In [ ]:

CHICAGO CITY ACCIDENT PREDICTION

INTRODUCTION: PROJECT OVERVIEW

The city of chicago has noted a spike in the number of accidents in the past year(2023). It aims to identify the major causes of the accidents inorder to ways by which they caan reduce the number of accidents

Stakeholder: The primary stakeholder in this scenario is the Vehicle Safety Board of Chicago. They are launching a new campaign with the goal of reducing car crashes and injuries in the city.

Business Problem: The city of chicago main goal is to be able to come up withsolutions to help end or reduce number of accidents. The city of chicago seeks to develop a predictive model that will help indentify the major causes of accidents

Possible question

1. Does lack of traffic control device influence accidents and injuries?
2. In which weather conditions do most injuries and accidents occur?
3. Does the number of years since manufacture of a car influence the accidents?
4. In most accidents and injuries occuring do they relate to a certain age group of drivers?

DATA UNDERSTANDING

Dataset used contains 3 data sets one having details on crashes, on people and on vehicles. Crashes database: holds information of crashes recorded in the since 2013, contains the conditins that contributed to the crashes,as well as the extrtent of injuries and location among afew Peoples database: holds information abouut people that wer invonved in thae accidents rrecorded, their age, gender,licence registration status, their physical conditions Vehicles database: holds information on the vehicles that were involved in the various crashes in terms of their make, model, physical appearance, dimension as well as the year of registration

Data Exploration

- Loading the data
- Data cleaning
- Merging the data
- Creating bins

In [2]:
```
#importing liblaries
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.impute import SimpleImputer
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_s
```

In [3]:
```
# load the traffic crashes dataset

Crashes = pd.read_csv("Traffic_Crashes_-_Crashes_20240208.csv", low_memory=Fals
Crashes.head()
```

Out[3]:

	CRASH_RECORD_ID	CRASH_DATE_EST_I	CRASH_DATE	POST
0	23a79931ef555d54118f64dc9be2cf2dbf59636ce253f7...	NaN	09/05/2023 07:05:00 PM	
1	2675c13fd0f474d730a5b780968b3cafc7c12d7adb661f...	NaN	09/22/2023 06:45:00 PM	
2	5f54a59fcb087b12ae5b1acff96a3caf4f2d37e79f8db4...	NaN	07/29/2023 02:45:00 PM	
3	7ebf015016f83d09b321afd671a836d6b148330535d5df...	NaN	08/09/2023 11:00:00 PM	
4	6c1659069e9c6285a650e70d6f9b574ed5f64c12888479...	NaN	08/18/2023 12:50:00 PM	

5 rows × 48 columns

In [4]:
```
# checking the null values in the dataset
Crashes.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 804163 entries, 0 to 804162
Data columns (total 48 columns):
 #   Column                           Non-Null Count   Dtype  
---  ------                           --------------   -----  
 0   CRASH_RECORD_ID                  804163 non-null  object 
 1   CRASH_DATE_EST_I                 60229 non-null   object 
 2   CRASH_DATE                       804163 non-null  object 
 3   POSTED_SPEED_LIMIT               804163 non-null  int64  
 4   TRAFFIC_CONTROL_DEVICE           804163 non-null  object 
 5   DEVICE_CONDITION                 804163 non-null  object 
 6   WEATHER_CONDITION                804163 non-null  object 
 7   LIGHTING_CONDITION               804163 non-null  object 
 8   FIRST_CRASH_TYPE                 804163 non-null  object 
 9   TRAFFICWAY_TYPE                  804163 non-null  object 
 10  LANE_CNT                         199007 non-null  float64
 11  ALIGNMENT                        804163 non-null  object 
 12  ROADWAY_SURFACE_COND             804163 non-null  object 
 13  ROAD_DEFECT                      804163 non-null  object 
 14  REPORT_TYPE                      780500 non-null  object 
 15  CRASH_TYPE                       804163 non-null  object 
 16  INTERSECTION_RELATED_I           184348 non-null  object 
 17  NOT_RIGHT_OF_WAY_I               37115 non-null   object 
 18  HIT_AND_RUN_I                    251501 non-null  object 
 19  DAMAGE                           804163 non-null  object 
 20  DATE_POLICE_NOTIFIED             804163 non-null  object 
 21  PRIM_CONTRIBUTORY_CAUSE          804163 non-null  object 
 22  SEC_CONTRIBUTORY_CAUSE           804163 non-null  object 
 23  STREET_NO                        804163 non-null  int64  
 24  STREET_DIRECTION                 804159 non-null  object 
 25  STREET_NAME                      804162 non-null  object 
 26  BEAT_OF_OCCURRENCE               804158 non-null  float64
 27  PHOTOS_TAKEN_I                   10504 non-null   object 
 28  STATEMENTS_TAKEN_I               17803 non-null   object 
 29  DOORING_I                        2474 non-null    object 
 30  WORK_ZONE_I                      4628 non-null    object 
 31  WORK_ZONE_TYPE                   3591 non-null    object 
 32  WORKERS_PRESENT_I                1184 non-null    object 
 33  NUM_UNITS                        804163 non-null  int64  
 34  MOST_SEVERE_INJURY               802392 non-null  object 
 35  INJURIES_TOTAL                   802404 non-null  float64
 36  INJURIES_FATAL                   802404 non-null  float64
 37  INJURIES_INCAPACITATING          802404 non-null  float64
 38  INJURIES_NON_INCAPACITATING      802404 non-null  float64
 39  INJURIES_REPORTED_NOT_EVIDENT    802404 non-null  float64
 40  INJURIES_NO_INDICATION           802404 non-null  float64
 41  INJURIES_UNKNOWN                 802404 non-null  float64
 42  CRASH_HOUR                       804163 non-null  int64  
 43  CRASH_DAY_OF_WEEK                804163 non-null  int64  
 44  CRASH_MONTH                      804163 non-null  int64  
 45  LATITUDE                         798674 non-null  float64
 46  LONGITUDE                        798674 non-null  float64
 47  LOCATION                         798674 non-null  object 
dtypes: float64(11), int64(6), object(31)
memory usage: 294.5+ MB
```

In [5]:
```
# dropping columns with the highest null values and will not have effect in our
columns_drop = ['LONGITUDE','INTERSECTION_RELATED_I','NOT_RIGHT_OF_WAY_I','HIT_
Crashes2 = Crashes.drop(columns=columns_drop)
Crashes2.head(5)
```

Out[5]:

	CRASH_RECORD_ID	CRASH_DATE	POSTED_SPEED_LIMIT	TRA
0	23a79931ef555d54118f64dc9be2cf2dbf59636ce253f7...	09/05/2023 07:05:00 PM	30	
1	2675c13fd0f474d730a5b780968b3cafc7c12d7adb661f...	09/22/2023 06:45:00 PM	50	
2	5f54a59fcb087b12ae5b1acff96a3caf4f2d37e79f8db4...	07/29/2023 02:45:00 PM	30	
3	7ebf015016f83d09b321afd671a836d6b148330535d5df...	08/09/2023 11:00:00 PM	30	
4	6c1659069e9c6285a650e70d6f9b574ed5f64c12888479...	08/18/2023 12:50:00 PM	15	

In [6]:
```
# checking the null values after cleaning
Crashes2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 804163 entries, 0 to 804162
Data columns (total 19 columns):
 #   Column                  Non-Null Count   Dtype
---  ------                  --------------   -----
 0   CRASH_RECORD_ID         804163 non-null  object
 1   CRASH_DATE              804163 non-null  object
 2   POSTED_SPEED_LIMIT      804163 non-null  int64
 3   TRAFFIC_CONTROL_DEVICE  804163 non-null  object
 4   DEVICE_CONDITION        804163 non-null  object
 5   WEATHER_CONDITION       804163 non-null  object
 6   LIGHTING_CONDITION      804163 non-null  object
 7   ROAD_DEFECT             804163 non-null  object
 8   CRASH_TYPE              804163 non-null  object
 9   DAMAGE                  804163 non-null  object
 10  PRIM_CONTRIBUTORY_CAUSE 804163 non-null  object
 11  SEC_CONTRIBUTORY_CAUSE  804163 non-null  object
 12  STREET_NAME             804162 non-null  object
 13  MOST_SEVERE_INJURY      802392 non-null  object
 14  INJURIES_TOTAL          802404 non-null  float64
 15  CRASH_HOUR              804163 non-null  int64
 16  CRASH_DAY_OF_WEEK       804163 non-null  int64
 17  CRASH_MONTH             804163 non-null  int64
 18  LOCATION                798674 non-null  object
dtypes: float64(1), int64(4), object(14)
memory usage: 116.6+ MB
```

In [7]:
```
# dropping rows in location that have nullvalues
Crashes3 = Crashes2.dropna(subset = ['LOCATION'])
```

In [8]:
```
# checking the final values of the dataset and datatypes
Crashes3.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 798674 entries, 2 to 804162
Data columns (total 19 columns):
 #   Column                   Non-Null Count   Dtype  
---  ------                   --------------   -----  
 0   CRASH_RECORD_ID          798674 non-null  object 
 1   CRASH_DATE               798674 non-null  object 
 2   POSTED_SPEED_LIMIT       798674 non-null  int64  
 3   TRAFFIC_CONTROL_DEVICE   798674 non-null  object 
 4   DEVICE_CONDITION         798674 non-null  object 
 5   WEATHER_CONDITION        798674 non-null  object 
 6   LIGHTING_CONDITION       798674 non-null  object 
 7   ROAD_DEFECT              798674 non-null  object 
 8   CRASH_TYPE               798674 non-null  object 
 9   DAMAGE                   798674 non-null  object 
 10  PRIM_CONTRIBUTORY_CAUSE  798674 non-null  object 
 11  SEC_CONTRIBUTORY_CAUSE   798674 non-null  object 
 12  STREET_NAME              798674 non-null  object 
 13  MOST_SEVERE_INJURY       796915 non-null  object 
 14  INJURIES_TOTAL           796927 non-null  float64
 15  CRASH_HOUR               798674 non-null  int64  
 16  CRASH_DAY_OF_WEEK        798674 non-null  int64  
 17  CRASH_MONTH              798674 non-null  int64  
 18  LOCATION                 798674 non-null  object 
dtypes: float64(1), int64(4), object(14)
memory usage: 121.9+ MB
```

In [12]:
```
# loading the second dataset of vehicles involved in accidents
Vehicles = pd.read_csv("Traffic_Crashes_-_Vehicles_20240208.csv", low_memory=Fa
Vehicles.head()
```

Out[12]:

	CRASH_UNIT_ID	CRASH_RECORD_ID	CRASH_DATE	UNIT_NO
0	1727162	f5943b05f46b8d4148a63b7506a59113eae0cf1075aabc...	12/21/2023 08:57:00 AM	2
1	1717556	7b1763088507f77e0e552c009a6bf89a4d6330c7527706...	12/06/2023 03:24:00 PM	1
2	1717574	2603ff5a88f0b9b54576934c5ed4e4a64e8278e005687b...	12/06/2023 04:00:00 PM	2
3	1717579	a52ef70e33d468b855b5be44e8638a564434dcf99c0edf...	12/06/2023 04:30:00 PM	1
4	1720118	609055f4b1a72a44d6ec40ba9036cefd7c1287a755eb6c...	12/10/2023 12:12:00 PM	1

5 rows × 71 columns

In [13]:
```
# checking the null values
Vehicles.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1640449 entries, 0 to 1640448
Data columns (total 71 columns):
 #   Column                    Non-Null Count    Dtype
---  ------                    --------------    -----
 0   CRASH_UNIT_ID             1640449 non-null  int64
 1   CRASH_RECORD_ID           1640449 non-null  object
 2   CRASH_DATE                1640449 non-null  object
 3   UNIT_NO                   1640449 non-null  int64
 4   UNIT_TYPE                 1638343 non-null  object
 5   NUM_PASSENGERS            242654 non-null   float64
 6   VEHICLE_ID                1602902 non-null  float64
 7   CMRC_VEH_I                30550 non-null    object
 8   MAKE                      1602897 non-null  object
 9   MODEL                     1602752 non-null  object
 10  LIC_PLATE_STATE           1455956 non-null  object
 11  VEHICLE_YEAR              1344942 non-null  float64
 12  VEHICLE_DEFECT            1602902 non-null  object
 13  VEHICLE_TYPE              1602902 non-null  object
```

