'''This problem statement and objective consists of two parts: Part 1 and Part 2. Part 1: Autonomous vehicles (AV) and intelligent transport systems (ITS) are the future of road transport. Automatic detection of vehicles on the road in real-time helps AV technology and makes ITS more intelligent in terms of vehicle tracking, vehicle counting, and road incident response'''

'''Problem statement 2: Tesla, Inc. is an American multinational automotive and artificial intelligence company. In October 2020, Tesla started a full self-driving capability beta program in the United States. Tesla has over 100k people in this program.'''

'''As the second part of this project, you need to analyze the usage of autopilot and its effect on road safety. Steps to be followed as below

1. Preliminary data inspection and cleaning

a. Perform preliminary data inspection, checking for data types, missing values, and duplicates . Remove any columns that might not be relevant for the analy

1. Exploratory Data Analysis

a. Perform an in-depth exploratory data analysis on the number of events by date, per year, and per day for each state and country

b. Analyze the different aspects of the death events. For example: - What is the number of victims (deaths) in each accident? - How many times did tesla drivers die? - What is the proportion of events in which one or more occupants died? - What is the distribution of events in which the vehicle hit a cyclist or a pedestrian? - How many times did the accident involve the death of an occupant or driver of a Tesla along with a cyclist or pedestrian? - What is the frequency of Tesla colliding with other vehicles?

1. Data Science
   a. Study the event distribution across models
   b. Check the distribution of verified Tesla autopilot deaths'''''

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import nltk
nltk.download('punkt')
nltk.download('averaged_perceptron_tagger')
nltk.download('wordnet')
nltk.download('omw-1.4')
from nltk import word_tokenize
from nltk.stem import WordNetLemmatizer
from nltk.corpus import wordnet
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.cluster import KMeans
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\HP\AppData\Roaming\nltk_data...
```

```
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     C:\Users\HP\AppData\Roaming\nltk_data...
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\HP\AppData\Roaming\nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package omw-1.4 to
[nltk_data]     C:\Users\HP\AppData\Roaming\nltk_data...
[nltk_data]   Package omw-1.4 is already up-to-date!
```

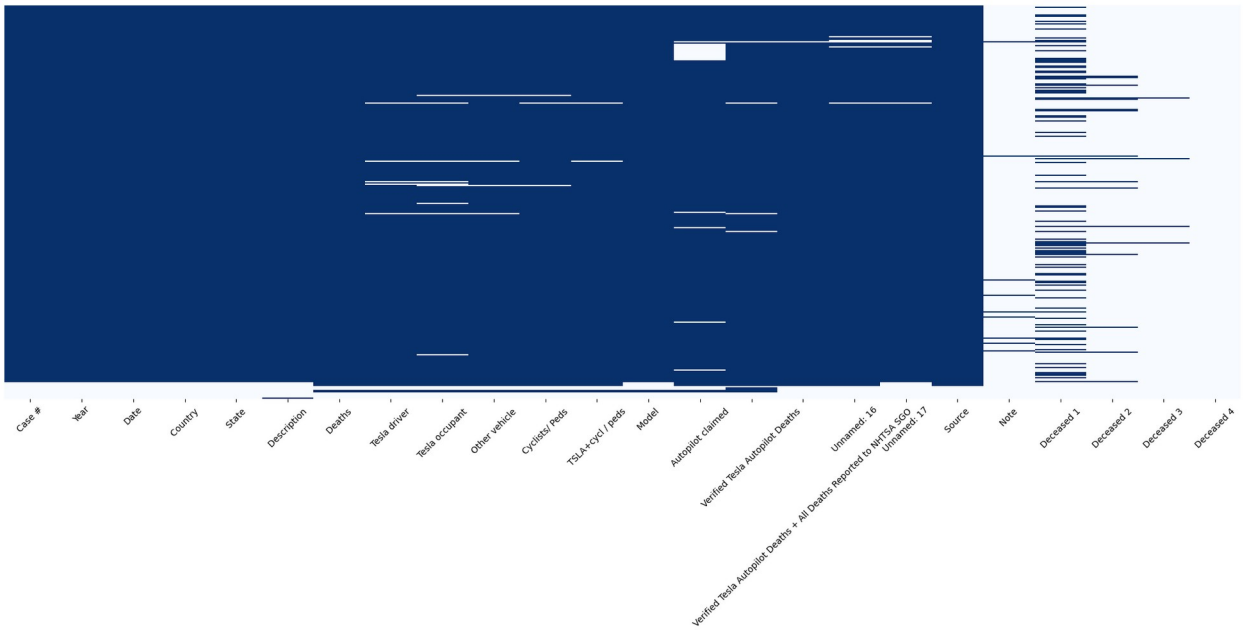
```
data=pd.read_csv("C:\Mukti\AI & ML Training\Capstone Project\
Autonomous Driving\Datasets\Part 2\Tesla - Deaths.csv")
data.head()
```

```
   Case #    Year        Date  Country   State   \
0   294.0  2022.0   1/17/2023      USA       CA
1   293.0  2022.0    1/7/2023   Canada        -
2   292.0  2022.0    1/7/2023      USA       WA
3   291.0  2022.0  12/22/2022      USA       GA
4   290.0  2022.0  12/19/2022   Canada        -

                         Description    Deaths   Tesla driver   \
0    Tesla crashes into back of semi        1.0             1
1                      Tesla crashes        1.0             1
2   Tesla hits pole, catches on fire        1.0             -
3            Tesla crashes and burns        1.0             1
4      Tesla crashes into storefront        1.0             -

   Tesla occupant   Other vehicle   ...  Verified Tesla Autopilot
Deaths   \
0               -               -   ...
-
1               -               -   ...
-
2               1               -   ...
-
3               -               -   ...
-
4               -               -   ...
-

   Verified Tesla Autopilot Deaths + All Deaths Reported to NHTSA SGO
\
0                                                   -

1                                                   -

2                                                   -
```

```
3                                                   -

4                                                   -


                                              Unnamed: 16  \
0   https://web.archive.org/web/20221222203930/ht...
1   https://web.archive.org/web/20221222203930/ht...
2   https://web.archive.org/web/20221222203930/ht...
3   https://web.archive.org/web/20221222203930/ht...
4   https://web.archive.org/web/20221223203725/ht...

