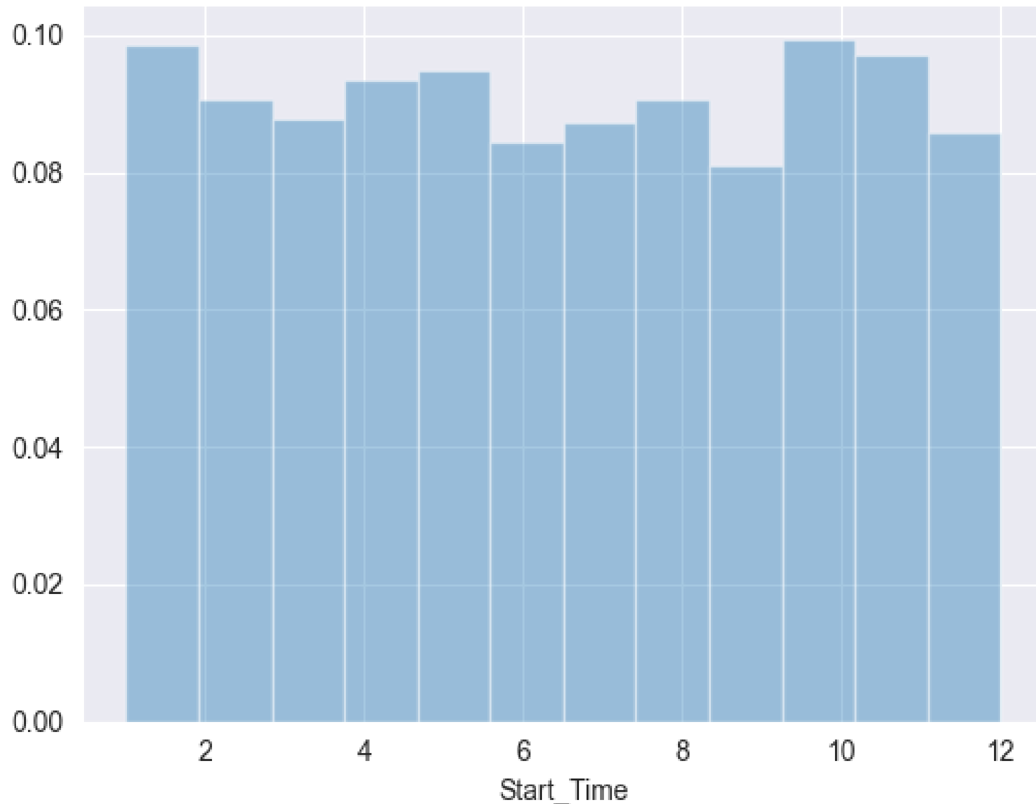
```
  sns.distplot(df_2019_s1.Start_Time.dt.month, bins=12, kde=False,
norm_hist = True, hist_kws={'rwidth': 1})

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




```
df_2019 = df[df.Start_Time.dt.year == 2019]
df_2019_s2 = df_2019[df_2019.Source == 'Source2']
sns.distplot(df_2019_s2.Start_Time.dt.month, bins=12, kde=False,
norm_hist = True, hist_kws={'rwidth': 1})

C:\Users\user\AppData\Local\Temp\ipykernel_14092\440801143.py:3:
UserWarning:

`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_s2.Start_Time.dt.month, bins=12, kde=False,
norm_hist = True, hist_kws={'rwidth': 1})
```

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

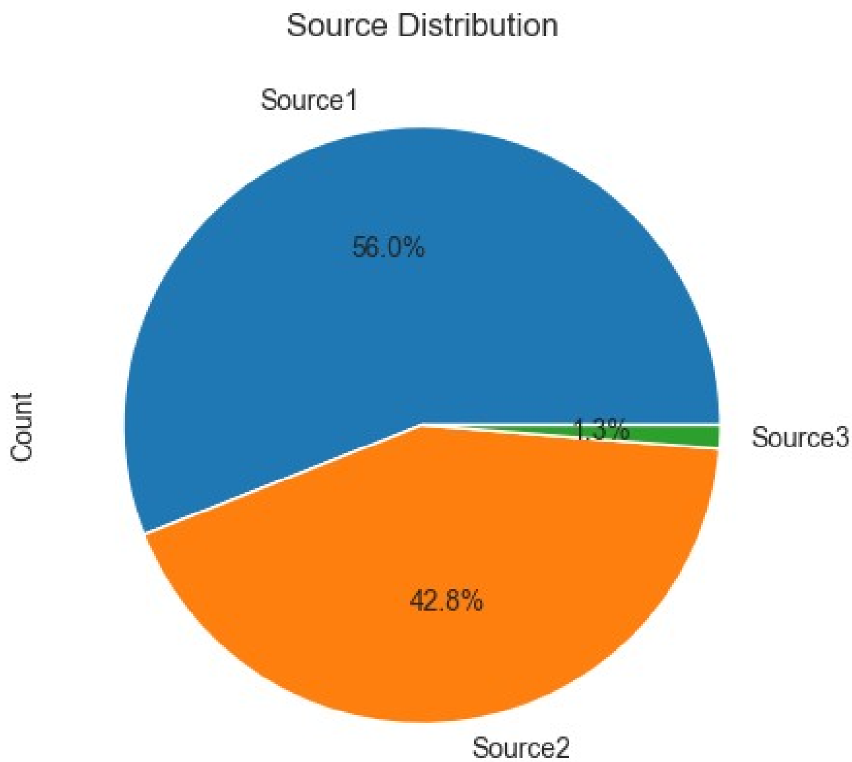


Can you expain the month-wise trend of accidents?

- Much data is missing for 2016. Maybe even 2017
- There is some issue with the Source1 data

```
df['Source'].value_counts().plot(
    kind='pie',
    autopct='%1.1f%%',  # Adds percentages to the slices
    figsize=(5, 5),

)
plt.ylabel('Count')  # Remove the default ylabel
```

```
plt.title('Source Distribution')  # Optional: Add a title
plt.show()
```



Start Latitude & Longitude

```
df.Start_Lat
```

```
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
```

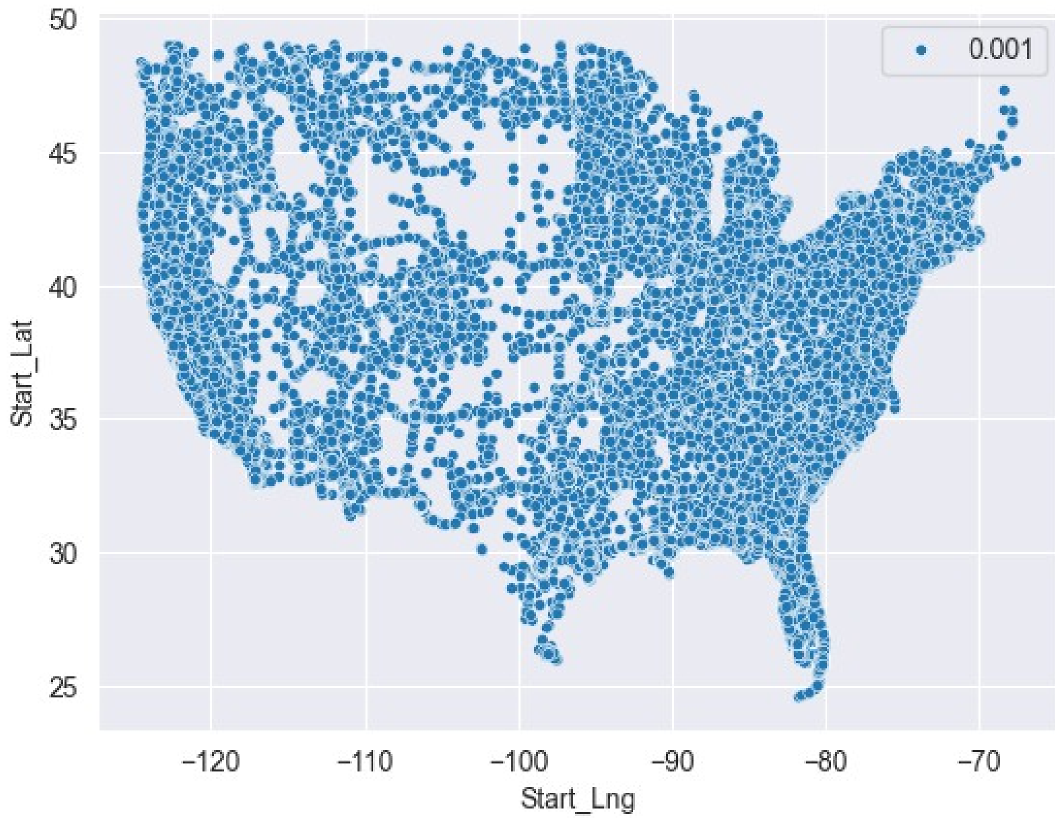
```
df.Start_Lng
```