In [22]:
```
# dropping columns that have the highest null values
columns_drop = ['HAZMAT_CLASS','MCS_OUT_OF_SERVICE_I','HAZMAT_OUT_OF_SERVICE_I
Vehicles = Vehicles.drop(columns=columns_drop)
Vehicles.head(5)
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
Cell In[22], line 3
      1 # dropping columns that have the highest null values
      2 columns_drop = ['HAZMAT_CLASS','MCS_OUT_OF_SERVICE_I','HAZMAT_OUT_OF_
SERVICE_I','LOAD_TYPE','CARGO_BODY_TYPE','VEHICLE_CONFIG','AXLE_CNT','TOTAL_V
EHICLE_LENGTH','TRAILER2_LENGTH','TRAILER1_LENGTH','TRAILER2_WIDTH','TRAILER1
_WIDTH','WIDE_LOAD_I','IDOT_PERMIT_NO','MCS_VIO_CAUSE_CRASH_I','HAZMAT_VIO_CA
USE_CRASH_I','MCS_REPORT_NO','MCS_REPORT_I','HAZMAT_REPORT_NO','HAZMAT_REPORT
_I','HAZMAT_PRESENT_I','UN_NO','HAZMAT_NAME','HAZMAT_PLACARDS_I','CARRIER_CIT
Y','CARRIER_STATE','CARRIER_NAME','GVWR','COMMERCIAL_SRC','ILCC_NO','CCMC_N
O','USDOT_NO','CMV_ID','EXCEED_SPEED_LIMIT_I','TOWED_BY','TOWED_TO', 'AREA_00
_I','AREA_01_I','AREA_02_I','AREA_03_I','AREA_04_I','AREA_05_I','AREA_06_
I','AREA_07_I','AREA_08_I','AREA_09_I','AREA_10_I','AREA_11_I','AREA_12_I','A
REA_99_I','FIRE_I','TRAVEL_DIRECTION','LIC_PLATE_STATE','CMRC_VEH_I','NUM_PAS
SENGERS','CRASH_UNIT_ID','TOWED_I','CRASH_DATE','FIRST_CONTACT_POINT']
----> 3 Vehicles = Vehicles.drop(columns=columns_drop)
      4 Vehicles.head(5)

File ~\anaconda3\Lib\site-packages\pandas\core\frame.py:5258, in DataFrame.dr
op(self, labels, axis, index, columns, level, inplace, errors)
   5110 def drop(
   5111     self,
   5112     labels: IndexLabel = None,
   (...)
   5119     errors: IgnoreRaise = "raise",
   5120 ) -> DataFrame | None:
   5121     """
   5122     Drop specified labels from rows or columns.
   5123
   (...)
   5256             weight  1.0     0.8
   5257     """
-> 5258     return super().drop(
   5259         labels=labels,
   5260         axis=axis,
   5261         index=index,
   5262         columns=columns,
   5263         level=level,
   5264         inplace=inplace,
   5265         errors=errors,
   5266     )

File ~\anaconda3\Lib\site-packages\pandas\core\generic.py:4549, in NDFrame.dr
op(self, labels, axis, index, columns, level, inplace, errors)
   4547 for axis, labels in axes.items():
   4548     if labels is not None:
-> 4549         obj = obj._drop_axis(labels, axis, level=level, errors=error
s)
   4551 if inplace:
   4552     self._update_inplace(obj)

File ~\anaconda3\Lib\site-packages\pandas\core\generic.py:4591, in NDFrame._d
rop_axis(self, labels, axis, level, errors, only_slice)
   4589         new_axis = axis.drop(labels, level=level, errors=errors)
   4590     else:
-> 4591         new_axis = axis.drop(labels, errors=errors)
```

```
   4592     indexer = axis.get_indexer(new_axis)
   4594 # Case for non-unique axis
   4595 else:

File ~\anaconda3\Lib\site-packages\pandas\core\indexes\base.py:6699, in Inde
x.drop(self, labels, errors)
   6697 if mask.any():
   6698     if errors != "ignore":
-> 6699         raise KeyError(f"{list(labels[mask])} not found in axis")
   6700     indexer = indexer[~mask]
   6701 return self.delete(indexer)

KeyError: "['HAZMAT_CLASS', 'MCS_OUT_OF_SERVICE_I', 'HAZMAT_OUT_OF_SERVICE_
I', 'LOAD_TYPE', 'CARGO_BODY_TYPE', 'VEHICLE_CONFIG', 'AXLE_CNT', 'TOTAL_VEHI
CLE_LENGTH', 'TRAILER2_LENGTH', 'TRAILER1_LENGTH', 'TRAILER2_WIDTH', 'TRAILER
1_WIDTH', 'WIDE_LOAD_I', 'IDOT_PERMIT_NO', 'MCS_VIO_CAUSE_CRASH_I', 'HAZMAT_V
IO_CAUSE_CRASH_I', 'MCS_REPORT_NO', 'MCS_REPORT_I', 'HAZMAT_REPORT_NO', 'HAZM
AT_REPORT_I', 'HAZMAT_PRESENT_I', 'UN_NO', 'HAZMAT_NAME', 'HAZMAT_PLACARDS_
I', 'CARRIER_CITY', 'CARRIER_STATE', 'CARRIER_NAME', 'GVWR', 'COMMERCIAL_SR
C', 'ILCC_NO', 'CCMC_NO', 'USDOT_NO', 'CMV_ID', 'EXCEED_SPEED_LIMIT_I', 'TOWE
D_BY', 'TOWED_TO', 'AREA_00_I', 'AREA_01_I', 'AREA_02_I', 'AREA_03_I', 'AREA_
04_I', 'AREA_05_I', 'AREA_06_I', 'AREA_07_I', 'AREA_08_I', 'AREA_09_I', 'AREA
_10_I', 'AREA_11_I', 'AREA_12_I', 'AREA_99_I', 'FIRE_I', 'TRAVEL_DIRECTION',
'LIC_PLATE_STATE', 'CMRC_VEH_I', 'NUM_PASSENGERS', 'CRASH_UNIT_ID', 'TOWED_
I', 'CRASH_DATE', 'FIRST_CONTACT_POINT'] not found in axis"
```

In [23]:
```
# checking the dataset after dropping the null columns
Vehicles.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1640449 entries, 0 to 1640448
Data columns (total 12 columns):
 #   Column           Non-Null Count    Dtype
---  ------           --------------    -----
 0   CRASH_RECORD_ID  1640449 non-null  object
 1   UNIT_NO          1640449 non-null  int64
 2   UNIT_TYPE        1638343 non-null  object
 3   VEHICLE_ID       1602902 non-null  float64
 4   MAKE             1602897 non-null  object
 5   MODEL            1602752 non-null  object
 6   VEHICLE_YEAR     1344942 non-null  float64
 7   VEHICLE_DEFECT   1602902 non-null  object
 8   VEHICLE_TYPE     1602902 non-null  object
 9   VEHICLE_USE      1602902 non-null  object
 10  MANEUVER         1602902 non-null  object
 11  OCCUPANT_CNT     1602902 non-null  float64
dtypes: float64(3), int64(1), object(8)
memory usage: 150.2+ MB
```

In [24]:
```
# dropping the rows in the vehicle year that have null values
Vehicles3 = Vehicles.dropna(subset = ['VEHICLE_YEAR'])
```

In [25]:
```
# checking the cleaned dataset datatypes
Vehicles3.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 1344942 entries, 1 to 1640447
Data columns (total 12 columns):
 #   Column           Non-Null Count    Dtype  
---  ------           --------------    -----  
 0   CRASH_RECORD_ID  1344942 non-null  object 
 1   UNIT_NO          1344942 non-null  int64  
 2   UNIT_TYPE        1344932 non-null  object 
 3   VEHICLE_ID       1344942 non-null  float64
 4   MAKE             1344937 non-null  object 
 5   MODEL            1344805 non-null  object 
 6   VEHICLE_YEAR     1344942 non-null  float64
 7   VEHICLE_DEFECT   1344942 non-null  object 
 8   VEHICLE_TYPE     1344942 non-null  object 
 9   VEHICLE_USE      1344942 non-null  object 
 10  MANEUVER         1344942 non-null  object 
 11  OCCUPANT_CNT     1344942 non-null  float64
dtypes: float64(3), int64(1), object(8)
memory usage: 133.4+ MB
```

In [26]:
```
# loading the people involved in accidents dataset
People = pd.read_csv("Traffic_Crashes_-_People_20240208.csv", low_memory=False)
People.head()
```

Out[26]:

	PERSON_ID	PERSON_TYPE	CRASH_RECORD_ID	VEHICLE_ID
0	O749947	DRIVER	81dc0de2ed92aa62baccab641fa377be7feb1cc47e6554...	834816.0
1	O871921	DRIVER	af84fb5c8d996fcd3aefd36593c3a02e6e7509eeb27568...	827212.0
2	O10018	DRIVER	71162af7bf22799b776547132ebf134b5b438dcf3dac6b...	9579.0
3	O10038	DRIVER	c21c476e2ccc41af550b5d858d22aaac4ffc88745a1700...	9598.0
4	O10039	DRIVER	eb390a4c8e114c69488f5fb8a097fe629f5a92fd528cf4...	9600.0

5 rows × 29 columns

In [27]:
```
# checking the dataset shape
People.shape
```

Out[27]:
```
(1765148, 29)
```

In [28]:
```
# checking the datatypes of the dataset
People.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1765148 entries, 0 to 1765147
Data columns (total 29 columns):
 #   Column                 Dtype  
---  ------                 -----  
 0   PERSON_ID              object 
 1   PERSON_TYPE            object 
 2   CRASH_RECORD_ID        object 
 3   VEHICLE_ID             float64
 4   CRASH_DATE             object 
 5   SEAT_NO                float64
 6   CITY                   object 
 7   STATE                  object 
 8   ZIPCODE                object 
 9   SEX                    object 
 10  AGE                    float64
 11  DRIVERS_LICENSE_STATE  object 
 12  DRIVERS_LICENSE_CLASS  object 
 13  SAFETY_EQUIPMENT       object 
 14  AIRBAG_DEPLOYED        object 
 15  EJECTION               object 
 16  INJURY_CLASSIFICATION  object 
 17  HOSPITAL               object 
 18  EMS_AGENCY             object 
 19  EMS_RUN_NO             object 
 20  DRIVER_ACTION          object 
 21  DRIVER_VISION          object 
 22  PHYSICAL_CONDITION     object 
 23  PEDPEDAL_ACTION        object 
 24  PEDPEDAL_VISIBILITY    object 
 25  PEDPEDAL_LOCATION      object 
 26  BAC_RESULT             object 
 27  BAC_RESULT VALUE       float64
 28  CELL_PHONE_USE         object 
dtypes: float64(4), object(25)
memory usage: 390.5+ MB
```

In [29]:
```
# checking the null values in the dataset
#get the count of null values in each column:
null_summary = People.isnull().sum()
data_types = People.dtypes
# Combine null values and data types into a summary DataFrame
summary_People = pd.DataFrame({'Null Values': null_summary, 'Data Types': data_
print(summary_People)
```

```
                       Null Values Data Types
PERSON_ID                        0     object
PERSON_TYPE                      0     object
CRASH_RECORD_ID                  0     object
VEHICLE_ID                   35294    float64
CRASH_DATE                       0     object
SEAT_NO                    1408309    float64
CITY                        479735     object
STATE                       461035     object
ZIPCODE                     586327     object
SEX                          29048     object
AGE                         515185    float64
DRIVERS_LICENSE_STATE       730369     object
DRIVERS_LICENSE_CLASS       898888     object
SAFETY_EQUIPMENT              4924     object
AIRBAG_DEPLOYED              33947     object
EJECTION                     21835     object
INJURY_CLASSIFICATION          718     object
HOSPITAL                   1471525     object
EMS_AGENCY                 1583351     object
EMS_RUN_NO                 1735026     object
DRIVER_ACTION               360007     object
DRIVER_VISION               360529     object
PHYSICAL_CONDITION          359047     object
PEDPEDAL_ACTION            1731573     object
PEDPEDAL_VISIBILITY        1731636     object
PEDPEDAL_LOCATION          1731576     object
BAC_RESULT                  359026     object
BAC_RESULT VALUE           1763160    float64
CELL_PHONE_USE             1763989     object
```

In [30]:
```
# dropping the columns with the highest null values
columns_drop = ['CELL_PHONE_USE','BAC_RESULT VALUE','BAC_RESULT','PEDPEDAL_LOCA
People2 = People.drop(columns=columns_drop)
People2.head(5)
```

Out[30]:

	PERSON_ID	PERSON_TYPE	CRASH_RECORD_ID	CITY	S
0	O749947	DRIVER	81dc0de2ed92aa62baccab641fa377be7feb1cc47e6554...	CHICAGO	
1	O871921	DRIVER	af84fb5c8d996fcd3aefd36593c3a02e6e7509eeb27568...	CHICAGO	
2	O10018	DRIVER	71162af7bf22799b776547132ebf134b5b438dcf3dac6b...	NaN	
3	O10038	DRIVER	c21c476e2ccc41af550b5d858d22aaac4ffc88745a1700...	NaN	
4	O10039	DRIVER	eb390a4c8e114c69488f5fb8a097fe629f5a92fd528cf4...	NaN	