                                              Unnamed: 17  \
0   https://web.archive.org/web/20221222203930/ht...
1   https://web.archive.org/web/20221222203930/ht...
2   https://web.archive.org/web/20221222203930/ht...
3   https://web.archive.org/web/20221222203930/ht...
4   https://web.archive.org/web/20221223203725/ht...

                                                 Source   Note   \
0   https://web.archive.org/web/20230118162813/ht...    NaN
1   https://web.archive.org/web/20230109041434/ht...    NaN
2   https://web.archive.org/web/20230107232745/ht...    NaN
3   https://web.archive.org/web/20221222203930/ht...    NaN
4   https://web.archive.org/web/20221223203725/ht...    NaN

          Deceased 1   Deceased 2   Deceased 3   Deceased 4
0                NaN          NaN          NaN          NaN
1   Taren Singh Lal           NaN          NaN          NaN
2                NaN          NaN          NaN          NaN
3                NaN          NaN          NaN          NaN
4                NaN          NaN          NaN          NaN

[5 rows x 24 columns]
```

Perform preliminary data inspection checking for data types. missing values, duplicates

```
missing_values=data.isna()
plt.figure(figsize=(25,8))
sns.heatmap(missing_values,cmap="Blues_r",cbar=False)
plt.tick_params(left=False,labelleft=False)
plt.xticks(rotation=45)
plt.show()
```



Remove any columns which may not be relevant for the analysis

```
data.isna().sum()
```

```
Case #
13
Year
13
Date
13
 Country
13
 State
13
 Description
12
 Deaths
8
 Tesla driver
13
 Tesla occupant
17
 Other vehicle
12
 Cyclists/ Peds
11
 TSLA+cycl / peds
10
 Model
11
 Autopilot claimed
```

```
26
 Verified Tesla Autopilot Deaths
10
 Verified Tesla Autopilot Deaths + All Deaths Reported to NHTSA SGO
11
Unnamed: 16
15
Unnamed: 17
18
 Source
10
 Note
298
 Deceased 1
220
 Deceased 2
290
 Deceased 3
303
 Deceased 4
307
dtype: int64
```

```
data.columns
```

```
Index(['Case #', 'Year', 'Date', ' Country ', ' State ', ' Description
',
       ' Deaths ', ' Tesla driver ', ' Tesla occupant ', ' Other
vehicle ',
       ' Cyclists/ Peds ', ' TSLA+cycl / peds ', ' Model ',
       ' Autopilot claimed ', ' Verified Tesla Autopilot Deaths ',
       ' Verified Tesla Autopilot Deaths + All Deaths Reported to
NHTSA SGO ',
       'Unnamed: 16', 'Unnamed: 17', ' Source ', ' Note ', ' Deceased
1 ',
       ' Deceased 2 ', ' Deceased 3 ', ' Deceased 4 '],
      dtype='object')
```

Observations

#####1. Unnamed: 16 & Unnamed: 17 columns contain weblinks which are not useful

#####2. Source: Contains weblink

#####3. Note: Contains additional info

#####4. Deceased 1,2,3&4 Contains name of deceased which are irrelevant for the analysis

#####5. Case # not relevant

#####6. Year can be derived from date. Hence Year column is duplicate information

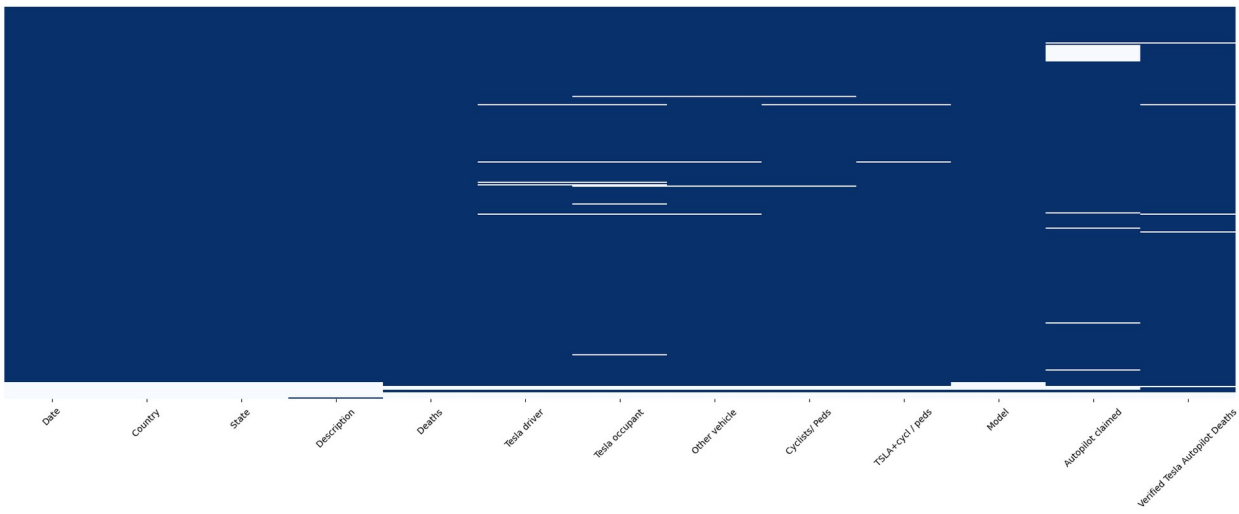
Hence dropping all above columns which are not relevant for this analysis

