Project Name:Traffic Accident Analysis: Identifying Patterns and Insights to Enhance Road Safety

By Mehul Chafekar

Project Introduction

- Traffic accidents are a critical public safety concern worldwide, leading to loss of life, property damage, and economic burden.
- With the increasing availability of data, analyzing accident patterns can provide actionable insights to improve road safety.
- This project explores a comprehensive dataset of traffic accidents to identify key trends and factors contributing to accidents, including weather conditions, time of day, location, and accident severity.
- By leveraging data visualization and analysis techniques, we aim to extract meaningful patterns to aid decision-making and preventive measures.

Project Summary

This project involves an extensive analysis of traffic accident data. Key tasks include:

1. Cleaning and preprocessing the dataset to handle missing and inconsistent values.
2. Performing exploratory data analysis (EDA) to identify significant patterns related to accident severity, time of occurrence, and geographic distribution.
3. Visualizing insights using charts like pie charts, bar plots, histograms, and line plots to effectively communicate findings.
4. Highlighting trends in accidents by state, time of day, and weather conditions.
5. Providing actionable recommendations to stakeholders for enhancing road safety and reducing accidents.

Business Objective

The primary business objective is to provide insights into traffic accident patterns to help policymakers, urban planners, and traffic authorities:

1. Identify high-risk areas (accident hotspots) for targeted interventions.
2. Understand the impact of weather and time of day on accidents to optimize traffic management strategies.
3. Enhance public awareness campaigns by focusing on key contributing factors like speeding, low visibility, or road infrastructure.
4. Reduce accidents and improve safety outcomes, leading to lower economic losses and better public well-being.

Importing Libraries

In [66]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

Loading the Dataset

In [67]:
```
df= pd.read_csv('US_Accidents_Dec21_updated.csv')
```

In [68]:
```
df
```

Out[68]:

	ID	Severity	Start_Time	End_Time	Start_Lat	Start_Lng	End_Lat	End
0	A-1	3	2016-02-08 00:37:08	2016-02-08 06:37:08	40.108910	-83.092860	40.112060	-83.03
1	A-2	2	2016-02-08 05:56:20	2016-02-08 11:56:20	39.865420	-84.062800	39.865010	-84.04
2	A-3	2	2016-02-08 06:15:39	2016-02-08 12:15:39	39.102660	-84.524680	39.102090	-84.52
3	A-4	2	2016-02-08 06:51:45	2016-02-08 12:51:45	41.062130	-81.537840	41.062170	-81.53
4	A-5	3	2016-02-08 07:53:43	2016-02-08 13:53:43	39.172393	-84.492792	39.170476	-84.50
...	...	...	...	...	...	...	...	
2845337	A-2845338	2	2019-08-23 18:03:25	2019-08-23 18:32:01	34.002480	-117.379360	33.998880	-117.37
2845338	A-2845339	2	2019-08-23 19:11:30	2019-08-23 19:38:23	32.766960	-117.148060	32.765550	-117.15
2845339	A-2845340	2	2019-08-23 19:00:21	2019-08-23 19:28:49	33.775450	-117.847790	33.777400	-117.85
2845340	A-2845341	2	2019-08-23 19:00:21	2019-08-23 19:29:42	33.992460	-118.403020	33.983110	-118.39
2845341	A-2845342	2	2019-08-23 18:52:06	2019-08-23 19:21:31	34.133930	-117.230920	34.137360	-117.23

2845342 rows × 47 columns

Understanding the Data

In [69]:
```
df.head(10)
```

Out[69]:

	ID	Severity	Start_Time	End_Time	Start_Lat	Start_Lng	End_Lat	End_Lng	Distanc
0	A-1	3	2016-02-08 00:37:08	2016-02-08 06:37:08	40.108910	-83.092860	40.112060	-83.031870	[illegible]
1	A-2	2	2016-02-08 05:56:20	2016-02-08 11:56:20	39.865420	-84.062800	39.865010	-84.048730	[illegible]
2	A-3	2	2016-02-08 06:15:39	2016-02-08 12:15:39	39.102660	-84.524680	39.102090	-84.523960	[illegible]
3	A-4	2	2016-02-08 06:51:45	2016-02-08 12:51:45	41.062130	-81.537840	41.062170	-81.535470	[illegible]
4	A-5	3	2016-02-08 07:53:43	2016-02-08 13:53:43	39.172393	-84.492792	39.170476	-84.501798	[illegible]
5	A-6	2	2016-02-08 08:16:57	2016-02-08 14:16:57	39.063240	-84.032430	39.067310	-84.058510	[illegible]
6	A-7	2	2016-02-08 08:15:41	2016-02-08 14:15:41	39.775650	-84.186030	39.772750	-84.188050	[illegible]
7	A-8	2	2016-02-08 11:51:46	2016-02-08 17:51:46	41.375310	-81.820170	41.367860	-81.821740	[illegible]
8	A-9	2	2016-02-08 14:19:57	2016-02-08 20:19:57	40.702247	-84.075887	40.699110	-84.084293	[illegible]
9	A-10	2	2016-02-08 15:16:43	2016-02-08 21:16:43	40.109310	-82.968490	40.110780	-82.984000	[illegible]

10 rows × 47 columns

In [70]:
```
df.tail()
```

Out[70]:

	ID	Severity	Start_Time	End_Time	Start_Lat	Start_Lng	End_Lat	End_Lng
2845337	A-2845338	2	2019-08-23 18:03:25	2019-08-23 18:32:01	34.00248	-117.37936	33.99888	-117.3709
2845338	A-2845339	2	2019-08-23 19:11:30	2019-08-23 19:38:23	32.76696	-117.14806	32.76555	-117.1536
2845339	A-2845340	2	2019-08-23 19:00:21	2019-08-23 19:28:49	33.77545	-117.84779	33.77740	-117.8572
2845340	A-2845341	2	2019-08-23 19:00:21	2019-08-23 19:29:42	33.99246	-118.40302	33.98311	-118.3956
2845341	A-2845342	2	2019-08-23 18:52:06	2019-08-23 19:21:31	34.13393	-117.23092	34.13736	-117.2393

5 rows × 47 columns

checking columns in data

In [71]:
```
df.columns
```

Out[71]:
```
Index(['ID', 'Severity', 'Start_Time', 'End_Time', 'Start_Lat', 'Start_Ln
g',
       'End_Lat', 'End_Lng', 'Distance(mi)', 'Description', 'Number', 'Str
eet',
       'Side', 'City', 'County', 'State', 'Zipcode', 'Country', 'Timezon
e',
       'Airport_Code', 'Weather_Timestamp', 'Temperature(F)', 'Wind_Chill
(F)',
       'Humidity(%)', 'Pressure(in)', 'Visibility(mi)', 'Wind_Direction',
       'Wind_Speed(mph)', 'Precipitation(in)', 'Weather_Condition', 'Ameni
ty',
       'Bump', 'Crossing', 'Give_Way', 'Junction', 'No_Exit', 'Railway',
       'Roundabout', 'Station', 'Stop', 'Traffic_Calming', 'Traffic_Signa
l',
       'Turning_Loop', 'Sunrise_Sunset', 'Civil_Twilight', 'Nautical_Twili
ght',
       'Astronomical_Twilight'],
      dtype='object')
```

