Accidents in France from 2005 to 2016

DESCRIPTION: My dataset includes accident data categorized by year, month, and day, specifying details such as road and light conditions, the number of fatalities and injuries, atmospheric conditions, and more.

OBJECTIVE: We are working on information about car accidents that happened in France between 2005 and 2016. Our goal is to discover valuable insights that can make the roads safer, help design better roads, and reduce the harm caused by accidents.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import warnings
warnings.filterwarnings('ignore')
```

```
#reading the dataset
```

```
df = pd.read_excel("/content/drive/MyDrive/caracteristics+_caracteristics+ (Multiple Connections)_caracteristics+_caracteristics.xlsx")
df.head()
```

	Adr	Gps	Agg	An	An Nais	Atm	Catr	Catu	Circ	Col	...	Mois	Nbv	Num Acc	Num Acc1	Place
0	NaN	NaN	1	9	1982.0	1.0	1.0	1	NaN	3.0	...	1	NaN	200900019965	200900019965	1.0
1	RD 153	NaN	1	9	1966.0	1.0	3.0	1	NaN	3.0	...	8	NaN	200900034800	200900034800	1.0
2	Chemin de Sathonay	NaN	1	9	1953.0	1.0	4.0	1	NaN	6.0	...	5	NaN	200900031213	200900031213	1.0
3	NaN	NaN	1	9	1929.0	1.0	3.0	1	NaN	3.0	...	4	2.0	200900029987	200900029987	1.0
4	NaN	NaN	1	9	1989.0	1.0	3.0	1	NaN	6.0		4	NaN	200900029988	200900029988	1.0

```
#number of rows is 839985 and columns is 19
```

```
df.shape
```

```
(839985, 27)
```

```
df.describe()#statistical analysis
```

	Agg	An	An Nais	Atm	Catr	Catu	Circ	
count	839985.000000	839985.000000	838934.000000	839930.000000	839984.000000	839985.000000	839187.000000	8399
mean	1.685924	10.011129	1971.805590	1.547116	3.418247	1.064531	1.855246	
std	0.464147	3.458059	16.892703	1.587668	1.207917	0.336850	0.720949	

```
#checking the column wise null values count

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 839985 entries, 0 to 839984
Data columns (total 27 columns):
 #   Column    Non-Null Count   Dtype
---  ------    --------------   -----
 0   Adr       699403 non-null  object
 1   Gps       366226 non-null  object
 2   Agg       839985 non-null  int64
 3   An        839985 non-null  int64
 4   An Nais   838934 non-null  float64
 5   Atm       839930 non-null  float64
 6   Catr      839984 non-null  float64
 7   Catu      839985 non-null  int64
 8   Circ      839187 non-null  float64
 9   Col       839974 non-null  float64
 10  Com       839983 non-null  float64
 11  Etatp     839242 non-null  float64
 12  Grav      839985 non-null  int64
 13  Hrmn      839985 non-null  int64
 14  Infra     838707 non-null  float64
 15  Jour      839985 non-null  int64
 16  Lum       839985 non-null  int64
 17  Mois      839985 non-null  int64
 18  Nbv       838195 non-null  float64
 19  Num Acc   839985 non-null  int64
 20  Num Acc1  839985 non-null  int64
 21  Place     826818 non-null  float64
 22  Prof      838924 non-null  float64
 23  Secu      834421 non-null  float64
 24  Sexe      839985 non-null  int64
 25  Situ      838983 non-null  float64
 26  Surf      838968 non-null  float64
dtypes: float64(14), int64(11), object(2)
memory usage: 173.0+ MB
```

```
#Checking null values

df.isnull().sum()
```

```
Adr         140582
Gps         473759
Agg              0
An               0
An Nais       1051
Atm             55
Catr             1
Catu             0
Circ           798
Col             11
Com              2
Etatp          743
Grav             0
Hrmn             0
Infra         1278
Jour             0
Lum              0
Mois             0
Nbv           1790
Num Acc          0
Num Acc1         0
Place        13167
Prof          1061
Secu          5564
Sexe             0
Situ          1002
Surf          1017
dtype: int64
```