```
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
```

```python
sample_df = df.sample(int(0.1 * len(df)))

sns.scatterplot(x = sample_df.Start_Lng, y = sample_df.Start_Lat, size
= 0.001)
```

```
<Axes: xlabel='Start_Lng', ylabel='Start_Lat'>
```




```python
import folium

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

```
(np.float64(39.865147), np.float64(-84.058723))
```

```python
for index, row in df[['Start_Lat',
'Start_Lng']].sample(100).iterrows():
    print(index, row)
```

```
4372675 Start_Lat    25.881461
Start_Lng   -80.212522
Name: 4372675, dtype: float64
481142 Start_Lat    40.110828
Start_Lng   -76.508110
Name: 481142, dtype: float64
682486 Start_Lat    42.370361
Start_Lng   -71.065208
Name: 682486, dtype: float64
3407768 Start_Lat     38.890309
Start_Lng   -123.053240
Name: 3407768, dtype: float64
7382011 Start_Lat    25.85834
Start_Lng   -80.32283
Name: 7382011, dtype: float64
894520 Start_Lat     37.824409
Start_Lng   -122.268402
Name: 894520, dtype: float64
3402775 Start_Lat    39.13877
Start_Lng   -84.53394
Name: 3402775, dtype: float64
3852015 Start_Lat    40.253140
Start_Lng   -75.307893
Name: 3852015, dtype: float64
1451503 Start_Lat     37.992165
Start_Lng   -121.252846
Name: 1451503, dtype: float64
6148085 Start_Lat     33.710829
Start_Lng   -117.188147
Name: 6148085, dtype: float64
929470 Start_Lat     33.904369
Start_Lng   -117.460419
Name: 929470, dtype: float64
643432 Start_Lat     34.072796
Start_Lng   -118.466736
Name: 643432, dtype: float64
5502774 Start_Lat     37.826521
Start_Lng   -122.277336
Name: 5502774, dtype: float64
2001668 Start_Lat    39.478081
Start_Lng   -76.247459
Name: 2001668, dtype: float64
3598642 Start_Lat     47.241901
Start_Lng   -122.392722
Name: 3598642, dtype: float64
4497465 Start_Lat    36.047663
```

```
Start_Lng   -84.000182
Name: 4497465, dtype: float64
2400429 Start_Lat    34.011192
Start_Lng   -81.148834
Name: 2400429, dtype: float64
4127946 Start_Lat    35.739545
Start_Lng   -78.788935
Name: 4127946, dtype: float64
383732 Start_Lat    39.523487
Start_Lng   -77.602005
Name: 383732, dtype: float64
2678194 Start_Lat    33.136955
Start_Lng   -80.310371
Name: 2678194, dtype: float64
6519437 Start_Lat    30.176474
Start_Lng   -81.743602
Name: 6519437, dtype: float64
6789270 Start_Lat     47.604188
Start_Lng   -122.327824
Name: 6789270, dtype: float64
1093936 Start_Lat     37.741016
Start_Lng   -121.581253
Name: 1093936, dtype: float64
558717 Start_Lat    35.327572
Start_Lng   -97.565269
Name: 558717, dtype: float64
4201691 Start_Lat     37.549983
Start_Lng   -122.025213
Name: 4201691, dtype: float64
3677127 Start_Lat     34.029202
Start_Lng   -118.010121
Name: 3677127, dtype: float64
6081061 Start_Lat     32.337547
Start_Lng   -111.038778
Name: 6081061, dtype: float64
3727710 Start_Lat     39.679888
Start_Lng   -104.828929
Name: 3727710, dtype: float64
5555035 Start_Lat     33.413937
Start_Lng   -111.909096
Name: 5555035, dtype: float64
609040 Start_Lat    41.937065
Start_Lng   -88.080177
Name: 609040, dtype: float64
3166425 Start_Lat    34.821613
Start_Lng   -82.284744
Name: 3166425, dtype: float64
3464909 Start_Lat    28.471761
Start_Lng   -81.396540
```

```
Name: 3464909, dtype: float64
7158122 Start_Lat     39.163731
Start_Lng   -120.151749
Name: 7158122, dtype: float64
1804467 Start_Lat    34.509327
Start_Lng   -80.998085
Name: 1804467, dtype: float64
4096410 Start_Lat    26.611423
Start_Lng   -80.068780
Name: 4096410, dtype: float64
3395130 Start_Lat    39.753593
Start_Lng   -86.166122
Name: 3395130, dtype: float64
4318425 Start_Lat     37.917986
Start_Lng   -121.787208
Name: 4318425, dtype: float64
3148539 Start_Lat    35.916870
Start_Lng   -78.778488
Name: 3148539, dtype: float64
5272799 Start_Lat     39.986913
Start_Lng   -105.235542
Name: 5272799, dtype: float64
4426239 Start_Lat    43.231151
Start_Lng   -73.691959
Name: 4426239, dtype: float64
1045213 Start_Lat     32.254139
Start_Lng   -110.918350
Name: 1045213, dtype: float64
5294606 Start_Lat     33.760137
Start_Lng   -117.920177
Name: 5294606, dtype: float64
6360387 Start_Lat    35.822576
Start_Lng   -78.632982
Name: 6360387, dtype: float64
4502534 Start_Lat    34.147275
Start_Lng   -80.743715
Name: 4502534, dtype: float64
2419772 Start_Lat     40.654812
Start_Lng   -111.901863
Name: 2419772, dtype: float64
4691036 Start_Lat    33.560098
Start_Lng   -81.806406
Name: 4691036, dtype: float64
7011729 Start_Lat    33.52423
Start_Lng   -86.80775
Name: 7011729, dtype: float64
2003632 Start_Lat    35.536850
Start_Lng   -97.533661
Name: 2003632, dtype: float64
```

```
7054430 Start_Lat    45.07568
Start_Lng   -93.05242
Name: 7054430, dtype: float64
5622573 Start_Lat     33.449925
Start_Lng   -112.108301
Name: 5622573, dtype: float64
1227484 Start_Lat     34.000633
Start_Lng   -117.374931
Name: 1227484, dtype: float64
411177 Start_Lat    28.450129
Start_Lng   -81.474159
Name: 411177, dtype: float64
2873363 Start_Lat    39.266644
Start_Lng   -84.607697
Name: 2873363, dtype: float64
3718191 Start_Lat    42.994456
Start_Lng   -82.445359
Name: 3718191, dtype: float64
5320189 Start_Lat    39.972781
Start_Lng   -76.677876
Name: 5320189, dtype: float64
5835692 Start_Lat    30.619275
Start_Lng   -81.649808
Name: 5835692, dtype: float64
4766426 Start_Lat     40.672512
Start_Lng   -111.871343
Name: 4766426, dtype: float64
3721497 Start_Lat    33.382930
Start_Lng   -84.672381
Name: 3721497, dtype: float64
1790858 Start_Lat    40.824169
Start_Lng   -73.225365
Name: 1790858, dtype: float64
4083108 Start_Lat    37.241729
Start_Lng   -77.659163
Name: 4083108, dtype: float64
4009696 Start_Lat     34.594327
Start_Lng   -117.256726
Name: 4009696, dtype: float64
660636 Start_Lat    40.091724
Start_Lng   -82.827980
Name: 660636, dtype: float64
1032055 Start_Lat    42.268719
Start_Lng   -71.161720
Name: 1032055, dtype: float64
7248509 Start_Lat     34.06837
Start_Lng   -117.60341
Name: 7248509, dtype: float64
502017 Start_Lat    43.011951
```

```
Start_Lng   -83.689178
Name: 502017, dtype: float64
3800477 Start_Lat     38.047584
Start_Lng   -112.572315
Name: 3800477, dtype: float64
5058499 Start_Lat    30.031932
Start_Lng   -90.005806
Name: 5058499, dtype: float64
1311227 Start_Lat     38.654758
Start_Lng   -122.922119
Name: 1311227, dtype: float64
84599 Start_Lat     33.968163
Start_Lng   -118.167870
Name: 84599, dtype: float64
3071502 Start_Lat    41.725883
Start_Lng   -87.972076
Name: 3071502, dtype: float64
4541635 Start_Lat     38.642361
Start_Lng   -121.367355
Name: 4541635, dtype: float64
1411108 Start_Lat    32.926010
Start_Lng   -96.820946
Name: 1411108, dtype: float64
3200249 Start_Lat    39.903831
Start_Lng   -75.096016
Name: 3200249, dtype: float64
2678921 Start_Lat     37.552986
Start_Lng   -122.295914
Name: 2678921, dtype: float64
3197497 Start_Lat    39.774033
Start_Lng   -76.680260
Name: 3197497, dtype: float64
6663788 Start_Lat    40.435137
Start_Lng   -78.391987
Name: 6663788, dtype: float64
3230969 Start_Lat    39.911636
Start_Lng   -86.227036
Name: 3230969, dtype: float64
3619937 Start_Lat    41.89680
Start_Lng   -70.95531
Name: 3619937, dtype: float64
3553918 Start_Lat     47.46719
Start_Lng   -122.26778
Name: 3553918, dtype: float64
5503645 Start_Lat     37.426847
Start_Lng   -105.429579
Name: 5503645, dtype: float64
1706536 Start_Lat    33.765076
Start_Lng   -84.493866
```

```
Name: 1706536, dtype: float64
2242061 Start_Lat    39.134472
Start_Lng   -84.520187
Name: 2242061, dtype: float64
956671 Start_Lat    41.830872
Start_Lng   -87.699524
Name: 956671, dtype: float64
5407104 Start_Lat    41.791391
Start_Lng   -73.684990
Name: 5407104, dtype: float64
6634385 Start_Lat     34.073807
Start_Lng   -117.752763
Name: 6634385, dtype: float64
5097931 Start_Lat    41.202239
Start_Lng   -79.950975
Name: 5097931, dtype: float64
5264876 Start_Lat    38.966041
Start_Lng   -77.140257
Name: 5264876, dtype: float64
6561267 Start_Lat     37.929643
Start_Lng   -122.387727
Name: 6561267, dtype: float64
7093885 Start_Lat    42.94420
Start_Lng   -83.66864
Name: 7093885, dtype: float64
7501171 Start_Lat     45.60013
Start_Lng   -118.50376
Name: 7501171, dtype: float64
6687425 Start_Lat    34.648811
Start_Lng   -82.562424
Name: 6687425, dtype: float64
6616338 Start_Lat    39.766441
Start_Lng   -78.283027
Name: 6616338, dtype: float64
6277016 Start_Lat    37.051166
Start_Lng   -76.672318
Name: 6277016, dtype: float64
231509 Start_Lat    40.007721
Start_Lng   -75.273949
Name: 231509, dtype: float64
4397604 Start_Lat    32.851166
Start_Lng   -96.815362
Name: 4397604, dtype: float64
4775101 Start_Lat    37.630125
Start_Lng   -77.513338
Name: 4775101, dtype: float64
3088144 Start_Lat    34.889675
Start_Lng   -82.586899
Name: 3088144, dtype: float64
```

```
6924107 Start_Lat    40.009750
Start_Lng   -75.187221
Name: 6924107, dtype: float64
5198466 Start_Lat    32.895944
Start_Lng   -96.700636
Name: 5198466, dtype: float64
4232992 Start_Lat     37.567351
Start_Lng   -122.513730
Name: 4232992, dtype: float64
```

```
from folium.plugins import HeatMap
```

```
zip(list(df.Start_Lat), list(df.Start_Lng))
```

```
<zip at 0x1ae4fb2b400>
```