Shape of the dataframe

In [72]:
```
df.shape
```

Out[72]:
```
(2845342, 47)
```

Info of the dataframe

In [73]:
```
df.info()
```

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

In [74]:
```
df.dtypes.value_counts()
```

Out[74]:
```
object     20
float64    13
bool       13
int64       1
dtype: int64
```

In [75]:
```
df.describe()
```

Out[75]:

	Severity	Start_Lat	Start_Lng	End_Lat	End_Lng	Distance(mi)
count	2.845342e+06	2.845342e+06	2.845342e+06	2.845342e+06	2.845342e+06	2.845342e+06
mean	2.137572e+00	3.624520e+01	-9.711463e+01	3.624532e+01	-9.711439e+01	7.026779e-01
std	4.787216e-01	5.363797e+00	1.831782e+01	5.363873e+00	1.831763e+01	1.560361e+00
min	1.000000e+00	2.456603e+01	-1.245481e+02	2.456601e+01	-1.245457e+02	0.000000e+00
25%	2.000000e+00	3.344517e+01	-1.180331e+02	3.344628e+01	-1.180333e+02	5.200000e-02
50%	2.000000e+00	3.609861e+01	-9.241808e+01	3.609799e+01	-9.241772e+01	2.440000e-01
75%	2.000000e+00	4.016024e+01	-8.037243e+01	4.016105e+01	-8.037338e+01	7.640000e-01
max	4.000000e+00	4.900058e+01	-6.711317e+01	4.907500e+01	-6.710924e+01	1.551860e+02

In [76]:
```
df.State.unique
```

Out[76]:
```
<bound method Series.unique of 0          OH
1          OH
2          OH
3          OH
4          OH
           ..
2845337    CA
2845338    CA
2845339    CA
2845340    CA
2845341    CA
Name: State, Length: 2845342, dtype: object>
```

In [82]:
```
df_new=df[df['State']=='CA']
```

In [78]:
```
df_new['IDD'] = df_new['ID'].astype('str').str.extractall('(\d+)').unstack(
```

In [79]:
```
df_new
```

Out[79]:

	ID	Severity	Start_Time	End_Time	Start_Lat	Start_Lng	End_Lat	End
988	A-989	3	2016-03-22 18:53:11	2016-03-23 00:53:11	38.825840	-120.029214	38.827194	-120.0:
989	A-990	2	2016-03-22 19:00:49	2016-03-23 01:00:49	37.358209	-121.840017	37.361596	-121.84
990	A-991	3	2016-03-22 20:07:32	2016-03-23 02:07:32	37.881943	-122.307987	37.885882	-122.3(
991	A-992	2	2016-03-22 21:40:18	2016-03-23 03:40:18	37.881038	-122.307788	37.883458	-122.3(
992	A-993	2	2016-03-22 21:36:42	2016-03-23 03:36:42	38.518811	-121.101664	38.518811	-121.1(
...	...	...	...	...	...	...	...	
2845337	A-2845338	2	2019-08-23 18:03:25	2019-08-23 18:32:01	34.002480	-117.379360	33.998880	-117.3
2845338	A-2845339	2	2019-08-23 19:11:30	2019-08-23 19:38:23	32.766960	-117.148060	32.765550	-117.1
2845339	A-2845340	2	2019-08-23 19:00:21	2019-08-23 19:28:49	33.775450	-117.847790	33.777400	-117.8
2845340	A-2845341	2	2019-08-23 19:00:21	2019-08-23 19:29:42	33.992460	-118.403020	33.983110	-118.3
2845341	A-2845342	2	2019-08-23 18:52:06	2019-08-23 19:21:31	34.133930	-117.230920	34.137360	-117.2

795868 rows × 48 columns

In [80]:
```
df_new.head(10)
```

Out[80]:

	ID	Severity	Start_Time	End_Time	Start_Lat	Start_Lng	End_Lat	End_Lng	Di
988	A-989	3	2016-03-22 18:53:11	2016-03-23 00:53:11	38.825840	-120.029214	38.827194	-120.030632	
989	A-990	2	2016-03-22 19:00:49	2016-03-23 01:00:49	37.358209	-121.840017	37.361596	-121.842044	
990	A-991	3	2016-03-22 20:07:32	2016-03-23 02:07:32	37.881943	-122.307987	37.885882	-122.308878	
991	A-992	2	2016-03-22 21:40:18	2016-03-23 03:40:18	37.881038	-122.307788	37.883458	-122.308366	
992	A-993	2	2016-03-22 21:36:42	2016-03-23 03:36:42	38.518811	-121.101664	38.518811	-121.101664	
993	A-994	2	2016-03-22 21:36:42	2016-03-23 03:36:42	38.518811	-121.101664	38.518811	-121.101664	
994	A-995	2	2016-03-23 03:48:55	2016-03-23 09:48:55	36.990300	-119.711460	36.990460	-119.711380	
995	A-996	2	2016-03-23 05:55:55	2016-03-23 11:55:55	37.425920	-122.098790	37.430420	-122.103520	
996	A-997	2	2016-03-23 06:39:54	2016-03-23 12:39:54	37.757450	-122.211310	37.750850	-122.205490	
997	A-998	2	2016-03-23 06:45:09	2016-03-23 12:45:09	37.316480	-121.967460	37.318100	-121.978100	

