Exploratory Data Analysis 📊

> Student ID : **22200022**
>
> Name : **Jiho Kang**
>
> 1st Major : **Management**
>
> 2nd Major : **AI** Convergence

Minimum 3, Maximum 6 figures : **6 / 6**
Minimum 1, Maximum 3 tables : **3 / 3**

💡 Topics Considered Before Starting the Analysis

1. Correlation Between Accident Severity and Environmental Conditions
2. Analysis of Accident Patterns by Region
3. Analysis of Accident Patterns by Time of Day
4. Analysis of Accident Patterns by Day of the Week

I plan to narrow down the topic as I explore the data. (A completely new topic may emerge.)

💡 Background

> Traffic accidents are often viewed through a narrow lens, attributing causality to mere chance, such as the misfortune of victims or the inattentiveness of drivers. However, **have you ever considered the broader question of when and why these accidents occur?** Through this exploratory data analysis process, we aim to expand our understanding of traffic accidents. This analysis will reveal that a multitude of factors significantly influence the occurrence of traffic accidents. By recognizing these complexities, we can cultivate the ability to consider effective prevention strategies for reducing traffic incidents.

💡 Problem Statement

> Traffic accidents represent a **significant social issue globally**, resulting in countless casualties and substantial economic losses each year. Notably, the frequency and severity of accidents exhibit distinct patterns based on specific regions and times. However, there is a lack of in-depth understanding of the underlying causes and contexts of these patterns, leading to insufficient effective preventive measures. Therefore, the purpose of this study is to analyze accident occurrence patterns and to identify the relationships between these patterns and environmental factors, time of day, and types of accidents. Through this analysis, **we aim to propose concrete and practical strategies for preventing traffic accidents.**

1. Data overview 👀

> **Before starting the analysis,**
>
> I encountered a challenge :
> The **size of the accidents_data.csv file** is too large to be loaded all at once.

> **How should I approach this?**
>
> Let's consider the following :

Key concern :

The accidents_data file itself is already a sample. If we proceed with preprocessing,
such as sampling or transforming the original data, ***will the representativeness be maintained ?***

To claim representativeness, a ***comparative process*** is essential.

1. I will implement a method of loading the data in ***chunks*** by **setting a chunk size**.
(I plan to **compare** the differences between chunks and assess their statistical characteristics.)
Additionally, I will explore **missing values** : removing unnecessary variables
and using **predictive models to fill in important missing data**.

2. If the first approach proves unsuitable, I will load the entire dataset in **chunks**
and then **reduce its size** through ***sampling*** before proceeding with the analysis.
In this case, rows containing missing values will be deleted entirely
(justified by the large volume of data available).

🧭 *··· The process of identifying the most suitable methods for data retrieval ···*

```
pip install contextily
```

Show hidden output

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.cluster import KMeans
import geopandas as gpd
from shapely.geometry import Point
import statsmodels.api as sm
import contextily as ctx
```

[Method 1 Applied]

' accidents_data.csv ' is read in chunks, with two chunks loaded at a time.
Statistical characteristics of each chunk are computed and compared.
By analyzing whether datasets from different chunks exhibit similar characteristics,
I assess the extent to which the representativeness of the entire dataset is maintained,
regardless of the chosen chunk size.

```
chunks = []
chunk_sizes = [10000, 20000]  # Define chunk sizes
csv_file_path = '/content/drive/MyDrive/빅데이터종합설계/Assign1 [ EDA ]/accidents_data.csv'

# Read the first chunk with 10,000 rows
chunks.append(pd.read_csv(csv_file_path, nrows=chunk_sizes[0]))

# Read the second chunk with 20,000 rows
chunks.append(pd.read_csv(csv_file_path, skiprows=chunk_sizes[0], nrows=chunk_sizes[1]))

# Rename the columns of the second chunk to match the first chunk
chunks[1].columns = chunks[0].columns[:len(chunks[1].columns)]

# Function to calculate statistical properties
def calculate_statistics(df):
    stats = {}
    numeric_cols = df.select_dtypes(include=['number']).columns  # Select numeric columns
    for col in numeric_cols:
        stats[col] = {
```

```
            'mean': df[col].mean(),
            'std': df[col].std(),
            'min': df[col].min(),
            'max': df[col].max()
        }
    return stats

# Calculate statistical properties for the two chunks
chunk_stats = [calculate_statistics(chunk) for chunk in chunks]

# Function to compare statistics between two sets
def compare_statistics(stats1, stats2):
    comparison = {}
    common_cols = set(stats1.keys()).intersection(set(stats2.keys()))  # Get common columns
    for col in common_cols:
        comparison[col] = {
            'mean_diff': stats1[col]['mean'] - stats2[col]['mean'],
            'std_diff': stats1[col]['std'] - stats2[col]['std'],
            'min_diff': stats1[col]['min'] - stats2[col]['min'],
            'max_diff': stats1[col]['max'] - stats2[col]['max']
        }
    return comparison

# Calculate the differences between the two chunks
comparison_stats = compare_statistics(chunk_stats[0], chunk_stats[1])
comparison_df = pd.DataFrame(comparison_stats).T  # Convert to DataFrame

# Print the differences
print(comparison_df)
```

```
                    mean_diff  std_diff   min_diff    max_diff
Temperature(F)     -3.401333  5.787432 -26.100000    3.100000
Precipitation(in)  -0.003642 -0.011570   0.000000   -0.060000
Humidity(%)        -0.718778  2.892357   4.000000    0.000000
Start_Lng           2.822848  9.487821   0.003174   38.989639
End_Lat                  NaN       NaN        NaN         NaN
Severity            0.017300  0.001687   0.000000    0.000000
End_Lng                  NaN       NaN        NaN         NaN
Pressure(in)        0.015202  0.170512 -26.270000   -2.420000
Start_Lat           0.206114  0.197082   0.000000    1.832489
Wind_Chill(F)      -6.324954  6.356979 -28.600000   20.800000
Distance(mi)        0.008750  0.202569   0.000000   10.870000
Wind_Speed(mph)     0.346517 -0.035527   0.000000 -109.300000
Visibility(mi)     -0.442197  0.554429   0.100000   40.000000
```

⚡

I observe that the two chunks exhibit generally **similar patterns** ;
however, I note significant differences in certain variables.
This suggests that some chunks may contain extreme values,
indicating that the chunks might not adequately capture the **extreme range or characteristics** of the overall dataset.
I highlight that such discrepancies underscore the need for further analysis regarding the causes of accidents or environmental factors.
Additionally, I plan to attempt **Method 2** due to capacity issues.

[Method 2 Applied]

I must emphasize that Method 2 also requires the adjustment of chunk sizes for data loading, as it cannot accommodate the entire dataset at once. Therefore, it is crucial to warn that specific characteristics of the data may influence the results of the chunk analyses, necessitating caution when interpreting the findings. For instance, I will state, "The analysis results may not adequately reflect extreme values or rare situations within the overall dataset, and caution is warranted in the interpretation of these results."

```
chunks = []

# Read in chunks of 5,000 rows (initial attempt failed with 100,000) > adjusted to 10,000 > nov
chunksize = 5000
for chunk in pd.read_csv('/content/drive/MyDrive/빅데이터종합설계/Assign1 [ EDA ]/accidents_data.cs
    chunks.append(chunk)
```

```
accidents_data = pd.concat(chunks, ignore_index=True)
accidents_data.head()
```

1 to 5 of 5 entries Filter

index	ID	Source	Severity	Start_Time	End_Time	Start_Lat	Start_Lng	End_Lat	End_Lng	Distance(mi)	Description	St
0	A-1	Source2	3	2016-02-08 05:46:00	2016-02-08 11:00:00	39.865147	-84.058723	NaN	NaN	0.01	Right lane blocked due to accident on I-70 Eastbound at Exit 41 OH-235 State Route 4.	I-70
1	A-2	Source2	2	2016-02-08 06:07:59	2016-02-08 06:37:59	39.92805900000001	-82.831184	NaN	NaN	0.01	Accident on Brice Rd at Tussing Rd. Expect delays.	Brice
2	A-3	Source2	2	2016-02-08 06:49:27	2016-02-08 07:19:27	39.063148	-84.032608	NaN	NaN	0.01	Accident on OH-32 State Route 32 Westbound at Dela Palma Rd. Expect delays.	State Rout
3	A-4	Source2	3	2016-02-08 07:23:34	2016-02-08 07:53:34	39.747753	-84.20558199999998	NaN	NaN	0.01	Accident on I-75 Southbound at Exits 52 52B US-35. Expect delays.	I-75
4	A-5	Source2	2	2016-02-08 07:39:07	2016-02-08 08:09:07	39.627781	-84.188354	NaN	NaN	0.01	Accident on McEwen Rd at OH-725 Miamisburg Centerville Rd. Expect delays.	Miam Cent Rd

Show 25 per page

Like what you see? Visit the data table notebook to learn more about interactive tables.

```
Warning: Total number of columns (46) exceeds max_columns (20) limiting to first (20) columns.
```

Use a table to determine what the initial source data looks like.

```
chunk.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3394 entries, 7725000 to 7728393
Data columns (total 46 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   ID                    3394 non-null   object
 1   Source                3394 non-null   object
 2   Severity              3394 non-null   int64
 3   Start_Time            3394 non-null   object
 4   End_Time              3394 non-null   object
 5   Start_Lat             3394 non-null   float64
 6   Start_Lng             3394 non-null   float64
 7   End_Lat               3394 non-null   float64
 8   End_Lng               3394 non-null   float64
 9   Distance(mi)          3394 non-null   float64
 10  Description           3394 non-null   object
 11  Street                3394 non-null   object
 12  City                  3394 non-null   object
 13  County                3394 non-null   object
 14  State                 3394 non-null   object
 15  Zipcode               3393 non-null   object
 16  Country               3394 non-null   object
 17  Timezone              3381 non-null   object
 18  Airport_Code          3379 non-null   object
 19  Weather_Timestamp     3336 non-null   object
 20  Temperature(F)        3324 non-null   float64
 21  Wind_Chill(F)         3304 non-null   float64
```

```
 22  Humidity(%)            3323 non-null   float64
 23  Pressure(in)           3334 non-null   float64
 24  Visibility(mi)         3326 non-null   float64
 25  Wind_Direction         3315 non-null   object
 26  Wind_Speed(mph)        3315 non-null   float64
 27  Precipitation(in)      3239 non-null   float64
 28  Weather_Condition      3332 non-null   object
 29  Amenity                3394 non-null   bool
 30  Bump                   3394 non-null   bool
 31  Crossing               3394 non-null   bool
 32  Give_Way               3394 non-null   bool
 33  Junction               3394 non-null   bool
 34  No_Exit                3394 non-null   bool
 35  Railway                3394 non-null   bool
 36  Roundabout             3394 non-null   bool
 37  Station                3394 non-null   bool
 38  Stop                   3394 non-null   bool
 39  Traffic_Calming        3394 non-null   bool
 40  Traffic_Signal         3394 non-null   bool
 41  Turning_Loop           3394 non-null   bool
 42  Sunrise_Sunset         3394 non-null   object
 43  Civil_Twilight         3394 non-null   object
 44  Nautical_Twilight      3394 non-null   object
 45  Astronomical_Twilight  3394 non-null   object
dtypes: bool(13), float64(12), int64(1), object(20)
memory usage: 918.2+ KB
```

The data is read in **chunks** for processing.

The entire imported dataset is then concatenated and stored as ' accidents_data '.

To verify that the data has been successfully imported, I will utilize the ' **.info ( )** ' method for confirmation.