```
#droping null values

a=df.dropna()
```

```
#after droping null values we have 219771 rows and 27 columns
a.shape
```

```
(219771, 27)
```

```
#Checking null values are removed or not

a.isnull().sum()
```

```
Adr         0
Gps         0
Agg         0
An          0
An Nais     0
Atm         0
Catr        0
Catu        0
Circ        0
Col         0
Com         0
Etatp       0
Grav        0
Hrmn        0
Infra       0
Jour        0
Lum         0
Mois        0
Nbv         0
Num Acc     0
Num Acc1    0
Place       0
Prof        0
Secu        0
Sexe        0
Situ        0
Surf        0
dtype: int64
```

```
a['An'] = a['An'] + 2000
a['An'].head()
```

```
103    2006
109    2006
111    2005
113    2011
114    2005
Name: An, dtype: int64
```

```
a['An']
```

```
103       2006
109       2006
111       2005
113       2011
114       2005
          ...
838599    2005
838601    2005
838602    2005
838603    2005
838605    2005
Name: An, Length: 219771, dtype: int64
```

Feature Engineering

```
a['Years']=a['An']-a['An Nais']
a['Years']
```

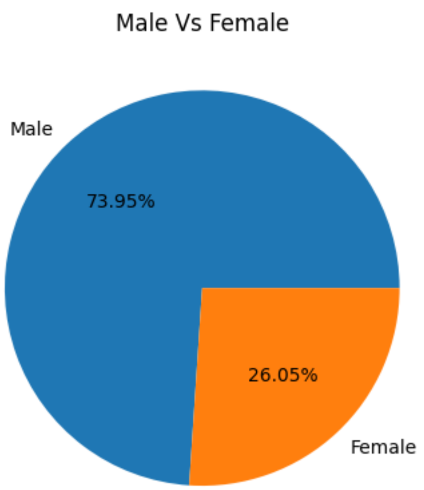
```
103       21.0
109       39.0
111       33.0
113       48.0
114       25.0
          ...
838599    21.0
838601    65.0
838602    19.0
838603    17.0
838605    10.0
Name: Years, Length: 219771, dtype: float64
```

```
def categorize_status(grav):
    if grav == 2:
```

```
        return "Dead"
    else:
        return "Alive"
```

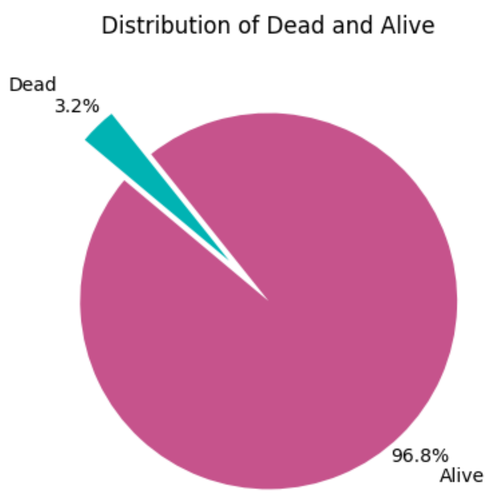
```
a['Status'] = a['Grav'].apply(categorize_status)
```

