By Ajitha

Road accident Data Analysis

Jupyter notebook vs jupyter lab

- Jupyter Notebook has a simpler and more limited interface compared to JupyterLab. JupyterLab offers a more flexible and customizable workspace with support for multiple documents and interactive components.
- While both Jupyter Notebook and JupyterLab support interactive computing and document creation, JupyterLab provides a more extensive set of features and a more integrated development environment.
- Jupyter Notebook is widely used for interactive computing, data analysis, and scientific research. JupyterLab is gaining popularity as a more powerful and versatile environment for interactive computing, software development, and collaborative work.

Importing lib and dataset

1.Importing lib

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

2.Importing data set

```
data=pd.read_excel("E:\cubeAi\Road Accident Data.xlsx")
```

3.To find the no.of columns and rows

```
data.shape
```

```
(307973, 21)
```

4.Data in the first 5 rows

```
data.head()
```

	Accident_Index	Accident Date	Day_of_Week	Junction_Control	Junction_Detail	Accident_Severity	Latitude	Light_Conditions	Local_Authority_(District)	Carriageway_Haz
0	BS0000001	2021-01-01	Thursday	Give way or uncontrolled	T or staggered junction	Serious	51.512273	Daylight	Kensington and Chelsea	
1	BS0000002	2021-01-05	Monday	Give way or uncontrolled	Crossroads	Serious	51.514399	Daylight	Kensington and Chelsea	
2	BS0000003	2021-01-04	Sunday	Give way or uncontrolled	T or staggered junction	Slight	51.486668	Daylight	Kensington and Chelsea	
3	BS0000004	2021-01-05	Monday	Auto traffic signal	T or staggered junction	Serious	51.507804	Daylight	Kensington and Chelsea	
4	BS0000005	2021-01-06	Tuesday	Auto traffic signal	Crossroads	Serious	51.482076	Darkness - lights lit	Kensington and Chelsea	

5 rows × 21 columns

5.Data in the last 5 rows

```
data.tail()
```

	Accident_Index	Accident Date	Day_of_Week	Junction_Control	Junction_Detail	Accident_Severity	Latitude	Light_Conditions	Local_Authority_(District)	Carriageway
307968	BS0307969	2022-02-18	Thursday	Data missing or out of range	Not at junction or within 20 metres	Slight	57.374005	Daylight	Highland	
307969	BS0307970	2022-02-21	Sunday	Data missing or out of range	Not at junction or within 20 metres	Slight	57.232273	Darkness - no lighting	Highland	
307970	BS0307971	2022-02-23	Tuesday	Give way or uncontrolled	T or staggered junction	Slight	57.585044	Daylight	Highland	
307971	BS0307972	2022-02-23	Tuesday	Give way or uncontrolled	T or staggered junction	Serious	57.214898	Darkness - no lighting	Highland	
307972	BS0307973	2022-02-28	Sunday	Give way or uncontrolled	T or staggered junction	Serious	57.575210	Daylight	Highland	Other obje

5 rows × 21 columns

6.To find the data types of all the columns

```
data.dtypes
```

```
Accident_Index                          object
Accident Date                   datetime64[ns]
Day_of_Week                             object
Junction_Control                        object
Junction_Detail                         object
Accident_Severity                       object
Latitude                               float64
Light_Conditions                        object
Local_Authority_(District)              object
Carriageway_Hazards                     object
Longitude                              float64
Number_of_Casualties                     int64
Number_of_Vehicles                       int64
Police_Force                            object
Road_Surface_Conditions                 object
Road_Type                               object
Speed_limit                              int64
Time                                    object
Urban_or_Rural_Area                     object
Weather_Conditions                      object
Vehicle_Type                            object
dtype: object
```

7.To get the column names

```
data.columns
```

```
Index(['Accident_Index', 'Accident Date', 'Day_of_Week', 'Junction_Control',
       'Junction_Detail', 'Accident_Severity', 'Latitude', 'Light_Conditions',
       'Local_Authority_(District)', 'Carriageway_Hazards', 'Longitude',
       'Number_of_Casualties', 'Number_of_Vehicles', 'Police_Force',
       'Road_Surface_Conditions', 'Road_Type', 'Speed_limit', 'Time',
       'Urban_or_Rural_Area', 'Weather_Conditions', 'Vehicle_Type'],
      dtype='object')
```