In [31]:
```
# checking the nullvalues again after dropping the highest numbers
null_summary = People2.isnull().sum()
data_types = People2.dtypes
# Combine null values and data types into a summary DataFrame
summary_People2 = pd.DataFrame({'Null Values': null_summary, 'Data Types': data
print(summary_People2)
```

```
                       Null Values Data Types
PERSON_ID                        0     object
PERSON_TYPE                      0     object
CRASH_RECORD_ID                  0     object
CITY                        479735     object
STATE                       461035     object
SEX                          29048     object
AGE                         515185    float64
DRIVERS_LICENSE_STATE       730369     object
DRIVERS_LICENSE_CLASS       898888     object
SAFETY_EQUIPMENT              4924     object
AIRBAG_DEPLOYED              33947     object
EJECTION                     21835     object
INJURY_CLASSIFICATION          718     object
DRIVER_ACTION               360007     object
DRIVER_VISION               360529     object
PHYSICAL_CONDITION          359047     object
```

In [32]:
```
# dropping the null rows in the selected columns
People3 = People2.dropna(subset = ['DRIVERS_LICENSE_CLASS','AGE','CITY','STATE
```

In [33]:
```
# checking the null values
null_summary = People3.isnull().sum()
data_types = People3.dtypes
# Combine null values and data types into a summary DataFrame
summary_People3 = pd.DataFrame({'Null Values': null_summary, 'Data Types': data
print(summary_People3)
```

```
                        Null Values Data Types
PERSON_ID                         0     object
PERSON_TYPE                       0     object
CRASH_RECORD_ID                   0     object
CITY                              0     object
STATE                             0     object
SEX                               0     object
AGE                               0    float64
DRIVERS_LICENSE_STATE             0     object
DRIVERS_LICENSE_CLASS             0     object
SAFETY_EQUIPMENT                  1     object
AIRBAG_DEPLOYED                   1     object
EJECTION                          1     object
INJURY_CLASSIFICATION             0     object
DRIVER_ACTION                     0     object
DRIVER_VISION                     0     object
PHYSICAL_CONDITION                1     object
```

In [34]:
```
# preview the columns in the cleaned crahes dataset
Crashes3.columns
```

Out[34]:
```
Index(['CRASH_RECORD_ID', 'CRASH_DATE', 'POSTED_SPEED_LIMIT',
       'TRAFFIC_CONTROL_DEVICE', 'DEVICE_CONDITION', 'WEATHER_CONDITION',
       'LIGHTING_CONDITION', 'ROAD_DEFECT', 'CRASH_TYPE', 'DAMAGE',
       'PRIM_CONTRIBUTORY_CAUSE', 'SEC_CONTRIBUTORY_CAUSE', 'STREET_NAME',
       'MOST_SEVERE_INJURY', 'INJURIES_TOTAL', 'CRASH_HOUR',
       'CRASH_DAY_OF_WEEK', 'CRASH_MONTH', 'LOCATION'],
      dtype='object')
```

In [35]:
```
# preview the columns in the cleaned people dataset
People3.columns
```

Out[35]:
```
Index(['PERSON_ID', 'PERSON_TYPE', 'CRASH_RECORD_ID', 'CITY', 'STATE', 'SEX',
       'AGE', 'DRIVERS_LICENSE_STATE', 'DRIVERS_LICENSE_CLASS',
       'SAFETY_EQUIPMENT', 'AIRBAG_DEPLOYED', 'EJECTION',
       'INJURY_CLASSIFICATION', 'DRIVER_ACTION', 'DRIVER_VISION',
       'PHYSICAL_CONDITION'],
      dtype='object')
```

In [36]:
```
# preview the columns in the cleaned vehicles dataset
Vehicles3.columns
```

Out[36]:
```
Index(['CRASH_RECORD_ID', 'UNIT_NO', 'UNIT_TYPE', 'VEHICLE_ID', 'MAKE',
       'MODEL', 'VEHICLE_YEAR', 'VEHICLE_DEFECT', 'VEHICLE_TYPE',
       'VEHICLE_USE', 'MANEUVER', 'OCCUPANT_CNT'],
      dtype='object')
```

MERGE CLEAN DATASETS

In [37]:
```
# mergedcrashes = pd.merge(Crashes3,People3 on='CRASH_RECORD_ID')
mergecrashes = People3.merge(Crashes3,on='CRASH_RECORD_ID', how='left')
```

In [38]:
```
# merge vehicles dataset to the already merged  people and crash dataset(merged
mergecrashes1 = mergecrashes.merge(Vehicles3,on='CRASH_RECORD_ID', how='left')
mergecrashes1.columns
```

Out[38]:
```
Index(['PERSON_ID', 'PERSON_TYPE', 'CRASH_RECORD_ID', 'CITY', 'STATE', 'SEX',
       'AGE', 'DRIVERS_LICENSE_STATE', 'DRIVERS_LICENSE_CLASS',
       'SAFETY_EQUIPMENT', 'AIRBAG_DEPLOYED', 'EJECTION',
       'INJURY_CLASSIFICATION', 'DRIVER_ACTION', 'DRIVER_VISION',
       'PHYSICAL_CONDITION', 'CRASH_DATE', 'POSTED_SPEED_LIMIT',
       'TRAFFIC_CONTROL_DEVICE', 'DEVICE_CONDITION', 'WEATHER_CONDITION',
       'LIGHTING_CONDITION', 'ROAD_DEFECT', 'CRASH_TYPE', 'DAMAGE',
       'PRIM_CONTRIBUTORY_CAUSE', 'SEC_CONTRIBUTORY_CAUSE', 'STREET_NAME',
       'MOST_SEVERE_INJURY', 'INJURIES_TOTAL', 'CRASH_HOUR',
       'CRASH_DAY_OF_WEEK', 'CRASH_MONTH', 'LOCATION', 'UNIT_NO', 'UNIT_TYP
E',
       'VEHICLE_ID', 'MAKE', 'MODEL', 'VEHICLE_YEAR', 'VEHICLE_DEFECT',
       'VEHICLE_TYPE', 'VEHICLE_USE', 'MANEUVER', 'OCCUPANT_CNT'],
      dtype='object')
```

MORE CLEANING ON THE COMBINED DATASET

In [39]:
```
# check for nullvalues in the merged dataset
mergecrashes1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1553246 entries, 0 to 1553245
Data columns (total 45 columns):
 #   Column                   Non-Null Count    Dtype  
---  ------                   --------------    -----  
 0   PERSON_ID                1553246 non-null  object 
 1   PERSON_TYPE              1553246 non-null  object 
 2   CRASH_RECORD_ID          1553246 non-null  object 
 3   CITY                     1553246 non-null  object 
 4   STATE                    1553246 non-null  object 
 5   SEX                      1553246 non-null  object 
 6   AGE                      1553246 non-null  float64
 7   DRIVERS_LICENSE_STATE    1553246 non-null  object 
 8   DRIVERS_LICENSE_CLASS    1553246 non-null  object 
 9   SAFETY_EQUIPMENT         1553244 non-null  object 
 10  AIRBAG_DEPLOYED          1553245 non-null  object 
 11  EJECTION                 1553245 non-null  object 
 12  INJURY_CLASSIFICATION    1553246 non-null  object 
 13  DRIVER_ACTION            1553246 non-null  object 
 14  DRIVER_VISION            1553246 non-null  object 
 15  PHYSICAL_CONDITION       1553242 non-null  object 
 16  CRASH_DATE               1540287 non-null  object 
 17  POSTED_SPEED_LIMIT       1540287 non-null  float64
 18  TRAFFIC_CONTROL_DEVICE   1540287 non-null  object 
 19  DEVICE_CONDITION         1540287 non-null  object 
 20  WEATHER_CONDITION        1540287 non-null  object 
 21  LIGHTING_CONDITION       1540287 non-null  object 
 22  ROAD_DEFECT              1540287 non-null  object 
 23  CRASH_TYPE               1540287 non-null  object 
 24  DAMAGE                   1540287 non-null  object 
 25  PRIM_CONTRIBUTORY_CAUSE  1540287 non-null  object 
 26  SEC_CONTRIBUTORY_CAUSE   1540287 non-null  object 
 27  STREET_NAME              1540287 non-null  object 
 28  MOST_SEVERE_INJURY       1540287 non-null  object 
 29  INJURIES_TOTAL           1540287 non-null  float64
 30  CRASH_HOUR               1540287 non-null  float64
 31  CRASH_DAY_OF_WEEK        1540287 non-null  float64
 32  CRASH_MONTH              1540287 non-null  float64
 33  LOCATION                 1540287 non-null  object 
 34  UNIT_NO                  1543581 non-null  float64
 35  UNIT_TYPE                1543571 non-null  object 
 36  VEHICLE_ID               1543581 non-null  float64
 37  MAKE                     1543578 non-null  object 
 38  MODEL                    1543396 non-null  object 
 39  VEHICLE_YEAR             1543581 non-null  float64
 40  VEHICLE_DEFECT           1543581 non-null  object 
 41  VEHICLE_TYPE             1543581 non-null  object 
 42  VEHICLE_USE              1543581 non-null  object 
 43  MANEUVER                 1543581 non-null  object 
 44  OCCUPANT_CNT             1543581 non-null  float64
dtypes: float64(10), object(35)
memory usage: 533.3+ MB
```

In [40]:
```
# Dropping the rows with null values
mergecrashes2 = mergecrashes1.dropna(subset = ['MODEL','MAKE','SAFETY_EQUIPMEN
```

In [41]:
```
# checking if the dataset is clear of null values
mergecrashes2.isna().sum()
```

Out[41]:
```
PERSON_ID                  0
PERSON_TYPE                0
CRASH_RECORD_ID            0
CITY                       0
STATE                      0
SEX                        0
AGE                        0
DRIVERS_LICENSE_STATE      0
DRIVERS_LICENSE_CLASS      0
SAFETY_EQUIPMENT           0
AIRBAG_DEPLOYED            0
EJECTION                   0
INJURY_CLASSIFICATION      0
DRIVER_ACTION              0
DRIVER_VISION              0
PHYSICAL_CONDITION         0
CRASH_DATE                 0
POSTED_SPEED_LIMIT         0
TRAFFIC_CONTROL_DEVICE     0
DEVICE_CONDITION           0
WEATHER_CONDITION          0
LIGHTING_CONDITION         0
ROAD_DEFECT                0
CRASH_TYPE                 0
DAMAGE                     0
PRIM_CONTRIBUTORY_CAUSE    0
SEC_CONTRIBUTORY_CAUSE     0
STREET_NAME                0
MOST_SEVERE_INJURY         0
INJURIES_TOTAL             0
CRASH_HOUR                 0
CRASH_DAY_OF_WEEK          0
CRASH_MONTH                0
LOCATION                   0
UNIT_NO                    0
UNIT_TYPE                  0
VEHICLE_ID                 0
MAKE                       0
MODEL                      0
VEHICLE_YEAR               0
VEHICLE_DEFECT             0
VEHICLE_TYPE               0
VEHICLE_USE                0
MANEUVER                   0
OCCUPANT_CNT               0
dtype: int64
```

In [42]:
```
# shape of the dataset
mergecrashes2.shape
```

Out[42]:
```
(1530593, 45)
```

In [43]:
```
# preview the dataset columns
mergecrashes2.columns
```

Out[43]:
```
Index(['PERSON_ID', 'PERSON_TYPE', 'CRASH_RECORD_ID', 'CITY', 'STATE', 'SEX',
       'AGE', 'DRIVERS_LICENSE_STATE', 'DRIVERS_LICENSE_CLASS',
       'SAFETY_EQUIPMENT', 'AIRBAG_DEPLOYED', 'EJECTION',
       'INJURY_CLASSIFICATION', 'DRIVER_ACTION', 'DRIVER_VISION',
       'PHYSICAL_CONDITION', 'CRASH_DATE', 'POSTED_SPEED_LIMIT',
       'TRAFFIC_CONTROL_DEVICE', 'DEVICE_CONDITION', 'WEATHER_CONDITION',
       'LIGHTING_CONDITION', 'ROAD_DEFECT', 'CRASH_TYPE', 'DAMAGE',
       'PRIM_CONTRIBUTORY_CAUSE', 'SEC_CONTRIBUTORY_CAUSE', 'STREET_NAME',
       'MOST_SEVERE_INJURY', 'INJURIES_TOTAL', 'CRASH_HOUR',
       'CRASH_DAY_OF_WEEK', 'CRASH_MONTH', 'LOCATION', 'UNIT_NO', 'UNIT_TYP
E',
       'VEHICLE_ID', 'MAKE', 'MODEL', 'VEHICLE_YEAR', 'VEHICLE_DEFECT',
       'VEHICLE_TYPE', 'VEHICLE_USE', 'MANEUVER', 'OCCUPANT_CNT'],
      dtype='object')
```

In [44]:
```
# removing duplicates rows with the same crash id
chicagocrush = mergecrashes2.drop_duplicates(subset=['CRASH_RECORD_ID'])
```

In [45]:
```
# preview of the final dataset to be used in modelling
chicagocrush.head()
```

Out[45]:

	PERSON_ID	PERSON_TYPE	CRASH_RECORD_ID	CI
0	O749947	DRIVER	81dc0de2ed92aa62baccab641fa377be7feb1cc47e6554...	CHICA
3	O848601	DRIVER	f25f09798b51603bde602ded37fea826fc8b7f962fc3b0...	CHICA
5	O877654	DRIVER	e9146986f4b0884d00ff3a54da5249263b4b36c15d01ce...	CHICA
8	O879685	DRIVER	cb7e25c6e63094307e5b999a8a5ec6f781a54781aeaef3...	MELRO PA
10	O880033	DRIVER	2bdb105a834f9917ee9be1937302d775525192c91e7398...	CARBONDA

5 rows × 45 columns

In [46]:
```
# checking for null values
chicagocrush.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 529191 entries, 0 to 1553242
Data columns (total 45 columns):
 #   Column                   Non-Null Count   Dtype
---  ------                   --------------   -----
 0   PERSON_ID                529191 non-null  object
 1   PERSON_TYPE              529191 non-null  object
 2   CRASH_RECORD_ID          529191 non-null  object
 3   CITY                     529191 non-null  object
 4   STATE                    529191 non-null  object
 5   SEX                      529191 non-null  object
 6   AGE                      529191 non-null  float64
 7   DRIVERS_LICENSE_STATE    529191 non-null  object
 8   DRIVERS_LICENSE_CLASS    529191 non-null  object
 9   SAFETY_EQUIPMENT         529191 non-null  object
 10  AIRBAG_DEPLOYED          529191 non-null  object
 11  EJECTION                 529191 non-null  object
 12  INJURY_CLASSIFICATION    529191 non-null  object
 13  DRIVER_ACTION            529191 non-null  object
```

In [47]:
```
# dropping irrelevant columns in the merged dataset
columns_drop = ['PERSON_ID','PERSON_TYPE','DRIVERS_LICENSE_STATE','DRIVERS_LICE
chicagocrush1 = chicagocrush.drop(columns=columns_drop)
chicagocrush1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 529191 entries, 0 to 1553242
Data columns (total 39 columns):
 #   Column                   Non-Null Count   Dtype
---  ------                   --------------   -----
 0   CRASH_RECORD_ID          529191 non-null  object
 1   CITY                     529191 non-null  object
 2   STATE                    529191 non-null  object
 3   SEX                      529191 non-null  object
 4   AGE                      529191 non-null  float64
 5   AIRBAG_DEPLOYED          529191 non-null  object
 6   EJECTION                 529191 non-null  object
 7   INJURY_CLASSIFICATION    529191 non-null  object
 8   DRIVER_ACTION            529191 non-null  object
 9   DRIVER_VISION            529191 non-null  object
 10  PHYSICAL_CONDITION       529191 non-null  object
 11  CRASH_DATE               529191 non-null  object
 12  POSTED_SPEED_LIMIT       529191 non-null  float64
 13  TRAFFIC_CONTROL_DEVICE   529191 non-null  object
```

Data was too large we focused on data published on the last year 2023

In [48]:
```
chicagocrush1['CRASH_DATE'] = pd.to_datetime(chicagocrush1['CRASH_DATE'])

# Filter the data for the year 2023
Chicagocrush2023 = chicagocrush1[chicagocrush1['CRASH_DATE'].dt.year == 2023]

# Display the first few rows of the filtered DataFrame
Chicagocrush2023.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 68767 entries, 324 to 1553242
Data columns (total 39 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   CRASH_RECORD_ID          68767 non-null  object
 1   CITY                     68767 non-null  object
 2   STATE                    68767 non-null  object
 3   SEX                      68767 non-null  object
 4   AGE                      68767 non-null  float64
 5   AIRBAG_DEPLOYED          68767 non-null  object
 6   EJECTION                 68767 non-null  object
 7   INJURY_CLASSIFICATION    68767 non-null  object
 8   DRIVER_ACTION            68767 non-null  object
 9   DRIVER_VISION            68767 non-null  object
 10  PHYSICAL_CONDITION       68767 non-null  object
 11  CRASH_DATE               68767 non-null  datetime64[ns]
 12  POSTED_SPEED_LIMIT       68767 non-null  float64
 13  TRAFFIC_CONTROL_DEVICE   68767 non-null  object
 14  DEVICE_CONDITION         68767 non-null  object
 15  WEATHER_CONDITION        68767 non-null  object
 16  LIGHTING_CONDITION       68767 non-null  object
 17  ROAD_DEFECT              68767 non-null  object
 18  CRASH_TYPE               68767 non-null  object
 19  DAMAGE                   68767 non-null  object
 20  PRIM_CONTRIBUTORY_CAUSE  68767 non-null  object
 21  SEC_CONTRIBUTORY_CAUSE   68767 non-null  object
 22  STREET_NAME              68767 non-null  object
 23  MOST_SEVERE_INJURY       68767 non-null  object
 24  INJURIES_TOTAL           68767 non-null  float64
 25  CRASH_HOUR               68767 non-null  float64
 26  CRASH_DAY_OF_WEEK        68767 non-null  float64
 27  CRASH_MONTH              68767 non-null  float64
 28  LOCATION                 68767 non-null  object
 29  UNIT_NO                  68767 non-null  float64
 30  UNIT_TYPE                68767 non-null  object
 31  VEHICLE_ID               68767 non-null  float64
 32  MAKE                     68767 non-null  object
 33  MODEL                    68767 non-null  object
 34  VEHICLE_YEAR             68767 non-null  float64
 35  VEHICLE_DEFECT           68767 non-null  object
 36  VEHICLE_TYPE             68767 non-null  object
 37  VEHICLE_USE              68767 non-null  object
 38  MANEUVER                 68767 non-null  object
dtypes: datetime64[ns](1), float64(9), object(29)
memory usage: 21.0+ MB
```

In [49]:
```
# Encode categorical variables
label_encoder = LabelEncoder()

Chicagocrush2023.loc[:,'INJURY_CLASSIFICATION'] = label_encoder.fit_transform(C
Chicagocrush2023.loc[:,'DRIVER_VISION'] = label_encoder.fit_transform(Chicagocr
Chicagocrush2023.loc[:,'DRIVER_ACTION'] = label_encoder.fit_transform(Chicagocr
Chicagocrush2023.loc[:,'PHYSICAL_CONDITION'] = label_encoder.fit_transform(Chic
Chicagocrush2023.loc[:,'LIGHTING_CONDITION'] = label_encoder.fit_transform(Chic
Chicagocrush2023.loc[:,'AIRBAG_DEPLOYED'] = label_encoder.fit_transform(Chicago
Chicagocrush2023.loc[:,'MAKE'] = label_encoder.fit_transform(Chicagocrush2023[
Chicagocrush2023.loc[:,'CITY'] = label_encoder.fit_transform(Chicagocrush2023[
Chicagocrush2023.loc[:,'SEX'] = label_encoder.fit_transform(Chicagocrush2023['S
Chicagocrush2023.loc[:,'TRAFFIC_CONTROL_DEVICE'] = label_encoder.fit_transform(
Chicagocrush2023.loc[:,'ROAD_DEFECT'] = label_encoder.fit_transform(Chicagocrus
Chicagocrush2023.loc[:,'VEHICLE_USE'] = label_encoder.fit_transform(Chicagocrus
```

In [50]:
```
Chicagocrush2023.head()
```

Out[50]:

	CRASH_RECORD_ID	CITY	STATE	SEX	AGE	AIRBAG_DE
324	a073a023f6713d0957a093f7e550175fd8699ec037c9d5...	222	CT	1	32.0	
1389	c026bf2731fafc1ce64b302a52f58c9110627fca47541e...	384	IL	1	24.0	
27431	b51f3812261afa62974edad0c600168daac53c678adea3...	384	IL	0	36.0	
31253	41c564000224de0aa0bd4a0af0a8985a4538a2f89512e0...	1140	IL	1	24.0	
40908	0fc7fd50aac51f2a839c01ce53180d43f87803c598dd6b...	384	IL	0	40.0	

5 rows × 39 columns

In [51]:
```
# columns in the cleaned merged dataset
threshold = 0  # Set the threshold

# Create a new binary column indicating whether there are injuries (1) or not (
Chicagocrush2023.loc[:, 'INJURIES'] = (Chicagocrush2023['INJURIES_TOTAL'] > thr


# Display the first few rows of the DataFrame with the new binary column
print(Chicagocrush2023[['INJURIES_TOTAL', 'INJURIES']].head())
```

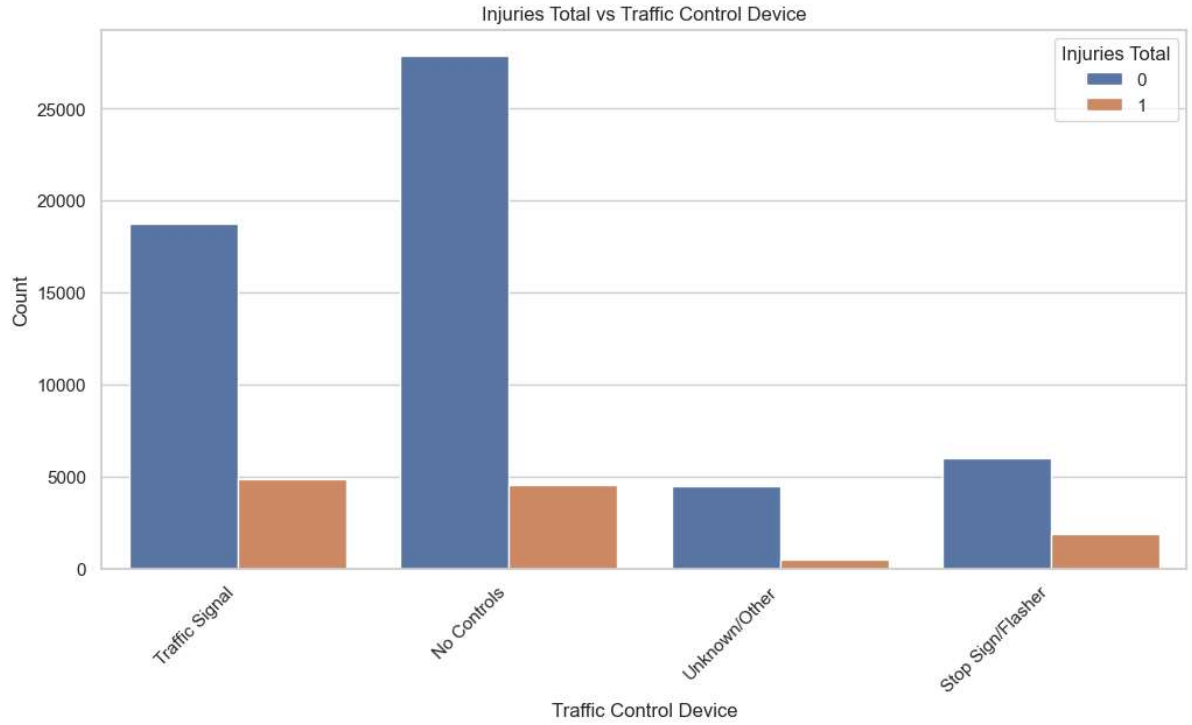
```
       INJURIES_TOTAL  INJURIES
324               0.0         0
1389              0.0         0
27431             0.0         0
31253             0.0         0
40908             0.0         0

C:\Users\Home\AppData\Local\Temp\ipykernel_1180\2471359133.py:5: SettingWithC
opyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/s
table/user_guide/indexing.html#returning-a-view-versus-a-copy (https://panda
s.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-ver
sus-a-copy)
  Chicagocrush2023.loc[:, 'INJURIES'] = (Chicagocrush2023['INJURIES_TOTAL'] >
threshold).astype(int)
```

Data Exploration

In [52]:
```
#   group different types of traffic control devices into a smaller set of cate
traffic_control_map = {
    'NO CONTROLS': 'No Controls',
    'TRAFFIC SIGNAL': 'Traffic Signal',
    'STOP SIGN/FLASHER': 'Stop Sign/Flasher',
    'UNKNOWN': 'Unknown/Other',
    'OTHER': 'Unknown/Other',
    'LANE USE MARKING': 'Traffic Signal',
    'YIELD': 'Unknown/Other',
    'OTHER REG. SIGN': 'Traffic Signal',
    'OTHER WARNING SIGN': 'Traffic Signal',
    'RAILROAD CROSSING GATE': 'Traffic Signal',
    'PEDESTRIAN CROSSING SIGN': 'Traffic Signal',
    'DELINEATORS': 'Traffic Signal',
    'FLASHING CONTROL SIGNAL': 'Traffic Signal',
    'POLICE/FLAGMAN': 'Traffic Signal',
    'SCHOOL ZONE': 'Traffic Signal',
    'OTHER RAILROAD CROSSING': 'Traffic Signal',
    'RR CROSSING SIGN': 'Traffic Signal',
    'NO PASSING': 'Traffic Signal',
    'BICYCLE CROSSING SIGN': 'Traffic Signal'
}

# Remap values in the 'TRAFFIC_CONTROL_DEVICE' column
Chicagocrush2023.loc[:,'TRAFFIC_CONTROL_DEVICE'] = chicagocrush1['TRAFFIC_CONTR

# Check the updated value counts
print(Chicagocrush2023['TRAFFIC_CONTROL_DEVICE'].value_counts())
```

```
TRAFFIC_CONTROL_DEVICE
No Controls          32454
Traffic Signal       23561
Stop Sign/Flasher     7832
Unknown/Other         4920
Name: count, dtype: int64
```

