Data Analysis of Motor Vechile Collisions in USA

Loading the Data and necessary libraries

```
#importing the necessary libraries for the Analysis
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

```
#Path of the file from which the data is taken
file = '/content/drive/MyDrive/Motor_Vehicle_Collisions_-_Vehicles.csv'
```

```
#Read the csv file and converted it into a dataframe.
df = pd.read_csv(file)
```

```
/usr/local/lib/python3.7/dist-packages/IPython/core/interactiveshell.py:3326: DtypeWa
  exec(code_obj, self.user_global_ns, self.user_ns)
```

Data Pre-Processing

```
#Checked the shape to identify the total no of rows and columns
df.shape
```

```
(3704406, 25)
```

```
#To see the representation of CRASH DATE column to make appropriate patterns for extractio
df['CRASH_DATE']
```

```
0          09/07/2012
1          09/23/2019
2          10/02/2015
3          10/04/2015
4          04/25/2013
              ...
3704401    11/15/2021
3704402    11/24/2021
3704403    11/11/2021
3704404    11/06/2021
3704405    12/02/2021
Name: CRASH_DATE, Length: 3704406, dtype: object
```

```
df['CRASH_DATE'] = pd.to_datetime(df['CRASH_DATE'])
```

```
df.head(5)
```

	UNIQUE_ID	COLLISION_ID	CRASH_DATE	CRASH_TIME	VEHICLE_ID	STATE_REGISTRATION
0	10385780	100201	2012-09-07	9:03	1	NY
1	19140702	4213082	2019-09-23	8:15	0553ab4d-9500-4cba-8d98-f4d7f89d5856	NY
2	14887647	3307608	2015-10-02	17:18	2	NY
3	14889754	3308693	2015-10-04	20:34	1	NY
4	14400270	297666	2013-04-25	21:15	1	NY

5 rows × 25 columns

```
# df['CRASH_DATE'] = df['CRASH_DATE'].dt.strftime('%d-%m-%Y')
```

```
start_date = '2018-09-01'
end_date = '2020-08-31'
# mvc_data = df.query('CRASH_DATE <= @end_date and CRASH_DATE >= @start_date')
mask = (df['CRASH_DATE'] >= start_date) & (df['CRASH_DATE'] <= end_date)
```

```
df2 = df.loc[mask]
```

```
df2.shape
```

```
(741086, 25)
```

```
# Stored the crash data of the years 2015 and 2021 to a csv file "MVC.csv" for future use.
mvc_data = df2.to_csv('/content/drive/MyDrive/MVC.csv')
```

Crash data between the date 1st September 2018 to 31st August 2020

```
# Loaded the the data of the year 2015 and 2021 and made it to a dataframe "new_df and rep
new_data = '/content/drive/MyDrive/MVC.csv'
missing_values = ["?", "--"]
new_df = pd.read_csv(new_data, na_values= missing_values)
```

```
new_df.shape
```

```
(741086, 26)
```

Crash data between the dates 1st September 2018 to 31st August 2020 consist of 741086 accidents. That is 20% of the accidents happened in during this period.

Initial Data Pre-Processing

Replace all missing values with NaN

```
new_df.replace({"?":np.NaN,"--":np.NaN}, inplace = True)
```

Finding Missing Values

```
# To find the columns that contain NaN values.
new_df.isna().any()
```

```
Unnamed: 0                        False
UNIQUE_ID                         False
COLLISION_ID                      False
CRASH_DATE                        False
CRASH_TIME                        False
VEHICLE_ID                        False
STATE_REGISTRATION                 True
VEHICLE_TYPE                       True
VEHICLE_MAKE                       True
VEHICLE_MODEL                      True
VEHICLE_YEAR                       True
TRAVEL_DIRECTION                   True
VEHICLE_OCCUPANTS                  True
DRIVER_SEX                         True
DRIVER_LICENSE_STATUS              True
DRIVER_LICENSE_JURISDICTION        True
PRE_CRASH                          True
POINT_OF_IMPACT                    True
VEHICLE_DAMAGE                     True
VEHICLE_DAMAGE_1                   True
VEHICLE_DAMAGE_2                   True
VEHICLE_DAMAGE_3                   True
PUBLIC_PROPERTY_DAMAGE            False
PUBLIC_PROPERTY_DAMAGE_TYPE        True
CONTRIBUTING_FACTOR_1              True
```

```
CONTRIBUTING_FACTOR_2           True
dtype: bool
```

```
# To find the no of missing values in each column.
new_df.isna().sum()
```

```
Unnamed: 0                          0
UNIQUE_ID                           0
COLLISION_ID                        0
CRASH_DATE                          0
CRASH_TIME                          0
VEHICLE_ID                          0
STATE_REGISTRATION              74525
VEHICLE_TYPE                    59835
VEHICLE_MAKE                    85791
VEHICLE_MODEL                  741086
VEHICLE_YEAR                    88815
TRAVEL_DIRECTION                37107
VEHICLE_OCCUPANTS               64914
DRIVER_SEX                     175432
DRIVER_LICENSE_STATUS          192516
DRIVER_LICENSE_JURISDICTION    192465
PRE_CRASH                       44956
POINT_OF_IMPACT                 45833
VEHICLE_DAMAGE                  51541
VEHICLE_DAMAGE_1               293707
VEHICLE_DAMAGE_2               404362
VEHICLE_DAMAGE_3               483146
PUBLIC_PROPERTY_DAMAGE              0
PUBLIC_PROPERTY_DAMAGE_TYPE    734583
CONTRIBUTING_FACTOR_1           35702
CONTRIBUTING_FACTOR_2           41405
dtype: int64
```

```
new_df.head(5)
```

	Unnamed: 0	UNIQUE_ID	COLLISION_ID	CRASH_DATE	CRASH_TIME	VEHICLE_ID	STATE_RE(
0	1	19140702	4213082	2019-09-23	8:15	0553ab4d-9500-4cba-8d98-f4d7f89d5856	
1	6	19138701	4229067	2019-10-24	13:15	c53b43d9-419a-4ab1-9361-3f2979078d89	

```
""" Column 'Unnamed: 0' represented the previous indexing of the data before extraction no
only NaN values for the entire rows so we can drop that too."""
data = new_df.drop(labels = ['Unnamed: 0', 'VEHICLE_MODEL'], axis = 1)
```

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

```
UNIQUE_ID                          0
COLLISION_ID                       0
CRASH_DATE                         0
CRASH_TIME                         0
VEHICLE_ID                         0
STATE_REGISTRATION             74525
VEHICLE_TYPE                   59835
VEHICLE_MAKE                   85791
VEHICLE_YEAR                   88815
TRAVEL_DIRECTION               37107
VEHICLE_OCCUPANTS              64914
DRIVER_SEX                    175432
DRIVER_LICENSE_STATUS         192516
DRIVER_LICENSE_JURISDICTION   192465
PRE_CRASH                      44956
POINT_OF_IMPACT                45833
VEHICLE_DAMAGE                 51541
VEHICLE_DAMAGE_1              293707
VEHICLE_DAMAGE_2              404362
VEHICLE_DAMAGE_3              483146
PUBLIC_PROPERTY_DAMAGE             0
PUBLIC_PROPERTY_DAMAGE_TYPE   734583
CONTRIBUTING_FACTOR_1          35702
CONTRIBUTING_FACTOR_2          41405
dtype: int64
```

```
data['CRASH_DATE'] = pd.to_datetime(data['CRASH_DATE'])
```

```
data['YEAR'], data['MONTH'] , data['DAY']= data['CRASH_DATE'].dt.year, data['CRASH_DATE'].
```

```
data.head()
```

	UNIQUE_ID	COLLISION_ID	CRASH_DATE	CRASH_TIME	VEHICLE_ID	STATE_REGISTRATION
0	19140702	4213082	2019-09-23	8:15	0553ab4d-9500-4cba-8d98-f4d7f89d5856	NY
1	19138701	4229067	2019-10-24	13:15	c53b43d9-419a-4ab1-9361-3f2979078d89	NY
2	19140791	4229563	2019-10-21	17:55	86a294b4-6672-4a7e-8357-39d6d2eff9f7	PA
3	19694316	4322767	2020-06-06	18:30	fdc195a7-8127-4c00-834d-bac78b0cf88e	NaN
4	19140656	4229538	2019-10-24	17:30	70e5262a-bd27-48a6-99a1-1ec659804088	NY

5 rows × 27 columns

```
data.drop(labels='DAY', axis = 1, inplace = True)
```

```
data.columns
```

```
Index(['UNIQUE_ID', 'COLLISION_ID', 'CRASH_DATE', 'CRASH_TIME', 'VEHICLE_ID',
       'STATE_REGISTRATION', 'VEHICLE_TYPE', 'VEHICLE_MAKE', 'VEHICLE_YEAR',
       'TRAVEL_DIRECTION', 'VEHICLE_OCCUPANTS', 'DRIVER_SEX',
       'DRIVER_LICENSE_STATUS', 'DRIVER_LICENSE_JURISDICTION', 'PRE_CRASH',
       'POINT_OF_IMPACT', 'VEHICLE_DAMAGE', 'VEHICLE_DAMAGE_1',
       'VEHICLE_DAMAGE_2', 'VEHICLE_DAMAGE_3', 'PUBLIC_PROPERTY_DAMAGE',
       'PUBLIC_PROPERTY_DAMAGE_TYPE', 'CONTRIBUTING_FACTOR_1',
       'CONTRIBUTING_FACTOR_2', 'YEAR', 'MONTH'],
      dtype='object')
```

```
#There can be chances of empty values that are not np.NaN, ??, -- in the dataset so we hav
data = data.reindex(['UNIQUE_ID', 'COLLISION_ID', 'CRASH_DATE', 'YEAR', 'MONTH', 'CRASH_TI
       'STATE_REGISTRATION', 'VEHICLE_TYPE', 'VEHICLE_MAKE', 'VEHICLE_YEAR',
       'TRAVEL_DIRECTION', 'VEHICLE_OCCUPANTS', 'DRIVER_SEX',
       'DRIVER_LICENSE_STATUS', 'DRIVER_LICENSE_JURISDICTION', 'PRE_CRASH',
       'POINT_OF_IMPACT', 'VEHICLE_DAMAGE', 'VEHICLE_DAMAGE_1',
       'VEHICLE_DAMAGE_2', 'VEHICLE_DAMAGE_3', 'PUBLIC_PROPERTY_DAMAGE',
       'PUBLIC_PROPERTY_DAMAGE_TYPE', 'CONTRIBUTING_FACTOR_1',
       'CONTRIBUTING_FACTOR_2'], axis=1, fill_value = np.NaN)
```

```
data.head()
```

	UNIQUE_ID	COLLISION_ID	CRASH_DATE	YEAR	MONTH	CRASH_TIME	VEHICLE_ID	STATE_
0	19140702	4213082	2019-09-23	2019	9	8:15	0553ab4d-9500-4cba-8d98-f4d7f89d5856	
1	19138701	4229067	2019-10-24	2019	10	13:15	c53b43d9-419a-4ab1-9361-3f2979078d89	
2	19140791	4229563	2019-10-21	2019	10	17:55	86a294b4-6672-4a7e-8357-39d6d2eff9f7	
3	19694316	4322767	2020-06-06	2020	6	18:30	fdc195a7-8127-4c00-834d-bac78b0cf88e	
4	19140656	4229538	2019-10-24	2019	10	17:30	70e5262a-bd27-48a6-99a1-1ec659804088	

5 rows × 26 columns

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

```
UNIQUE_ID                           0
COLLISION_ID                        0
CRASH_DATE                          0
YEAR                                0
MONTH                               0
CRASH_TIME                          0
VEHICLE_ID                          0
STATE_REGISTRATION              74525
VEHICLE_TYPE                    59835
VEHICLE_MAKE                    85791
VEHICLE_YEAR                    88815
TRAVEL_DIRECTION                37107
VEHICLE_OCCUPANTS               64914
DRIVER_SEX                     175432
DRIVER_LICENSE_STATUS          192516
DRIVER_LICENSE_JURISDICTION    192465
PRE_CRASH                       44956
POINT_OF_IMPACT                 45833
VEHICLE_DAMAGE                  51541
VEHICLE_DAMAGE_1               293707
VEHICLE_DAMAGE_2               404362
VEHICLE_DAMAGE_3               483146
PUBLIC_PROPERTY_DAMAGE              0
PUBLIC_PROPERTY_DAMAGE_TYPE    734583
CONTRIBUTING_FACTOR_1           35702
```

```
CONTRIBUTING_FACTOR_2          41405
dtype: int64
```

```
"""We are using VECHILE_MAKE for our analysis of the missing values in the particular colu
So we can remove the 85791 rows of data in which the VECHILE_MAKE column is a null value."
data.dropna(axis=0, subset=['VEHICLE_MAKE'], inplace = True)
```

```
# data_new = data_new.dropna(axis=0, subset=['STATE_REGISTRATION'])
data.isna().sum()
```

```
UNIQUE_ID                          0
COLLISION_ID                       0
CRASH_DATE                         0
YEAR                               0
MONTH                              0
CRASH_TIME                         0
VEHICLE_ID                         0
STATE_REGISTRATION              4096
VEHICLE_TYPE                    4571
VEHICLE_MAKE                       0
VEHICLE_YEAR                   11112
TRAVEL_DIRECTION                2888
VEHICLE_OCCUPANTS              10445
DRIVER_SEX                    110414
DRIVER_LICENSE_STATUS         120242
DRIVER_LICENSE_JURISDICTION   119691
PRE_CRASH                       5152
POINT_OF_IMPACT                 5075
VEHICLE_DAMAGE                  7336
VEHICLE_DAMAGE_1              227869
VEHICLE_DAMAGE_2              335580
VEHICLE_DAMAGE_3              412667
PUBLIC_PROPERTY_DAMAGE             0
PUBLIC_PROPERTY_DAMAGE_TYPE   655295
CONTRIBUTING_FACTOR_1           2634
CONTRIBUTING_FACTOR_2           4237
dtype: int64
```