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


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

```
<folium.folium.Map at 0x1ae4fa46060>
```

- Through this Heat Map we can Visualize in which Areas of the country accidents most likely occur

Top 5 States with the Highest Accidents Per Capita (2016-2023)

```
# Extract the year from the 'Start_Time' column
df['Year'] = df['Start_Time'].dt.year
```

```
df.head(2)
```

```
    ID   Source  Severity          Start_Time            End_Time
Start_Lat  \
0  A-1  Source2         3 2016-02-08 05:46:00  2016-02-08 11:00:00
39.865147
1  A-2  Source2         2 2016-02-08 06:07:59  2016-02-08 06:37:59
39.928059
```

```
   Start_Lng  End_Lat  End_Lng  Distance(mi)  ... Station   Stop  \
0 -84.058723      NaN      NaN          0.01  ...   False  False
1 -82.831184      NaN      NaN          0.01  ...   False  False

  Traffic_Calming Traffic_Signal Turning_Loop Sunrise_Sunset
Civil_Twilight  \
0           False          False        False          Night
Night
1           False          False        False          Night
Night

  Nautical_Twilight Astronomical_Twilight  Year
0             Night                 Night  2016
1             Night                   Day  2016

[2 rows x 47 columns]

# Filter accidents data for the years 2016-2023
df = df[df['Year'].between(2016, 2023)]
# Filter population data for the years 2016-2023
df_population = df_population[df_population['Year'].between(2016,
2023)]

# Group by 'State' and 'Year' to count the number of accidents
df_accidents_by_state = df.groupby(['State',
'Year']).size().reset_index(name='Accidents')

df_accidents_by_state.head(2)

  State  Year  Accidents
0    AL  2016        135
1    AL  2017       2904

df_merged = pd.merge(df_accidents_by_state, df_population,
on=['State', 'Year'], how='inner')

df_merged.head(2)

  State  Year  Accidents     Name Population
0    AL  2016        135  Alabama    4863300
1    AL  2017       2904  Alabama    4874747

# Convert the Population column to a numeric data type
df_merged['Population'] = pd.to_numeric(df_merged['Population'],
errors='coerce')



df_merged['Accidents_Per_Capita'] = df_merged['Accidents'] /
df_merged['Population']
```

```
df_merged
```