In [55]:
```
# Create a countplot to visualize the distribution of injuries for each traffic

sns.set(style="whitegrid")


plt.figure(figsize=(12, 6))
sns.countplot(x='TRAFFIC_CONTROL_DEVICE', hue='INJURIES', data=Chicagocrush2023
plt.title('Injuries Total vs Traffic Control Device')
plt.xlabel('Traffic Control Device')
plt.ylabel('Count')
plt.xticks(rotation=45, ha='right')  # Rotate x-axis labels for better visibilit
plt.legend(title='Injuries Total', loc='upper right')
plt.show()
```



In [56]:
```
Chicagocrush2023['INJURIES_TOTAL'].value_counts()
```

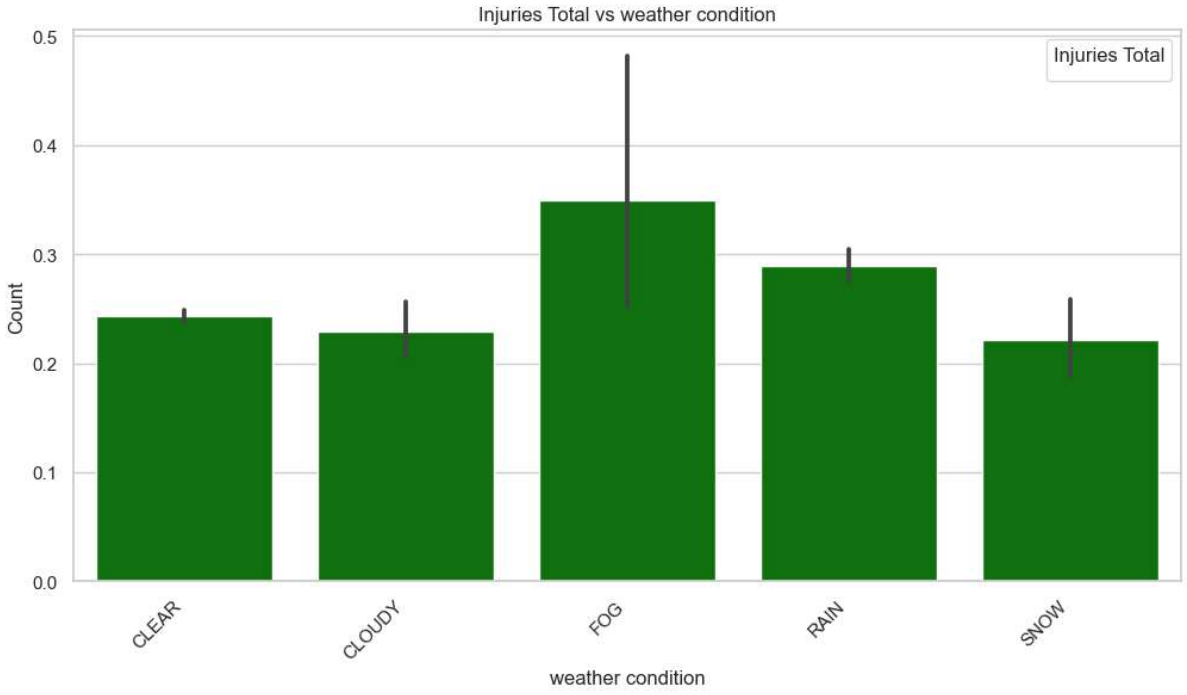
Out[56]:
```
INJURIES_TOTAL
0.0     57048
1.0      8303
2.0      2291
3.0       693
4.0       269
5.0        94
6.0        35
7.0        16
8.0         6
10.0        5
9.0         3
15.0        1
14.0        1
12.0        1
11.0        1
Name: count, dtype: int64
```

In [57]:
```
# Defining a dictionary map for remapping weather conditions to plot against ir
weather_condition_map = {
    'CLEAR': 'CLEAR',
    'RAIN': 'RAIN',
    'SNOW': 'SNOW',
    'CLOUDY/OVERCAST': 'CLOUDY',
    'UNKNOWN': 'CLEAR',
    'OTHER': 'CLEAR',
    'FREEZING RAIN/DRIZZLE': 'RAIN',
    'FOG/SMOKE/HAZE': 'FOG',
    'SLEET/HAIL': 'FOG',
    'BLOWING SNOW': 'FOG',
    'SEVERE CROSS WIND GATE': 'FOG',
    'BLOWING SAND, SOIL, DIRT': 'FOG'
}

# Remap values in the 'WEATHER_CONDITION' column using the defined map
Chicagocrush2023.loc[:, 'WEATHER_CONDITION'] = Chicagocrush2023['WEATHER_CONDI

# Check the updated value counts
weather_condition_counts = Chicagocrush2023['WEATHER_CONDITION'].value_counts()
print(weather_condition_counts)
```

```
WEATHER_CONDITION
CLEAR     57350
RAIN       7444
CLOUDY     2301
SNOW       1406
FOG         266
Name: count, dtype: int64
```

In [58]:
```
plt.figure(figsize=(12, 6))
sns.barplot(x='WEATHER_CONDITION', y='INJURIES_TOTAL', data=Chicagocrush2023, 
plt.title('Injuries Total vs weather condition')
plt.xlabel('weather condition')
plt.ylabel('Count')
plt.xticks(rotation=45, ha='right')  # Rotate x-axis labels for better visibil
plt.legend(title='Injuries Total', loc='upper right')
plt.show()
```

```
No artists with labels found to put in legend.  Note that artists whose label
start with an underscore are ignored when legend() is called with no argumen
t.
```



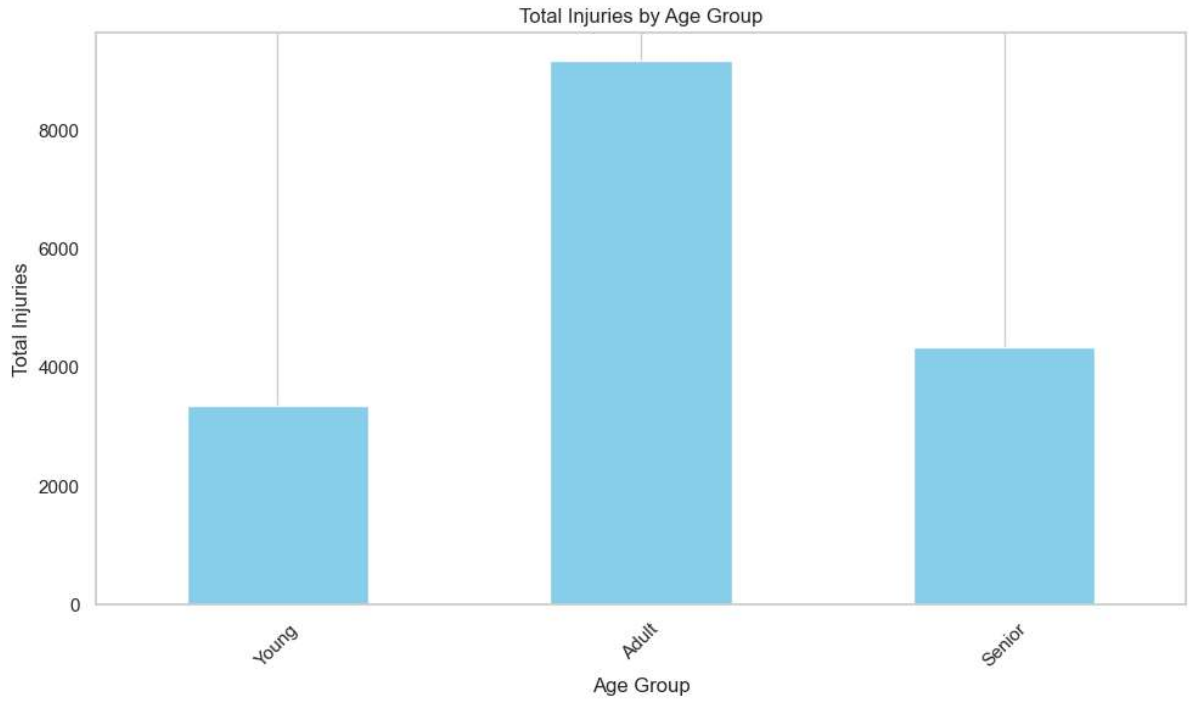
FEATURE ENGINEERING

In [59]:
```
# Binning age
bins = [0, 25, 50, 100]
labels = ['Young', 'Adult', 'Senior']
Chicagocrush2023.loc[:,'AGE_GROUP'] = pd.cut(Chicagocrush2023['AGE'], bins=bins
```

```
C:\Users\Home\AppData\Local\Temp\ipykernel_1180\3919465822.py:5: SettingWithC
opyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/s
table/user_guide/indexing.html#returning-a-view-versus-a-copy (https://panda
s.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-ver
sus-a-copy)
  Chicagocrush2023.loc[:,'AGE_GROUP'] = pd.cut(Chicagocrush2023['AGE'], bins=
bins, labels=labels)
```

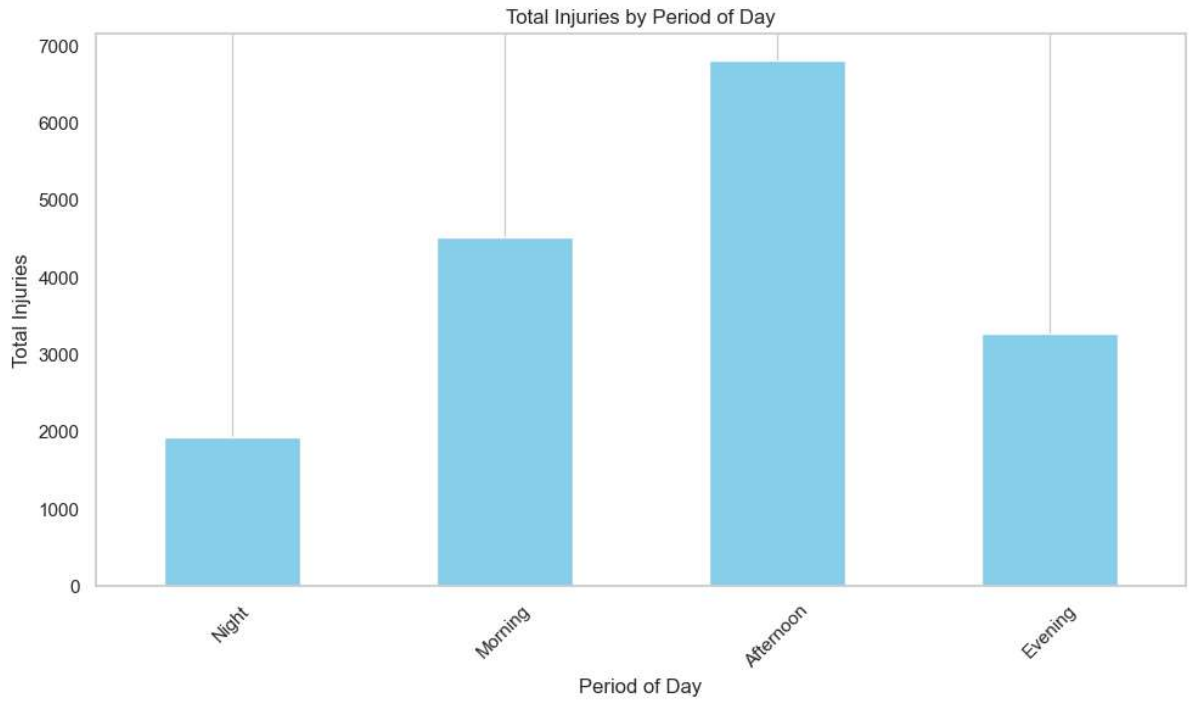
In [60]:

```
# Assuming you have a DataFrame named 'data' with 'AGE_GROUP' and 'TOTAL_INJURI

# Group the data by 'AGE_GROUP' and calculate the total injuries for each group
agegroup_total_injuries = Chicagocrush2023.groupby('AGE_GROUP')['INJURIES_TOTAL

# Plot the bar chart
plt.figure(figsize=(10, 6))
agegroup_total_injuries.plot(kind='bar', color='skyblue')
plt.title('Total Injuries by Age Group')
plt.xlabel('Age Group')
plt.ylabel('Total Injuries')
plt.xticks(rotation=45)  # Rotate x-axis labels for better visibility
plt.grid(axis='y')  # Add gridlines along y-axis
plt.tight_layout()  # Adjust layout to prevent overlapping labels
plt.show()
```