To get the info of the data set

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 307973 entries, 0 to 307972
Data columns (total 21 columns):
 #   Column                       Non-Null Count   Dtype
---  ------                       --------------   -----
 0   Accident_Index               307973 non-null  object
 1   Accident Date                307973 non-null  datetime64[ns]
 2   Day_of_Week                  307973 non-null  object
 3   Junction_Control             307973 non-null  object
 4   Junction_Detail              307973 non-null  object
 5   Accident_Severity            307973 non-null  object
 6   Latitude                     307973 non-null  float64
 7   Light_Conditions             307973 non-null  object
 8   Local_Authority_(District)   307973 non-null  object
 9   Carriageway_Hazards          5424 non-null    object
 10  Longitude                    307973 non-null  float64
 11  Number_of_Casualties         307973 non-null  int64
 12  Number_of_Vehicles           307973 non-null  int64
 13  Police_Force                 307973 non-null  object
 14  Road_Surface_Conditions      307973 non-null  object
 15  Road_Type                    307973 non-null  object
 16  Speed_limit                  307973 non-null  int64
 17  Time                         307956 non-null  object
 18  Urban_or_Rural_Area          307973 non-null  object
 19  Weather_Conditions           307973 non-null  object
 20  Vehicle_Type                 307973 non-null  object
dtypes: datetime64[ns](1), float64(2), int64(3), object(15)
memory usage: 49.3+ MB
```

Dtype vs info() function

- **DTYPES ATTRIBUTE**
  The dtypes attribute, on the other hand, returns a Series with the data type of each column. It doesn't provide additional information such as memory usage or the number of non-null values. It's mainly used to access the data types programmatically or to quickly inspect the data types without additional information

- **INFO() METHOD**
  The info() method provides a concise summary of the DataFrame, including the data types of each column, the number of non-null values, and memory usage. It's a very useful method for quickly understanding the structure and properties of a DataFrame.

To generate descriptive statistics of the numerical

```
data.describe()
```

	Accident Date	Latitude	Longitude	Number_of_Casualties	Number_of_Vehicles	Speed_limit
count	307973	307973.000000	307973.000000	307973.000000	307973.000000	307973.000000
mean	2021-12-23 22:19:39.804722944	52.487005	-1.368884	1.356882	1.829063	38.866037
min	2021-01-01 00:00:00	49.914488	-7.516225	1.000000	1.000000	10.000000
25%	2021-06-28 00:00:00	51.485248	-2.247937	1.000000	1.000000	30.000000
50%	2021-12-08 00:00:00	52.225943	-1.349258	1.000000	2.000000	30.000000
75%	2022-06-25 00:00:00	53.415517	-0.206810	1.000000	2.000000	50.000000
max	2022-12-31 00:00:00	60.598055	1.759398	48.000000	32.000000	70.000000
std	NaN	1.339011	1.356092	0.815857	0.710477	14.032933

To find any duplicate values

```
data.duplicated()
```

```
0         False
1         False
2         False
3         False
4         False
          ...
307968    False
307969    False
307970    False
307971    False
307972    False
Length: 307973, dtype: bool
```

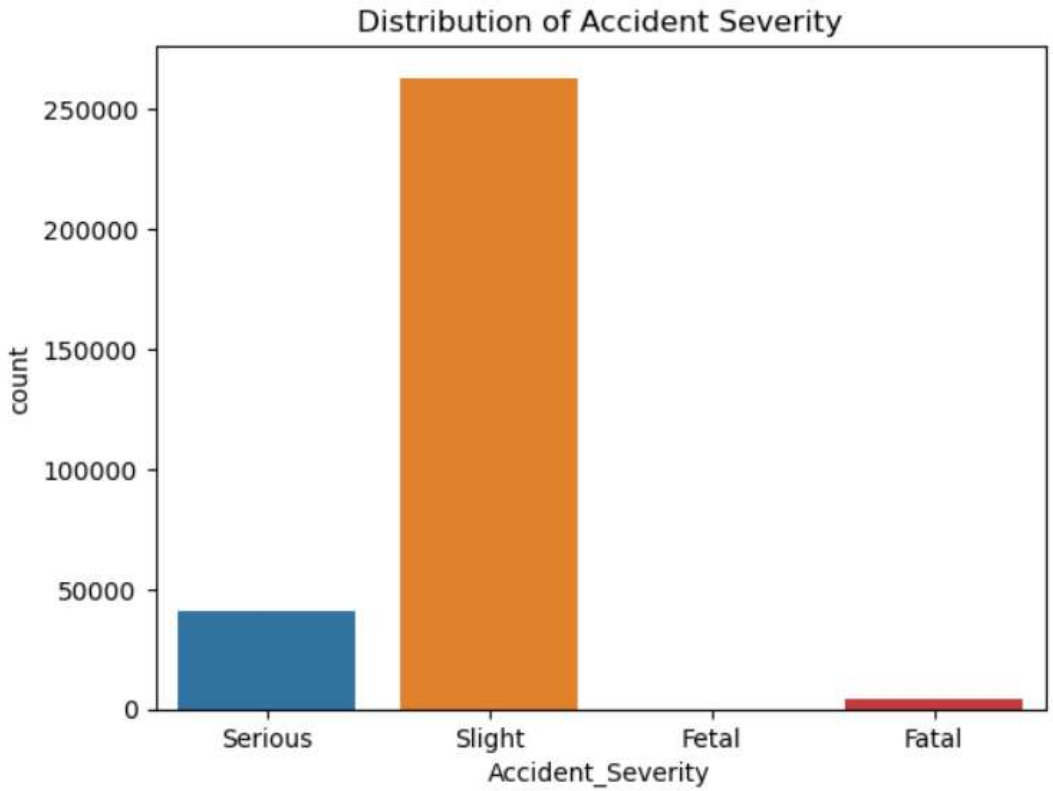
To find unique value of particular column