```
# 1. Basic Information
print(f"Sample Size: {accidents_data.shape[0]}")
print(f"Number of Variables: {accidents_data.shape[1]}")  # Print the number of variables
```

```
Sample Size: 7728394
Number of Variables: 46
```

```
# 2. Descriptive Statistics
print(accidents_data.describe(include='all'))
```

```
                ID   Source      Severity           Start_Time  \
count    7728394  7728394  7.728394e+06              7728394
unique   7728394        3           NaN              6131796
top          A-1  Source1           NaN  2021-01-26 16:16:13
freq           1  4325632           NaN                  225
mean         NaN      NaN  2.212384e+00                  NaN
std          NaN      NaN  4.875313e-01                  NaN
min          NaN      NaN  1.000000e+00                  NaN
25%          NaN      NaN  2.000000e+00                  NaN
50%          NaN      NaN  2.000000e+00                  NaN
75%          NaN      NaN  2.000000e+00                  NaN
max          NaN      NaN  4.000000e+00                  NaN

                    End_Time     Start_Lat     Start_Lng       End_Lat  \
count                7728394  7.728394e+06  7.728394e+06  4.325632e+06
unique               6705355           NaN           NaN           NaN
top      2021-11-22 08:00:00           NaN           NaN           NaN
freq                     112           NaN           NaN           NaN
mean                     NaN  3.620119e+01 -9.470255e+01  3.626183e+01
std                      NaN  5.076079e+00  1.739176e+01  5.272905e+00
min                      NaN  2.455480e+01 -1.246238e+02  2.456601e+01
25%                      NaN  3.339963e+01 -1.172194e+02  3.346207e+01
50%                      NaN  3.582397e+01 -8.776662e+01  3.618349e+01
75%                      NaN  4.008496e+01 -8.035368e+01  4.017892e+01
max                      NaN  4.900220e+01 -6.711317e+01  4.907500e+01

             End_Lng  Distance(mi)  ... Roundabout  Station     Stop  \
count   4.325632e+06  7.728394e+06  ...    7728394  7728394  7728394
unique           NaN           NaN  ...          2        2        2
top              NaN           NaN  ...      False    False    False
freq             NaN           NaN  ...    7728145  7526493  7514023
mean   -9.572557e+01  5.618423e-01  ...        NaN      NaN      NaN
std     1.810793e+01  1.776811e+00  ...        NaN      NaN      NaN
min    -1.245457e+02  0.000000e+00  ...        NaN      NaN      NaN
25%    -1.177543e+02  0.000000e+00  ...        NaN      NaN      NaN
50%    -8.802789e+01  3.000000e-02  ...        NaN      NaN      NaN
75%    -8.024709e+01  4.640000e-01  ...        NaN      NaN      NaN
```

```
max    -6.710924e+01  4.417500e+02  ...        NaN      NaN      NaN

       Traffic_Calming Traffic_Signal Turning_Loop Sunrise_Sunset  \
count          7728394        7728394      7728394        7705148
unique               2              2            1              2
top              False          False        False            Day
freq           7720796        6584622      7728394        5334553
mean               NaN            NaN          NaN            NaN
std                NaN            NaN          NaN            NaN
min                NaN            NaN          NaN            NaN
25%                NaN            NaN          NaN            NaN
50%                NaN            NaN          NaN            NaN
75%                NaN            NaN          NaN            NaN
max                NaN            NaN          NaN            NaN

       Civil_Twilight Nautical_Twilight Astronomical_Twilight
count         7705148           7705148               7705148
unique              2                 2                     2
top               Day               Day                   Day
freq          5695619           6076156               6377548
mean              NaN               NaN                   NaN
```

```
# 3. Data Range
print("Data Range (Minimum and Maximum Values):")
for column in accidents_data.columns:
    if pd.api.types.is_numeric_dtype(accidents_data[column]):
        min_value = accidents_data[column].min()
        max_value = accidents_data[column].max()
        print(f"{column}: Min = {min_value}, Max = {max_value}")
```

```
Data Range (Minimum and Maximum Values):
Severity: Min = 1, Max = 4
Start_Lat: Min = 24.5548, Max = 49.002201
Start_Lng: Min = -124.623833, Max = -67.113167
End_Lat: Min = 24.566013, Max = 49.075
End_Lng: Min = -124.545748, Max = -67.10924200000001
Distance(mi): Min = 0.0, Max = 441.75
Temperature(F): Min = -89.0, Max = 207.0
Wind_Chill(F): Min = -89.0, Max = 207.0
Humidity(%): Min = 1.0, Max = 100.0
Pressure(in): Min = 0.0, Max = 58.63
Visibility(mi): Min = 0.0, Max = 140.0
Wind_Speed(mph): Min = 0.0, Max = 1087.0
Precipitation(in): Min = 0.0, Max = 36.47
Amenity: Min = False, Max = True
Bump: Min = False, Max = True
Crossing: Min = False, Max = True
Give_Way: Min = False, Max = True
Junction: Min = False, Max = True
No_Exit: Min = False, Max = True
Railway: Min = False, Max = True
Roundabout: Min = False, Max = True
Station: Min = False, Max = True
Stop: Min = False, Max = True
Traffic_Calming: Min = False, Max = True
Traffic_Signal: Min = False, Max = True
Turning_Loop: Min = False, Max = False
```

```
# 4. Missing Value Analysis
print("Number of Missing Values:")
print(accidents_data.isnull().sum())
```

```
Number of Missing Values:
ID                        0
Source                    0
Severity                  0
Start_Time                0
End_Time                  0
Start_Lat                 0
Start_Lng                 0
End_Lat             3402762
End_Lng             3402762
Distance(mi)              0
Description               5
Street                10869
City                    253
County                    0
State                     0
Zipcode                1915
Country                   0
Timezone               7808
```

```
Airport_Code               22635
Weather_Timestamp         120228
Temperature(F)            163853
Wind_Chill(F)            1999019
Humidity(%)               174144
Pressure(in)              140679
Visibility(mi)            177098
Wind_Direction            175206
Wind_Speed(mph)           571233
Precipitation(in)        2203586
Weather_Condition         173459
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             23246
Civil_Twilight             23246
Nautical_Twilight          23246
Astronomical_Twilight      23246
dtype: int64
```

⚡

Through the preceding processes,
I have successfully understood the structure of the data I will be using.
It is evident that there are a **substantial number of missing values present.**
Before directly addressing these missing values, it is crucial to comprehend
the reasons behind their occurrence. For instance, certain variables
may contain information that is not always necessary to record in accident reports.

1.

For continuous variables, one can utilize a **predictive model to replace the missing values.**

2.

Alternatively, one may choose to **remove the rows containing missing values.**
However, this approach could lead to data loss. Nonetheless, given the considerable
volume of data currently being worked with, it is assessed
that this will not pose a significant issue.

```
# Remove rows with missing values in variables that are not needed for analysis
accidents_data = accidents_data.dropna(subset=['Description', 'Street', 'City', 'Zipcode', 'Airport_Code'])
```

- In the case of location data such as **Start_Lat & Lng** and **End_Lat & Lng**,
  these are crucial for identifying the precise location of accidents.
  It would be ideal to replace missing values using predictive models.
- Similarly, for the weather-related variables, using predictive models
  to impute missing values would be the preferred approach.

Unfortunately, despite attempts to use predictive models to impute these missing values,
I **encountered a limitation** where the system kept restarting due to memory constraints.

Thus, instead of simply removing the missing data, I decided to focus on imputing
the missing values for location variables using predictive models,
as these are **too important to discard.**

```
predictors = ['Start_Lat', 'Start_Lng', 'Distance(mi)', 'Severity']

# Data without missing values
no_na = accidents_data.dropna(subset=['End_Lat', 'End_Lng'])
with_na = accidents_data[accidents_data['End_Lat'].isna()]

# Ensure no_na also has no missing values in predictors
no_na = no_na.dropna(subset=predictors)

X_train = no_na[predictors]
y_train_lat = no_na['End_Lat']
y_train_lng = no_na['End_Lng']

# Training the models
model_lat = LinearRegression()
model_lng = LinearRegression()

model_lat.fit(X_train.compute(), y_train_lat.compute())
model_lng.fit(X_train.compute(), y_train_lng.compute())

# Predicting missing values
X_missing = with_na[predictors]
predicted_lat = model_lat.predict(X_missing.compute())
predicted_lng = model_lng.predict(X_missing.compute())

# Imputing the predicted values into the missing data
accidents_data.loc[accidents_data['End_Lat'].isna(), 'End_Lat'] = predicted_lat
accidents_data.loc[accidents_data['End_Lng'].isna(), 'End_Lng'] = predicted_lng

accidents_data.head(10)
```

✔

This approach is not feasible, as the data required
for prediction contains missing values, making it impossible to generate predictions.
Therefore, I have ultimately decided to **exclude any rows containing missing values from the dataset.**

To verify the representativeness of the data mentioned in the Data Overview section,
I will create two datasets through data sampling. Subsequently, I will **compare**
the statistical information of each dataset to ascertain whether they exhibit similar trends.

```
sample_size = 100000

# Perform the first sampling from the original DataFrame
accidents_1 = accidents_data.sample(n=sample_size, random_state=42)
accidents_1.to_csv('accidents_1.csv', index=False)

# Perform the second sampling from the original DataFrame
# - using a different random seed for a different sample)
accidents_2 = accidents_data.sample(n=sample_size, random_state=43)
accidents_2.to_csv('accidents_2.csv', index=False)

df1 = pd.read_csv('accidents_1.csv')
df2 = pd.read_csv('accidents_2.csv')

# Check the information of the DataFrames (columns, data types, etc.)
info1 = df1.info()
info2 = df2.info()

# Compare the statistical summary information of the two DataFrames
describe1 = df1.describe(include='all')
describe2 = df2.describe(include='all')
```

```
describe1
describe2

accidents_data = pd.read_csv('accidents_1.csv')
```

Show hidden output

> After comparing the results of describe1 and describe2,
> I've found that there are some slight differences,
> but they are **very similar overall.**
> I believe that using one of these datasets,
> obtained through sampling, for analysis would not pose any issues.

```
# Filter out rows with no missing values
clean_data = accidents_data.dropna()

# Sample 10,000 rows (returns the entire dataset if it's smaller than 10,000)
sample_size = min(10000, len(clean_data))
accidents_pp = clean_data.sample(n=sample_size, random_state=42)

accidents_pp.to_csv('accidents_pp', index=False)
print("Shape of the sampled data without missing values:", accidents_pp.shape)
```

```
Shape of the sampled data without missing values: (10000, 46)
```

```
print(accidents_pp.isnull().sum())
```

```
ID                    0
Source                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
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(in)          0
Visibility(mi)        0
Wind_Direction        0
Wind_Speed(mph)       0
Precipitation(in)     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
```

2. Univariate analysis 🔬

Presentation of key variables from various aspects.

