1 PROJECT OVERVIEW

The complex ecosystem of airports, supporting organisations, and aircraft used for general aviation, commercial aviation, and military activities is referred to as the aviation industry. General aviation incorporates private air travel wheteher for business or pleasure. Commercial aviation focusses on public transportation of passengers and cargo. On the other hand, millitary aviation encompasses all flights operated by the armed forces.

4.6% of Kenya's yearly GDP is contributed by the aviation sector (KNBS, 2022). In 2017, Kenya handled over 4.6 million passenger travels, resulting in a $3.2 billion gross value addition to the country's GDP from the tourism and aviation sectors. This also translated into 410,000 jobs.

Internationally, the commercial aviation sector experienced 30 accidents in 2023 as compared to 42 accidents in 2022.Therefore, between 2022 and 2023, the overall accident rate dropped from 1.30 per million sectors to 0.80. Overall, one accident occurred per every 880,293 flights on average (IATA Annual Safety Report, 2023). According to statistics, human error is to blame for up to 80% of all aviation accidents. Takeoff and landing, as well as the moments just before and after, are the riskiest times.

According to a 2022 security audit conducted by the International Civil Aviation Organisation (ICAO), Kenya has the second-best aviation safety standards in Africa, scoring 91.77%.

2 BUSINESS UNDERSTANDING

Maureen Inc. is diversifying its holdings and considers entry into the aviation industry. Its specific interest is in the genearal and commericial services. The company plans to procure aircrafts for the new venture. To help in decision making, the company would like insihgts on the aircrafts that would have the lowest risk for the aviation division.

#DATA UNDERSTANDING

2.1 Problem Statement

Identify the least risky planes (aircrafts) for Maureen Inc. to launch its new venture

2.2 Metrics of Success

The project will be a success if i am able to identify low risk aircrafts and the factors affecting aircraft safety.

2.3 Methodology

The analysis adopts the first three phases of CRISP - DM that is Domain knowledge (Business Understanding), Data Understanding and Data Preparation.

2.4 Data Description

Importing Libraries

In [52]:
```
#importing libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

Reading the dataset from csv files

In [53]:
```
#Reading data from the CSV file

df = pd.read_csv('Data/AviationData.csv')
df1 = pd.read_csv('Data/USState_Codes.csv')
```

```
/opt/anaconda3/envs/learn-env/lib/python3.8/site-packages/IPython/core/
interactiveshell.py:3145: DtypeWarning: Columns (6,7,28) have mixed typ
es.Specify dtype option on import or set low_memory=False.
  has_raised = await self.run_ast_nodes(code_ast.body, cell_name,
```

Previewing the Aviation Data Set

In [54]:
```
#first five rows
df.head()
```

Out[54]:

	Event.Id	Investigation.Type	Accident.Number	Event.Date	Location	Country
0	20001218X45444	Accident	SEA87LA080	1948-10-24	MOOSE CREEK, ID	Unit Stat
1	20001218X45447	Accident	LAX94LA336	1962-07-19	BRIDGEPORT, CA	Unit Stat
2	20061025X01555	Accident	NYC07LA005	1974-08-30	Saltville, VA	Unit Stat
3	20001218X45448	Accident	LAX96LA321	1977-06-19	EUREKA, CA	Unit Stat
4	20041105X01764	Accident	CHI79FA064	1979-08-02	Canton, OH	Unit Stat

5 rows × 31 columns

In [55]:
```
#Last five rows
df.tail()
```

Out[55]:

	Event.Id	Investigation.Type	Accident.Number	Event.Date	Location	Country
88884	20221227106491	Accident	ERA23LA093	2022-12-26	Annapolis, MD	Un Sta
88885	20221227106494	Accident	ERA23LA095	2022-12-26	Hampton, NH	Un Sta
88886	20221227106497	Accident	WPR23LA075	2022-12-26	Payson, AZ	Un Sta
88887	20221227106498	Accident	WPR23LA076	2022-12-26	Morgan, UT	Un Sta
88888	20221230106513	Accident	ERA23LA097	2022-12-29	Athens, GA	Un Sta

5 rows × 31 columns

Accessing information about the data set

In [56]:
```
#Information about the dataset
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 88889 entries, 0 to 88888
Data columns (total 31 columns):
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   Event.Id                88889 non-null  object 
 1   Investigation.Type      88889 non-null  object 
 2   Accident.Number         88889 non-null  object 
 3   Event.Date              88889 non-null  object 
 4   Location                88837 non-null  object 
 5   Country                 88663 non-null  object 
 6   Latitude                34382 non-null  object 
 7   Longitude               34373 non-null  object 
 8   Airport.Code            50249 non-null  object 
 9   Airport.Name            52790 non-null  object 
 10  Injury.Severity         87889 non-null  object 
 11  Aircraft.damage         85695 non-null  object 
 12  Aircraft.Category       32287 non-null  object 
 13  Registration.Number     87572 non-null  object 
 14  Make                    88826 non-null  object 
 15  Model                   88797 non-null  object 
 16  Amateur.Built           88787 non-null  object 
 17  Number.of.Engines       82805 non-null  float64
 18  Engine.Type             81812 non-null  object 
 19  FAR.Description         32023 non-null  object 
 20  Schedule                12582 non-null  object 
 21  Purpose.of.flight       82697 non-null  object 
 22  Air.carrier             16648 non-null  object 
 23  Total.Fatal.Injuries    77488 non-null  float64
 24  Total.Serious.Injuries  76379 non-null  float64
 25  Total.Minor.Injuries    76956 non-null  float64
 26  Total.Uninjured         82977 non-null  float64
 27  Weather.Condition       84397 non-null  object 
 28  Broad.phase.of.flight   61724 non-null  object 
 29  Report.Status           82508 non-null  object 
 30  Publication.Date        75118 non-null  object 
dtypes: float64(5), object(26)
memory usage: 21.0+ MB
```

The dataset has 88889 entries with 31 columns. A number of the columns had missing values e.g. Airport Code, Aircraft Category, Engine Type, Aircraft Dmage etc

The dataset is made up of two main data types, 5 columns of integer (float) type and 25 columns of type string (object)

In [57]:
```
df.shape
```

Out[57]:
```
(88889, 31)
```

The data set ia made up of 88889 rows and 31 columns

In [58]:
```
#names of the data columns
df.columns
```

Out[58]:
```
Index(['Event.Id', 'Investigation.Type', 'Accident.Number', 'Event.Dat
e',
       'Location', 'Country', 'Latitude', 'Longitude', 'Airport.Code',
       'Airport.Name', 'Injury.Severity', 'Aircraft.damage',
       'Aircraft.Category', 'Registration.Number', 'Make', 'Model',
       'Amateur.Built', 'Number.of.Engines', 'Engine.Type', 'FAR.Descri
ption',
       'Schedule', 'Purpose.of.flight', 'Air.carrier', 'Total.Fatal.Inj
uries',
       'Total.Serious.Injuries', 'Total.Minor.Injuries', 'Total.Uninjur
ed',
       'Weather.Condition', 'Broad.phase.of.flight', 'Report.Status',
       'Publication.Date'],
      dtype='object')
```

Summary Statistics

In [59]:
```
df.describe()
```

Out[59]:

	Number.of.Engines	Total.Fatal.Injuries	Total.Serious.Injuries	Total.Minor.Injuries	Tota
count	82805.000000	77488.000000	76379.000000	76956.000000	
mean	1.146585	0.647855	0.279881	0.357061	
std	0.446510	5.485960	1.544084	2.235625	
min	0.000000	0.000000	0.000000	0.000000	
25%	1.000000	0.000000	0.000000	0.000000	
50%	1.000000	0.000000	0.000000	0.000000	
75%	1.000000	0.000000	0.000000	0.000000	
max	8.000000	349.000000	161.000000	380.000000	

A total of 77,488 fatalities were reported from aircraft accident between 1962 and 2023. Serious injuries reported from the accidents during the same period were 76,379 compared to 76,956 minor injuries. 82,977 were uninjured from the incidents. The aircrafts under analysis had an average of one engine

Creating a deep copy of the dataframe df.

In [60]:
```
#Deep copy copies the original data frame and assigns it to a new datafr
#The deep copy is a true replica of the original dataset and independent
# Any changes make on the original dataset will not affect the copy and