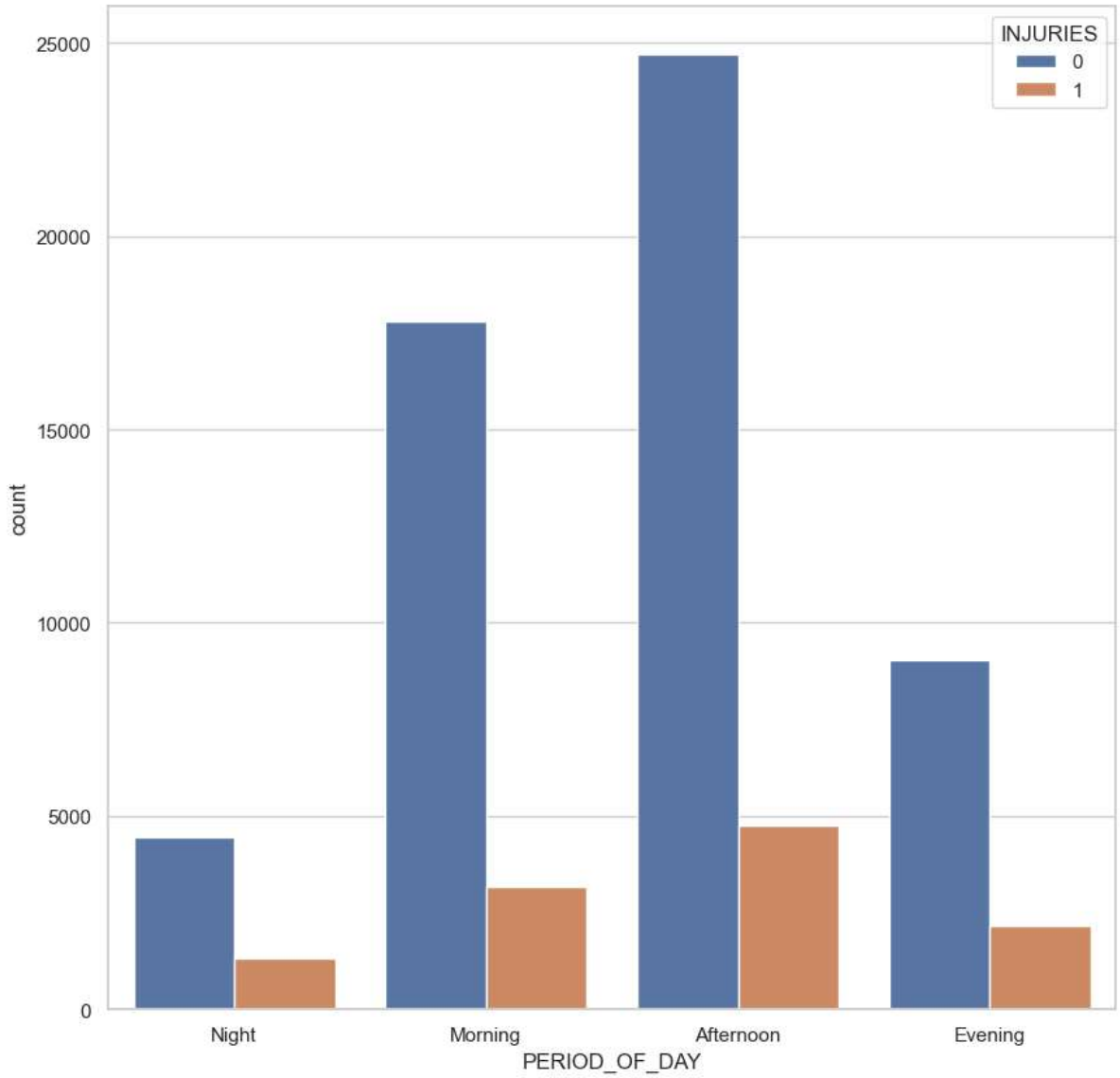
In [61]:
```
# Time period encoding
Chicagocrush2023.loc[:,'PERIOD_OF_DAY'] = pd.cut(Chicagocrush2023['CRASH_HOUR'
# Group the data by 'PERIOD_OF_DAY' and calculate the total injuries for each 
period_of_day_total_injuries = Chicagocrush2023.groupby('PERIOD_OF_DAY')['INJU

# Plot the bar chart
plt.figure(figsize=(10, 6))
period_of_day_total_injuries.plot(kind='bar', color='skyblue')
plt.title('Total Injuries by Period of Day')
plt.xlabel('Period of Day')
plt.ylabel('Total Injuries')
plt.xticks(rotation=45)  # Rotate x-axis labels for better visibility
plt.grid(axis='y')  # Add gridlines along y-axis
plt.tight_layout()  # Adjust layout to prevent overlapping labels
plt.show()
```

```
C:\Users\Home\AppData\Local\Temp\ipykernel_1180\3928030688.py:2: SettingWithC
opyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/s
table/user_guide/indexing.html#returning-a-view-versus-a-copy (https://panda
s.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-ver
sus-a-copy)
  Chicagocrush2023.loc[:,'PERIOD_OF_DAY'] = pd.cut(Chicagocrush2023['CRASH_HO
UR'], bins=[0, 6, 12, 18, 24], labels=['Night', 'Morning', 'Afternoon', 'Even
ing'])
```



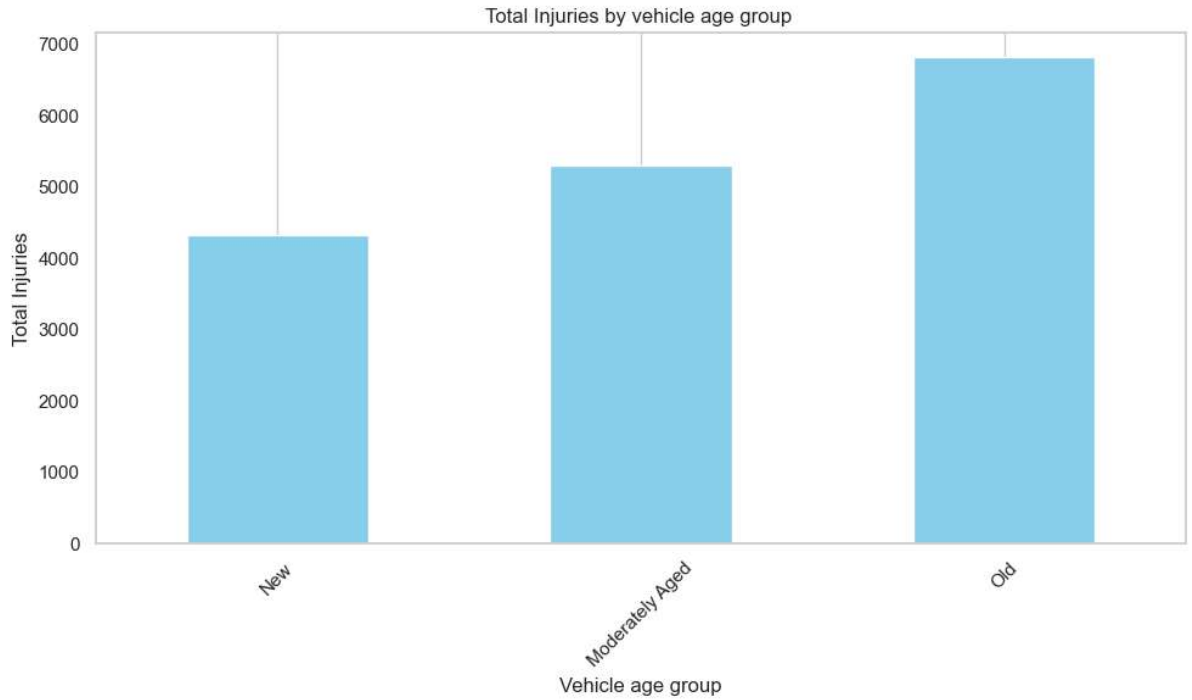
In [62]:
```
# representing period of day in relation to injuries

plt.figure(figsize=(10,10))
sns.countplot(x="PERIOD_OF_DAY", hue="INJURIES", data=Chicagocrush2023)
```

Out[62]: <Axes: xlabel='PERIOD_OF_DAY', ylabel='count'>



In [63]:
```
# Convert 'CRASH_DATE' to datetime format
Chicagocrush2023.loc[:,'CRASH_DATE'] = pd.to_datetime(Chicagocrush2023['CRASH_D

# Extract just the year
Chicagocrush2023.loc[:,'YEAR'] = Chicagocrush2023['CRASH_DATE'].dt.year
# Calculate the age of the vehicle at the time of the accident
Chicagocrush2023.loc[:,'VEHICLE_AGE'] = Chicagocrush2023['YEAR'] - Chicagocrush

# Binning vehicle age
bins = [0, 5, 10, float('inf')]  # Example bins: 0-5 years, 5-10 years, >10 yea
labels = ['New', 'Moderately Aged', 'Old']
Chicagocrush2023.loc[:,'VEHICLE_AGE_GROUP'] = pd.cut(Chicagocrush2023['VEHICLE_
#  Group the data by 'VEHICLE_AGE_GROUP' and calculate the total injuries for e
vehicle_age_group_total_injuries = Chicagocrush2023.groupby('VEHICLE_AGE_GROUP

# Plot the bar chart
plt.figure(figsize=(10, 6))
vehicle_age_group_total_injuries.plot(kind='bar', color='skyblue')
plt.title('Total Injuries by vehicle age group ')
plt.xlabel('Vehicle age group')
plt.ylabel('Total Injuries')
plt.xticks(rotation=45)  # Rotate x-axis labels for better visibility
plt.grid(axis='y')  # Add gridlines along y-axis
plt.tight_layout()  # Adjust layout to prevent overlapping labels
plt.show()
```

```
C:\Users\Home\AppData\Local\Temp\ipykernel_1180\1816627131.py:5: SettingWithC
opyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/s
table/user_guide/indexing.html#returning-a-view-versus-a-copy (https://panda
s.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-ver
sus-a-copy)
  Chicagocrush2023.loc[:,'YEAR'] = Chicagocrush2023['CRASH_DATE'].dt.year
C:\Users\Home\AppData\Local\Temp\ipykernel_1180\1816627131.py:7: SettingWithC
opyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/s
table/user_guide/indexing.html#returning-a-view-versus-a-copy (https://panda
s.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-ver
sus-a-copy)
  Chicagocrush2023.loc[:,'VEHICLE_AGE'] = Chicagocrush2023['YEAR'] - Chicagoc
rush2023['VEHICLE_YEAR']
C:\Users\Home\AppData\Local\Temp\ipykernel_1180\1816627131.py:12: SettingWith
CopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/s
table/user_guide/indexing.html#returning-a-view-versus-a-copy (https://panda
s.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-ver
sus-a-copy)
  Chicagocrush2023.loc[:,'VEHICLE_AGE_GROUP'] = pd.cut(Chicagocrush2023['VEHI
CLE_AGE'], bins=bins, labels=labels)
```



In [64]:
```
# preview the columns in the dataset
Chicagocrush2023.columns
```

Out[64]:
```
Index(['CRASH_RECORD_ID', 'CITY', 'STATE', 'SEX', 'AGE', 'AIRBAG_DEPLOYED',
       'EJECTION', 'INJURY_CLASSIFICATION', 'DRIVER_ACTION', 'DRIVER_VISION',
       'PHYSICAL_CONDITION', 'CRASH_DATE', 'POSTED_SPEED_LIMIT',
       'TRAFFIC_CONTROL_DEVICE', 'DEVICE_CONDITION', 'WEATHER_CONDITION',
       'LIGHTING_CONDITION', 'ROAD_DEFECT', 'CRASH_TYPE', 'DAMAGE',
       'PRIM_CONTRIBUTORY_CAUSE', 'SEC_CONTRIBUTORY_CAUSE', 'STREET_NAME',
       'MOST_SEVERE_INJURY', 'INJURIES_TOTAL', 'CRASH_HOUR',
       'CRASH_DAY_OF_WEEK', 'CRASH_MONTH', 'LOCATION', 'UNIT_NO', 'UNIT_TYP
E',
       'VEHICLE_ID', 'MAKE', 'MODEL', 'VEHICLE_YEAR', 'VEHICLE_DEFECT',
       'VEHICLE_TYPE', 'VEHICLE_USE', 'MANEUVER', 'INJURIES', 'AGE_GROUP',
       'PERIOD_OF_DAY', 'YEAR', 'VEHICLE_AGE', 'VEHICLE_AGE_GROUP'],
      dtype='object')
```