```
accidents_data = pd.read_csv('/content/accidents_pp')
```

```
# Statistical summary for continuous variables
continuous_vars = accidents_data.select_dtypes(include=[np.number])
continuous_vars.describe()
```

1 to 8 of 8 entries Filter

index	Severity	Start_Lat	Start_Lng	End_Lat	End_Lng	Distance(mi)	Temper
count	10000.0	10000.0	10000.0	10000.0	10000.0	10000.0	
mean	2.0748	36.17192554851896	-95.04793362352073	36.17222493911896	-95.04745911472072	0.8592155	
std	0.38317051291063153	5.372556192980263	17.951115405479715	5.372827241890133	17.95003541181239	1.812113265419251	19.562411
min	1.0	25.096292	-124.35776	25.09669	-124.35776	0.0	
25%	2.0	33.1307515	-117.50086825	33.12738575	-117.50228200000001	0.068	
50%	2.0	36.093796	-87.207768	36.090882	-87.2122905	0.258	
75%	2.0	40.254534976048525	-80.2097357825	40.2530005	-80.209216	0.922	
max	4.0	48.995778	-70.026776	48.998144	-70.033968	55.312	

Show 25 per page

Like what you see? Visit the data table notebook to learn more about interactive tables.

Before diving into the analysis, I used the '.describe()' method
to understand the tendencies of the data I would be using.
I printed the results, but due to poor readability, I generated a table instead.

Examination of Each Variable

- **Severity**
  With a mean value of 2.078, accident severity tends to be **concentrated around mild levels 2**.
  However, the range from a minimum of **1 to a maximum of 4** indicates
  that severity is distributed across a wide spectrum.
  Accident severity is one of the most critical variables in accident analysis.
  Investigating how more severe accidents are associated
  with specific environmental conditions
  (e.g., weather, time of day) could yield valuable insights.

- **Temperature (F)**
  The average temperature is approximately 61.21°F,
  indicating moderate conditions that are neither excessively hot nor cold.
  However, the **temperature spans a vast range, from -35°F to 111°F.**
  The standard deviation of 19.39 shows substantial variability in temperature.
  Temperature can have a significant influence on accident occurrence.
  For instance, it would be worthwhile to analyze how extreme cold
  or heat affects the likelihood of accidents.

- **Humidity**
  The mean humidity is about 63.63%, suggesting that **accidents predominantly occur
  in relatively humid conditions.** With values ranging from a minimum of 3% to a maximum of 100%,
  accidents are observed across a broad spectrum of humidity levels. Further analysis is needed
  to explore how high or low humidity levels impact accident frequency.

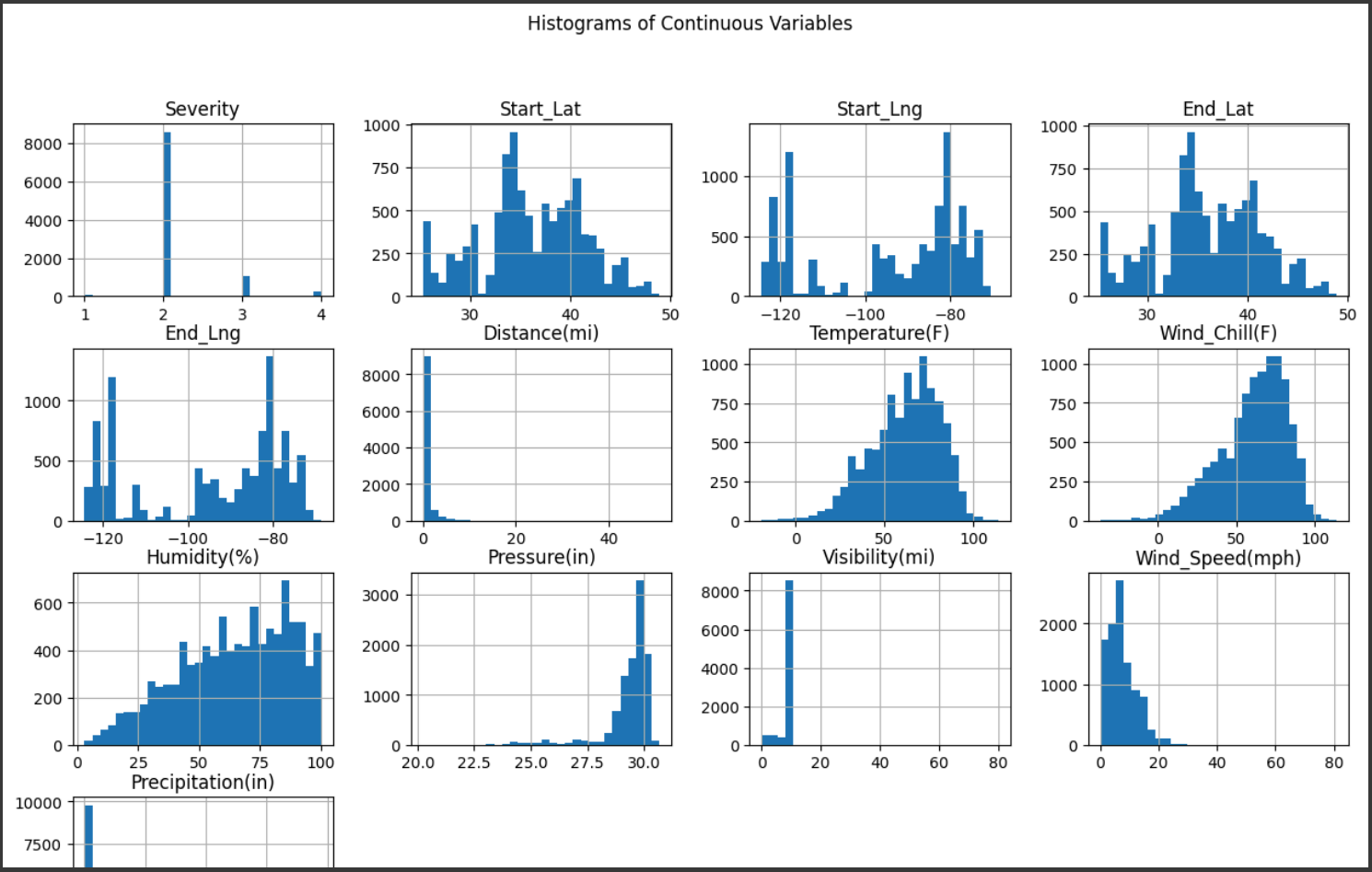
- **Visibility (mi)** The average visibility distance is approximately **9.07 miles**.
  It ranges from **0 miles** (complete lack of visibility) to a maximum of 50 miles.

Given that poor visibility conditions may increase accident occurrence,
it would be beneficial to analyze the impact of visibility distance on accidents.

- **Wind Speed (mph)**
  The average wind speed is around 7.40 mph, indicating that accidents
  tend to occur at relatively low wind speeds. However, accidents are recorded across
  a wind speed range from 0 to 55 mph. Further analysis is necessary
  to determine the extent to which wind speed influences accident occurrence,
  as higher wind speeds could potentially increase the risk of accidents.

- **Distance (mi)**
  The average accident distance is approximately 0.83 miles,
  suggesting that most accidents occur over **short distances**.
  The range spans from 0 to 65 miles. It would be insightful to explore
  how accident severity or patterns differ between short- and long-distance accidents.

```
# Plot histograms of continuous variables
accidents_data.select_dtypes(include=[np.number]).hist(bins=30, figsize=(15, 10))
plt.suptitle('Histograms of Continuous Variables')
plt.show()
```



I applied **histogram plots** to the **continuous variables** for the following reason :
While countable variables can be easily understood through textual representations like frequency,
I found it challenging to grasp the shape of continuous variables solely by looking at the numbers.
Therefore, I decided to create histograms for better visualization.