```
# Get the unique values using Series methods
unique_values = data['Local_Authority_(District)'].unique()

# Print the unique values
print(unique_values)
```

```
['Kensington and Chelsea' 'Hammersmith and Fulham' 'Westminster'
 'Hounslow' 'City of London' 'Tower Hamlets' 'Hackney' 'Camden'
 'Southwark' 'Brent' 'Haringey' 'Islington' 'Barnet' 'Ealing' 'Newham'
 'London Airport (Heathrow)' 'Hillingdon' 'Waltham Forest' 'Redbridge'
 'Barking and Dagenham' 'Havering' 'Lambeth' 'Croydon' 'Wandsworth'
 'Bromley' 'Lewisham' 'Greenwich' 'Bexley' 'Harrow' 'Enfield' 'Sutton'
 'Merton' 'Kingston upon Thames' 'Richmond upon Thames' 'Eden' 'Copeland'
 'South Lakeland' 'Barrow-in-Furness' 'Allerdale' 'Carlisle' 'Fylde'
 'Blackpool' 'Wyre' 'Lancaster' 'Chorley' 'West Lancashire' 'South Ribble'
 'Preston' 'Blackburn with Darwen' 'Hyndburn' 'Ribble Valley' 'Burnley'
 'Pendle' 'Rossendale' 'Wirral' 'Sefton' 'St. Helens' 'Liverpool'
 'Knowsley' 'Manchester' 'Salford' 'Tameside' 'Stockport' 'Bolton' 'Wigan'
 'Trafford' 'Bury' 'Rochdale' 'Oldham' 'Vale Royal' 'Crewe and Nantwich'
 'Halton' 'Chester' 'Macclesfield' 'Cheshire East' 'Warrington'
 'Crewe and ntwich' 'Cheshire West and Chester' 'Congleton'
 'Ellesmere Port and Neston' 'Wansbeck' 'Blyth Valley' 'North Tyneside'
 'Newcastle upon Tyne' 'Tynedale' 'Alnwick' 'South Tyneside' 'Gateshead'