```
drop_columns=['Case #', 'Year',' Source ',' Deceased 1 ',' Deceased 2
', ' Deceased 3 ', ' Deceased 4 ','Unnamed: 16', 'Unnamed: 17',' Note
']
data.drop(columns=drop_columns,inplace=True)

data[' Verified Tesla Autopilot Deaths + All Deaths Reported to NHTSA
SGO '].value_counts(dropna=False)
```

```
 -       269
  1       21
NaN       11
  2        3
  3        1
  24       1
  27       1
Name:  Verified Tesla Autopilot Deaths + All Deaths Reported to NHTSA
SGO , dtype: int64
```

On closer observation, it seems ' Verified Tesla Autopilot Deaths + All Deaths Reported to NHTSA SGO 'contains mostly '-'

```
data.drop(columns=' Verified Tesla Autopilot Deaths + All Deaths
Reported to NHTSA SGO ',inplace=True)

missing_values=data.isna()
plt.figure(figsize=(25,8))
sns.heatmap(missing_values,cmap="Blues_r",cbar=False)
plt.tick_params(left=False,labelleft=False)
plt.xticks(rotation=45)
plt.show()
```



We still have missing values so we will impute missing values may be in other forms also like in this data they are in "-"

```
data.head()
```

```
         Date  Country   State                            Description
Deaths   \
0   1/17/2023      USA       CA    Tesla crashes into back of semi
1.0
1    1/7/2023   Canada        -                        Tesla crashes
1.0
2    1/7/2023      USA       WA   Tesla hits pole, catches on fire
1.0
3  12/22/2022      USA       GA             Tesla crashes and burns
1.0
4  12/19/2022   Canada        -      Tesla crashes into storefront
1.0

   Tesla driver   Tesla occupant   Other vehicle   Cyclists/ Peds   \
0             1                -               -                -
1             1                -               -                -
2             -                1               -                -
3             1                -               -                -
4             -                -               -                1

   TSLA+cycl / peds   Model   Autopilot claimed   \
0                 1       -                   -
1                 1       -                   -
2                 1       -                   -
3                 1       -                   -
4                 1       -                   -

   Verified Tesla Autopilot Deaths
0                                -
1                                -
2                                -
3                                -
4                                -
```

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 307 entries, 0 to 306
Data columns (total 13 columns):
 #   Column                              Non-Null Count  Dtype
---  ------                              --------------  -----
 0   Date                                294 non-null    object
 1    Country                            294 non-null    object
 2    State                              294 non-null    object
 3    Description                        295 non-null    object
 4    Deaths                             299 non-null    float64
```

```
 5    Tesla driver                          294 non-null    object
 6    Tesla occupant                        290 non-null    object
 7    Other vehicle                         295 non-null    object
 8    Cyclists/ Peds                        296 non-null    object
 9    TSLA+cycl / peds                      297 non-null    object
 10   Model                                 296 non-null    object
 11   Autopilot claimed                     281 non-null    object
 12   Verified Tesla Autopilot Deaths       297 non-null    object
dtypes: float64(1), object(12)
memory usage: 31.3+ KB
```

```
data.columns
```

```
Index(['Date', ' Country ', ' State ', ' Description ', ' Deaths ',
       ' Tesla driver ', ' Tesla occupant ', ' Other vehicle ',
       ' Cyclists/ Peds ', ' TSLA+cycl / peds ', ' Model ',
       ' Autopilot claimed ', ' Verified Tesla Autopilot Deaths '],
      dtype='object')
```

```
cols = data.columns[5:]
for col in cols:
    if col != ' Model ':
        print(col)

        # Handle NaN values explicitly
        data[col] = data[col].fillna("-")

        # Convert columns to strings, if not already
        data[col] = data[col].astype(str)

        # Strip whitespace and replace '-' with '0'
        data[col] = data[col].str.strip()
        data[col] = data[col].str.replace("-", "0")

        # Convert to numeric using pd.to_numeric with error coercion
        data[col] = pd.to_numeric(data[col],
errors='coerce').fillna(0).astype(int)

        print(data[col].unique())
```

```
 Tesla driver
[   1    0   28   89  117    6 2014]
 Tesla occupant
[   0    1    3    2    7   41   48    5 2015]
 Other vehicle
[   0    1    2    3    4   29  101  130   16 2016]
 Cyclists/ Peds
[   0    1    2   20   26   46   11 2017]
 TSLA+cycl / peds
[   1    0    2    3    4   61  149  210   21 2018]
```