```
data.dropna(axis=0, subset=['STATE_REGISTRATION'], inplace = True)
```

```
data.dropna(axis=0, subset=['VEHICLE_TYPE'], inplace = True)
```

```
#check is any value in public property damage is yes insted of N
```

```
data.drop(labels ='PUBLIC_PROPERTY_DAMAGE_TYPE', axis = 1)
```

	UNIQUE_ID	COLLISION_ID	CRASH_DATE	YEAR	MONTH	CRASH_TIME	VEHICLE_ID
0	19140702	4213082	2019-09-23	2019	9	8:15	0553ab4d-9500-4cba-8d98-f4d7f89d5856
1	19138701	4229067	2019-10-24	2019	10	13:15	c53b43d9-419a-4ab1-9361-3f2979078d89
2	19140791	4229563	2019-10-21	2019	10	17:55	86a294b4-6672-4a7e-8357-39d6d2eff9f7
4	19140656	4229538	2019-10-24	2019	10	17:30	70e5262a-bd27-48a6-99a1-1ec659804088
5	19139721	4228839	2019-10-24	2019	10	16:00	5bb0b59a-ce74-4a04-9f92-1446ebfe4f46
...	...	...	...	...	...	...	...
741079	20099856	4063673	2019-01-07	2019	1	12:45	57c0614a-9816-46d1-ad24-517de24722ec
741080	20113435	4060927	2019-01-01	2019	1	17:00	ea2d2b88-d405-4233-b0e2-a0c5988cb18f
741082	20099855	4063673	2019-01-07	2019	1	12:45	ae6c19f2-30a0-4892-8f08-d80fcbb057c6
741083	20101746	4295822	2020-02-27	2020	2	10:00	5ec4913a-a77d-439a-81b9-0ce8a2d27c2d
							154f05e4-

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

```
UNIQUE_ID                     0
COLLISION_ID                  0
CRASH_DATE                    0
YEAR                          0
MONTH                         0
CRASH_TIME                    0
VEHICLE_ID                    0
STATE_REGISTRATION            0
VEHICLE_TYPE                  0
```

```
VEHICLE_MAKE                        0
VEHICLE_YEAR                     6566
TRAVEL_DIRECTION                 1534
VEHICLE_OCCUPANTS                6768
DRIVER_SEX                     104399
DRIVER_LICENSE_STATUS          113984
DRIVER_LICENSE_JURISDICTION    113316
PRE_CRASH                        3476
POINT_OF_IMPACT                  3142
VEHICLE_DAMAGE                   4804
VEHICLE_DAMAGE_1               222976
VEHICLE_DAMAGE_2               330243
VEHICLE_DAMAGE_3               407045
PUBLIC_PROPERTY_DAMAGE              0
PUBLIC_PROPERTY_DAMAGE_TYPE    647732
CONTRIBUTING_FACTOR_1            1287
CONTRIBUTING_FACTOR_2            2452
dtype: int64
```

Pre processing for PUBLIC_PROPERTY_DAMAGE Column

```
data['PUBLIC_PROPERTY_DAMAGE'].unique()
```

```
array(['N', 'Unspecified', 'Y'], dtype=object)
```

```
data['PUBLIC_PROPERTY_DAMAGE'].value_counts()
```

```
N              613515
Unspecified     30866
Y                3351
Name: PUBLIC_PROPERTY_DAMAGE, dtype: int64
```

```
data[data['PUBLIC_PROPERTY_DAMAGE']== 'Unspecified'].shape
# There are a total of 30866 accidents in which the public property damage is not availabl
#This can be considered as NaN values itself as we cannot conclude it as an yes or no. Thi
```

```
(30866, 26)
```

Pre processing for DRIVER_SEX Column

```
data['DRIVER_SEX'].unique()
# 104399 missing values are presented remove this would affect the initial analysis. So it
```

```
array(['M', 'F', nan, 'U'], dtype=object)
```

```
data['DRIVER_SEX'].value_counts()
```

```
M    399902
F    142382
```

```
U      1049
Name: DRIVER_SEX, dtype: int64
```

Pre processing for POINT_OF_IMPACT Column

```
data['POINT_OF_IMPACT'].unique()
# 3142 missing values are present but it is not neccessary for initial analysis.
```

```
array(['Left Front Bumper', 'Left Front Quarter Panel',
       'Center Front End', 'Left Rear Quarter Panel', 'Right Side Doors',
       'Left Side Doors', 'Right Front Bumper', 'Center Back End',
       'Right Rear Quarter Panel', 'Left Rear Bumper',
       'Right Front Quarter Panel', 'Right Rear Bumper', 'No Damage', nan,
       'Trailer', 'Other', 'Roof', 'Overturned', 'Demolished',
       'Undercarriage'], dtype=object)
```

```
data['POINT_OF_IMPACT'].value_counts()
# This data seems to be of good quality and there is nothing to be done for this column.
```

```
Center Front End            102861
Left Front Bumper            82498
Center Back End              81641
Right Front Bumper           72240
Right Front Quarter Panel    48525
Left Front Quarter Panel     48249
Left Rear Quarter Panel      39859
Left Rear Bumper             35509
Left Side Doors              35426
Right Side Doors             29139
Right Rear Quarter Panel     28042
Right Rear Bumper            23253
No Damage                     9438
Other                         4557
Roof                          1381
Trailer                        902
Undercarriage                  407
Overturned                     373
Demolished                     290
Name: POINT_OF_IMPACT, dtype: int64
```

Pre processing for STATE_REGISTRATION Column

```
data['STATE_REGISTRATION'].unique()
```

```
array(['NY', 'PA', 'NC', 'NM', 'OK', 'NJ', 'VA', 'FL', 'MN', 'ON', 'IL',
       'AL', 'TX', 'MI', 'TN', 'MA', 'MD', 'CT', 'AZ', 'IN', 'ZZ', 'ME',
       'NH', 'WA', 'GA', 'AK', 'OH', 'MO', 'KS', 'SC', 'KY', 'CA', 'PQ',
       'DE', 'MS', 'VT', 'CO', 'AR', 'RI', 'UT', 'IA', 'LA', 'OR', 'NF',
       'WI', 'NB', 'ID', 'MT', 'WV', 'NV', 'SK', 'ND', 'BC', 'DC', 'SD',
```

```
       'MX', 'PE', 'NS', 'HI', 'WY', 'NT', 'AB', 'MB', 'YT', 'UA', 'LR'],
      dtype=object)
```

```
#There are only 50 states in usa so the state registration column is having some errors it
#There are fifty (50) states and Washington D.C.The last two states to join the Union were
data['STATE_REGISTRATION'].value_counts()
# Plot the state registration with folium in last.
```

```
NY    532856
NJ     42015
PA     16388
CT      8271
FL      8061
       ...
MX         5
SK         4
YT         3
UA         1
LR         1
Name: STATE_REGISTRATION, Length: 66, dtype: int64
```

```
len(['ON', 'ZZ','PQ','NF','SK','BC', 'MX', 'PE', 'NS','NT', 'AB', 'MB', 'YT', 'UA', 'LR'])
# ON can be converted to OH as it can be due to error happened while typing since N is clo
# ZZ can be converted to AZ
# PQ can be converted to PA
# NF can be ND as they are on the same row in keyboard
""" Our assumption was wrong these were the vechiles coming from neighbouring countries th
```

```
' Our assumption was wrong these were the vechiles coming from neighbouring countrie
s that got into accidents inside the newyork'
```

```
df_ON = data[data['STATE_REGISTRATION'] =='ON']
# There is no state prefix 'ON' for a state in USA.
# Later identified that the datasets is the accidents occured in newyork which includes th
df_ON[df_ON['VEHICLE_TYPE'] == 'TRUCK'].shape
# 38 of the vehicle are trucks as per the final analysis.
```

```
(38, 26)
```

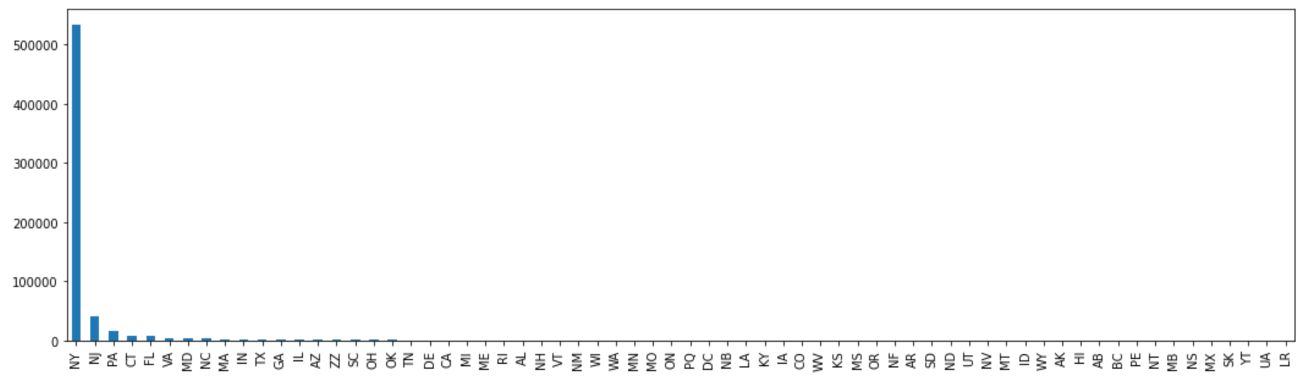
```
lst = ['NY', 'PA', 'NC', 'NM', 'OK', 'NJ', 'VA', 'FL', 'MN', 'ON', 'IL',
       'AL', 'TX', 'MI', 'TN', 'MA', 'MD', 'CT', 'AZ', 'IN', 'ZZ', 'ME',
       'NH', 'WA', 'GA', 'AK', 'OH', 'MO', 'KS', 'SC', 'KY', 'CA', 'PQ',
       'DE', 'MS', 'VT', 'CO', 'AR', 'RI', 'UT', 'IA', 'LA', 'OR', 'NF',
       'WI', 'NB', 'ID', 'MT', 'WV', 'NV', 'SK', 'ND', 'BC', 'DC', 'SD',
       'MX', 'PE', 'NS', 'HI', 'WY', 'NT', 'AB', 'MB', 'YT', 'UA', 'LR']
tst = ['ON', 'ZZ','PQ','NF','SK','BC', 'MX', 'PE', 'NS','NT', 'AB', 'MB', 'YT', 'UA', 'LR'
for i in tst:
  if i in lst:
    lst.remove(i)
print(lst)
# The state registration includes the state registration as well as state registration und
# So there are a total of 55 registrations inside usa.
# Question is whether we should remove the remaining data 11 state prefix. It can be the v
# Example 'ON' is the state registration of ontario, canada
```

```
['NY', 'PA', 'NC', 'NM', 'OK', 'NJ', 'VA', 'FL', 'MN', 'IL', 'AL', 'TX', 'MI', 'TN',
```

There is no need of removing the state registration nor replacing it since the NYC accidents data include the vehicle from usa and from outside the country. Several trucks from cananda and near by countries come into usa with goods. for example, Vechiles with state registration ON(ontario, canada) has 261 accidents occured in within NYC

```
plt.rcParams["figure.figsize"] = (18,5)
data['STATE_REGISTRATION'].value_counts().plot(kind ='bar')
# Majority of accidents happened by the vehicle within 'NY'-New York and on the second pla
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc3e66e4d90>
```



Pre processing for DRIVER_LICENSE_JURISDICTION Column

```
data['DRIVER_LICENSE_JURISDICTION'].unique()
```

```
array(['NY', 'PA', 'NC', nan, 'NJ', 'VA', 'MN', 'ON', 'AL', 'FL', 'MX',
       'MA', 'MD', 'TX', 'CT', 'ZZZ', 'IL', 'TN', 'NB', 'CA', 'PQ', 'WA',
       'AK', 'RI', 'OH', 'MO', 'MI', 'KS', 'GA', 'NH', 'MS', 'AZ', 'SC',
       'ID', 'CO', 'DE', 'VT', 'IN', 'OK', 'NF', 'DC', 'MT', 'NM', 'AB',
       'ME', 'IA', 'ND', 'LA', 'OR', 'NB1', 'NV', 'WI', 'HI', 'WV', 'UT',
       'KY', 'SD', 'BC', 'AR', 'WY', 'YT', 'NS', 'PE', 'PR', 'MB', 'SK',
       'NT', 'A,NEJADE', 'NE', "PA'"], dtype=object)
```

```
data[data['DRIVER_LICENSE_JURISDICTION'] =='A,NEJADE']
# This is an error there is not state with such a license jurisdiction.
#(647732, 26)
data.drop(data[data['DRIVER_LICENSE_JURISDICTION'] == 'A,NEJADE'].index, inplace = True)
```

```
data.shape
# The error row is removed.
```

```
(647731, 26)
```

Pre processing for DRIVER_LICENSE_STATUS Column

```
data['DRIVER_LICENSE_STATUS'].value_counts()
# 113984 missing values are also present in this data. But still accidents are mostly crea
# We can handle the missing values later as it will remove a major portion of the valuable
```

```
Licensed      523716
Unlicensed      6772
Permit          3259
Name: DRIVER_LICENSE_STATUS, dtype: int64
```

Pre processing for VEHICLE_MAKE Column

```
data['VEHICLE_MAKE'].value_counts()
```

```
TOYT -CAR/SUV          115565
HOND -CAR/SUV           81443
NISS -CAR/SUV           68071
FORD -CAR/SUV           53675
CHEV -CAR/SUV           31045
                          ...
fdny truck                  1
Winnebago                   1
ALEXANDER DENNIS            1
VESPA (GTS300IE)            1
SEAGRAVE TOWERLADDER        1
Name: VEHICLE_MAKE, Length: 3779, dtype: int64
```

```
import re
def vechile_make_extractor(name):
  if re.search('\-.*', name):
    p = re.search('\-.*', name).start()
    return name[:p-1] # We want the space to be removed otherwise 'GMC ' and 'GMC' will be
  else:
    return name
data['VEHICLE_MAKE'] = data['VEHICLE_MAKE'].apply(vechile_make_extractor)
```