```
# Check frequency counts of categorical variables
categorical_vars = accidents_data.select_dtypes(include=[object])
categorical_vars.describe(include=[object])
```

1 to 4 of 4 entries Filter

index	ID	Source	Start_Time	End_Time	Description	Street	City	County	State	Zipcode	Country	Timezone	Airport_Code	Weatl
count	10000	10000	10000	10000	10000	10000	10000	10000	10000	10000	10000	10000	10000	10000
unique	10000	1	9973	9991	9571	5476	2613	768	48	6574	1	4	1002	9350
top	A-5562876	Source1	2022-09-21 08:04:00	2020-12-16 17:35:42	Accident	I-5 N	Miami	Los Angeles	CA	91761	US	US/Eastern	KMIA	2021-
freq	1	10000	2	2	17	107	405	629	2383	21	10000	4787	172	4

Show 25 per page

Like what you see? Visit the data table notebook to learn more about interactive tables.

By presenting the **frequency** of categorical variables in a table,
I aimed to enhance clarity and facilitate **easy comparisons** of common values across different variables.
This format contributes to understanding data quality and trends, and aids in identifying patterns and outliers.

Examination of Each Variable

Start_Time & End_Time
The time-related data for accidents is predominantly unique ;
however, some accidents may occur or conclude at the "same time."
An analysis of accident occurrence patterns considering time zones may be feasible.

Description
The descriptions of accidents exhibit considerable diversity ;
nonetheless, certain descriptions are recurrent.
This indicates that similar types of accidents occur frequently.

Street
A total of 5,510 entries exist, with the street "I-5 S" appearing 106 times.
Accidents frequently occur on specific roadways,
suggesting the potential for further analysis of high-incident areas.

City
There are 2,577 unique city names, with "Miami" being the most frequently mentioned, appearing 381 times.
It may be possible to determine whether accidents are concentrated in specific cities,
which could be beneficial for analyzing regional accident patterns.

State
Records indicate 47 states, with "CA" (California) appearing most frequently at 2,512 instances.
This allows for the examination of whether accidents are concentrated in particular states.

Zipcode
There are 6,574 unique postal codes, with "91761" being the most common, appearing 24 times.
It may be possible to assess whether accidents frequently occur in certain postal codes,
which could be useful for analyzing regional accident risk.

Timezone
Four unique time zones are present, with the "US/Eastern" time zone appearing most frequently at 4,708 instances.
This allows for the investigation of whether accidents occur more frequently within specific time zones,
indicating a high incidence of accidents in the US/Eastern time zone.

Weather_Timestamp
There are 987 unique weather timestamps.

This data can be utilized to analyze the impact of weather information on accidents.

Wind_Direction
There are 23 unique wind direction values, with "CALM" being the most common, appearing 1,761 times.
It is possible to determine whether specific wind directions recur during accident occurrences,
facilitating an analysis of the influence of wind direction on accidents.

Weather_Condition
A total of 53 unique weather conditions are recorded, with "Fair" appearing most frequently at 4,835 instances.
This allows for an analysis of the variations in accident rates based on weather conditions.

Sunrise_Sunset, Civil_Twilight, Nautical_Twilight, Astronomical_Twilight
Each of these variables has two unique values, with accidents primarily occurring during the "Day."
This suggests that accidents predominantly happen during daylight hours, providing an opportunity
for further analysis of accident occurrence patterns by time of day.

> Through the comprehensive analysis I conducted, I have examined intriguing
> and distinctive variables that warrant further exploration.

2.1 **Severity**

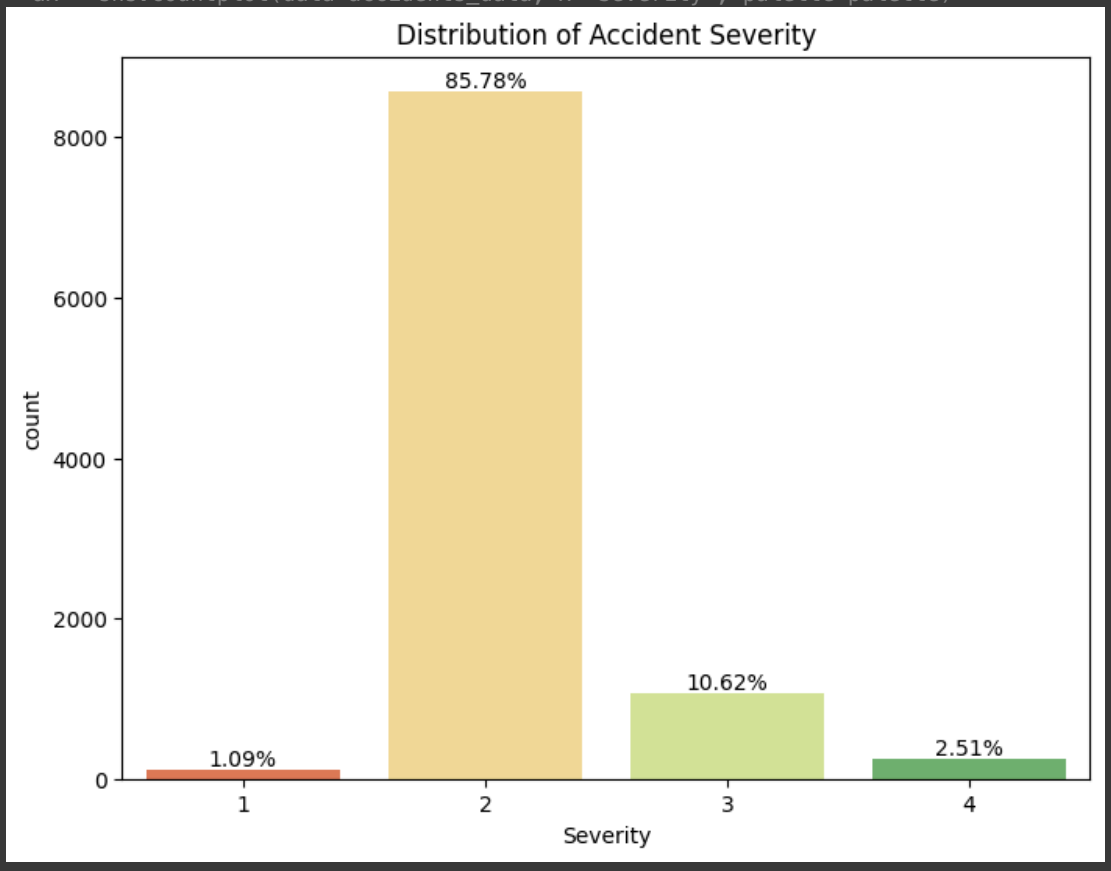
```
# Basic statistics for the Severity variable
severity_stats = accidents_data['Severity'].describe()

# Frequency counts for each value in the Severity variable
severity_freq = accidents_data['Severity'].value_counts()

print("Severity Basic Stats:\n", severity_stats)
print("\nSeverity Frequency:\n", severity_freq)
```

```
Severity Basic Stats:
 count    10000.000000
mean         2.074800
std          0.383171
min          1.000000
25%          2.000000
50%          2.000000
75%          2.000000
max          4.000000
Name: Severity, dtype: float64

Severity Frequency:
 Severity
2    9424
4     316
3     188
1      72
Name: count, dtype: int64
```

```
# Set the color palette (color variation based on severity)
palette = sns.color_palette("RdYlGn", n_colors=4)  # Define the color palette

# Calculate the distribution of accident severity (percentage calculation)
severity_counts = accidents_data['Severity'].value_counts()
severity_percent = (severity_counts / severity_counts.sum()) * 100

# Visualization
plt.figure(figsize=(8, 6))
ax = sns.countplot(data=accidents_data, x='Severity', palette=palette)

# Display percentage (%) above each bar
for p in ax.patches:
    height = p.get_height()
    # Show percentage value (set coordinates and format to 2 decimal places)
    ax.text(p.get_x() + p.get_width() / 2.,
            height + 50,
```

```
            '{:.2f}%'.format((height / len(accidents_data)) * 100),
            ha="center")

# Set graph title and X-axis label
plt.title('Distribution of Accident Severity')
plt.xlabel('Severity')

# Display the graph
plt.show()
```

```
Severity Basic Stats:
 count    10000.000000
mean         2.145500
std          0.443113
min          1.000000
25%          2.000000
50%          2.000000
75%          2.000000
max          4.000000
Name: Severity, dtype: float64

Severity Frequency:
 Severity
2    8578
3    1062
4     251
1     109
Name: count, dtype: int64
<ipython-input-18-8a2b49cc43bc>:10: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable t

  ax = sns.countplot(data=accidents_data, x='Severity', palette=palette)
```



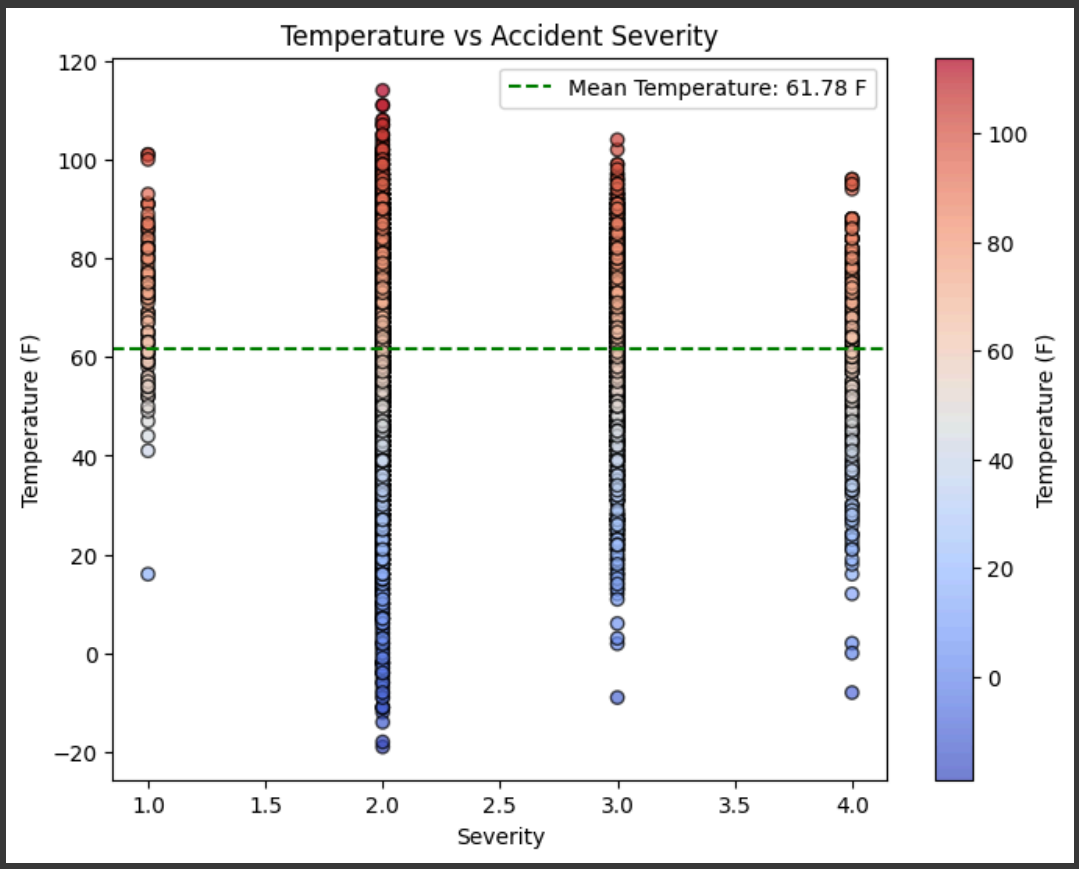
I considered ***Severity*** to be a crucial variable in analyzing accident data.
This is because a higher Severity level allows for a better understanding of other variables that influence the seriousness of an accident. Through this analysis, I was able to identify the frequency of accidents at varying levels of severity. The results indicated a notable occurrence of accidents between Severity levels 2 and 3, suggesting that moderate-level accidents are prevalent. Therefore, I highlighted Severity as an important variable, as it serves as a benchmark for assessing the **impact of various environmental factors in the context of accident analysis**.

2.2 Temperature (F)

```
# Basic statistics for the Temperature variable
temp_stats = accidents_data['Temperature(F)'].describe()
print("Temperature Basic Stats:\n", temp_stats)
```

```
Temperature Basic Stats:
 count    10000.000000
mean        61.777060
std         19.237451
min        -19.000000
25%         49.000000
50%         64.000000
75%         77.000000
max        114.000000
Name: Temperature(F), dtype: float64
```

```
plt.figure(figsize=(8,6))

# Set temperature as the color value for each data point
points = plt.scatter(x=accidents_data['Severity'],
                     y=accidents_data['Temperature(F)'],
                     c=accidents_data['Temperature(F)'],
                     cmap='coolwarm',
                     edgecolor='k', alpha=0.7)

cbar = plt.colorbar(points)
cbar.set_label('Temperature (F)')

# Calculate the mean temperature
mean_temp = accidents_data['Temperature(F)'].mean()
plt.axhline(mean_temp, color='green', linestyle='--',
            label=f'Mean Temperature: {mean_temp:.2f} F')

plt.title('Temperature vs Accident Severity')
plt.xlabel('Severity')
plt.ylabel('Temperature (F)')
plt.legend()
plt.show()
```



Temperature is a significant environmental factor that can influence road conditions and driver physiological responses. Low temperatures can lead to issues such as slippery or icy roads,

while high temperatures pose risks related to fatigue and heat-related illnesses. I believe it is essential to examine the relationship between temperature and accident severity. For instance, if severe accidents occur more frequently within a specific temperature range, it suggests that temperature may act as a contributing factor to accident risk. Notably, the data indicates that accidents do not merely cluster around a specific temperature but rather span a wide range of temperatures.

+ **Wind_Chill (F)**

The variable Wind_Chill (F) is very similar to Temperature (F).
Therefore, I plan to use Temperature (F) instead of Wind_Chill (F).