```
b=a['Sexe'].value_counts().reset_index()
c=['Male','Female']
plt.pie(b['Sexe'],labels=c,autopct='%1.2f%%')
plt.title("Male Vs Female")
plt.show()
```



The chart shows that more accidents happen to males than to females.

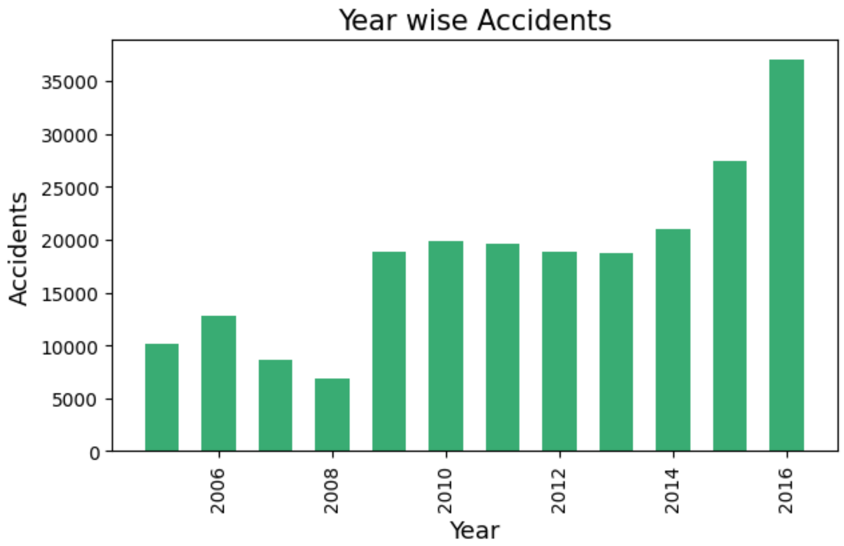
```
status_counts = a['Status'].value_counts()
plt.figure(figsize=(4,5))
color=['#c6538c','#00b3b3']
exp=(0.0,0.3)
plt.pie(status_counts, labels=status_counts.index, colors=color,autopct='%1.1f%%',explode=exp,
startangle=140,labeldistance=1.30, pctdistance=1.15)
plt.title("Distribution of Dead and Alive")
plt.axis('equal')
plt.show()
```



The graph illustrates a 3.2% mortality rate and a 96.8% survival rate, emphasizing a low incidence of deaths and a high proportion of individuals who have survived.

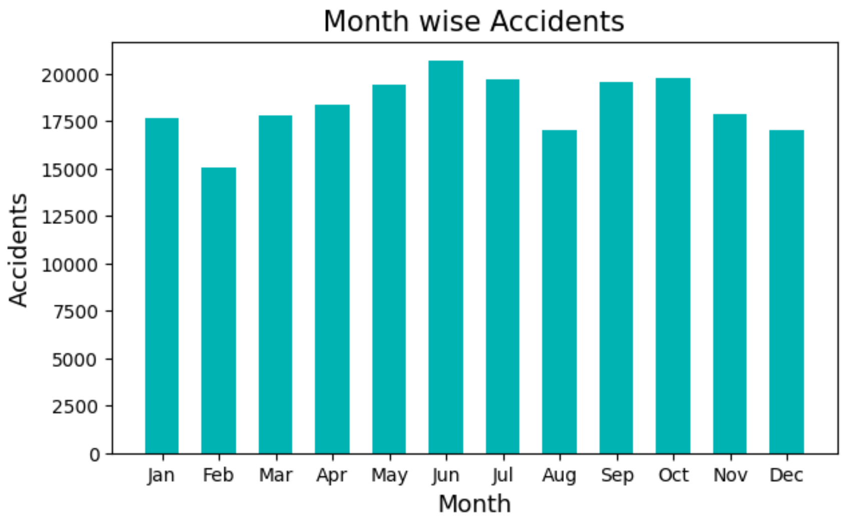
```
b=a['An'].value_counts().sort_index()
c=list(sorted(a['An'].unique()))
plt.figure(figsize=(7,4))
```

```
plt.bar(c,b,color='#39ac73',width=0.6)
plt.xlabel("Year",fontsize=13)
#plt.xticks(rotation='vertical')
plt.ylabel("Accidents",fontsize=13)
plt.title("Year wise Accidents",fontsize=15)
plt.show()
```



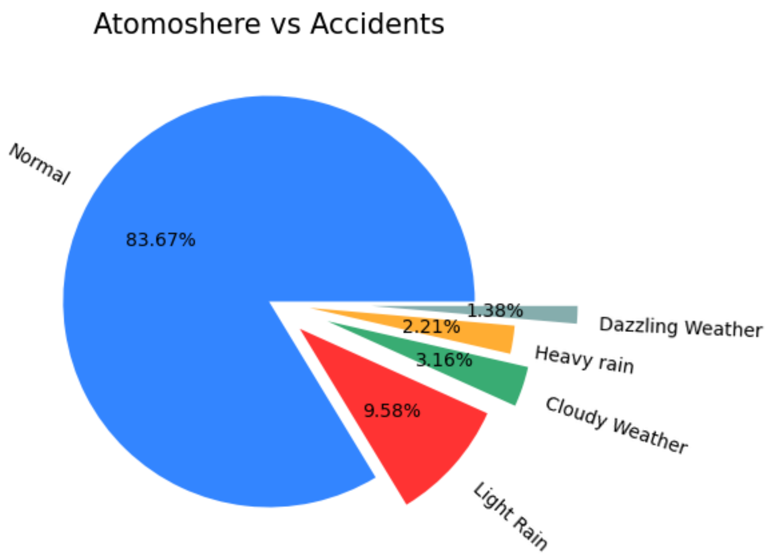
The graph displays fluctuations from 2005 to 2013, with some years show increases while others show decreases, but in the years 2014 to 2016, the accident rates increased.