```
data['VEHICLE_MAKE'].unique()
```

```
array(['TOYT', 'FRH', 'BMW', ..., 'BMW/ 535I XDRIV', 'DONG',
       'SEAGRAVE TOWERLADDER'], dtype=object)
```

```
a = list(data['VEHICLE_MAKE'].unique())
b = []
for i in a:
  try:
    if i[0] == 'C' or i[0] == 'c': # checked with the first letter of the vehicle make lab
      b.append(i)
  except:
    pass
print(b)
```

```
['CHEV', 'CHRY', 'CADI', 'COLLI', 'CHE', 'CATER', 'CHEVROLET', 'COOP', 'CAT', 'colli
```

Different ways in which the vehicle make is represented within the VEHICLE_MAKE column

```
"""
'TOYT' = 'TOYOTA', 'toyota', 'TOY', 'TOYOTA FORKLIFT', 'TOYOTA PRIUS'
'SUBA' =  'SUBAR','subaru'
'CADI' = 'Cadillac','CADIL', 'cadil',
'GMC' = 'GM', 'gmc', 'GMC9', 'Gmc', 'GMC 999', 'GMC TANK',
'DODGE' = 'DODG','DOGE', 'DOD', 'dodge', 'DONGE', 'Dodge RAM',
'SUBN' = ''TRANS SUBN', 'subn', 'SUBN', 'suburban'
'DUCATI' = 'DUCA', 'ducati',
'TESLA' = 'TESL'
'ISUZU'= 'ISU', 'IS', 'IZUZU', 'isu', 'isuzu', 'IZU',
'MERZ' = 'MERCEDEZ BENZ', 'MER', 'mercedes benz', 'MERCEDES BENZ', 'Mercedes Benz', 'MERCE
'ORION' = 'orion', 'Orio', 'Orion', 'OPION', 'ORIN', 'ORIO', 'ORION/OMNIBUS', 'ORION NONTR
'FREIGHTLINER' = 'freightliner', 'freig', 'frht', 'FRH', 'FREIGHT', 'FREIG', 'FREIGHLINER'
                 'FREIGHT LINER', 'Freig', 'Freightliner', 'freight liner', 'FRIEGHT', 'FR
                 'freightline', 'FRIEGHTLINER CORP', 'frgh', 'Freigh', 'FRGH', 'Frt', 'fri
                 'FR/LT', 'Freightleiner', 'FREIG TRUCK', 'freightlnr', 'friegthliner', 'f
                 'frieght', 'frieghtliner', 'Freightliner Bus', 'FREIGHTLINER CORP. 999, '
                 'FREIGHTLINER CORP.', 'freihtliner', 'freightliner corp', 'FREIGHTLINER C
'AMBULANCE' = 'ambulance', 'Ambulance','ford ambulance', 'AMBU WH/RD', 'FORD AMBULANCE', '
'YAMAHA' = 'YAMA', 'YAMAHA', 'yamah', 'Yamaha'
'NE/FL' = 'ne/fl', 'NE /FL', 'NE/F', 'newfl', 'NEWFL', 'NFLY', 'new flyer', 'newflyer', 'N
          'NEW FLYER, WHITE BLUE BUS(OMNIBUS)',
'INTL' = 'intl'
'REVEL' = 'Revel', 'revel', 'REVEL LANDEY'
'KIA' = 'Kia',
'RAM' = 'ram', 'Ram', 'RAM 550','RAMS'
'HUMMER' = 'HUMM',
'UNKNOWN' = 'UNKOWN', 'unk', 'Unknown', '-CAR/SU', 'UKN', 'unk.', 'unknown', 'UNKN',
'MACK' = 'mack', 'Mack', 'Mac', 'MACK DUMP TRUCK', 'MACK TRUCKS', 'mack truck', 'MIC', 'ma
```

```
         'mack dumptruck', 'MACK TRUCKS, INC.', 'Mac trailer',
'HYUNDAI' = 'HYU', 'HYUN', 'Hyundai',
'HINO' = 'hino', 'Hino', 'Hin', 'HINDO', 'HINO FLAT', 'HINO ND', 'HINO 999', 'HINO TRUCK',
"""
```

```
'\n'TOYT' = 'TOYOTA', 'toyota', 'TOY', 'TOYOTA FORKLIFT', 'TOYOTA PRIUS'\n'SUBA' =
'SUBAR','subaru'\n'CADI' = 'Cadillac','CADIL', 'cadil', \n'GMC' = 'GM', 'gmc', 'GMC
9', 'Gmc', 'GMC 999', 'GMC TANK',\n'DODGE' = 'DODG','DOGE', 'DOD', 'dodge', 'DONGE',
'Dodge RAM', \n'SUBN' = ''TRANS SUBN', 'subn', 'SUBN', 'suburban'\n'DUCATI' = 'DUC
A', 'ducati',  \n'TESLA' = 'TESL'\n'ISUZU'= 'ISU', 'IS', 'IZUZU', 'isu', 'isuzu', 'I
ZU', \n'MERZ' = 'MERCEDEZ BENZ', 'MER', 'mercedes benz', 'MERCEDES BENZ', 'Mercedes
Benz', 'MERCE','MERCEDES', 'MERCDES BENZ', 'me/benz', \n'ORION' = 'orion', 'Orio',
```

Vehicle make is represented by different name for the same maker. Inorder to properly analyze the data we have to change all the similar values to their group. We will be replacing the errors of the vehicle make of TOYT, CADI, SUBA, GMC to their original forms for analysis.

```
# Correcting the mistakes of VEHICLE_MAKE -- TOYT
data['VEHICLE_MAKE'] = data['VEHICLE_MAKE'].replace(['TOYOTA', 'toyota', 'TOY', 'TOYOTA FO
```

```
# Correcting the mistakes of VEHICLE_MAKE -- SUBA and HYUNDAI
data['VEHICLE_MAKE'] = data['VEHICLE_MAKE'].replace(['SUBAR','subaru', 'HYU', 'HYUN', 'Hyu
```

```
# Correcting the mistakes of VEHICLE_MAKE -- CADI and DODGE
data['VEHICLE_MAKE'] = data['VEHICLE_MAKE'].replace(['Cadillac','CADIL', 'cadil', 'DODG','
```

```
# Correcting the mistakes of VEHICLE_MAKE -- GMC
data['VEHICLE_MAKE'] = data['VEHICLE_MAKE'].replace(['GM', 'gmc', 'GMC9', 'Gmc', 'GMC 999'
```

```
data['VEHICLE_MAKE'].value_counts()
# We will also be correcting the mistakes of HONDA, NISSAN, FORD and CHEVROLET too as majo
```

```
TOYT                     115668
HOND                      81999
NISS                      68071
FORD                      54357
CHEV                      31072
                          ...
Gdan                          1
harley davison                1
REGINAL BUS OPERATIONS        1
fdny truck                    1
SEAGRAVE TOWERLADDER          1
Name: VEHICLE_MAKE, Length: 3643, dtype: int64
```

Different Ways in which the VEHICLE MAKE 'HONDA', 'NISSAN', 'FORD', 'CHEVROLET' are represented in the data.

```
"""
'HONDA' = 'HOND', 'honda', 'Honda 250', 'Honda'
'NISSAN'= 'NISS','NIS', 'NISSAN','NISSIAN ZX6K',
          'Nissan', 'UD / NISSAN', 'NISSAN DIESEL MOTOR',
          'NISSIAN','NISSAN DIESEL MOTOR VAN','Nissan Diesal Motor',
          'NISSAM N 20','nissan', 'NISSAN DIESEL MOTOR COMPANY','nissa'
'FORD' =  'FOR', 'Ford', 'ford', 'FORD F450', 'FORD CEMENT TRUCK',
          'FORD EC2', 'FORD XXX', 'FORD 550', 'FORD E350', 'FORD WAGON',
          'FORD UTILITY','FORD SUBN','ford econoline ambulance','FoRD',
          'FORD/AMBULANCE','ford van','ford transit', 'Ford ambulance',
          'Ford Transit', 'FORD RANGER', 'FORD DUMP TRUCK',
          'FORD AMBU', 'Ford van', 'ford f550 fdny nys ambulance',
          'Ford dump truck', 'FORD WHITE VAN','FORD TCN', 'FORD ECONOLINE',
          'Ford Taxi', 'FORD USPS2TON', 'Ford FDNY Ambulance', 'FORDA',
          'FORD / TRANSIT CONNECT', 'Ford EC3', 'FPRD', 'FORDAMBULANCE', 'FORD EC2'
'CHEVROLET' = 'CHEV', 'CHE', 'CHEVY', 'CHEVROLET EXPRESS', 'CHEVROLET VAN', 'chevy',
              'chevr', 'CHEVROLET EXPRESS YELLOW SUBURBAN', 'CHEVR', 'Chevrolet',
              'CHERVOLET', 'CHEVROLET EXP', 'CHEVY EXPRESS', 'chevrolet',
              'chevrolet commercial van', 'cheverleot', 'CHEVEROLET', 'chevrolet van',
              'CHEVY VAN', 'Chevolet', 'CHEV. GULF', 'CHEVROVELT', 'Chevr bus', 'CHEVY SIL
              'chevy express', 'Chevy express', 'cvevrolet', 'CHEVROLET AVALACHE', 'CHEVOR
"""
```

```
'\n'HONDA' = 'HOND', 'honda', 'Honda 250', 'Honda'\n'NISSAN'= 'NISS','NIS', 'NISSA
N','NISSIAN ZX6K', \n          'Nissan', 'UD / NISSAN', 'NISSAN DIESEL MOTOR',\n
'NISSIAN','NISSAN DIESEL MOTOR VAN','Nissan Diesal Motor',\n          'NISSAM N 2
0','nissan', 'NISSAN DIESEL MOTOR COMPANY','nissa'\n'FORD' =  'FOR', 'Ford', 'ford',
'FORD F450', 'FORD CEMENT TRUCK',\n          'FORD EC2', 'FORD XXX', 'FORD 550', 'FO
RD E350', 'FORD WAGON', \n          'FORD UTILITY','FORD SUBN','ford econoline ambul
ance','FoRD',\n          'FORD/AMBULANCE','ford van','ford transit', 'Ford ambulanc
```

```
#Replacing the errors for HONDA
data['VEHICLE_MAKE'] = data['VEHICLE_MAKE'].replace(['HOND', 'honda', 'Honda 250', 'Honda'
```

```
#Replacing the errors for NISSAN
data['VEHICLE_MAKE'] = data['VEHICLE_MAKE'].replace(['NISS','NIS', 'NISSAN','NISSIAN ZX6K'
                                                     'Nissan', 'UD / NISSAN', 'NISSAN DIES
                                                     'NISSIAN','NISSAN DIESEL MOTOR VAN','
                                                     'NISSAM N 20','nissan', 'NISSAN DIESE
```

```
#Replacing the errors for FORD
data['VEHICLE_MAKE'] = data['VEHICLE_MAKE'].replace(['FOR', 'Ford', 'ford', 'FORD F450', '
                                                     'FORD EC2', 'FORD XXX', 'FORD 550', '
                                                     'FORD UTILITY','FORD SUBN','ford econ
                                                     'FORD/AMBULANCE','ford van','ford tra
                                                     'Ford Transit', 'FORD RANGER', 'FORD
```

```
                                                 'FORD AMBU', 'Ford van', 'ford f550 f
                                                 'Ford dump truck', 'FORD WHITE VAN','
                                                 'Ford Taxi', 'FORD USPS2TON', 'Ford F
                                                 'FORD / TRANSIT CONNECT', 'Ford EC3',
```

```
#Replacing the errors for CHEVROLET
data['VEHICLE_MAKE'] = data['VEHICLE_MAKE'].replace(['CHEV', 'CHE', 'CHEVY', 'CHEVROLET EX
                                                 'chevr', 'CHEVROLET EXPRESS YELLOW SU
                                                 'CHERVOLET', 'CHEVROLET EXP', 'CHEVY 
                                                 'cheverleot', 'CHEVEROLET', 'chevrole
                                                 'CHEVROVELT', 'Chevr bus', 'CHEVY SIL
                                                 'CHEVROLET AVALACHE', 'CHEVORLET', 'C
```

```
data['VEHICLE_MAKE'].value_counts()
# We can see the difference in each of Vehicle make frequency.
```

```
TOYT                     115668
HONDA                     82025
NISSAN                    68321
FORD                      61437
CHEVROLET                 32690
                          ...
M2106                         1
Gdan                          1
harley davison                1
REGINAL BUS OPERATIONS        1
SEAGRAVE TOWERLADDER          1
Name: VEHICLE_MAKE, Length: 3560, dtype: int64
```

Pre processing for VEHICLE_TYPE Column

Several cleaning is to be done for Vehicle type, since same Vehicle type is represented in different ways.