 'Castle Morpeth' 'Sunderland' 'Berwick-upon-Tweed' 'Northumberland'
 'Durham' 'County Durham' 'Easington' 'Chester-le-Street' 'Derwentside'
 'Wear Valley' 'Teesdale' 'Darlington' 'Sedgefield' 'Hambleton' 'York'
 'Craven' 'Richmondshire' 'Scarborough' 'Selby' 'Harrogate' 'Ryedale'
 'Calderdale' 'Bradford' 'Wakefield' 'Leeds' 'Kirklees' 'Doncaster'
 'Rotherham' 'Barnsley' 'Sheffield' 'North East Lincolnshire'
 'North Lincolnshire' 'East Riding of Yorkshire'
 'Kingston upon Hull, City of' 'Hartlepool' 'Redcar and Cleveland'
 'Middlesbrough' 'Stockton-on-Tees' 'Birmingham' 'Wolverhampton' 'Walsall'
 'Dudley' 'Sandwell' 'Solihull' 'Coventry' 'Lichfield' 'Stafford'
 'Stoke-on-Trent' 'East Staffordshire' 'Newcastle-under-Lyme'
 'Cannock Chase' 'South Staffordshire' 'Tamworth'
 'Staffordshire Moorlands' 'Wychavon' 'Malvern Hills' 'Worcester'
 'Wyre Forest' 'Herefordshire, County of' 'Shropshire' 'Redditch'
 'Bromsgrove' 'South Shropshire' 'North Shropshire'
 'Shrewsbury and Atcham' 'Oswestry' 'Telford and Wrekin' 'Bridgnorth'
 'Stratford-upon-Avon' 'Warwick' 'North Warwickshire' 'Rugby'
 'Nuneaton and Bedworth' 'Amber Valley' 'Erewash' 'Bolsover'
 'Derbyshire Dales' 'High Peak' 'Chesterfield' 'North East Derbyshire'
```

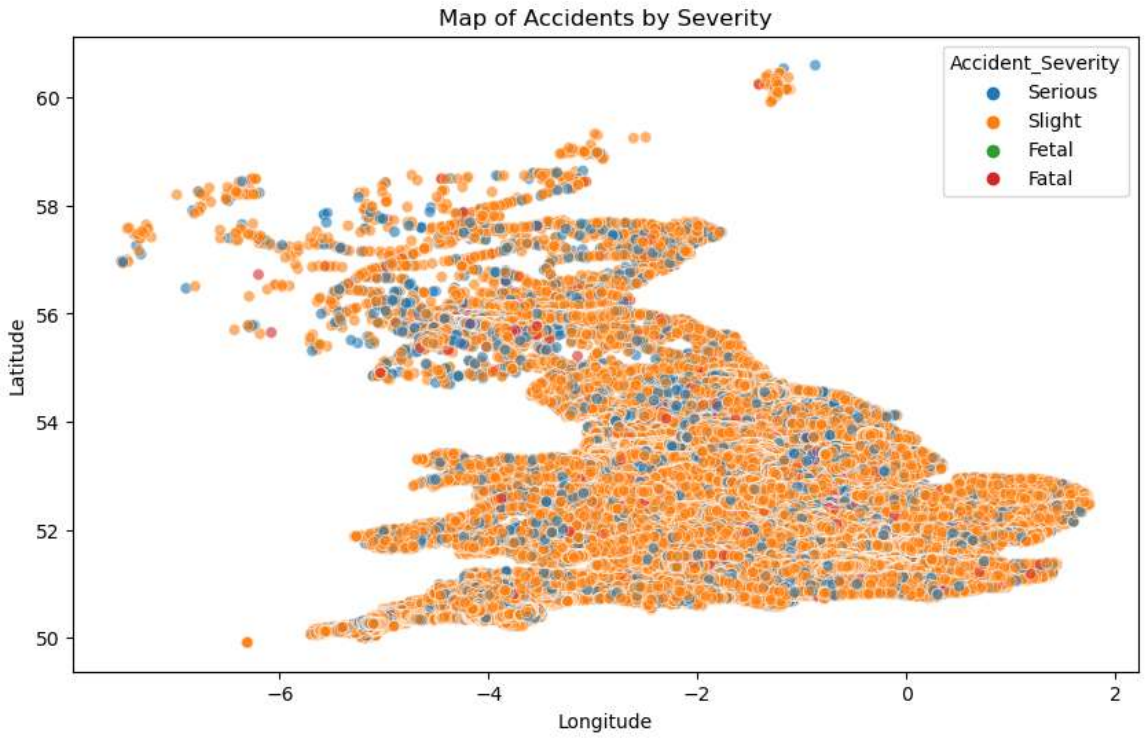
Severity counts

