GIT Console restricted only for developers

GIT CONSOLE

[ ] ↳ *10 cells hidden*

Git Console II

↳ *2 cells hidden*

Data Feed

```
from google.colab import drive
import os
Drive_dir = '/content/drive/MyDrive/'
folder_name = 'AccidentDataset'

drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

Upload kaggle json file, if the below code does not work then upload kaggle Json and run this again

```
from google.colab import files
files.upload()  # Upload the 'kaggle.json' file here
```

```
Choose Files No file chosen    Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
{}
```

```
# !mkdir -p ~/.kaggle
# !cp kaggle.json ~/.kaggle/
# !chmod 600 ~/.kaggle/kaggle.json  # Set permissions
```

```
cp: cannot stat 'kaggle.json': No such file or directory
chmod: cannot access '/root/.kaggle/kaggle.json': No such file or directory
```

This will download the dataset directly into your linked google drive account

```
%cd Drive_dir

# Full path to the folder
folder_path = os.path.join(Drive_dir, folder_name)

# Check if the folder already exists
if not os.path.exists(folder_path):
    # Create the folder
    os.makedirs(folder_path)
    print(f"Folder '{folder_name}' created at '{folder_path}'")
else:
    # Print message if folder exists
    print('Folder exists')
```

```
[Errno 2] No such file or directory: 'Drive_dir'
/content
Folder exists
```

```
%cd /content/drive/MyDrive/AccidentDataset
!kaggle datasets download -d sobhanmoosavi/us-accidents
```

```
/content/drive/MyDrive/AccidentDataset
Dataset URL: https://www.kaggle.com/datasets/sobhanmoosavi/us-accidents
License(s): CC-BY-NC-SA-4.0
Downloading us-accidents.zip to /content/drive/MyDrive/AccidentDataset
100% 653M/653M [00:07<00:00, 110MB/s]
100% 653M/653M [00:07<00:00, 94.6MB/s]
```

```
!unzip us-accidents.zip -d /content/drive/MyDrive/AccidentDataset
```

```
Archive:  us-accidents.zip
  inflating: /content/drive/MyDrive/AccidentDataset/US_Accidents_March23.csv
```

View Dataset

Will exhaust RAM if worked with free tier colab

```
import pandas as pd
data = pd.read_csv('/content/drive/MyDrive/US_Accidents_March23.csv')
```

Processing data in chunks or batches for memory efficiency (Works well but time consuming)

```
import pandas as pd

# Load dataset in chunks of 2,000 rows
chunk_size = 2000
chunk_list = []  # Append each chunk's dataframe here

for chunk in pd.read_csv('/content/drive/MyDrive/AccidentDataset/US_Accidents_March23.csv', chunksize=chunk_size):
    chunk_list.append(chunk)

# Concatenate chunks into a single dataframe
data = pd.concat(chunk_list, axis=0)
```

```
data.head()
```

	ID	Source	Severity	Start_Time	End_Time	Start_Lat	Start_Lng	End_Lat	End_Lng	Distance(mi)	...	Roundabout	Station	Stop
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	...	False	False	False
1	A-2	Source2	2	2016-02-08 06:07:59	2016-02-08 06:37:59	39.928059	-82.831184	NaN	NaN	0.01	...	False	False	False
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	...	False	False	False
3	A-4	Source2	3	2016-02-08 07:23:34	2016-02-08 07:53:34	39.747753	-84.205582	NaN	NaN	0.01	...	False	False	False
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	...	False	False	False

5 rows × 46 columns