```
data['VEHICLE_TYPE'].value_counts()
# There are huge no of different values we need to categorize this into 10 different types
```

```
Sedan                                  296310
Station Wagon/Sport Utility Vehicle    247628
Taxi                                    29050
Pick-up Truck                           20214
Box Truck                               14148
                                        ...
35 FT                                       1
TOWIN                                       1
VERIZ                                       1
posta                                       1
ROAD SWEEE                                  1
Name: VEHICLE_TYPE, Length: 741, dtype: int64
```

```
data['VEHICLE_TYPE'].unique()
```

```
'SWT', 'motor', 'posta', 'Schoo', 'TOWIN', 'Amb', 'MARK', 'DSNY',
'moter', 'Stree', 'Work', 'SKATE', 'POSTA', 'ARMY', 'PICK-',
'fork', 'APPOR', 'AMbul', 'Enclosed Body - Nonremovable Enclosure',
'NYCHA', 'SANITATION', 'DOLLY', 'vespa', 'POLIC', 'USPS2', 'U-TRU',
'SC', 'Trac', 'PASS', 'ken', 'CO', 'HORSE', 'fire', 'FRHT',
'crane', 'WINNE', 'NV150', 'REP', 'Motor Home', 'SEMI-', 'LTRL',
'Pas', 'Backh', 'CHEVY', 'sanit', 'acces', 'CATER', 'NTTRL', 'st',
'GATOR', 'Flat', 'OIL T', 'BLACK', 'VAV', 'rd/s', 'DOT T',
'Bucket Tru', 'Small', 'Liebh', 'speci', 'dp', 'mopd', 'CAMPE',
'c7c', 'const', 'Other', 'COUPE', 'S/SP', 'delv', 'Log', 'BUCKE',
'horse', 'Tractor', 'PEDIC', 'dsny', 'SELF-', 'rep', 'MOVING VAN',
'PSD', 'BK', 'van c', 'CONT', 'movin', 'E-SCO', 'BROOM', 'cate',
'PCH', 'CEMEN', 'VAN/T', 'van a', 'UTLL', 'nyc b', 'SWEEP', 'UNKN',
'conta', 'mecha', 'HARVE', 'POST', 'Const', 'RESCU', 'SUBN/',
'EXCAV', 'VESPA', 'NYC BUS', 'sedan', 'BTM', 'limo', 'COMER',
'18 WHEELER', 'bed', 'SANTI', 'EMS', 'D', 'PU', 'Attac', 'CAT P',
'picku', 'BROWN', 'TCR', 'wheel', 'DELVI', 'ECONO', 'L1', 'spec-',
'TRIM', 'EMT', 'GAS T', '2 HOR', 'FDNY EMS', 'escavator',
'SLINGSHOT', 'FDNY TRUCK', 'SPECIAL PU', 'unk', 'GARBAGE TR',
'TRACTOR', 'PICK UP', 'FORKLIFT', 'FRIEGHTLIN', 'FREIGHT FL',
'Firetruck', 'cross', 'PALFINGER', 'LIGHT TRAI', 'FDNY Ambul',
'government', 'suburban', 'dump truck', 'FDNY Truck', 'Front-Load',
'INTERNATIO', 'Pumper', 'TRUCK FLAT', 'Tractor tr', 'GENAMBUL',
'street cle', 'DELIVERY', '4dsd', 'FLATBED TR', 'JOHN DEERE',
'Dirt Bike', 'dilevery t', 'COURIER', 'PICKUP', 'STREET SWE',
'Work Van', 'uhaul truc', 'ford van', 'TRUCK VAN', 'AMAZON SPR',
'Postal Veh', 'box truck', '18 WEELER', 'Tow truck', 'Light trai',
'Tractor Tr', 'TR-Trailer', 'passenger', 'historical',
'PICKUP TRU', 'sanitation', 'MTA BUS', 'Golf Cart', 'food truck',
'Delivery', 'D/V WB', 'constructi', 'TOW TRUCK', 'UTIL WH',
'power shov', 'postal ser', 'BLU BUS', 'SCHOOLBUS', 'FDNY LADDE',
'E-BIKE', 'FDNY FIRE', 'omnibus', 'White ambu', 'BUs', 'TRANSPORT',
'SUBURBAN', 'bmw moped', 'LCOMM', 'DEPT VAN #', 'City MTA b',
'Road Sweep', 'TRANSIT VA', 'PICK RD', 'access a r', 'Horse Trai',
'US POSTAL', 'SEMI TRAIL', 'Piggy back', 'Utility', 'FDNY EMS V',
'Dump truck', 'FREIGHT TR', 'street swe', 'UTILITY', 'LCOM',
'USPS TRUCK', 'NYC DOT', 'TTRAILER', 'ESU RESCUE', 'tractor tr',
'SCHOOL BUS', 'FD TRUCK', 'Livery Omn', 'E450', 'pickup',
'delivery t', 'PICK-UP TR', 'tow trk', 'OMT', 'yellow cab',
'RV/VAN', 'AMBULETTE', 'DELIVERY V', 'SWEEPER', 'R/V',
'self insur', 'excavator', 'Chevy', 'POSTAL TRU', 'FDNY FIRET',
'ESCAVATOR', 'FIRE ENGIN', 'FORK LIFT', 'school bus', 'SPRINTER V',
'DUMP TRUCK', 'ORION', 'postal tru', 'Forklift t', 'HORSE CARR',
'ambulence', 'H1', 'amb', 'Wagon', 'NYS AMBULA', 'YELLOWPOWE',
'FDNY EMT', 'GOV', 'School bus', 'BULDOZER', 'FEDERAL EX',
'Utility.', 'RD BLDNG M', 'TANK WH', 'Ford Van', 'POWER SHOV',
'MTA', 'SELF INSUR', 'Fdny ambul', 'pc', '197209', 'sprinter v',
'FOOD TRUCK', 'semi-trail', 'util', 'G com', 'Fire Engin',
'TRACTOR TR', 'Unknown', 'TOUR BUS', 'flatbed', 'Mack',
'armored tr', 'ford econo', 'DOT TRUCK', 'HINO TANK', 'TL',
'SPINTER VA', 'FREIG DELV', 'MTA Bus', 'GOLF CART', 'ambulette',
'LIT DIRECT', 'T880', 'Cargo Truc', 'tow truck', 'Short Bus',
'SUBN WHI', 'pay loader', 'FLATBED', 'E-Scoter', 'BOX Truck', '0',
'BACK HOE', 'CHEVROLET', 'E-scooter', 'firetruck', 'HRSE', 'f-250',
'Pick up', 'Cargo Van', 'RDS', 'FDNY truck', 'Trc', 'camper tra',
'NYC FD', 'NYC AMBULA', 'F150XL PIC', 'WORK VAN', 'MECHANICAL',
'PC', 'UTILITY TR', 'JETSKI', 'ESCOVATOR', 'Tree cutte', '1C',
'GLP050VXEV', 'DELIVERY T', 'ROAD SWEEE'], dtype=object)
```

```
"""
SEDAN = '4 dr sedan', 'Sedan', 'CHEVROLET', 'ORION', 'Chevy', 'E450', 'sedan', '2 dr sedan
PASSENGER VEHICLE = 'NYC AMBULA', 'ambulette', 'SELF INSUR', 'Fdny ambul', 'pc', 'HORSE CA
                    'FDNY EMT', 'GOV', 'R/V', 'RV/VAN', 'AMBULETTE', 'SUBURBAN', 'White am
                    'government', 'suburban', 'SUBN/', 'PASS', 'AMbul', 'limo', 'COMER', '
                    'White', 'AMB', 'ROADS', 'RV', 'ambul', 'Motorized Home', 'SUBUR', 'SE
                    'MOBILE', 'WHITE', 'E350', 'FDNY Engin', 'E250', 'AMBULENCE, 'COMMERCI
SPORT UTILITY VEHICLE = 'Sport Utility Vehicle', 'STATION WAGON', 'Sport Utility Vehicle',
                        'firetruck', 'tow truck', 'ROAD SWEEE', 'GLP050VXEV', 'PC', 'MECHA
                        'HRSE', 'LIT DIRECT', 'T880', 'GOLF CART', 'RDS', 'TL', 'FREIG DEL
                        'BULDOZER', 'FEDERAL EX', 'Utility.', 'RD BLDNG M', 'TANK WH', 'Wa
                        'FDNY FIRET', 'ESCAVATOR', 'FIRE ENGIN', 'FORK LIFT', 'self insur'
                        'TTRAILER', 'ESU RESCUE', 'tractor tr', 'SEMI TRAIL', 'Piggy back'
                        'US POSTAL', 'FDNY FIRE', 'FDNY LADDE', 'D/V WB', 'constructi', 'T
                        'street cle','Firetruck', 'power shov', 'Tow truck', 'Light trai',
                        'PICKUP', 'STREET SWE', 'Work Van', 'uhaul truc', 'AMAZON SPR','Pu
                        'BROWN', 'TCR', 'wheel', 'ECONO', 'L1', 'spec-', 'TRIM', 'EMT', 'G
                        'PICK UP', 'FORKLIFT', 'Stree', 'Work', 'SKATE', 'POSTA', 'PICK-',
                        'DOLLY', 'POLIC', 'SC', 'Trac', 'ken', 'CO', 'HORSE', 'fire', 'FRH
                        'sanit', 'acces', 'CATER', 'NTTRL', 'st', 'GATOR', 'OIL T', 'BLACK
                        'c7c', 'const', 'Other', 'COUPE', 'S/SP', 'delv', 'Log', 'BUCKE',
                        'BROOM', 'cate', 'PCH', 'CEMEN', 'UTLL', 'SWEEP', 'conta', 'mecha'
                        'SPC', 'COM.', 'cater', 'Well Driller', 'Pickup with mounted Campe
                        'E - B', '52? t', 'SAFET', '12 Pa', 'LMB', 'LTR', 'VMS T', 'SE', '
                        'TRANS', 'FLAT', 'dump', "GOV'T", 'scava', 'santa', 'OML/', 'FORK'
                        'BOBCAT FOR', 'E REVEL SC', 'tow', 'Comm', 'COURI', 'Track', '7200
                        'fire truck', 'JLG L', 'Sanit', 'COMMU', 'wagon', 'EMRGN', 'E COM'
                        'Pick', 'Sprin', 'F650', 'WORK', 'SEA', 'CITY', 'comm.', 'axo', 'n
                        'commercial', 'ASTRO', 'City', 'MOVIN', 'ROAD SWEEP', 'TKTR', 'Hrs
                        'SWT', 'posta', 'TOWIN', 'DSNY', 'Station Wagon/Sport Utility Vehi
                        'Concrete Mixer', 'TRAILOR', 'TRAILER', 'Lift Boom', 'USPS', 'gato
                        'Crane', 'Flat Rack', '3-Door', 'FIRE', 'Tow Truck', 'TRK', 'tr',
                        'IP', 'Hopper', 'tour', 'TRACT', 'UTIL', 'Jeep', 'Forkl', 'DELIV',
                        'STREE', 'Tow', 'BULLD', 'Train', 'LIMOU', 'PICKU', 'PAS', 'POWER'
                        '\x7fomm', 'C1', 'Tow T', '38AB-', 'Deliv', 'Pallet', 'tract', 'Co
                        'Pedicab', 'Fire Truck', 'TOWTR', 'CRANE', 'utili', 'SKID', 'OMR',

UNKNOWN ='Unkno','UNKNO', 'UNKN', 'other', 'UNK'

TAXI = 'TAXI', 'Taxi', 'yellow cab', 'Chassis Cab','YELLO', 'Taxi'

VAN = 'WORK VAN', 'Cargo Van', 'ford econo', 'sprinter v', 'SPINTER VA', 'Ford Van', 'SPRI
      'ford van', 'TRUCK VAN', 'Postal Veh', 'DELVI', 'van c', 'MOVING VAN', 'Vanette', ,
      'deliv', 'SUV', 'van t', 'VAN/T', 'van a', 'Van', 'ICE CREAM', 'CARGO VAN', 'VAN/TRA

BIKE = 'Bike', 'E-scooter', 'E-BIKE', 'Dirt Bike', 'VESPA', 'E-SCO', 'BK', 'vespa', 'elect
       'E BIK', 'E-BIK', 'E-Bike', 'E-Bik', 'E-Sco', 'MOPED', 'SCOOT', 'E-SCOOTER', 'Moped
       'MOPD', 'E SCO', 'Scoot'

BUS = 'Bus', 'MTA Bus', 'Short Bus', 'TOUR BUS', 'MTA', 'School bus', 'school bus', 'SCHOO
      'MTA BUS', 'NYC BUS', 'nyc b', 'MTA B', 'mta b', 'Schoo' 'Bus', 'School Bus', 'SCHOO

TRUCK = 'Pick-up Truck', 'Truck', 'DELIVERY T', 'UTILITY TR', 'FDNY truck', 'BOX Truck','C
```

```
        'armored tr', 'FOOD TRUCK', 'postal tru', 'DUMP TRUCK', 'POSTAL TRU', 'FD TRUCK',
        'Dump truck', 'FREIGHT TR', 'street swe', 'UTILITY', 'LCOM', 'USPS TRUCK', 'food t
        'TRUCK FLAT', 'dump truck', 'FDNY Truck', 'Front-Load','FRIEGHTLIN', 'FREIGHT FL',
        '18 WHEELER','Bucket Tru', 'USPS2', 'U-TRU', 'ARMY', 'MARK', 'FREIGHTLIN', 'BoxTr'
        'TANK','box t','dumps','box', 'BOX T', 'INTER', 'BOX TRUCK', 'tank', 'Fd fi', 'Fre
        'WASTE', 'Flat', 'Enclosed Body - Removable Enclosure', 'flatb', 'FLAT/', 'FRIEG',
        'g spc', 'TOYOT', 'trlr', 'backh', 'firet','NYC', 'Tract', 'Stake or Rack', 'Bulk
        'FEDEX', 'GLBEN', 'mail', 'mack', 'GARBA', 'FDNY', 'Box T', '18 WEELER', 'box truc
        'FREIG', 'MAIL TRUCK', 'UPS TRUCK', 'Food', 'BOX', 'Truck', 'Flat Bed', 'FLAT BED'
        'BACKH', 'Armored Truck', 'PK', 'DUMP', 'TRAC', 'Beverage Truck', 'FIRETRUCK', 'Tr
        'Multi-Wheeled Vehicle', 'FRE T', 'UTILI', 'MAC T', 'DUMPT', 'garba', 'Tanker', 'P

MOTORCYCLE = 'Motorcycle', 'JETSKI', 'semi-trail', 'SEMI TRAIL', 'SEMI-','moter', 'semi',
BICYCLE  = 'Bicycle', 'BTM', 'Minicycle'
"""
```

```
'\nSEDAN = \'4 dr sedan\', \'Sedan\', \'CHEVROLET\', \'ORION\', \'Chevy\', \'E450\',
\'sedan\', \'2 dr sedan\', \'Sedan\',\'4 dr sedan\'\nPASSENGER VEHICLE = \'NYC AMBUL
A\', \'ambulette\', \'SELF INSUR\', \'Fdny ambul\', \'pc\', \'HORSE CARR\', \'ambule
nce\', \'H1\', \'amb\', \'NYS AMBULA\', \'YELLOWPOWE\', \n                    \'FDNY
EMT\', \'GOV\', \'R/V\', \'RV/VAN\', \'AMBULETTE\', \'SUBURBAN\', \'White ambu\',
\'passenger\', \'cross\', \'PALFINGER\', \'LIGHT TRAI\', \'FDNY Ambul\', \'Pas\',\n
\'government\', \'suburban\', \'SUBN/\', \'PASS\', \'AMbul\', \'limo\', \'COMER\',
```

Grouping the similar 'VEHICLE_TYPE' into a groups.