	State	Year	Accidents	Name	Population	Accidents_Per_Capita
0	AL	2016	135	Alabama	4863300.0	0.000028
1	AL	2017	2904	Alabama	4874747.0	0.000596
2	AL	2018	14100	Alabama	4887871.0	0.002885
3	AL	2019	19238	Alabama	4903185.0	0.003924
4	AL	2020	20185	.Alabama	NaN	NaN
..	...	...	...	...	...	...
383	WY	2019	112	Wyoming	578759.0	0.000194
384	WY	2020	29	.Wyoming	NaN	NaN
385	WY	2021	744	Wyoming	NaN	NaN
386	WY	2022	2075	Wyoming	NaN	NaN
387	WY	2023	418	Wyoming	NaN	NaN

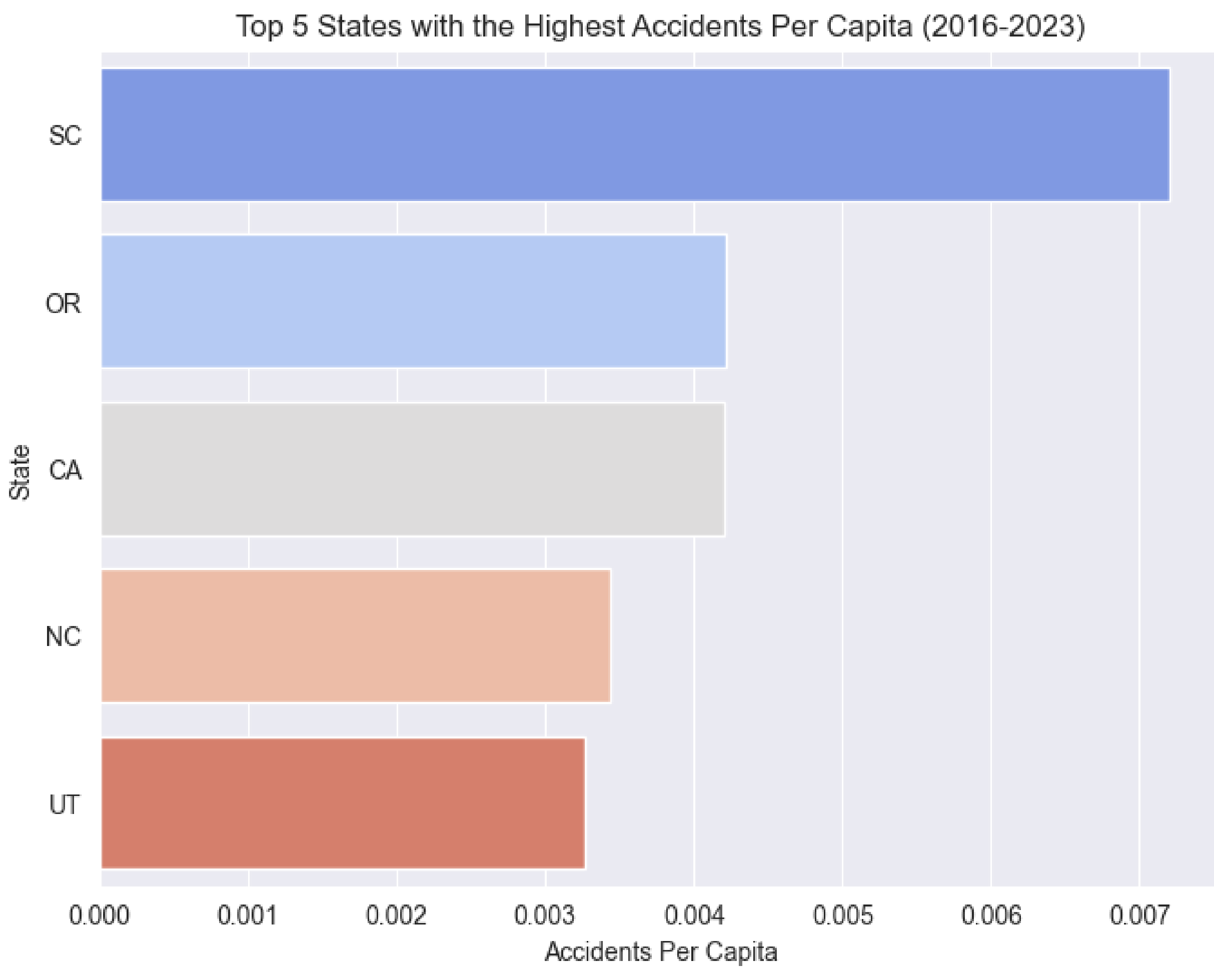
[388 rows x 6 columns]

```
sample_df = df.sample(int(0.01 * len(df)))  # Sample 1% of the
DataFrame
lat_lon_pairs = list(zip(list(sample_df.Start_Lat),
list(sample_df.Start_Lng)))

# Calculate the average accidents per capita for each city
df_top_states = df_merged.groupby('State')
['Accidents_Per_Capita'].mean().reset_index()

# Sort by accidents per capita in descending order and select the top
5 states
df_top_states = df_top_states.sort_values(by='Accidents_Per_Capita',
ascending=False).head(5)

df_top_states
```

	State	Accidents_Per_Capita
38	SC	0.007198
35	OR	0.004221
3	CA	0.004203
25	NC	0.003433
42	UT	0.003270

```
# Plot the top 5 states
plt.figure(figsize=(8, 6))
sns.barplot(x='Accidents_Per_Capita', y='State', data=df_top_states,
palette='coolwarm')
plt.title("Top 5 States with the Highest Accidents Per Capita (2016-
2023)")
plt.xlabel("Accidents Per Capita")
plt.ylabel("State")
plt.show()
```

```
C:\Users\user\AppData\Local\Temp\ipykernel_18192\3961660060.py:3:
FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `y` variable to `hue` and set
`legend=False` for the same effect.

  sns.barplot(x='Accidents_Per_Capita', y='State', data=df_top_states,
palette='coolwarm')
```



Summary:

-In this analysis, we identified the top 5 states in the United States with the highest number of accidents per capita from 2016 to 2023. The process involved merging accident data and population data, then calculating the number of accidents per capita for each state.

-The Top 5 States with the Highest Accidents Per Capita (2016-2023) are:

1. South Carolina (SC)
2. Oregon (OR)

3. California (CA)
4. North Carolina (NC)
5. Utah (UT) These states have the highest ratio of accidents to population size, which can help inform safety measures and resource planning for traffic-related issues.