```
b=a['Mois'].value_counts().sort_index()
#c=list(sorted(a['Mois'].unique()))
c=['Jan','Feb','Mar','Apr','May','Jun','Jul','Aug','Sep','Oct','Nov','Dec']
plt.figure(figsize=(7,4))
plt.bar(c,b,color='#00b3b3',width=0.6)
plt.xlabel("Month",fontsize=13)
plt.ylabel("Accidents",fontsize=13)
plt.title("Month wise Accidents",fontsize=15)
plt.show()
```



In the graph, it is evident that the month of June (Month 6) records the highest number of accidents compared to all other months.

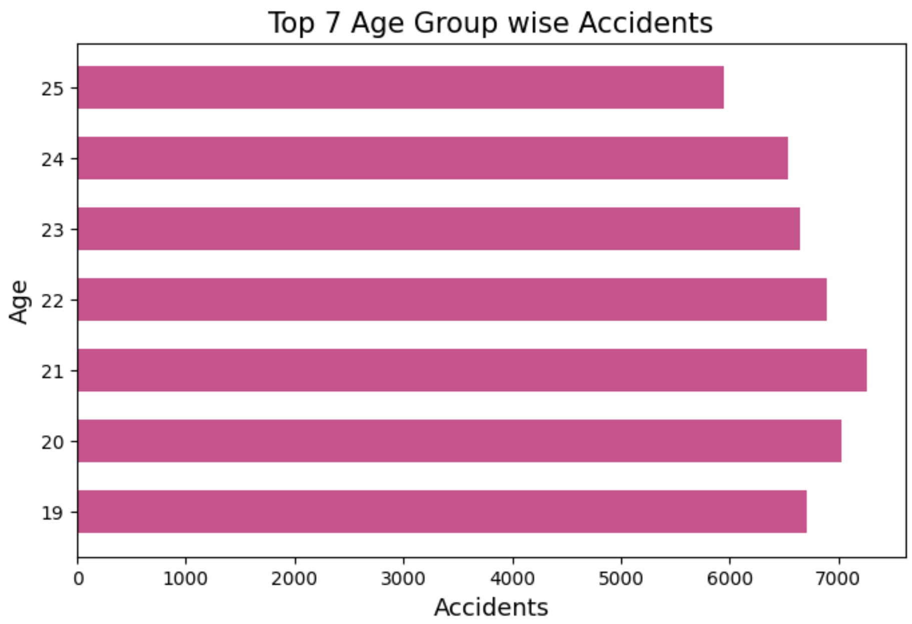
```
b=a.groupby('Atm')['An'].sum().reset_index()
c=b.sort_values(by='An',ascending=False)
d=c.head(5)
plt.figure(figsize=(7,5))
exp=(0,0.2,0.3,0.2,0.5)
c=['Normal','Light Rain','Cloudy Weather','Heavy rain','Dazzling Weather']
colors=['#3385ff','#ff3333','#39ac73','#ffad33','#85adad','#ff5500','#cccc00','#c6538c','#00b3b3']
plt.pie(d['An'],labels=c,autopct="%1.2f%%",colors=colors,explode=exp,rotatelabels=True)
plt.title("Atomoshere vs Accidents",fontsize=15)
plt.show()
```



According to the pie chart, accidents are most frequent in 'normal' atmospheric conditions when compared to other weather conditions.

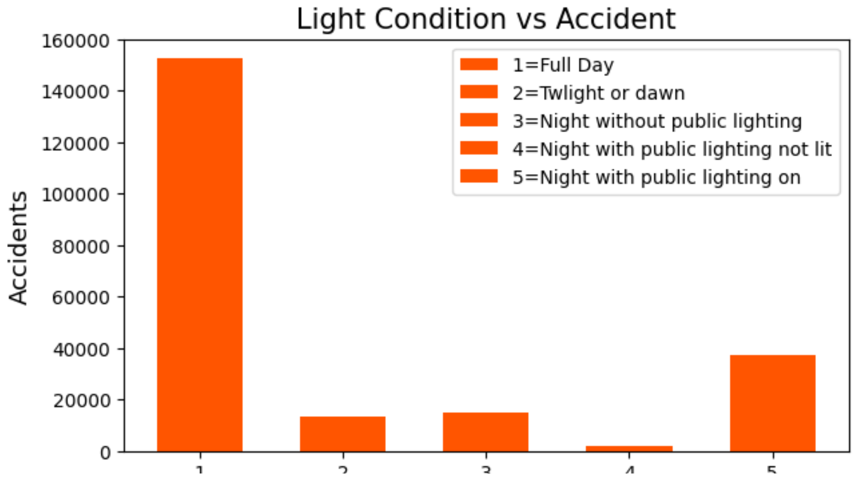
```
b=a['Years'].value_counts().reset_index()
c=b.sort_values(by='Years',ascending=False)
e=int(input("for how many age group you want to see the Accidents :"))
d=c.head(e)
plt.figure(figsize=(8,5))
plt.barh(d['index'],d['Years'],color='#c6538c',height=0.6)
plt.xlabel("Accidents",fontsize=13)
plt.ylabel("Age",fontsize=13)
plt.title(f"Top {e} Age Group wise Accidents",fontsize=15)
plt.show()
```

```
for how many age group you want to see the Accidents :7
```



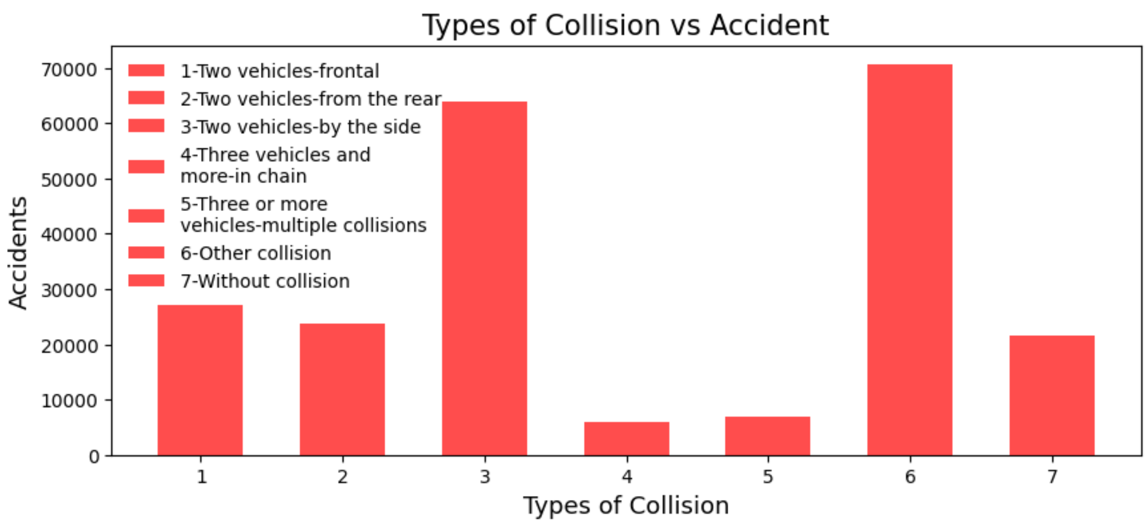
The graph displays age-group-wise accident data, with the age group '21' exhibiting the highest accident rate, followed by the other age groups.

```
b=a['Lum'].value_counts().reset_index()
c=b.sort_values(by='index')
d=['1=Full Day','2=Twlight or dawn','3=Night without public lighting','4=Night with public lighting not lit',
'5=Night with public lighting on']
plt.figure(figsize=(7,4))
plt.bar(c['index'],c['Lum'],color='#ff5500',width=0.6,label=d)
plt.legend()
plt.xlabel("Light Condition",fontsize=13)
plt.ylabel("Accidents",fontsize=13)
plt.title("Light Condition vs Accident",fontsize=15)
plt.show()
```



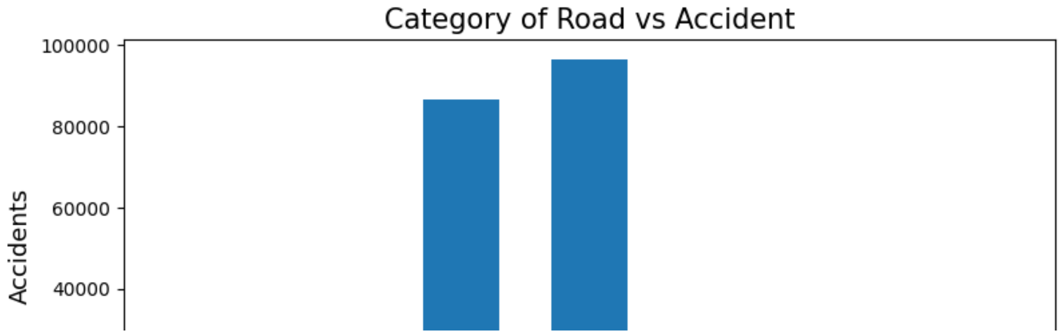
The graph illustrates that accidents occur more frequently in 'full daylight' conditions compared to other lighting conditions because more people are on the road during the day, leading to higher traffic volume as compared to other light conditions.

```
b=a['Col'].value_counts().reset_index()
c=b.sort_values(by='index')
d=['1-Two vehicles-frontal','2-Two vehicles-from the rear','3-Two vehicles-by the side','4-Three vehicles and\nmore-in chain',
'5-Three or more\nvehicles-multiple collisions','6-Other collision','7-Without collision']
plt.figure(figsize=(10,4))
plt.bar(c['index'],c['Col'],color='#ff4d4d',width=0.6,label=d)
plt.legend(loc='upper left',framealpha=0.0)
plt.xlabel("Types of Collision",fontsize=13)
plt.ylabel("Accidents",fontsize=13)
plt.title("Types of Collision vs Accident",fontsize=15)
plt.show()
```



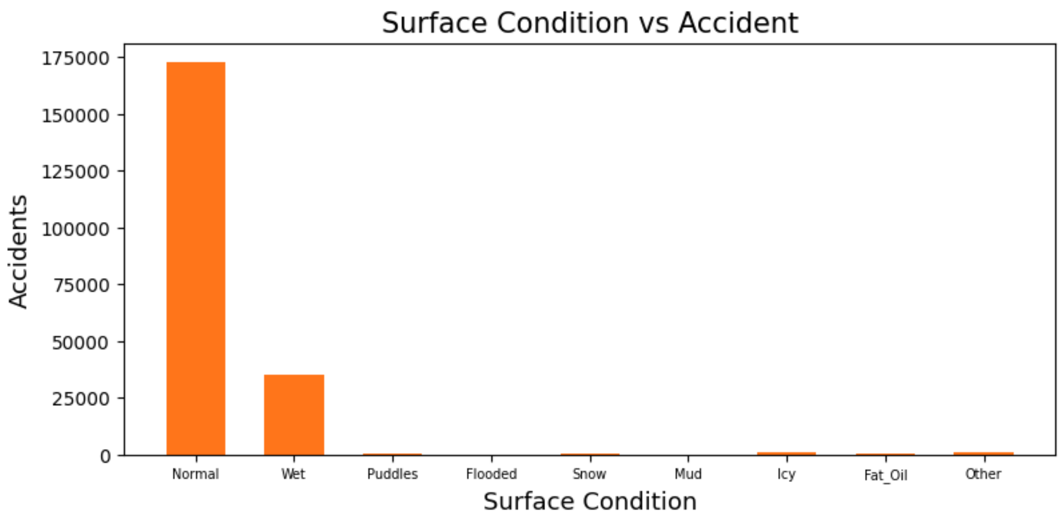
The graph represents that collisions involving two vehicles side by side, indicating that accidents are more likely to occur during overtaking.

```
b=a['Catr'].value_counts().reset_index()
c=b.sort_values(by='index')
d=['Highway','National Road','Department Road','Communal Way',
'Off Public Network','Parking lot open\nto public traffic','Other']
plt.figure(figsize=(9,4))
plt.bar(d,c['Catr'],width=0.6)
plt.xticks(fontsize=7)
plt.xlabel("Category of Road",fontsize=13)
plt.ylabel("Accidents",fontsize=13)
plt.title("Category of Road vs Accident",fontsize=15)
plt.show()
```



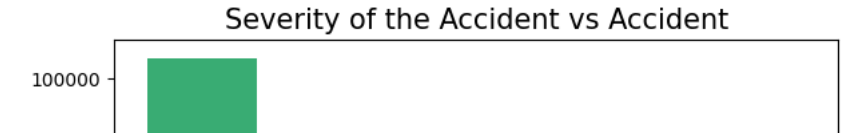
The graph illustrates that 'communal way' roads experience a significantly higher rate of accidents compared to other road categories.

```
b=a['Surf'].value_counts().reset_index()
e=b[b['index']!=0]
c=e.sort_values(by='index')
d=['Normal','Wet','Puddles','Flooded','Snow','Mud','Icy','Fat_Oil','Other']
plt.figure(figsize=(9,4))
plt.bar(d,c['Surf'],width=0.6,color='#ff751a')
plt.xticks(fontsize=7)
#plt.yticks(range(0,200000,10000))
plt.xlabel("Surface Condition",fontsize=13)
plt.ylabel("Accidents",fontsize=13)
plt.title("Surface Condition vs Accident",fontsize=15)
plt.show()
```



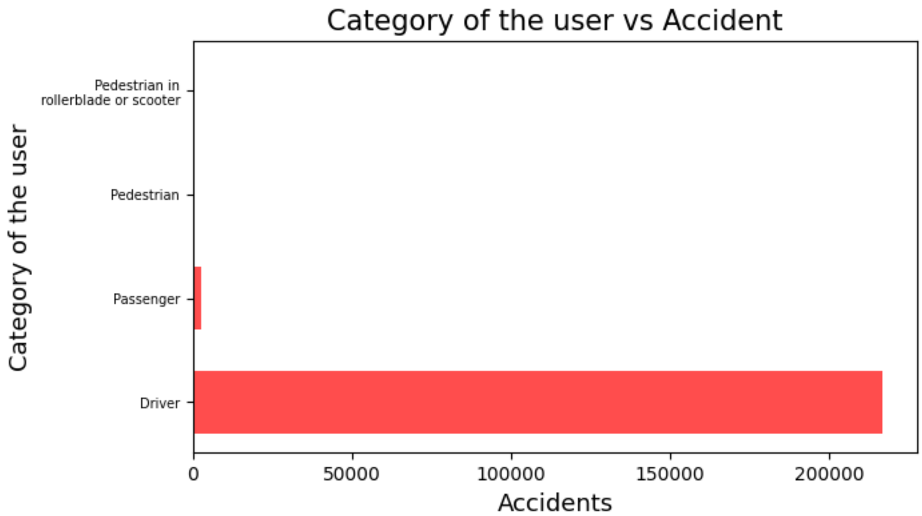
The graph illustrates that accidents are more frequent on roads with 'normal' surface conditions compared to other surface conditions.

```
b=a['Grav'].value_counts().reset_index()
c=b.sort_values(by='index')
d=['Unscathed','Killed','Hospitalized Wounded','Light Injury']
plt.figure(figsize=(7,4))
plt.bar(d,c['Grav'],width=0.6,color='#39ac73')
plt.xticks(fontsize=7)
plt.xlabel("Severity of the Accident",fontsize=13)
plt.ylabel("Accidents",fontsize=13)
plt.title("Severity of the Accident vs Accident",fontsize=15)
plt.show()
```



The graph shows that most accidents result in people being unharmed, while very few accidents lead to fatalities.

```
b=a['Catu'].value_counts().reset_index()
d=['Driver','Passenger','Pedestrian',' Pedestrian in\nrollerblade or scooter']
plt.figure(figsize=(7,4))
plt.barh(d,b['Catu'],height=0.6,color='#ff4d4d')
plt.yticks(fontsize=7)
plt.ylabel("Category of the user",fontsize=13)
plt.xlabel("Accidents",fontsize=13)
plt.title("Category of the user vs Accident",fontsize=15)
plt.show()
```



This graph shows the accident rate by user category, with drivers having a higher accident rate compared to others

```
s=a[a['Secu']==11  ].groupby('Status')['Secu'].count().reset_index()
t=a[a['Secu']==12].groupby('Status')['Secu'].count().reset_index()
u=a[a['Secu']==21].groupby('Status')['Secu'].count().reset_index()
v=a[a['Secu']==22].groupby('Status')['Secu'].count().reset_index()



secu_values = s.loc[s['Status'] == 'Alive', 'Secu'].reset_index()
secu_values1 = t.loc[s['Status'] == 'Alive', 'Secu'].reset_index()
secu_values2 = u.loc[s['Status'] == 'Alive', 'Secu'].reset_index()
secu_values3 = v.loc[s['Status'] == 'Alive', 'Secu'].reset_index()


b=a['Secu'].value_counts().reset_index()
c=b.loc[b['index']==11,'Secu'].reset_index()
d=(secu_values['Secu']/c['Secu'])*100
d
```

```
0    98.191379
Name: Secu, dtype: float64
```

```
e=b.loc[b['index']==12,'Secu'].reset_index()
f=(secu_values1['Secu']/e['Secu'])*100
print(f)
```

```
0    73.713978
Name: Secu, dtype: float64
```

```
g=b.loc[b['index']==21,'Secu'].reset_index()
h=(secu_values2['Secu']/g['Secu'])*100
print(h)
```

```
0    95.447614
Name: Secu, dtype: float64
```

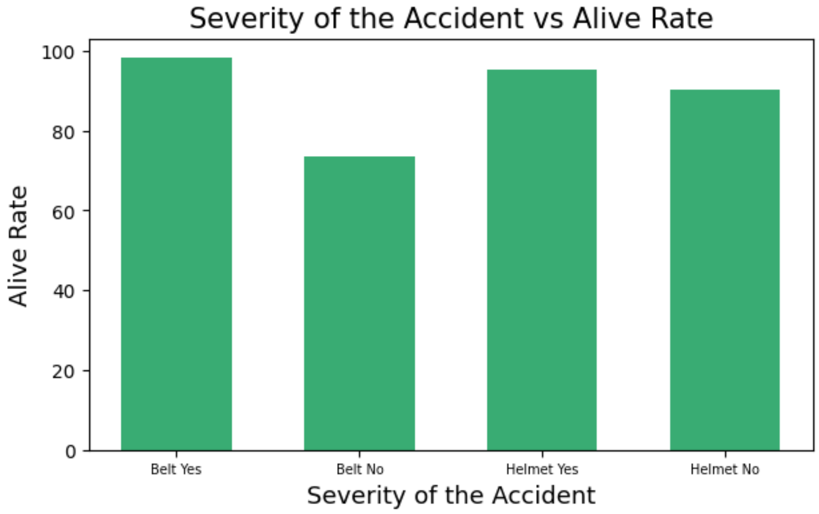
```
i=b.loc[b['index']==22,'Secu'].reset_index()
j=(secu_values3['Secu']/i['Secu'])*100
print(j)
```

```
0    90.271377
Name: Secu, dtype: float64
```

```
combined_df = pd.concat([d,f,h,j], ignore_index=True).reset_index()
combined_df
```

	index	Secu
0	0	98.191379
1	1	73.713978
2	2	95.447614
3	3	90.271377

```
safety=['Belt Yes','Belt No','Helmet Yes','Helmet No']
plt.figure(figsize=(7,4))
plt.bar(safety,combined_df['Secu'],width=0.6,color='#39ac73')
plt.xticks(fontsize=7)
plt.xlabel("Severity of the Accident",fontsize=13)
plt.ylabel("Alive Rate",fontsize=13)
plt.title("Severity of the Accident vs Alive Rate",fontsize=15)
plt.show()
```



The graph illustrates that using seatbelts and helmets ('yes') is associated with higher survival rates compared to not using them ('no'). This highlights the critical role of safety equipment in preventing fatalities

CONCLUSION:

- **Educate Male Drivers:** Targeted education for males to drive safer.
- **Improve Healthcare:** Keep healthcare effective for lower mortality rates.
- **Tackle Fluctuations:** Address years with more accidents (2014-2016).
- **June Safety:** Special plans for June's high accident rates.
- **All-Weather Prep:** Prepare for different weather conditions.
- **Age 21 Education:** Focus on safety programs for age 21 drivers.
- **Daytime Caution:** Promote safe driving during the day.
- **Overtaking Safety:** Teach safe overtaking and road manners.
- **Better Communal Roads:** Upgrade 'communal way' roads for safety.
- **Reduce Injuries:** Keep working on reducing injuries and deaths.
- **Driver Education:** Improve education for all drivers.
- **Safety Gear:** Encourage seatbelts and helmets ('yes') for safety.