In [65]:
```
# dropping columns
columns_drop = ['DEVICE_CONDITION','VEHICLE_DEFECT','MODEL']
Chicagocrush2023 = Chicagocrush2023.drop(columns=columns_drop)
Chicagocrush2023.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 68767 entries, 324 to 1553242
Data columns (total 42 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   CRASH_RECORD_ID          68767 non-null  object
 1   CITY                     68767 non-null  object
 2   STATE                    68767 non-null  object
 3   SEX                      68767 non-null  object
 4   AGE                      68767 non-null  float64
 5   AIRBAG_DEPLOYED          68767 non-null  object
 6   EJECTION                 68767 non-null  object
 7   INJURY_CLASSIFICATION    68767 non-null  object
 8   DRIVER_ACTION            68767 non-null  object
 9   DRIVER_VISION            68767 non-null  object
 10  PHYSICAL_CONDITION       68767 non-null  object
 11  CRASH_DATE               68767 non-null  datetime64[ns]
 12  POSTED_SPEED_LIMIT       68767 non-null  float64
 13  TRAFFIC_CONTROL_DEVICE   68767 non-null  object
 14  WEATHER_CONDITION        68767 non-null  object
 15  LIGHTING_CONDITION       68767 non-null  object
 16  ROAD_DEFECT              68767 non-null  object
 17  CRASH_TYPE               68767 non-null  object
 18  DAMAGE                   68767 non-null  object
 19  PRIM_CONTRIBUTORY_CAUSE  68767 non-null  object
 20  SEC_CONTRIBUTORY_CAUSE   68767 non-null  object
 21  STREET_NAME              68767 non-null  object
 22  MOST_SEVERE_INJURY       68767 non-null  object
 23  INJURIES_TOTAL           68767 non-null  float64
 24  CRASH_HOUR               68767 non-null  float64
 25  CRASH_DAY_OF_WEEK        68767 non-null  float64
 26  CRASH_MONTH              68767 non-null  float64
 27  LOCATION                 68767 non-null  object
 28  UNIT_NO                  68767 non-null  float64
 29  UNIT_TYPE                68767 non-null  object
 30  VEHICLE_ID               68767 non-null  float64
 31  MAKE                     68767 non-null  object
 32  VEHICLE_YEAR             68767 non-null  float64
 33  VEHICLE_TYPE             68767 non-null  object
 34  VEHICLE_USE              68767 non-null  object
 35  MANEUVER                 68767 non-null  object
 36  INJURIES                 68767 non-null  int32
 37  AGE_GROUP                68223 non-null  category
 38  PERIOD_OF_DAY            67439 non-null  category
 39  YEAR                     68767 non-null  int32
 40  VEHICLE_AGE              68767 non-null  float64
 41  VEHICLE_AGE_GROUP        66247 non-null  category
dtypes: category(3), datetime64[ns](1), float64(10), int32(2), object(26)
memory usage: 20.7+ MB
```

In [66]:
```
# previewing the data columns and type
Chicagocrush2023.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 68767 entries, 324 to 1553242
Data columns (total 42 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   CRASH_RECORD_ID          68767 non-null  object
 1   CITY                     68767 non-null  object
 2   STATE                    68767 non-null  object
 3   SEX                      68767 non-null  object
 4   AGE                      68767 non-null  float64
 5   AIRBAG_DEPLOYED          68767 non-null  object
 6   EJECTION                 68767 non-null  object
 7   INJURY_CLASSIFICATION    68767 non-null  object
 8   DRIVER_ACTION            68767 non-null  object
 9   DRIVER_VISION            68767 non-null  object
 10  PHYSICAL_CONDITION       68767 non-null  object
 11  CRASH_DATE               68767 non-null  datetime64[ns]
 12  POSTED_SPEED_LIMIT       68767 non-null  float64
 13  TRAFFIC_CONTROL_DEVICE   68767 non-null  object
 14  WEATHER_CONDITION        68767 non-null  object
 15  LIGHTING_CONDITION       68767 non-null  object
 16  ROAD_DEFECT              68767 non-null  object
 17  CRASH_TYPE               68767 non-null  object
 18  DAMAGE                   68767 non-null  object
 19  PRIM_CONTRIBUTORY_CAUSE  68767 non-null  object
 20  SEC_CONTRIBUTORY_CAUSE   68767 non-null  object
 21  STREET_NAME              68767 non-null  object
 22  MOST_SEVERE_INJURY       68767 non-null  object
 23  INJURIES_TOTAL           68767 non-null  float64
 24  CRASH_HOUR               68767 non-null  float64
 25  CRASH_DAY_OF_WEEK        68767 non-null  float64
 26  CRASH_MONTH              68767 non-null  float64
 27  LOCATION                 68767 non-null  object
 28  UNIT_NO                  68767 non-null  float64
 29  UNIT_TYPE                68767 non-null  object
 30  VEHICLE_ID               68767 non-null  float64
 31  MAKE                     68767 non-null  object
 32  VEHICLE_YEAR             68767 non-null  float64
 33  VEHICLE_TYPE             68767 non-null  object
 34  VEHICLE_USE              68767 non-null  object
 35  MANEUVER                 68767 non-null  object
 36  INJURIES                 68767 non-null  int32
 37  AGE_GROUP                68223 non-null  category
 38  PERIOD_OF_DAY            67439 non-null  category
 39  YEAR                     68767 non-null  int32
 40  VEHICLE_AGE              68767 non-null  float64
 41  VEHICLE_AGE_GROUP        66247 non-null  category
dtypes: category(3), datetime64[ns](1), float64(10), int32(2), object(26)
memory usage: 20.7+ MB
```

In [67]:
```
# dropping irrelevant columns
columns_drop = ['STATE','EJECTION','INJURY_CLASSIFICATION','CRASH_TYPE','DAMAGE
                'SEC_CONTRIBUTORY_CAUSE','STREET_NAME', 'MOST_SEVERE_INJURY','(
                'UNIT_NO','UNIT_TYPE','VEHICLE_ID','MAKE','MANEUVER','INJURIES_
                'CRASH_DATE','CRASH_RECORD_ID','VEHICLE_TYPE','VEHICLE_YEAR','Y

Chicagocrush2023 = Chicagocrush2023.drop(columns=columns_drop)
```

In [68]:
```
# data preview
Chicagocrush2023.head()
```

Out[68]:

	CITY	SEX	AIRBAG_DEPLOYED	DRIVER_ACTION	DRIVER_VISION	PHYSICAL_CONDITION
324	222	1	5	18	12	8
1389	384	1	5	13	12	8
27431	384	0	3	7	12	8
31253	1140	1	5	12	7	8
40908	384	0	5	18	7	8

Train test split

In [69]:
```
# Define features (X) and target variable (y)
X = Chicagocrush2023.drop(columns=['INJURIES'])  # Features
y = Chicagocrush2023['INJURIES']  # Target variable

# Split the data into training and temporary set (70% training, 30% temporary)
X_train, X_temp, y_train, y_temp = train_test_split(X, y, test_size=0.3, randor

# Split the temporary set into validation and testing sets (50% validation, 50%
X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp, test_size=0.5,

# Display the shapes of the resulting sets
print("Training set:", X_train.shape, y_train.shape)
print("Validation set:", X_val.shape, y_val.shape)
print("Testing set:", X_test.shape, y_test.shape)
```

```
Training set: (48136, 17) (48136,)
Validation set: (10315, 17) (10315,)
Testing set: (10316, 17) (10316,)
```

In [70]:
```
num_columns = Chicagocrush2023.drop(columns='INJURIES').select_dtypes('number')
num_columns
```

Out[70]:
```
['POSTED_SPEED_LIMIT', 'VEHICLE_AGE']
```

In [71]:
```
cat_cols = Chicagocrush2023.drop(columns='INJURIES').select_dtypes('object').co
cat_cols
```

Out[71]:
```
['CITY',
 'SEX',
 'AIRBAG_DEPLOYED',
 'DRIVER_ACTION',
 'DRIVER_VISION',
 'PHYSICAL_CONDITION',
 'TRAFFIC_CONTROL_DEVICE',
 'WEATHER_CONDITION',
 'LIGHTING_CONDITION',
 'ROAD_DEFECT',
 'LOCATION',
 'VEHICLE_USE']
```

In [72]:
```
numerical_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='mean')),
    ('scaler', StandardScaler())
])


categorical_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='most_frequent')),
    ('onehot', OneHotEncoder(handle_unknown='ignore'))
])

# combine both pipelines into one using columntransformer
preprocessor = ColumnTransformer(
    transformers=[
        ('num', numerical_transformer, num_columns),
        ('cat', categorical_transformer, cat_cols)
    ])



# preprocessing X_train and X_test
X_train_transformed = preprocessor.fit_transform(X_train)
X_test_transformed = preprocessor.transform(X_test)
# Fit and transform the data
# X_transformed = preprocessor.fit_transform(Chicagocrush2023)
```

In [73]:
```
X_test_transformed
```

Out[73]:
```
<10316x36832 sparse matrix of type '<class 'numpy.float64'>'
        with 138145 stored elements in Compressed Sparse Row format>
```

In [88]:
```
categorical_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='most_frequent')),
    ('onehot', OneHotEncoder(handle_unknown='error', drop='first'))
])
```

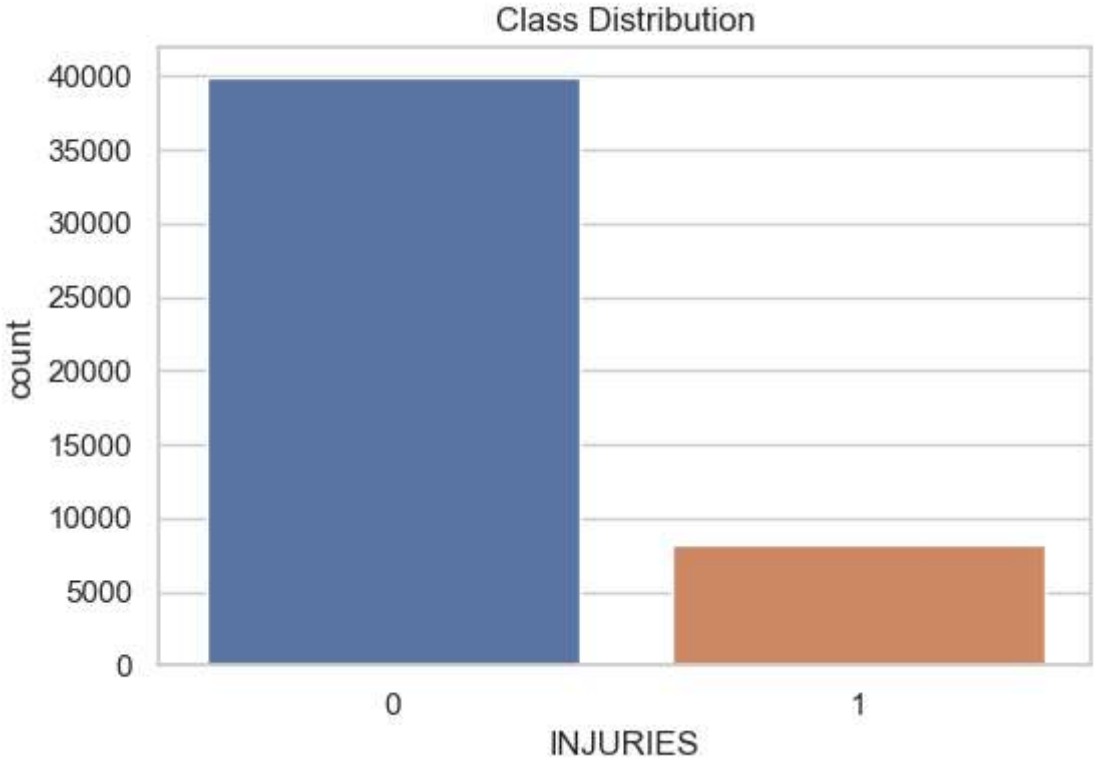
In [90]:
```
for col in X_train.columns:
    unique_values = X_train[col].unique()
    print(f"Column: {col}, Unique Values: {unique_values}")
```

```
Column: CITY, Unique Values: [384 1456 1956 ... 1864 1083 1307]
Column: SEX, Unique Values: [0 1 2]
Column: AIRBAG_DEPLOYED, Unique Values: [6 5 1 2 4 3 0]
Column: DRIVER_ACTION, Unique Values: [5 12 4 18 10 9 13 19 1 6 8 17 7 14 0 1
5 3 2 16 11]
Column: DRIVER_VISION, Unique Values: [7 12 6 3 1 13 8 9 2 0 11 4 5]
Column: PHYSICAL_CONDITION, Unique Values: [8 11 9 1 4 0 10 5 2 7 3 6]
Column: POSTED_SPEED_LIMIT, Unique Values: [30. 25. 35. 15. 45. 10. 20. 40.
5. 50. 55.  0.  9. 36. 39.  3. 34. 26.
 24. 60.  1.  2.]
Column: TRAFFIC_CONTROL_DEVICE, Unique Values: ['Stop Sign/Flasher' 'No Contr
ols' 'Traffic Signal' 'Unknown/Other']
Column: WEATHER_CONDITION, Unique Values: ['CLEAR' 'RAIN' 'CLOUDY' 'SNOW' 'FO
G']
Column: LIGHTING_CONDITION, Unique Values: [3 1 5 0 4 2]
Column: ROAD_DEFECT, Unique Values: [1 5 2 6 4 3 0]
Column: LOCATION, Unique Values: ['POINT (-87.631784530895 41.849118565114)'
 'POINT (-87.752102443463 41.91674599163)'
 'POINT (-87.63185003155 41.857643037815)' ...
 'POINT (-87.621107789652 41.68909160202)'
 'POINT (-87.726460715353 41.903055612189)'
 'POINT (-87.624315022826 41.900097591532)']
Column: VEHICLE_USE, Unique Values: [13 6 20 18 16 12 3 11 15 4 5 1 14 9 7 8
17 19 0 2 10]
Column: AGE_GROUP, Unique Values: ['Adult', 'Senior', 'Young', NaN]
Categories (3, object): ['Young' < 'Adult' < 'Senior']
Column: PERIOD_OF_DAY, Unique Values: ['Morning', 'Afternoon', 'Night', NaN,
'Evening']
Categories (4, object): ['Night' < 'Morning' < 'Afternoon' < 'Evening']
Column: VEHICLE_AGE, Unique Values: [ 2.000e+00  3.000e+00  1.300e+01  1.800e
+01  2.300e+01  2.100e+01
  1.000e+01  4.000e+00  8.000e+00  9.000e+00  1.200e+01  0.000e+00
  1.400e+01  5.000e+00  1.000e+00  6.000e+00  1.900e+01  2.000e+01
  1.100e+01  2.400e+01  1.500e+01  1.600e+01  7.000e+00  1.700e+01
  1.230e+02  2.200e+01  2.900e+01  2.500e+01  2.800e+01  2.600e+01
  2.700e+01 -1.000e+00  3.300e+01  3.900e+01  3.500e+01 -7.976e+03
  3.600e+01 -7.700e+01  3.400e+01  3.000e+01  1.220e+02 -1.840e+02
 -4.920e+02 -2.000e+00  4.300e+01  4.800e+01 -4.300e+01  3.700e+01
  5.400e+01  3.100e+01  4.400e+01  3.200e+01  5.700e+01  3.800e+01
 -1.870e+02  4.700e+01  5.000e+01  5.100e+01 -2.500e+01 -9.900e+02
 -3.260e+02 -1.000e+01 -8.910e+02  5.300e+01  5.600e+01 -1.990e+02
 -6.000e+00  4.100e+01 -1.830e+02 -8.600e+01 -8.400e+01 -2.982e+03
  6.600e+01  4.600e+01 -2.988e+03 -2.400e+01 -2.994e+03 -9.000e+00]
Column: VEHICLE_AGE_GROUP, Unique Values: ['New', 'Old', 'Moderately Aged', N
aN]
Categories (3, object): ['New' < 'Moderately Aged' < 'Old']
```