df2 = df.copy(deep=True)
```

3 DATA PREPARATION

3.1 Data Cleaning

3.1.1 Validity Test/Challenges

a) Dropping irrelevant observations

In [61]:
```
df2.drop(['Publication.Date', 'Schedule', 'Latitude', 'Longitude', 'FAR.
df2.head()
```

Out[61]:

	Event.Id	Investigation.Type	Accident.Number	Event.Date	Location	Country
0	20001218X45444	Accident	SEA87LA080	1948-10-24	MOOSE CREEK, ID	Unit Stat
1	20001218X45447	Accident	LAX94LA336	1962-07-19	BRIDGEPORT, CA	Unit Stat
2	20061025X01555	Accident	NYC07LA005	1974-08-30	Saltville, VA	Unit Stat
3	20001218X45448	Accident	LAX96LA321	1977-06-19	EUREKA, CA	Unit Stat
4	20041105X01764	Accident	CHI79FA064	1979-08-02	Canton, OH	Unit Stat

5 rows × 22 columns

In [62]:
```
df2.columns
```

Out[62]:
```
Index(['Event.Id', 'Investigation.Type', 'Accident.Number', 'Event.Dat
e',
       'Location', 'Country', 'Injury.Severity', 'Aircraft.damage',
       'Registration.Number', 'Make', 'Model', 'Amateur.Built',
       'Number.of.Engines', 'Engine.Type', 'Purpose.of.flight',
       'Total.Fatal.Injuries', 'Total.Serious.Injuries',
       'Total.Minor.Injuries', 'Total.Uninjured', 'Weather.Condition',
       'Broad.phase.of.flight', 'Report.Status'],
      dtype='object')
```

Syntax Errors: b) Removing leading and trailing spaces in columns

In [63]:
```
df2.columns = df2.columns.str.strip()
df2.head()
```

Out[63]:

	Event.Id	Investigation.Type	Accident.Number	Event.Date	Location	Country
0	20001218X45444	Accident	SEA87LA080	1948-10-24	MOOSE CREEK, ID	Unit Stat
1	20001218X45447	Accident	LAX94LA336	1962-07-19	BRIDGEPORT, CA	Unit Stat
2	20061025X01555	Accident	NYC07LA005	1974-08-30	Saltville, VA	Unit Stat
3	20001218X45448	Accident	LAX96LA321	1977-06-19	EUREKA, CA	Unit Stat
4	20041105X01764	Accident	CHI79FA064	1979-08-02	Canton, OH	Unit Stat

5 rows × 22 columns

c) Removing . between column names and replacing with space

In [64]:
```
df2.columns = df2.columns.str.replace('.', ' ')
df2.columns
```

```
<ipython-input-64-3410bcadee4b>:1: FutureWarning: The default value of
regex will change from True to False in a future version. In addition,
single character regular expressions will*not* be treated as literal st
rings when regex=True.
  df2.columns = df2.columns.str.replace('.', ' ')
```

Out[64]:
```
Index(['Event Id', 'Investigation Type', 'Accident Number', 'Event Dat
e',
       'Location', 'Country', 'Injury Severity', 'Aircraft damage',
       'Registration Number', 'Make', 'Model', 'Amateur Built',
       'Number of Engines', 'Engine Type', 'Purpose of flight',
       'Total Fatal Injuries', 'Total Serious Injuries',
       'Total Minor Injuries', 'Total Uninjured', 'Weather Condition',
       'Broad phase of flight', 'Report Status'],
      dtype='object')
```

In [65]:
```
df2.dtypes
```

Out[65]:
```
Event Id                  object
Investigation Type        object
Accident Number           object
Event Date                object
Location                  object
Country                   object
Injury Severity           object
Aircraft damage           object
Registration Number       object
Make                      object
Model                     object
Amateur Built             object
Number of Engines        float64
Engine Type               object
Purpose of flight         object
Total Fatal Injuries     float64
Total Serious Injuries   float64
Total Minor Injuries     float64
Total Uninjured          float64
Weather Condition         object
Broad phase of flight     object
Report Status             object
dtype: object
```

d) Convert Event date from string to date data type

In [66]:
```
df2['Event Date'] = pd.to_datetime(df2['Event Date'])
df2.dtypes
```

Out[66]:
```
Event Id                          object
Investigation Type                object
Accident Number                   object
Event Date                datetime64[ns]
Location                          object
Country                           object
Injury Severity                   object
Aircraft damage                   object
Registration Number               object
Make                              object
Model                             object
Amateur Built                     object
Number of Engines                float64
Engine Type                       object
Purpose of flight                 object
Total Fatal Injuries             float64
Total Serious Injuries           float64
Total Minor Injuries             float64
Total Uninjured                  float64
Weather Condition                 object
Broad phase of flight             object
Report Status                     object
dtype: object
```

In [67]:
```
df2.head()
```

Out[67]:

	Event Id	Investigation Type	Accident Number	Event Date	Location	Country	Injury Severity	d
0	20001218X45444	Accident	SEA87LA080	1948-10-24	MOOSE CREEK, ID	United States	Fatal(2)	
1	20001218X45447	Accident	LAX94LA336	1962-07-19	BRIDGEPORT, CA	United States	Fatal(4)	
2	20061025X01555	Accident	NYC07LA005	1974-08-30	Saltville, VA	United States	Fatal(3)	
3	20001218X45448	Accident	LAX96LA321	1977-06-19	EUREKA, CA	United States	Fatal(2)	
4	20041105X01764	Accident	CHI79FA064	1979-08-02	Canton, OH	United States	Fatal(1)	

5 rows × 22 columns

In [68]:
```
df2['Number of Engines'].unique()
```

Out[68]:
```
array([ 1., nan,  2.,  0.,  3.,  4.,  8.,  6.])
```

In [69]:
```
#Adding a new column for total Injuiries
df2['Total Injuries'] = df2['Total Fatal Injuries'] + df2['Total Serious
df2['Total Injuries']
```

Out[69]:
```
0        2.0
1        4.0
2        NaN
3        2.0
4        NaN
        ...
88884    1.0
88885    0.0
88886    0.0
88887    0.0
88888    1.0
Name: Total Injuries, Length: 88889, dtype: float64
```

In [70]:
```
df2.head()
```

Out[70]:

	Event Id	Investigation Type	Accident Number	Event Date	Location	Country	Injury Severity	A d
0	20001218X45444	Accident	SEA87LA080	1948-10-24	MOOSE CREEK, ID	United States	Fatal(2)	
1	20001218X45447	Accident	LAX94LA336	1962-07-19	BRIDGEPORT, CA	United States	Fatal(4)	
2	20061025X01555	Accident	NYC07LA005	1974-08-30	Saltville, VA	United States	Fatal(3)	
3	20001218X45448	Accident	LAX96LA321	1977-06-19	EUREKA, CA	United States	Fatal(2)	
4	20041105X01764	Accident	CHI79FA064	1979-08-02	Canton, OH	United States	Fatal(1)	

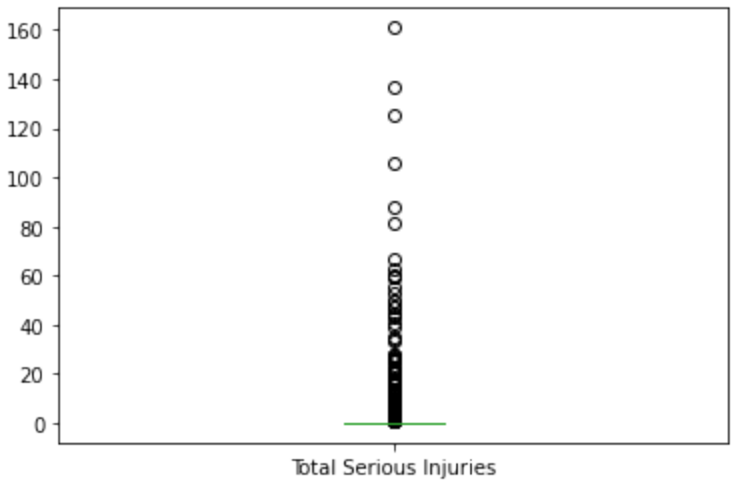
5 rows × 23 columns

3.1.2 Accuracy Test/Challenges

Identifying and clearing outliers

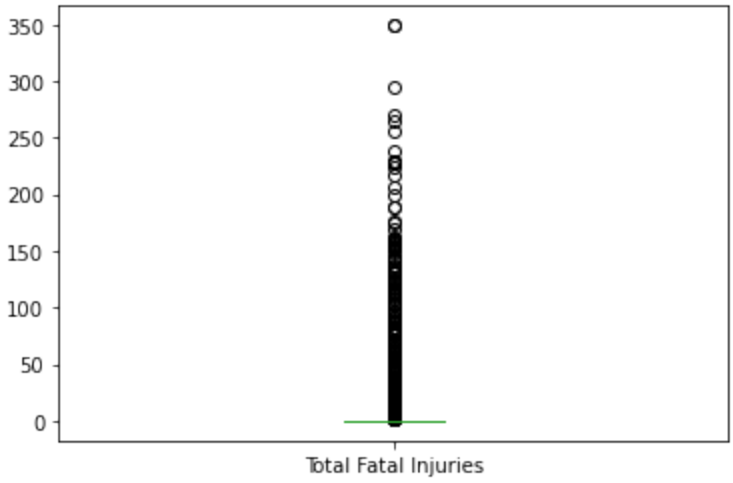
In [71]:
```
df2.boxplot(column=['Total Serious Injuries'], grid=False)
```

Out[71]: <AxesSubplot:>



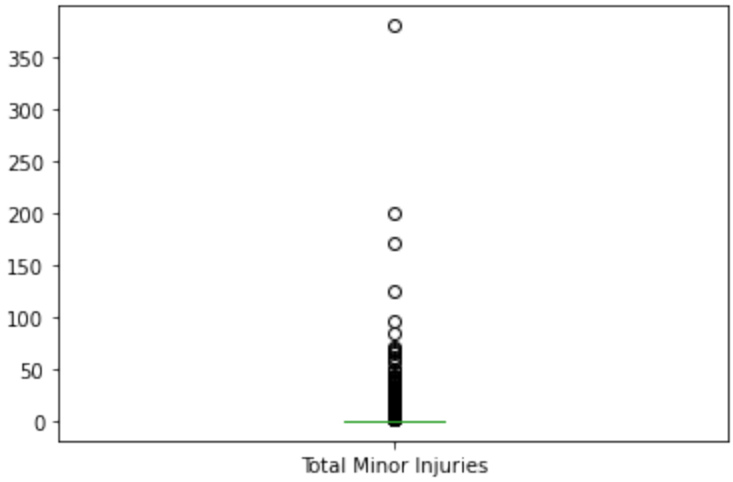
In [72]:
```
df2.boxplot(column=['Total Fatal Injuries'], grid=False)
```

Out[72]: <AxesSubplot:>



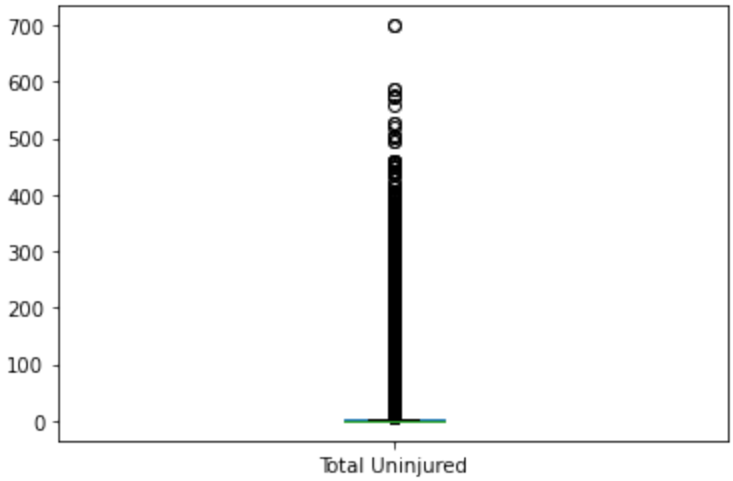
In [73]:
```
df2.boxplot(column=['Total Minor Injuries'], grid=False)
```

Out[73]: <AxesSubplot:>



In [74]:
```
df2.boxplot(column=['Total Uninjured'], grid=False)
```

Out[74]: <AxesSubplot:>



In [75]:
```
# definition a function for handling outliers
def handle_outliers(df2, column):
    Q1 = df2[column].quantile(0.25)
    Q3 = df2[column].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 - 1.5 * IQR
    df2[column] = df2[column].clip(lower=lower_bound, upper=upper_bound)
    return df2
```

In [76]:
```
df2 = handle_outliers(df2, 'Total Minor Injuries')
df2
```

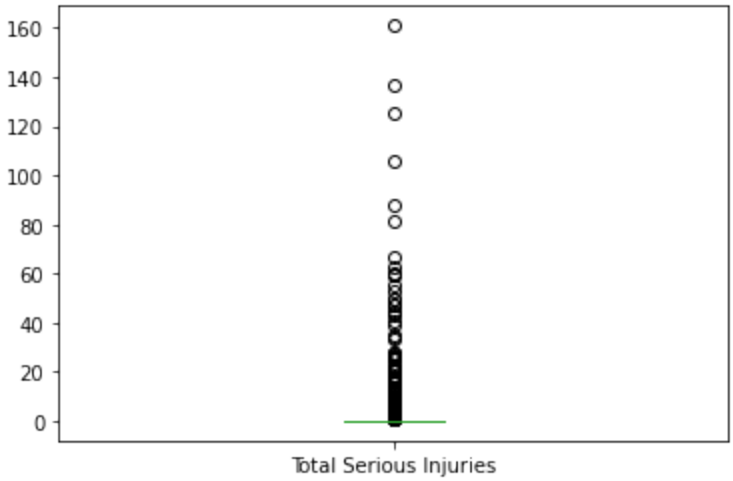
Out[76]:

	Event Id	Investigation Type	Accident Number	Event Date	Location	Country	Injury Severity
0	20001218X45444	Accident	SEA87LA080	1948-10-24	MOOSE CREEK, ID	United States	Fatal(2
1	20001218X45447	Accident	LAX94LA336	1962-07-19	BRIDGEPORT, CA	United States	Fatal(4
2	20061025X01555	Accident	NYC07LA005	1974-08-30	Saltville, VA	United States	Fatal(3
3	20001218X45448	Accident	LAX96LA321	1977-06-19	EUREKA, CA	United States	Fatal(2
4	20041105X01764	Accident	CHI79FA064	1979-08-02	Canton, OH	United States	Fatal(1
...	...	...	...	...	...	...	..
88884	20221227106491	Accident	ERA23LA093	2022-12-26	Annapolis, MD	United States	Mino
88885	20221227106494	Accident	ERA23LA095	2022-12-26	Hampton, NH	United States	NaN
88886	20221227106497	Accident	WPR23LA075	2022-12-26	Payson, AZ	United States	Non-Fata
88887	20221227106498	Accident	WPR23LA076	2022-12-26	Morgan, UT	United States	NaN
88888	20221230106513	Accident	ERA23LA097	2022-12-29	Athens, GA	United States	Mino

88889 rows × 23 columns

In [77]:
```
df2.boxplot(column=['Total Serious Injuries'], grid=False)
```

Out[77]: <AxesSubplot:>



3.1.3 Completeness Test/Challenges

Dropping all rows with missing values and creating a new cleaned dataframe to provide for credibility of the dataset

In [78]:
```
clean_df2 = df2.dropna()
clean_df2
```

Out[78]:

	Event Id	Investigation Type	Accident Number	Event Date	Location	Country	Injury Severity
0	20001218X45444	Accident	SEA87LA080	1948-10-24	MOOSE CREEK, ID	United States	Fatal(2
1	20001218X45447	Accident	LAX94LA336	1962-07-19	BRIDGEPORT, CA	United States	Fatal(4
3	20001218X45448	Accident	LAX96LA321	1977-06-19	EUREKA, CA	United States	Fatal(2
6	20001218X45446	Accident	CHI81LA106	1981-08-01	COTTON, MN	United States	Fatal(4
7	20020909X01562	Accident	SEA82DA022	1982-01-01	PULLMAN, WA	United States	Non-Fata
...	...	...	...	...	...	...	..
63893	20080104X00022	Accident	MIA08LA032	2007-12-26	SARASOTA, FL	United States	Non-Fata
63896	20071231X02008	Incident	DEN08IA044	2007-12-26	Aspen, CO	United States	Inciden
63900	20080102X00006	Accident	SEA08LA054	2007-12-28	MURRIETA, CA	United States	Non-Fata
63906	20080103X00010	Accident	DFW08LA052	2007-12-29	Crowley, TX	United States	Fatal(2
63908	20080109X00032	Accident	NYC08FA071	2007-12-30	CHEROKEE, AL	United States	Fatal(3

46498 rows × 23 columns

In [79]:
```
#checking missing values
clean_df2.isnull().sum()
```

Out[79]:
```
Event Id                  0
Investigation Type        0
Accident Number           0
Event Date                0
Location                  0
Country                   0
Injury Severity           0
Aircraft damage           0
Registration Number       0
Make                      0
Model                     0
Amateur Built             0
Number of Engines         0
Engine Type               0
Purpose of flight         0
Total Fatal Injuries      0
Total Serious Injuries    0
Total Minor Injuries      0
Total Uninjured           0
Weather Condition         0
Broad phase of flight     0
Report Status             0
Total Injuries            0
dtype: int64
```

3.1.4 Checking for duplicates in the dataset

In [80]:
```
clean_df2.duplicated().any()
```

Out[80]:
```
False
```

No duplicates observed

In [81]:
```
clean_df2.dtypes
```

Out[81]:
```
Event Id                      object
Investigation Type            object
Accident Number               object
Event Date                datetime64[ns]
Location                      object
Country                       object
Injury Severity               object
Aircraft damage               object
Registration Number           object
Make                          object
Model                         object
Amateur Built                 object
Number of Engines            float64
Engine Type                   object
Purpose of flight             object
Total Fatal Injuries         float64
Total Serious Injuries       float64
Total Minor Injuries         float64
Total Uninjured              float64
Weather Condition             object
Broad phase of flight         object
Report Status                 object
Total Injuries               float64
dtype: object
```

In [82]:
```
# Fixing colum names to be in upper case
clean_df2.columns = map(lambda x: str(x).upper(), clean_df2)
clean_df2.head()
```

Out[82]:

	EVENT ID	INVESTIGATION TYPE	ACCIDENT NUMBER	EVENT DATE	LOCATION	COUNTRY	INJ SEVEI
0	20001218X45444	Accident	SEA87LA080	1948-10-24	MOOSE CREEK, ID	United States	
1	20001218X45447	Accident	LAX94LA336	1962-07-19	BRIDGEPORT, CA	United States	
3	20001218X45448	Accident	LAX96LA321	1977-06-19	EUREKA, CA	United States	
6	20001218X45446	Accident	CHI81LA106	1981-08-01	COTTON, MN	United States	
7	20020909X01562	Accident	SEA82DA022	1982-01-01	PULLMAN, WA	United States	Nc

5 rows × 23 columns

3.1.5 Exporting the Cleaned Dataset

In [83]:
```
clean_df2.to_csv('cleaned_aviation_data.csv')
```

3.2 Data Analysis

In [84]:
```
df3 = pd.read_csv('cleaned_aviation_data.csv')
df3.head()
```

Out[84]:

	Unnamed: 0	EVENT ID	INVESTIGATION TYPE	ACCIDENT NUMBER	EVENT DATE	LOCATION	COUN
0	0	20001218X45444	Accident	SEA87LA080	1948-10-24	MOOSE CREEK, ID	Unitec
1	1	20001218X45447	Accident	LAX94LA336	1962-07-19	BRIDGEPORT, CA	Unitec
2	3	20001218X45448	Accident	LAX96LA321	1977-06-19	EUREKA, CA	Unitec
3	6	20001218X45446	Accident	CHI81LA106	1981-08-01	COTTON, MN	Unitec
4	7	20020909X01562	Accident	SEA82DA022	1982-01-01	PULLMAN, WA	Unitec

5 rows × 24 columns

In [85]:
```
#delete the unnamed column
df3.drop(['Unnamed: 0'], axis=1, inplace=True)
df3.head()
```

Out[85]:

	EVENT ID	INVESTIGATION TYPE	ACCIDENT NUMBER	EVENT DATE	LOCATION	COUNTRY	INJ SEVEI
0	20001218X45444	Accident	SEA87LA080	1948-10-24	MOOSE CREEK, ID	United States	
1	20001218X45447	Accident	LAX94LA336	1962-07-19	BRIDGEPORT, CA	United States	
2	20001218X45448	Accident	LAX96LA321	1977-06-19	EUREKA, CA	United States	
3	20001218X45446	Accident	CHI81LA106	1981-08-01	COTTON, MN	United States	
4	20020909X01562	Accident	SEA82DA022	1982-01-01	PULLMAN, WA	United States	Nc

5 rows × 23 columns

In [86]:
```
df3.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 46498 entries, 0 to 46497
Data columns (total 23 columns):
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   EVENT ID                46498 non-null  object 
 1   INVESTIGATION TYPE      46498 non-null  object 
 2   ACCIDENT NUMBER         46498 non-null  object 
 3   EVENT DATE              46498 non-null  object 
 4   LOCATION                46498 non-null  object 
 5   COUNTRY                 46498 non-null  object 
 6   INJURY SEVERITY         46498 non-null  object 
 7   AIRCRAFT DAMAGE         46498 non-null  object 
 8   REGISTRATION NUMBER     46498 non-null  object 
 9   MAKE                    46498 non-null  object 
 10  MODEL                   46498 non-null  object 
 11  AMATEUR BUILT           46498 non-null  object 
 12  NUMBER OF ENGINES       46498 non-null  float64
 13  ENGINE TYPE             46498 non-null  object 
 14  PURPOSE OF FLIGHT       46498 non-null  object 
 15  TOTAL FATAL INJURIES    46498 non-null  float64
 16  TOTAL SERIOUS INJURIES  46498 non-null  float64
 17  TOTAL MINOR INJURIES    46498 non-null  float64
 18  TOTAL UNINJURED         46498 non-null  float64
 19  WEATHER CONDITION       46498 non-null  object 
 20  BROAD PHASE OF FLIGHT   46498 non-null  object 
 21  REPORT STATUS           46498 non-null  object 
 22  TOTAL INJURIES          46498 non-null  float64
dtypes: float64(6), object(17)
memory usage: 8.2+ MB
```

In [87]:
```
df3.info(verbose = False)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 46498 entries, 0 to 46497
Columns: 23 entries, EVENT ID to TOTAL INJURIES
dtypes: float64(6), object(17)
memory usage: 8.2+ MB
```

Concise Summary Statistics

In [88]:
```
df3.describe()
```

Out[88]:

	NUMBER OF ENGINES	TOTAL FATAL INJURIES	TOTAL SERIOUS INJURIES	TOTAL MINOR INJURIES	TOTAL UNINJURED	TOTAL INJURIES
count	46498.000000	46498.000000	46498.000000	46498.0	46498.000000	46498.000000
mean	1.133834	0.399910	0.192718	0.0	2.704934	0.918986
std	0.409843	2.338675	0.824727	0.0	16.927730	3.295152
min	0.000000	0.000000	0.000000	0.0	0.000000	0.000000
25%	1.000000	0.000000	0.000000	0.0	0.000000	0.000000
50%	1.000000	0.000000	0.000000	0.0	1.000000	0.000000
75%	1.000000	0.000000	0.000000	0.0	2.000000	1.000000
max	4.000000	230.000000	81.000000	0.0	507.000000	283.000000

Describing Categorical data

In [89]:
```
df3.describe(include='object')
```

Out[89]:

	EVENT ID	INVESTIGATION TYPE	ACCIDENT NUMBER	EVENT DATE	LOCATION	COUNTRY	S
count	46498	46498	46498	46498	46498	46498	
unique	45897	2	46498	6957	13280	36	
top	20001214X45071	Accident	NYC95LA005	1982-05-16	ANCHORAGE, AK	United States	
freq	3	45530	1	25	346	46231	

Out of 46,498 incidents and accidents, there was a reported 36,677 non-fatal injuries and 32,801 substantial damages to the aircraft. 11,366 of the incidences occurred during landing with 42,290 occuring during VMC weather conditions. 41,976 of the incidences involved aircrafts with reciprocating engine.

3.2.1 Univariate analysis

In [90]:
```
mode_values = df3[['INVESTIGATION TYPE', 'NUMBER OF ENGINES' ,'INJURY SE
mode_values
```

Out[90]:

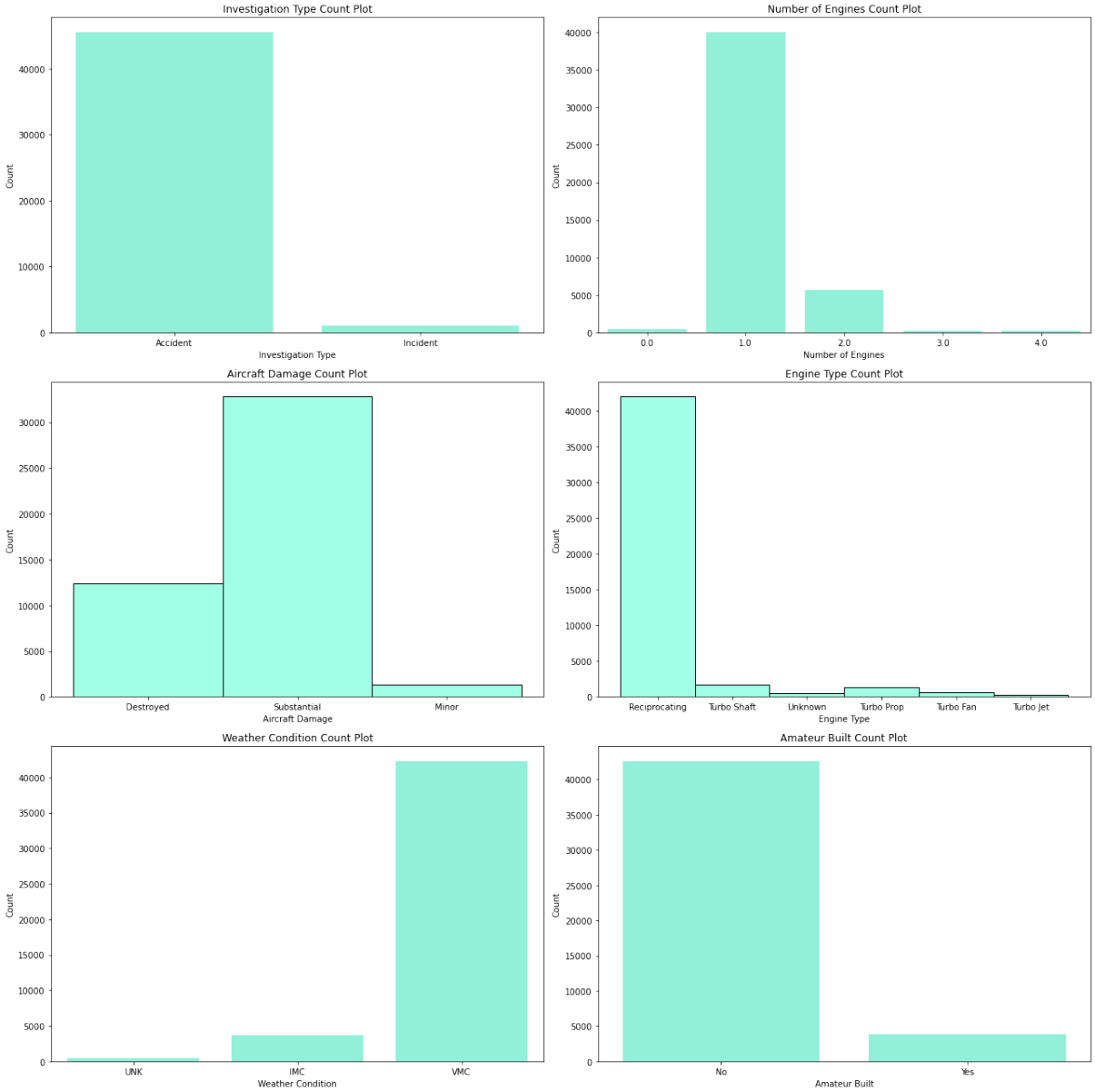
	INVESTIGATION TYPE	NUMBER OF ENGINES	INJURY SEVERITY	AIRCRAFT DAMAGE	MAKE	MODEL	ENGINE TYPE
0	Accident	1.0	Non-Fatal	Substantial	Cessna	152	Reciprocating

Most incidences resulting from aircraft accidents were non fatal. There was however substantial damage to the aircrafts resulting from the incidences.

Cessna model 152 aircraft with one reciprocating engine experienced the most accidents and incidents.

Most of the safety incidences and accidents were reported on general aviation as compared to commercial services and military planes.

In [91]:
```
fig, axs = plt.subplots(nrows=3, ncols=2, figsize=(18,18))

sns.countplot(x=df3['INVESTIGATION TYPE'], ax=axs[0,0], color='#83FFE1')
axs[0,0].set_title('Investigation Type Count Plot')
axs[0,0].set_ylabel('Count')
axs[0,0].set_xlabel('Investigation Type')


sns.countplot(x=df3['NUMBER OF ENGINES'], ax=axs[0,1], color='#83FFE1')
axs[0,1].set_title('Number of Engines Count Plot')
axs[0,1].set_ylabel('Count')
axs[0,1].set_xlabel('Number of Engines')

#flight_count = df3['PURPOSE OF FLIGHT'].value_counts()
sns.histplot(x=df3['ENGINE TYPE'], bins=5, ax=axs[1,1], color='#83FFE1')
axs[1,1].set_title('Engine Type Count Plot')
axs[1,1].set_ylabel('Count')
axs[1,1].set_xlabel('Engine Type')

sns.histplot(x=df3['AIRCRAFT DAMAGE'], bins=5, ax=axs[1,0], color='#83FF
axs[1,0].set_title('Aircraft Damage Count Plot')
axs[1,0].set_ylabel('Count')
axs[1,0].set_xlabel('Aircraft Damage')

sns.countplot(x=df3['WEATHER CONDITION'], ax=axs[2,0], color='#83FFE1')
axs[2,0].set_title('Weather Condition Count Plot')
axs[2,0].set_ylabel('Count')
axs[2,0].set_xlabel('Weather Condition')


sns.countplot(x=df3['AMATEUR BUILT'], ax=axs[2,1], color='#83FFE1')
axs[2,1].set_title('Amateur Built Count Plot')
axs[2,1].set_ylabel('Count')
axs[2,1].set_xlabel('Amateur Built')


plt.tight_layout()
plt.show()
```

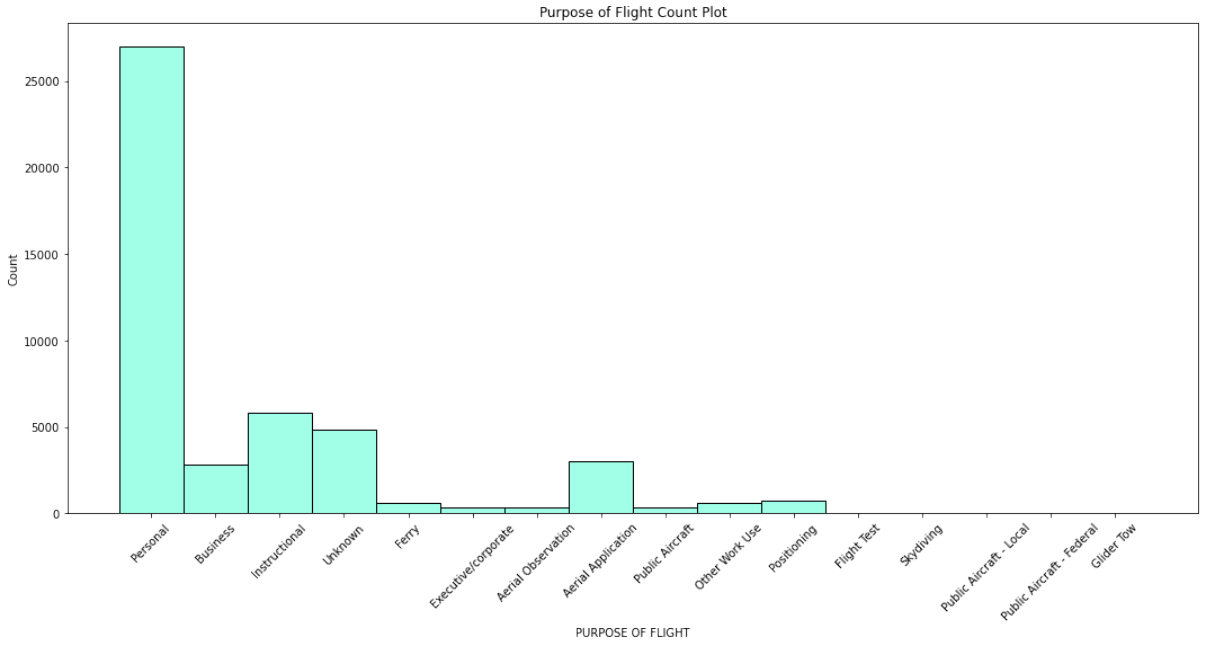


A majority of the aircrafts involved in accidents were not amateur built. Amateur-built aircraft is defined as an aircraft "the major portion of which has been fabricated and assembled by person(s) who undertook the construction project solely for their own education or recreation."

Instrument Meteorological Conditions (IMC) are weather conditions that require pilots to fly primarily by reference to flight instruments, and therefore under instrument flight rules (IFR). On the other hand, Visual Meteorological Conditions (VMC) is an aviation flight category that allows visual flight rules (VFR) in public and private flights. This basically means that the pilot of an aircraft can fly according to their visual ability versus relying on their instrumentation. These criteria vary depending on the airspace class and include: Visibility: The minimum visibility requirements under VMC range from 1 mile (1.6 km) to 5 miles (8 km), depending on the airspace class and whether the flight is conducted during the day or night.UNK impies Unknown Weather Conditions.

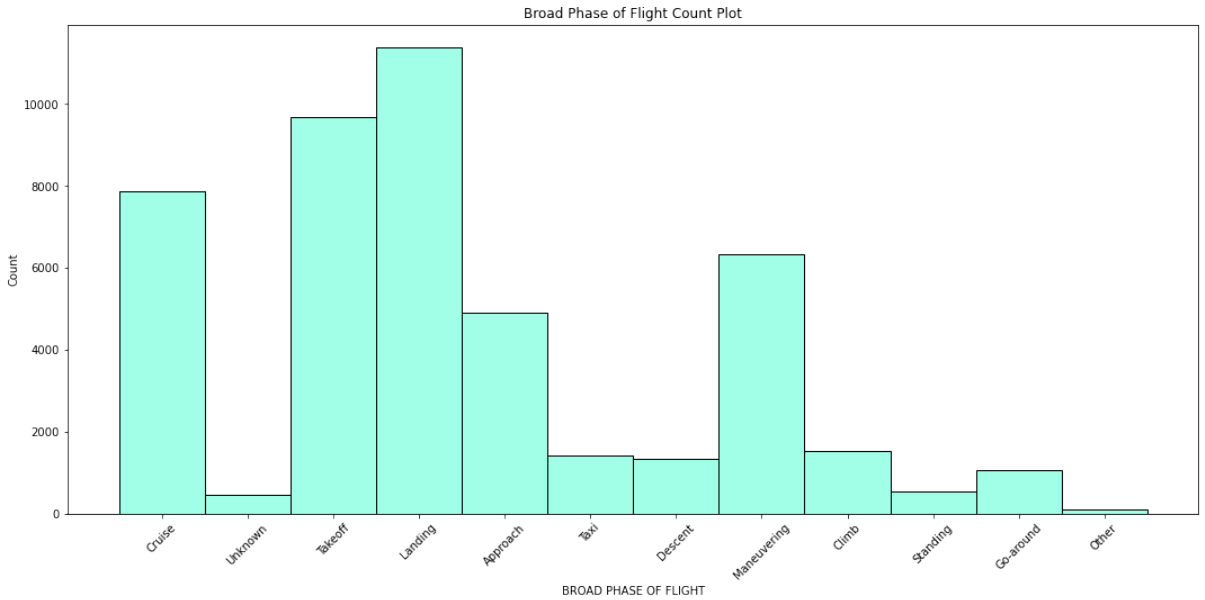
Most accidents were reported during landing with VMC weather conditions.

In [92]:
```
plt.figure(figsize=(18,8))
sns.histplot(x=df3['PURPOSE OF FLIGHT'], bins=15, color='#83FFE1')
plt.title('Purpose of Flight Count Plot')
plt.xticks(rotation=45)
plt.show()
```



Flights for personal use experienced the highest accidents followed by instructional flights. The higher rates of incidents from instructional flights could be attributed to the trainee pilots guiding the aircraft.

In [93]:
```
plt.figure(figsize=(18,8))
sns.histplot(x=df3['BROAD PHASE OF FLIGHT'], bins=15, color='#83FFE1')
plt.title('Broad Phase of Flight Count Plot')
plt.xticks(rotation=45)
plt.show()
```



Most accidents occurred during landing followed by takeoff, cruise, maneuvering and approach

In [94]:
```
#selecting only numerical columns
df3_num = df3.select_dtypes(include=['int', 'float'])
df3_num.head()
```

Out[94]:

	NUMBER OF ENGINES	TOTAL FATAL INJURIES	TOTAL SERIOUS INJURIES	TOTAL MINOR INJURIES	TOTAL UNINJURED	TOTAL INJURIES
0	1.0	2.0	0.0	0.0	0.0	2.0
1	1.0	4.0	0.0	0.0	0.0	4.0
2	1.0	2.0	0.0	0.0	0.0	2.0
3	1.0	4.0	0.0	0.0	0.0	4.0
4	1.0	0.0	0.0	0.0	2.0	0.0

In [95]:
```
df3_num.agg(['mean', 'median'])
```

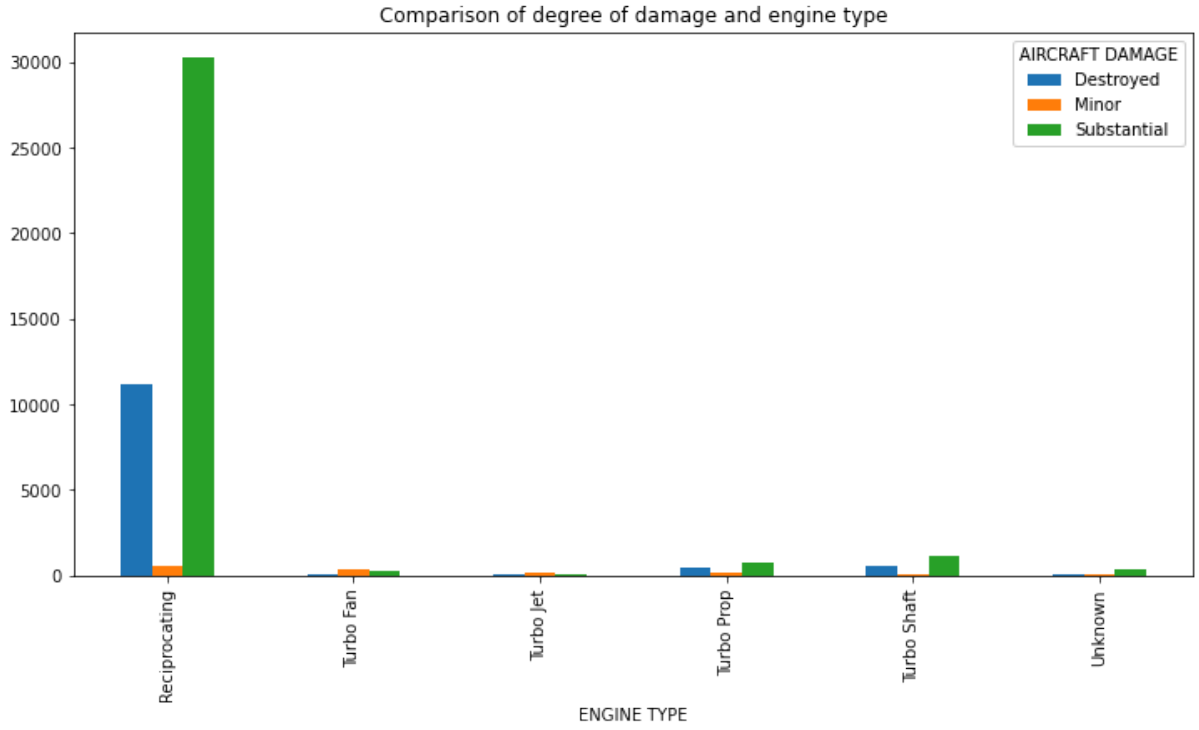
Out[95]:

	NUMBER OF ENGINES	TOTAL FATAL INJURIES	TOTAL SERIOUS INJURIES	TOTAL MINOR INJURIES	TOTAL UNINJURED	TOTAL INJURIES
mean	1.133834	0.39991	0.192718	0.0	2.704934	0.918986
median	1.000000	0.00000	0.000000	0.0	1.000000	0.000000

The average fatality rate per incident is 0.4 with a serious injury rate of 0.19.

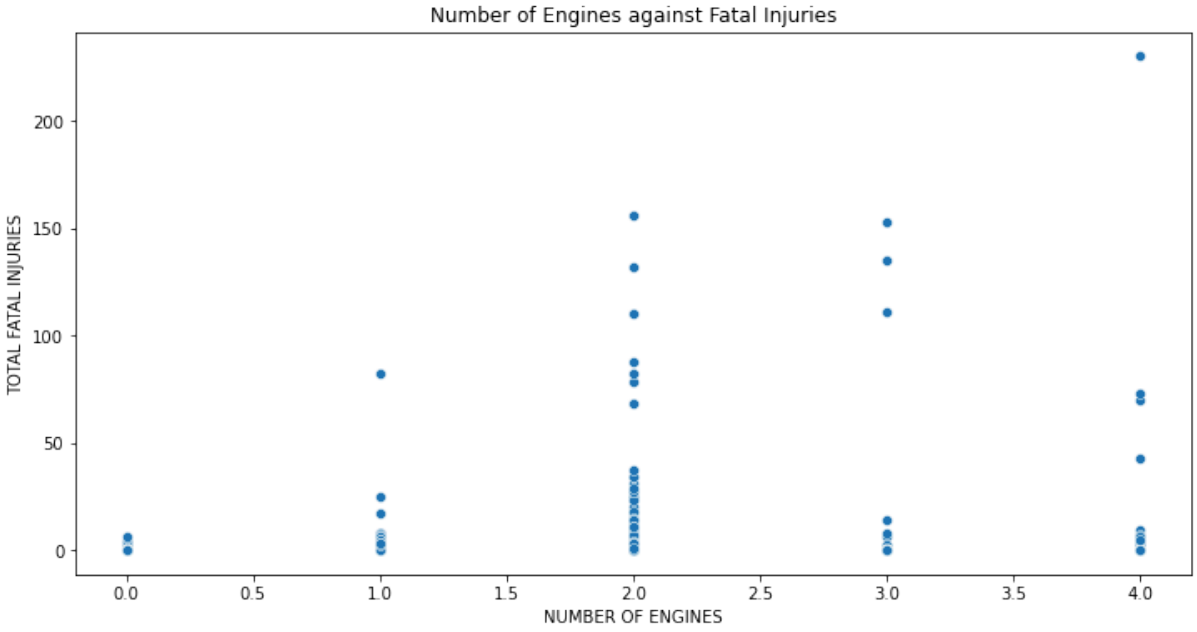
3.2.2 Bivariate Analysis

In [96]:
```
model_class = pd.crosstab(df3['ENGINE TYPE'], df3['AIRCRAFT DAMAGE'])
model_class.plot(kind='bar', figsize=(12,6))
plt.title('Comparison of degree of damage and engine type')
plt.show()
```



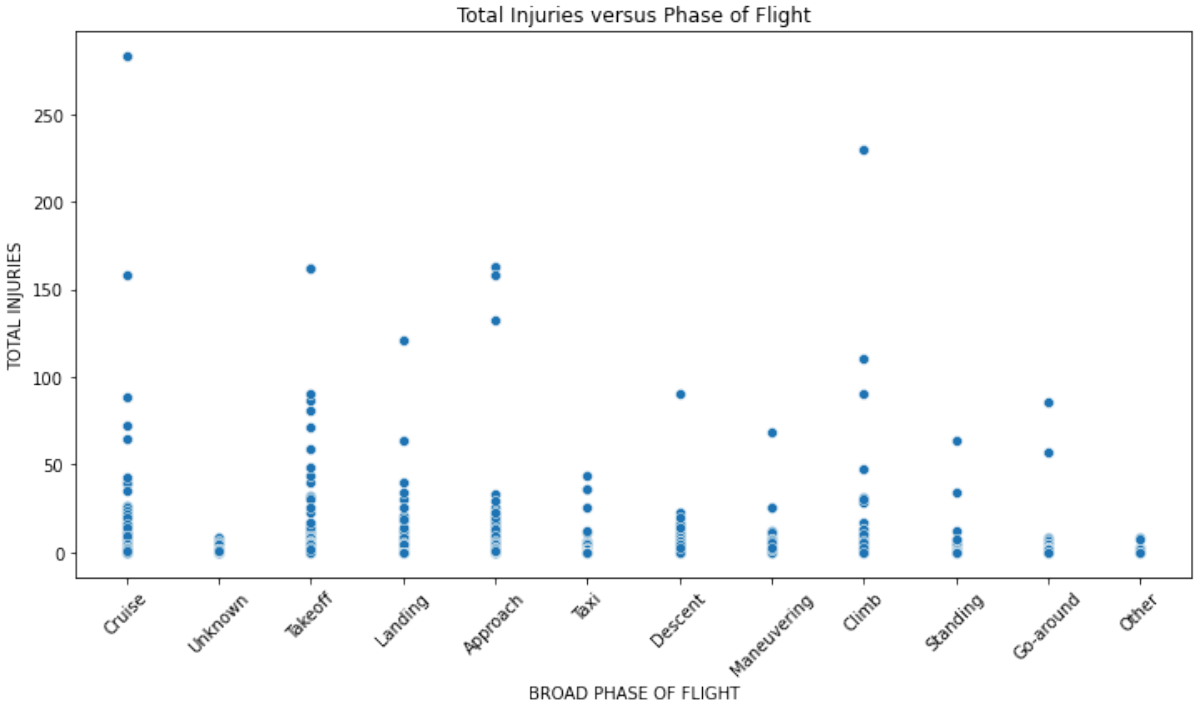
Reciprocating engines experience more failures. The best performing engines are Turbo jet

In [97]:
```
plt.figure(figsize=(12,6))
sns.scatterplot(x=df3['NUMBER OF ENGINES'], y=df3['TOTAL FATAL INJURIES'
plt.title('Number of Engines against Fatal Injuries')
plt.show()
```



Accidents involving aircrafts with two engines resulted to the most fatal injuries.

In [98]:
```
plt.figure(figsize=(12,6))
sns.scatterplot(x=df3['BROAD PHASE OF FLIGHT'], y=df3['TOTAL INJURIES'])
plt.title('Total Injuries versus Phase of Flight')
plt.xticks(rotation=45)
plt.show()
```



The flight phases that require keen attention are takeoff, cruise, approach, landing and climb

In [99]:
```
data_int = df3[['NUMBER OF ENGINES', 'TOTAL FATAL INJURIES', 'TOTAL SERI
corr = data_int.corr()
corr
```

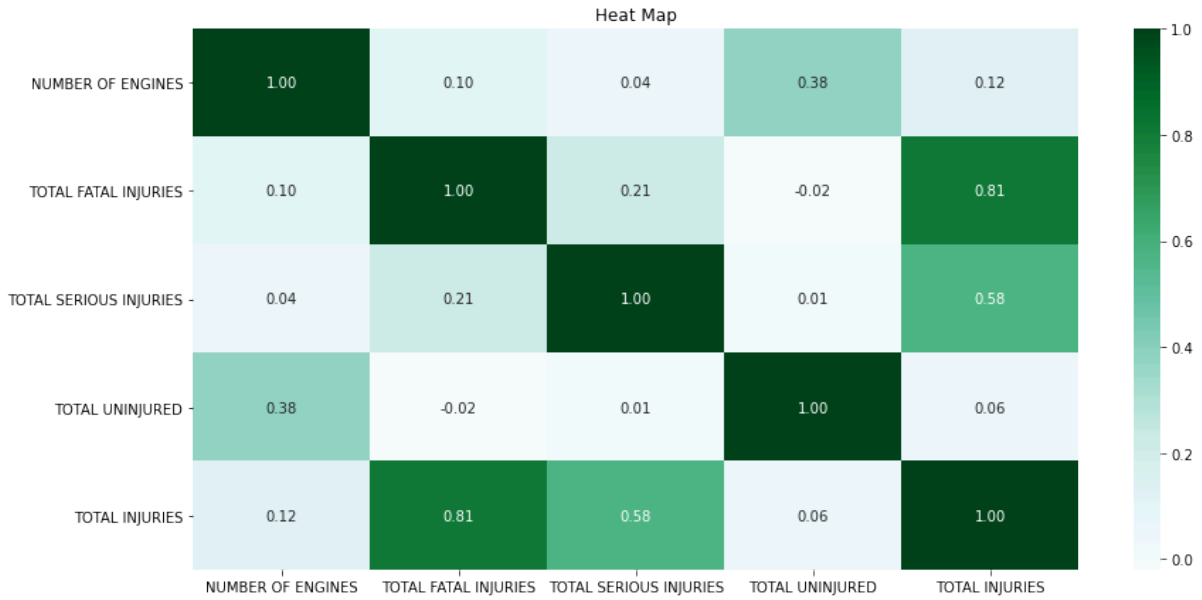
Out[99]:

	NUMBER OF ENGINES	TOTAL FATAL INJURIES	TOTAL SERIOUS INJURIES	TOTAL UNINJURED	TOTAL INJURIES
NUMBER OF ENGINES	1.000000	0.104144	0.042613	0.375839	0.124680
TOTAL FATAL INJURIES	0.104144	1.000000	0.208296	-0.019982	0.813973
TOTAL SERIOUS INJURIES	0.042613	0.208296	1.000000	0.009554	0.577254
TOTAL UNINJURED	0.375839	-0.019982	0.009554	1.000000	0.059870
TOTAL INJURIES	0.124680	0.813973	0.577254	0.059870	1.000000

In [100]:
```
print(plt.colormaps(), end= ' ')
```

```
['Accent', 'Accent_r', 'Blues', 'Blues_r', 'BrBG', 'BrBG_r', 'BuGn', 'B
uGn_r', 'BuPu', 'BuPu_r', 'CMRmap', 'CMRmap_r', 'Dark2', 'Dark2_r', 'Gn
Bu', 'GnBu_r', 'Greens', 'Greens_r', 'Greys', 'Greys_r', 'OrRd', 'OrRd_
r', 'Oranges', 'Oranges_r', 'PRGn', 'PRGn_r', 'Paired', 'Paired_r', 'Pa
stel1', 'Pastel1_r', 'Pastel2', 'Pastel2_r', 'PiYG', 'PiYG_r', 'PuBu',
'PuBuGn', 'PuBuGn_r', 'PuBu_r', 'PuOr', 'PuOr_r', 'PuRd', 'PuRd_r', 'Pu
rples', 'Purples_r', 'RdBu', 'RdBu_r', 'RdGy', 'RdGy_r', 'RdPu', 'RdPu_
r', 'RdYlBu', 'RdYlBu_r', 'RdYlGn', 'RdYlGn_r', 'Reds', 'Reds_r', 'Set
1', 'Set1_r', 'Set2', 'Set2_r', 'Set3', 'Set3_r', 'Spectral', 'Spectral
_r', 'Wistia', 'Wistia_r', 'YlGn', 'YlGnBu', 'YlGnBu_r', 'YlGn_r', 'YlO
rBr', 'YlOrBr_r', 'YlOrRd', 'YlOrRd_r', 'afmhot', 'afmhot_r', 'autumn',
'autumn_r', 'binary', 'binary_r', 'bone', 'bone_r', 'brg', 'brg_r', 'bw
r', 'bwr_r', 'cividis', 'cividis_r', 'cool', 'cool_r', 'coolwarm', 'coo
lwarm_r', 'copper', 'copper_r', 'crest', 'crest_r', 'cubehelix', 'cubeh
elix_r', 'flag', 'flag_r', 'flare', 'flare_r', 'gist_earth', 'gist_eart
h_r', 'gist_gray', 'gist_gray_r', 'gist_heat', 'gist_heat_r', 'gist_nca
r', 'gist_ncar_r', 'gist_rainbow', 'gist_rainbow_r', 'gist_stern', 'gis
t_stern_r', 'gist_yarg', 'gist_yarg_r', 'gnuplot', 'gnuplot2', 'gnuplot
2_r', 'gnuplot_r', 'gray', 'gray_r', 'hot', 'hot_r', 'hsv', 'hsv_r', 'i
cefire', 'icefire_r', 'inferno', 'inferno_r', 'jet', 'jet_r', 'magma',
'magma_r', 'mako', 'mako_r', 'nipy_spectral', 'nipy_spectral_r', 'ocea
n', 'ocean_r', 'pink', 'pink_r', 'plasma', 'plasma_r', 'prism', 'prism_
r', 'rainbow', 'rainbow_r', 'rocket', 'rocket_r', 'seismic', 'seismic_
r', 'spring', 'spring_r', 'summer', 'summer_r', 'tab10', 'tab10_r', 'ta
b20', 'tab20_r', 'tab20b', 'tab20b_r', 'tab20c', 'tab20c_r', 'terrain',
'terrain_r', 'turbo', 'turbo_r', 'twilight', 'twilight_r', 'twilight_sh
ifted', 'twilight_shifted_r', 'viridis', 'viridis_r', 'vlag', 'vlag_r',
'winter', 'winter_r']
```

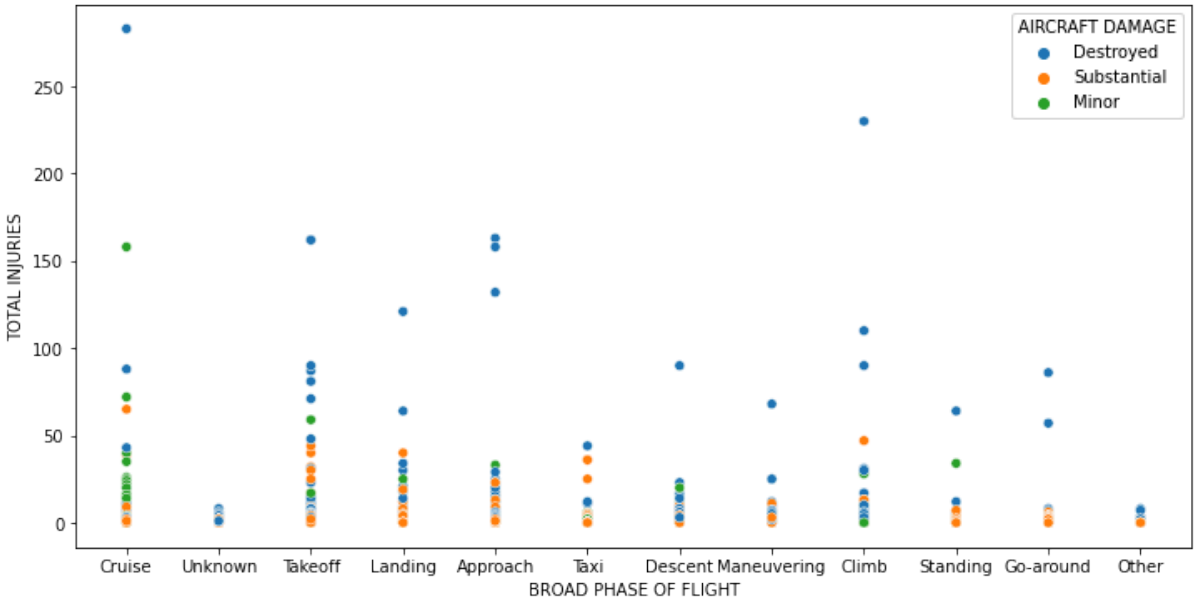
In [101]:
```
plt.figure(figsize=(14, 7))
sns.heatmap(corr, annot= True, fmt='.2f', cmap='BuGn')
plt.title('Heat Map')
plt.show()
```



There is a low positive correlation between the number of engines and total injuries experienced during an accident

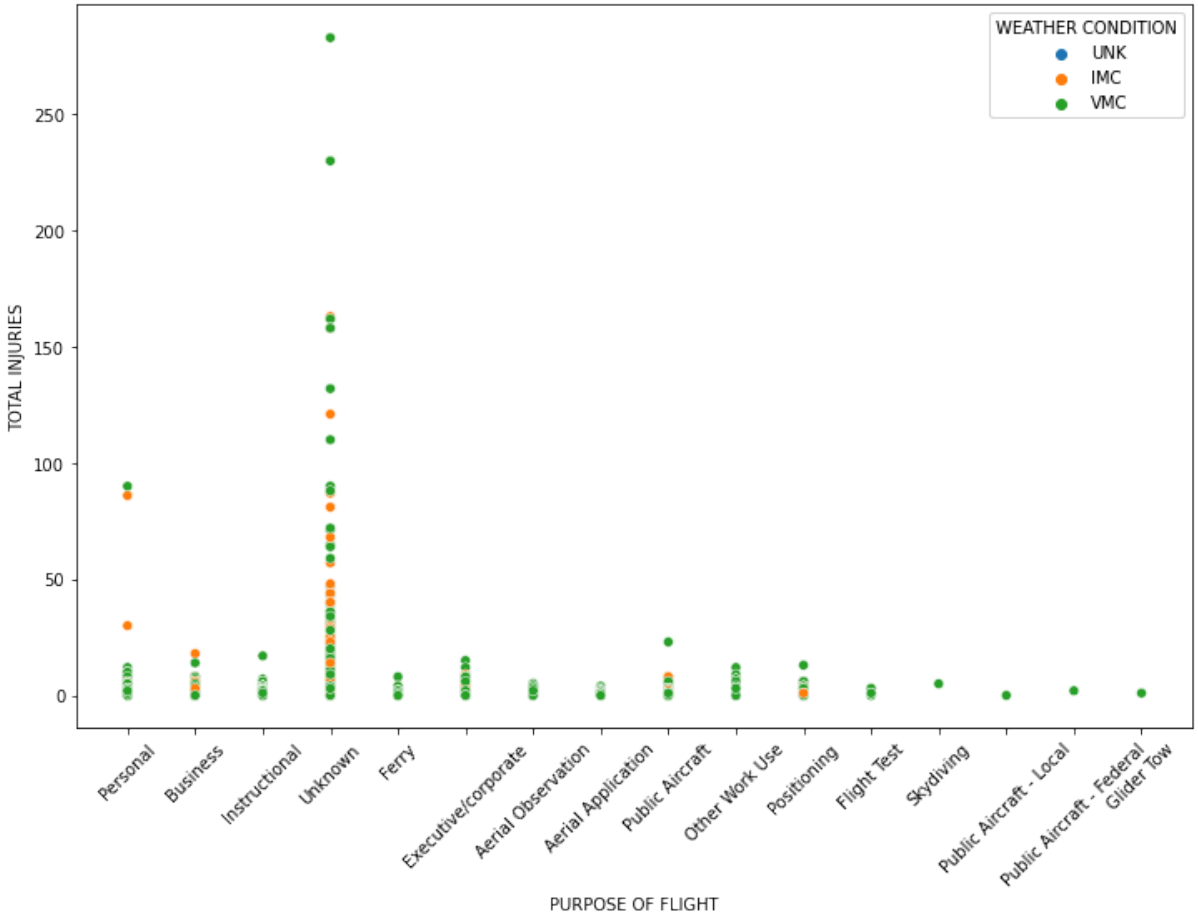
3.2.3 Multivariate Analysis

In [102]:
```
plt.figure(figsize=(12,6))
sns.scatterplot(x=df3['BROAD PHASE OF FLIGHT'], y=df3['TOTAL INJURIES'],
plt.show()
```

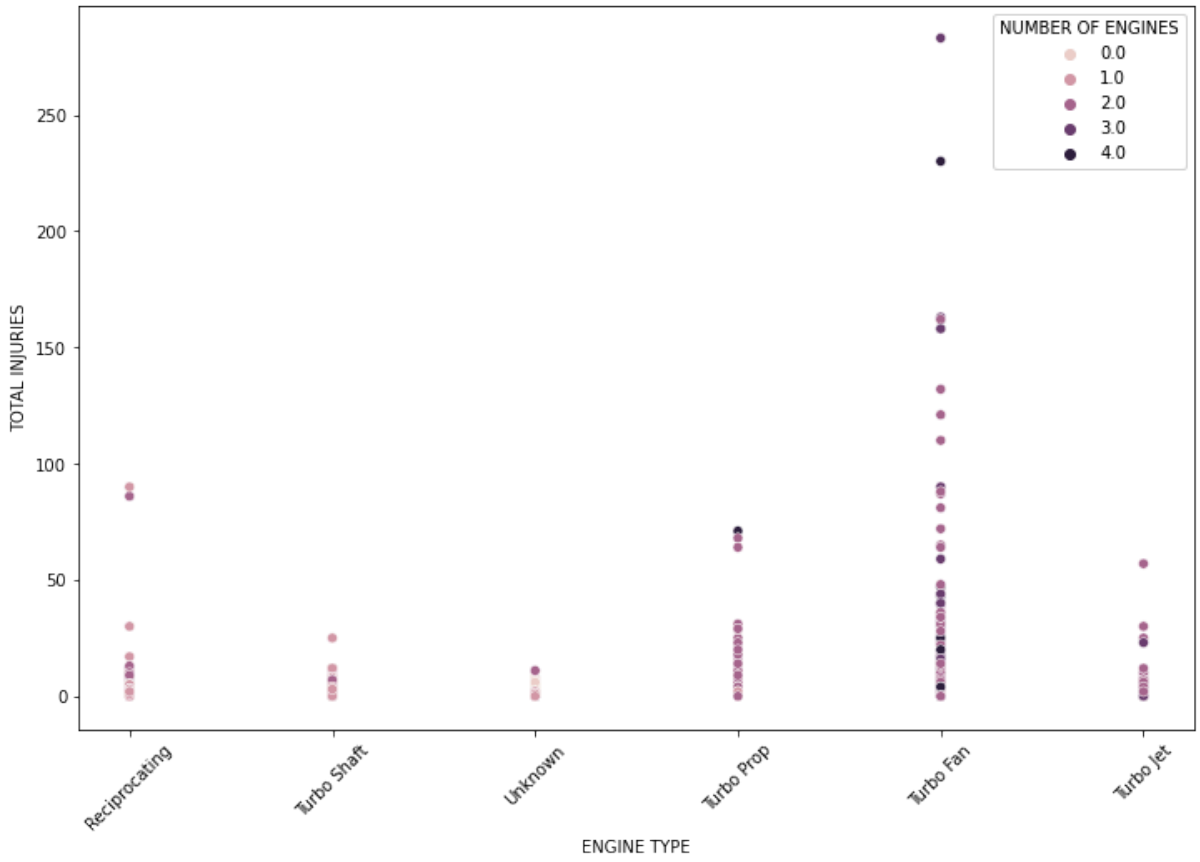


Accidents occurrences during takeoff, cruise, landing ans approach resulted to either destruction of the aircraft or its substantial damage. Safety concerns on this areas should be stringent

In [103]:
```
plt.figure(figsize=(12,8))
sns.scatterplot(y=df3['TOTAL INJURIES'], x=df3['PURPOSE OF FLIGHT'], hue
plt.xticks(rotation=45)
plt.show()
```



In [104]:
```
plt.figure(figsize=(12,8))
sns.scatterplot(y=df3['TOTAL INJURIES'], hue=df3['NUMBER OF ENGINES'], x
plt.xticks(rotation=45)
plt.show()
```



4 RECOMMENDATIONS

The company should focus on provide safe aviation transport in the personal (general aviation) niches. This is because the data indicated flights for personal use experience most accidents and incidences. This is an area which can per explored for potential growth.

It is recommended that the aircrafts to be procured should have turbo jet or turbo fan engines with at least 3 engines. Aircrafts mounted with reciprocating engines are worst performing in terms of aviation safety.

The aircrafts to be procured should have advance and more sophisticate weather monitoring instrumentations to be used during IMC condiitons. In addition, a greater percentage of the recorded accidents occurred during VMC weather conditions. It is therefore recommended that Maureen Inc develops Standard Operating Procedures that are safety stringent during VMC conditions.

Considering the commercial nature of the company, it is recommended that the organizations procures professionally built aircrafts as opposed to amateur built despite the statistics indiation most accidents occured on professionally built aircrafts.