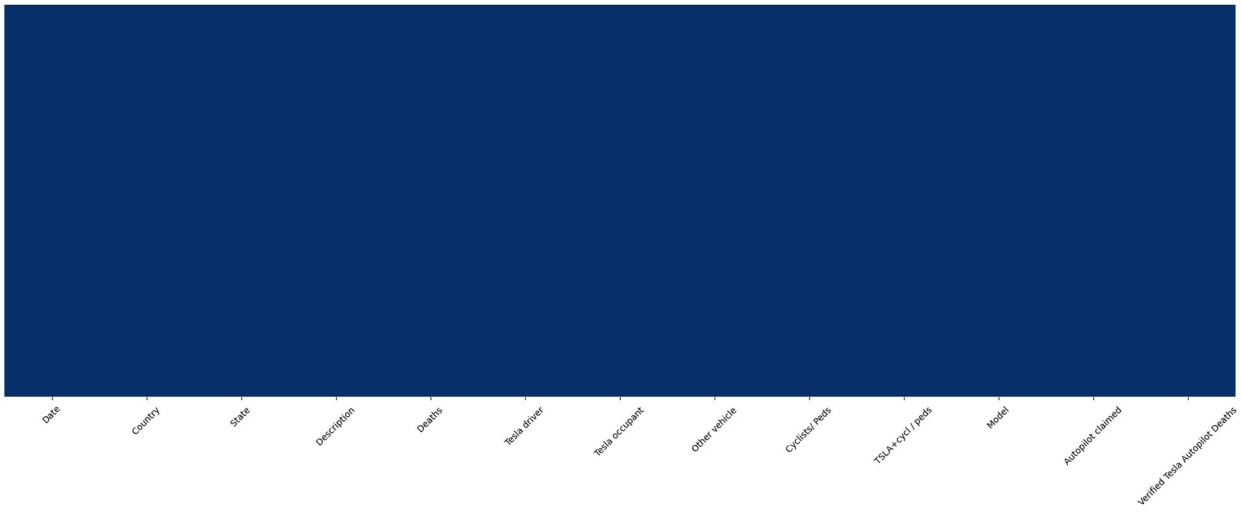
```
 Autopilot claimed
[   0    1    2    8   30   38   47 2020]
 Verified Tesla Autopilot Deaths
[   0    1    2    3   16   19  118 2022   75 2021]
```

```
data.isna().sum()
```

```
Date                                0
 Country                            0
 State                              0
 Description                        0
 Deaths                             0
 Tesla driver                       0
 Tesla occupant                     0
 Other vehicle                      0
 Cyclists/ Peds                     0
 TSLA+cycl / peds                   0
 Model                              0
 Autopilot claimed                  0
 Verified Tesla Autopilot Deaths    0
dtype: int64
```

```
data.dropna(inplace=True)
```

```
missing_values=data.isna()
plt.figure(figsize=(25,8))
sns.heatmap(missing_values,cmap="Blues_r",cbar=False)
plt.tick_params(left=False,labelleft=False)
plt.xticks(rotation=45)
plt.show()
```



Change the variable names in accordance with python norms

```
data.columns=data.columns.str.strip()
data.columns
```

```
Index(['Date', 'Country', 'State', 'Description', 'Deaths', 'Tesla
driver',
       'Tesla occupant', 'Other vehicle', 'Cyclists/ Peds', 'TSLA+cycl
/ peds',
       'Model', 'Autopilot claimed', 'Verified Tesla Autopilot
Deaths'],
      dtype='object')
```

```
data.columns=data.columns.str.replace("
","",regex=True).str.replace("[+/]","_",regex=True)
data.columns
```

```
Index(['Date', 'Country', 'State', 'Description', 'Deaths',
'Tesladriver',
       'Teslaoccupant', 'Othervehicle', 'Cyclists_Peds',
'TSLA_cycl_peds',
       'Model', 'Autopilotclaimed', 'VerifiedTeslaAutopilotDeaths'],
      dtype='object')
```

```
data.rename(columns={"Autopilotclaimed":"Claimed","VerifiedTeslaAutopi
lotDeaths":"VTAD","Teslaoccupant":"Tesla_Occupant",

"Othervehicle":"Other_Vehicle","Tesladriver":"Tesla_Driver"},inplace=T
rue)
```

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 294 entries, 0 to 293
Data columns (total 13 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Date            294 non-null    object
 1   Country         294 non-null    object
 2   State           294 non-null    object
 3   Description     294 non-null    object
 4   Deaths          294 non-null    float64
 5   Tesla_Driver    294 non-null    int32
 6   Tesla_Occupant  294 non-null    int32
 7   Other_Vehicle   294 non-null    int32
 8   Cyclists_Peds   294 non-null    int32
 9   TSLA_cycl_peds  294 non-null    int32
 10  Model           294 non-null    object
 11  Claimed         294 non-null    int32
 12  VTAD            294 non-null    int32
dtypes: float64(1), int32(7), object(5)
memory usage: 24.1+ KB
```

Exploratory Data Analysis

Perform an in-depth exploratory data analysis on number of events by date, per year, and per day for each state and country

Split the date into year, month and date