10 rows × 48 columns

In [81]:
```
df_new.shape
```
Out[81]:
```
(795868, 48)
```

```
In [83]: df_new.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 795868 entries, 988 to 2845341
Data columns (total 47 columns):
 #   Column                 Non-Null Count   Dtype  
---  ------                 --------------   -----  
 0   ID                     795868 non-null  object 
 1   Severity               795868 non-null  int64  
 2   Start_Time             795868 non-null  object 
 3   End_Time               795868 non-null  object 
 4   Start_Lat              795868 non-null  float64
 5   Start_Lng              795868 non-null  float64
 6   End_Lat                795868 non-null  float64
 7   End_Lng                795868 non-null  float64
 8   Distance(mi)           795868 non-null  float64
 9   Description            795868 non-null  object 
 10  Number                 256963 non-null  float64
 11  Street                 795867 non-null  object 
 12  Side                   795868 non-null  object 
 13  City                   795861 non-null  object 
 14  County                 795868 non-null  object 
 15  State                  795868 non-null  object 
 16  Zipcode                795469 non-null  object 
 17  Country                795868 non-null  object 
 18  Timezone               795469 non-null  object 
 19  Airport_Code           794898 non-null  object 
 20  Weather_Timestamp      780498 non-null  object 
 21  Temperature(F)         773334 non-null  float64
 22  Wind_Chill(F)          664721 non-null  float64
 23  Humidity(%)            772282 non-null  float64
 24  Pressure(in)           778334 non-null  float64
 25  Visibility(mi)         777381 non-null  float64
 26  Wind_Direction         773393 non-null  object 
 27  Wind_Speed(mph)        750221 non-null  float64
 28  Precipitation(in)      625445 non-null  float64
 29  Weather_Condition      776759 non-null  object 
 30  Amenity                795868 non-null  bool   
 31  Bump                   795868 non-null  bool   
 32  Crossing               795868 non-null  bool   
 33  Give_Way               795868 non-null  bool   
 34  Junction               795868 non-null  bool   
 35  No_Exit                795868 non-null  bool   
 36  Railway                795868 non-null  bool   
 37  Roundabout             795868 non-null  bool   
 38  Station                795868 non-null  bool   
 39  Stop                   795868 non-null  bool   
 40  Traffic_Calming        795868 non-null  bool   
 41  Traffic_Signal         795868 non-null  bool   
 42  Turning_Loop           795868 non-null  bool   
 43  Sunrise_Sunset         795761 non-null  object 
 44  Civil_Twilight         795761 non-null  object 
 45  Nautical_Twilight      795761 non-null  object 
 46  Astronomical_Twilight  795761 non-null  object 
dtypes: bool(13), float64(13), int64(1), object(20)
memory usage: 222.4+ MB
```

```
In [84]: df_new.columns
```

```
Out[84]: Index(['ID', 'Severity', 'Start_Time', 'End_Time', 'Start_Lat', 'Start_Ln
g',
       'End_Lat', 'End_Lng', 'Distance(mi)', 'Description', 'Number', 'Str
eet',
       'Side', 'City', 'County', 'State', 'Zipcode', 'Country', 'Timezon
e',
       'Airport_Code', 'Weather_Timestamp', 'Temperature(F)', 'Wind_Chill
(F)',
       'Humidity(%)', 'Pressure(in)', 'Visibility(mi)', 'Wind_Direction',
       'Wind_Speed(mph)', 'Precipitation(in)', 'Weather_Condition', 'Ameni
ty',
       'Bump', 'Crossing', 'Give_Way', 'Junction', 'No_Exit', 'Railway',
       'Roundabout', 'Station', 'Stop', 'Traffic_Calming', 'Traffic_Signa
l',
       'Turning_Loop', 'Sunrise_Sunset', 'Civil_Twilight', 'Nautical_Twili
ght',
       'Astronomical_Twilight'],
      dtype='object')
```

Number of duplicated rows

```
In [85]: df_new.duplicated().sum()
```

```
Out[85]: 0
```

find the Number of missing values in each column

In [86]:
```
df_new.isnull().sum()
```

Out[86]:
```
ID                            0
Severity                      0
Start_Time                    0
End_Time                      0
Start_Lat                     0
Start_Lng                     0
End_Lat                       0
End_Lng                       0
Distance(mi)                  0
Description                   0
Number                   538905
Street                        1
Side                          0
City                          7
County                        0
State                         0
Zipcode                     399
Country                       0
Timezone                    399
Airport_Code                970
Weather_Timestamp         15370
Temperature(F)            22534
Wind_Chill(F)            131147
Humidity(%)               23586
Pressure(in)              17534
Visibility(mi)            18487
Wind_Direction            22475
Wind_Speed(mph)           45647
Precipitation(in)        170423
Weather_Condition         19109
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              107
Civil_Twilight              107
Nautical_Twilight           107
Astronomical_Twilight       107
dtype: int64
```

Check the percentage of missing values for each column

In [87]:
```
missing_percentage = (df_new.isnull().sum() / len(df_new)) * 100
print(missing_percentage)
```

```
ID                        0.000000
Severity                  0.000000
Start_Time                0.000000
End_Time                  0.000000
Start_Lat                 0.000000
Start_Lng                 0.000000
End_Lat                   0.000000
End_Lng                   0.000000
Distance(mi)              0.000000
Description               0.000000
Number                   67.712862
Street                    0.000126
Side                      0.000000
City                      0.000880
County                    0.000000
State                     0.000000
Zipcode                   0.050134
Country                   0.000000
Timezone                  0.050134
Airport_Code              0.121880
Weather_Timestamp         1.931225
Temperature(F)            2.831374
Wind_Chill(F)            16.478486
Humidity(%)               2.963557
Pressure(in)              2.203129
Visibility(mi)            2.322873
Wind_Direction            2.823961
Wind_Speed(mph)           5.735499
Precipitation(in)        21.413476
Weather_Condition         2.401026
Amenity                   0.000000
Bump                      0.000000
Crossing                  0.000000
Give_Way                  0.000000
Junction                  0.000000
No_Exit                   0.000000
Railway                   0.000000
Roundabout                0.000000
Station                   0.000000
Stop                      0.000000
Traffic_Calming           0.000000
Traffic_Signal            0.000000
Turning_Loop              0.000000
Sunrise_Sunset            0.013444
Civil_Twilight            0.013444
Nautical_Twilight         0.013444
Astronomical_Twilight     0.013444
dtype: float64
```

Drop Columns

In [88]:
```
df_new.drop(columns=['Precipitation(in)', 'Wind_Chill(F)'], inplace=True)
```

In [89]:
```
df_new.drop(columns=['Number'], inplace=True)
```

Street: 1 missing value

In [90]:
```
df_new['Street'].fillna(df_new['Street'].mode()[0], inplace=True)
```

City: 7 missing values

In [91]:
```
df_new['City'].fillna(df_new['City'].mode()[0], inplace=True)
```

Sunrise_Sunset, Civil_Twilight, Nautical_Twilight, Astronomical_Twilight: 107 missing values each

In [92]:
```
twilight_cols = ['Sunrise_Sunset', 'Civil_Twilight', 'Nautical_Twilight', '
for col in twilight_cols:
    df_new[col].fillna(df_new[col].mode()[0], inplace=True)
```

Zipcode: 399 missing values, Timezone: 399 missing values, Airport_Code: 970 missing values

In [93]:
```
code_cols = ['Zipcode','Timezone','Airport_Code']
for col in code_cols:
    df_new[col].fillna(df_new[col].mode()[0], inplace=True)
```

In [94]:
```
# List of numerical columns
numerical_cols = ['Temperature(F)', 'Humidity(%)', 'Pressure(in)', 'Visibil

# Impute with median
for col in numerical_cols:
    df_new[col].fillna(df_new[col].median(), inplace=True)
```

In [95]:
```
# List of categorical columns
categorical_cols = ['Wind_Direction', 'Weather_Condition']

# Impute with mode
for col in categorical_cols:
    df_new[col].fillna(df_new[col].mode()[0], inplace=True)
```

In [96]:
```python
# Drop Wind_Speed(mph) if not relevant
df_new.drop(columns=['Wind_Speed(mph)'], inplace=True)
```

In [97]:
```python
# Convert 'Weather_Timestamp' to datetime format
df_new['Weather_Timestamp'] = pd.to_datetime(df_new['Weather_Timestamp'])
```

In [98]:
```python
# Use forward fill (fills missing values with the last valid timestamp)
df_new['Weather_Timestamp'].fillna(method='ffill', inplace=True)
```

Again check the Number of missing values in each column

In [99]:
```
df_new.isnull().sum()
```

Out[99]:
```
ID                       0
Severity                 0
Start_Time               0
End_Time                 0
Start_Lat                0
Start_Lng                0
End_Lat                  0
End_Lng                  0
Distance(mi)             0
Description              0
Street                   0
Side                     0
City                     0
County                   0
State                    0
Zipcode                  0
Country                  0
Timezone                 0
Airport_Code             0
Weather_Timestamp        0
Temperature(F)           0
Humidity(%)              0
Pressure(in)             0
Visibility(mi)           0
Wind_Direction           0
Weather_Condition        0
Amenity                  0
Bump                     0
Crossing                 0
Give_Way                 0
Junction                 0
No_Exit                  0
Railway                  0
Roundabout               0
Station                  0
Stop                     0
Traffic_Calming          0
Traffic_Signal           0
Turning_Loop             0
Sunrise_Sunset           0
Civil_Twilight           0
Nautical_Twilight        0
Astronomical_Twilight    0
dtype: int64
```