```
# Basic statistics for the Wind_Chill variable
wind_chill_stats = accidents_data['Wind_Chill(F)'].describe()
print("Wind Chill Basic Stats:\n", wind_chill_stats)
```

```
Wind Chill Basic Stats:
 count    10000.000000
mean        60.476890
std         21.506891
min        -37.000000
25%         48.000000
50%         64.000000
75%         77.000000
max        114.000000
Name: Wind_Chill(F), dtype: float64
```

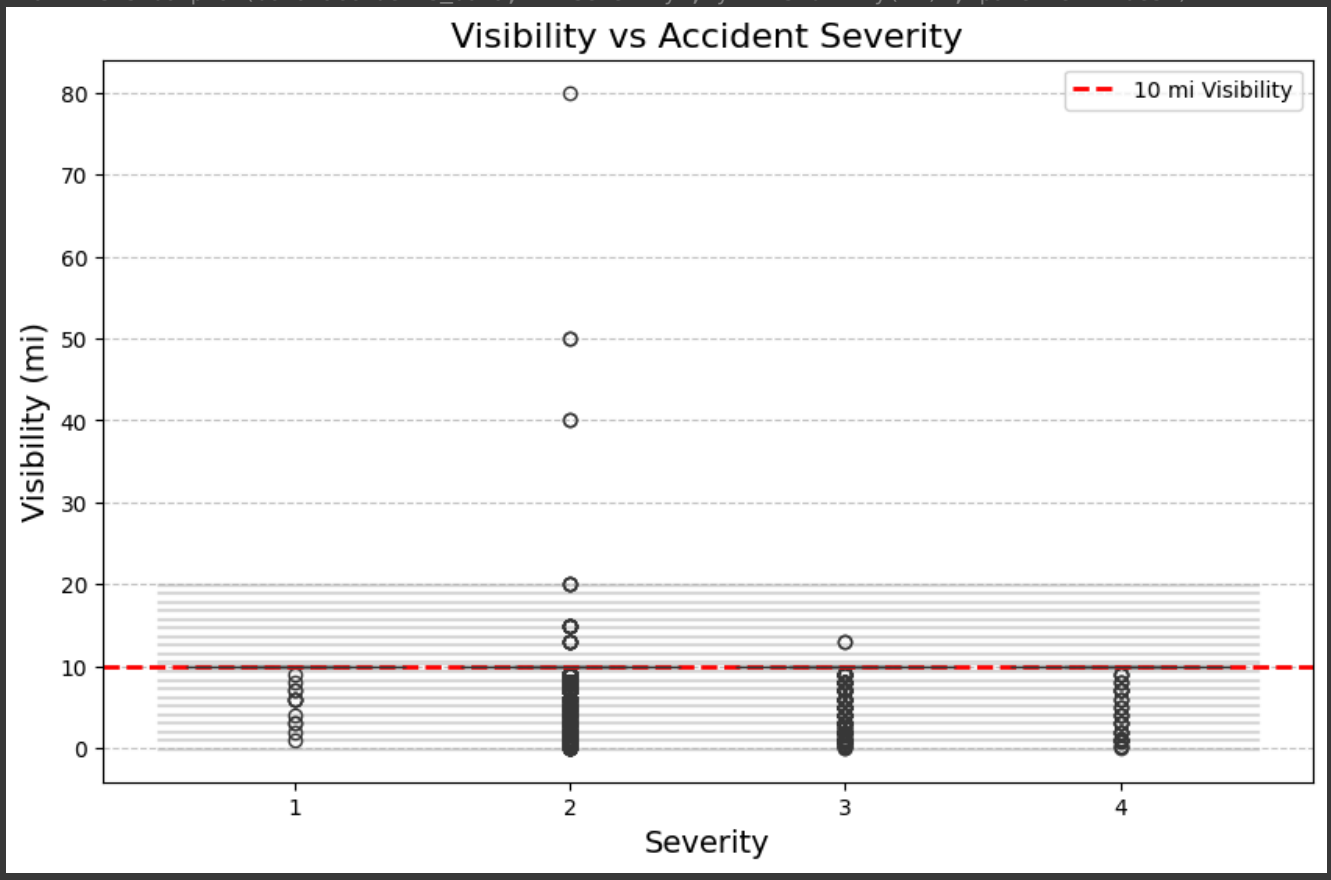
2.3 **Visibility (mi)** 🧹

```
# Basic statistics for the Visibility variable
visibility_stats = accidents_data['Visibility(mi)'].describe()
print("Visibility Basic Stats:\n", visibility_stats)
```

```
Visibility Basic Stats:
 count    10000.000000
mean         9.061114
std          2.595125
min          0.000000
25%         10.000000
50%         10.000000
75%         10.000000
max         80.000000
Name: Visibility(mi), dtype: float64
```

```
plt.figure(figsize=(10, 6))

ax = sns.boxplot(data=accidents_data, x='Severity', y='Visibility(mi)', palette="Blues")

for visibility in np.linspace(0, 20, num=20):
    ax.fill_between([-0.5, 3.5], visibility - 0.1, visibility + 0.1, color='grey', alpha=0.2)

plt.axhline(y=10, color='red', linestyle='--', label='10 mi Visibility', linewidth=2)
plt.title('Visibility vs Accident Severity', fontsize=16)
plt.xlabel('Severity', fontsize=14)
plt.ylabel('Visibility (mi)', fontsize=14)
plt.legend()
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()
```

```
<ipython-input-23-d22e8a1e67a8>:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable t

  ax = sns.boxplot(data=accidents_data, x='Severity', y='Visibility(mi)', palette="Blues")
```



Visibility refers to the distance that a driver can visually perceive.
When visibility is limited, the risk of accidents increases dramatically.
It is essential to examine whether the severity of accidents increases as visibility decreases.
If accidents frequently occur in weather conditions with reduced visibility, it may be considered a significant factor.
Most accidents appear to occur at visibility levels between **0 to 10**, where visibility is nearly nonexistent.

+ Humidity (%)

Humidity (%.) represents the moisture content in the air.
In contrast, Visibility (mi) refers to the driver's line of sight, which has a direct impact on driving safety.
Therefore, it is more efficient to focus on Visibility (mi) rather than Humidity (%).
As such, Visibility will be treated as the primary variable for analysis, and I'm open
to eliminating Humidity from consideration if deemed unnecessary later on.

```
# Basic statistics for the Humidity variable
humidity_stats = accidents_data['Humidity(%)'].describe()
print("Humidity Basic Stats:\n", humidity_stats)
```

```
Humidity Basic Stats:
 count    10000.00000
mean        64.43270
std         22.88014
min          3.00000
25%         48.00000
50%         67.00000
75%         84.00000
max        100.00000
Name: Humidity(%), dtype: float64
```

2.4 Weather_Condition 🌤️

```
# Frequency analysis of unique values in the Weather_Condition variable
weather_condition_freq = accidents_data['Weather_Condition'].value_counts()
print("Weather Condition Frequency:\n", weather_condition_freq)
```

Show hidden output

```
# Select the top 10 weather conditions
top_10_weather_conditions = weather_condition_freq.head(10)

print("Top 10 Weather Conditions:\n")
#top_10_weather_conditions
print(top_10_weather_conditions)
```

```
Top 10 Weather Conditions:

Weather_Condition
Fair             4696
Cloudy           1430
Mostly Cloudy    1345
Partly Cloudy     957
Light Rain        463
Light Snow        198
Fog               153
Rain              102
Haze               82
Fair / Windy       55
Name: count, dtype: int64
```

I think that weather is one of the key factors that directly influence the occurrence of accidents.
Conditions such as rain, snow, and fog can deteriorate road conditions and limit visibility.
One surprising insight obtained from the analysis is that accidents occurred most frequently under fair weather conditions.

+ I intend to remove the **Precipitation (in)** 🌧️ variable,

as I believe that the **Weather_Condition** variable sufficiently captures the information regarding precipitation.

```
# Basic statistics for the Precipitation variable
precipitation_stats = accidents_data['Precipitation(in)'].describe()
print("Precipitation Basic Stats:\n", precipitation_stats)
```

```
Precipitation Basic Stats:
 count    10000.000000
mean         0.005680
std          0.042327
min          0.000000
25%          0.000000
50%          0.000000
75%          0.000000
max          1.950000
Name: Precipitation(in), dtype: float64
```

2.5 **Sunrise_Sunset**

```
# Frequency analysis of unique values for the Sunrise_Sunset variable
sunrise_sunset_freq = accidents_data['Sunrise_Sunset'].value_counts()
print("Sunrise Sunset Frequency:\n", sunrise_sunset_freq)
```

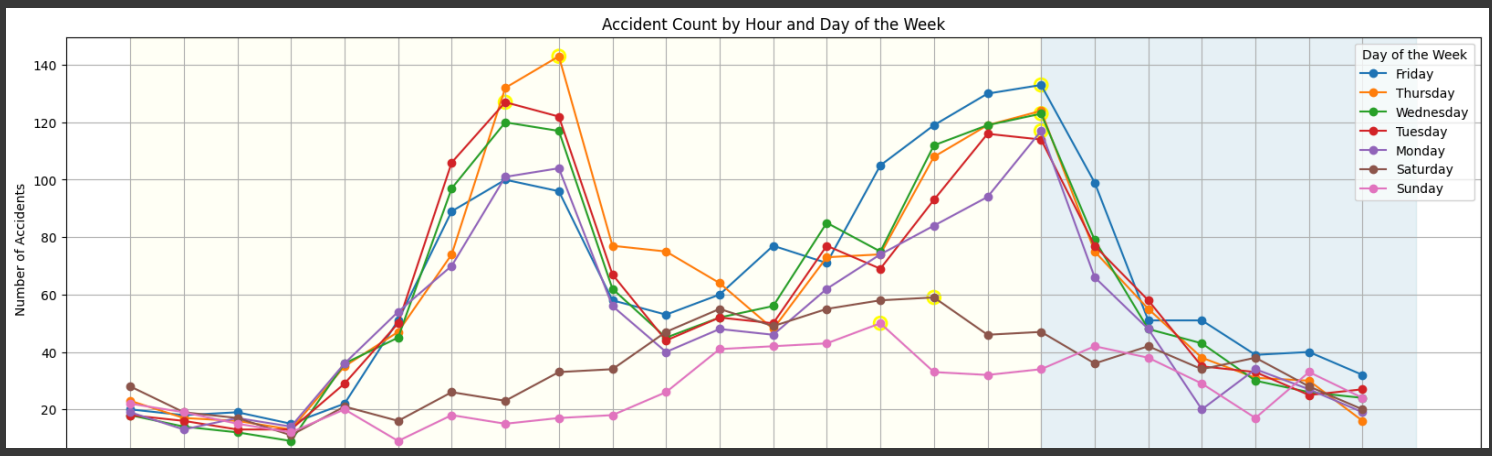
```
Sunrise Sunset Frequency:
 Sunrise_Sunset
Day      6763
Night    3237
Name: count, dtype: int64
```

The **Sunrise_Sunset** variable, which indicates whether accidents occurred around **sunrise or sunset**, can influence visibility and traffic volume. This variable allows for the analysis of whether accidents happen during the **day or night**. By examining this data, it is possible to determine which time period, day or night,

is associated with more severe accidents. *I hypothesize that accidents are likely to be more serious at night ,*
as drivers may experience slower reaction times and decreased attentiveness due to visibility issues.
In fact, the results indicate that the ***number of accidents occurring during the day is more*** than twice
that of those occurring at night.

Furthermore, I believe it would be insightful to analyze which days of the **week**
and which **time periods** experience the highest number of accidents.
To conduct this analysis, I plan to utilize the **Time (Start_Time)** variable
along with the **Day of the Week (Start_Time)**.

```
accidents_data['Start_Time'] = pd.to_datetime(accidents_data['Start_Time'], errors='coerce')
accidents_data['Day_of_Week'] = accidents_data['Start_Time'].dt.day_name()
total_accidents_by_day = accidents_data['Day_of_Week'].value_counts()

# Order of days sorted in descending order
sorted_days = total_accidents_by_day.index

# Visualization
plt.figure(figsize=(20, 6))
plt.axvspan(0, 17, color='lightyellow', alpha=0.3)
plt.axvspan(17, 24, color='lightblue', alpha=0.3)

# Calculate accident counts by hour
hourly_weekday_accidents = accidents_data.groupby([accidents_data['Start_Time'].dt.hour, 'Day_of_Week']).size().unsta

# Line graph: Add average lines for each day of the week
for day in sorted_days:
    plt.plot(hourly_weekday_accidents.index, hourly_weekday_accidents[day], marker='o', label=day)

    max_index = hourly_weekday_accidents[day].idxmax()
    max_value = hourly_weekday_accidents[day].max()
    plt.scatter(max_index, max_value, edgecolor='yellow', facecolor='none', s=100, linewidth=2)  # Add outline to the

plt.xticks(ticks=range(24), labels=[f'{hour:02d}:00' for hour in range(24)])
plt.title('Accident Count by Hour and Day of the Week')
plt.xlabel('Hour of the Day')
plt.ylabel('Number of Accidents')
plt.legend(title='Day of the Week', loc='upper right')
plt.grid()
plt.show()
```



I created a line graph to identify which days of the week and specific time periods have the highest accident rates,
while also assessing the overall trends in accident rates across different times, regardless of the day.
This analysis reveals that accident rates are significantly higher during daytime hours compared to evening hours,
particularly noting that the peak accident occurrences occur at 07:00 and 17:00.

✔ Part 2 : Summary of Insights from Univariate Analysis

1. **Severity**
   The severity of accidents plays a crucial role in understanding
   the relationships among various variables. Contrary to expectations,
   it was observed that minor incidents, specifically those classified as severity level 2,

occurred most frequently.

2. **Temperature**
   Temperature serves as an environmental factor influencing accident occurrences.
   Contrary to my initial hypothesis that accidents would be more prevalent
   at specific temperatures, the analysis revealed a wide range of temperature values.
   The direct impact of temperature on the analysis remains uncertain.

3. **Visibility**
   Visibility is directly related to accident risk, and the analysis confirmed that
   accidents frequently occur under conditions of reduced visibility.
   However, several incidents also transpired in conditions of high visibility,
   suggesting potential factors such as driver inattention, poor road conditions,
   or violations of traffic laws. Further investigation into these incidents would be beneficial.

4. **Weather Condition**
   Weather conditions significantly impact accident occurrences. Notably, it was surprising
   to find that the highest number of accidents occurred under "Fair" weather conditions.

5. **Sunrise/Sunset**
   The time of day—whether accidents occur during daylight or nighttime—affects driver visibility.
   Contrary to the hypothesis that nighttime accidents would be more prevalent due to reduced visibility,
   the analysis revealed that daytime accidents were more than double those that occurred at night.

💡 Redefining the Direction of Exploratory Data Analysis (EDA)

1. Research Question and Objective
   : Examine how environmental factors affect accident severity to inform traffic safety policies.
2. Key Variables and Analysis
   : Investigate the impact of temperature, visibility, weather, and time of day on accident severity.
3. Analysis Methodology
   : Use regression models and visualizations to predict and analyze accident severity.
4. Social Value Creation
   : Propose safety policies, educational programs, and awareness campaigns based on findings.

Please note that plans are subject to change.

3. Multivariate analysis

Presenation of hidden patterns between variables (correlation, clustering, etc.)

💡 Analytic Method

3.1 Correlation

```
numeric_data = accidents_data.select_dtypes(include=[np.number])
correlation_matrix = numeric_data.corr()

# Drop the self-correlation for 'Temperature(F)' to analyze its correlation with other variables
Temperature_correlation = correlation_matrix['Temperature(F)'].drop('Temperature(F)')
```

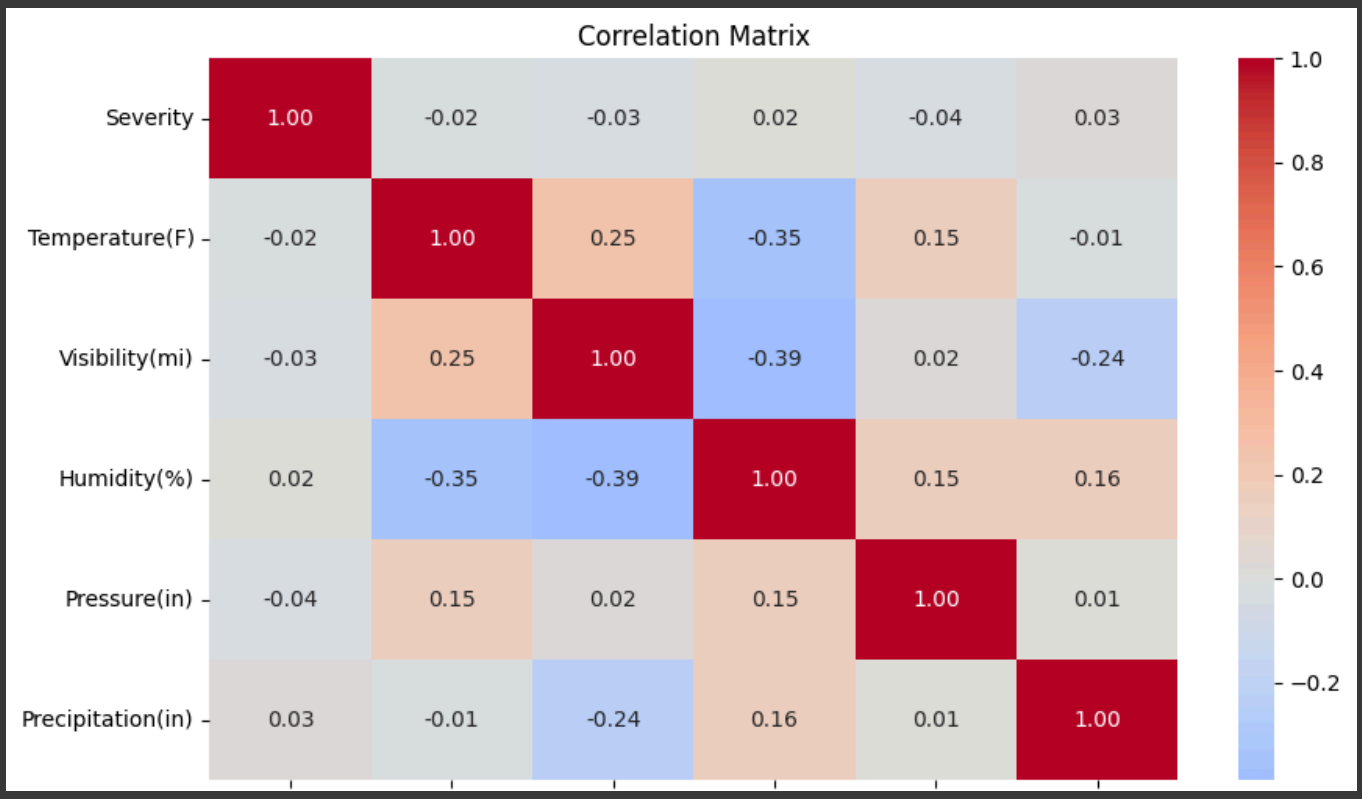
```
Temperature_correlation_sorted = Temperature_correlation.abs().sort_values(ascending=False)

print(Temperature_correlation_sorted)
```

```
Wind_Chill(F)        0.993711
Start_Lat            0.453206
End_Lat              0.453204
Humidity(%)          0.350505
Visibility(mi)       0.250565
Pressure(in)         0.148206
Distance(mi)         0.022896
Wind_Speed(mph)      0.018658
Severity             0.016970
Precipitation(in)    0.013975
Start_Lng            0.005919
End_Lng              0.005913
Name: Temperature(F), dtype: float64
```

```
numeric_data = accidents_data.select_dtypes(include=[np.number])
correlation_matrix = numeric_data.corr()

# Drop the self-correlation for 'Visibility(mi)' to analyze its correlation with other variables
Visibility_correlation = correlation_matrix['Visibility(mi)'].drop('Visibility(mi)')
Visibility_correlation_sorted = Visibility_correlation.abs().sort_values(ascending=False)

print(Visibility_correlation_sorted)
```

```
Humidity(%)          0.387189
Wind_Chill(F)        0.257317
Temperature(F)       0.250565
Precipitation(in)    0.241752
End_Lat              0.105322
Start_Lat            0.105309
Severity             0.031498
Pressure(in)         0.023342
Distance(mi)         0.011877
Start_Lng            0.004119
End_Lng              0.004117
Wind_Speed(mph)      0.000096
Name: Visibility(mi), dtype: float64
```

```
environmental_variables = ['Temperature(F)', 'Visibility(mi)', 'Humidity(%)', 'Pressure(in)', 'Precipitation(in)']
data_selected = accidents_data[['Severity'] + environmental_variables]

# Calculate the correlation coefficients
correlation_matrix = data_selected.corr()

# Extract correlation values with respect to 'Severity'
severity_correlation = correlation_matrix['Severity'].drop('Severity')

# Sort the absolute correlation values in descending order
severity_correlation_sorted = severity_correlation.abs().sort_values(ascending=False)

# Visualize the correlation matrix as a heatmap
plt.figure(figsize=(10, 6))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt='.2f', center=0)
plt.title('Correlation Matrix')
plt.show()
```



I generated a **heatmap** to provide a clear visual representation of the correlation levels between variables, utilizing **color** coding to facilitate **easier interpretation.**
This approach allows for an immediate understanding of how different variables relate to one another, highlighting significant correlations that may warrant further investigation.

🔑 A Shift in Perspective is Needed

> During the initial stages of exploratory data analysis, it was hypothesized that variables such as Temperature (F), Visibility (mi), Humidity (%), Pressure (in), and Precipitation (in) would have a significant correlation with the severity of accidents and thus aid in predictive modeling. **However, upon conducting a correlation analysis in Part 3 : Multivariate Analysis, it became evident that the correlations were minimal.**
>
> This outcome can be interpreted in several ways. It suggests that **environmental factors have a limited impact on the severity of accidents**, indicating that ***other underlying factors may play a more critical role.*** For instance, one might consider variables such as **driver concentration, traffic infrastructure** (including road conditions, GPS systems, traffic signals, etc.), the **design of highways, and the complexity of intersections** (e.g., four-way and five-way junctions) as potential contributors to accident severity.

🔗 *The following content is a continuation of the previous idea ⋯.*

```
# Select relevant variables
variables = ['Severity', 'Temperature(F)', 'Visibility(mi)', 'Humidity(%)', 'Pressure(in)', 'Precipitation(in)']
data_selected = accidents_data[variables].dropna()  # Remove rows with missing values

# Define independent variables (X) and dependent variable (y)
X = data_selected[['Temperature(F)', 'Visibility(mi)', 'Humidity(%)', 'Pressure(in)', 'Precipitation(in)']]
y = data_selected['Severity']

# Add a constant (intercept) term to the model
X = sm.add_constant(X)

# Create and fit the regression model
model = sm.OLS(y, X).fit()
```

```
# Display the results of the regression
print(model.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:               Severity   R-squared:                       0.003
Model:                            OLS   Adj. R-squared:                  0.002
Method:                 Least Squares   F-statistic:                     5.900
Date:                Sat, 21 Sep 2024   Prob (F-statistic):           1.88e-05
Time:                        06:28:03   Log-Likelihood:                -6034.9
No. Observations:               10000   AIC:                         1.208e+04
Df Residuals:                    9994   BIC:                         1.212e+04
Df Model:                           5
Covariance Type:            nonrobust
==========================================================================================
                             coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------------------
const                      2.6087      0.116     22.422      0.000       2.381       2.837
Temperature(F)         -4.895e-05      0.000     -0.193      0.847      -0.001       0.000
Visibility(mi)             -0.0035      0.002     -1.834      0.067      -0.007       0.000
Humidity(%)                 0.0002      0.000      0.906      0.365      -0.000       0.001
Pressure(in)               -0.0151      0.004     -3.696      0.000      -0.023      -0.007
Precipitation(in)           0.2363      0.108      2.181      0.029       0.024       0.449
==============================================================================
Omnibus:                     4921.470   Durbin-Watson:                   2.020
Prob(Omnibus):                  0.000   Jarque-Bera (JB):            27831.454
Skew:                           2.375   Prob(JB):                         0.00
Kurtosis:                       9.651   Cond. No.                     2.53e+03
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 2.53e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
```

- **R-squared (Coefficient of Determination)**
  The R-squared value is 0.003, indicating that the model explains
  almost none of the variance in accident severity.
  This suggests that the environmental factors used
  have **little influence on accident severity.**

- **Temperature (F)**
  Coefficient = -0.000049, p-value = 0.847.
  The effect of temperature on accident severity is statistically **insignificant.**

- **Visibility (mi)**
  Coefficient = -0.0035, p-value = 0.067. The impact of visibility may be somewhat significant,
  but the p-value is above 0.05, making **it hard to draw strong conclusions.**

- **Humidity (%)**
  Coefficient = 0.0002, p-value = 0.365.
  Humidity has **no statistically significant** effect on accident severity.

- **Visibility (mi)** and **Precipitation (in)** *do have an impact, but their influence is not substantial.*

> 💬 It seems that using environmental variables to predict accident severity may **not be an effective approach for exploration.**

[Stage Between 3.1 and 3.2]
Before attempting 3.2 Clustering, I tried to build a model to predict the 'likelihood' of accidents.
I managed to create the model and reach the stage of predicting on sample data,
but the accuracy and precision were around 50%, making it feel like random guessing.
To improve the model's performance, I tried tuning certain hyperparameters and adding more features,

but predicting the 'likelihood' and severity of accidents remained challenging.
(This may be due to a lack of strong correlations.)
Therefore, I plan to change direction and focus on clustering regions instead.

💡 Analytic Method

3.2 **Clustering**

Clustering Regions with Similar Accident Patterns :
Developing Prevention Strategies 🚦

```
# 1. Analyze accident frequency by region
accident_counts = accidents_data['City'].value_counts()
accident_counts_df = accident_counts.reset_index()
accident_counts_df.columns = ['City', 'Accident_Count']

# 2. Compare accident frequency and severity
accidents_data['Severity'] = pd.to_numeric(accidents_data['Severity'], errors='coerce')

# 3. Calculate accident frequency and average severity
severity_counts = accidents_data.groupby('City')['Severity'].mean().reset_index()
severity_counts.columns = ['City', 'Average_Severity']
comparison_df = pd.merge(accident_counts_df, severity_counts, on='City')
comparison_df['Accident_Count'] = pd.to_numeric(comparison_df['Accident_Count'], errors='coerce')
comparison_df['Average_Severity'] = pd.to_numeric(comparison_df['Average_Severity'], errors='coerce')

# 4. Clustering
numeric_features = ['Accident_Count', 'Average_Severity']

data_selected = comparison_df[numeric_features]

kmeans = KMeans(n_clusters=3, random_state=42)
comparison_df['Cluster'] = kmeans.fit_predict(data_selected)

# Check cluster characteristics: calculate mean values for each cluster
cluster_summary = comparison_df.groupby('Cluster')[numeric_features].mean()
print(cluster_summary)
```

```
         Accident_Count  Average_Severity
Cluster
0              2.577112          2.178600
1            180.000000          2.094460
2             40.278689          2.147687
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:1416: FutureWarning: The default value of `n_i
  super()._check_params_vs_input(X, default_n_init=10)
```

Analysis of Accident Frequency and Severity by Cluster

Cluster Interpretations

1. **Cluster 0** :
   - **Accident Frequency** : Approximately 2.58 incidents
   - **Average Severity** : Approximately 2.18
   - **Interpretation** : This cluster exhibits low accident frequency and moderate severity. Accidents are infrequent, but when they occur, they tend to be of average severity.
2. **Cluster 1** :
   - **Accident Frequency** : 180 incidents
   - **Average Severity** : Approximately 2.09
   - **Interpretation** : This cluster shows very high accident frequency with relatively low average severity. While many accidents occur in this area, they are generally minor.

3. **Cluster 2** :
    - **Accident Frequency** : Approximately 40.28 incidents
    - **Average Severity** : Approximately 2.15
    - **Interpretation** : This cluster demonstrates medium accident frequency and average severity. Accidents occur at a moderate rate, with severity similar to that of Cluster 0.

Overall Interpretation

- Accident Patterns : Each cluster reflects distinct patterns based on differences in accident frequency and severity. This information is valuable for developing prevention and response strategies.
- Policy Approach : In high-frequency areas like Cluster 1, preventive measures may be necessary to reduce accidents, while in Cluster 0, where accidents are infrequent but of lower severity, management and response strategies may differ.

Based on this information, specific measures or actions can be considered for each cluster.

🖋️ Action Plans by Cluster

Cluster 0
(Low Frequency , Moderate Severity) :

- **Improve Response Strategies**: Although accidents are infrequent , the moderate severity necessitates strengthening effective emergency response systems for better management of incidents.

Cluster 1
(High Frequency , Low Severity) :

- **Enhance Preventive Measures**: Given the high accident frequency, there is a need for stronger traffic regulations, road improvements, and awareness campaigns to reduce accidents.
- **Manage Minor Incidents**: With a low average severity, it is essential to establish a rapid response system and improve access to medical services when accidents occur.

Cluster 2
(Medium Frequency , Moderate Severity) :

- **Balanced Approach**: With both accident frequency and severity at moderate levels, a balanced strategy that incorporates both preventive measures and effective response plans is required.

🔗 *The following content continues from the previous results ⋯.*

🌍 Identify the areas where accidents occur most frequently

Outline of the Progression

1. Aggregate the number of accidents by region.
2. Organize the data in descending order.
3. Examine which specific regions these correspond to.
4. Display the top N regions on an actual map.
5. Investigate the characteristics of the selected top N regions in detail.

```
# Create a tuple of latitude and longitude for each accident
accidents_data['Location'] = accidents_data[['Start_Lat', 'Start_Lng']].apply(tuple, axis=1)
location_counts = accidents_data['Location'].value_counts()

# Sort locations in descending order
```

```
sorted_locations = location_counts.sort_values(ascending=False)

# Get the top 10 locations
top_n = sorted_locations.head(10)
print(top_n)
```

```
Location
(26.318445, -80.116196)                    3
(26.527572870128022, -80.1724612709037)    3
(38.374718, -77.462502)                    3
(25.963024, -80.15508100000002)            2
(27.538059000000004, -82.511054)           2
(34.691803, -82.504829)                    2
(34.857304, -81.008507)                    2
(30.25445, -97.865692)                     2
(35.363312, -97.547638)                    2
(33.581978, -112.116951)                   2
Name: count, dtype: int64
```

Google Map 🗺️

> Boca Raton, Florida, USA (26.318445, -80.116196)
> West Palm Beach, Florida, USA (26.527572870128022, -80.1724612709037)
> Stafford, Virginia, USA (38.374718, -77.462502)
> Miami, Florida, USA. (25.963024, -80.15508100000002)
> Sarasota, Florida, USA (27.538059000000004, -82.511054)
> Anderson, South Carolina, USA (34.691803, -82.504829)
> Spartanburg, South Carolina, USA(34.857304, -81.008507)
> Austin, Texas, USA (30.25445, -97.865692)
> Norman, Oklahoma, USA (35.363312, -97.547638)
> Phoenix, Arizona, USA (33.581978, -112.116951)

Describe Each Region's Features

- **Florida Region**
  - Climate
    Due to the **warm climate**, there are **frequent outdoor activities**,
    leading to increased **pedestrian traffic**.
  - Traffic Volume
    The **mix of commercial and residential areas** results in
    frequent collisions between vehicles and pedestrians.
  - Population Structure
    The large number of **tourists** causes temporary population surges,
    leading to traffic congestion.
- **Virginia and South Carolina**
  - Suburban Areas
    The **blend of residential and commercial** facilities increases traffic volume.
  - Road Structure
    The **combination of multiple intersections and large roads** poses a higher risk of accidents.
- **Texas and Arizona**
  - Road Design
    Wide roads and **numerous intersections** increase the likelihood of collisions
    between high-speed vehicles and pedestrians.
  - Infrastructure
    The **lack of traffic safety facilities** raises the risk of accidents.

Common Factors of the Top 10 High Accident Areas

[**Urban Density**]
Most of the top regions are urban or suburban areas
with **high population density and heavy traffic**.

[**Proximity to Commercial Areas**]
Many of these areas are **close to commercial facilities or major roads**,
making them prone to **traffic congestion**.

[**Intersections and Road Structure**]
The **high number of intersections and complex road layouts** increase the likelihood of accidents,
especially collisions between pedestrians and vehicles.

[**Driving and Pedestrian Patterns**]
The combination of heavy traffic and varied driving and pedestrian patterns
contributes to the heightened risk of accidents.

[**Lack of Traffic Safety Infrastructure**]
Some areas **lack proper traffic safety facilities**, such as traffic lights
and crosswalks, or have outdated infrastructure.

Then ,

💡 This follow-up question arises . . .

" Are there areas in South Korea with similar characteristics? "

In other words , does the area where accidents occur in South Korea also have theses characteristics?

🔗 *To solve the previous curiosity ⋯.*

The Korea Road Traffic Authority downloaded the file
' ***Road Traffic Authority_Traffic Accident Prevention Area_20240909.csv*** ' and
tried to analyze the characteristics of the accident prone area in South Korea and the top area.
Korea Road Traffic Authority