```
sns.countplot(x='Accident_Severity', data=data)
plt.title('Distribution of Accident Severity')
plt.show()
```



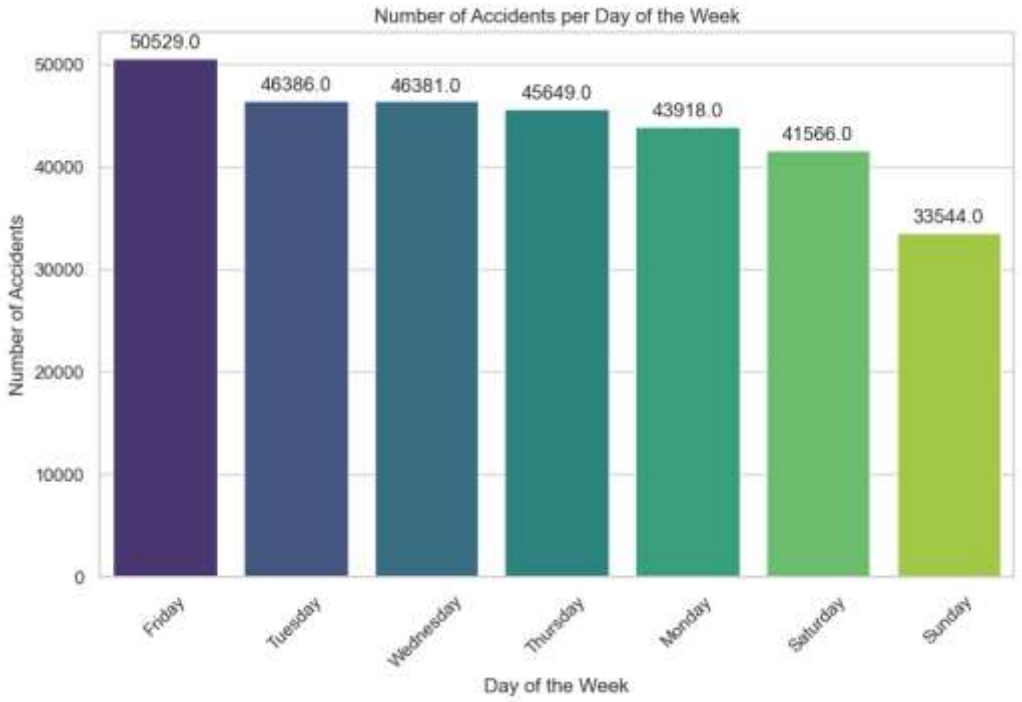
Map of accident by severity

```
plt.figure(figsize=(10, 6))
sns.scatterplot(x='Longitude', y='Latitude', data=data, hue='Accident_Severity', alpha=0.6)
plt.title('Map of Accidents by Severity')
plt.xlabel('Longitude')
plt.ylabel('Latitude')
plt.show()
```



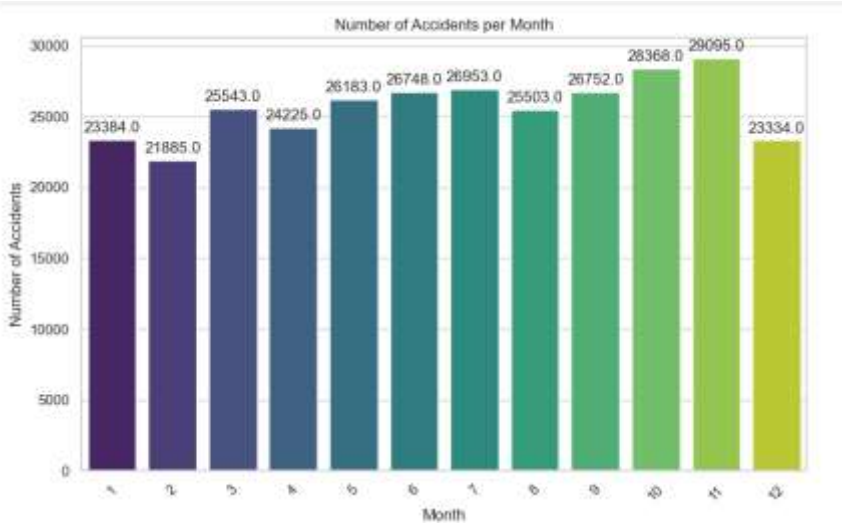
Accident count in each day of the week

```
accidents_by_day = data['Day_of_Week'].value_counts()

# Set the aesthetic style of the plots
sns.set(style="whitegrid")

# Create a bar plot
plt.figure(figsize=(10, 6))
accident_plot = sns.barplot(x=accidents_by_day.index, y=accidents_by_day.values, palette='viridis')

plt.title('Number of Accidents per Day of the Week')
plt.xlabel('Day of the Week')
plt.ylabel('Number of Accidents')
plt.xticks(rotation=45)  # Rotate labels to fit them better