```
data['VEHICLE_TYPE'] = data['VEHICLE_TYPE'].replace(['NYC AMBULA', 'ambulette', 'SELF INSU
                    'FDNY EMT', 'GOV', 'R/V', 'RV/VAN', 'AMBULETTE', 'SUBURBAN', 'White am
                    'government', 'suburban', 'SUBN/', 'PASS', 'AMbul', 'limo', 'COMER', '
                    'White', 'AMB', 'ROADS', 'RV', 'ambul', 'Motorized Home', 'SUBUR', 'SE
                    'MOBILE', 'WHITE', 'E350', 'FDNY Engin', 'E250', 'AMBULENCE', 'NYC FD'
```

```
data['VEHICLE_TYPE'] = data['VEHICLE_TYPE'].replace(['Pick-up Truck', 'Truck', 'DELIVERY T
        'armored tr', 'FOOD TRUCK', 'postal tru', 'DUMP TRUCK', 'POSTAL TRU', 'FD TRUCK',
        'Dump truck', 'FREIGHT TR', 'street swe', 'UTILITY', 'LCOM', 'USPS TRUCK', 'food t
        'TRUCK FLAT', 'dump truck', 'FDNY Truck', 'Front-Load','FRIEGHTLIN', 'FREIGHT FL',
        '18 WHEELER','Bucket Tru', 'USPS2', 'U-TRU', 'ARMY', 'MARK', 'FREIGHTLIN', 'BoxTr'
        'TANK','box t','dumps','box', 'BOX T', 'INTER', 'BOX TRUCK', 'tank', 'Fd fi', 'Fre
        'WASTE', 'Flat', 'Enclosed Body - Removable Enclosure', 'flatb', 'FLAT/', 'FRIEG',
        'g spc', 'TOYOT', 'trlr', 'backh', 'firet','NYC', 'Tract', 'Stake or Rack', 'Bulk
        'FEDEX', 'GLBEN', 'mail', 'mack', 'GARBA', 'FDNY', 'Box T', '18 WEELER', 'box truc
        'FREIG', 'MAIL TRUCK', 'UPS TRUCK', 'Food', 'BOX', 'Truck', 'Flat Bed', 'FLAT BED'
        'BACKH', 'Armored Truck', 'PK', 'DUMP', 'TRAC', 'Beverage Truck', 'FIRETRUCK', 'Tr
        'Multi-Wheeled Vehicle', 'FRE T', 'UTILI', 'MAC T', 'DUMPT', 'garba', 'Tanker', 'P
```

```
data['VEHICLE_TYPE'] = data['VEHICLE_TYPE'].replace(['Bicycle', 'BTM', 'Minicycle'],'BICYC
```

```
data['VEHICLE_TYPE'] = data['VEHICLE_TYPE'].replace(['Motorcycle', 'JETSKI', 'semi-trail',
                                                   'motor', 'MOTOR', 'motorcycle', 'SEMI
```

```
data['VEHICLE_TYPE'] = data['VEHICLE_TYPE'].replace(['Bus', 'MTA Bus', 'Short Bus', 'TOUR
```

```
        'MTA BUS', 'NYC BUS', 'nyc b', 'MTA B', 'mta b', 'Schoo','omnibus', 'Bus', 'School B

data['VEHICLE_TYPE'] = data['VEHICLE_TYPE'].replace(['Bike', 'E-scooter', 'E-BIKE', 'Dirt
        'E BIK', 'E-BIK', 'E-Bike', 'E-Bik', 'E-Sco', 'MOPED', 'SCOOT', 'E-SCOOTER', 'Moped
        'MOPD', 'E SCO', 'Scoot', 'E-Scoter', 'E BIK'],'BIKE')


data['VEHICLE_TYPE'] = data['VEHICLE_TYPE'].replace(['WORK VAN', 'Cargo Van', 'ford econo'
       'ford van', 'TRUCK VAN', 'Postal Veh', 'DELVI', 'van c', 'MOVING VAN', 'Vanette', 'V
       'deliv', 'SUV', 'van t', 'VAN/T', 'van a', 'Van', 'ICE CREAM', 'CARGO VAN', 'VAN/TRA


data['VEHICLE_TYPE'] = data['VEHICLE_TYPE'].replace(['TAXI', 'Taxi', 'yellow cab', 'Chassi


data['VEHICLE_TYPE'] = data['VEHICLE_TYPE'].replace(['Unkno','UNKNO', 'UNKN', 'other', 'UN


# SPORT UTILITY VEHICLE INCLUDES SUV's as well as all kinds of utility vehicle.
data['VEHICLE_TYPE'] = data['VEHICLE_TYPE'].replace(['Sport Utility Vehicle', 'STATION WAG
                         'firetruck', 'tow truck', 'ROAD SWEEE', 'GLP050VXEV', 'PC', 'MECHA
                         'HRSE', 'LIT DIRECT', 'T880', 'GOLF CART', 'RDS', 'TL','BACK HOE',
                         'BULDOZER', 'FEDERAL EX', 'Utility.', 'RD BLDNG M', 'TANK WH', 'Wa
                         'FDNY FIRET', 'ESCAVATOR', 'FIRE ENGIN', 'FORK LIFT', 'self insur'
                         'TTRAILER', 'ESU RESCUE', 'tractor tr', 'SEMI TRAIL', 'Piggy back'
                         'US POSTAL', 'FDNY FIRE', 'FDNY LADDE', 'D/V WB', 'constructi', 'T
                         'street cle','Firetruck', 'power shov', 'Tow truck', 'Light trai',
                         'PICKUP', 'STREET SWE', 'Work Van', 'uhaul truc', 'AMAZON SPR','Pu
                         'BROWN', 'TCR', 'wheel', 'ECONO', 'L1', 'spec-', 'TRIM', 'EMT', 'G
                         'PICK UP', 'FORKLIFT', 'Stree', 'Work', 'SKATE', 'POSTA', 'PICK-',
                         'DOLLY', 'POLIC', 'SC', 'Trac', 'ken', 'CO', 'HORSE', 'fire', 'FRH
                         'sanit', 'acces', 'CATER', 'NTTRL', 'st', 'GATOR', 'OIL T', 'BLACK
                         'c7c', 'const', 'Other', 'COUPE', 'S/SP', 'delv', 'Log', 'BUCKE',
                         'BROOM', 'cate', 'PCH', 'CEMEN', 'UTLL', 'SWEEP', 'conta', 'mecha'
                         'SPC', 'COM.', 'cater', 'Well Driller', 'Pickup with mounted Campe
                         'E - B', '52? t', 'SAFET', '12 Pa', 'LMB', 'LTR', 'VMS T', 'SE', '
                         'TRANS', 'FLAT', 'dump', "GOV'T", 'scava', 'santa', 'OML/', 'FORK'
                         'BOBCAT FOR', 'E REVEL SC', 'tow', 'Comm', 'COURI', 'Track', '7200
                         'fire truck', 'JLG L', 'Sanit', 'COMMU', 'wagon', 'EMRGN', 'E COM'
                         'Pick', 'Sprin', 'F650', 'WORK', 'SEA', 'CITY', 'comm.', 'axo', 'n
                         'commercial', 'ASTRO', 'City', 'MOVIN', 'ROAD SWEEP', 'TKTR', 'Hrs
                         'SWT', 'posta', 'TOWIN', 'DSNY', 'Station Wagon/Sport Utility Vehi
                         'Concrete Mixer', 'TRAILOR', 'TRAILER', 'Lift Boom', 'USPS', 'gato
                         'Crane', 'Flat Rack', '3-Door', 'FIRE', 'Tow Truck', 'TRK', 'tr',
                         'IP', 'Hopper', 'tour', 'TRACT', 'UTIL', 'Jeep', 'Forkl', 'DELIV',
                         'STREE', 'Tow', 'BULLD', 'Train', 'LIMOU', 'PICKU', 'PAS', 'POWER'
                         '\x7fomm', 'C1', 'Tow T', '38AB-', 'Deliv', 'Pallet', 'tract', 'Co
                         'Pedicab', 'Fire Truck', 'TOWTR', 'CRANE', 'utili', 'SKID', 'OMR',


data['VEHICLE_TYPE'] = data['VEHICLE_TYPE'].replace(['4 dr sedan', 'F150XL PIC', 'Sedan',


data['VEHICLE_TYPE'].value_counts()
```

```
SEDAN                    296860
```

```
SPORT UTILITY VEHICLE    257926
TRUCK                     42894
TAXI                      29501
BUS                        9936
VAN                        4193
MOTORCYCLE                 3367
PASSENGER VEHICLE          2190
BIKE                        839
UNKNOWN                      15
BICYCLE                      10
Name: VEHICLE_TYPE, dtype: int64
```

```
# Now we can remove all the data with unknown VEHICLE_TYPE.
data.drop(data[data['VEHICLE_TYPE'] == 'UNKNOWN'].index, inplace = True)
```

```
data['VEHICLE_TYPE'].value_counts()
#Now it's clean and we can use this for Analysis 3. ie, Analysis of VEHICLE TYPE and the f
```

```
SEDAN                    296860
SPORT UTILITY VEHICLE    257926
TRUCK                     42894
TAXI                      29501
BUS                        9936
VAN                        4193
MOTORCYCLE                 3367
PASSENGER VEHICLE          2190
BIKE                        839
BICYCLE                      10
Name: VEHICLE_TYPE, dtype: int64
```

Taking the sample for sample Analysis.

```
sample_data = data.sample(100, random_state= 7301998) # Date of Birth is 30/0/1998
```

```
sample_data['VEHICLE_MAKE'].unique()
```

```
array(['VOLK', 'TOYT', 'FORD', 'MAC', 'YAMA', 'NISSAN', 'MITS', 'HONDA',
       'JEEP', 'GREYHOUND', 'MERZ', 'VOLV', 'LINC', 'GMC', 'INFI',
       'DODGE', 'CHEVROLET', 'AUDI', 'HINO', 'CHRY', 'BMW', 'NIU', 'SUBA',
       'LEXS', 'bus', 'hino', 'LNDR', 'ORION'], dtype=object)
```

```
sample_data['VEHICLE_MAKE'].value_counts()
```

```
TOYT         17
HONDA        14
NISSAN       12
FORD         10
CHEVROLET     5
INFI          4
DODGE         4
MERZ          3
GMC           3
```

```
VOLK          3
MAC           3
CHRY          2
LNDR          2
JEEP          2
HINO          2
BMW           2
NIU           1
hino          1
bus           1
LEXS          1
SUBA          1
GREYHOUND     1
VOLV          1
AUDI          1
YAMA          1
MITS          1
LINC          1
ORION         1
Name: VEHICLE_MAKE, dtype: int64
```

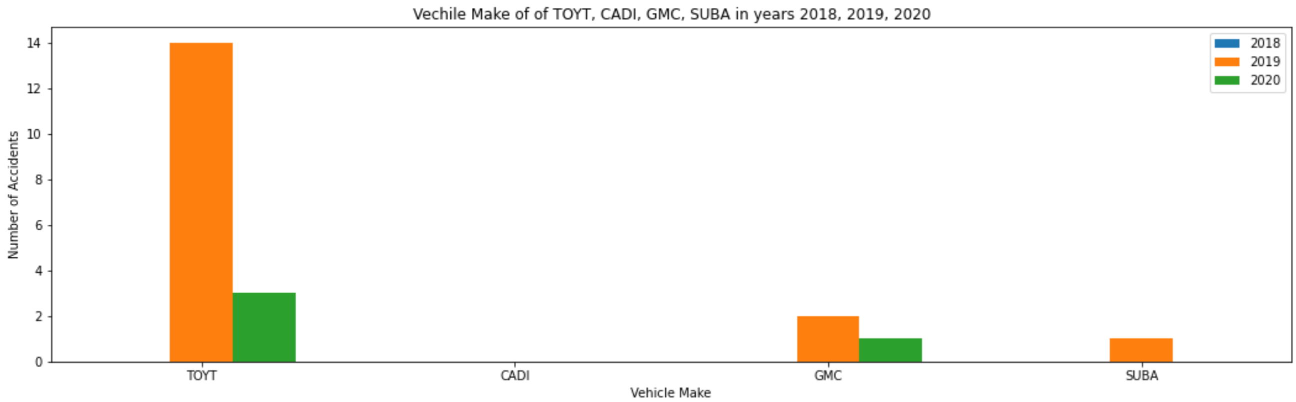
Analysis 1 (Vechile Make vs Accidents in year 2018, 2019, 2020) for sample data.

```
# Converting the data into 3 groups according to the year in which the accident happened.
# sample data of 2018 is represented by sd_2018 and similarly for other years.
sd_2018 = sample_data[sample_data['YEAR'] == 2018]
sd_2019 = sample_data[sample_data['YEAR'] == 2019]
sd_2020 = sample_data[sample_data['YEAR'] == 2020]
sd_2018.shape,sd_2019.shape, sd_2020.shape
```

```
((15, 26), (66, 26), (19, 26))
```

```
X = ['TOYT', 'CADI', 'GMC', 'SUBA'] # labels of to be plotted.
no_of_accidents_2018 = [sd_2018[sd_2018['VEHICLE_MAKE'] =='TOYT'].shape[0],
                        sd_2018[sd_2018['VEHICLE_MAKE'] =='CADI'].shape[0],
                        sd_2018[sd_2018['VEHICLE_MAKE'] =='GMC'].shape[0],
                        sd_2018[sd_2018['VEHICLE_MAKE'] =='SUBA'].shape[0]]
no_of_accidents_2019 = [sd_2019[sd_2019['VEHICLE_MAKE'] =='TOYT'].shape[0],
                        sd_2019[sd_2019['VEHICLE_MAKE'] =='CADI'].shape[0],
                        sd_2019[sd_2019['VEHICLE_MAKE'] =='GMC'].shape[0],
                        sd_2019[sd_2019['VEHICLE_MAKE'] =='SUBA'].shape[0]]
no_of_accidents_2020 = [sd_2020[sd_2020['VEHICLE_MAKE'] =='TOYT'].shape[0],
                        sd_2020[sd_2020['VEHICLE_MAKE'] =='CADI'].shape[0],
                        sd_2020[sd_2020['VEHICLE_MAKE'] =='GMC'].shape[0],
                        sd_2020[sd_2020['VEHICLE_MAKE'] =='SUBA'].shape[0]]

X_axis = np.arange( len(X))

plt.bar(X_axis-0.2, no_of_accidents_2018, 0.2, label = '2018')
plt.bar(X_axis, no_of_accidents_2019, 0.2, label = '2019')
```

```
plt.bar(X_axis + 0.2, no_of_accidents_2020, 0.2, label = '2020')

plt.xticks(X_axis, X)
plt.xlabel("Vehicle Make")
plt.ylabel("Number of Accidents" )
plt.title("Vechile Make of of TOYT, CADI, GMC, SUBA in years 2018, 2019, 2020")
plt.legend()
plt.show()
```




```
sd_2020[sd_2020['VEHICLE_MAKE'] =='GMC'].shape[0] #1
sd_2019[sd_2019['VEHICLE_MAKE'] =='GMC'].shape[0] #
#Hence our graph is correct since we are getting the same results as of value_counts of th
```

```
2
```

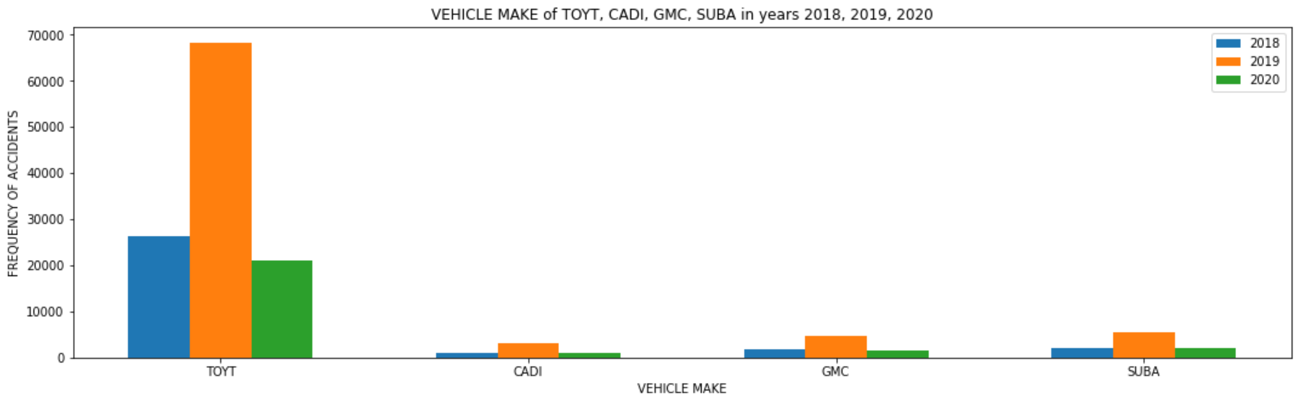
when we are comparing the value_counts of sample data with the data that we obtained from the graph we can understand that it is correct.

Analysis 1 (Vechile Make vs Accidents in year 2018, 2019, 2020) for the Orginal data.

```
# Converting the data into 3 groups according to the year in which the accident happened.
df_2018 = data[data['YEAR'] == 2018]
df_2019 = data[data['YEAR'] == 2019]
df_2020 = data[data['YEAR'] == 2020]
df_2018.shape,df_2019.shape, df_2020.shape
```

```
((141351, 26), (377495, 26), (128870, 26))
```

```
X = ['TOYT', 'CADI', 'GMC', 'SUBA'] # labels of to be plotted.
no_of_accidents_2018 = [df_2018[df_2018['VEHICLE_MAKE'] =='TOYT'].shape[0],
                        df_2018[df_2018['VEHICLE_MAKE'] =='CADI'].shape[0],
                        df_2018[df_2018['VEHICLE_MAKE'] =='GMC'].shape[0],
                        df_2018[df_2018['VEHICLE_MAKE'] =='SUBA'].shape[0]]
no_of_accidents_2019 = [df_2019[df_2019['VEHICLE_MAKE'] =='TOYT'].shape[0],
                        df_2019[df_2019['VEHICLE_MAKE'] =='CADI'].shape[0],
                        df_2019[df_2019['VEHICLE_MAKE'] =='GMC'].shape[0],
                        df_2019[df_2019['VEHICLE_MAKE'] =='SUBA'].shape[0]]
no_of_accidents_2020 = [df_2020[df_2020['VEHICLE_MAKE'] =='TOYT'].shape[0],
                        df_2020[df_2020['VEHICLE_MAKE'] =='CADI'].shape[0],
                        df_2020[df_2020['VEHICLE_MAKE'] =='GMC'].shape[0],
                        df_2020[df_2020['VEHICLE_MAKE'] =='SUBA'].shape[0]]

X_axis = np.arange( len(X))

plt.bar(X_axis-0.2, no_of_accidents_2018, 0.2, label = '2018')
plt.bar(X_axis, no_of_accidents_2019, 0.2, label = '2019')
plt.bar(X_axis + 0.2, no_of_accidents_2020, 0.2, label = '2020')

plt.xticks(X_axis, X)
plt.xlabel("VEHICLE MAKE")
plt.ylabel("FREQUENCY OF ACCIDENTS" )
plt.title("VEHICLE MAKE of TOYT, CADI, GMC, SUBA in years 2018, 2019, 2020")
plt.legend()
plt.show()
```



The accidents for TOYOTA is maximum in 2019 but we can't say since we only took 4 months of data of 2018 and

8 months for 2020. The conclusion that we can get is that TOYOTA is the vehicle that got into accidents mostly when compared with CADI, GMC and SUBA with a high margin.

Analysis 2 (Vechile Make vs Months) for sample data.

```
sample_data['VEHICLE_MAKE'].value_counts()
```

```
TOYT         17
HONDA        14
NISSAN       12
FORD         10
CHEVROLET     5
INFI          4
DODGE         4
MERZ          3
GMC           3
VOLK          3
MAC           3
CHRY          2
LNDR          2
JEEP          2
HINO          2
BMW           2
NIU           1
hino          1
bus           1
LEXS          1
SUBA          1
GREYHOUND     1
VOLV          1
AUDI          1
YAMA          1
MITS          1
LINC          1
ORION         1
Name: VEHICLE_MAKE, dtype: int64
```

```
# Split the dataframe into 4 dataframe using the vehicle make condition and then take the
sd_TOYT = sample_data[sample_data['VEHICLE_MAKE'] == 'TOYT']
sd_CADI = sample_data[sample_data['VEHICLE_MAKE'] == 'CADI']
sd_GMC = sample_data[sample_data['VEHICLE_MAKE'] == 'GMC']
sd_SUBA = sample_data[sample_data['VEHICLE_MAKE'] == 'SUBA']
```

```
sd_TOYT['MONTH'].value_counts()
```

```
1     3
2     3
6     3
11    2
```

```
12    2
10    1
7     1
3     1
5     1
Name: MONTH, dtype: int64
```

```
a = sd_TOYT['MONTH'].value_counts().to_dict()
print(a)
```

```
{1: 3, 2: 3, 6: 3, 11: 2, 12: 2, 10: 1, 7: 1, 3: 1, 5: 1}
```

```
x = [1, 2, 3, 4, 5, 6 ,7 ,8, 9, 10, 11, 12]
m ={k: 0 for v, k in enumerate(x)}
# Making a dictionary with keys as months and values as 0 which are going to be updated.
```

```
m.update(a)
print(m)
```

```
{1: 3, 2: 3, 3: 1, 4: 0, 5: 1, 6: 3, 7: 1, 8: 0, 9: 0, 10: 1, 11: 2, 12: 2}
```

```
# Function to return the frequency of accidents in each month with zero as values for mont
def monthly_accidents(Vehile_column):
  a = Vehile_column.value_counts().to_dict()
  x = [1, 2, 3, 4, 5, 6 ,7 ,8, 9, 10, 11, 12]
  m = {k: 0 for v, k in enumerate(x)}
  m.update(a)
  return list(m.values())

print(monthly_accidents(sd_TOYT['MONTH']))
```

```
[3, 3, 1, 0, 1, 3, 1, 0, 0, 1, 2, 2]
```

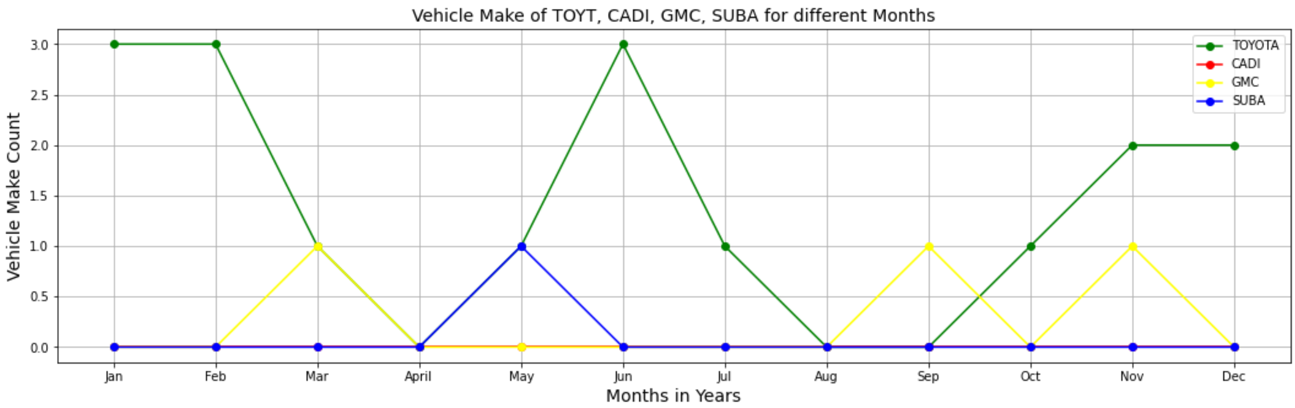
```
a = sd_CADI['MONTH'].value_counts().to_dict()
print(a)
# There is no value for CADI in the graph too as well as other values are plotted correctl
```

```
{}
```

```
X = [1, 2, 3, 4, 5, 6 ,7 ,8, 9, 10, 11, 12]
TOYT = monthly_accidents(sd_TOYT['MONTH'])
CADI = monthly_accidents(sd_CADI['MONTH'])
GMC = monthly_accidents(sd_GMC['MONTH'])
SUBA = monthly_accidents(sd_SUBA['MONTH'])
```

```
plot1, = plt.plot(X, TOYT, color='green', marker='o')
plot2, = plt.plot(X, CADI, color='red', marker='o')
plot3, = plt.plot(X, GMC, color='yellow', marker='o')
plot4, = plt.plot(X, SUBA, color='blue', marker='o')
plt.rcParams["figure.figsize"] = (18,10)
```

```
plt.xticks(ticks =X, labels = ['Jan', 'Feb', 'Mar', 'April', 'May', 'Jun', 'Jul', 'Aug', '
plt.title('Vehicle Make of TOYT, CADI, GMC, SUBA for different Months', fontsize=14)
plt.xlabel('Months in Years', fontsize=14)
plt.ylabel('Vehicle Make Count', fontsize=14)
plt.legend([plot1, plot2, plot3, plot4], ['TOYOTA', 'CADI', 'GMC', 'SUBA'])
plt.grid(True)
plt.show()
```



Analysis 2 (Vechile Make vs Months) for Orginal data.

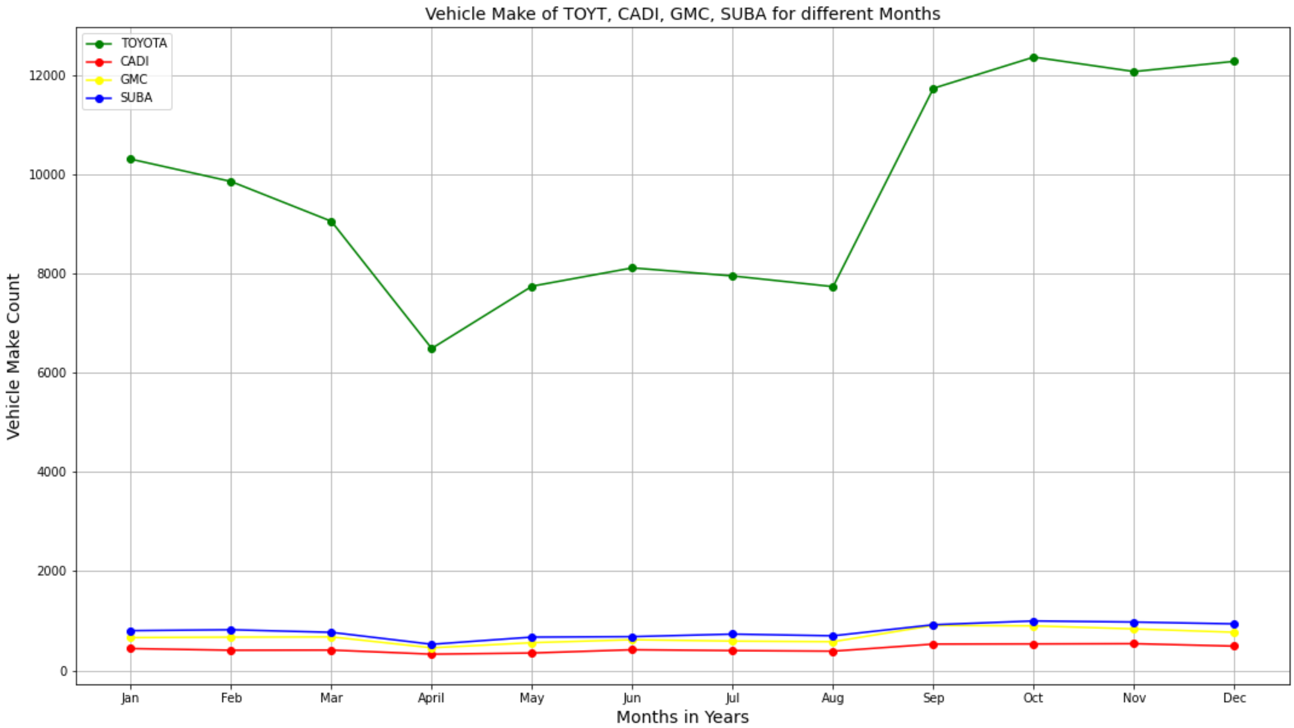
```
# Split the dataframe into 4 dataframe using the vehicle make condition and then take the
df_TOYT = data[data['VEHICLE_MAKE'] == 'TOYT']
df_CADI = data[data['VEHICLE_MAKE'] == 'CADI']
df_GMC = data[data['VEHICLE_MAKE'] == 'GMC']
df_SUBA = data[data['VEHICLE_MAKE'] == 'SUBA']
```