```
file_path = '/content/drive/MyDrive/빅데이터종합설계/Assign1 [ EDA ]/도로교통공단_교통사고다발지역_20240909.csv'
accidents_data_korea = pd.read_csv(file_path)

accidents_data_korea = accidents_data_korea[['위도','경도','사고건수','사고지역위치명']]
print(accidents_data_korea.isnull().sum())
```

```
위도          0
경도          0
사고건수        0
사고지역위치명     0
dtype: int64
```

```
sorted_accidents_data = accidents_data_korea.sort_values(by='사고건수', ascending=False)
print(sorted_accidents_data.head(10))
```

```
             위도          경도  사고건수                                사고지역위치명
3021  37.484117  126.929482    27            서울특별시 관악구 신림동(신림역 부근)
1593  37.265498  127.001433    26  경기도 수원시 팔달구 매산로1가(수원역남단교차로 부근)
3008  37.519199  126.908637    24      서울특별시 영등포구 영등포동3가(선한치과 부근)
2980  37.570243  126.987575    23          서울특별시 종로구 낙원동(종로2가 부근)
3255  37.266356  127.001493    23     경기도 수원시 팔달구 교동(수원역남단교차로 부근)
1281  35.185305  129.082146    23         부산광역시 연제구 연산동(연산교차로 부근)
6343  37.480615  126.891547    22       서울특별시 구로구 가리봉동(만민중앙교회 부근)
6489  37.266833  127.001830    20  경기도 수원시 팔달구 매산로1가(수원역동측단일로 부근)
3087  35.186417  129.081709    19         부산광역시 연제구 연산동(연산교차로 부근)
3032  37.480764  126.891652    19       서울특별시 구로구 가리봉동(만민중앙교회 부근)
```

Google Map 🗺️

> 서울특별시 관악구 신림동(신림역 부근) (37.484117 , 126.929482)
> 경기도 수원시 팔달구 매산로1가(수원역남단교차로 부근) (37.265498 , 127.001433)
> 서울특별시 영등포구 영등포동3가(선한치과 부근) (37.519199 , 126.908637)
> 서울특별시 종로구 낙원동(종로2가 부근). (37.570243 , 126.987575)
> 경기도 수원시 팔달구 교동(수원역남단교차로 부근) (37.266356 , 127.001493)
> 부산광역시 연제구 연산동(연산교차로 부근) (35.185305 , 129.082146)
> 서울특별시 구로구 가리봉동(만민중앙교회 부근)(37.480615 , 126.891547)
> 경기도 수원시 팔달구 매산로1가(수원역동측단일로 부근) (37.266833 , 127.001830)
> 부산광역시 연제구 연산동(연산교차로 부근) (35.186417 , 129.081709)
> 서울특별시 구로구 가리봉동(만민중앙교회 부근) (37.480764 , 126.891652)

Key Characteristics of Korea's Accident - Following Area

The top accident-prone areas in Korea share several key characteristics.
High traffic density is a common factor across most regions, particularly in areas near major transportation hubs such as **Seoul's Sinlim Station** and **Suwon Station**, where a mix of commercial and residential zones increases the likelihood of pedestrian and vehicle collisions. The **congestion at intersections** is another critical issue, notably in **Yeongdeungpo** and **Guro**, where high pedestrian traffic interacts with busy roads. Additionally, **tourist hotspots** like **Jongno** see frequent accidents due to the influx of both locals and visitors.
In all these areas, **the lack of adequate traffic infrastructure** further exacerbates the accident risk, as overcrowding of both pedestrians and vehicles leads to frequent incidents, especially in places like **Busan's Yeonsan Intersection**.

Common Characteristics of the Top 10 Accidents in the U.S. and South Korea

The top accident-prone areas in South Korea and the U.S. share several key characteristics that elevate the risk of accidents. In both regions, **high traffic density** is prevalent, particularly in urban areas near major transportation hubs like **Seoul's Sinlim Station** and **Suwon Station**, where the mix of commercial and residential zones increases the likelihood of **collisions between vehicles and pedestrians**. **Congestion at intersections** is another critical issue, especially in areas such as **Yeongdeungpo** and **Guro** in South Korea, and similarly in various U.S. urban centers, where busy roads intersect with heavy pedestrian traffic. Additionally, both regions experience **increased accident risk** due to the **lack of adequate traffic safety infrastructure**, including insufficient traffic lights and crosswalks, which exacerbates the dangers posed by high pedestrian volumes and complex road layouts. The combination of **diverse driving and pedestrian patterns** further contributes to the heightened risk, making these accident-prone areas particularly vulnerable.

- Key words
  - High traffic density
  - Congestion at intersections
  - Lack of adequate traffic safety infrastructure
  - High pedestrian volumes
  - Complex road layouts.

💡 Conclusion [Suggestion]

> This exploratory data analysis (EDA) has provided a profound understanding of the patterns of traffic accidents and the various factors influencing their occurrence. It has been confirmed that traffic accidents are not merely the result of driver negligence or unfortunate events; rather, they are the outcome of a complex interplay of various environmental and temporal factors. Based on this understanding, the following insights and recommendations have been formulated.

1. **Relationship between Accident Severity and Environmental Factors**
   The analysis revealed that the severity of accidents is only weakly correlated with traditional environmental factors such as temperature and weather conditions. Notably, accidents occur consistently throughout all seasons, indicating that other factors—such as driver attentiveness, the design and maintenance of traffic infrastructure, and roadway complexity—play a more critical role. Therefore, it is suggested that when formulating traffic accident prevention policies, a greater emphasis should be placed on factors beyond mere environmental conditions. For instance, enhancing educational programs and campaigns aimed at improving driver awareness and strengthening policies for the improvement of traffic infrastructure are essential.

2. **Analysis of Time-based and Day-of-the-Week Accident Patterns**
   The analysis indicated that the frequency and severity of accidents vary by time of day and day of the week. Specifically, accidents are more frequent during daylight hours, particularly during peak traffic periods such as 7-8 AM and 2-5 PM, where the incidence of accidents doubles. Utilizing this data, it is imperative to implement traffic safety campaigns tailored to specific time frames.

3. **Regional Accident Pattern Analysis**
   Cluster analysis has shown that the frequency and severity of accidents differ by region. Areas with high accident rates necessitate more robust preventive measures, while regions with lower frequency but higher severity require enhanced emergency response systems. Tailored safety measures that consider regional characteristics can be proposed accordingly.

4. **Policy Recommendations for Accident Prevention**
   Based on the lessons learned from the data analysis, recommendations for traffic safety policies, educational programs, and awareness campaigns are presented. For example, road improvement initiatives in high-accident areas, coupled with targeted educational programs for drivers, can contribute significantly to accident reduction. Special attention should be given to safety education focused on peak accident hours.

5. **Future Research Directions**
   Building on the findings of this analysis, the incorporation of additional variables in multivariate analyses or the application of machine learning techniques for predictive modeling is necessary. This approach will facilitate the development of more sophisticated and practical traffic accident prevention strategies. Furthermore, continuous data collection and analysis will be crucial for monitoring changes in accident patterns over time and adapting policies accordingly.

6. **Conclusion**
   It is essential to recognize that traffic accidents are not mere coincidences but rather complex phenomena involving multiple interrelated factors. Through the insights gained from this EDA, it is anticipated that effective strategies and policies can be developed to contribute to traffic accident prevention efforts.

💡 Expected Effect

[**Reduction in Accident Frequency**]
By incorporating a multifaceted approach that considers not only environmental factors but also critical elements such as driver attention, the efficiency of traffic infrastructure, and road complexity, a substantial decrease in traffic accident rates is expected. Targeting specific times and locations with historically high accident rates will facilitate more effective preventive measures, thereby minimizing the occurrence of traffic incidents.

[**Mitigation of Accident Severity**]
In addition to reducing the frequency of accidents, the severity of incidents is also expected to decline. Enhanced awareness and improved traffic infrastructure can lead to safer driving behaviors, ultimately reducing the potential for severe injuries and fatalities during accidents.

[**Increased Traffic Safety Awareness**]
The activation of traffic safety campaigns and educational programs is likely to elevate awareness levels among both drivers and pedestrians. Such initiatives play a crucial role in fostering a culture of safety, which is fundamental in mitigating traffic accidents.

[**Economic Savings through Cost Reduction**]
A decrease in the frequency and severity of traffic accidents will result in significant social cost savings. These savings encompass medical expenses, insurance costs, and economic losses associated with traffic congestion. Collectively, this will contribute to enhanced economic stability within communities.

[**Establishment of a Sustainable Traffic System**]
Ultimately, the implementation of the strategies derived from this research is expected to contribute to the development of a more sustainable and safe traffic system. This will not only facilitate the prevention of traffic accidents but will also serve as a foundation for the advancement and safety of mobility within the community over the long term.

💡 Future Task [Idea]

> Based on the insights derived from this study,
> the following tasks are proposed for future research and policy implementation :

1. **Utilization of Multivariate Analysis and Machine Learning Techniques**
   The current research focused on specific variables, but **there is a need for multivariate analysis that includes various factors influencing traffic accident occurrence**. Particularly, analyzing **traffic volume, road conditions, and social factors in an integrated manner** will help identify the complex causes of accidents. Additionally, the development of **predictive models using machine learning techniques** can present a new paradigm for traffic accident prevention. Such models can effectively analyze non-linear relationships and complex patterns, contributing to the ability to **anticipate accident probabilities under specific conditions and implement proactive prevention measures**.

2. **National and Regional Comparative Studies and Research on Social Perception Changes**
   Traffic accident occurrence patterns may differ across regions, necessitating research that compares data from various areas. This approach will provide essential foundational data for developing **customized traffic safety measures** that reflect the characteristics of each region. Furthermore, studying **changes in social perception regarding traffic safety and subsequent behavioral changes** will enable the assessment of the effectiveness of educational programs and campaigns, as well as the identification of future improvement strategies. In particular, it is crucial to emphasize the need for **tailored approaches based on demographic groups**.

3. **Policy Effectiveness Analysis**
   There is a need for research that analyzes and evaluates the effectiveness of policies and programs aimed at preventing traffic accidents. Such studies can help **verify the effectiveness of each strategy** and establish clear criteria for future policy improvements. This will contribute to the establishment of **effective accident prevention strategies** and enhance the effectiveness of traffic safety policies.

In conclusion, the results derived from this study provide **various directions for traffic accident prevention**. Future research should focus on deepening and specifying these directions, contributing to the creation of **a safer traffic environment**.