# Optionally add labels on the bars
for p in accident_plot.patches:
    accident_plot.annotate(format(p.get_height(), '.1f'),
                           (p.get_x() + p.get_width() / 2., p.get_height()),
                           ha = 'center', va = 'center',
                           xytext = (0, 9),
                           textcoords = 'offset points')

plt.show()
```

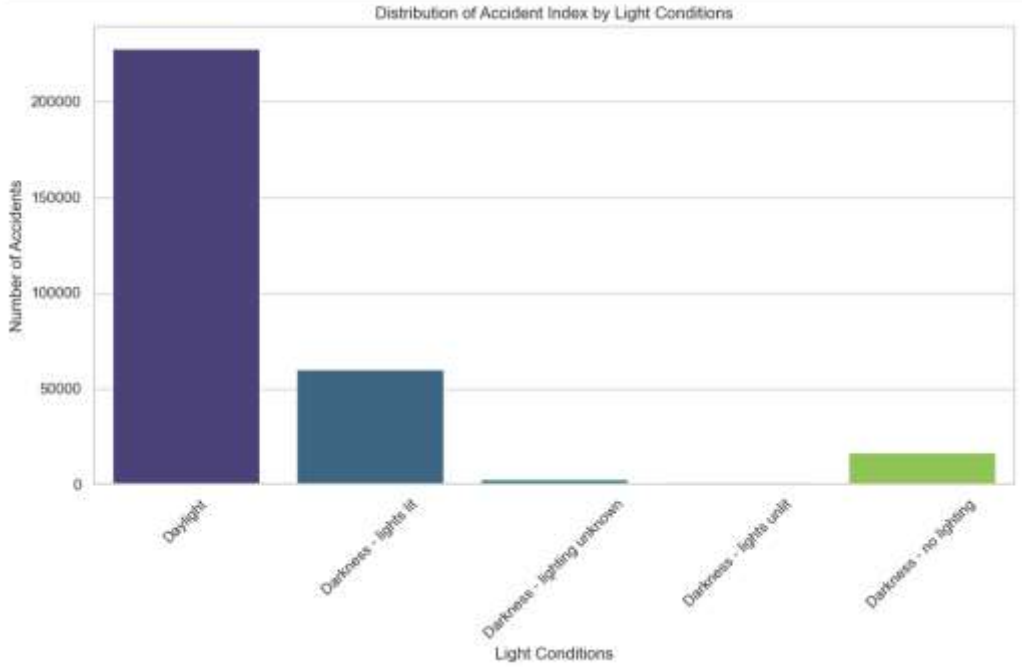


Accident counts in each month

```
# Convert the 'Accident Date' column to datetime
data['Accident Date'] = pd.to_datetime(data['Accident Date'])

# Extract the month from the 'Accident Date' column
data['Month'] = data['Accident Date'].dt.month

# Count the number of accidents for each month
accidents_by_month = data['Month'].value_counts().sort_index()

# Set the aesthetic style of the plots
sns.set(style="whitegrid")

# Create a bar plot
plt.figure(figsize=(10, 6))
accident_plot = sns.barplot(x=accidents_by_month.index, y=accidents_by_month.values, palette='viridis')

plt.title('Number of Accidents per Month')
plt.xlabel('Month')
plt.ylabel('Number of Accidents')
plt.xticks(rotation=45)  # Rotate labels to fit them better

# Optionally add labels on the bars
for p in accident_plot.patches:
    accident_plot.annotate(format(p.get_height(), '.1f'),
                           (p.get_x() + p.get_width() / 2., p.get_height()),
                           ha = 'center', va = 'center',
                           xytext = (0, 9),
                           textcoords = 'offset points')

plt.show()
```



```
# Set the aesthetic style of the plots
sns.set(style="whitegrid")

# Create a count plot of Accident Index with respect to Light Conditions
plt.figure(figsize=(12, 6))
sns.countplot(x='Light_Conditions', data=data, palette='viridis')

plt.title('Distribution of Accident Index by Light Conditions')
plt.xlabel('Light Conditions')
plt.ylabel('Number of Accidents')

plt.xticks(rotation=45)  # Rotate x-axis labels for better readability

plt.show()
```



Analysis

- With the analysis, it is found that Friday has registered more number of accidents than Sunday or Monday.So,weekend has more number of accidents.
- November month has registered more number of accidents
- Most of the accidents are not so severe and only a very few are fatal.
- Its interesting that most of the accidents occur at daylight and only few occurred in darkness with no light.

THANK YOU!!