```
df_TOYT['MONTH'].value_counts()
```

```
10    12360
12    12275
11    12068
9     11731
1     10304
2      9853
3      9051
6      8112
7      7949
5      7743
8      7734
4      6488
Name: MONTH, dtype: int64
```

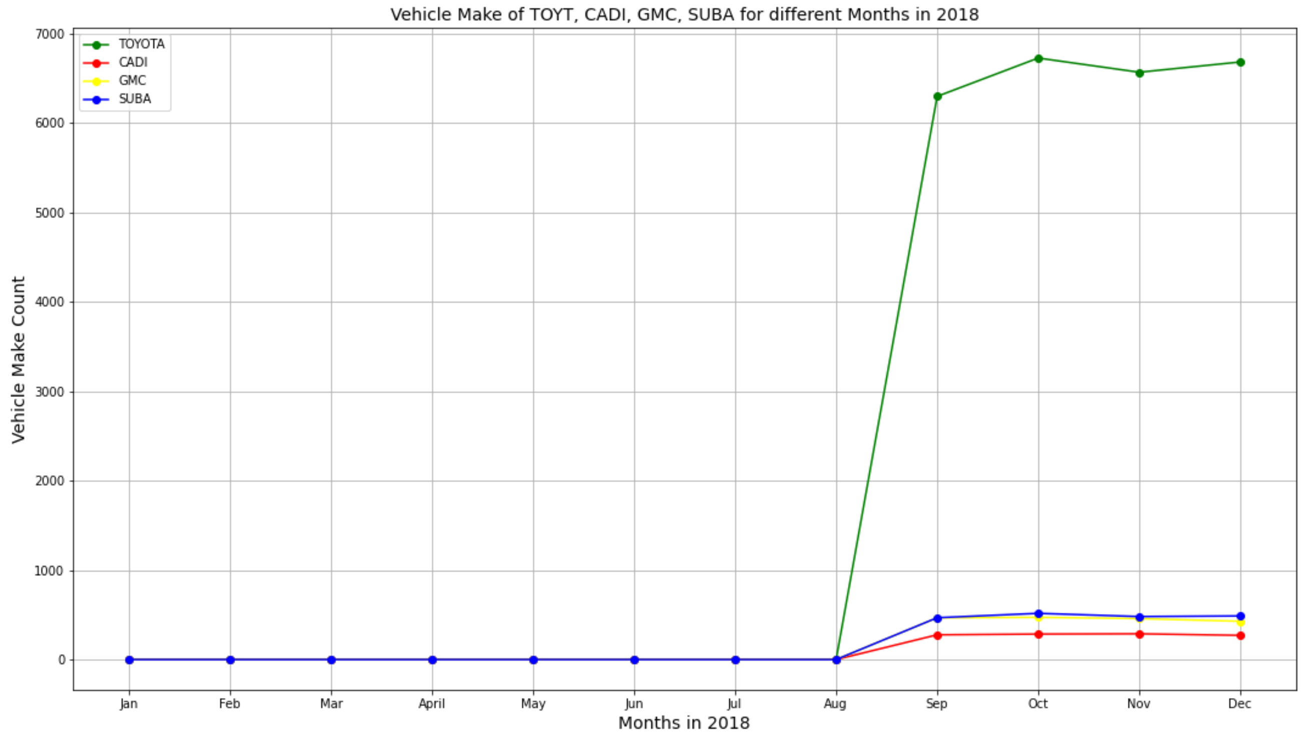
```
X = [1, 2, 3, 4, 5, 6 ,7 ,8, 9, 10, 11, 12]
```

```
TOYT = monthly_accidents(df_TOYT['MONTH'])
CADI = monthly_accidents(df_CADI['MONTH'])
GMC = monthly_accidents(df_GMC['MONTH'])
SUBA = monthly_accidents(df_SUBA['MONTH'])


plot1, = plt.plot(X, TOYT, color='green', marker='o')
plot2, = plt.plot(X, CADI, color='red', marker='o')
plot3, = plt.plot(X, GMC, color='yellow', marker='o')
plot4, = plt.plot(X, SUBA, color='blue', marker='o')
plt.rcParams["figure.figsize"] = (18,10)
plt.xticks(ticks =X, labels = ['Jan', 'Feb', 'Mar', 'April', 'May', 'Jun', 'Jul', 'Aug', '
plt.title('Vehicle Make of TOYT, CADI, GMC, SUBA for different Months', fontsize=14)
plt.xlabel('Months in Years', fontsize=14)
plt.ylabel('Vehicle Make Count', fontsize=14)
plt.legend([plot1, plot2, plot3, plot4], ['TOYOTA', 'CADI', 'GMC', 'SUBA'])
plt.grid(True)
plt.show()
```



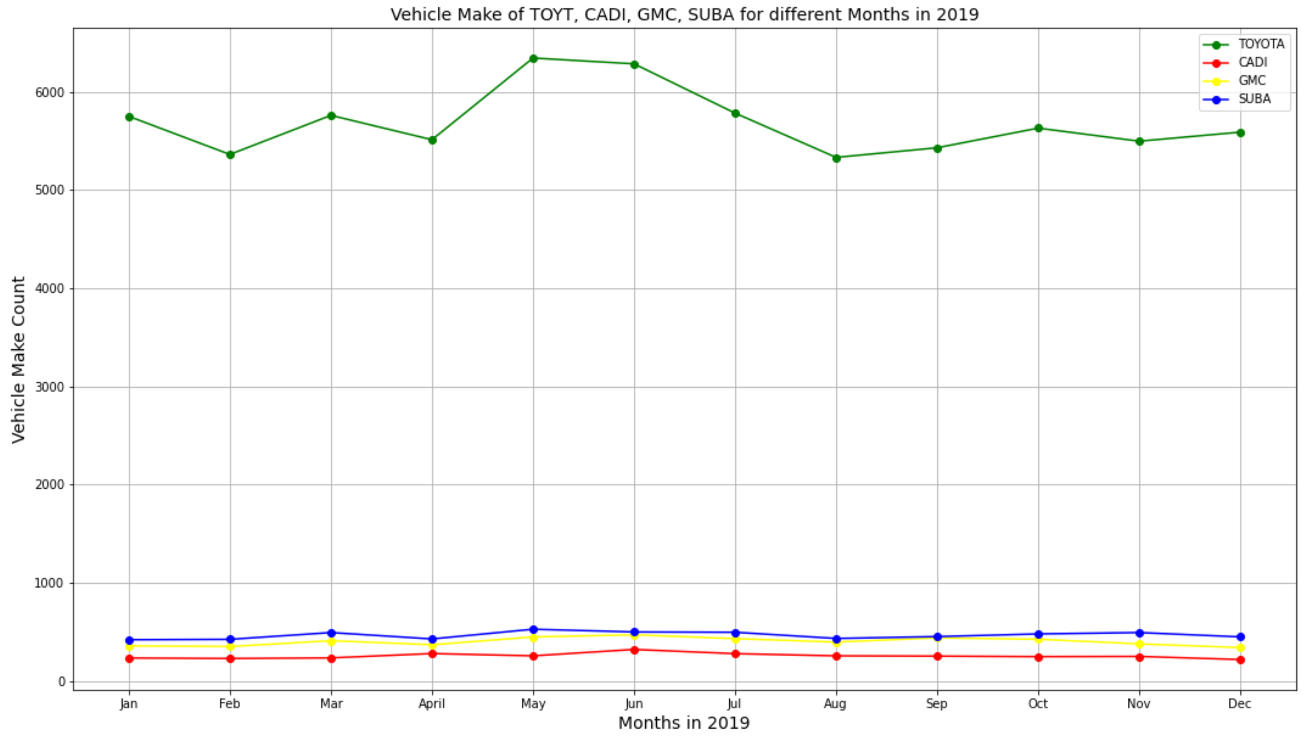
We can identify from the line graph that VEHICLE MAKE TOYATA has the maximum no of accidents in this period and other VEHICLE MAKE are almost similar. VEHICLE MAKE CADI is the one which has least accidents. We are not able to properly tell which month has more accidents since the data interval we took contains 4 months from 2018, All the months of 2019, and 8 months of 2020. So we need to plot year wise monthly analysis.

```
df_2018_TOYT = df_2018[df_2018['VEHICLE_MAKE'] == 'TOYT']
df_2018_CADI = df_2018[df_2018['VEHICLE_MAKE'] == 'CADI']
df_2018_GMC = df_2018[df_2018['VEHICLE_MAKE'] == 'GMC']
df_2018_SUBA = df_2018[df_2018['VEHICLE_MAKE'] == 'SUBA']

X = [1, 2, 3, 4, 5, 6 ,7 ,8, 9, 10, 11, 12]
TOYT = monthly_accidents(df_2018_TOYT['MONTH'])
CADI = monthly_accidents(df_2018_CADI['MONTH'])
GMC = monthly_accidents(df_2018_GMC['MONTH'])
SUBA = monthly_accidents(df_2018_SUBA['MONTH'])


plot1, = plt.plot(X, TOYT, color='green', marker='o')
plot2, = plt.plot(X, CADI, color='red', marker='o')
plot3, = plt.plot(X, GMC, color='yellow', marker='o')
plot4, = plt.plot(X, SUBA, color='blue', marker='o')
plt.rcParams["figure.figsize"] = (18,10)
plt.xticks(ticks =X, labels = ['Jan', 'Feb', 'Mar', 'April', 'May', 'Jun', 'Jul', 'Aug', '
plt.title('Vehicle Make of TOYT, CADI, GMC, SUBA for different Months in 2018', fontsize=1
plt.xlabel('Months in 2018', fontsize=14)
plt.ylabel('Vehicle Make Count', fontsize=14)
plt.legend([plot1, plot2, plot3, plot4], ['TOYOTA', 'CADI', 'GMC', 'SUBA'])
plt.grid(True)
plt.show()
```



Toyota is the VEHICLE MAKE that got into accidents mostly in the months of 2018.

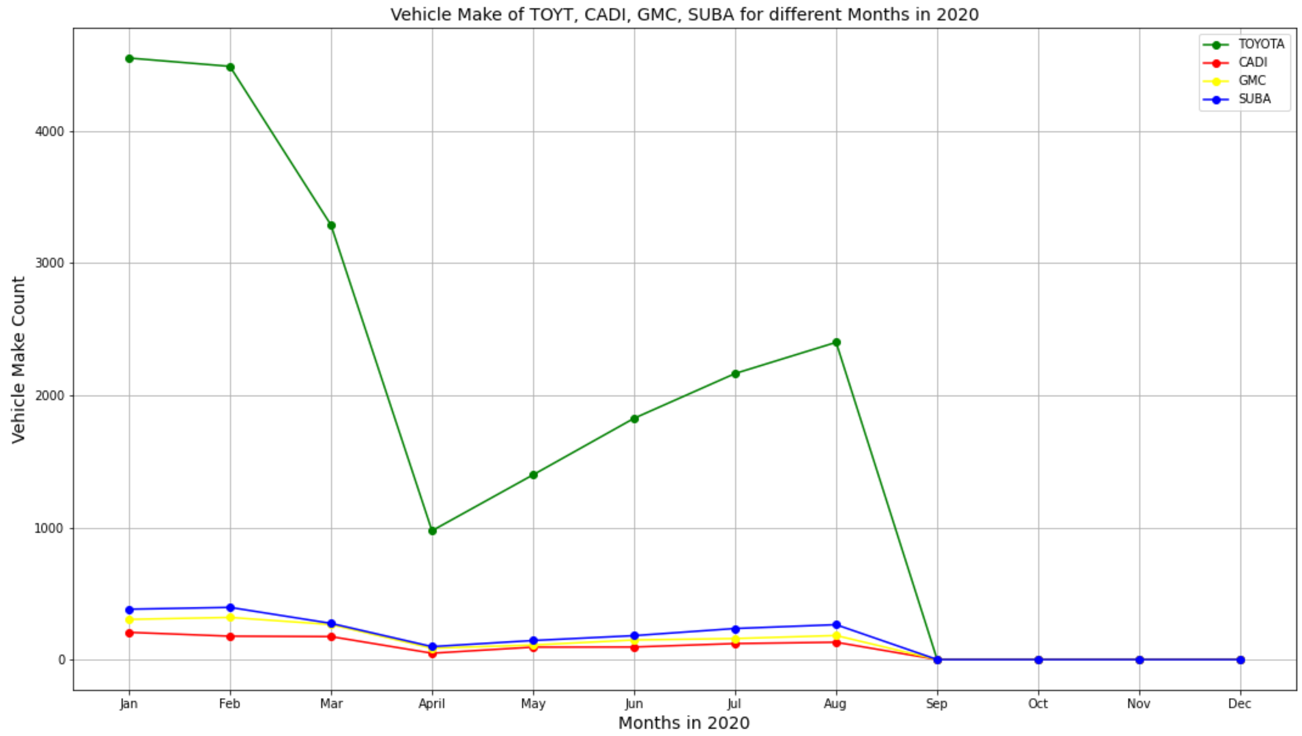
```
df_2019_TOYT =df_2019[df_2019['VEHICLE_MAKE'] == 'TOYT']
df_2019_CADI =df_2019[df_2019['VEHICLE_MAKE'] == 'CADI']
df_2019_GMC =df_2019[df_2019['VEHICLE_MAKE'] == 'GMC']
df_2019_SUBA =df_2019[df_2019['VEHICLE_MAKE'] == 'SUBA']