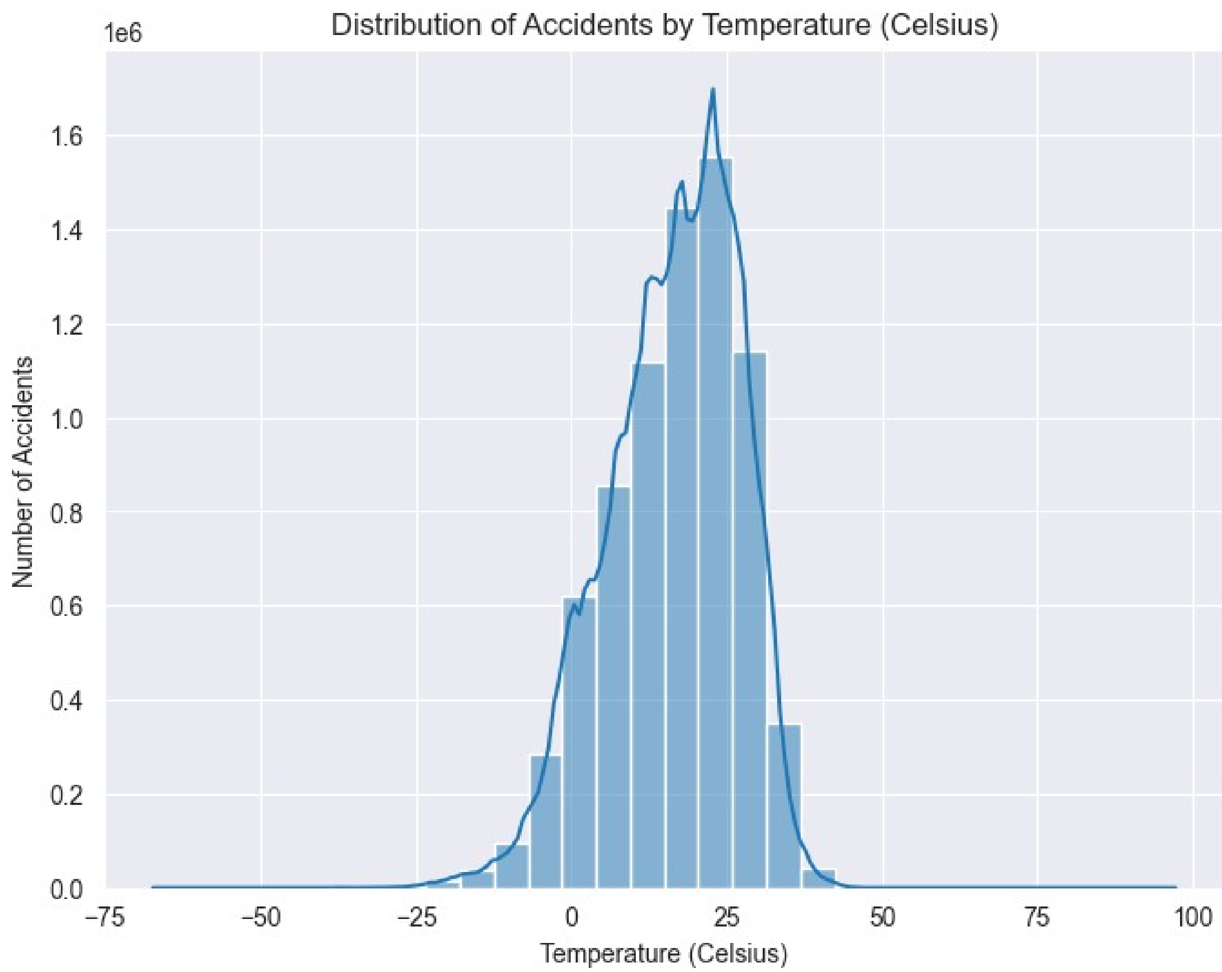
Temperature Column Analysis

Convert Fahrenheit to Celsius:

```
df['Temperature_C'] = (df['Temperature(F)'] - 32) * 5/9

print(df['Temperature_C'].describe())
```

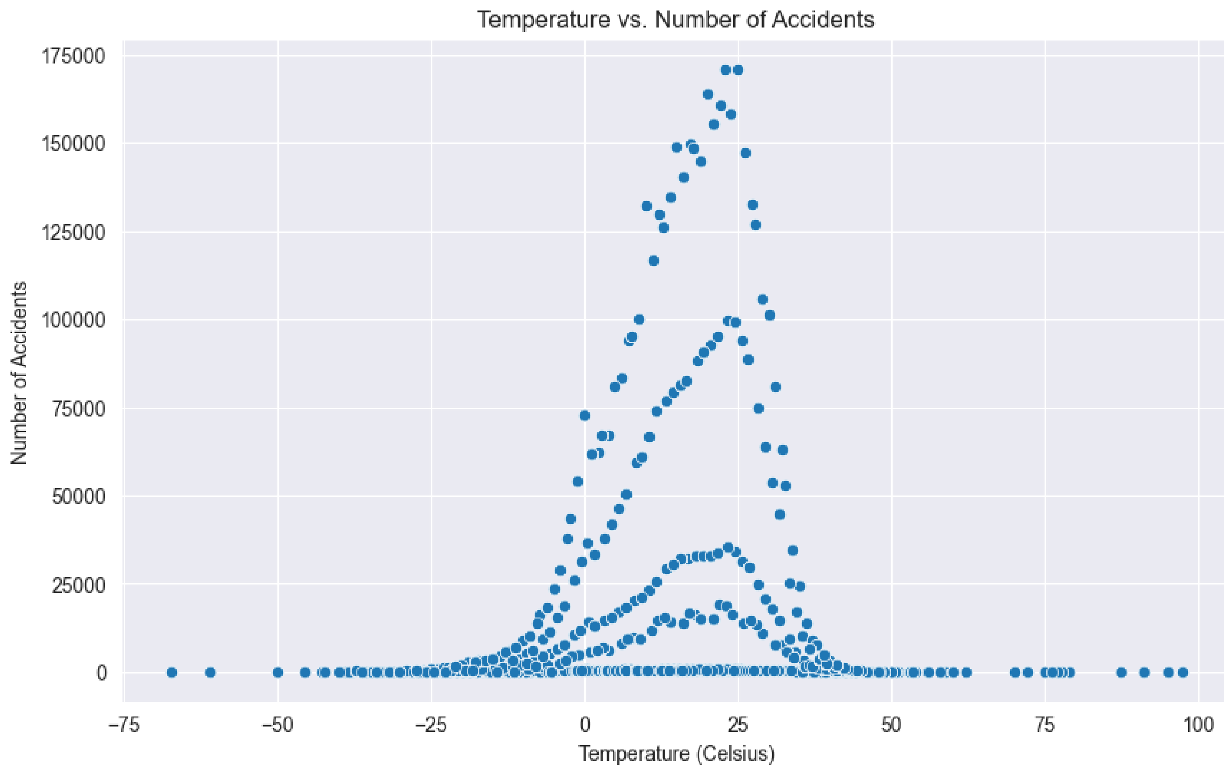
```
count    7.564541e+06
mean     1.647960e+01
std      1.056314e+01
min     -6.722222e+01
25%      9.444444e+00
50%      1.777778e+01
75%      2.444444e+01
max      9.722222e+01
Name: Temperature_C, dtype: float64
```

```
plt.figure(figsize=(8,6))
sns.histplot(df['Temperature_C'], bins=30, kde=True)
plt.title("Distribution of Accidents by Temperature (Celsius)")
plt.xlabel("Temperature (Celsius)")
plt.ylabel("Number of Accidents")
plt.show()
```




```
df_grouped =
df.groupby('Temperature_C').size().reset_index(name='Accident_Count')

df.columns

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', 'Temperature_C', 'Accident_Count_x',
       'Accident_Count_y', 'Accident_Count'],
      dtype='object')

df = df.merge(df_grouped, on='Temperature_C')

plt.figure(figsize=(10, 6))
sns.scatterplot(x='Temperature_C', y='Accident_Count', data=df)
plt.title("Temperature vs. Number of Accidents")
plt.xlabel("Temperature (Celsius)")
plt.ylabel("Number of Accidents")
plt.show()
```