```
data.Date=pd.to_datetime(data.Date)
data.loc[:,"event_year"] = data.Date.dt.year
data.loc[:,"event_month"] = data.Date.dt.month
data.loc[:,"event_day"] = data.Date.dt.day
```

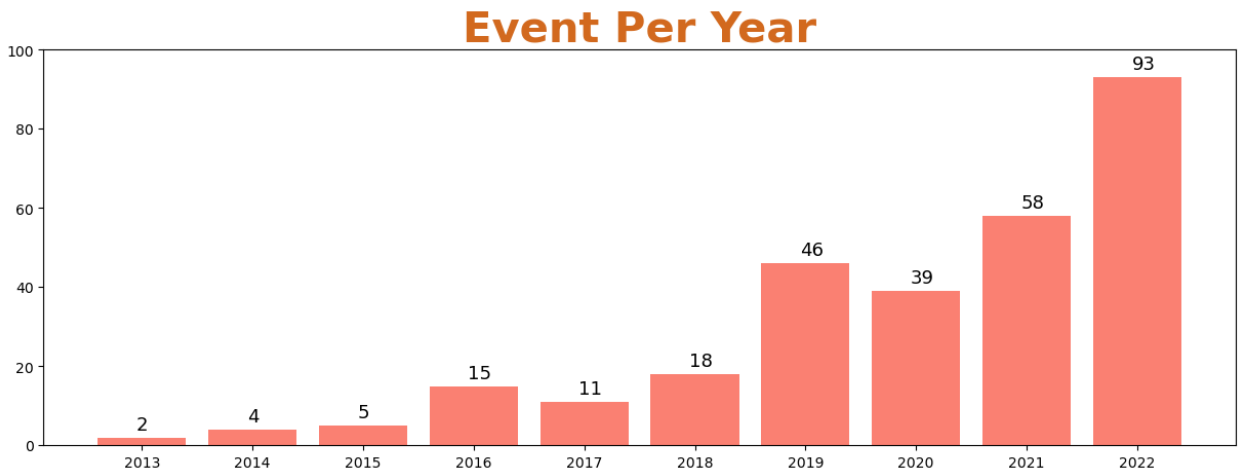
Year wise info

Remove year 2023 as too little info available

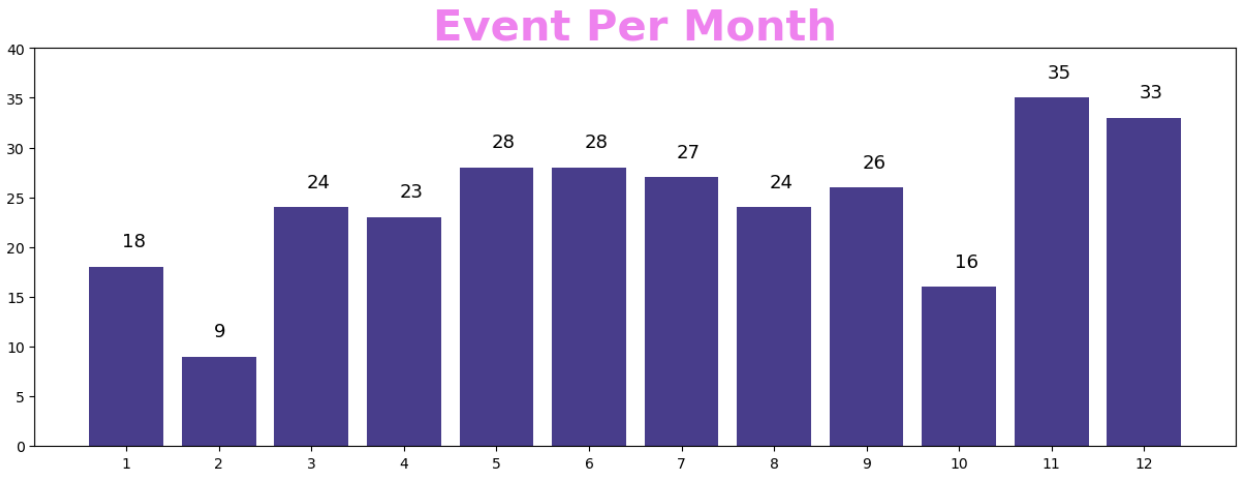
```
data = data[data.event_year != 2023]

vc=data.event_year.value_counts()
vc=vc.sort_index()
vc
```

```
2013     2
2014     4
2015     5
2016    15
2017    11
2018    18
2019    46
2020    39
2021    58
2022    93
Name: event_year, dtype: int64
```

```
plt.figure(figsize=(15,5))
plt.bar(height=vc.values,x=vc.index,color="salmon")
plt.xticks(vc.index,vc.index)
for i in vc.index:
    plt.annotate(vc[i],xy=(i-0.05,vc[i]+2),size=13)
plt.ylim(0,100)
plt.title("Event Per Year",size=30,color="chocolate",weight="heavy")
plt.show()
```

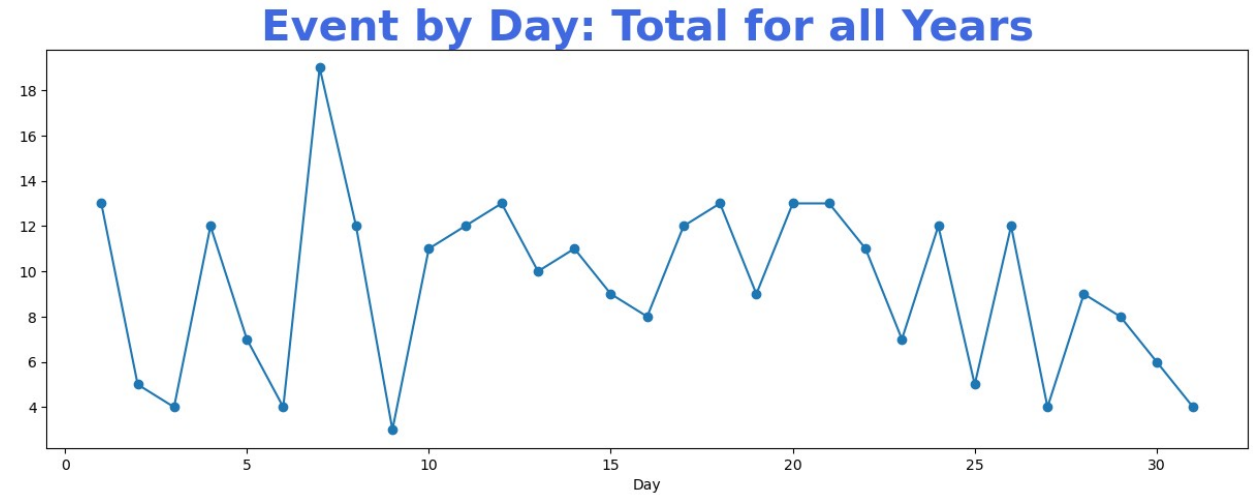