```
data.tail()
```

	ID	Source	Severity	Start_Time	End_Time	Start_Lat	Start_Lng	End_Lat	End_Lng	Distance(mi)	...	Roundabout
7728389	A-7777757	Source1	2	2019-08-23 18:03:25	2019-08-23 18:32:01	34.00248	-117.37936	33.99888	-117.37094	0.543	...	False
7728390	A-7777758	Source1	2	2019-08-23 19:11:30	2019-08-23 19:38:23	32.76696	-117.14806	32.76555	-117.15363	0.338	...	False
7728391	A-7777759	Source1	2	2019-08-23 19:00:21	2019-08-23 19:28:49	33.77545	-117.84779	33.77740	-117.85727	0.561	...	False
7728392	A-7777760	Source1	2	2019-08-23 19:00:21	2019-08-23 19:29:42	33.99246	-118.40302	33.98311	-118.39565	0.772	...	False
7728393	A-7777761	Source1	2	2019-08-23 18:52:06	2019-08-23 19:21:31	34.13393	-117.23092	34.13736	-117.23934	0.537	...	False

5 rows × 46 columns

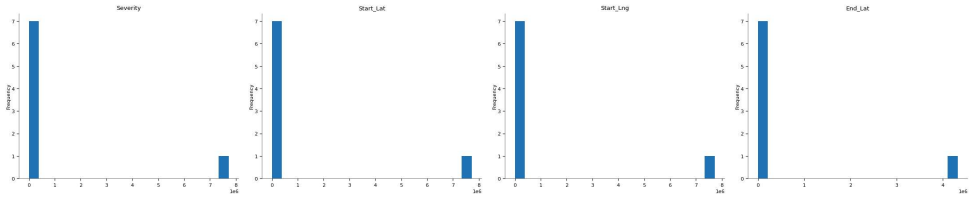
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7728394 entries, 0 to 7728393
Data columns (total 46 columns):
 #   Column                 Dtype
---  ------                 -----
 0   ID                     object
 1   Source                 object
 2   Severity               int64
 3   Start_Time             object
 4   End_Time               object
 5   Start_Lat              float64
 6   Start_Lng              float64
 7   End_Lat                float64
 8   End_Lng                float64
 9   Distance(mi)           float64
 10  Description            object
 11  Street                 object
 12  City                   object
 13  County                 object
 14  State                  object
 15  Zipcode                object
 16  Country                object
 17  Timezone               object
 18  Airport_Code           object
 19  Weather_Timestamp      object
 20  Temperature(F)         float64
 21  Wind_Chill(F)          float64
 22  Humidity(%)            float64
 23  Pressure(in)           float64
 24  Visibility(mi)         float64
 25  Wind_Direction         object
 26  Wind_Speed(mph)        float64
 27  Precipitation(in)      float64
 28  Weather_Condition      object
 29  Amenity                bool
 30  Bump                   bool
 31  Crossing               bool
 32  Give_Way               bool
 33  Junction               bool
 34  No_Exit                bool
 35  Railway                bool
 36  Roundabout             bool
 37  Station                bool
 38  Stop                   bool
 39  Traffic_Calming        bool
 40  Traffic_Signal         bool
 41  Turning_Loop           bool
 42  Sunrise_Sunset         object
 43  Civil_Twilight         object
 44  Nautical_Twilight      object
 45  Astronomical_Twilight  object
dtypes: bool(13), float64(12), int64(1), object(20)
memory usage: 2.0+ GB
```

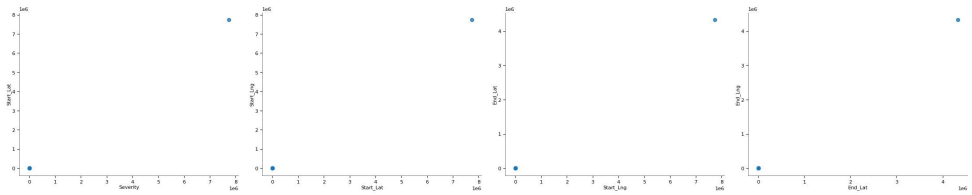
```
data.describe()
```

	Severity	Start_Lat	Start_Lng	End_Lat	End_Lng	Distance(mi)	Temperature(F)	Wind_Chill(F)	Humidity(%)
count	7.728394e+06	7.728394e+06	7.728394e+06	4.325632e+06	4.325632e+06	7.728394e+06	7.564541e+06	5.729375e+06	7.554250e+06
mean	2.212384e+00	3.620119e+01	-9.470255e+01	3.626183e+01	-9.572557e+01	5.618423e-01	6.166329e+01	5.825105e+01	6.483104e+01
std	4.875313e-01	5.076079e+00	1.739176e+01	5.272905e+00	1.810793e+01	1.776811e+00	1.901365e+01	2.238983e+01	2.282097e+01
min	1.000000e+00	2.455480e+01	-1.246238e+02	2.456601e+01	-1.245457e+02	0.000000e+00	-8.900000e+01	-8.900000e+01	1.000000e+00
25%	2.000000e+00	3.339963e+01	-1.172194e+02	3.346207e+01	-1.177543e+02	0.000000e+00	4.900000e+01	4.300000e+01	4.800000e+01
50%	2.000000e+00	3.582397e+01	-8.776662e+01	3.618349e+01	-8.802789e+01	3.000000e-02	6.400000e+01	6.200000e+01	6.700000e+01
75%	2.000000e+00	4.008496e+01	-8.035368e+01	4.017892e+01	-8.024709e+01	4.640000e-01	7.600000e+01	7.500000e+01	8.400000e+01
max	4.000000e+00	4.900220e+01	-6.711317e+01	4.907500e+01	-6.710924e+01	4.417500e+02	2.070000e+02	2.070000e+02	1.000000e+02