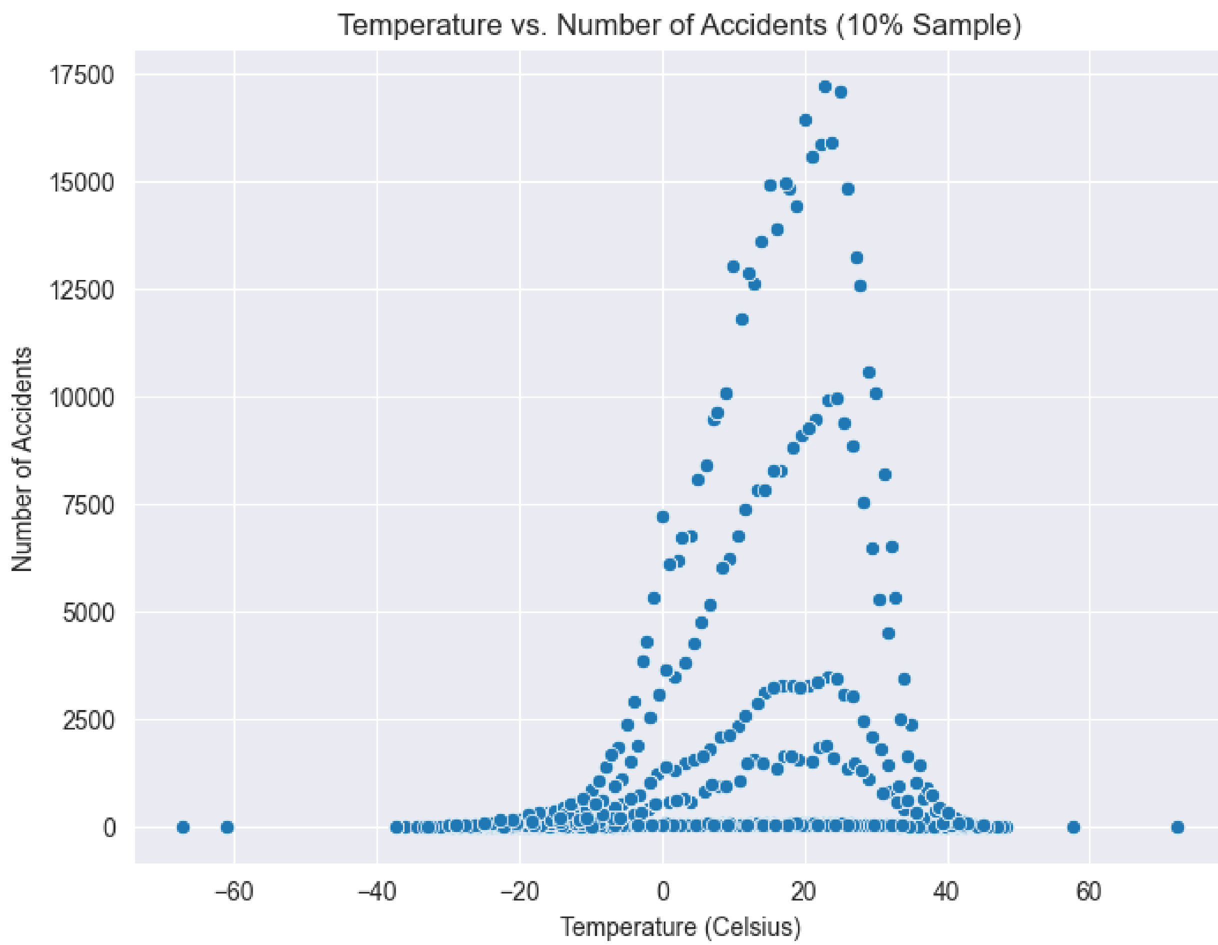



```
import random
```

Sample 10% of the data

```
# Sample 10% of the data
sample_size = int(0.1 * len(df))
df_sample = df.sample(n=sample_size, random_state=42)
```

```
# Group by Temperature and Count Accidents
df_grouped_sample =
df_sample.groupby('Temperature_C').size().reset_index(name='Accident_C
ount')

# Merge with the sample data (optional)
df_sample = df_sample.merge(df_grouped_sample, on='Temperature_C')



# Create the scatter plot
plt.figure(figsize=(8,6))
sns.scatterplot(x='Temperature_C', y='Accident_Count', data=df_sample)
plt.title("Temperature vs. Number of Accidents (10% Sample)")
plt.xlabel("Temperature (Celsius)")
plt.ylabel("Number of Accidents")
plt.show()
```



Correlation analysis:

```
correlation = df['Temperature_C'].corr(df['Accident_Count'])
print(f"Correlation between Temperature and Accidents: {correlation}")
```

```
Correlation between Temperature and Accidents: 0.30394114454048776
```

The value of 0.30 is closer to 0 than to 1, suggesting that the relationship between temperature and accidents is not very pronounced.

The analysis reveals a weak positive correlation between temperature and the number of accidents.

This suggests that as temperature increases, there is a slight tendency for the number of accidents to also increase.

However, the correlation is not strong, indicating that other factors likely play a more significant role in determining accident frequency.

Weather Condition Analysis

```
df.Weather_Condition.head(40)
```

```
0        Light Rain
1        Light Rain
2          Overcast
3     Mostly Cloudy
4     Mostly Cloudy
5        Light Rain
6          Overcast
7          Overcast
8     Mostly Cloudy
9        Light Rain
10             Rain
11       Light Rain
12         Overcast
13    Mostly Cloudy
14       Light Rain
15         Overcast
16    Mostly Cloudy
17    Mostly Cloudy
18         Overcast
19    Mostly Cloudy
20       Light Snow
21    Mostly Cloudy
22         Overcast
```