```
vc=data.event_month.value_counts()
vc=vc.sort_index()
plt.figure(figsize=(15,5))
plt.bar(height=vc.values,x=vc.index,color="darkslateblue")
plt.xticks(vc.index,vc.index)
for i in vc.index:
    plt.annotate(vc[i],xy=(i-0.05,vc[i]+2),size=13)
plt.ylim(0,5*round(vc.max()/5)+5)
plt.title("Event Per Month",size=30,color="violet",weight="heavy")
plt.show()
```




```
vc=data.event_day.value_counts()
vc=vc.sort_index()
plt.figure(figsize=(15,5))
plt.plot(vc.index,vc.values)
plt.scatter(vc.index,vc.values)
plt.xlabel("Day")
plt.title("Event by Day: Total for all
```

```
Years",size=30,color="royalblue",weight="heavy")
plt.show()
```



Conclusion

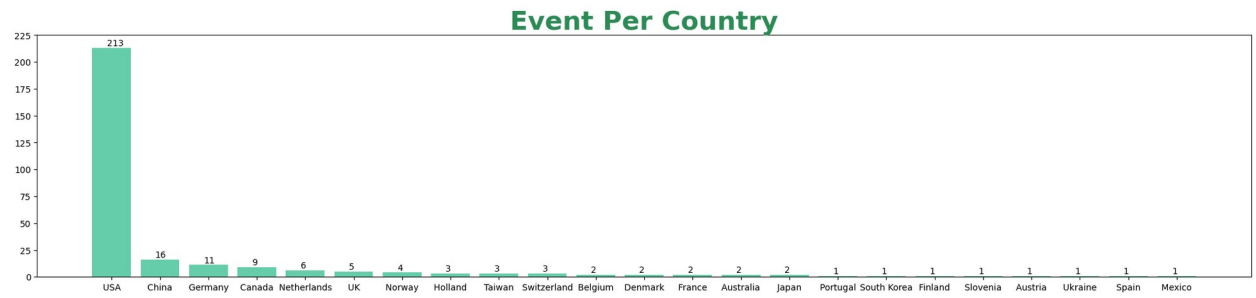
- Tesla's accident volume tends to increase each year

- The number of accidents in November and December are highest

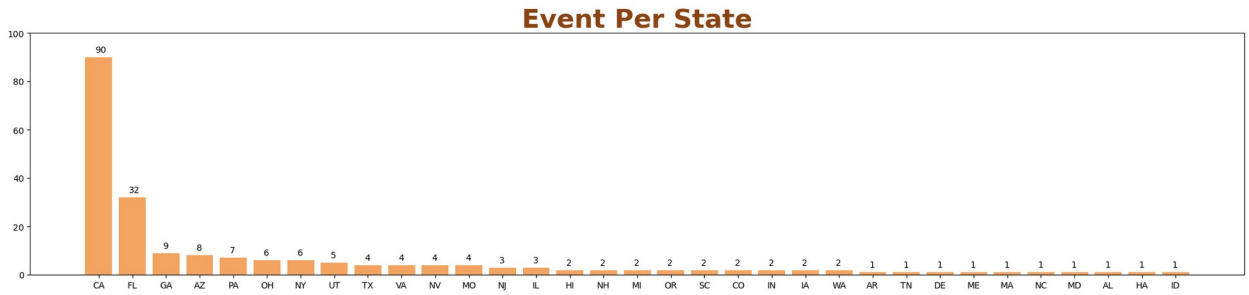
- Day wise no concrete indo as the distribution pattern is irregular

''' However, for a year wise pattern we should be considering the accidents vs sales data. More Tesla on road would defintely bring a rise in no. of accidents while proportion of accidents might not increase as such'''

```
vc=data.Country.value_counts()
plt.figure(figsize=(25,5))
plt.bar(height=vc.values,x=vc.index,color="mediumaquamarine")
plt.xticks(vc.index,vc.index)
for i in range(len(vc.index)):
    plt.annotate(vc[i],xy=(i-0.1,vc[i]+2),size=10)
plt.ylim(0,25*round(vc.max()/25))
plt.title("Event Per Country",size=30,color="seagreen",weight="heavy")
plt.show()
```