In [104]:
```
# accessing categorical columns from pipeline then converting to dataframe
slice_pipe = preprocessor.named_transformers_['cat']
cat_features = slice_pipe.named_steps['onehot'].get_feature_names_out(cat_cols)
X_train_transformed = pd.DataFrame(X_train_transformed,columns=[*num_columns, *
X_train_transformed
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[104], line 4
      2 slice_pipe = preprocessor.named_transformers_['cat']
      3 cat_features = slice_pipe.named_steps['onehot'].get_feature_names_out
(cat_cols)
----> 4 X_train_transformed = pd.DataFrame(X_train_transformed,columns=[*num_
columns, *cat_features])
      5 X_train_transformed

File ~\anaconda3\Lib\site-packages\pandas\core\frame.py:798, in DataFrame.__i
nit__(self, data, index, columns, dtype, copy)
    790         mgr = arrays_to_mgr(
    791             arrays,
    792             columns,
   (...)
    795             typ=manager,
    796         )
    797     else:
--> 798         mgr = ndarray_to_mgr(
    799             data,
    800             index,
    801             columns,
    802             dtype=dtype,
    803             copy=copy,
    804             typ=manager,
    805         )
    806 else:
    807     mgr = dict_to_mgr(
    808         {},
    809         index,
   (...)
    812         typ=manager,
    813     )

File ~\anaconda3\Lib\site-packages\pandas\core\internals\construction.py:337,
in ndarray_to_mgr(values, index, columns, dtype, copy, typ)
    332 # _prep_ndarraylike ensures that values.ndim == 2 at this point
    333 index, columns = _get_axes(
    334     values.shape[0], values.shape[1], index=index, columns=columns
    335 )
--> 337 _check_values_indices_shape_match(values, index, columns)
    339 if typ == "array":
    340     if issubclass(values.dtype.type, str):

File ~\anaconda3\Lib\site-packages\pandas\core\internals\construction.py:408,
in _check_values_indices_shape_match(values, index, columns)
    406 passed = values.shape
    407 implied = (len(index), len(columns))
--> 408 raise ValueError(f"Shape of passed values is {passed}, indices imply
{implied}")

ValueError: Shape of passed values is (48136, 1), indices imply (48136, 3683
2)
```

In [75]:
```
# Checking for imbalance visually
plt.figure(figsize=(6, 4))
sns.countplot(x=y_train)
plt.title('Class Distribution')
plt.show()
```



In [76]:
```
# checking for imbalance
class_distribution = y_train.value_counts(normalize=True)
print(class_distribution)
```

```
INJURIES
0    0.829525
1    0.170475
Name: proportion, dtype: float64
```

In [ ]:
```
# observation there is high imbalance and to solve we will use resampling(SMOTE
# from imblearn.over_sampling import SMOTE
# smote = SMOTE(sampling_strategy='auto', random_state=42)
# X_train_resampled, y_train_resampled = smote.fit_resample(X_train, y_train)
```

In [115]:
```
from sklearn.metrics import classification_report

# Predictions on the test set
y_pred_balanced = balanced_pipeline.predict(X_test)

# Evaluate the model
classification_rep_balanced = classification_report(y_test, y_pred_balanced)
print('Classification Report (Balanced Model):\n', classification_rep_balanced)
```

```
Classification Report (Balanced Model):
               precision    recall  f1-score   support

           0       0.83      0.50      0.62     11399
           1       0.18      0.52      0.26      2355

    accuracy                           0.50     13754
   macro avg       0.50      0.51      0.44     13754
weighted avg       0.72      0.50      0.56     13754
```

LOGISTIC REGRESSION MODEL

In [97]:
```
# instantiate LogisticRegression
# instantiate LogisticRegression

log_reg = LogisticRegression()

label_encoder = LabelEncoder()
y_encoded = label_encoder.fit_transform(y)

label_encoder = LabelEncoder()
X_encoded = label_encoder.fit_transform(y)
```

In [102]:
```
print(Chicagocrush2023.columns)
```

```
Index(['CITY', 'SEX', 'AIRBAG_DEPLOYED', 'DRIVER_ACTION', 'DRIVER_VISION',
       'PHYSICAL_CONDITION', 'POSTED_SPEED_LIMIT', 'TRAFFIC_CONTROL_DEVICE',
       'WEATHER_CONDITION', 'LIGHTING_CONDITION', 'ROAD_DEFECT', 'LOCATION',
       'VEHICLE_USE', 'INJURIES', 'AGE_GROUP', 'PERIOD_OF_DAY', 'VEHICLE_AG
E',
       'VEHICLE_AGE_GROUP'],
      dtype='object')
```

In [105]:
```
# Selecting features (X) and target variable (y)
features = ['AGE_GROUP', 'VEHICLE_AGE']
target = 'INJURIES'

X = Chicagocrush2023[features]
y = Chicagocrush2023[target]

# Perform train-test split (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, rando

# Define numerical and categorical features
numerical_features = X.select_dtypes(include=['float64']).columns
categorical_features = X.select_dtypes(include=['object']).columns

# Create transformers for numerical and categorical features
numerical_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='median')),
    ('scaler', StandardScaler())
])

categorical_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='constant', fill_value='missing')),
    ('onehot', OneHotEncoder(handle_unknown='error'))
])


# Combine transformers using ColumnTransformer
preprocessor = ColumnTransformer(
    transformers=[
        ('num', numerical_transformer, numerical_features),
        ('cat', categorical_transformer, categorical_features)
    ])

# Create the final pipeline with preprocessing and logistic regression
pipeline = Pipeline([
    ('preprocessor', preprocessor),
    ('classifier', LogisticRegression())
])

# Fit the model
pipeline.fit(X_train, y_train)

# Predictions on the test set
y_pred = pipeline.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy:.2f}')



classification_rep = classification_report(y_test, y_pred)
print('Classification Report:\n', classification_rep)
```

```
Accuracy: 0.83
Classification Report:
              precision    recall  f1-score   support

           0       0.83      1.00      0.91     11399
           1       0.00      0.00      0.00      2355

    accuracy                           0.83     13754
   macro avg       0.41      0.50      0.45     13754
weighted avg       0.69      0.83      0.75     13754


C:\Users\Home\AppData\Roaming\Python\Python311\site-packages\sklearn\metrics
\_classification.py:1497: UndefinedMetricWarning: Precision is ill-defined an
d being set to 0.0 in labels with no predicted samples. Use `zero_division` p
arameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
C:\Users\Home\AppData\Roaming\Python\Python311\site-packages\sklearn\metrics
\_classification.py:1497: UndefinedMetricWarning: Precision is ill-defined an
d being set to 0.0 in labels with no predicted samples. Use `zero_division` p
arameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
C:\Users\Home\AppData\Roaming\Python\Python311\site-packages\sklearn\metrics
\_classification.py:1497: UndefinedMetricWarning: Precision is ill-defined an
d being set to 0.0 in labels with no predicted samples. Use `zero_division` p
arameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```

KNN MODEL

In [80]:

```
knn_clf = KNeighborsClassifier()
```

In [81]:

```
# fit the model
knn_clf.fit(X_train_transformed, y_train)

# predict
y_pred = knn_clf.predict(X_test_transformed)
```

In [82]:
```
# Compute accuracy
accuracy = accuracy_score(y_test, y_pred)

# Compute precision, recall, and F1-score
precision = precision_score(y_test, y_pred, average='weighted')
recall = recall_score(y_test, y_pred, average='weighted')
f1 = f1_score(y_test, y_pred, average='weighted')

# Compute confusion matrix
conf_matrix = confusion_matrix(y_test, y_pred)

# Print classification report
class_report = classification_report(y_test, y_pred)

# Print the evaluation metrics
print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1 Score:", f1)
print("Confusion Matrix:\n", conf_matrix)
print("Classification Report:\n", class_report)
```

```
Accuracy: 0.8267739433889104
Precision: 0.785723295045503
Recall: 0.8267739433889104
F1 Score: 0.7932818006167912
Confusion Matrix:
 [[8228  362]
 [1425  301]]
Classification Report:
               precision    recall  f1-score   support

           0       0.85      0.96      0.90      8590
           1       0.45      0.17      0.25      1726

    accuracy                           0.83     10316
   macro avg       0.65      0.57      0.58     10316
weighted avg       0.79      0.83      0.79     10316
```

Decision trees

In [94]:
```
# instantiate DecisionTreeClassifier
tree_clf = DecisionTreeClassifier()
```

In [95]:
```
# fit the model
tree_clf.fit(X_train_transformed, y_train)

# predict
y_pred = tree_clf.predict(X_test_transformed)
```

In [96]:
```
# Compute accuracy
accuracy = accuracy_score(y_test, y_pred)

# Compute precision, recall, and F1-score
precision = precision_score(y_test, y_pred, average='weighted')
recall = recall_score(y_test, y_pred, average='weighted')
f1 = f1_score(y_test, y_pred, average='weighted')

# Compute confusion matrix
conf_matrix = confusion_matrix(y_test, y_pred)

# Print classification report
class_report = classification_report(y_test, y_pred)

# Print the evaluation metrics
print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1 Score:", f1)
print("Confusion Matrix:\n", conf_matrix)
print("Classification Report:\n", class_report)
```

```
Accuracy: 0.8231872818922062
Precision: 0.7883710756121971
Recall: 0.8231872818922062
F1 Score: 0.7978242356866035
Confusion Matrix:
 [[8111  479]
 [1345  381]]
Classification Report:
               precision    recall  f1-score   support

           0       0.86      0.94      0.90      8590
           1       0.44      0.22      0.29      1726

    accuracy                           0.82     10316
   macro avg       0.65      0.58      0.60     10316
weighted avg       0.79      0.82      0.80     10316
```

Intrepretation

KNN and Logical regression returned an accuracy rate of 83% while Decision trees returned an accuracy rate of 82%

Conclusion

- Most of the injuries recorded were caused by drivers between the age of 25 to 50 years falling into the adult category
- Based on the data, it shows that most accidents occur in the afternoon/rush hour.
- It also shows that most accidents occur in speed limit zones labeled between 30-40 mph.

- Most of the vehicles involved in the accidents fell into the old category that is they were in the road for a period of more than ten years
- Most of the accidents occured where there were no traffic signals but less injuries were observed

Recommendations

The following are our recommendations ;

- The city to put restrictions on vehicles allowed in roads to be below ten years
- The city lowers the speed limit during afternoon/rush hour or more patrol in the 30-40 mph zones.
- The city should increase the traffic control devices to reduce in accidents occurences

In [ ]: