Data Loading

Importing of libaries

In [2]:
```
# Import necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

Load datasets using the absolute paths

In [3]:
```
import pandas as pd

# Load datasets
crashes = pd.read_csv('C:/Users/MNJOROGE16/Desktop/Moringa/phase_3/project_
people = pd.read_csv('C:/Users/MNJOROGE16/Desktop/Moringa/phase_3/project__
vehicles = pd.read_csv('C:/Users/MNJOROGE16/Desktop/Moringa/phase_3/project
```

```
c:\Users\MNJOROGE16\AppData\Local\anaconda3\envs\learn-env\lib\site-packa
ges\IPython\core\interactiveshell.py:3145: DtypeWarning: Columns (19,23,2
4,25,28) have mixed types.Specify dtype option on import or set low_memor
y=False.
  has_raised = await self.run_ast_nodes(code_ast.body, cell_name,
c:\Users\MNJOROGE16\AppData\Local\anaconda3\envs\learn-env\lib\site-packa
ges\IPython\core\interactiveshell.py:3145: DtypeWarning: Columns (20,39,4
0,41,43,47,48,49,52,54,57,58,60,70) have mixed types.Specify dtype option
on import or set low_memory=False.
  has_raised = await self.run_ast_nodes(code_ast.body, cell_name,
```

Data Inspection

Reviewing each dataset structure (rows, columns)

In [4]:
```
print(crashes.shape)
```

```
(866411, 48)
```

In [5]:
```
print(people.shape)
```

```
(1467049, 29)
```

```
In [6]: print(vehicles.shape)
```

```
(1764900, 71)
```

Inspecting the first few rows of each data set

This is to confirm data is loaded correctly

```
In [7]: print(crashes.head())
```

```
                                     CRASH_RECORD_ID CRASH_DATE_EST_I  \
0  23a79931ef555d54118f64dc9be2cf2dbf59636ce253f7...              NaN   
1  2675c13fd0f474d730a5b780968b3cafc7c12d7adb661f...              NaN   
2  5f54a59fcb087b12ae5b1acff96a3caf4f2d37e79f8db4...              NaN   
3  7ebf015016f83d09b321afd671a836d6b148330535d5df...              NaN   
4  6c1659069e9c6285a650e70d6f9b574ed5f64c12888479...              NaN   

               CRASH_DATE  POSTED_SPEED_LIMIT TRAFFIC_CONTROL_DEVICE  \
0  09/05/2023 07:05:00 PM                  30         TRAFFIC SIGNAL   
1  09/22/2023 06:45:00 PM                  50            NO CONTROLS   
2  07/29/2023 02:45:00 PM                  30         TRAFFIC SIGNAL   
3  08/09/2023 11:00:00 PM                  30            NO CONTROLS   
4  08/18/2023 12:50:00 PM                  15                  OTHER   

      DEVICE_CONDITION WEATHER_CONDITION      LIGHTING_CONDITION  \
0  FUNCTIONING PROPERLY             CLEAR                    DUSK   
1           NO CONTROLS             CLEAR  DARKNESS, LIGHTED ROAD   
2  FUNCTIONING PROPERLY             CLEAR                DAYLIGHT   
3           NO CONTROLS             CLEAR  DARKNESS, LIGHTED ROAD   
4  FUNCTIONING PROPERLY             CLEAR                DAYLIGHT   

           FIRST_CRASH_TYPE                  TRAFFICWAY_TYPE  ...  \
0                     ANGLE            FIVE POINT, OR MORE  ...   
1                  REAR END         DIVIDED - W/MEDIAN BARRIER  ...   
2      PARKED MOTOR VEHICLE  DIVIDED - W/MEDIAN (NOT RAISED)  ...   
3  SIDESWIPE SAME DIRECTION                      NOT DIVIDED  ...   
4                  REAR END                            OTHER  ...   

   INJURIES_NON_INCAPACITATING INJURIES_REPORTED_NOT_EVIDENT  \
0                          2.0                           0.0   
1                          0.0                           0.0   
2                          0.0                           0.0   
3                          0.0                           0.0   
4                          1.0                           0.0   

  INJURIES_NO_INDICATION INJURIES_UNKNOWN CRASH_HOUR CRASH_DAY_OF_WEEK  \
0                    2.0              0.0         19                 3   
1                    2.0              0.0         18                 6   
2                    1.0              0.0         14                 7   
3                    2.0              0.0         23                 4   
4                    1.0              0.0         12                 6   

  CRASH_MONTH  LATITUDE  LONGITUDE                                  LOCAT
ION  
0           9       NaN        NaN                                        
NaN  
1           9       NaN        NaN                                        
NaN  
2           7  41.85412 -87.665902  POINT (-87.665902342962 41.8541202629
52)  
3           8       NaN        NaN                                        
NaN  
4           8       NaN        NaN                                        
NaN  

[5 rows x 48 columns]
```

```
In [8]: print(people.head())
```

```
  PERSON_ID PERSON_TYPE                                    CRASH_RECORD_I
D  \
0   O749947      DRIVER  81dc0de2ed92aa62baccab641fa377be7feb1cc47e655
4...
1   O871921      DRIVER  af84fb5c8d996fcd3aefd36593c3a02e6e7509eeb2756
8...
2    O10018      DRIVER  71162af7bf22799b776547132ebf134b5b438dcf3dac6
b...
3    O10038      DRIVER  c21c476e2ccc41af550b5d858d22aaac4ffc88745a170
0...
4    O10039      DRIVER  eb390a4c8e114c69488f5fb8a097fe629f5a92fd528cf
4...

   VEHICLE_ID              CRASH_DATE  SEAT_NO     CITY STATE ZIPCODE SEX
\
0    834816.0  09/28/2019 03:30:00 AM      NaN  CHICAGO    IL   60651   M
1    827212.0  04/13/2020 10:50:00 PM      NaN  CHICAGO    IL   60620   M
2      9579.0  11/01/2015 05:00:00 AM      NaN      NaN   NaN     NaN   X
3      9598.0  11/01/2015 08:00:00 AM      NaN      NaN   NaN     NaN   X
4      9600.0  11/01/2015 10:15:00 AM      NaN      NaN   NaN     NaN   X

   ...  EMS_RUN_NO     DRIVER_ACTION DRIVER_VISION PHYSICAL_CONDITION  \
0  ...         NaN           UNKNOWN       UNKNOWN            UNKNOWN
1  ...         NaN              NONE  NOT OBSCURED             NORMAL
2  ...         NaN  IMPROPER BACKING       UNKNOWN            UNKNOWN
3  ...         NaN           UNKNOWN       UNKNOWN            UNKNOWN
4  ...         NaN           UNKNOWN       UNKNOWN            UNKNOWN

  PEDPEDAL_ACTION PEDPEDAL_VISIBILITY PEDPEDAL_LOCATION        BAC_RESULT
\
0             NaN                 NaN               NaN  TEST NOT OFFERED
1             NaN                 NaN               NaN  TEST NOT OFFERED
2             NaN                 NaN               NaN  TEST NOT OFFERED
3             NaN                 NaN               NaN  TEST NOT OFFERED
4             NaN                 NaN               NaN  TEST NOT OFFERED

  BAC_RESULT VALUE CELL_PHONE_USE
0              NaN            NaN
1              NaN            NaN
2              NaN            NaN
3              NaN            NaN
4              NaN            NaN

[5 rows x 29 columns]
```

```
In [9]: print(vehicles.head())
```

```
   CRASH_UNIT_ID                                    CRASH_RECORD_ID  \
0        1727162  f5943b05f46b8d4148a63b7506a59113eae0cf1075aabc...
1        1717556  7b1763088507f77e0e552c009a6bf89a4d6330c7527706...
2        1717574  2603ff5a88f0b9b54576934c5ed4e4a64e8278e005687b...
3        1717579  a52ef70e33d468b855b5be44e8638a564434dcf99c0edf...
4        1720118  609055f4b1a72a44d6ec40ba9036cefd7c1287a755eb6c...

               CRASH_DATE  UNIT_NO   UNIT_TYPE  NUM_PASSENGERS  VEHICLE_I
D  \
0  12/21/2023 08:57:00 AM        2  PEDESTRIAN             NaN         Na
N
1  12/06/2023 03:24:00 PM        1      DRIVER             NaN   1634931.
0
2  12/06/2023 04:00:00 PM        2      DRIVER             NaN   1634978.
0
3  12/06/2023 04:30:00 PM        1      DRIVER             NaN   1634948.
0
4  12/10/2023 12:12:00 PM        1      DRIVER             NaN   1637401.
0

  CMRC_VEH_I      MAKE    MODEL  ... TRAILER1_LENGTH  TRAILER2_LENGTH  \
0        NaN       NaN      NaN  ...             NaN              NaN
1        NaN    NISSAN   SENTRA  ...             NaN              NaN
2        NaN  CHRYSLER  SEBRING  ...             NaN              NaN
3        NaN    SUBARU  OUTBACK  ...             NaN              NaN
4        NaN    TOYOTA     RAV4  ...             NaN              NaN

  TOTAL_VEHICLE_LENGTH AXLE_CNT VEHICLE_CONFIG CARGO_BODY_TYPE LOAD_TYPE
\
0                  NaN      NaN            NaN             NaN       NaN
1                  NaN      NaN            NaN             NaN       NaN
2                  NaN      NaN            NaN             NaN       NaN
3                  NaN      NaN            NaN             NaN       NaN
4                  NaN      NaN            NaN             NaN       NaN

  HAZMAT_OUT_OF_SERVICE_I MCS_OUT_OF_SERVICE_I  HAZMAT_CLASS
0                     NaN                  NaN           NaN
1                     NaN                  NaN           NaN
2                     NaN                  NaN           NaN
3                     NaN                  NaN           NaN
4                     NaN                  NaN           NaN

[5 rows x 71 columns]
```

Inspecting Column Names

In [10]:
```
print(crashes.columns)
```

```
Index(['CRASH_RECORD_ID', 'CRASH_DATE_EST_I', 'CRASH_DATE',
       'POSTED_SPEED_LIMIT', 'TRAFFIC_CONTROL_DEVICE', 'DEVICE_CONDITIO
N',
       'WEATHER_CONDITION', 'LIGHTING_CONDITION', 'FIRST_CRASH_TYPE',
       'TRAFFICWAY_TYPE', 'LANE_CNT', 'ALIGNMENT', 'ROADWAY_SURFACE_CON
D',
       'ROAD_DEFECT', 'REPORT_TYPE', 'CRASH_TYPE', 'INTERSECTION_RELATED_
I',
       'NOT_RIGHT_OF_WAY_I', 'HIT_AND_RUN_I', 'DAMAGE', 'DATE_POLICE_NOTI
FIED',
       'PRIM_CONTRIBUTORY_CAUSE', 'SEC_CONTRIBUTORY_CAUSE', 'STREET_NO',
       'STREET_DIRECTION', 'STREET_NAME', 'BEAT_OF_OCCURRENCE',
       'PHOTOS_TAKEN_I', 'STATEMENTS_TAKEN_I', 'DOORING_I', 'WORK_ZONE_
I',
       'WORK_ZONE_TYPE', 'WORKERS_PRESENT_I', 'NUM_UNITS',
       'MOST_SEVERE_INJURY', 'INJURIES_TOTAL', 'INJURIES_FATAL',
       'INJURIES_INCAPACITATING', 'INJURIES_NON_INCAPACITATING',
       'INJURIES_REPORTED_NOT_EVIDENT', 'INJURIES_NO_INDICATION',
       'INJURIES_UNKNOWN', 'CRASH_HOUR', 'CRASH_DAY_OF_WEEK', 'CRASH_MONT
H',
       'LATITUDE', 'LONGITUDE', 'LOCATION'],
      dtype='object')
```

In [11]:
```
print(people.columns)
```

```
Index(['PERSON_ID', 'PERSON_TYPE', 'CRASH_RECORD_ID', 'VEHICLE_ID',
       'CRASH_DATE', 'SEAT_NO', 'CITY', 'STATE', 'ZIPCODE', 'SEX', 'AGE',
       'DRIVERS_LICENSE_STATE', 'DRIVERS_LICENSE_CLASS', 'SAFETY_EQUIPMEN
T',
       'AIRBAG_DEPLOYED', 'EJECTION', 'INJURY_CLASSIFICATION', 'HOSPITA
L',
       'EMS_AGENCY', 'EMS_RUN_NO', 'DRIVER_ACTION', 'DRIVER_VISION',
       'PHYSICAL_CONDITION', 'PEDPEDAL_ACTION', 'PEDPEDAL_VISIBILITY',
       'PEDPEDAL_LOCATION', 'BAC_RESULT', 'BAC_RESULT VALUE',
       'CELL_PHONE_USE'],
      dtype='object')
```

```
In [12]: print(vehicles.columns)
```

```
Index(['CRASH_UNIT_ID', 'CRASH_RECORD_ID', 'CRASH_DATE', 'UNIT_NO',
       'UNIT_TYPE', 'NUM_PASSENGERS', 'VEHICLE_ID', 'CMRC_VEH_I', 'MAKE',
       'MODEL', 'LIC_PLATE_STATE', 'VEHICLE_YEAR', 'VEHICLE_DEFECT',
       'VEHICLE_TYPE', 'VEHICLE_USE', 'TRAVEL_DIRECTION', 'MANEUVER',
       'TOWED_I', 'FIRE_I', 'OCCUPANT_CNT', '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',
       'FIRST_CONTACT_POINT', 'CMV_ID', 'USDOT_NO', 'CCMC_NO', 'ILCC_NO',
       'COMMERCIAL_SRC', 'GVWR', 'CARRIER_NAME', 'CARRIER_STATE',
       'CARRIER_CITY', 'HAZMAT_PLACARDS_I', 'HAZMAT_NAME', 'UN_NO',
       'HAZMAT_PRESENT_I', 'HAZMAT_REPORT_I', 'HAZMAT_REPORT_NO',
       'MCS_REPORT_I', 'MCS_REPORT_NO', 'HAZMAT_VIO_CAUSE_CRASH_I',
       'MCS_VIO_CAUSE_CRASH_I', 'IDOT_PERMIT_NO', 'WIDE_LOAD_I',
       'TRAILER1_WIDTH', 'TRAILER2_WIDTH', 'TRAILER1_LENGTH',
       'TRAILER2_LENGTH', 'TOTAL_VEHICLE_LENGTH', 'AXLE_CNT', 'VEHICLE_CO
NFIG',
       'CARGO_BODY_TYPE', 'LOAD_TYPE', 'HAZMAT_OUT_OF_SERVICE_I',
       'MCS_OUT_OF_SERVICE_I', 'HAZMAT_CLASS'],
      dtype='object')
```

Data Types and Schema

In [13]:
```
# Check data types of each dataset
print(crashes.dtypes)


# Identify any incorrect data types (e.g., numeric columns read as objects)
```

```
CRASH_RECORD_ID                   object
CRASH_DATE_EST_I                  object
CRASH_DATE                        object
POSTED_SPEED_LIMIT                 int64
TRAFFIC_CONTROL_DEVICE            object
DEVICE_CONDITION                  object
WEATHER_CONDITION                 object
LIGHTING_CONDITION                object
FIRST_CRASH_TYPE                  object
TRAFFICWAY_TYPE                   object
LANE_CNT                         float64
ALIGNMENT                         object
ROADWAY_SURFACE_COND              object
ROAD_DEFECT                       object
REPORT_TYPE                       object
CRASH_TYPE                        object
INTERSECTION_RELATED_I            object
NOT_RIGHT_OF_WAY_I                object
HIT_AND_RUN_I                     object
DAMAGE                            object
DATE_POLICE_NOTIFIED              object
PRIM_CONTRIBUTORY_CAUSE           object
SEC_CONTRIBUTORY_CAUSE            object
STREET_NO                          int64
STREET_DIRECTION                  object
STREET_NAME                       object
BEAT_OF_OCCURRENCE               float64
PHOTOS_TAKEN_I                    object
STATEMENTS_TAKEN_I                object
DOORING_I                         object
WORK_ZONE_I                       object
WORK_ZONE_TYPE                    object
WORKERS_PRESENT_I                 object
NUM_UNITS                          int64
MOST_SEVERE_INJURY                object
INJURIES_TOTAL                   float64
INJURIES_FATAL                   float64
INJURIES_INCAPACITATING          float64
INJURIES_NON_INCAPACITATING      float64
INJURIES_REPORTED_NOT_EVIDENT    float64
INJURIES_NO_INDICATION           float64
INJURIES_UNKNOWN                 float64
CRASH_HOUR                         int64
CRASH_DAY_OF_WEEK                  int64
CRASH_MONTH                        int64
LATITUDE                         float64
LONGITUDE                        float64
LOCATION                          object
dtype: object
```

```
In [14]: print(people.dtypes)
```

```
PERSON_ID                  object
PERSON_TYPE                object
CRASH_RECORD_ID            object
VEHICLE_ID                float64
CRASH_DATE                 object
SEAT_NO                   float64
CITY                       object
STATE                      object
ZIPCODE                    object
SEX                        object
AGE                       float64
DRIVERS_LICENSE_STATE      object
DRIVERS_LICENSE_CLASS      object
SAFETY_EQUIPMENT           object
AIRBAG_DEPLOYED            object
EJECTION                   object
INJURY_CLASSIFICATION      object
HOSPITAL                   object
EMS_AGENCY                 object
EMS_RUN_NO                 object
DRIVER_ACTION              object
DRIVER_VISION              object
PHYSICAL_CONDITION         object
PEDPEDAL_ACTION            object
PEDPEDAL_VISIBILITY        object
PEDPEDAL_LOCATION          object
BAC_RESULT                 object
BAC_RESULT VALUE          float64
CELL_PHONE_USE             object
dtype: object
```

```
In [15]: print(vehicles.dtypes)
```

```
CRASH_UNIT_ID                int64
CRASH_RECORD_ID             object
CRASH_DATE                  object
UNIT_NO                      int64
UNIT_TYPE                   object
                             ...
CARGO_BODY_TYPE             object
LOAD_TYPE                   object
HAZMAT_OUT_OF_SERVICE_I     object
MCS_OUT_OF_SERVICE_I        object
HAZMAT_CLASS                object
Length: 71, dtype: object
```

Initial Summary Statistics

In [16]:
```
# Summary statistics for numerical features - Crashes
print(crashes.describe())
```

```
       POSTED_SPEED_LIMIT      LANE_CNT      STREET_NO  BEAT_OF_OCCURRENC
E  \
count       866411.000000  1.990150e+05  866411.000000       866406.00000
0
mean            28.415733  1.332981e+01    3687.152034         1244.46922
7
std              6.131785  2.961557e+03    2882.599171          705.12605
9
min              0.000000  0.000000e+00       0.000000          111.00000
0
25%             30.000000  2.000000e+00    1250.000000          714.00000
0
50%             30.000000  2.000000e+00    3201.000000         1212.00000
0
75%             30.000000  4.000000e+00    5580.000000         1822.00000
0
max             99.000000  1.191625e+06  451100.000000         6100.00000
0

           NUM_UNITS  INJURIES_TOTAL  INJURIES_FATAL  INJURIES_INCAPACITA
TING  \
count  866411.000000   864508.000000   864508.000000            864508.00
0000
mean        2.035117        0.192690        0.001194                 0.01
9823
std         0.452753        0.570222        0.037455                 0.16
4843
min         1.000000        0.000000        0.000000                 0.00
0000
25%         2.000000        0.000000        0.000000                 0.00
0000
50%         2.000000        0.000000        0.000000                 0.00
0000
75%         2.000000        0.000000        0.000000                 0.00
0000
max        18.000000       21.000000        4.000000                10.00
0000

       INJURIES_NON_INCAPACITATING  INJURIES_REPORTED_NOT_EVIDENT  \
count                864508.000000                  864508.000000   
mean                      0.108248                       0.063426   
std                       0.424294                       0.323900   
min                       0.000000                       0.000000   
25%                       0.000000                       0.000000   
50%                       0.000000                       0.000000   
75%                       0.000000                       0.000000   
max                      21.000000                      15.000000   

       INJURIES_NO_INDICATION  INJURIES_UNKNOWN     CRASH_HOUR  \
count           864508.000000          864508.0  866411.000000   
mean                 2.001795               0.0      13.205135   
std                  1.157261               0.0       5.573549   
min                  0.000000               0.0       0.000000   
25%                  1.000000               0.0       9.000000   
50%                  2.000000               0.0      14.000000   
75%                  2.000000               0.0      17.000000   
max                 61.000000               0.0      23.000000   

       CRASH_DAY_OF_WEEK    CRASH_MONTH       LATITUDE      LONGITUDE  
count      866411.000000  866411.000000  860273.000000  860273.000000  
mean            4.122962       6.606381      41.855078     -87.673657  
```

```
std             1.981495     3.377482     0.333591     0.677645
min             1.000000     1.000000     0.000000   -87.936193
25%             2.000000     4.000000    41.782879   -87.721774
50%             4.000000     7.000000    41.874945   -87.674177
75%             6.000000    10.000000    41.924490   -87.633463
max             7.000000    12.000000    42.022780     0.000000
```

In [17]:
```
# Summary statistics for numerical features - People
print(people.describe())
```

```
         VEHICLE_ID        SEAT_NO           AGE  BAC_RESULT VALUE
count  1.438245e+06  299132.000000  1.040853e+06       1775.000000
mean   6.905905e+05       4.160906  3.781694e+01          0.169448
std    4.038528e+05       2.198771  1.710846e+01          0.102295
min    2.000000e+00       1.000000 -1.770000e+02          0.000000
25%    3.444260e+05       3.000000  2.500000e+01          0.120000
50%    6.826620e+05       3.000000  3.500000e+01          0.170000
75%    1.034161e+06       5.000000  5.000000e+01          0.220000
max    1.801497e+06      12.000000  1.100000e+02          1.000000
```

In [18]:
```
# Summary statistics for numerical features - Vehicles
print(vehicles.describe())
```

```
       CRASH_UNIT_ID       UNIT_NO  NUM_PASSENGERS    VEHICLE_ID  \
count   1.764900e+06  1.764900e+06   261313.000000  1.724034e+06
mean    9.438774e+05  3.705683e+00        1.470750  8.976615e+05
std     5.463825e+05  2.843844e+03        1.055718  5.186095e+05
min     2.000000e+00  0.000000e+00        1.000000  2.000000e+00
25%     4.698658e+05  1.000000e+00        1.000000  4.489182e+05
50%     9.450715e+05  2.000000e+00        1.000000  8.959025e+05
75%     1.417380e+06  2.000000e+00        2.000000  1.346214e+06
max     1.888827e+06  3.778035e+06       59.000000  1.799377e+06

       VEHICLE_YEAR  OCCUPANT_CNT        CMV_ID  TRAILER1_LENGTH  \
count  1.448829e+06  1.724034e+06  17859.000000      2393.000000
mean   2.014207e+03  1.079142e+00   9960.036564        48.511910
std    1.385204e+02  7.815274e-01   5757.641443        20.695514
min    1.900000e+03  0.000000e+00      1.000000         1.000000
25%    2.007000e+03  1.000000e+00   4918.500000        45.000000
50%    2.013000e+03  1.000000e+00   9988.000000        53.000000
75%    2.017000e+03  1.000000e+00  14967.500000        53.000000
max    9.999000e+03  9.900000e+01  19878.000000       740.000000

       TRAILER2_LENGTH  TOTAL_VEHICLE_LENGTH      AXLE_CNT
count        70.000000           2918.000000   4396.000000
mean         44.271429             53.225497      9.619882
std          28.008240             31.291466    392.233256
min           1.000000              1.000000      1.000000
25%          24.250000             35.000000      2.000000
50%          50.000000             53.000000      3.000000
75%          53.000000             66.000000      5.000000
max         123.000000            999.000000  26009.000000
```

Data Relationships Before Dropping Columns with High Missing Values

Scatter Plots

To visualize relationships between pairs of numerical feature to spot trends, correlations, or anomalies.

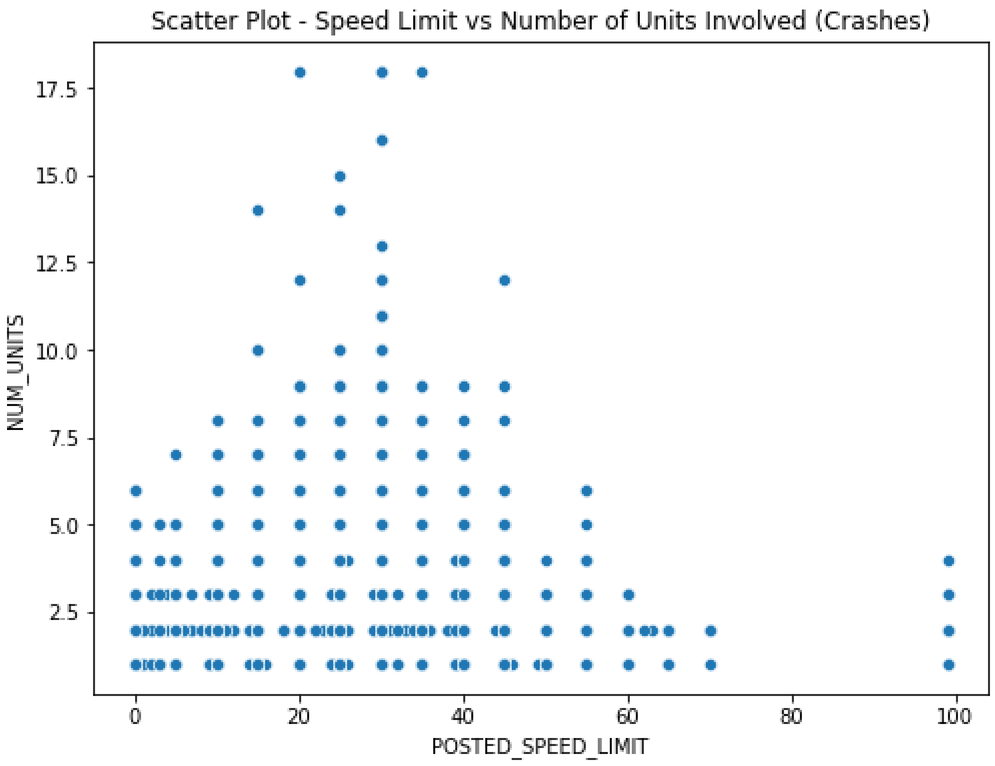
Correlation Matrix

To understand how numerical features are related to one another, which is crucial for avoiding multicollinearity and for feature selection.

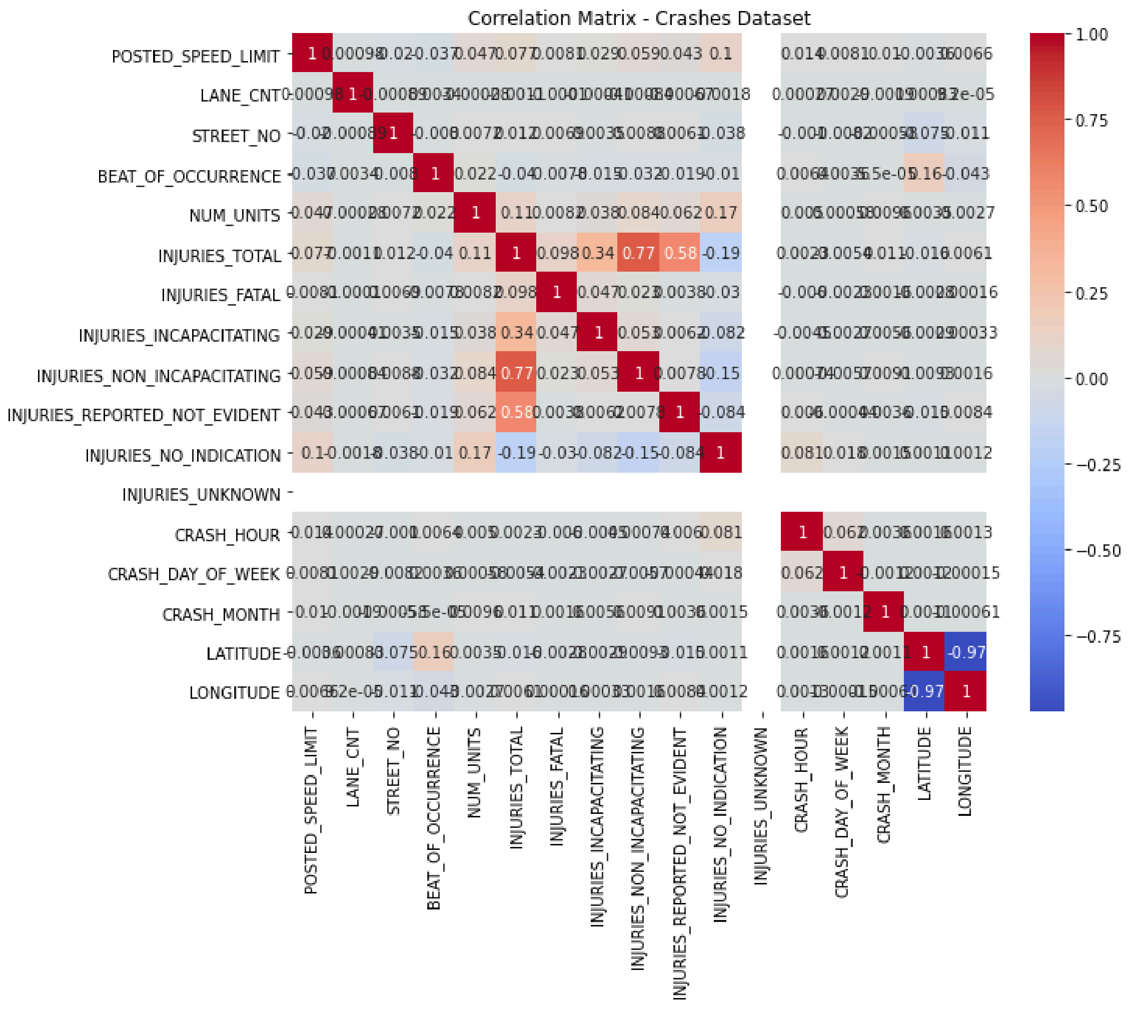
Cross-Tabulation

To explore relationships between categorical variables to show how the distribution of one categorical variable is related to another.

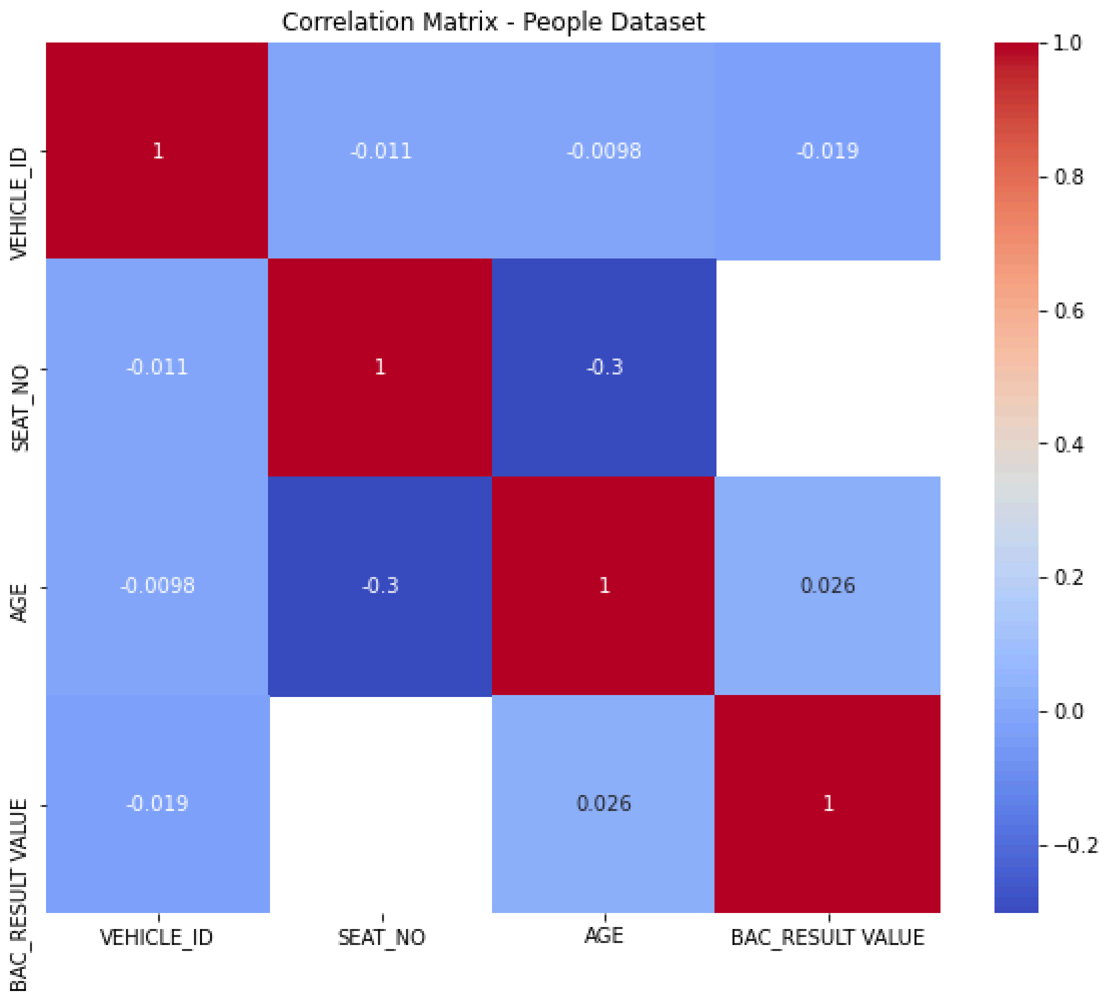
In [19]:
```
# Scatter plot to examine relationships between numerical features in Crash
plt.figure(figsize=(8, 6))
sns.scatterplot(x='POSTED_SPEED_LIMIT', y='NUM_UNITS', data=crashes)
plt.title('Scatter Plot - Speed Limit vs Number of Units Involved (Crashes)
plt.show()
```



```
In [20]: # Correlation matrix for numerical features in Crashes Dataset
         plt.figure(figsize=(10, 8))
         corr_matrix_crashes = crashes.corr()
         sns.heatmap(corr_matrix_crashes, annot=True, cmap='coolwarm')
         plt.title('Correlation Matrix - Crashes Dataset')
         plt.show()
```



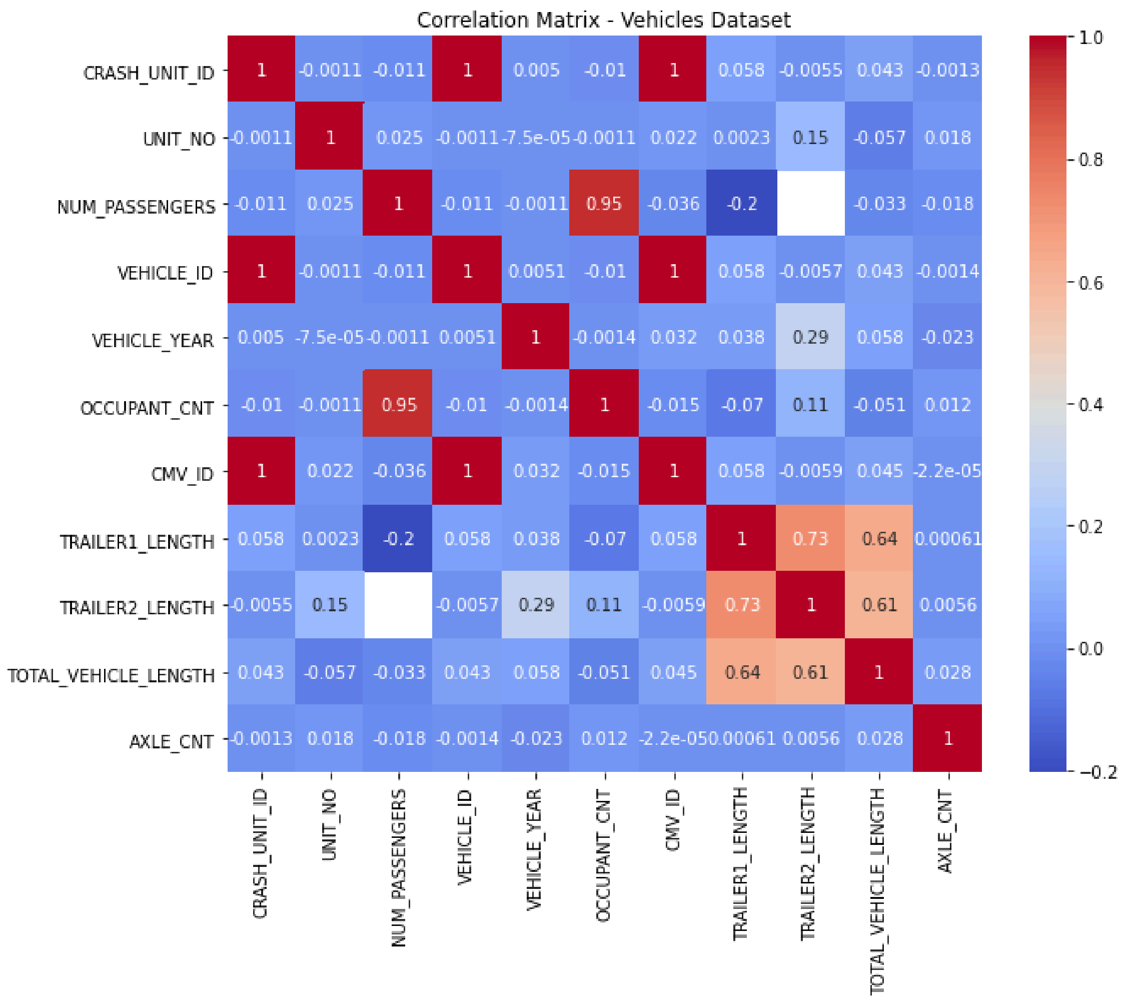
In [21]:
```
# Correlation matrix for numerical features in People Dataset
plt.figure(figsize=(10, 8))
corr_matrix_people = people.corr()
sns.heatmap(corr_matrix_people, annot=True, cmap='coolwarm')
plt.title('Correlation Matrix - People Dataset')
plt.show()
```



```
In [22]: # Cross-tabulation for categorical features in People Dataset
         cross_tab_people = pd.crosstab(people['PERSON_TYPE'], people['INJURY_CLASSI
         print("\nCross-tabulation of 'PERSON_TYPE' and 'INJURY_CLASSIFICATION' in P
```

```
Cross-tabulation of 'PERSON_TYPE' and 'INJURY_CLASSIFICATION' in People D
ataset:
 INJURY_CLASSIFICATION  FATAL  INCAPACITATING INJURY  NO INDICATION OF IN
JURY  \
PERSON_TYPE
BICYCLE                   31                    943                     3
185
DRIVER                   405                   6595                  1072
120
NON-CONTACT VEHICLE        0                      0
209
NON-MOTOR VEHICLE          3                     22
808
PASSENGER                144                   2960                   263
584
PEDESTRIAN               206                   2839                     2
251

INJURY_CLASSIFICATION  NONINCAPACITATING INJURY  REPORTED, NOT EVIDENT
PERSON_TYPE
BICYCLE                                    4926                   1140
DRIVER                                    37093                  23391
NON-CONTACT VEHICLE                           1                      0
NON-MOTOR VEHICLE                            58                     21
PASSENGER                                 19140                  13010
PEDESTRIAN                                 8781                   2539
```

In [23]:
```python
#Correlation matrix for numerical features in Crashes Dataset
plt.figure(figsize=(10, 8))
corr_matrix_vehicles = vehicles.corr()
sns.heatmap(corr_matrix_vehicles, annot=True, cmap='coolwarm')
plt.title('Correlation Matrix - Vehicles Dataset')
plt.show()
```



In [24]:
```python
# Cross-tabulation for categorical features in Vehicles Dataset
cross_tab_vehicles = pd.crosstab(vehicles['VEHICLE_TYPE'], crashes['CRASH_T
print("\nCross-tabulation of 'VEHICLE_TYPE' and 'CRASH_TYPE' in Vehicles Da
```

```
Cross-tabulation of 'VEHICLE_TYPE' and 'CRASH_TYPE' in Vehicles Dataset:
 CRASH_TYPE                                        INJURY AND / OR TOW DUE TO CRASH
\
VEHICLE_TYPE
3-WHEELED MOTORCYCLE (2 REAR WHEELS)                                           11
ALL-TERRAIN VEHICLE (ATV)                                                      26
AUTOCYCLE                                                                      74
BUS OVER 15 PASS.                                                            2222
BUS UP TO 15 PASS.                                                            775
FARM EQUIPMENT                                                                 11
MOPED OR MOTORIZED BICYCLE                                                    102
MOTOR DRIVEN CYCLE                                                             30
MOTORCYCLE (OVER 150CC)                                                       533
OTHER                                                                        2843
OTHER VEHICLE WITH TRAILER                                                    335
PASSENGER                                                                  142214
PICKUP                                                                       7746
RECREATIONAL OFF-HIGHWAY VEHICLE (ROV)                                          3
SINGLE UNIT TRUCK WITH TRAILER                                                459
SNOWMOBILE                                                                      1
SPORT UTILITY VEHICLE (SUV)                                                 32520
TRACTOR W/ SEMI-TRAILER                                                      2065
TRACTOR W/O SEMI-TRAILER                                                      259
TRUCK - SINGLE UNIT                                                          4009
UNKNOWN/NA                                                                  21034
VAN/MINI-VAN                                                                10165

CRASH_TYPE                               NO INJURY / DRIVE AWAY
VEHICLE_TYPE
3-WHEELED MOTORCYCLE (2 REAR WHEELS)                         31
ALL-TERRAIN VEHICLE (ATV)                                    74
AUTOCYCLE                                                   185
BUS OVER 15 PASS.                                          6097
BUS UP TO 15 PASS.                                         2092
FARM EQUIPMENT                                               23
MOPED OR MOTORIZED BICYCLE                                  281
MOTOR DRIVEN CYCLE                                           84
MOTORCYCLE (OVER 150CC)                                    1499
OTHER                                                      7941
OTHER VEHICLE WITH TRAILER                                  788
PASSENGER                                                387267
PICKUP                                                    21073
RECREATIONAL OFF-HIGHWAY VEHICLE (ROV)                       15
SINGLE UNIT TRUCK WITH TRAILER                             1190
SNOWMOBILE                                                    2
SPORT UTILITY VEHICLE (SUV)                               89083
TRACTOR W/ SEMI-TRAILER                                    5532
TRACTOR W/O SEMI-TRAILER                                    708
TRUCK - SINGLE UNIT                                       11390
UNKNOWN/NA                                                57099
VAN/MINI-VAN                                              27633
```

Missing Values

Objective: Identify missing data and assess the percentage of missing values per feature.

Identfying Missing Values

In [25]:
```python
# Identify missing values
print(crashes.isnull().sum())
```

```
CRASH_RECORD_ID                        0
CRASH_DATE_EST_I                  802055
CRASH_DATE                             0
POSTED_SPEED_LIMIT                     0
TRAFFIC_CONTROL_DEVICE                 0
DEVICE_CONDITION                       0
WEATHER_CONDITION                      0
LIGHTING_CONDITION                     0
FIRST_CRASH_TYPE                       0
TRAFFICWAY_TYPE                        0
LANE_CNT                          667396
ALIGNMENT                              0
ROADWAY_SURFACE_COND                   0
ROAD_DEFECT                            0
REPORT_TYPE                        26403
CRASH_TYPE                             0
INTERSECTION_RELATED_I            667808
NOT_RIGHT_OF_WAY_I                826744
HIT_AND_RUN_I                     594832
DAMAGE                                 0
DATE_POLICE_NOTIFIED                   0
PRIM_CONTRIBUTORY_CAUSE                0
SEC_CONTRIBUTORY_CAUSE                 0
STREET_NO                              0
STREET_DIRECTION                       4
STREET_NAME                            1
BEAT_OF_OCCURRENCE                     5
PHOTOS_TAKEN_I                    854736
STATEMENTS_TAKEN_I                846649
DOORING_I                         863692
WORK_ZONE_I                       861513
WORK_ZONE_TYPE                    862631
WORKERS_PRESENT_I                 865156
NUM_UNITS                              0
MOST_SEVERE_INJURY                  1916
INJURIES_TOTAL                      1903
INJURIES_FATAL                      1903
INJURIES_INCAPACITATING             1903
INJURIES_NON_INCAPACITATING         1903
INJURIES_REPORTED_NOT_EVIDENT       1903
INJURIES_NO_INDICATION              1903
INJURIES_UNKNOWN                    1903
CRASH_HOUR                             0
CRASH_DAY_OF_WEEK                      0
CRASH_MONTH                            0
LATITUDE                            6138
LONGITUDE                           6138
LOCATION                            6138
dtype: int64
```

```
In [26]: print(people.isnull().sum())
```

```
PERSON_ID                      0
PERSON_TYPE                    4
CRASH_RECORD_ID                4
VEHICLE_ID                 28804
CRASH_DATE                     5
SEAT_NO                  1167917
CITY                      395404
STATE                     381164
ZIPCODE                   487700
SEX                        22984
AGE                       426196
DRIVERS_LICENSE_STATE     605557
DRIVERS_LICENSE_CLASS     736377
SAFETY_EQUIPMENT            4149
AIRBAG_DEPLOYED            27652
EJECTION                   17696
INJURY_CLASSIFICATION        644
HOSPITAL                 1212552
EMS_AGENCY               1308279
EMS_RUN_NO               1440923
DRIVER_ACTION             301796
DRIVER_VISION             302212
PHYSICAL_CONDITION        300982
PEDPEDAL_ACTION          1439792
PEDPEDAL_VISIBILITY      1439851
PEDPEDAL_LOCATION        1439797
BAC_RESULT                300779
BAC_RESULT VALUE         1465274
CELL_PHONE_USE           1465892
dtype: int64
```

```
In [27]: print(vehicles.isnull().sum())
```

```
CRASH_UNIT_ID                  0
CRASH_RECORD_ID                0
CRASH_DATE                     0
UNIT_NO                        0
UNIT_TYPE                   2209
                          ...
CARGO_BODY_TYPE          1750780
LOAD_TYPE                1751403
HAZMAT_OUT_OF_SERVICE_I  1752567
MCS_OUT_OF_SERVICE_I     1752327
HAZMAT_CLASS             1763763
Length: 71, dtype: int64
```

Calculate the percentage of missing values

```
In [28]: crashes_missing_percentage = (crashes.isnull().sum() / len(crashes)) * 100
         print("Crashes Missing Data Percentage:\n", crashes_missing_percentage)
```

```
Crashes Missing Data Percentage:
 CRASH_RECORD_ID                   0.000000
CRASH_DATE_EST_I                92.572116
CRASH_DATE                       0.000000
POSTED_SPEED_LIMIT               0.000000
TRAFFIC_CONTROL_DEVICE           0.000000
DEVICE_CONDITION                 0.000000
WEATHER_CONDITION                0.000000
LIGHTING_CONDITION               0.000000
FIRST_CRASH_TYPE                 0.000000
TRAFFICWAY_TYPE                  0.000000
LANE_CNT                        77.029955
ALIGNMENT                        0.000000
ROADWAY_SURFACE_COND             0.000000
ROAD_DEFECT                      0.000000
REPORT_TYPE                      3.047399
CRASH_TYPE                       0.000000
INTERSECTION_RELATED_I          77.077507
NOT_RIGHT_OF_WAY_I              95.421688
HIT_AND_RUN_I                   68.654715
DAMAGE                           0.000000
DATE_POLICE_NOTIFIED             0.000000
PRIM_CONTRIBUTORY_CAUSE          0.000000
SEC_CONTRIBUTORY_CAUSE           0.000000
STREET_NO                        0.000000
STREET_DIRECTION                 0.000462
STREET_NAME                      0.000115
BEAT_OF_OCCURRENCE               0.000577
PHOTOS_TAKEN_I                  98.652487
STATEMENTS_TAKEN_I              97.719096
DOORING_I                       99.686177
WORK_ZONE_I                     99.434679
WORK_ZONE_TYPE                  99.563717
WORKERS_PRESENT_I               99.855150
NUM_UNITS                        0.000000
MOST_SEVERE_INJURY               0.221142
INJURIES_TOTAL                   0.219642
INJURIES_FATAL                   0.219642
INJURIES_INCAPACITATING          0.219642
INJURIES_NON_INCAPACITATING      0.219642
INJURIES_REPORTED_NOT_EVIDENT    0.219642
INJURIES_NO_INDICATION           0.219642
INJURIES_UNKNOWN                 0.219642
CRASH_HOUR                       0.000000
CRASH_DAY_OF_WEEK                0.000000
CRASH_MONTH                      0.000000
LATITUDE                         0.708440
LONGITUDE                        0.708440
LOCATION                         0.708440
dtype: float64
```

```
In [29]: people_missing_percentage = (people.isnull().sum() / len(people)) * 100
         print("People Missing Data Percentage:\n", people_missing_percentage)
```

```
People Missing Data Percentage:
 PERSON_ID                   0.000000
PERSON_TYPE                 0.000273
CRASH_RECORD_ID             0.000273
VEHICLE_ID                  1.963397
CRASH_DATE                  0.000341
SEAT_NO                    79.609952
CITY                       26.952338
STATE                      25.981682
ZIPCODE                    33.243607
SEX                         1.566683
AGE                        29.051245
DRIVERS_LICENSE_STATE      41.277217
DRIVERS_LICENSE_CLASS      50.194438
SAFETY_EQUIPMENT            0.282813
AIRBAG_DEPLOYED             1.884872
EJECTION                    1.206231
INJURY_CLASSIFICATION       0.043898
HOSPITAL                   82.652454
EMS_AGENCY                 89.177594
EMS_RUN_NO                 98.219146
DRIVER_ACTION              20.571637
DRIVER_VISION              20.599994
PHYSICAL_CONDITION         20.516152
PEDPEDAL_ACTION            98.142053
PEDPEDAL_VISIBILITY        98.146074
PEDPEDAL_LOCATION          98.142393
BAC_RESULT                 20.502315
BAC_RESULT VALUE           99.879009
CELL_PHONE_USE             99.921134
dtype: float64
```

```
In [30]: vehicles_missing_percentage = (vehicles.isnull().sum() / len(vehicles)) * 1
         print("Vehicles Missing Data Percentage:\n", vehicles_missing_percentage)
```

```
Vehicles Missing Data Percentage:
 CRASH_UNIT_ID                 0.000000
CRASH_RECORD_ID               0.000000
CRASH_DATE                    0.000000
UNIT_NO                       0.000000
UNIT_TYPE                     0.125163
                                ...
CARGO_BODY_TYPE              99.199955
LOAD_TYPE                    99.235254
HAZMAT_OUT_OF_SERVICE_I      99.301207
MCS_OUT_OF_SERVICE_I         99.287608
HAZMAT_CLASS                 99.935577
Length: 71, dtype: float64
```

Dropping Columns with High % of Missing Values

Columns to Keep Despite Missing Values

People Dataset

DRIVERS_LICENSE_CLASS (50.19% missing): Could be important for understanding driver qualifications.

Crashes Dataset

REPORT_TYPE (3.05% missing): Low enough missing values that it might be worth keeping.

In [31]:
```
# Crashes dataset
crashes_cleaned = crashes.drop(columns=['CRASH_DATE_EST_I', 'LANE_CNT', 'IN
```

In [32]:
```
# People dataset
people_cleaned = people.drop(columns=['SEAT_NO', 'HOSPITAL', 'EMS_AGENCY',
```

In [33]:
```
# Vehicles dataset
vehicles_cleaned = vehicles.drop(columns=['CARGO_BODY_TYPE', 'LOAD_TYPE', '
```

Save the cleaned datasets to the processed_data folder

In [34]:
```
#Save the cleaned datasets to the processed_data folder

output_folder = 'C:/Users/MNJOROGE16/Desktop/Moringa/phase_3/project__phase

crashes_cleaned.to_csv(output_folder + 'cleaned_crashes.csv', index=False)
people_cleaned.to_csv(output_folder + 'cleaned_people.csv', index=False)
vehicles_cleaned.to_csv(output_folder + 'cleaned_vehicles.csv', index=False

print("Cleaned datasets saved successfully to the processed_data folder.")
```

```
Cleaned datasets saved successfully to the processed_data folder.
```

Summary Statistics After Dropping Columns

Summary statistics for numerical and categorical variables to understand their distributions.

In [35]:
```
# Summary statistics for numerical features - Crashes
print("Crashes Summary Statistics After Dropping Missing Values:\n", crashe
Crashes Summary Statistics After Dropping Missing Values:
       POSTED_SPEED_LIMIT      STREET_NO  BEAT_OF_OCCURRENCE      NUM_UN
ITS  \
count       866411.000000  866411.000000       866406.000000  866411.0000
00
mean            28.415733    3687.152034         1244.469227       2.0351
17
std              6.131785    2882.599171          705.126059       0.4527
53
min              0.000000       0.000000          111.000000       1.0000
00
25%             30.000000    1250.000000          714.000000       2.0000
00
50%             30.000000    3201.000000         1212.000000       2.0000
00
75%             30.000000    5580.000000         1822.000000       2.0000
00
max             99.000000  451100.000000         6100.000000      18.0000
00

       INJURIES_TOTAL  INJURIES_FATAL  INJURIES_INCAPACITATING  \
count   864508.000000   864508.000000            864508.000000
mean         0.192690        0.001194                 0.019823
std          0.570222        0.037455                 0.164843
min          0.000000        0.000000                 0.000000
25%          0.000000        0.000000                 0.000000
50%          0.000000        0.000000                 0.000000
75%          0.000000        0.000000                 0.000000
max         21.000000        4.000000                10.000000

       INJURIES_NON_INCAPACITATING  INJURIES_REPORTED_NOT_EVIDENT  \
count                864508.000000                  864508.000000
mean                      0.108248                       0.063426
std                       0.424294                       0.323900
min                       0.000000                       0.000000
25%                       0.000000                       0.000000
50%                       0.000000                       0.000000
75%                       0.000000                       0.000000
max                      21.000000                      15.000000

       INJURIES_NO_INDICATION  INJURIES_UNKNOWN     CRASH_HOUR  \
count           864508.000000          864508.0  866411.000000
mean                 2.001795               0.0      13.205135
std                  1.157261               0.0       5.573549
min                  0.000000               0.0       0.000000
25%                  1.000000               0.0       9.000000
50%                  2.000000               0.0      14.000000
75%                  2.000000               0.0      17.000000
max                 61.000000               0.0      23.000000

       CRASH_DAY_OF_WEEK    CRASH_MONTH       LATITUDE      LONGITUDE
count      866411.000000  866411.000000  860273.000000  860273.000000
mean            4.122962       6.606381      41.855078     -87.673657
std             1.981495       3.377482       0.333591       0.677645
min             1.000000       1.000000       0.000000     -87.936193
25%             2.000000       4.000000      41.782879     -87.721774
50%             4.000000       7.000000      41.874945     -87.674177
75%             6.000000      10.000000      41.924490     -87.633463
max             7.000000      12.000000      42.022780       0.000000
```

In [36]:
```
# Summary statistics for numerical features - People
print("People Summary Statistics After Dropping Missing Values:\n", people_
```

```
People Summary Statistics After Dropping Missing Values:
          VEHICLE_ID           AGE
count  1.438245e+06  1.040853e+06
mean   6.905905e+05  3.781694e+01
std    4.038528e+05  1.710846e+01
min    2.000000e+00 -1.770000e+02
25%    3.444260e+05  2.500000e+01
50%    6.826620e+05  3.500000e+01
75%    1.034161e+06  5.000000e+01
max    1.801497e+06  1.100000e+02
```

In [37]:
```
# Summary statistics for numerical features - Vehicles
print("Vehicles Summary Statistics After Dropping Missing Values:\n", vehic
```

```
Vehicles Summary Statistics After Dropping Missing Values:
        CRASH_UNIT_ID       UNIT_NO  NUM_PASSENGERS    VEHICLE_ID  \
count   1.764900e+06  1.764900e+06   261313.000000  1.724034e+06
mean    9.438774e+05  3.705683e+00        1.470750  8.976615e+05
std     5.463825e+05  2.843844e+03        1.055718  5.186095e+05
min     2.000000e+00  0.000000e+00        1.000000  2.000000e+00
25%     4.698658e+05  1.000000e+00        1.000000  4.489182e+05
50%     9.450715e+05  2.000000e+00        1.000000  8.959025e+05
75%     1.417380e+06  2.000000e+00        2.000000  1.346214e+06
max     1.888827e+06  3.778035e+06       59.000000  1.799377e+06

       VEHICLE_YEAR  OCCUPANT_CNT        CMV_ID  TRAILER1_LENGTH  \
count  1.448829e+06  1.724034e+06  17859.000000      2393.000000
mean   2.014207e+03  1.079142e+00   9960.036564        48.511910
std    1.385204e+02  7.815274e-01   5757.641443        20.695514
min    1.900000e+03  0.000000e+00      1.000000         1.000000
25%    2.007000e+03  1.000000e+00   4918.500000        45.000000
50%    2.013000e+03  1.000000e+00   9988.000000        53.000000
75%    2.017000e+03  1.000000e+00  14967.500000        53.000000
max    9.999000e+03  9.900000e+01  19878.000000       740.000000

       TRAILER2_LENGTH  TOTAL_VEHICLE_LENGTH      AXLE_CNT
count        70.000000           2918.000000   4396.000000
mean         44.271429             53.225497      9.619882
std          28.008240             31.291466    392.233256
min           1.000000              1.000000      1.000000
25%          24.250000             35.000000      2.000000
50%          50.000000             53.000000      3.000000
75%          53.000000             66.000000      5.000000
max         123.000000            999.000000  26009.000000
```

Merging the three data sets

Merging the crashes_cleaned, people_cleaned, and vehicles_cleaned datasets on the common key (CRASH_RECORD_ID).

In [38]:
```
#Merge the cleaned datasets on 'CRASH_RECORD_ID'
merged_df = pd.merge(pd.merge(crashes_cleaned, people_cleaned, on='CRASH_RE

#Save the merged dataset to the specified folder
output_path = 'C:/Users/MNJOROGE16/Desktop/Moringa/phase_3/project__phase3/
merged_df.to_csv(output_path, index=False)

print(f"Cleaned merged dataset saved successfully to {output_path}")
```

```
Cleaned merged dataset saved successfully to C:/Users/MNJOROGE16/Desktop/
Moringa/phase_3/project__phase3/Project-Ph3-Chicago-Car-Crashes-Predictio
n/data/processed_data/cleaned_merged_traffic_crashes.csv
```

Inspection of the Merged Dataset

```
In [39]: # Inspect the first few rows of the merged dataset
         print(merged_df.head())
```

```
                                     CRASH_RECORD_ID             CRASH_DAT
E_x  \
0  004cd14d0303a9163aad69a2d7f341b7da2a8572b2ab33...  11/26/2019 08:38:00
AM
1  004cd14d0303a9163aad69a2d7f341b7da2a8572b2ab33...  11/26/2019 08:38:00
AM
2  004cd14d0303a9163aad69a2d7f341b7da2a8572b2ab33...  11/26/2019 08:38:00
AM
3  004cd14d0303a9163aad69a2d7f341b7da2a8572b2ab33...  11/26/2019 08:38:00
AM
4  359bf9f5872d646bb63576e55b1e0b480dc93c2b935ab5...  01/31/2022 07:45:00
PM

   POSTED_SPEED_LIMIT TRAFFIC_CONTROL_DEVICE DEVICE_CONDITION  \
0                  25            NO CONTROLS      NO CONTROLS
1                  25            NO CONTROLS      NO CONTROLS
2                  25            NO CONTROLS      NO CONTROLS
3                  25            NO CONTROLS      NO CONTROLS
4                  25            NO CONTROLS      NO CONTROLS

  WEATHER_CONDITION LIGHTING_CONDITION FIRST_CRASH_TYPE TRAFFICWAY_TYPE
\
0             CLEAR           DAYLIGHT       PEDESTRIAN         ONE-WAY
1             CLEAR           DAYLIGHT       PEDESTRIAN         ONE-WAY
2             CLEAR           DAYLIGHT       PEDESTRIAN         ONE-WAY
3             CLEAR           DAYLIGHT       PEDESTRIAN         ONE-WAY
4             CLEAR           DARKNESS         REAR END         ONE-WAY

            ALIGNMENT  ... MCS_VIO_CAUSE_CRASH_I IDOT_PERMIT_NO WIDE_LOAD
_I  \
0      CURVE ON GRADE  ...                   NaN            NaN          N
aN
1      CURVE ON GRADE  ...                   NaN            NaN          N
aN
2      CURVE ON GRADE  ...                   NaN            NaN          N
aN
3      CURVE ON GRADE  ...                   NaN            NaN          N
aN
4  STRAIGHT AND LEVEL  ...                   NaN            NaN          N
aN

  TRAILER1_WIDTH TRAILER2_WIDTH TRAILER1_LENGTH TRAILER2_LENGTH  \
0            NaN            NaN             NaN             NaN
1            NaN            NaN             NaN             NaN
2            NaN            NaN             NaN             NaN
3            NaN            NaN             NaN             NaN
4            NaN            NaN             NaN             NaN

  TOTAL_VEHICLE_LENGTH  AXLE_CNT VEHICLE_CONFIG
0                  NaN       NaN            NaN
1                  NaN       NaN            NaN
2                  NaN       NaN            NaN
3                  NaN       NaN            NaN
4                  NaN       NaN            NaN

[5 rows x 129 columns]
```

```
In [40]: # Verify the number of rows and columns
         print("Merged Dataset Shape:", merged_df.shape)
```

```
Merged Dataset Shape: (3076376, 129)
```

```
In [41]: # Check the column names and data types
         print("Merged Dataset Columns and Data Types:\n", merged_df.dtypes)
```

```
Merged Dataset Columns and Data Types:
 CRASH_RECORD_ID             object
CRASH_DATE_x                object
POSTED_SPEED_LIMIT           int64
TRAFFIC_CONTROL_DEVICE      object
DEVICE_CONDITION            object
                             ...
TRAILER1_LENGTH            float64
TRAILER2_LENGTH            float64
TOTAL_VEHICLE_LENGTH       float64
AXLE_CNT                   float64
VEHICLE_CONFIG              object
Length: 129, dtype: object
```

Handling Missing Values- Merged Dataset

Identifying and address any missing values that may affect the analysis and modeling process.

```
In [42]: # Identify missing values in the merged dataset
         missing_values = merged_df.isnull().sum()

         print (missing_values)
```

```
CRASH_RECORD_ID                  0
CRASH_DATE_x                     0
POSTED_SPEED_LIMIT               0
TRAFFIC_CONTROL_DEVICE           0
DEVICE_CONDITION                 0
                              ...
TRAILER1_LENGTH            3072274
TRAILER2_LENGTH            3076240
TOTAL_VEHICLE_LENGTH       3070784
AXLE_CNT                   3068380
VEHICLE_CONFIG             3050604
Length: 129, dtype: int64
```

In [43]:
```
# Calculate the percentage of missing values for each feature
missing_percentage = (missing_values / len(merged_df)) * 100

# Display features with missing values
print("Missing Values in Merged Dataset:\n", missing_percentage[missing_per
```

```
Missing Values in Merged Dataset:
 REPORT_TYPE                3.832431
STREET_DIRECTION          0.000455
STREET_NAME               0.000130
BEAT_OF_OCCURRENCE        0.000780
PHOTOS_TAKEN_I           98.674317
                            ...
TRAILER1_LENGTH          99.866661
TRAILER2_LENGTH          99.995579
TOTAL_VEHICLE_LENGTH     99.818228
AXLE_CNT                 99.740084
VEHICLE_CONFIG           99.162261
Length: 94, dtype: float64
```

Drop Features with Extremely High Missing Values

Criteria: Features with more than 90% missing data are typically considered for removal unless they are critical.

Action: Drop features like PHOTOS_TAKEN_I, TRAILER1_LENGTH, TRAILER2_LENGTH, TOTAL_VEHICLE_LENGTH, AXLE_CNT, and VEHICLE_CONFIG because their missing rates are extremely high (> 99%).

In [44]:
```
# Drop features with more than 90% missing values
columns_to_drop = ['PHOTOS_TAKEN_I', 'TRAILER1_LENGTH', 'TRAILER2_LENGTH',
merged_df_dropped = merged_df.drop(columns=columns_to_drop)

print(f"Dropped columns: {columns_to_drop}")
print("New dataset shape after dropping columns:", merged_df_dropped.shape)
```

```
Dropped columns: ['PHOTOS_TAKEN_I', 'TRAILER1_LENGTH', 'TRAILER2_LENGTH',
'TOTAL_VEHICLE_LENGTH', 'AXLE_CNT', 'VEHICLE_CONFIG']
New dataset shape after dropping columns: (3076376, 123)
```

Impute Missing Values for Important Features

Criteria: For features with lower missing rates (e.g., REPORT_TYPE, STREET_DIRECTION, STREET_NAME, BEAT_OF_OCCURRENCE), imputation is appropriate.

Imputation Methods: Categorical Features: Impute missing values using the mode (most frequent value).

Numerical Features: If any were present, you could use mean, median, or other statistical methods.

```
In [45]: # Impute missing values for categorical features with the mode
         merged_df_dropped['REPORT_TYPE'].fillna(merged_df_dropped['REPORT_TYPE'].mc
         merged_df_dropped['STREET_DIRECTION'].fillna(merged_df_dropped['STREET_DIRE
         merged_df_dropped['STREET_NAME'].fillna(merged_df_dropped['STREET_NAME'].mc
         merged_df_dropped['BEAT_OF_OCCURRENCE'].fillna(merged_df_dropped['BEAT_OF_C

         # Check if there are any remaining missing values after imputation
         print("Remaining Missing Values After Imputation:\n", merged_df_dropped.isr
```

```
Remaining Missing Values After Imputation:
 STATEMENTS_TAKEN_I       2998531
DOORING_I                3066855
WORK_ZONE_I              3058959
WORK_ZONE_TYPE           3062786
WORKERS_PRESENT_I        3072073
                          ...
MCS_VIO_CAUSE_CRASH_I    3054541
IDOT_PERMIT_NO           3074778
WIDE_LOAD_I              3076133
TRAILER1_WIDTH           3071381
TRAILER2_WIDTH           3075765
Length: 84, dtype: int64
```

```
In [46]: # Identify columns with a high percentage of missing values (e.g., >90%)
         high_missing_columns = merged_df_dropped.columns[merged_df_dropped.isnull()

         print (high_missing_columns)
```

```
Index(['STATEMENTS_TAKEN_I', 'DOORING_I', 'WORK_ZONE_I', 'WORK_ZONE_TYP
E',
       'WORKERS_PRESENT_I', 'PEDPEDAL_VISIBILITY', 'PEDPEDAL_LOCATION',
       'CMRC_VEH_I', 'FIRE_I', 'EXCEED_SPEED_LIMIT_I', 'TOWED_TO', 'AREA_
00_I',
       'AREA_03_I', 'AREA_04_I', 'AREA_09_I', 'AREA_10_I', 'AREA_99_I',
       'CMV_ID', 'USDOT_NO', 'CCMC_NO', 'ILCC_NO', 'COMMERCIAL_SRC', 'GVW
R',
       'CARRIER_NAME', 'CARRIER_STATE', 'CARRIER_CITY', 'HAZMAT_PLACARDS_
I',
       'HAZMAT_NAME', 'UN_NO', 'HAZMAT_PRESENT_I', 'HAZMAT_REPORT_I',
       'HAZMAT_REPORT_NO', 'MCS_REPORT_I', 'MCS_REPORT_NO',
       'HAZMAT_VIO_CAUSE_CRASH_I', 'MCS_VIO_CAUSE_CRASH_I', 'IDOT_PERMIT_
NO',
       'WIDE_LOAD_I', 'TRAILER1_WIDTH', 'TRAILER2_WIDTH'],
      dtype='object')
```

In [47]:
```
# Drop these columns
merged_df_final = merged_df_dropped.drop(columns=high_missing_columns)

print(f"Dropped columns with high missing values: {high_missing_columns}")
print("New dataset shape after dropping high-missing-value columns:", merge
```

```
Dropped columns with high missing values: Index(['STATEMENTS_TAKEN_I', 'D
OORING_I', 'WORK_ZONE_I', 'WORK_ZONE_TYPE',
       'WORKERS_PRESENT_I', 'PEDPEDAL_VISIBILITY', 'PEDPEDAL_LOCATION',
       'CMRC_VEH_I', 'FIRE_I', 'EXCEED_SPEED_LIMIT_I', 'TOWED_TO', 'AREA_
00_I',
       'AREA_03_I', 'AREA_04_I', 'AREA_09_I', 'AREA_10_I', 'AREA_99_I',
       'CMV_ID', 'USDOT_NO', 'CCMC_NO', 'ILCC_NO', 'COMMERCIAL_SRC', 'GVW
R',
       'CARRIER_NAME', 'CARRIER_STATE', 'CARRIER_CITY', 'HAZMAT_PLACARDS_
I',
       'HAZMAT_NAME', 'UN_NO', 'HAZMAT_PRESENT_I', 'HAZMAT_REPORT_I',
       'HAZMAT_REPORT_NO', 'MCS_REPORT_I', 'MCS_REPORT_NO',
       'HAZMAT_VIO_CAUSE_CRASH_I', 'MCS_VIO_CAUSE_CRASH_I', 'IDOT_PERMIT_
NO',
       'WIDE_LOAD_I', 'TRAILER1_WIDTH', 'TRAILER2_WIDTH'],
      dtype='object')
New dataset shape after dropping high-missing-value columns: (3076376, 8
3)
```

In [48]:
```
# Check for any remaining missing values in the final dataset
remaining_missing_values = merged_df_final.isnull().sum()
remaining_missing_percentage = (remaining_missing_values / len(merged_df_fi

# Display features with remaining missing values
print("Remaining Missing Values After Dropping High-Missing-Value Columns:\
```

```
Remaining Missing Values After Dropping High-Missing-Value Columns:
 MOST_SEVERE_INJURY         0.000683
LATITUDE                   0.619723
LONGITUDE                  0.619723
LOCATION                   0.619723
VEHICLE_ID_x               2.039608
CITY                      26.815545
STATE                     25.820706
ZIPCODE                   33.082497
SEX                        1.587745
AGE                       28.843938
DRIVERS_LICENSE_STATE     41.419287
DRIVERS_LICENSE_CLASS     50.295575
SAFETY_EQUIPMENT           0.309195
AIRBAG_DEPLOYED            1.936369
EJECTION                   1.269481
INJURY_CLASSIFICATION      0.050709
DRIVER_ACTION             20.782083
DRIVER_VISION             20.812768
PHYSICAL_CONDITION        20.722955
BAC_RESULT                20.705824
UNIT_TYPE                  0.119264
NUM_PASSENGERS            74.837504
VEHICLE_ID_y               2.310738
MAKE                       2.311128
MODEL                      2.323578
LIC_PLATE_STATE            9.615665
VEHICLE_YEAR              15.715277
VEHICLE_DEFECT             2.310738
VEHICLE_TYPE               2.310738
VEHICLE_USE                2.310738
TRAVEL_DIRECTION           2.310738
MANEUVER                   2.310738
TOWED_I                   86.295888
OCCUPANT_CNT               2.310738
TOWED_BY                  89.707305
AREA_01_I                 71.485215
AREA_02_I                 83.198900
AREA_05_I                 84.180347
AREA_06_I                 84.124892
AREA_07_I                 86.402767
AREA_08_I                 84.885593
AREA_11_I                 82.518782
AREA_12_I                 82.184297
FIRST_CONTACT_POINT        2.517768
dtype: float64
```

In [49]:
```python
# Impute categorical features with the mode
for column in ['MOST_SEVERE_INJURY', 'SEX', 'DRIVERS_LICENSE_STATE', 'DRIVE
               'AIRBAG_DEPLOYED', 'EJECTION', 'INJURY_CLASSIFICATION', 'DRI
               'PHYSICAL_CONDITION', 'UNIT_TYPE', 'MAKE', 'FIRST_CONTACT_PO
    merged_df_final[column].fillna(merged_df_final[column].mode()[0], inpla

# Impute numerical features with the median
for column in ['LATITUDE', 'LONGITUDE', 'AGE']:
    merged_df_final[column].fillna(merged_df_final[column].median(), inplac
In [50]: # Check for any remaining missing values
         remaining_missing_values = merged_df_final.isnull().sum()
         print("Remaining missing values:\n", remaining_missing_values[remaining_mis


         print (remaining_missing_values)
```

```
Remaining missing values:
 LOCATION                 19065
VEHICLE_ID_x            62746
CITY                   824947
STATE                  794342
ZIPCODE               1017742
BAC_RESULT             636989
NUM_PASSENGERS        2302283
VEHICLE_ID_y            71087
MODEL                   71482
LIC_PLATE_STATE        295814
VEHICLE_YEAR           483461
VEHICLE_DEFECT          71087
VEHICLE_TYPE            71087
VEHICLE_USE             71087
TRAVEL_DIRECTION        71087
MANEUVER                71087
TOWED_I               2654786
OCCUPANT_CNT            71087
TOWED_BY              2759734
AREA_01_I             2199154
AREA_02_I             2559511
AREA_05_I             2589704
AREA_06_I             2587998
AREA_07_I             2658074
AREA_08_I             2611400
AREA_11_I             2538588
AREA_12_I             2528298
dtype: int64
CRASH_RECORD_ID                0
CRASH_DATE_x                   0
POSTED_SPEED_LIMIT             0
TRAFFIC_CONTROL_DEVICE         0
DEVICE_CONDITION               0
                            ...   
AREA_07_I                2658074
AREA_08_I                2611400
AREA_11_I                2538588
AREA_12_I                2528298
FIRST_CONTACT_POINT            0
Length: 83, dtype: int64
```

Analyze Feature Importance and Missing Data Percentage

For each feature, consider its importance to the model and the percentage of missing data. This will guide whether to impute or drop the feature.

High Importance & Low Missing Data (<20%): Impute missing values.

High Importance & High Missing Data (>20%): Consider imputation if the feature is crucial, otherwise consider dropping.

Low Importance & High Missing Data (>20%): Likely candidates for dropping.

Review of Missing Data Percentages

From the previous data, we have the following features with missing values and their percentages:

LOCATION: 0.62% missing

VEHICLE_ID_x: 2.04% missing

CITY: 26.82% missing

STATE: 25.82% missing

ZIPCODE: 33.08% missing

BAC_RESULT: 20.71% missing

NUM_PASSENGERS: 74.84% missing

VEHICLE_ID_y: 2.31% missing

MODEL: 2.32% missing

LIC_PLATE_STATE: 9.61% missing

VEHICLE_YEAR: 15.72% missing

VEHICLE_DEFECT: 2.31% missing

VEHICLE_TYPE: 2.31% missing

VEHICLE_USE: 2.31% missing

TRAVEL_DIRECTION: 2.31% missing

MANEUVER: 2.31% missing

TOWED_I: 86.34% missing

OCCUPANT_CNT: 2.31% missing

TOWED_BY: 89.70% missing

AREA_01_I - AREA_12_I: Varies, mostly >70% missing

High Importance Features based on their importance to (INJURY_CLASSIFICATION) and the percentage of missing data.

LOCATION (0.62%): High importance, low missing data. Impute.

VEHICLE_ID_x (2.04%) and VEHICLE_ID_y (2.31%): Moderate importance for identifying specific vehicles, low missing data. Impute.

MODEL (2.32%): High importance for determining vehicle type, low missing data. Impute.

LIC_PLATE_STATE (9.61%): High importance for location-based analysis, moderate missing data. Impute.

VEHICLE_YEAR (15.72%): High importance for determining vehicle age, moderate missing data. Impute.

VEHICLE_TYPE, VEHICLE_DEFECT, VEHICLE_USE, TRAVEL_DIRECTION, MANEUVER, OCCUPANT_CNT (All ~2.31%): High importance for understanding crash

Moderate to Low Importance Features

BAC_RESULT (20.71%): Important, but a higher percentage of missing data. Impute

CITY (26.82%) and STATE (25.82%): Important for location-based analysis but high missing data. Consider Dropping.

ZIPCODE (33.08%): Similar to CITY and STATE, consider dropping due to high missing data.

TOWED_I (86.34%): Low importance, very high missing data. Drop.

TOWED_BY (89.70%): Low importance, very high missing data. Drop.

AREA_01_I - AREA_12_I (>70%): Low importance, very high missing data. Drop

Further Imputing and Dropping

Impute

LOCATION

VEHICLE_ID_x, VEHICLE_ID_y

MODEL

LIC_PLATE_STATE

VEHICLE_YEAR

VEHICLE_TYPE, VEHICLE_DEFECT, VEHICLE_USE, TRAVEL_DIRECTION, MANEUVER, OCCUPANT_CNT

BAC_RESULT

Drop

CITY, STATE, ZIPCODE: Due to the high percentage of missing data, these features should be dropped unless you have strong reasons to keep them.

TOWED_I, TOWED_BY: Drop due to very high missing data and low importance.

AREA_01_I - AREA_12_I: Drop due to very high missing data and low importance.

In [51]:
```
# Impute high importance features with missing data
for column in ['LOCATION', 'VEHICLE_ID_x', 'VEHICLE_ID_y', 'MODEL', 'LIC_PL
               'VEHICLE_DEFECT', 'VEHICLE_TYPE', 'VEHICLE_USE', 'TRAVEL_DIR
               'OCCUPANT_CNT', 'BAC_RESULT']:
    merged_df_final[column].fillna(merged_df_final[column].mode()[0], inpla
```

In [52]:
```
# Drop features with high missing data and low importance
columns_to_drop = ['CITY', 'STATE', 'ZIPCODE', 'TOWED_I', 'TOWED_BY',
                   'AREA_01_I', 'AREA_02_I', 'AREA_05_I', 'AREA_06_I',
                   'AREA_07_I', 'AREA_11_I', 'AREA_12_I']

merged_df_final = merged_df_final.drop(columns=columns_to_drop)

print(f"Dropped columns: {columns_to_drop}")
print("Dataset shape after dropping columns:", merged_df_final.shape)
```

```
Dropped columns: ['CITY', 'STATE', 'ZIPCODE', 'TOWED_I', 'TOWED_BY', 'ARE
A_01_I', 'AREA_02_I', 'AREA_05_I', 'AREA_06_I', 'AREA_07_I', 'AREA_11_I',
'AREA_12_I']
Dataset shape after dropping columns: (3076376, 71)
```

In [53]:
```
# Final check for remaining missing values
remaining_missing_values = merged_df_final.isnull().sum()
print("Remaining missing values:\n", remaining_missing_values[remaining_mis
```

```
Remaining missing values:
 NUM_PASSENGERS    2302283
AREA_08_I         2611400
dtype: int64
```

In [54]:
```
# Drop the remaining features with high missing values
columns_to_drop = ['NUM_PASSENGERS', 'AREA_08_I']

merged_df_final = merged_df_final.drop(columns=columns_to_drop)
```

In [55]:
```
# Final check for any remaining missing values
remaining_missing_values = merged_df_final.isnull().sum()
print("Final check for remaining missing values:\n", remaining_missing_valu
```

```
Final check for remaining missing values:
 Series([], dtype: int64)
```

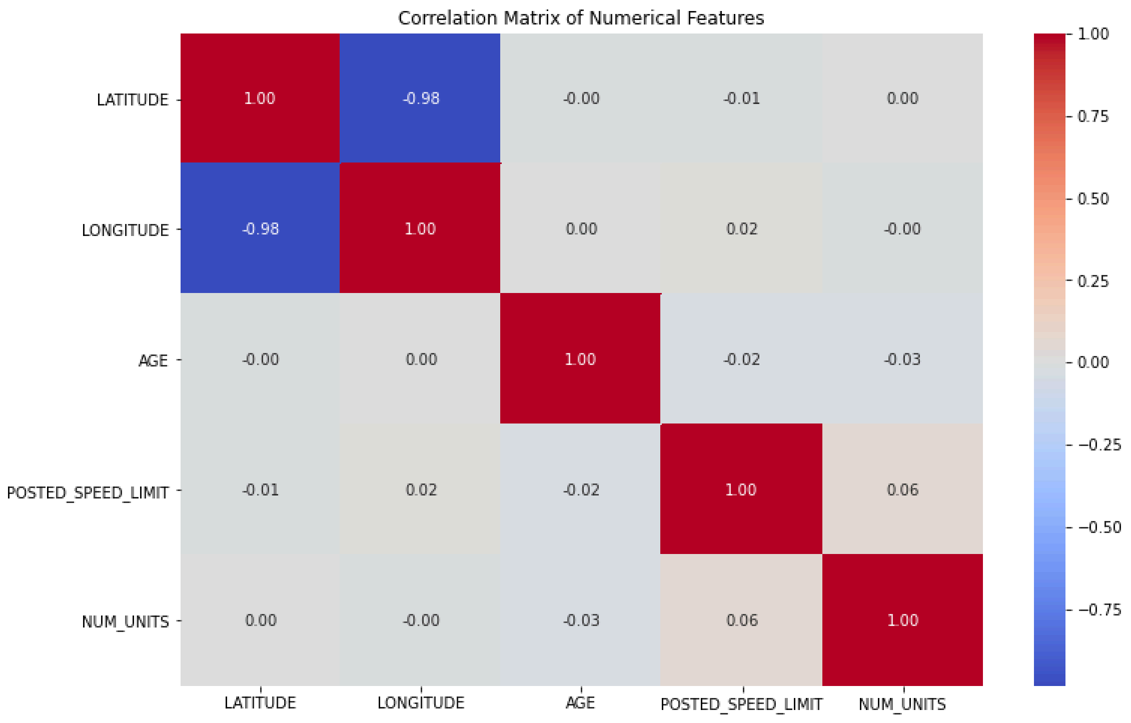
In [56]:
```
# Verify the shape of the final dataset
print("Final dataset shape after dropping remaining columns:", merged_df_fi
```

```
Final dataset shape after dropping remaining columns: (3076376, 69)
```

Data Relationships

In [57]:
```python
# Define the numerical variables of interest
numerical_features = ['LATITUDE', 'LONGITUDE', 'AGE', 'POSTED_SPEED_LIMIT',
```

In [58]:
```python
# Define the path where the image will be saved
save_path = r'C:\Users\MNJOROGE16\Desktop\Moringa\phase_3\project__phase3\P

# Correlation Matrix and Heatmap for Numerical Features
plt.figure(figsize=(12, 8))
corr_matrix = merged_df_final[numerical_features].corr()
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation Matrix of Numerical Features')

# Save the figure as a PNG file
plt.savefig(save_path)

# Display the plot
plt.show()
```



	LATITUDE	LONGITUDE	AGE	POSTED_SPEED_LIMIT	NUM_UNITS
LATITUDE	1.00	-0.98	-0.00	-0.01	0.00
LONGITUDE	-0.98	1.00	0.00	0.02	-0.00
AGE	-0.00	0.00	1.00	-0.02	-0.03
POSTED_SPEED_LIMIT	-0.01	0.02	-0.02	1.00	0.06
NUM_UNITS	0.00	-0.00	-0.03	0.06	1.00

Correlation Matrix Heatmap Interpretation

AGE Correlation with other features - AGE has almost no correlation with POSTED_SPEED_LIMIT (-0.02), NUM_UNITS (-0.03), LATITUDE (-0.00), and LONGITUDE (0.00).

Interpretation - Age does not appear to have a linear relationship with any of the other numerical features. This suggests that age might not directly influence or be influenced by these factors in the context of car crashes, making it a potentially independent variable in the model.

POSTED_SPEED_LIMIT Correlation with other features - POSTED_SPEED_LIMIT has very weak correlations with NUM_UNITS (0.06), LATITUDE (-0.01), and LONGITUDE (0.02).

Interpretation - The speed limit at the crash location shows minimal correlation with other numerical variables. This suggests that posted speed limits do not strongly interact with these variables, implying that speed limits might act independently in predicting injury severity.

NUM_UNITS Correlation with other features - NUM_UNITS has a weak correlation with POSTED_SPEED_LIMIT (0.06) and negligible correlation with other features.

Interpretation - The number of units involved in an accident (likely vehicles) is somewhat related to the speed limit, which makes sense, as different traffic conditions and regulations might influence both. However, the overall low correlations suggest that the number of units is generally independent of the other numerical features.

LATITUDE and LONGITUDE

Correlation with each other - There is a strong negative correlation between LATITUDE and LONGITUDE (-0.98).

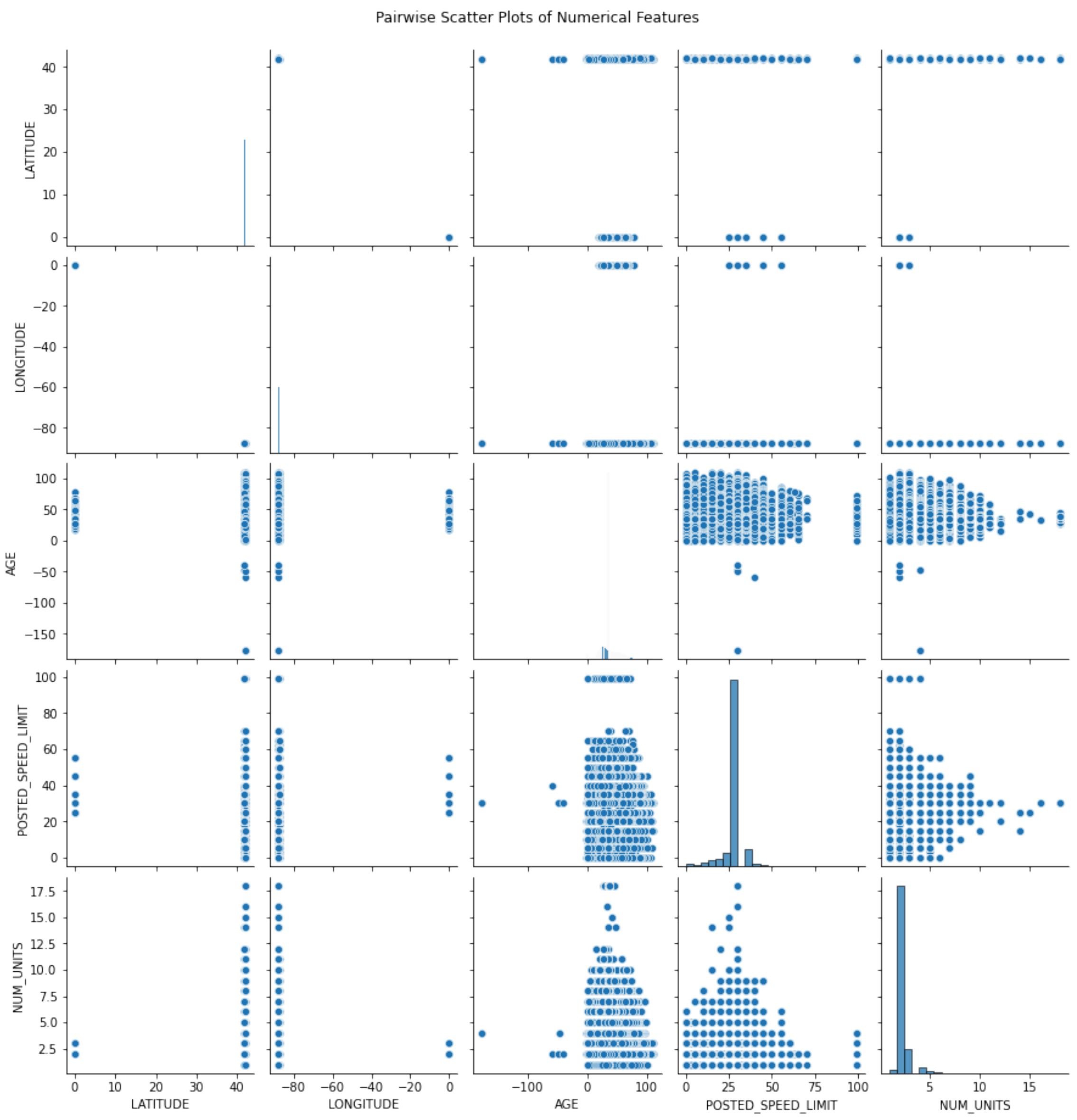
Interpretation - The strong negative correlation between latitude and longitude indicates that these two variables are closely related, likely due to the geographical layout of the region covered in the dataset. This relationship could be important for geospatial analysis but might not directly impact injury classification unless location-based patterns are significant.

Conclusion

Low Correlation Among Most Features - Most of the numerical features have very low correlations with each other. This suggests that these variables operate relatively independently, which is useful for modeling as it reduces the risk of multicollinearity, which can distort the predictive power of individual features.

Latitude and Longitude - The strong correlation between LATITUDE and LONGITUDE highlights that these two variables are geographically dependent. This could be useful if you plan to include geospatial analysis or location-based features in your model. However, since their correlation is high, you might consider using one or combining them into a new feature to avoid redundancy.

```
In [59]: # Define the save path
         save_path = r'C:\Users\MNJOROGE16\Desktop\Moringa\phase_3\project__phase3\P

         # Pairwise Scatter Plots for Numerical Features
         pair_plot = sns.pairplot(merged_df_final[numerical_features])

         # Adjust the title placement
         pair_plot.fig.suptitle('Pairwise Scatter Plots of Numerical Features', y=1.

         # Save the figure as a PNG file
         pair_plot.savefig(save_path)

         # Display the plot
         plt.show()
```



Interpretation of Pairwise Scatter Plots for Numerical Features

AGE vs. Other Features

AGE vs. POSTED_SPEED_LIMIT

Observation - There is no clear linear relationship between AGE and POSTED_SPEED_LIMIT. The data points are scattered widely across different speed limits, regardless of age.

Interpretation - Age does not seem to influence the speed limit at which accidents occur, indicating these features are likely independent.

AGE vs. NUM_UNITS

Observation - The scatter plot shows a broad distribution with no distinct pattern. Most accidents involve 2-3 units across all ages.

Interpretation - The number of vehicles involved in an accident does not appear to be directly related to the age of individuals involved.

AGE vs. LATITUDE/LONGITUDE

Observation - There's no visible relationship between age and geographical coordinates (latitude and longitude). The points are dispersed uniformly.

Interpretation - Age is not influenced by or correlated with the location of the accident, which aligns with expectations since age should not directly impact where an accident occurs.

POSTED_SPEED_LIMIT vs. Other Features

POSTED_SPEED_LIMIT vs. NUM_UNITS

Observation - There's a slight pattern where accidents with higher speed limits involve slightly fewer vehicles, but the relationship is weak.

Interpretation - While there might be a mild trend that higher speed limits involve fewer vehicles, it is not a strong correlation. This could suggest that speed and the number of vehicles involved operate relatively independently.

POSTED_SPEED_LIMIT vs. LATITUDE/LONGITUDE

Observation - No clear patterns are visible, indicating that speed limits do not vary significantly by location within the region covered by the dataset.

Interpretation - This suggests that the speed limit is fairly consistent across different geographical areas in the dataset.

NUM_UNITS vs. Other Features

NUM_UNITS vs. LATITUDE/LONGITUDE

Observation - Similar to the other plots, there's no distinct relationship between the number of units involved in an accident and the geographical coordinates.

Interpretation - The number of units involved in accidents does not vary significantly by location, suggesting that accident severity or scale (in terms of units involved) is not location-dependent within the dataset.

LATITUDE vs. LONGITUDE Observation - The scatter plot between LATITUDE and LONGITUDE shows a strong linear relationship, which is expected since they represent geographic coordinates. The linearity reflects the physical layout of the area where the data was collected.

Interpretation - The strong correlation between latitude and longitude reinforces that these features are related to the same underlying factor (location). For modeling purposes, these may be combined or used in location-based analysis.

Conclusion Independence of Features - Most numerical features, such as age, speed limit, and number of units, show little to no correlation with each other. This suggests that these features can independently contribute to the predictive power of your model.

Geographical Coordinates - The strong correlation between latitude and longitude is expected, but they do not show any relationship with other numerical features like age or speed limit.

Weak Relationships - The scatter plots suggest that the numerical features may not be strongly interrelated, which is beneficial for avoiding multicollinearity in the model.

Exploring Relationships Between Categorical Variables and the Target Variable

```
In [60]: # Cross-tabulation example: VEHICLE_TYPE vs INJURY_CLASSIFICATION
         cross_tab = pd.crosstab(merged_df_final['VEHICLE_TYPE'], merged_df_final['I
         print("Cross-Tabulation between Vehicle Type and Injury Classification:\n",
Cross-Tabulation between Vehicle Type and Injury Classification:
 INJURY_CLASSIFICATION                   FATAL  INCAPACITATING INJURY  \
VEHICLE_TYPE
3-WHEELED MOTORCYCLE (2 REAR WHEELS)        0                      7
ALL-TERRAIN VEHICLE (ATV)                   0                     24
AUTOCYCLE                                   1                      3
BUS OVER 15 PASS.                          19                    271
BUS UP TO 15 PASS.                          1                     24
FARM EQUIPMENT                              0                      1
MOPED OR MOTORIZED BICYCLE                  4                     65
MOTOR DRIVEN CYCLE                          2                     40
MOTORCYCLE (OVER 150CC)                    60                    543
OTHER                                      19                    261
OTHER VEHICLE WITH TRAILER                  1                     18
PASSENGER                                1215                  21198
PICKUP                                     51                    798
RECREATIONAL OFF-HIGHWAY VEHICLE (ROV)      0                      4
SINGLE UNIT TRUCK WITH TRAILER              1                     38
SNOWMOBILE                                  0                      0
SPORT UTILITY VEHICLE (SUV)               157                   3214
TRACTOR W/ SEMI-TRAILER                    26                    122
TRACTOR W/O SEMI-TRAILER                    9                     31
TRUCK - SINGLE UNIT                        26                    271
UNKNOWN/NA                                 86                   1306
VAN/MINI-VAN                               66                   1249

INJURY_CLASSIFICATION                   NO INDICATION OF INJURY  \
VEHICLE_TYPE
3-WHEELED MOTORCYCLE (2 REAR WHEELS)                         68
ALL-TERRAIN VEHICLE (ATV)                                   225
AUTOCYCLE                                                  1012
BUS OVER 15 PASS.                                         37132
BUS UP TO 15 PASS.                                         7988
FARM EQUIPMENT                                              119
MOPED OR MOTORIZED BICYCLE                                  494
MOTOR DRIVEN CYCLE                                          449
MOTORCYCLE (OVER 150CC)                                    4578
OTHER                                                     30746
OTHER VEHICLE WITH TRAILER                                 3703
PASSENGER                                               1821315
PICKUP                                                    84392
RECREATIONAL OFF-HIGHWAY VEHICLE (ROV)                       27
SINGLE UNIT TRUCK WITH TRAILER                             3581
SNOWMOBILE                                                    9
SPORT UTILITY VEHICLE (SUV)                              390581
TRACTOR W/ SEMI-TRAILER                                   22971
TRACTOR W/O SEMI-TRAILER                                   3334
TRUCK - SINGLE UNIT                                       48304
UNKNOWN/NA                                               202049
VAN/MINI-VAN                                             139828

INJURY_CLASSIFICATION                   NONINCAPACITATING INJURY  \
VEHICLE_TYPE
3-WHEELED MOTORCYCLE (2 REAR WHEELS)                          18
ALL-TERRAIN VEHICLE (ATV)                                     47
AUTOCYCLE                                                     50
BUS OVER 15 PASS.                                           1795
BUS UP TO 15 PASS.                                           237
FARM EQUIPMENT                                                10
MOPED OR MOTORIZED BICYCLE                                   135
MOTOR DRIVEN CYCLE                                           105
```

```
MOTORCYCLE (OVER 150CC)                                     1178
OTHER                                                       1249
OTHER VEHICLE WITH TRAILER                                   128
PASSENGER                                                 111587
PICKUP                                                      3991
RECREATIONAL OFF-HIGHWAY VEHICLE (ROV)                        10
SINGLE UNIT TRUCK WITH TRAILER                               107
SNOWMOBILE                                                     0
SPORT UTILITY VEHICLE (SUV)                                17803
TRACTOR W/ SEMI-TRAILER                                      728
TRACTOR W/O SEMI-TRAILER                                     138
TRUCK - SINGLE UNIT                                         1578
UNKNOWN/NA                                                  6602
VAN/MINI-VAN                                                6820

INJURY_CLASSIFICATION                 REPORTED, NOT EVIDENT
VEHICLE_TYPE
3-WHEELED MOTORCYCLE (2 REAR WHEELS)                        1
ALL-TERRAIN VEHICLE (ATV)                                  10
AUTOCYCLE                                                   5
BUS OVER 15 PASS.                                        1250
BUS UP TO 15 PASS.                                        120
FARM EQUIPMENT                                              3
MOPED OR MOTORIZED BICYCLE                                 31
MOTOR DRIVEN CYCLE                                         22
MOTORCYCLE (OVER 150CC)                                   210
OTHER                                                     677
OTHER VEHICLE WITH TRAILER                                 57
PASSENGER                                               61314
PICKUP                                                   2435
RECREATIONAL OFF-HIGHWAY VEHICLE (ROV)                      0
SINGLE UNIT TRUCK WITH TRAILER                             58
SNOWMOBILE                                                  0
SPORT UTILITY VEHICLE (SUV)                             12243
TRACTOR W/ SEMI-TRAILER                                   487
TRACTOR W/O SEMI-TRAILER                                   89
TRUCK - SINGLE UNIT                                      1068
UNKNOWN/NA                                               3431
VAN/MINI-VAN                                             4412
```

Interpretation of the Cross-Tabulation between Vehicle Type and Injury Classification

Passenger Vehicles FATAL Injuries: Passenger vehicles are involved in a significantly higher number of fatal injuries (1,215 cases). INCAPACITATING INJURY: Passenger vehicles also lead in incapacitating injuries, with 21,198 cases. NONINCAPACITATING INJURY: Again, passenger vehicles are involved in the majority of non-incapacitating injuries (49,444 cases).

Interpretation: Passenger vehicles are the most common vehicle type involved in accidents, which explains their high numbers across all injury classifications. Their involvement in severe injuries (both fatal and incapacitating) highlights the importance of focusing on safety measures for passenger vehicles.

Sport Utility Vehicles (SUVs) FATAL Injuries: SUVs are involved in 157 fatal injuries.

INCAPACITATING INJURY: There are 3,214 cases of incapacitating injuries involving SUVs.

NONINCAPACITATING INJURY: SUVs are involved in 6,409 non-incapacitating injuries.

Interpretation: SUVs also show high numbers across all injury classifications, indicating that, similar to passenger vehicles, SUVs are commonly involved in accidents that result in injuries. This might be due to their popularity and prevalence on the roads.

Motorcycles (Over 150cc) FATAL Injuries: Motorcycles are associated with a significant number of fatal injuries (60 cases).

INCAPACITATING INJURY: There are 543 cases of incapacitating injuries involving motorcycles.

NONINCAPACITATING INJURY: Motorcycles are involved in 918 non-incapacitating injuries.

Interpretation: Motorcycles, although less common than passenger vehicles, have a notably high rate of severe injuries relative to their numbers, especially fatal and incapacitating injuries. This suggests that accidents involving motorcycles are more likely to result in serious injury, likely due to the lack of protection for riders.

Trucks and Commercial Vehicles FATAL Injuries: Trucks and single-unit trucks are involved in several fatal injuries (e.g., 26 for tractor w/ semi-trailer, 51 for pickup trucks).

INCAPACITATING INJURY: These vehicles also show considerable numbers in incapacitating injuries. NONINCAPACITATING INJURY: These vehicles are involved in a fair number of non-incapacitating injuries, though lower than passenger vehicles and SUVs.

Interpretation: Commercial vehicles like trucks and tractors, while not as frequently involved in accidents as passenger vehicles, still contribute to a significant number of severe injuries. This highlights the potential risks associated with larger vehicles.

Unknown/NA Vehicle Types FATAL Injuries: Unknown/NA vehicle types are involved in 86 fatal injuries.

INCAPACITATING INJURY: There are 1,306 cases of incapacitating injuries involving these vehicles.

NONINCAPACITATING INJURY: These vehicles are involved in 3,431 non-incapacitating injuries.

Interpretation - The "Unknown/NA" category represents vehicles that were either not properly identified or categorized. The relatively high numbers in this category suggest that data quality or vehicle identification might be an issue, and improving this could lead to better insights.

Conclusion Passenger Vehicles and SUVs: These are the most frequently involved in accidents, leading in all injury classifications. This is likely due to their prevalence on the road.

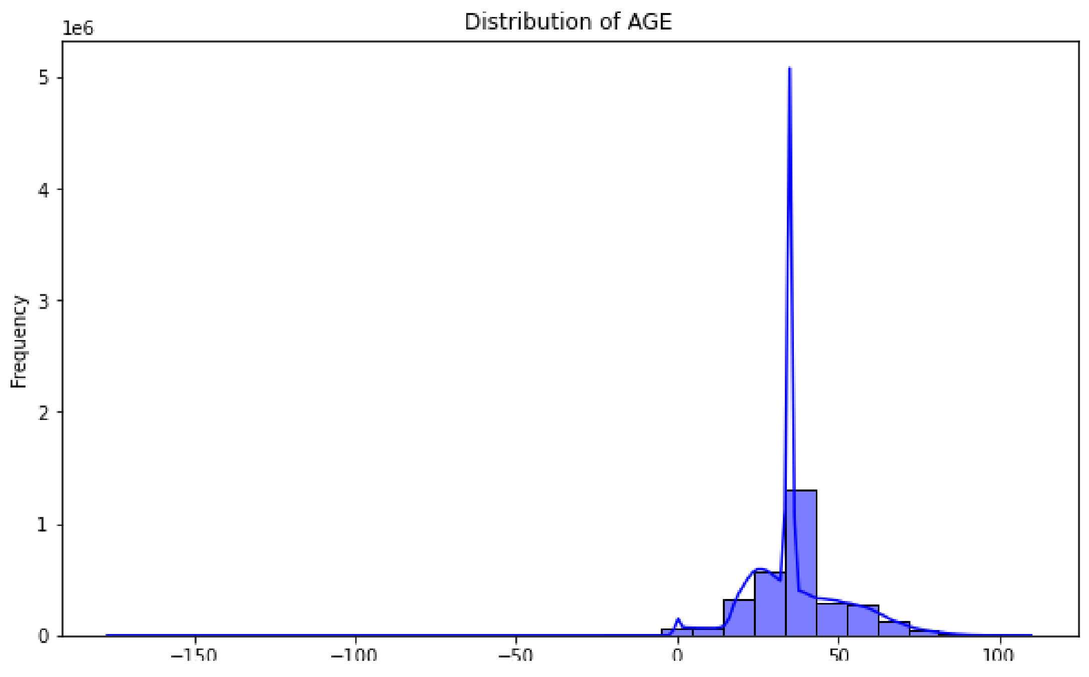
Motorcycles: Despite lower overall numbers, motorcycles are disproportionately involved in severe injuries, indicating a higher risk associated with motorcycle accidents.

Commercial Vehicles: Trucks and similar vehicles contribute significantly to severe injuries, underscoring the risks associated with larger vehicles.

Feature Exploration

Explore the Distribution of Numerical Features

In [61]:
```
# List of numerical features to explore
numerical_features = ['AGE', 'POSTED_SPEED_LIMIT', 'NUM_UNITS', 'LATITUDE',

# Base path where images will be saved
save_base_path = r'C:\Users\MNJOROGE16\Desktop\Moringa\phase_3\project__pha

# Plot histograms for numerical features and save them
for feature in numerical_features:
    plt.figure(figsize=(10, 6))
    sns.histplot(merged_df_final[feature], bins=30, kde=True, color='blue')
    plt.title(f'Distribution of {feature}')
    plt.xlabel(feature)
    plt.ylabel('Frequency')

    # Define the save path for each plot
    save_path = f'{save_base_path}\\distribution_{feature}.png'

    # Save the figure as a PNG file
    plt.savefig(save_path)

    # Show the plot
    plt.show()
```



Interpretation of the distributions of the numerical features

Distribution of AGE

Observation - The AGE distribution is heavily skewed, with a sharp peak around the 30-40 age range. There are also extreme outliers with negative and very high ages, which are likely data entry errors.

Interpretation - The peak around 30-40 years suggests that this age group is the most frequently involved in traffic accidents. The negative and extremely high values indicate data issues that should be addressed (e.g., by removing or correcting these outliers). For modeling, age is expected to be a significant factor in predicting injury severity, but the data quality needs to be improved for accurate predictions.

Distribution of LATITUDE

Observation -The LATITUDE values are clustered within a narrow range, which aligns with the geographical area covered by the dataset. The distribution shows that most of the data points are concentrated in a specific latitude range.

Interpretation - This indicates that the accidents are occurring within a specific geographical area, likely corresponding to the city or region being studied. Since latitude alone might not directly impact injury classification, it could be combined with longitude or used in geospatial analyses.

Distribution of LONGITUDE

Observation - Similar to latitude, the LONGITUDE values are tightly clustered within a narrow range, with the vast majority of values within a specific interval.

Interpretation - The longitude distribution supports the finding that the dataset is geographically concentrated in a particular region. Like latitude, longitude may not directly influence injury severity but could be useful in combination with other features or for location-based analysis.

Distribution of NUM_UNITS

Observation - The NUM_UNITS (number of units involved in the crash) distribution is heavily skewed to the left, with the majority of accidents involving 2-3 units. There are a few outliers with a higher number of units.

Interpretation - Most accidents involve a small number of units, typically 2-3 vehicles. The presence of outliers suggests that some accidents involve significantly more vehicles, which could be associated with more complex scenarios or higher severity, but these cases are rare.

Distribution of POSTED_SPEED_LIMIT

Observation - The POSTED_SPEED_LIMIT distribution shows a sharp peak around 20-40 mph, which is typical for urban areas. There are outliers at both the low and high ends of the speed limit range.

Interpretation - The concentration around 20-40 mph suggests that most accidents occur in urban settings where these speed limits are common. The outliers at lower and higher speed limits might correspond to rural or highway areas. The posted speed limit is likely an important factor in determining injury severity, especially when combined with other features like vehicle type or maneuver.

Conclusion

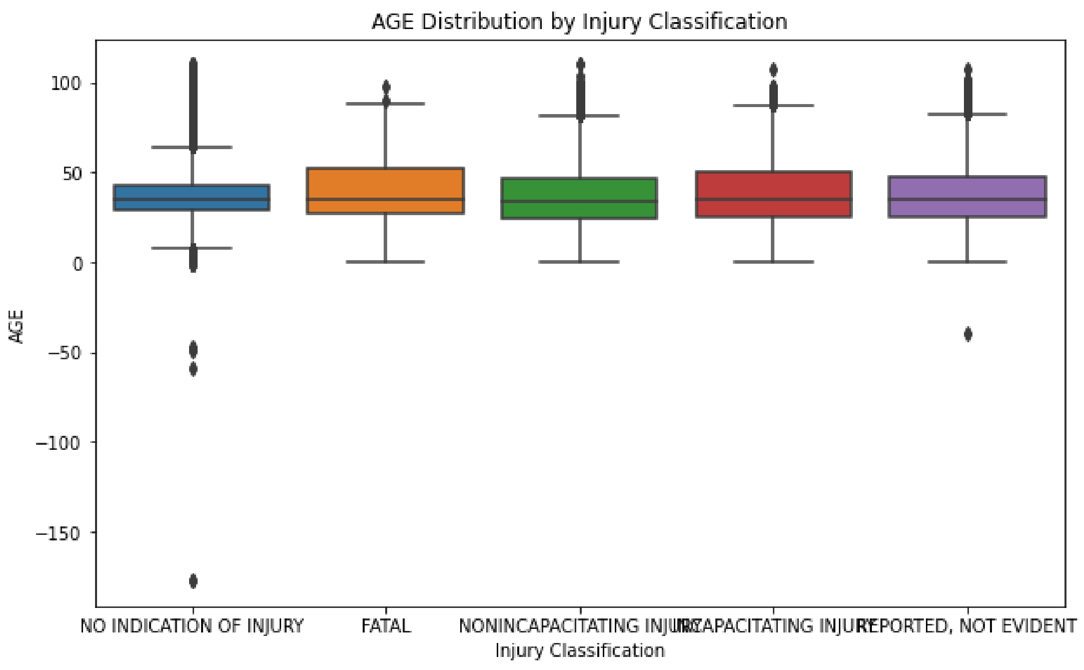
Age - This is a key feature with a significant peak in the 30-40 age range, but data cleaning is needed due to outliers.

Latitude and Longitude - Both are tightly clustered, indicating a specific geographical focus. They might not be directly predictive of injury classification but could be used in combination for location-based analysis.

Number of Units - Most accidents involve a small number of units, which could influence injury severity predictions.

Examine Relationships Between Numerical Features and the Target Variable

In [62]:
```
# Plot box plots for numerical features against the target variable

# List of numerical features to explore
numerical_features = ['AGE', 'POSTED_SPEED_LIMIT', 'NUM_UNITS', 'LATITUDE',

# Base path where images will be saved
save_base_path = r'C:\Users\MNJOROGE16\Desktop\Moringa\phase_3\project__pha

# Plot box plots for numerical features against the target variable and sav
for feature in numerical_features:
    plt.figure(figsize=(10, 6))
    sns.boxplot(x='INJURY_CLASSIFICATION', y=feature, data=merged_df_final)
    plt.title(f'{feature} Distribution by Injury Classification')
    plt.xlabel('Injury Classification')
    plt.ylabel(feature)

    # Define the save path for each plot
    save_path = f'{save_base_path}\\boxplot_{feature}_vs_injury_classificat

    # Save the figure as a PNG file
    plt.savefig(save_path)

    # Show the plot
    plt.show()
```



Boxplot Output Interpretation

AGE Distribution by Injury Classification Observation - The AGE distribution across different injury classifications is relatively similar, with the median age being around 30-40 years old for all injury types. There are outliers, with some negative and extremely high values, which might indicate data entry errors.

Interpretation - The similarity in age distribution across injury classifications suggests that age alone might not be a strong predictor of injury severity. However, the presence of outliers, especially negative values, indicates a need for further data cleaning or handling of these erroneous entries.

LATITUDE Distribution by Injury Classification Observation - The LATITUDE values are clustered tightly around a narrow range, likely corresponding to the geographical region covered by the dataset. There are outliers with very low values, which might be erroneous or indicate locations outside the expected range.

Interpretation - The tight clustering of latitude values suggests that the accidents occur within a specific geographical area. The outliers could represent data errors or unusual cases that might need to be handled separately.

LONGITUDE Distribution by Injury Classification

Observation - Similar to latitude, the LONGITUDE values are clustered within a specific range, with outliers that have very low or negative values. Interpretation: The longitude data shows a similar pattern to latitude, with most data points concentrated in a specific geographical region. The outliers here also suggest potential data entry errors or unusual cases.

NUM_UNITS Distribution by Injury Classification

Observation - The NUM_UNITS (number of units involved in the crash) generally ranges between 2 and 3 for most injury classifications, with higher numbers being less common. Outliers with higher numbers of units involved are present, particularly in the "NO INDICATION OF INJURY" category.

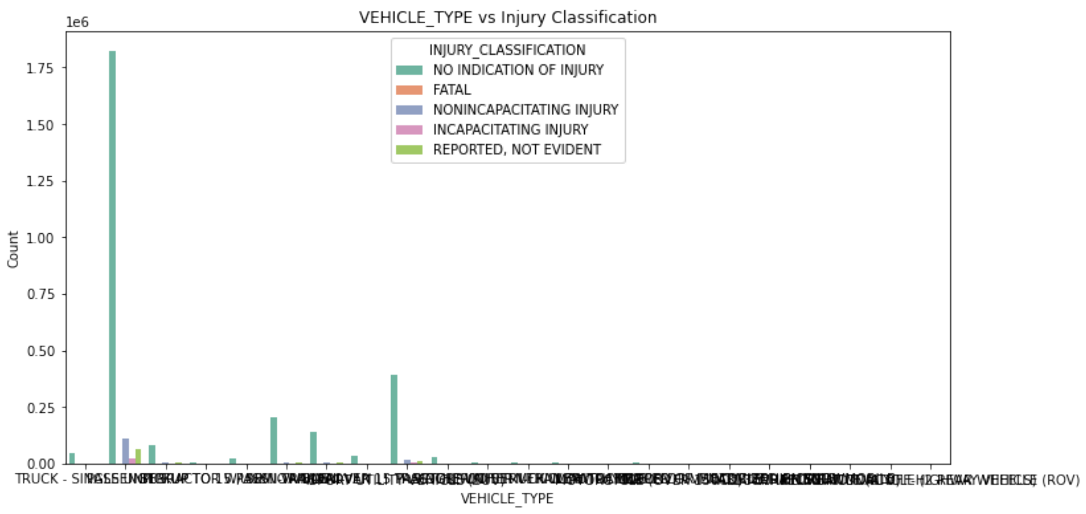
Interpretation - The distribution suggests that most accidents involve a small number of units (likely vehicles). Higher numbers of units involved don't necessarily correlate with more severe injuries, as they also appear in non-injury cases. This might indicate that the number of vehicles involved isn't a straightforward predictor of injury severity but could be a contributing factor when combined with other variables.

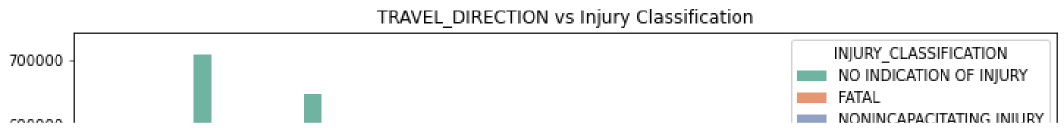
POSTED_SPEED_LIMIT Distribution by Injury Classification Observation - The POSTED_SPEED_LIMIT values show a wide range, with most data points clustering around typical urban speed limits (20-40 mph). Outliers exist at both ends of the spectrum, particularly in the "NO INDICATION OF INJURY" category.

Interpretation - The speed limit at the site of the crash varies widely but is most often in the 20-40 mph range, which is common in urban settings. The presence of outliers suggests that very high or very low-speed limits are less common but do exist. This distribution might indicate that the posted speed limit alone is not a strong predictor of injury severity, but it could be an important factor when considered alongside other variables like road conditions or vehicle type.

Explore Categorical Features and Their Relationship with the Target Variable

In [63]:
```python
# List of categorical features to explore
categorical_features = ['VEHICLE_TYPE', 'TRAVEL_DIRECTION', 'MANEUVER', 'SE

# Base path where images will be saved
save_base_path = r'C:\Users\MNJOROGE16\Desktop\Moringa\phase_3\project__pha

# Plot bar plots for categorical features and save them
for feature in categorical_features:
    plt.figure(figsize=(12, 6))
    sns.countplot(x=feature, hue='INJURY_CLASSIFICATION', data=merged_df_fi
    plt.title(f'{feature} vs Injury Classification')
    plt.xlabel(feature)
    plt.ylabel('Count')

    # Define the save path for each plot
    save_path = f'{save_base_path}\\barplot_{feature}_vs_injury_classificat

    # Save the figure as a PNG file
    plt.savefig(save_path)

    # Show the plot
    plt.show()
```





Barplots Interpretation

MANEUVER vs Injury Classification

Observation - The majority of maneuvers are concentrated around the "STRAIGHT AHEAD" maneuver, with a very high count in the "NO INDICATION OF INJURY" category. Other maneuvers have significantly lower counts. Interpretation - This suggests that most accidents occur while vehicles are moving straight ahead, and these incidents are often non-injurious. This could indicate that straight driving is common, but when incidents occur, they tend to be less severe. However, certain maneuvers with lower frequencies might be associated with more severe injuries.

SAFETY_EQUIPMENT vs Injury Classification

Observation - The vast majority of individuals were using "SAFETY BELT" equipment, predominantly resulting in "NO INDICATION OF INJURY". Other categories such as "NONE" have much lower frequencies but show higher incidences of injuries.

Interpretation - The data indicates that the use of safety equipment, especially safety belts, is strongly associated with a lower risk of injury. This supports the effectiveness of safety belts in reducing injury severity in traffic accidents.

SEX vs Injury Classification

Observation - There is a higher number of male ("M") participants in the data, with the majority having "NO INDICATION OF INJURY". Females ("F") also show a similar pattern but with fewer occurrences.

Interpretation - This suggests that more males are involved in accidents than females, but the injury distribution between sexes seems relatively similar, with the majority of both males and females not sustaining injuries in these incidents.

TRAVEL_DIRECTION vs Injury Classification

Observation - The most common travel directions are "N", "S", "E", and "W", with "NO INDICATION OF INJURY" being the most frequent classification for all directions. The "UNKNOWN" direction also appears but has fewer entries and a notable number of injuries.

Interpretation - The direction of travel appears to have little impact on injury severity, as most incidents across all directions result in no injuries. However, the "UNKNOWN" direction might be associated with less typical or more severe circumstances leading to injuries.

VEHICLE_TYPE vs Injury Classification Observation - The "PASSENGER" vehicle type dominates the data, with a very high count under "NO INDICATION OF INJURY". Other vehicle types, like "TRUCK" or "SUV", have significantly lower frequencies.

Interpretation - Most accidents involve passenger vehicles, which are more common on the roads, and the majority of these incidents do not result in injury. However, the data may

Data Preparation

Feature Selection

Review Data Understanding Insights

Objective - Reassess the key insights from the data understanding phase

In [64]:
```
# List of initial features after data understanding
initial_features = ['AGE', 'POSTED_SPEED_LIMIT', 'NUM_UNITS', 'LATITUDE', '
                    'TRAVEL_DIRECTION', 'MANEUVER', 'SEX', 'SAFETY_EQUIPMEN

# Target variable
target = 'INJURY_CLASSIFICATION'

# Review correlation matrix and scatter plots (already done in data underst
# No additional code needed here; just use insights from the previous steps
```

Remove Irrelevant Features

Objective - Drop features that are irrelevant or have little predictive power.

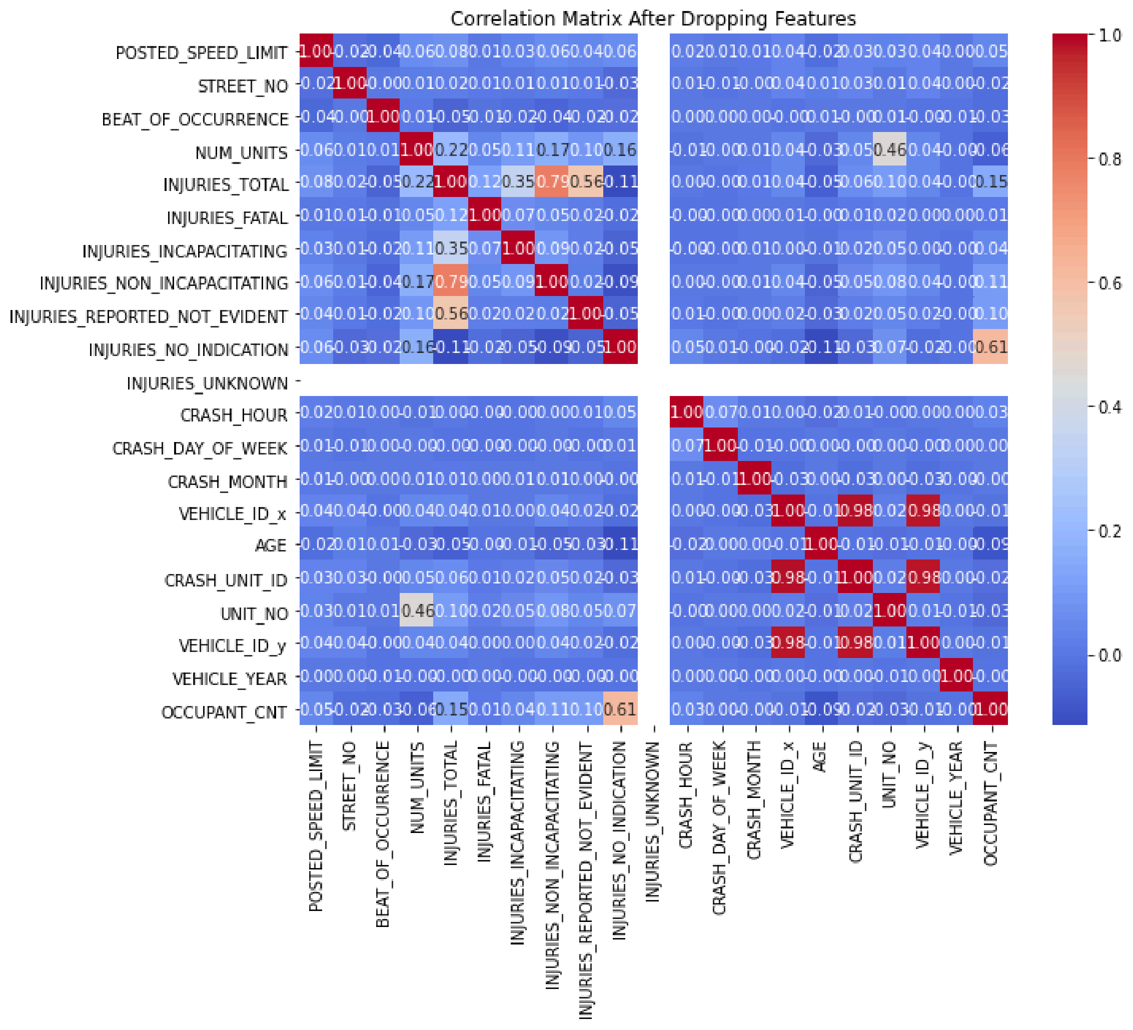
In [65]:
```
# Dropping irrelevant or redundant features identified earlier
features_to_drop = ['LATITUDE', 'LONGITUDE']
df_reduced = merged_df_final.drop(columns=features_to_drop)
print("Remaining features after dropping irrelevant ones:", df_reduced.colu
```

```
Remaining features after dropping irrelevant ones: Index(['CRASH_RECORD_I
D', 'CRASH_DATE_x', 'POSTED_SPEED_LIMIT',
       'TRAFFIC_CONTROL_DEVICE', 'DEVICE_CONDITION', 'WEATHER_CONDITION',
       'LIGHTING_CONDITION', 'FIRST_CRASH_TYPE', 'TRAFFICWAY_TYPE',
       'ALIGNMENT', 'ROADWAY_SURFACE_COND', 'ROAD_DEFECT', 'REPORT_TYPE',
       'CRASH_TYPE', 'DAMAGE', 'DATE_POLICE_NOTIFIED',
       'PRIM_CONTRIBUTORY_CAUSE', 'SEC_CONTRIBUTORY_CAUSE', 'STREET_NO',
       'STREET_DIRECTION', 'STREET_NAME', 'BEAT_OF_OCCURRENCE', 'NUM_UNIT
S',
       'MOST_SEVERE_INJURY', 'INJURIES_TOTAL', 'INJURIES_FATAL',
       'INJURIES_INCAPACITATING', 'INJURIES_NON_INCAPACITATING',
       'INJURIES_REPORTED_NOT_EVIDENT', 'INJURIES_NO_INDICATION',
       'INJURIES_UNKNOWN', 'CRASH_HOUR', 'CRASH_DAY_OF_WEEK', 'CRASH_MONT
H',
       'LOCATION', 'PERSON_ID', 'PERSON_TYPE', 'VEHICLE_ID_x', 'CRASH_DAT
E_y',
       'SEX', 'AGE', 'DRIVERS_LICENSE_STATE', 'DRIVERS_LICENSE_CLASS',
       'SAFETY_EQUIPMENT', 'AIRBAG_DEPLOYED', 'EJECTION',
       'INJURY_CLASSIFICATION', 'DRIVER_ACTION', 'DRIVER_VISION',
       'PHYSICAL_CONDITION', 'BAC_RESULT', 'CRASH_UNIT_ID', 'CRASH_DATE',
       'UNIT_NO', 'UNIT_TYPE', 'VEHICLE_ID_y', 'MAKE', 'MODEL',
       'LIC_PLATE_STATE', 'VEHICLE_YEAR', 'VEHICLE_DEFECT', 'VEHICLE_TYP
E',
       'VEHICLE_USE', 'TRAVEL_DIRECTION', 'MANEUVER', 'OCCUPANT_CNT',
       'FIRST_CONTACT_POINT'],
      dtype='object')
```

Address Multicollinearity

Objective - Handling multicollinearity, important for logistic regression.

In [66]:
```
# Calculate the correlation matrix
corr_matrix = df_reduced.corr()

# Plot the heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation Matrix After Dropping Features')

# Save the heatmap to the specified folder
output_path = r'C:\Users\MNJOROGE16\Desktop\Moringa\phase_3\project__phase3
plt.savefig(output_path)

# Display the plot
plt.show()
```



Interpretation of the Correlation Matrix Heatmap

Low Correlation Between Most Features

The majority of the features have low correlation values (close to 0), indicating that they do not have strong linear relationships with each other. This is beneficial for both logistic regression and decision trees because.

Logistic Regression Low correlation reduces the risk of multicollinearity, which can lead to unreliable estimates of coefficients.

Decision Trees The algorithm is robust to multicollinearity, but having uncorrelated features ensures that each feature contributes unique information to the model.

Highly Correlated Features

Injuries Features The features related to different types of injuries (e.g., INJURIES_TOTAL, INJURIES_FATAL, INCAPACITATING, etc.) show strong correlations with each other.

Action - For logistic regression, you might consider dropping or combining these highly correlated features to avoid multicollinearity. For decision trees, you might retain them, as the model can handle correlated features well.

Vehicle IDs Features like VEHICLE_ID_x and VEHICLE_ID_y are also highly correlated.

Action - These are likely identifiers or categorical variables that may not be necessary for modeling. Consider dropping them unless they provide meaningful insights.

Conclusion

Feature Selection The heatmap suggests focusing on features that are less correlated with each other, which can help in building more stable models.

Injury Features Given that many injury-related features are highly correlated, you might choose the most representative one(s) or create composite scores (e.g., summing or averaging certain features) for logistic regression.

Geographical Features LATITUDE and LONGITUDE were dropped due to low relevance or high correlation with each other. This decision appears justified given the project

Type *Markdown* and LaTeX: α^2

In [67]:
```
#Review the correlation matrix
corr_matrix = df_reduced.corr().abs()  # Use absolute values to consider po

#Identify features with high correlation
# Setting a threshold of 0.8 as an example for high correlation
high_corr_pairs = np.where(corr_matrix > 0.8)
high_corr_pairs = [(corr_matrix.index[x], corr_matrix.columns[y])
                   for x, y in zip(*high_corr_pairs)
                   if x != y and x < y]

print("Highly correlated feature pairs:")
for pair in high_corr_pairs:
    print(pair)

# Drop redundant features based on correlation and domain knowledge
# Dropping VEHICLE_ID_x and VEHICLE_ID_y as they are identifiers and not us
# Drop additional features identified from correlation matrix review
features_to_drop = [
    'NUM_UNITS', 'INJURIES_TOTAL', 'INJURIES_FATAL', 'INJURIES_INCAPACITATI
    'INJURIES_NON_INCAPACITATING', 'INJURIES_REPORTED_NOT_EVIDENT', 'INJURI
    'INJURIES_UNKNOWN', 'OCCUPANT_CNT']

df_final = df_reduced.drop(columns=features_to_drop)
print("Final feature set after removing redundant features:")
print(df_final.columns)

# Save the final refined dataset
df_final.to_csv(r'C:\Users\MNJOROGE16\Desktop\Moringa\phase_3\project__phas
```

```
Highly correlated feature pairs:
('VEHICLE_ID_x', 'CRASH_UNIT_ID')
('VEHICLE_ID_x', 'VEHICLE_ID_y')
('CRASH_UNIT_ID', 'VEHICLE_ID_y')
Final feature set after removing redundant features:
Index(['CRASH_RECORD_ID', 'CRASH_DATE_x', 'POSTED_SPEED_LIMIT',
       'TRAFFIC_CONTROL_DEVICE', 'DEVICE_CONDITION', 'WEATHER_CONDITION',
       'LIGHTING_CONDITION', 'FIRST_CRASH_TYPE', 'TRAFFICWAY_TYPE',
       'ALIGNMENT', 'ROADWAY_SURFACE_COND', 'ROAD_DEFECT', 'REPORT_TYPE',
       'CRASH_TYPE', 'DAMAGE', 'DATE_POLICE_NOTIFIED',
       'PRIM_CONTRIBUTORY_CAUSE', 'SEC_CONTRIBUTORY_CAUSE', 'STREET_NO',
       'STREET_DIRECTION', 'STREET_NAME', 'BEAT_OF_OCCURRENCE',
       'MOST_SEVERE_INJURY', 'CRASH_HOUR', 'CRASH_DAY_OF_WEEK', 'CRASH_MO
NTH',
       'LOCATION', 'PERSON_ID', 'PERSON_TYPE', 'VEHICLE_ID_x', 'CRASH_DAT
E_y',
       'SEX', 'AGE', 'DRIVERS_LICENSE_STATE', 'DRIVERS_LICENSE_CLASS',
       'SAFETY_EQUIPMENT', 'AIRBAG_DEPLOYED', 'EJECTION',
       'INJURY_CLASSIFICATION', 'DRIVER_ACTION', 'DRIVER_VISION',
       'PHYSICAL_CONDITION', 'BAC_RESULT', 'CRASH_UNIT_ID', 'CRASH_DATE',
       'UNIT_NO', 'UNIT_TYPE', 'VEHICLE_ID_y', 'MAKE', 'MODEL',
       'LIC_PLATE_STATE', 'VEHICLE_YEAR', 'VEHICLE_DEFECT', 'VEHICLE_TYP
E',
       'VEHICLE_USE', 'TRAVEL_DIRECTION', 'MANEUVER', 'FIRST_CONTACT_POIN
T'],
      dtype='object')
```

Handle Remaining Missing Values

Missing values were handled during data understanding

Feature Engineering

Create New Features

New features These features help the model understand temporal patterns, such as whether certain times of the day or week are associated with higher injury severity.

Interaction features To create meaningful interactions that can enhance the predictive power of the models

Binning Features Binning helps to simplify the relationship between age and injury severity by grouping ages into broader categories.

In [68]:
```
# Create time-based features
df_final['CRASH_HOUR'] = pd.to_datetime(df_final['CRASH_DATE_x']).dt.hour
df_final['CRASH_DAY_OF_WEEK'] = pd.to_datetime(df_final['CRASH_DATE_x']).dt
df_final['CRASH_MONTH'] = pd.to_datetime(df_final['CRASH_DATE_x']).dt.month

print("Created time-based features: CRASH_HOUR, CRASH_DAY_OF_WEEK, CRASH_MO
```

```
Created time-based features: CRASH_HOUR, CRASH_DAY_OF_WEEK, CRASH_MONTH
```

In [69]:
```
# Create an interaction feature between POSTED_SPEED_LIMIT and VEHICLE_TYPE

#The type of vehicle involved in a crash combined with the speed limit can
# For example, crashes involving motorcycles at high speeds might result in
df_final['SPEED_VEHICLE_TYPE'] = df_final['POSTED_SPEED_LIMIT'] * df_final[

print("Created interaction feature: SPEED_VEHICLE_TYPE")
```

```
Created interaction feature: SPEED_VEHICLE_TYPE
```

In [70]:
```
# Create an interaction feature between POSTED_SPEED_LIMIT and VEHICLE_TYPE

#The combination of weather and lighting conditions can significantly impac

df_final['SPEED_VEHICLE_TYPE'] = df_final['POSTED_SPEED_LIMIT'] * df_final[

print("Created interaction feature: SPEED_VEHICLE_TYPE")
```

```
Created interaction feature: SPEED_VEHICLE_TYPE
```

In [71]:
```
# Binning AGE into categories
bins = [0, 18, 30, 50, 70, 100]
labels = ['Youth', 'Young Adult', 'Adult', 'Senior', 'Elder']
df_final['AGE_BINNED'] = pd.cut(df_final['AGE'], bins=bins, labels=labels,

print("Binned AGE into categories: Youth, Young Adult, Adult, Senior, Elder
```

```
Binned AGE into categories: Youth, Young Adult, Adult, Senior, Elder
```

####

Feature Encoding

Identify Categorical Features

TRAFFIC_CONTROL_DEVICE

DEVICE_CONDITION

WEATHER_CONDITION

LIGHTING_CONDITION

FIRST_CRASH_TYPE

TRAFFICWAY_TYPE

ROADWAY_SURFACE_COND

ROAD_DEFECT

REPORT_TYPE

CRASH_TYPE

MOST_SEVERE_INJURY

PERSON_TYPE

SEX

DRIVERS_LICENSE_STATE

DRIVERS_LICENSE_CLASS

SAFETY_EQUIPMENT

AIRBAG_DEPLOYED

EJECTION

INJURY_CLASSIFICATION

DRIVER_ACTION

DRIVER_VISION

PHYSICAL_CONDITION

VEHICLE_TYPE

VEHICLE_USE

TRAVEL_DIRECTION

MANEUVER

One-Hot Encoding

One-Hot Encoding step is crucial because it ensures that all categorical data is in a format that can be utilized by logistic regression and decision tree models, ultimately contributing to more accurate predictions of injury severity.

In [72]:
```

# List of categorical features to encode
categorical_columns = [
    'TRAFFIC_CONTROL_DEVICE', 'DEVICE_CONDITION', 'WEATHER_CONDITION',
    'LIGHTING_CONDITION', 'FIRST_CRASH_TYPE', 'TRAFFICWAY_TYPE',
    'ROADWAY_SURFACE_COND', 'ROAD_DEFECT', 'REPORT_TYPE',
    'CRASH_TYPE', 'MOST_SEVERE_INJURY', 'PERSON_TYPE',
    'SEX', 'DRIVERS_LICENSE_STATE', 'DRIVERS_LICENSE_CLASS',
    'SAFETY_EQUIPMENT', 'AIRBAG_DEPLOYED', 'EJECTION',
    'INJURY_CLASSIFICATION', 'DRIVER_ACTION', 'DRIVER_VISION',
    'PHYSICAL_CONDITION', 'VEHICLE_TYPE', 'VEHICLE_USE',
    'TRAVEL_DIRECTION', 'MANEUVER'
]

# Apply one-hot encoding to categorical columns
df_encoded = pd.get_dummies(df_final, columns=categorical_columns, drop_fir
```

In [73]:
```
# Display the shape and columns of the resulting DataFrame
print("Data shape after one-hot encoding:", df_encoded.shape)
```

```
Data shape after one-hot encoding: (3076376, 749)
```

```
In [76]: print("Encoded columns:")
         print(df_encoded.columns)
```

```
Encoded columns:
Index(['CRASH_RECORD_ID', 'CRASH_DATE_x', 'POSTED_SPEED_LIMIT', 'ALIGNMEN
T',
       'DAMAGE', 'DATE_POLICE_NOTIFIED', 'PRIM_CONTRIBUTORY_CAUSE',
       'SEC_CONTRIBUTORY_CAUSE', 'STREET_NO', 'STREET_DIRECTION',
       ...
       'MANEUVER_SLOW/STOP - LOAD/UNLOAD', 'MANEUVER_SLOW/STOP - RIGHT TU
RN',
       'MANEUVER_SLOW/STOP IN TRAFFIC', 'MANEUVER_STARTING IN TRAFFIC',
       'MANEUVER_STRAIGHT AHEAD', 'MANEUVER_TURNING LEFT',
       'MANEUVER_TURNING ON RED', 'MANEUVER_TURNING RIGHT', 'MANEUVER_U-T
URN',
       'MANEUVER_UNKNOWN/NA'],
      dtype='object', length=749)
```

Feature Scaling

```
In [78]: # Check the columns available in df_encoded
         print("Available columns in df_encoded:")
         print(df_encoded.columns)
```

```
Available columns in df_encoded:
Index(['CRASH_RECORD_ID', 'CRASH_DATE_x', 'POSTED_SPEED_LIMIT', 'ALIGNMEN
T',
       'DAMAGE', 'DATE_POLICE_NOTIFIED', 'PRIM_CONTRIBUTORY_CAUSE',
       'SEC_CONTRIBUTORY_CAUSE', 'STREET_NO', 'STREET_DIRECTION',
       ...
       'MANEUVER_SLOW/STOP - LOAD/UNLOAD', 'MANEUVER_SLOW/STOP - RIGHT TU
RN',
       'MANEUVER_SLOW/STOP IN TRAFFIC', 'MANEUVER_STARTING IN TRAFFIC',
       'MANEUVER_STRAIGHT AHEAD', 'MANEUVER_TURNING LEFT',
       'MANEUVER_TURNING ON RED', 'MANEUVER_TURNING RIGHT', 'MANEUVER_U-T
URN',
       'MANEUVER_UNKNOWN/NA'],
      dtype='object', length=749)
```

```
In [79]: # List of numerical features to scale (adjusted to match the available colu
         numerical_columns = ['POSTED_SPEED_LIMIT', 'AGE']
```

```
In [81]: # Verify the availability of numerical columns before scaling
available_numerical_columns = [col for col in numerical_columns if col in c

if available_numerical_columns:
    # Apply scaling to available numerical features
    df_encoded[available_numerical_columns] = scaler.fit_transform(df_encod

    print("Numerical features after scaling:")
    print(df_encoded[available_numerical_columns].head())
else:
    print("No numerical columns available for scaling.")
```

```
Numerical features after scaling:
   POSTED_SPEED_LIMIT       AGE
0           -0.678505 -0.754361
1           -0.678505 -0.754361
2           -0.678505  2.283221
3           -0.678505  2.283221
4           -0.678505 -0.133038
```

Split Data into Training and Testing Sets

Define the Feature Matrix (X) and Target Variable (y)

Objective - Separate the dataset into the input features (X) and the target variable (y).

Drop the target variable from the dataset to create X and set y to the target column (INJURY_CLASSIFICATION).

Split the Data

Objective - Divide the dataset into training and testing sets to evaluate the performance of the models

Use train_test_split to create the training and testing datasets.

In [85]:
```
# Identify the one-hot encoded columns related to INJURY_CLASSIFICATION
target_columns = [
    'INJURY_CLASSIFICATION_INCAPACITATING INJURY',
    'INJURY_CLASSIFICATION_NO INDICATION OF INJURY',
    'INJURY_CLASSIFICATION_NONINCAPACITATING INJURY',
    'INJURY_CLASSIFICATION_REPORTED, NOT EVIDENT'
]

# Combine the one-hot encoded columns back into a single categorical column
df_encoded['INJURY_CLASSIFICATION'] = df_encoded[target_columns].idxmax(axi

# Drop the one-hot encoded columns since they are merged back
df_encoded = df_encoded.drop(columns=target_columns)

# Define the feature matrix (X) and target variable (y)
y = df_encoded['INJURY_CLASSIFICATION']
X = df_encoded.drop(columns=['INJURY_CLASSIFICATION'])  # Dropping the targ
```

In [86]:
```
from sklearn.model_selection import train_test_split

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, ra

print("Training data shape (X_train):", X_train.shape)
print("Training labels shape (y_train):", y_train.shape)
print("Testing data shape (X_test):", X_test.shape)
print("Testing labels shape (y_test):", y_test.shape)
```

```
Training data shape (X_train): (2153463, 745)
Training labels shape (y_train): (2153463,)
Testing data shape (X_test): (922913, 745)
Testing labels shape (y_test): (922913,)
```

Modelling

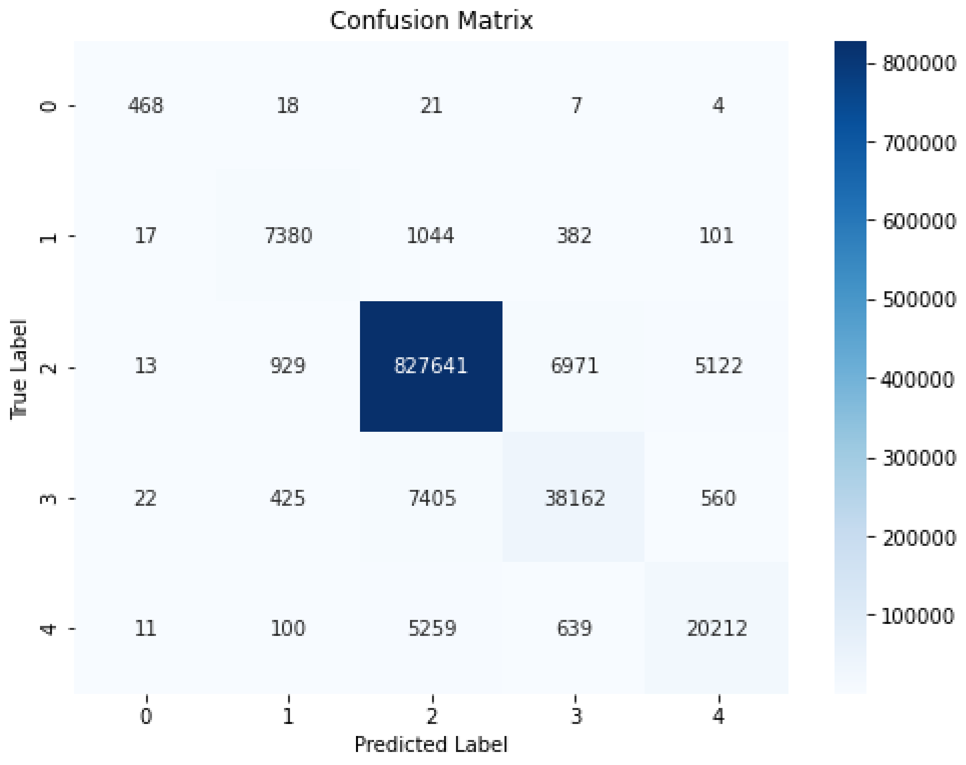
Decision Tree

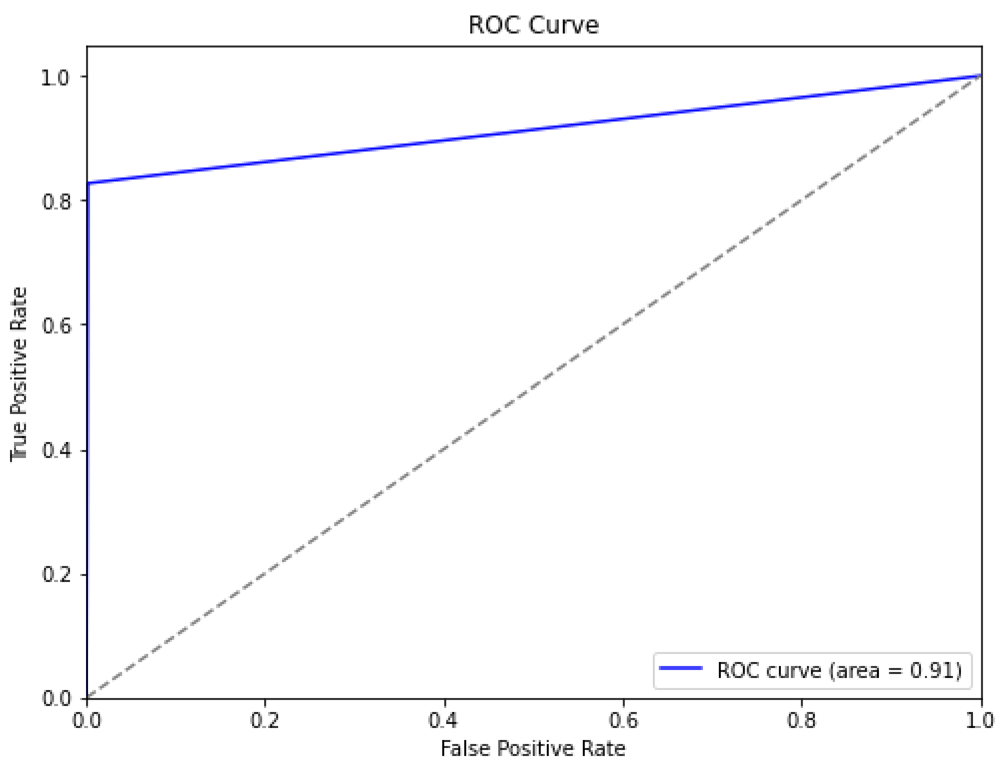
In [11]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, classification_report
from sklearn.preprocessing import LabelEncoder

# Identify high-cardinality categorical columns
categorical_columns = X.select_dtypes(include=['object']).columns
high_cardinality_columns = [col for col in categorical_columns if X[col].nu

# Apply label encoding to high-cardinality columns
label_encoders = {}
for col in high_cardinality_columns:
    le = LabelEncoder()
    X[col] = le.fit_transform(X[col].astype(str))
    label_encoders[col] = le

# Apply one-hot encoding to the remaining categorical columns with low card
low_cardinality_columns = [col for col in categorical_columns if col not in
X_encoded = pd.get_dummies(X, columns=low_cardinality_columns)

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_encoded, y, test_size

# Train a baseline Decision Tree model
dt_baseline = DecisionTreeClassifier(random_state=42)
dt_baseline.fit(X_train, y_train)

# Predict on the test set
y_pred = dt_baseline.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
report = classification_report(y_test, y_pred)

print("Baseline Decision Tree Model Accuracy:", accuracy)
print("\nClassification Report:\n", report)
```

```
Baseline Decision Tree Model Accuracy: 0.9685235769785451

Classification Report:
                           precision    recall  f1-score   support

                   FATAL       0.88      0.90      0.89       518
    INCAPACITATING INJURY       0.83      0.83      0.83      8924
  NO INDICATION OF INJURY       0.98      0.98      0.98    840676
NONINCAPACITATING INJURY       0.83      0.82      0.82     46574
   REPORTED, NOT EVIDENT       0.78      0.77      0.77     26221

                accuracy                           0.97    922913
               macro avg       0.86      0.86      0.86    922913
            weighted avg       0.97      0.97      0.97    922913
```

In [17]:
```
#Evaluate Baseline Model
import os
from sklearn.metrics import classification_report, confusion_matrix, accura
import matplotlib.pyplot as plt
import seaborn as sns

# Define the path where visualizations will be saved
visuals_path = r'C:\Users\MNJOROGE16\Desktop\Moringa\phase_3\project__phase

# Ensure the directory exists
os.makedirs(visuals_path, exist_ok=True)

# Evaluate the baseline model
accuracy = accuracy_score(y_test, y_pred)
report = classification_report(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)

print("Baseline Decision Tree Model Accuracy:", accuracy)
print("\nClassification Report:\n", report)

# Plotting Confusion Matrix
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt="d", cmap="Blues")
plt.title("Confusion Matrix")
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.savefig(os.path.join(visuals_path, "baseline_confusion_matrix.png"))
plt.show()

# ROC and AUC
y_pred_proba = dt_baseline.predict_proba(X_test)
fpr, tpr, _ = roc_curve(y_test, y_pred_proba[:, 1], pos_label=dt_baseline.c
roc_auc = auc(fpr, tpr)

plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, color="blue", label=f"ROC curve (area = {roc_auc:.2f})")
plt.plot([0, 1], [0, 1], color="gray", linestyle="--")
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ROC Curve")
plt.legend(loc="lower right")
plt.savefig(os.path.join(visuals_path, "baseline_roc_curve.png"))
plt.show()
```

```
Baseline Decision Tree Model Accuracy: 0.9685235769785451

Classification Report:
                           precision    recall  f1-score   support

                   FATAL       0.88      0.90      0.89       518
   INCAPACITATING INJURY       0.83      0.83      0.83      8924
  NO INDICATION OF INJURY      0.98      0.98      0.98    840676
NONINCAPACITATING INJURY       0.83      0.82      0.82     46574
   REPORTED, NOT EVIDENT       0.78      0.77      0.77     26221

                accuracy                           0.97    922913
               macro avg       0.86      0.86      0.86    922913
            weighted avg       0.97      0.97      0.97    922913
```





In [23]:
```
from sklearn.model_selection import train_test_split

# Reduce the dataset size using stratified sampling
X_small, _, y_small, _ = train_test_split(X, y, test_size=0.7, stratify=y,

# Print the size of the reduced dataset
print("Reduced dataset size:", X_small.shape, y_small.shape)
```

```
Reduced dataset size: (922912, 57) (922912,)
```