In [100]:
```
df_new['Weather_Condition'].value_counts()
```

Out[100]:
```
Fair                    420881
Cloudy                   78168
Mostly Cloudy            64487
Clear                    55664
Partly Cloudy            53889
                         ...  
Thunder and Hail             1
Thunder / Windy              1
Dust Whirls                  1
Light Snow Showers           1
Light Freezing Fog           1
Name: Weather_Condition, Length: 75, dtype: int64
```

In [101]:
```
df_new.Side.unique()
```

Out[101]:
```
array(['L', 'R', 'N'], dtype=object)
```

In [102]:
```
# Identify categorical and numerical columns
categorical_cols = df_new.select_dtypes(include=['object', 'category']).col
numerical_cols = df_new.select_dtypes(exclude=['object','category']).column

# Get the number of unique values for each categorical column
categorical_unique = df_new[categorical_cols].nunique()

# Get the number of unique values for each numerical column
numerical_unique = df_new[numerical_cols].nunique()

# Create separate dataframes for categorical and numerical columns
categorical_df = pd.DataFrame({'Column': categorical_unique.index,'Unique_V

numerical_df = pd.DataFrame({'Column': numerical_unique.index,'Unique_Value
```

```
In [103]: print("Categorical Columns Unique Values:")
          categorical_df
```

Categorical Columns Unique Values:

Out[103]:

	Column	Unique_Values
0	ID	795868
1	Start_Time	548609
2	End_Time	671908
3	Description	290526
4	Street	38047
5	Side	3
6	City	1194
7	County	58
8	State	1
9	Zipcode	73981
10	Country	1
11	Timezone	2
12	Airport_Code	141
13	Wind_Direction	24
14	Weather_Condition	75
15	Sunrise_Sunset	2
16	Civil_Twilight	2
17	Nautical_Twilight	2
18	Astronomical_Twilight	2

In [104]:
```
print("\nNumerical Columns Unique Values:")
numerical_df
```

```
Numerical Columns Unique Values:
```

Out[104]:

	Column	Unique_Values
0	Severity	4
1	Start_Lat	258409
2	Start_Lng	263363
3	End_Lat	257584
4	End_Lng	262861
5	Distance(mi)	8234
6	Weather_Timestamp	204758
7	Temperature(F)	484
8	Humidity(%)	100
9	Pressure(in)	809
10	Visibility(mi)	53
11	Amenity	2
12	Bump	2
13	Crossing	2
14	Give_Way	2
15	Junction	2
16	No_Exit	2
17	Railway	2
18	Roundabout	2
19	Station	2
20	Stop	2
21	Traffic_Calming	2
22	Traffic_Signal	2
23	Turning_Loop	1

Length of Unique Cities

In [117]:
```
# Number of unique cities
unique_cities = df_new['City'].nunique()
print(f"Number of unique cities: {unique_cities}")
```

```
Number of unique cities: 1194
```

Accident Count by Cities

In [118]:
```
# Count of accidents by city
accidents_by_cities = df_new['City'].value_counts()
accidents_by_cities
```

Out[118]:
```
Los Angeles      68963
Sacramento       32559
San Diego        26627
San Jose         13376
Riverside        12861
                 ...
Sultana              1
Westmorland          1
West Sierra          1
Dillon Beach         1
Canyon Lake          1
Name: City, Length: 1194, dtype: int64
```

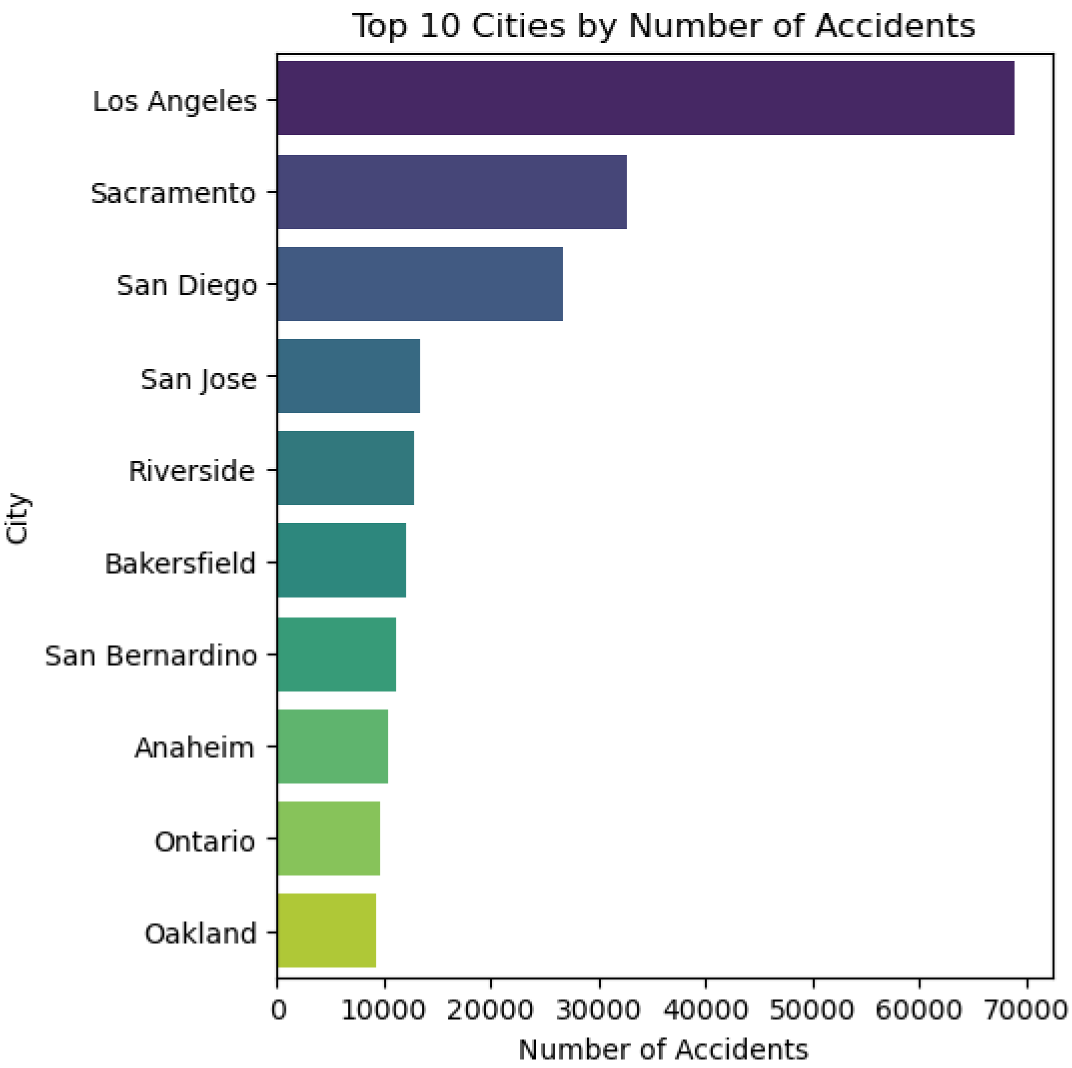
Top 10 Cities by Number of Accidents

In [120]:
```
# Top 10 cities with the most accidents
top_10_cities = accidents_by_cities.head(10)
print("Top 10 cities by number of accidents:")
print(top_10_cities)
```

```
Top 10 cities by number of accidents:
Los Angeles        68963
Sacramento         32559
San Diego          26627
San Jose           13376
Riverside          12861
Bakersfield        12044
San Bernardino     11249
Anaheim            10502
Ontario             9719
Oakland             9255
Name: City, dtype: int64
```

Bar Chart of Top 10 Cities

In [121]:
```
# Bar chart for top 10 cities
plt.figure(figsize=(5, 6))
sns.barplot(x=top_10_cities.values, y=top_10_cities.index, palette='viridis
plt.title('Top 10 Cities by Number of Accidents')
plt.xlabel('Number of Accidents')
plt.ylabel('City')
plt.show()
```



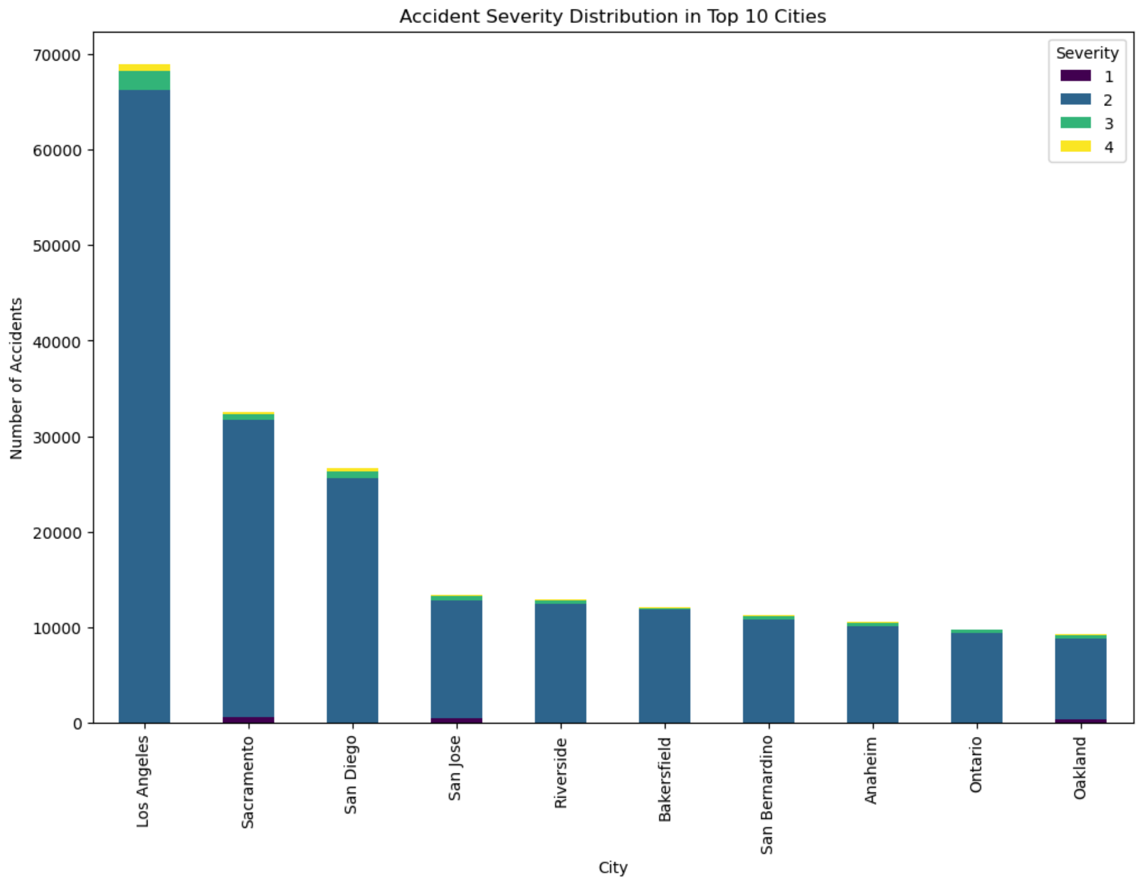
Group By Severity and City

In [122]:
```
# Group by city and severity
severity_by_city = df_new.groupby(['City', 'Severity']).size().unstack(fill

print("Accidents grouped by city and severity:")
print(severity_by_city.head())
```

```
Accidents grouped by city and severity:
Severity       1     2   3   4
City
Acampo        11   487   6   2
Acton          0  1231  57  21
Adelanto       0   188   2   5
Adin           0    11   0   0
Agoura Hills   0   656  38  14
```

Stacked Bar Chart for Severity by City

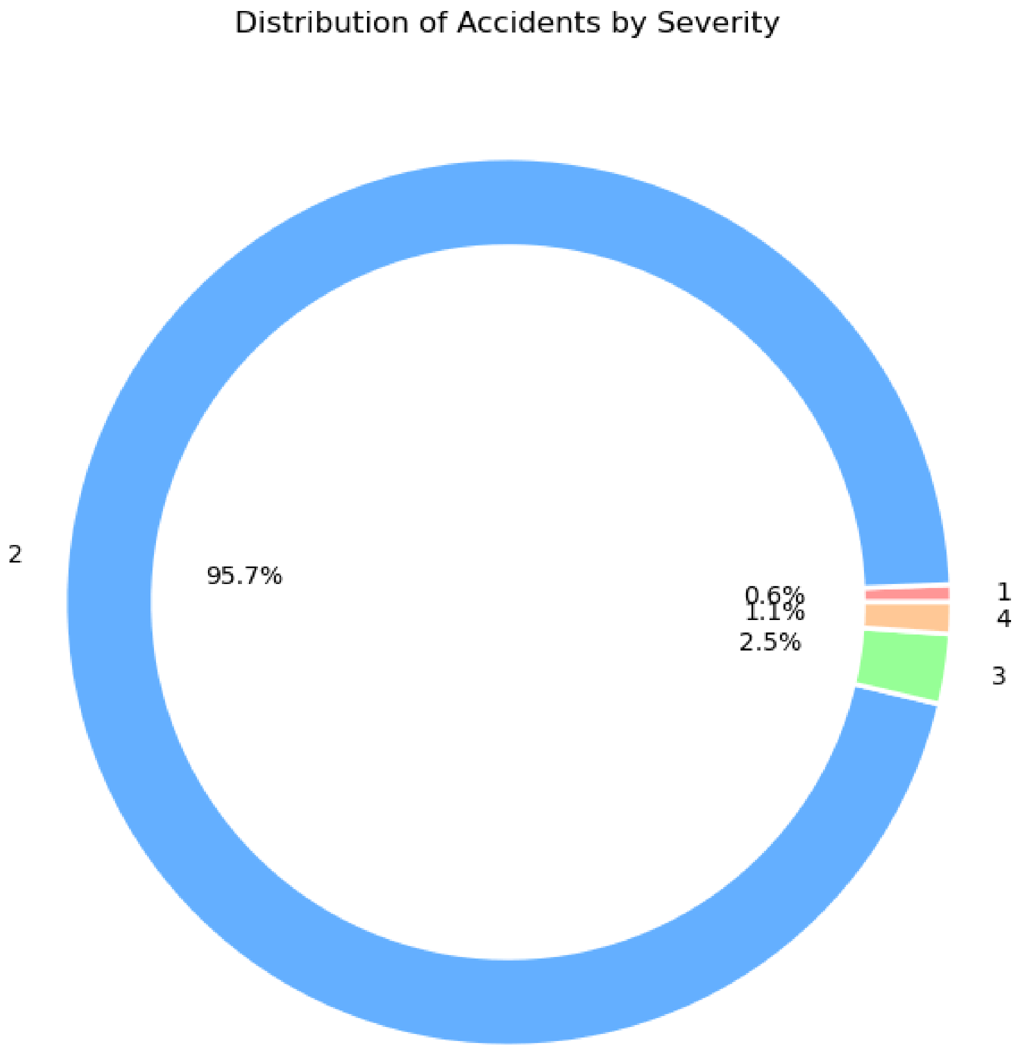
In [123]:
```
# Stacked bar chart for top 10 cities by severity
top_10_cities_severity = severity_by_city.loc[top_10_cities.index]

top_10_cities_severity.plot(kind='bar', stacked=True, figsize=(12, 8), colo
plt.title('Accident Severity Distribution in Top 10 Cities')
plt.xlabel('City')
plt.ylabel('Number of Accidents')
plt.legend(title='Severity')
plt.show()
```



Group By Severity and ID

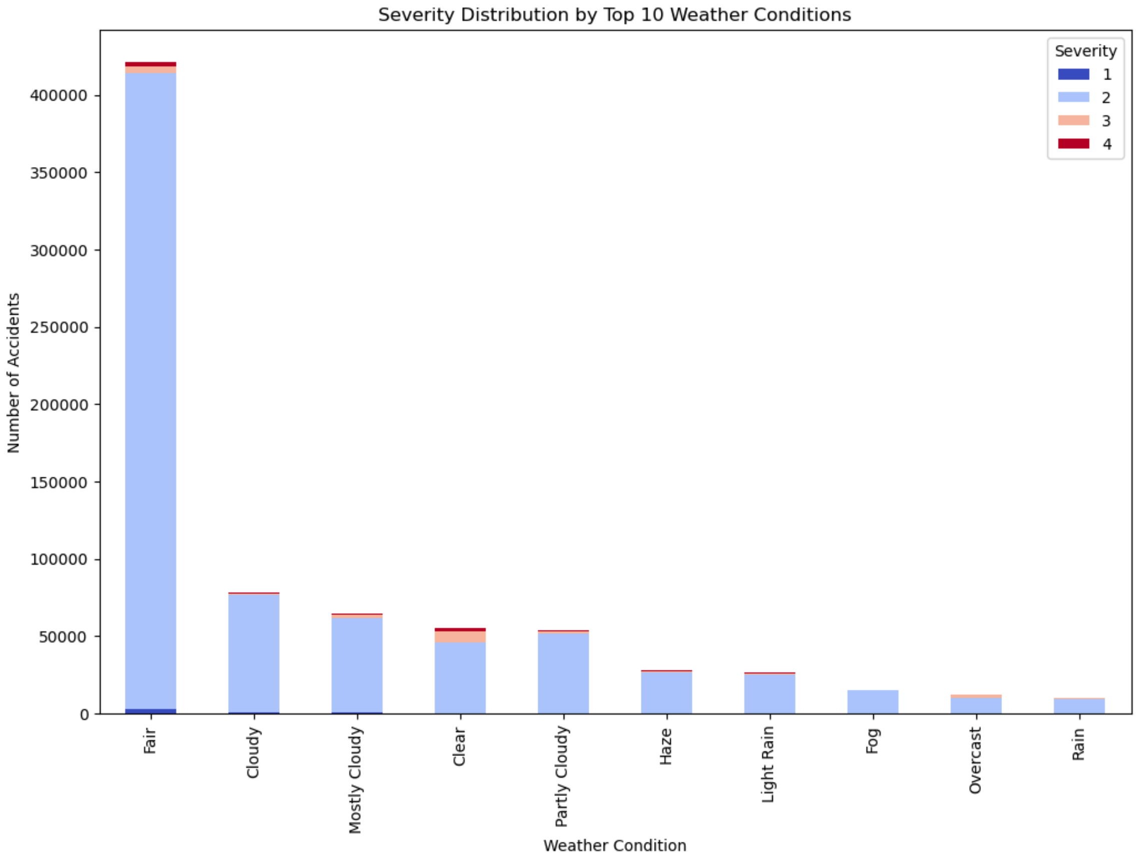
In [150]:
```
# Group by severity and count unique IDs
severity_count = df_new.groupby('Severity')['ID'].nunique()
print("Number of unique accidents by severity:")
print(severity_count)

# Pie chart for severity distribution

severity_count.plot(kind='pie', autopct='%1.1f%%', figsize=(8, 9),wedgeprop
 colors=['#ff9999', '#66b3ff', '#99ff99','#ffcc99'])
plt.title('Distribution of Accidents by Severity')
plt.ylabel('')  # Remove default ylabel
plt.show()
```

```
Number of unique accidents by severity:
Severity
1      5058
2    761462
3     20213
4      9135
Name: ID, dtype: int64
```



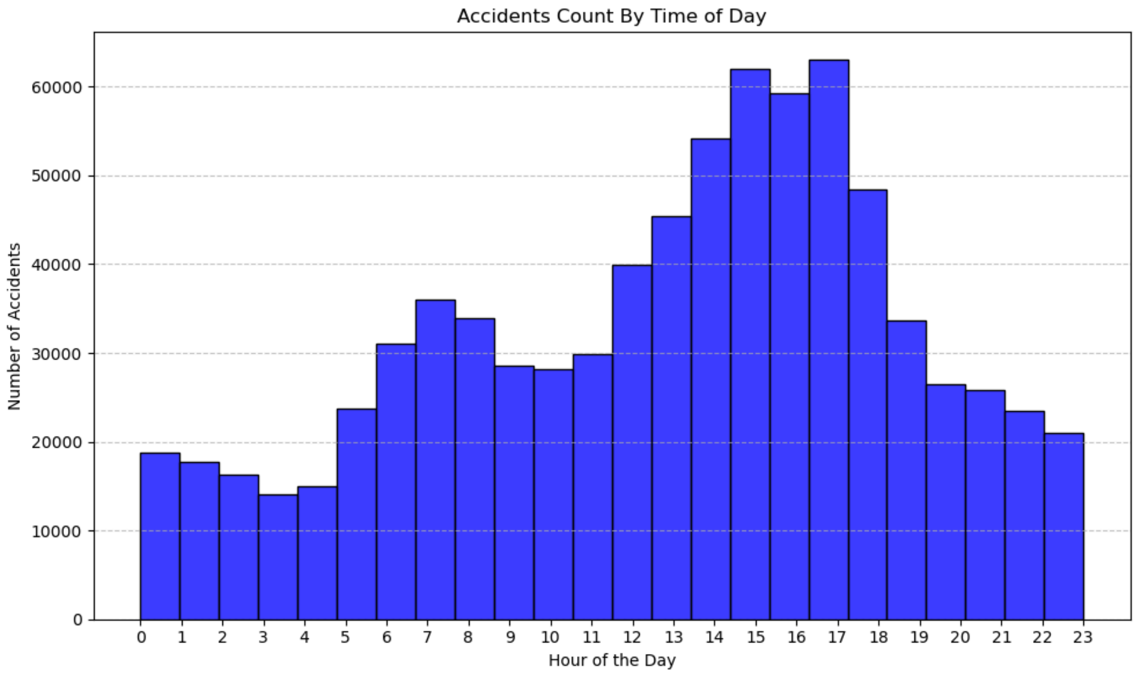
Severity Distribution by Weather Condition