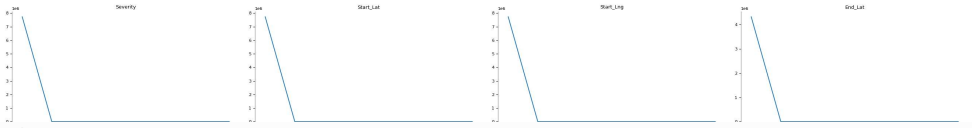
Distributions



2-d distributions



Values



Data Cleaning

```
# Check for missing values in the dataset
missing_values = data.isnull().sum()
print(missing_values[missing_values > 0])
```

```
End_Lat                 3402762
End_Lng                 3402762
Description                   5
Street                    10869
City                        253
Zipcode                    1915
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
Sunrise_Sunset            23246
Civil_Twilight            23246
Nautical_Twilight         23246
Astronomical_Twilight     23246
dtype: int64
```

```
columns_to_drop = ['End_Lat', 'End_Lng', 'Description', 'Street', 'City', 'Zipcode',
                   'Timezone', 'Airport_Code', 'Wind_Chill(F)', 'Weather_Timestamp']
data.drop(columns=columns_to_drop, inplace=True)
```

Handling Missing Values

```
data['Temperature(F)'].fillna(data['Temperature(F)'].median(), inplace=True)
data['Humidity(%)'].fillna(data['Humidity(%)'].median(), inplace=True)
data['Pressure(in)'].fillna(data['Pressure(in)'].median(), inplace=True)
data['Visibility(mi)'].fillna(data['Visibility(mi)'].median(), inplace=True)
data['Wind_Speed(mph)'].fillna(data['Wind_Speed(mph)'].median(), inplace=True)


data['Weather_Condition'].fillna(data['Weather_Condition'].mode()[0], inplace=True)


data['Precipitation(in)'].fillna(0, inplace=True)


time_columns = ['Sunrise_Sunset', 'Civil_Twilight', 'Nautical_Twilight', 'Astronomical_Twilight']
for col in time_columns:
    data[col].fillna(data[col].mode()[0], inplace=True)


print(data.isnull().sum())
```

```
ID                          0
Source                      0
Severity                    0
Start_Time                  0
End_Time                    0
Start_Lat                   0
Start_Lng                   0
Distance(mi)                0
County                      0
State                       0
Country                     0
Temperature(F)              0
Humidity(%)                 0
Pressure(in)                0
Visibility(mi)              0
Wind_Direction         175206
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
```

```
data.to_csv('/content/drive/MyDrive/AccidentDataset/cleaned_data.csv', index=False)
```

Saving checkpoint (Reset colab runtime to save RAM)

```
from google.colab import drive
import os
import pandas as pd
Drive_dir = '/content/drive/MyDrive/'
folder_name = 'AccidentDataset'

drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

slow chunking process

```
file_path = '/content/drive/MyDrive/AccidentDataset/cleaned_data.csv'

chunk_size = 1000
data_chunks = pd.read_csv(file_path, chunksize=chunk_size)
chunk_list = []

for chunk in data_chunks:
    chunk_list.append(chunk)
```

```
new_data = pd.concat(chunk_list, axis=0)

print(f'Total number of rows and columns in the dataset: {new_data.shape}')
```

```
Total number of rows and columns in the dataset: (7728394, 36)
```

Overview and Insights

Distribution of Accident Severity

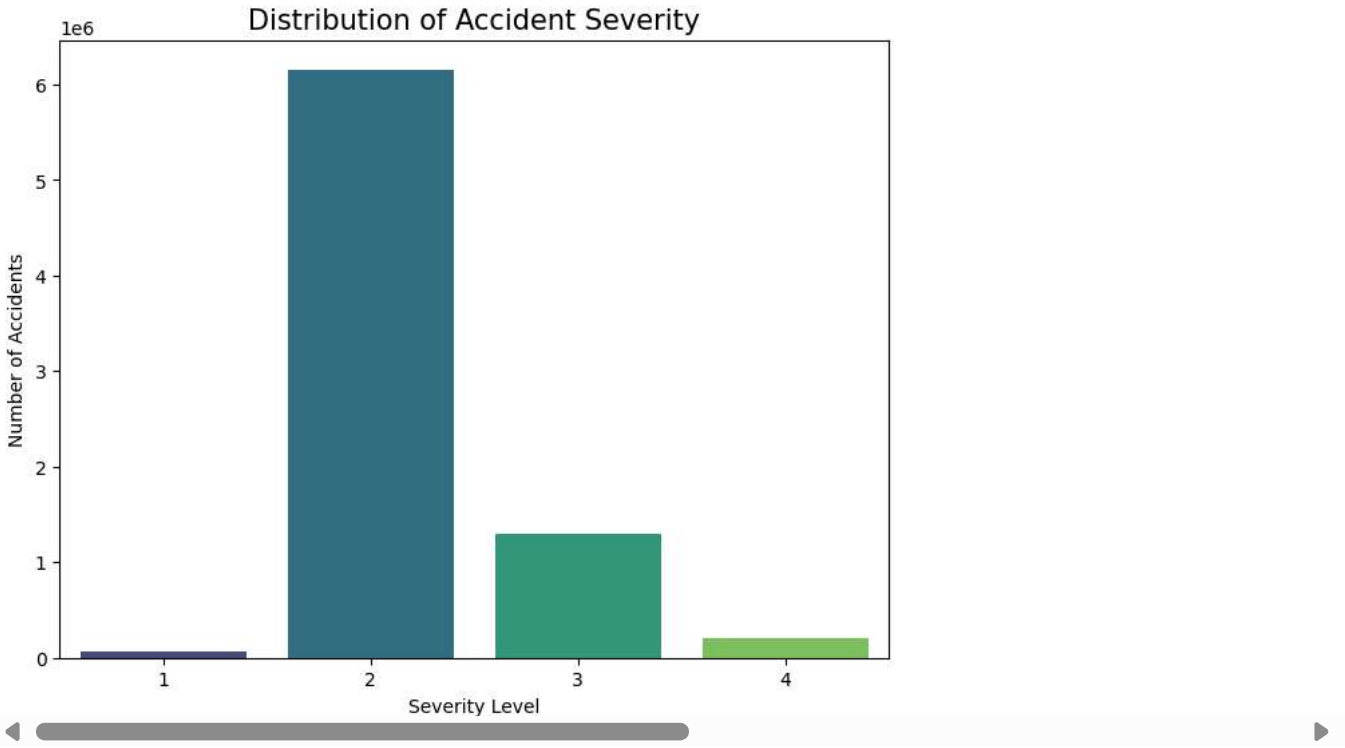
```
import matplotlib.pyplot as plt
import seaborn as sns

# Plot the distribution of accident severity
plt.figure(figsize=(8, 6))
sns.countplot(x='Severity', data=new_data, palette='viridis')
plt.title('Distribution of Accident Severity', fontsize=15)
plt.xlabel('Severity Level')
plt.ylabel('Number of Accidents')
plt.show()
```

```
<ipython-input-5-8708a8912358>:6: FutureWarning:

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

  sns.countplot(x='Severity', data=new_data, palette='viridis')
```



Create features Hour, Day(week), Month

```
# Convert 'Start_Time' to datetime format if not already done
new_data['Start_Time'] = pd.to_datetime(new_data['Start_Time'], errors='coerce')

# Extract hour, day of week, and month from 'Start_Time'
new_data['Hour'] = new_data['Start_Time'].dt.hour
new_data['DayOfWeek'] = new_data['Start_Time'].dt.dayofweek  # Monday=0, Sunday=6
new_data['Month'] = new_data['Start_Time'].dt.month
```

Time-based Analysis

```
# Plot accidents by hour of the day
plt.figure(figsize=(10, 6))
sns.countplot(x='Hour', data=new_data, palette='icefire')
plt.title('Number of Accidents by Hour of Day', fontsize=15)
plt.xlabel('Hour of Day')
plt.ylabel('Number of Accidents')
plt.show()
```

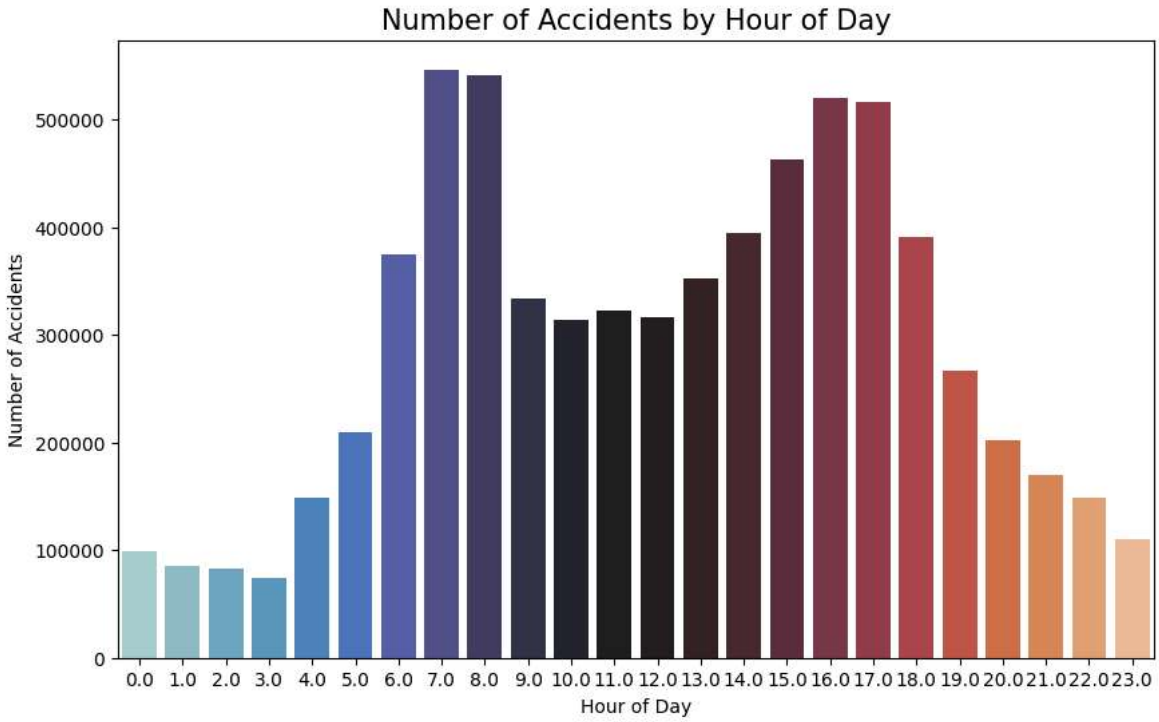
```
# Plot accidents by day of the week
plt.figure(figsize=(10, 6))
sns.countplot(x='DayOfWeek', data=new_data, palette='icefire')
plt.title('Number of Accidents by Day of the Week', fontsize=15)
plt.xlabel('Day of the Week (0=Monday, 6=Sunday)')
plt.ylabel('Number of Accidents')
plt.show()

# Plot accidents by month
plt.figure(figsize=(10, 6))
sns.countplot(x='Month', data=new_data, palette='icefire')
plt.title('Number of Accidents by Month', fontsize=15)
plt.xlabel('Month')
plt.ylabel('Number of Accidents')
plt.show()
```

```
<ipython-input-9-11bbd801f107>:3: FutureWarning:

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

  sns.countplot(x='Hour', data=new_data, palette='icefire')
```




```
<ipython-input-9-11bbd801f107>:11: FutureWarning:
```

Weather and Severity Analysis

```
# Boxplot of temperature vs. severity
plt.figure(figsize=(10, 6))
sns.boxplot(x='Severity', y='Temperature(F)', data=new_data, palette='coolwarm')
plt.title('Temperature vs. Accident Severity', fontsize=15)
plt.show()

# Boxplot of visibility vs. severity
plt.figure(figsize=(10, 6))
sns.boxplot(x='Severity', y='Visibility(mi)', data=new_data, palette='coolwarm')
plt.title('Visibility vs. Accident Severity', fontsize=15)
plt.show()

# Boxplot of wind speed vs. severity
plt.figure(figsize=(10, 6))
sns.boxplot(x='Severity', y='Wind_Speed(mph)', data=new_data, palette='coolwarm')
plt.title('Wind Speed vs. Accident Severity', fontsize=15)
```