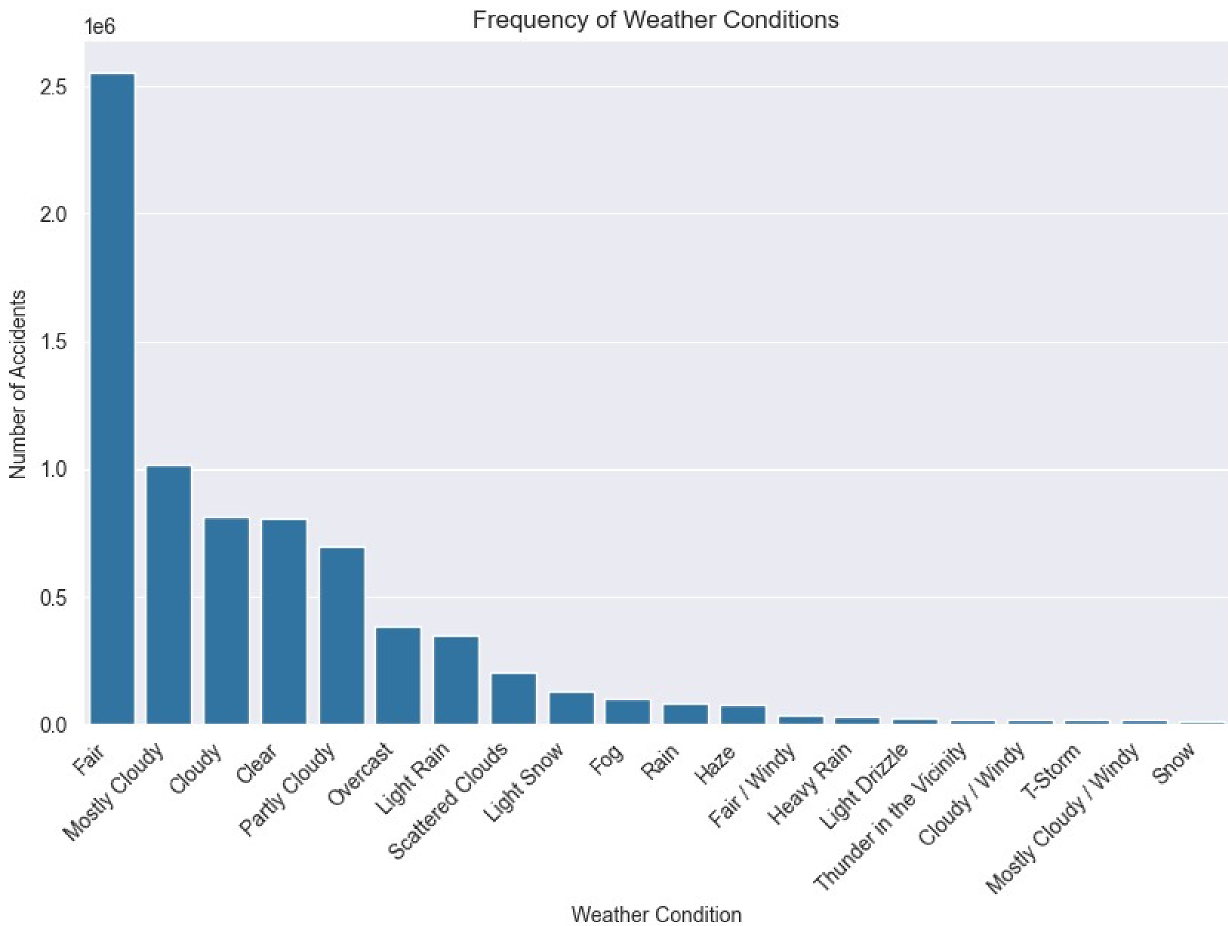
```
23          Overcast
24          Overcast
25        Light Snow
26        Light Snow
27     Mostly Cloudy
28     Mostly Cloudy
29     Mostly Cloudy
30          Overcast
31        Light Rain
32          Overcast
33          Overcast
34        Light Snow
35          Overcast
36        Light Snow
37        Light Snow
38        Light Snow
39        Light Snow
Name: Weather_Condition, dtype: object
```

```
weather_counts =
df['Weather_Condition'].value_counts().sort_values(ascending =
False).head(20)
print(weather_counts)
```

```
Weather_Condition
Fair                          2550361
Mostly Cloudy                 1013833
Cloudy                         814455
Clear                          805956
Partly Cloudy                  696566
Overcast                       381783
Light Rain                     351921
Scattered Clouds               204156
Light Snow                     128407
Fog                             98586
Rain                            83802
Haze                            75616
Fair / Windy                    35481
Heavy Rain                      32083
Light Drizzle                   22599
Thunder in the Vicinity         17484
Cloudy / Windy                  16964
T-Storm                         16742
Mostly Cloudy / Windy           16490
Snow                            15469
Name: count, dtype: int64
```

```
plt.figure(figsize=(10, 6))
sns.countplot(x='Weather_Condition', data=df,
order=weather_counts.index)
```

```
plt.xticks(rotation=45, ha='right')
plt.title('Frequency of Weather Conditions')
plt.xlabel('Weather Condition')
plt.ylabel('Number of Accidents')
plt.show()
```



- The highest number of accidents occur under Fair weather conditions suggests that weather itself might not be the direct cause of all accidents.

- Other chances might be Fair weather likely leads to increased traffic volume, which in turn increases the overall number of accidents

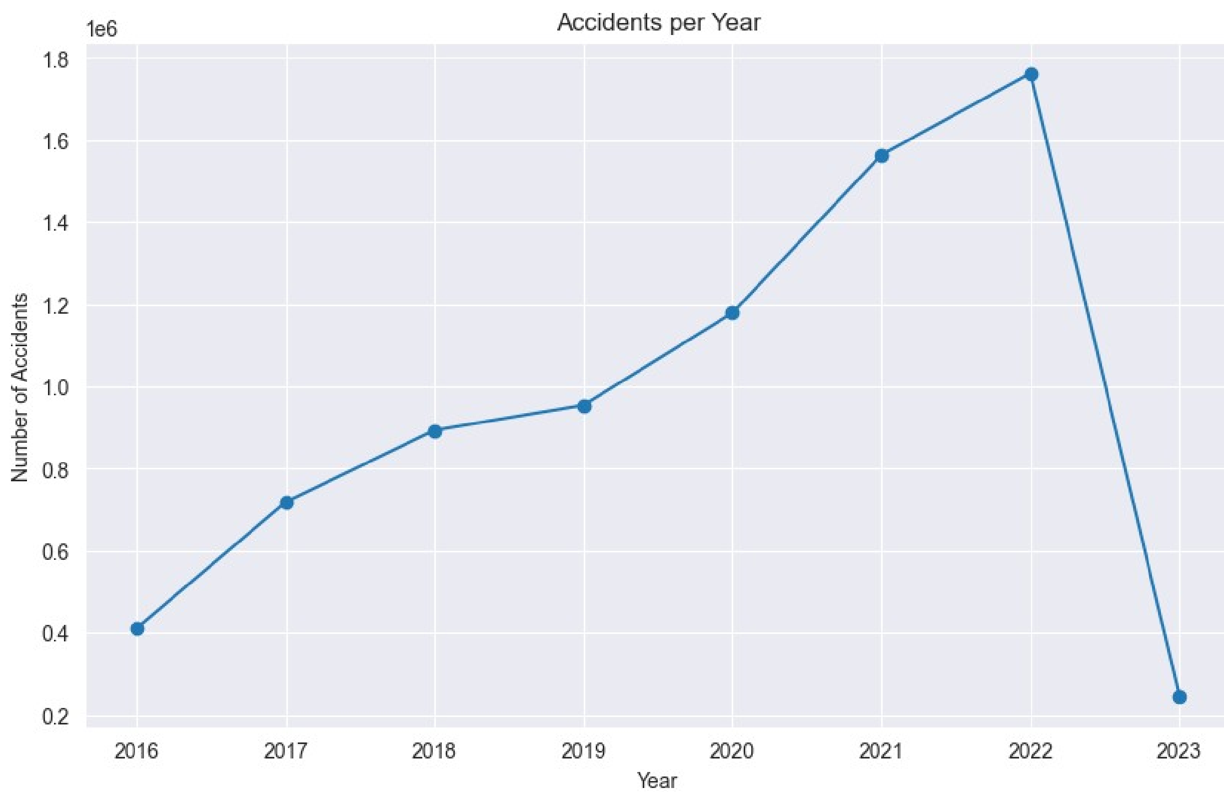
due to higher chances of collisions.

- Factors like human error, vehicle maintenance, and road conditions might play a more significant role in accidents, regardless of the weather.

But atleast we can say that Weather conditions are not the direct reason for the accidents