In [138]:
```
# Group by weather condition and severity
weather_severity = df_new.groupby(['Weather_Condition', 'Severity']).size()

# Top 10 weather conditions by total accidents
top_weather_conditions = weather_severity.sum(axis=1).sort_values(ascending

# Filter data for these conditions
top_weather_severity = weather_severity.loc[top_weather_conditions.index]

# Stacked bar chart
top_weather_severity.plot(kind='bar', stacked=True, figsize=(12, 8), colorm
plt.title('Severity Distribution by Top 10 Weather Conditions')
plt.xlabel('Weather Condition')
plt.ylabel('Number of Accidents')
plt.legend(title='Severity')
plt.show()
```



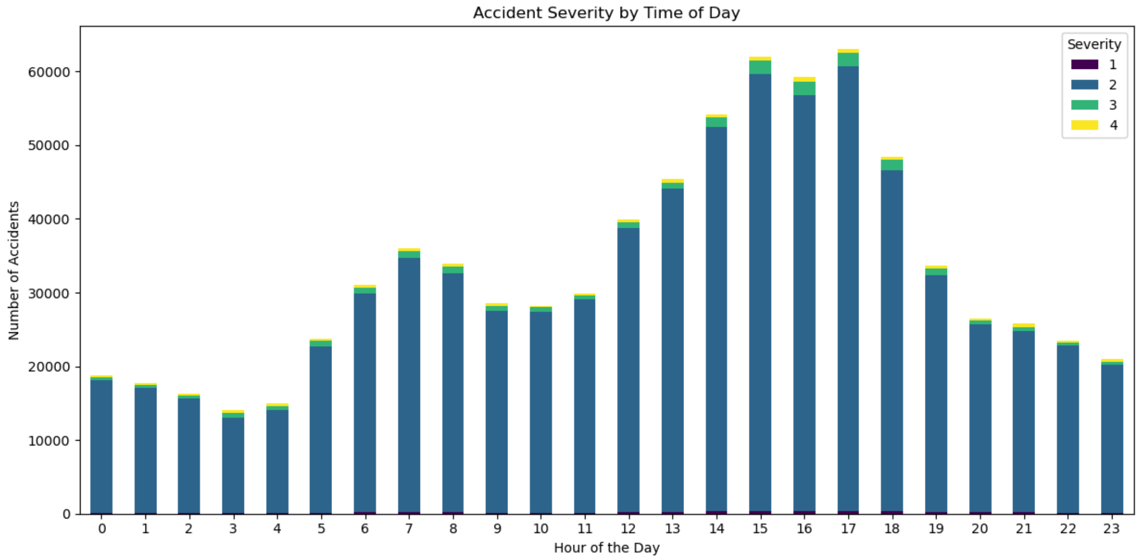
Accidents count by the time of day

In [155]:
```
# Extract hour from 'Start_Time'
df_new['Hour'] = df_new['Start_Time'].dt.hour

# Plot histogram for accidents count by time of day
plt.figure(figsize=(10, 6))
sns.histplot(df_new['Hour'], bins=24, kde=False, color='blue', edgecolor='b
plt.title('Accidents Count By Time of Day')
plt.xlabel('Hour of the Day')
plt.ylabel('Number of Accidents')
plt.xticks(range(0, 24))  # Set x-axis ticks from 0 to 23
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.tight_layout()  # Adjust layout for better fit
plt.show()
```



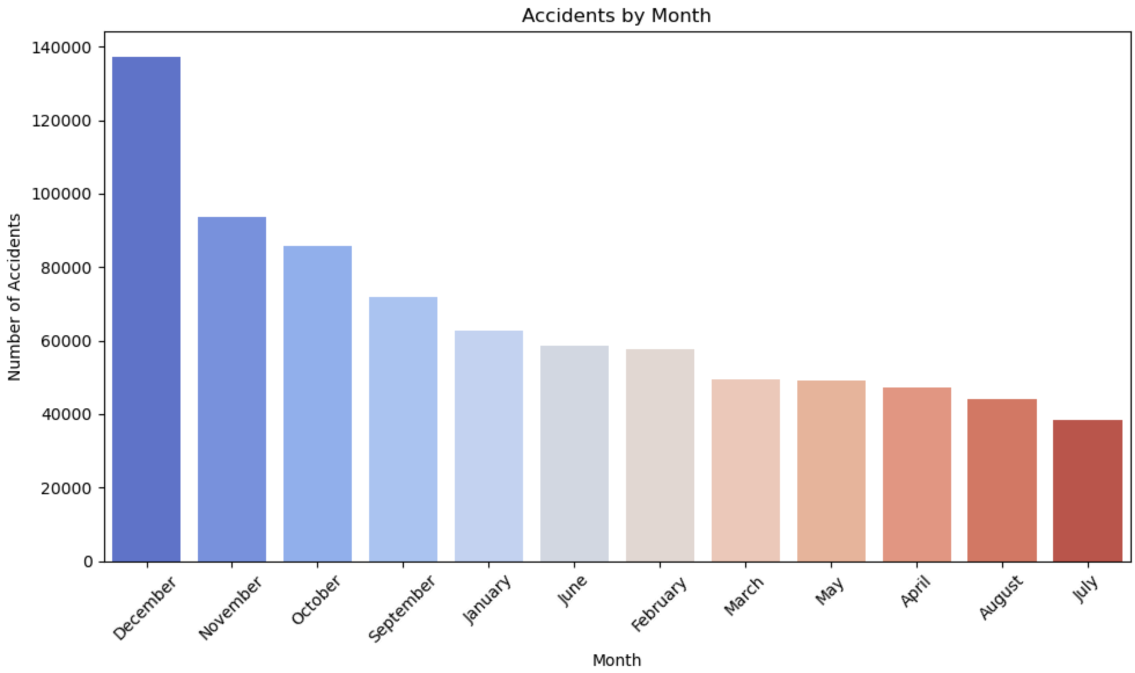
Accident Severity by Time of Day

In [157]:
```
# Group by hour and severity count
severity_by_hour = df_new.groupby(['Hour', 'Severity']).size().unstack()

# Plot severity by time of day
severity_by_hour.plot(kind='bar', stacked=True, figsize=(12, 6), colormap='
plt.title('Accident Severity by Time of Day')
plt.xlabel('Hour of the Day')
plt.ylabel('Number of Accidents')
plt.xticks(rotation=0)
plt.legend(title='Severity')
plt.tight_layout()
plt.show()
```



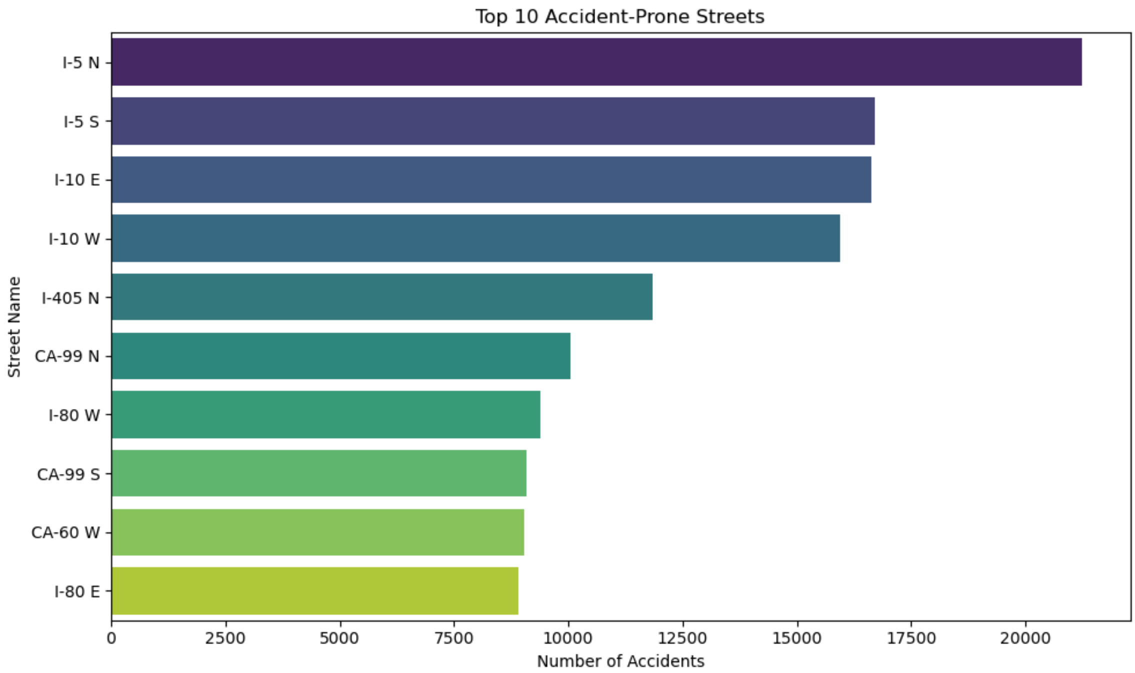
Accidents by Month

In [162]:
```
# Extract month from Start_Time
df_new['Month'] = df_new['Start_Time'].dt.month_name()

# Count accidents by month
monthly_accidents = df_new['Month'].value_counts()

# Plot bar chart
plt.figure(figsize=(10, 6))
sns.barplot(x=monthly_accidents.index, y=monthly_accidents.values, palette=
plt.title('Accidents by Month')
plt.xlabel('Month')
plt.ylabel('Number of Accidents')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



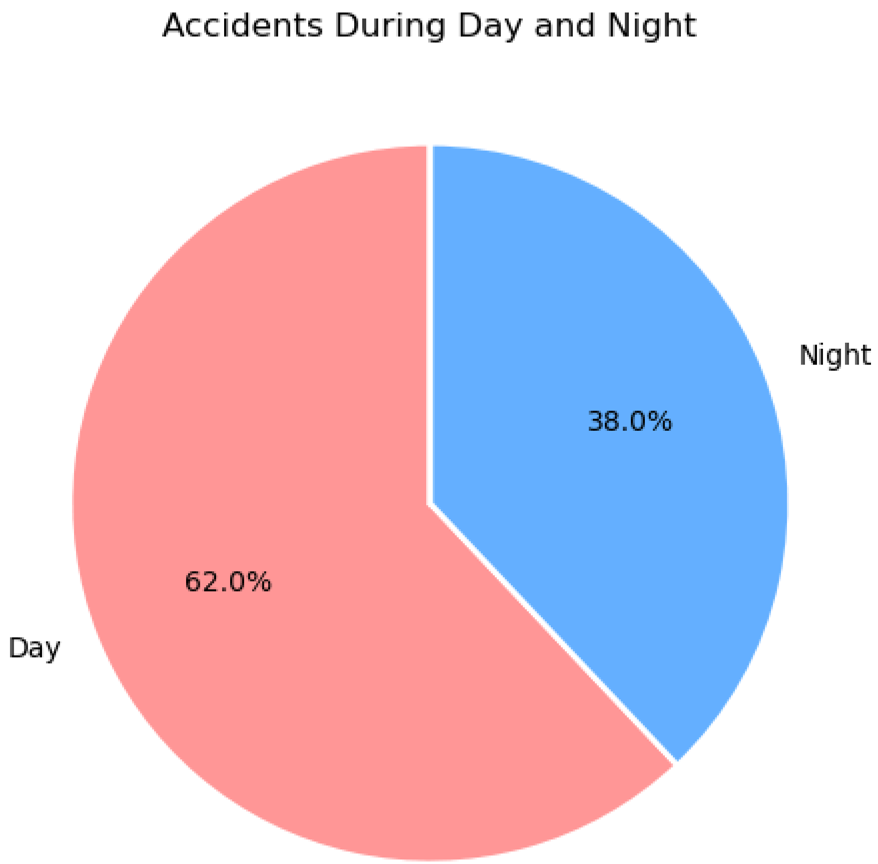
Count accidents by street

In [163]:
```
# Count accidents by street
top_streets = df_new['Street'].value_counts().head(10)

# Plot bar chart
plt.figure(figsize=(10, 6))
sns.barplot(x=top_streets.values, y=top_streets.index, palette='viridis')
plt.title('Top 10 Accident-Prone Streets')
plt.xlabel('Number of Accidents')
plt.ylabel('Street Name')
plt.tight_layout()
plt.show()
```



Accidents During Day/Night

In [167]:
```
# Count accidents by day and night
day_night_accidents = df_new['Sunrise_Sunset'].value_counts()

# Plot pie chart for accidents during day and night
day_night_accidents.plot(kind='pie', autopct='%1.1f%%', figsize=(6, 5),
                         colors=['#ff9999', '#66b3ff'], startangle=90,
                         wedgeprops={'edgecolor': 'w', 'linewidth': 2})
plt.title('Accidents During Day and Night')
plt.ylabel('')  # Remove default ylabel
plt.tight_layout()
plt.show()
```



Conclusion

- This project successfully identified key patterns in traffic accidents, such as the distribution of accidents by severity, time of day, weather conditions, and location.
- The analysis highlighted significant trends, including high accident occurrences during adverse weather and specific times of the day.
- By presenting these findings through intuitive visualizations, the project provides actionable insights for traffic authorities and policymakers.
- Implementing the recommendations derived from this analysis can contribute to improved road safety, reduced accidents, and enhanced public safety measures.