```
data.State=data.State.str.strip()

vc=data.State.value_counts()
vc=vc[vc.index !="-"]
plt.figure(figsize=(25,5))
plt.bar(height=vc.values,x=vc.index,color="sandybrown")
plt.xticks(vc.index,vc.index)
for i in range(len(vc.index)):
    plt.annotate(vc[i],xy=(i-0.1,vc[i]+2),size=10)
plt.ylim(0,25*round(vc.max()/25))
plt.title("Event Per
State",size=30,color="saddlebrown",weight="heavy")
plt.show()
```



Analyze the different aspects of the death events

1. What is the number of victims (deaths) in each accident?

2. How many times did tesla drivers die?

3. What is the proportion of events in which one or more occupants died?

4.What is the distribution of events in which the vehicle hit a cyclist or a pedestrian?

5.How many times did the accident involve the death of an occupant or driver of a Tesla along with a cyclist or pedestrian?

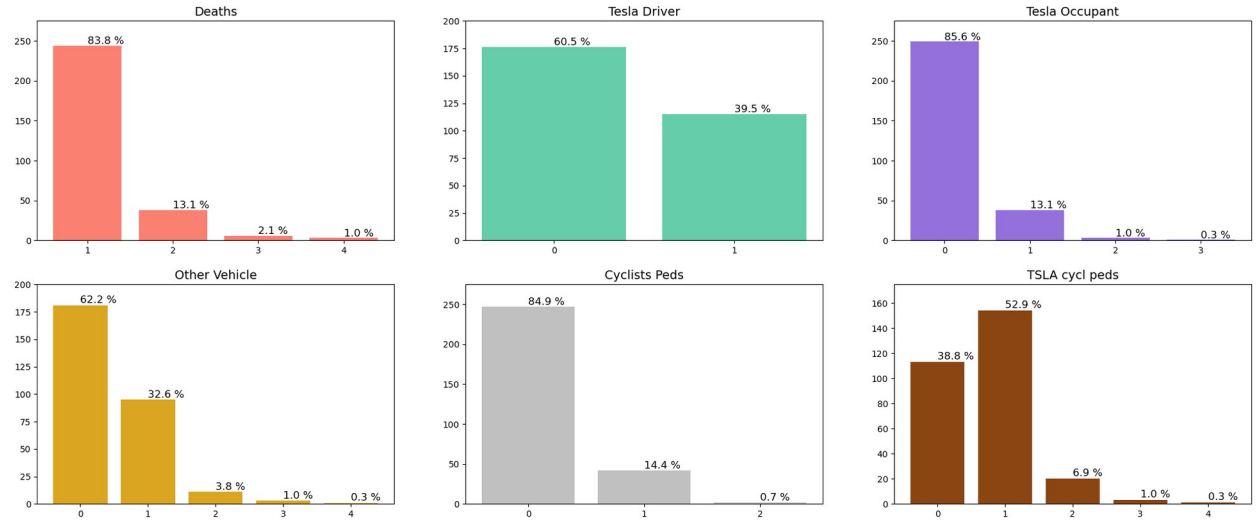
6.What is the frequency of Tesla colliding with other vehicles?

```
data.columns
```

```
Index(['Date', 'Country', 'State', 'Description', 'Deaths',
'Tesla_Driver',
       'Tesla_Occupant', 'Other_Vehicle', 'Cyclists_Peds',
'TSLA_cycl_peds',
       'Model', 'Claimed', 'VTAD', 'event_year', 'event_month',
'event_day'],
      dtype='object')
```

```
col_lists=['Deaths','Tesla_Driver','Tesla_Occupant', 'Other_Vehicle',
'Cyclists_Peds', 'TSLA_cycl_peds']
```

```
colr=['salmon','mediumaquamarine','mediumpurple','goldenrod','silver',
'saddlebrown']

f, ax = plt.subplots(2, 3, figsize=(25, 10))
i, j, k = 0, 0, 0

for col in col_lists:
    vc = data[col].value_counts()
    vc = vc.sort_index()
    perc = (vc / vc.sum() * 100).round(1)
    ax[i, j].bar(x=vc.index, height=vc.values, color=colr[k])
    ax[i, j].set_title(col.replace("_", " "), size=14)
    ax[i, j].set_xticks(vc.index)

    # Change loop variable here to avoid overwriting subplot indices
    for idx in vc.index:
        ax[i, j].annotate("{} %".format(perc[idx]), xy=(idx, vc[idx] +
2), size=12)
    ax[i, j].set_ylim(0, 25 * round(vc.max() / 25) + 25)

    j += 1
    k += 1
    if j == 3:
        j = 0
        i += 1
```

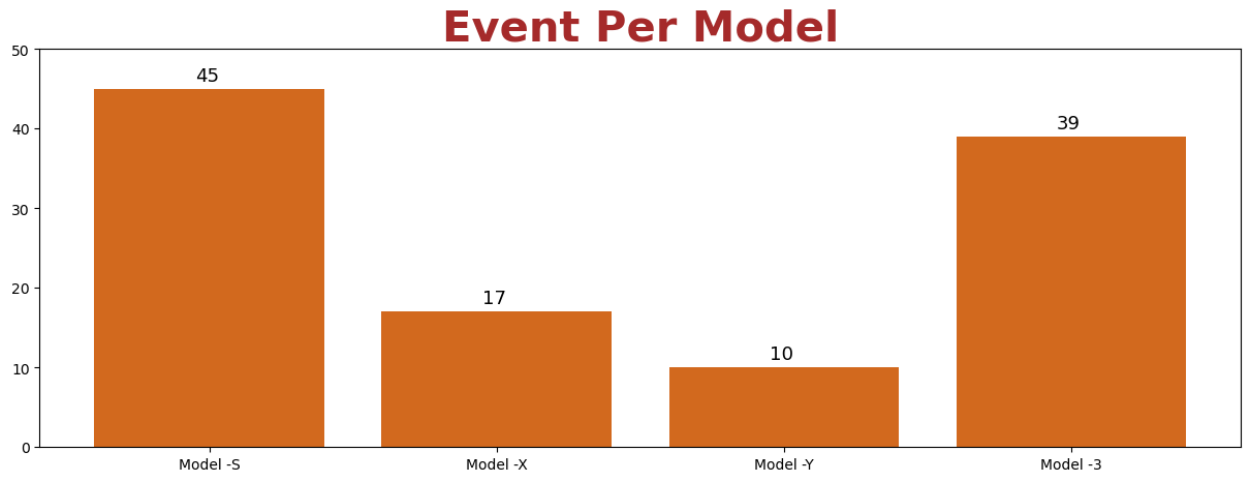


Study the event distribution across models