X = [1, 2, 3, 4, 5, 6 ,7 ,8, 9, 10, 11, 12]
TOYT = monthly_accidents(df_2019_TOYT['MONTH'])
CADI = monthly_accidents(df_2019_CADI['MONTH'])
GMC = monthly_accidents(df_2019_GMC['MONTH'])
SUBA = monthly_accidents(df_2019_SUBA['MONTH'])


plot1, = plt.plot(X, TOYT, color='green', marker='o')
plot2, = plt.plot(X, CADI, color='red', marker='o')
plot3, = plt.plot(X, GMC, color='yellow', marker='o')
plot4, = plt.plot(X, SUBA, color='blue', marker='o')
plt.rcParams["figure.figsize"] = (18,10)
plt.xticks(ticks =X, labels = ['Jan', 'Feb', 'Mar', 'April', 'May', 'Jun', 'Jul', 'Aug', '
plt.title('Vehicle Make of TOYT, CADI, GMC, SUBA for different Months in 2019', fontsize=1
plt.xlabel('Months in 2019', fontsize=14)
plt.ylabel('Vehicle Make Count', fontsize=14)
plt.legend([plot1, plot2, plot3, plot4], ['TOYOTA', 'CADI', 'GMC', 'SUBA'])
plt.grid(True)
plt.show()
```



Maximum accidents occured in the month of May in 2019 for TOYOTA, But for all the months the accidents of TOYOTA is very high compared to all other VEHICLE MAKE.

```
df_2020_TOYT =df_2020[df_2020['VEHICLE_MAKE'] == 'TOYT']
df_2020_CADI =df_2020[df_2020['VEHICLE_MAKE'] == 'CADI']
df_2020_GMC =df_2020[df_2020['VEHICLE_MAKE'] == 'GMC']
df_2020_SUBA =df_2020[df_2020['VEHICLE_MAKE'] == 'SUBA']

X = [1, 2, 3, 4, 5, 6 ,7 ,8, 9, 10, 11, 12]
TOYT = monthly_accidents(df_2020_TOYT['MONTH']) # Returns a list of values of 12 months.
CADI = monthly_accidents(df_2020_CADI['MONTH'])
GMC = monthly_accidents(df_2020_GMC['MONTH'])
SUBA = monthly_accidents(df_2020_SUBA['MONTH'])


plot1, = plt.plot(X, TOYT, color='green', marker='o')
plot2, = plt.plot(X, CADI, color='red', marker='o')
plot3, = plt.plot(X, GMC, color='yellow', marker='o')
plot4, = plt.plot(X, SUBA, color='blue', marker='o')
```

```
plt.rcParams["figure.figsize"] = (18,10)
plt.xticks(ticks =X, labels = ['Jan', 'Feb', 'Mar', 'April', 'May', 'Jun', 'Jul', 'Aug', '
plt.title('Vehicle Make of TOYT, CADI, GMC, SUBA for different Months in 2020', fontsize=1
plt.xlabel('Months in 2020', fontsize=14)
plt.ylabel('Vehicle Make Count', fontsize=14)
plt.legend([plot1, plot2, plot3, plot4], ['TOYOTA', 'CADI', 'GMC', 'SUBA'])
plt.grid(True)
plt.show()
```



https://www.driversautomart.com/why-is-the-toyota-brand-so-popular-among-consumers/

1. Toyota is very popular vehicle in usa as it builds solid, efficient, and reliable vehicles as per consumer reports. This can be the main reason for increased no of accidents as the VEHICLE MAKE 'TOYATA' is used by a major population. Thus, our analysis of the data is valid.

In initial months of 2020, we can see that all the VEHICLE MAKE accidents got declined rapidly. This decline in accidents is due to impact of COVID-19 pandemic. We can see from the above graph that, the accidents started declining from January and reached a bottom threshold at April.

https://en.wikipedia.org/wiki/COVID-19_pandemic_in_New_York_City

According to the data from internet we can see that the coronavirus has been spreading in newyork city from january.

1. **By March 29, over 30,000 cases were confirmed**
2. **Starting March 16, New York City schools were closed.**
3. **On March 20, the New York State governor's office issued an executive order closing "non-essential" businesses.**

These were the reasons for maximum rate of decline in accidents in the month of March and

Analysis 3 (Vechile TYPE vs Accidents Frequency) for sample data.

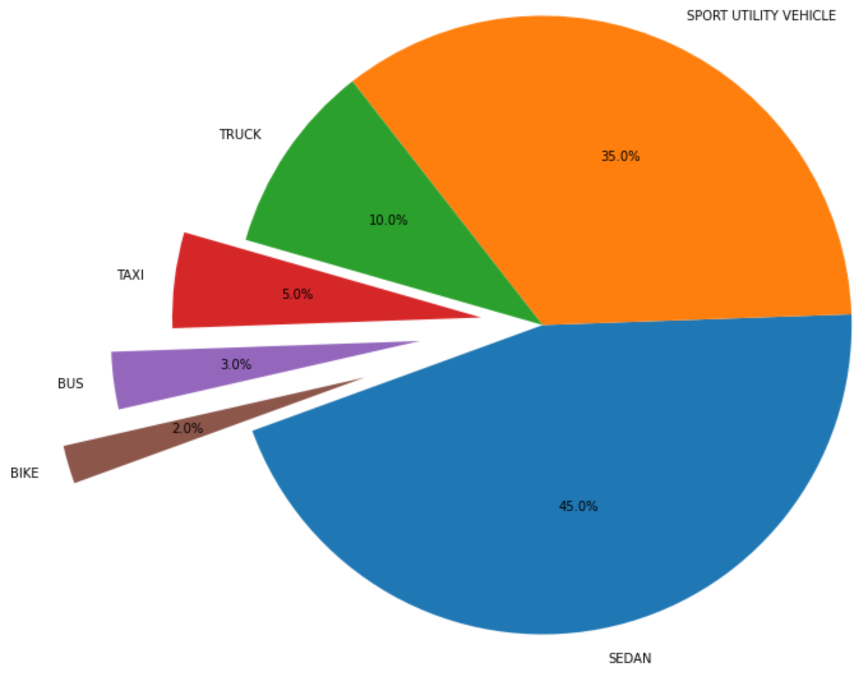
```
sample_data['VEHICLE_TYPE'].value_counts()
```

```
SEDAN                   45
SPORT UTILITY VEHICLE   35
TRUCK                   10
TAXI                     5
BUS                      3
BIKE                     2
Name: VEHICLE_TYPE, dtype: int64
```

```
data_dict = sample_data['VEHICLE_TYPE'].value_counts().to_dict() # Converting the value co
labels = []
sizes = []
K = int(input('Enter the no of portion of the pie chart to be exploded out ')) # We can gi
p = float(input('The width in which exploding to happen (0.1-0.5)'))
for x, y in data_dict.items():
    labels.append(x)
    sizes.append(y)
explode = list(np.zeros(len(sizes))) # Made a list of 0 for explode equal to the size of s
small_indexes = sorted(range(len(sizes)), key = lambda sub: sizes[sub])[:K] # An algorithm
for count, ele in enumerate(small_indexes[::-1], 1):
  explode[ele] = p* count # Respective index values of the explode gets replace by the p.

fig1, ax1 = plt.subplots()
ax1.pie(sizes, explode=explode, labels=labels, autopct='%1.1f%%', shadow=False, startangle
```

```
ax1.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle.

plt.show()
```

```
Enter the no of portion of the pie chart to be exploded out 3
The width in which exploding to happen (0.1-0.5).2
```



Analysis 3 (Vechile TYPE vs Accidents Frequency) for the data between 1st September 2018 to 31st August 2020.

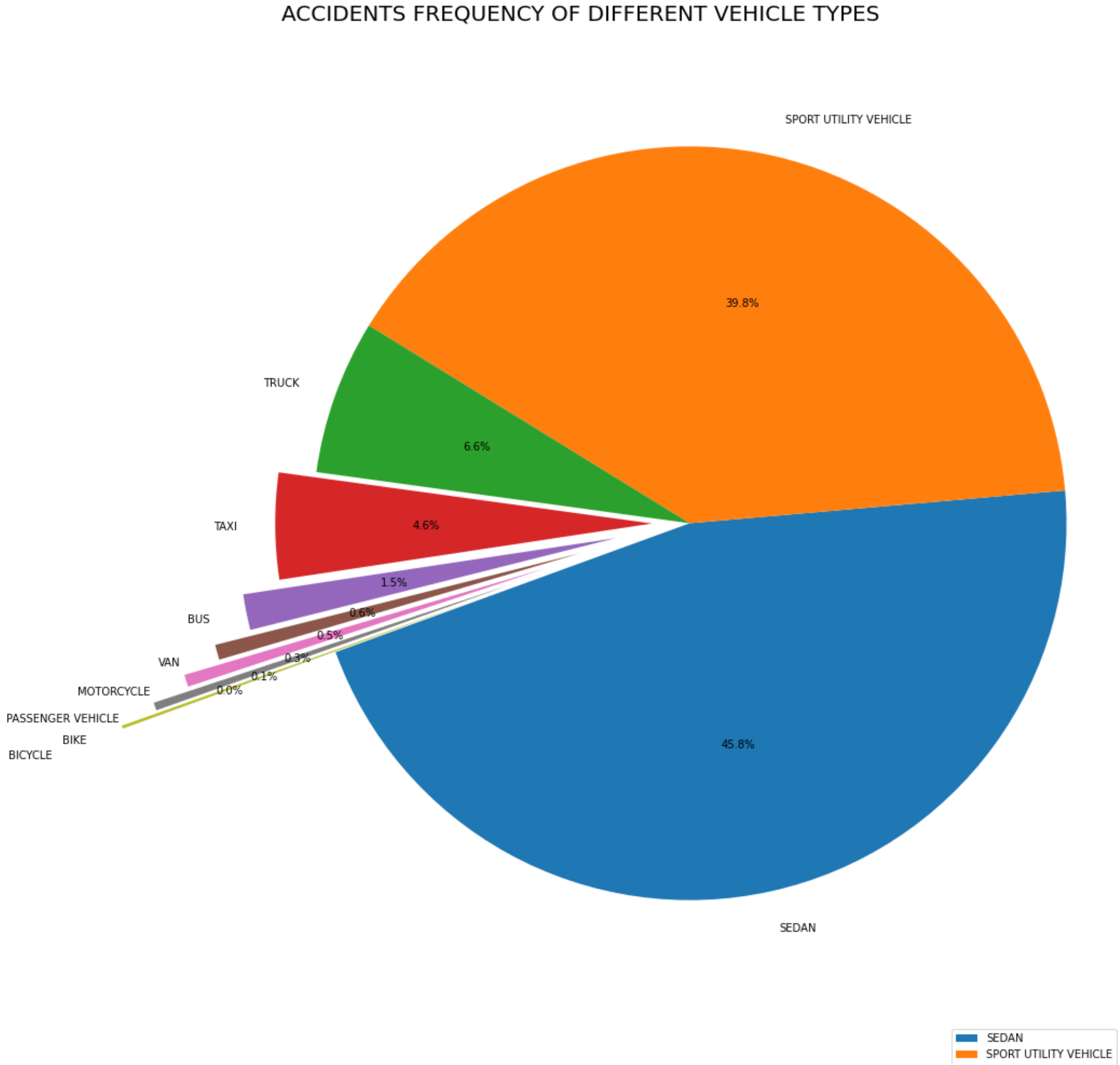
```
data['VEHICLE_TYPE'].value_counts()
```

```
SEDAN                    296860
SPORT UTILITY VEHICLE    257926
TRUCK                     42894
TAXI                      29501
BUS                        9936
VAN                        4193
MOTORCYCLE                 3367
PASSENGER VEHICLE          2190
BIKE                        839
```

```
    BICYCLE                    10
    Name: VEHICLE_TYPE, dtype: int64
```

```
data_dict = data['VEHICLE_TYPE'].value_counts().to_dict() # Converting the value counts of

labels = []
sizes = []

K = int(input('Enter the no of portion of the pie chart to be exploded out ')) # We can gi
p = float(input('The width in which exploding to happen (0.1-1)'))
for x, y in data_dict.items():
    labels.append(x)
    sizes.append(y)
explode = list(np.zeros(len(sizes))) # Made a list of 0 for explode equal to the size of s
small_indexes = sorted(range(len(sizes)), key = lambda sub: sizes[sub])[:K] # An algorithm
for count, ele in enumerate(small_indexes[::-1], 1):
  explode[ele] = p* count # Respective index values of the explode gets replace by the p.
plt.rcParams["figure.figsize"] = (18, 17) # Increasing the size of the pie chart.

fig1, ax1 = plt.subplots()
pie = ax1.pie(sizes, explode=explode, labels=labels, autopct='%1.1f%%', shadow=False, star

ax1.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle.

plt.title(label='ACCIDENTS FREQUENCY OF DIFFERENT VEHICLE TYPES', fontsize=20)
plt.legend(pie[0],labels, bbox_to_anchor=(1,0), loc='best')
plt.show()
```

```
Enter the no of portion of the pie chart to be exploded out 7
The width in which exploding to happen (0.1-1).1
```



From the pie chart we can see that the maximum accidents were occured by SEDAN and the least by BICYCLE.

```
cleaned_data = data.to_csv('/content/drive/MyDrive/Cleaned_data.csv')
```

✓ 15s completed at 23:13