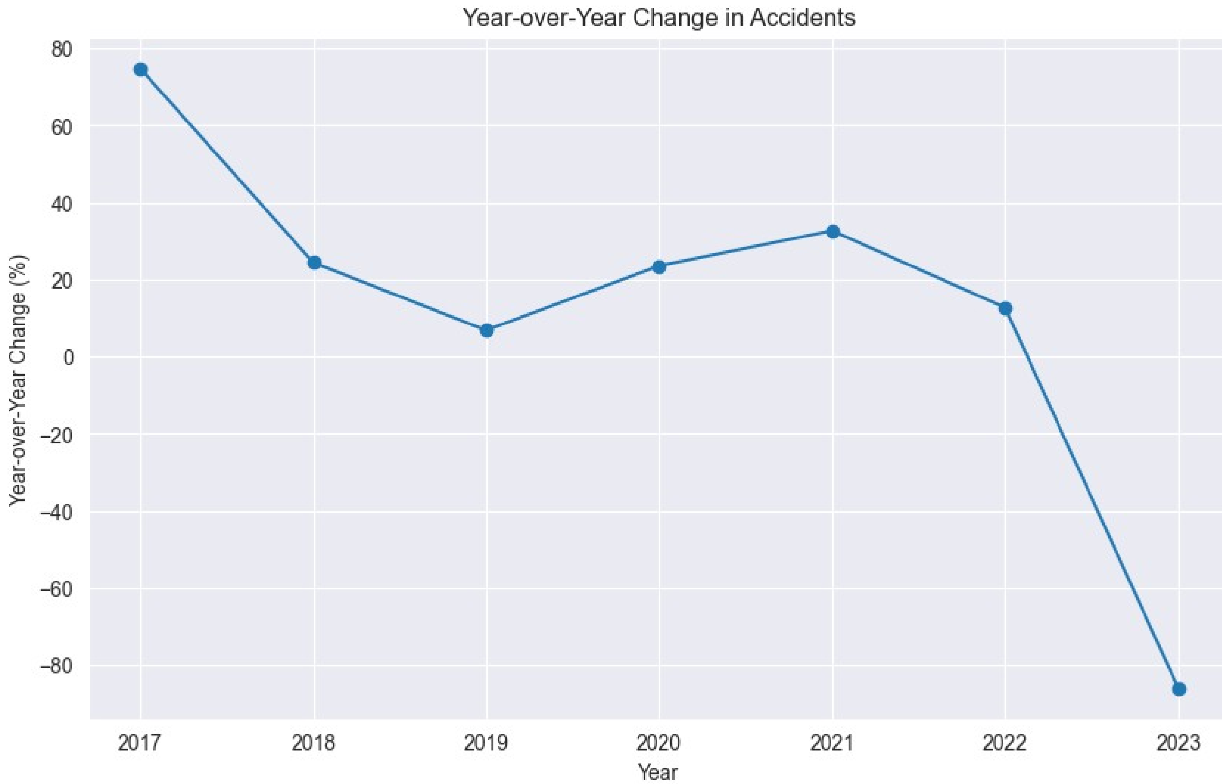
Trend of accidents year-over-year

```
df['Year'] = pd.to_datetime(df['Start_Time']).dt.year  # Create a 'Year' column

accidents_per_year = df.groupby('Year')['ID'].count() # Group the data by Year and count the number of accidents in each year.

accidents_per_year_change = accidents_per_year.pct_change() * 100  # Calculate Year-over-Year Change



plt.figure(figsize=(10, 6))
accidents_per_year.plot(marker='o')
plt.xlabel('Year')
plt.ylabel('Number of Accidents')
plt.title('Accidents per Year')
plt.show()
```




```


plt.figure(figsize=(10, 6))
accidents_per_year_change.plot(marker='o')
plt.xlabel('Year')
plt.ylabel('Year-over-Year Change (%)')
```

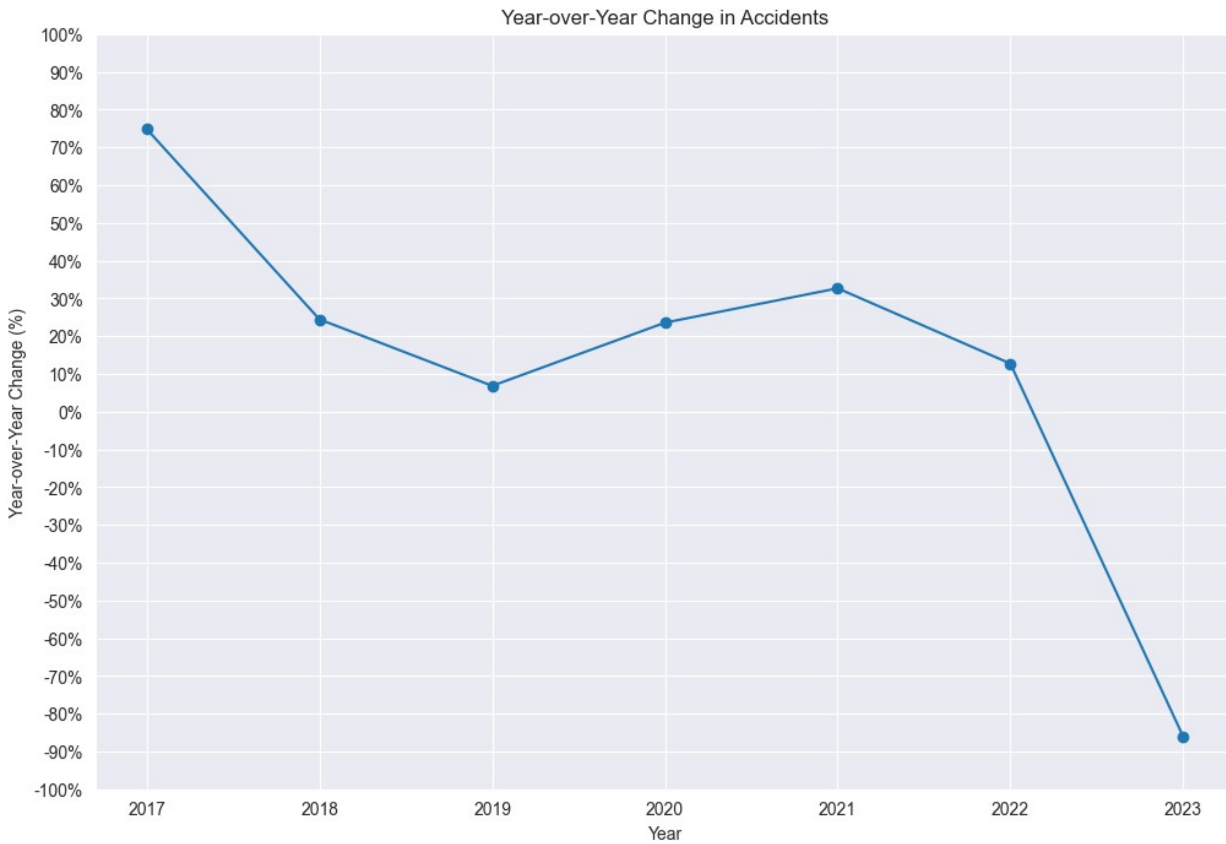
```
plt.title('Year-over-Year Change in Accidents')
plt.show()
```



More precise data points

```
plt.figure(figsize=(12, 8))

# Assuming you have 'accidents_per_year_change' variable containing
year-over-year change data

# Set ticks and labels for y-axis with 10 equal gaps between -100 and
100
ticks = range(-100, 110, 10)  # Creates a list from -100 to 100 with
steps of 10
tick_labels = [f"{x}%" for x in ticks]  # Creates labels with "%" sign

accidents_per_year_change.plot(marker='o')
plt.xlabel('Year')
plt.ylabel('Year-over-Year Change (%)')
plt.title('Year-over-Year Change in Accidents')

# Set ticks and labels for the y-axis
plt.yticks(ticks, tick_labels)

plt.grid(True)  # Add gridlines for better readability
```

```
plt.show()
```



Ask and answer questions

1. Are there more accidents in Warmer or colder areas?
2. Which 5 states have the highest number of accidents? How about per capita
3. Does New York show up in the data? if yes, why is the count lower if this is the most populated city.

4. What time of the day are accidents most frequent in?
5. Which days of the week have most accidents?
6. which months have the most accidents?
7. What is the trend of acciendents year-over-year (decreasing/increaseing)?

Summary and Conclusion

Insights:

- No data from New York
- The number of accidents per city decreases exponentially
- Less than 8% of the cities have more than 1000 yearly accidents.
- Over 1000 cities have reported just one accident (need to investigate)
- The Top 5 States with the Highest Accidents Per Capita (2016-2023) are:

1. South Carolina (SC)
2. Oregon (OR)
3. California (CA)
4. North Carolina (NC)
5. Utah (UT)