```
data.Model=data.Model.str.strip()
data.Model=data.Model.replace("-","0")
```

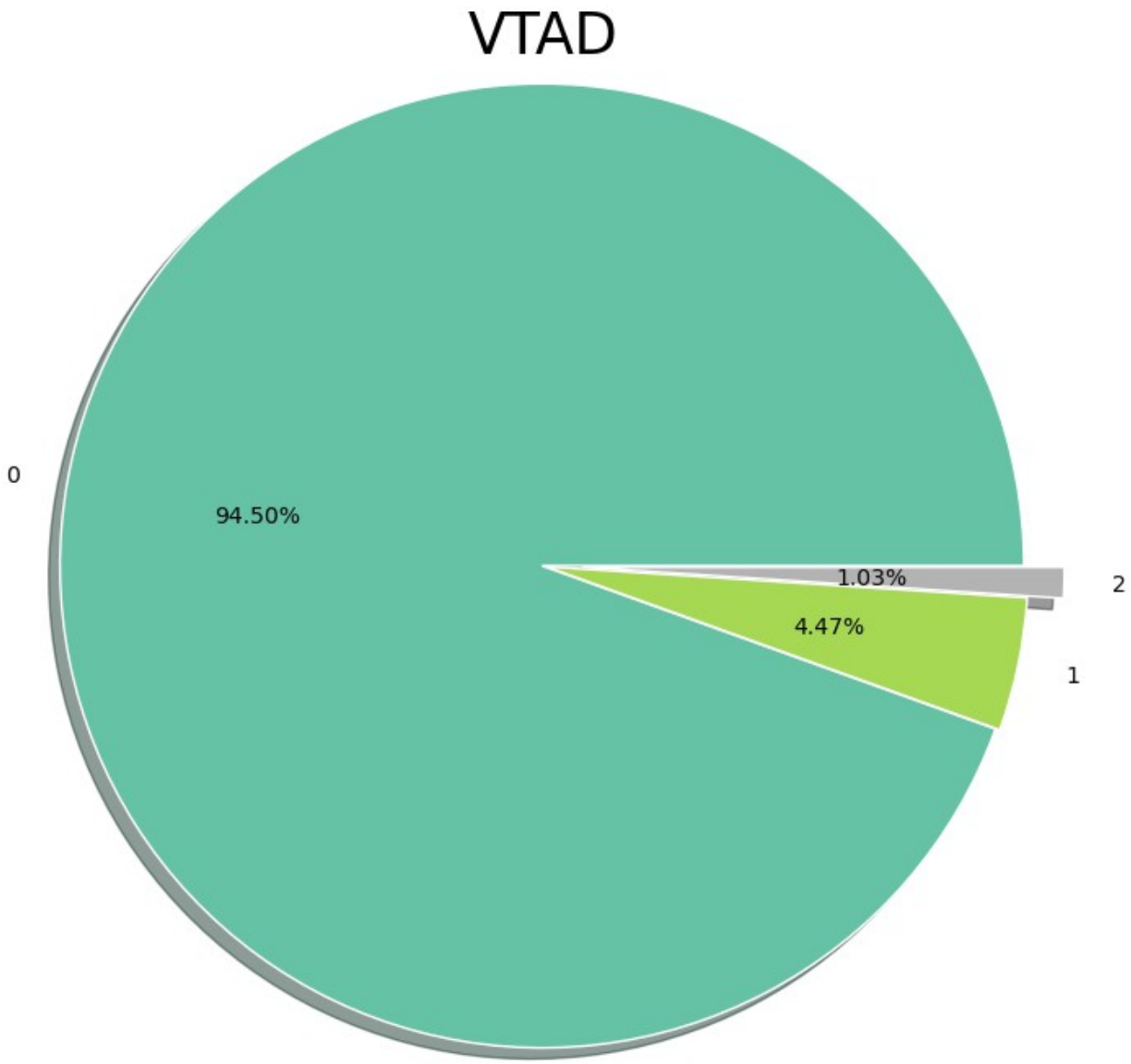
Since we have currently 4 existing models S,XY and 3 we will analyze them

```
vc=data.Model.value_counts()
vc=vc[["S","X","Y","3"]]
plt.figure(figsize=(15,5))
plt.bar(height=vc.values,x=vc.index,color="chocolate")
plt.xticks(vc.index,"Model -"+vc.index.astype(str))
for i in range(len(vc.index)):
    plt.annotate(vc[i],xy=(i-0.05,vc[i]+1),size=13)
plt.ylim(0,25*round(vc.max()/25))
plt.title("Event Per Model",size=30,color="brown",weight="heavy")
plt.show()
```



Check the distribution of verified Tesla Pilot Deaths

```
vc=data.VTAD.value_counts()
plt.figure(figsize=(10,8))
vc.plot.pie(radius=1.2,autopct="%1.2f%
%",shadow=True,wedgeprops={'edgecolor' :
"white"},cmap='Set2',explode=[0.001,0.01,0.1])
plt.ylabel("")
plt.title("VTAD",pad=5,size=25)
plt.show()
```


VTAD
0
94.50%
1.03%
2
4.47%
1