0.1 Loading Python packages

In [1]:
```
# numpy for high level mathematical functions and working with Arrays
import numpy as np
# pandas data manipulation and analysis for tablular data
import pandas as pd
# seaborn and matplotlib for data visualization
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

0.2 Reading the Dataset from our CSV file

In [2]:
```
df = pd.read_csv('AviationData.csv', encoding="cp1252")
```

```
C:\Users\Sam\AppData\Local\Temp\ipykernel_1852\3587365544.py:1: DtypeWarning: Columns (6,7,28) have mixed types. Specify dtype
option on import or set low_memory=False.
  df = pd.read_csv('AviationData.csv', encoding="cp1252")
```

0.3 Preview our dataset

In [3]:
```
#Checking the first 5 rows
df.head()
```

Out[3]:

	Event.Id	Investigation.Type	Accident.Number	Event.Date	Location	Country	Latitude	Longitude	Airport.Code	Airport.Name
0	20001218X45444	Accident	SEA87LA080	1948-10-24	MOOSE CREEK, ID	United States	NaN	NaN	NaN	NaN
1	20001218X45447	Accident	LAX94LA336	1962-07-19	BRIDGEPORT, CA	United States	NaN	NaN	NaN	NaN
2	20061025X01555	Accident	NYC07LA005	1974-08-30	Saltville, VA	United States	36.922223	-81.878056	NaN	NaN
3	20001218X45448	Accident	LAX96LA321	1977-06-19	EUREKA, CA	United States	NaN	NaN	NaN	NaN
4	20041105X01764	Accident	CHI79FA064	1979-08-02	Canton, OH	United States	NaN	NaN	NaN	NaN

5 rows × 31 columns

In [4]:
```
#checking the last five rows
df.tail()
```

Out[4]:

	Event.Id	Investigation.Type	Accident.Number	Event.Date	Location	Country	Latitude	Longitude	Airport.Code	Airport.Name
88884	20221227106491	Accident	ERA23LA093	2022-12-26	Annapolis, MD	United States	NaN	NaN	NaN	NaN
88885	20221227106494	Accident	ERA23LA095	2022-12-26	Hampton, NH	United States	NaN	NaN	NaN	NaN
88886	20221227106497	Accident	WPR23LA075	2022-12-26	Payson, AZ	United States	341525N	1112021W	PAN	PAYSON
88887	20221227106498	Accident	WPR23LA076	2022-12-26	Morgan, UT	United States	NaN	NaN	NaN	NaN
88888	20221230106513	Accident	ERA23LA097	2022-12-29	Athens, GA	United States	NaN	NaN	NaN	NaN

5 rows × 31 columns

In [5]:
```
#Checking the number of rows and columns in the dataset(Dataset has 88889 rows and 31 columns)
df.shape
```

Out[5]: (88889, 31)

In [7]:
```
#Checking the dataset information
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            50132 non-null  object 
 9   Airport.Name            52704 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     87507 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             81793 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           82505 non-null  object 
 30  Publication.Date        75118 non-null  object 
dtypes: float64(5), object(26)
memory usage: 21.0+ MB
```

In [8]:
```
#dataset summary statistics for numerical columns
df.describe()
```

Out[8]:

	Number.of.Engines	Total.Fatal.Injuries	Total.Serious.Injuries	Total.Minor.Injuries	Total.Uninjured
count	82805.000000	77488.000000	76379.000000	76956.000000	82977.000000
mean	1.146585	0.647855	0.279881	0.357061	5.325440
std	0.446510	5.485960	1.544084	2.235625	27.913634
min	0.000000	0.000000	0.000000	0.000000	0.000000
25%	1.000000	0.000000	0.000000	0.000000	0.000000
50%	1.000000	0.000000	0.000000	0.000000	1.000000
75%	1.000000	0.000000	0.000000	0.000000	2.000000
max	8.000000	349.000000	161.000000	380.000000	699.000000

In [149]:
```
#Describe categorical features

df.describe(include='object')
```

Out[149]:

	Event.Id	Investigation.Type	Accident.Number	Event.Date	Location	Country	Latitude	Longitude	Airport.Code	Airport.Name
count	88889	88889	88889	88889	88837	88663	34382	34373	50132	527
unique	87951	2	88863	14782	27758	219	25592	27156	10374	248
top	20001212X19172	Accident	CEN22LA149	1984-06-30	ANCHORAGE, AK	United States	332739N	0112457W	NONE	Priv
freq	3	85015	2	25	434	82248	19	24	1488	2

4 rows × 26 columns

0.4 Cleaning our Dataset

In [10]:
```
#create a copy of the dataset

df1 = df.copy(deep=True)
```

In [151]:
```
df1.columns
```

Out[151]:
```
Index(['Event.Id', 'Investigation.Type', 'Accident.Number', 'Event.Date',
       '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.Description',
       'Schedule', 'Purpose.of.flight', 'Air.carrier', 'Total.Fatal.Injuries',
       'Total.Serious.Injuries', 'Total.Minor.Injuries', 'Total.Uninjured',
       'Weather.Condition', 'Broad.phase.of.flight', 'Report.Status',
       'Publication.Date'],
      dtype='object')
```

In [150]:
```
#Dataframe for useful columns df2
df2 = df1[['Injury.Severity','Aircraft.damage','Make','Purpose.of.flight','Total.Fatal.Injuries','Total.Serious.Injuries','Total.
```

In [12]:
```
#Check for null values

df2.isnull().sum()
```

Out[12]:
```
Injury.Severity            1000
Aircraft.damage            3194
Make                         63
Purpose.of.flight          6192
Total.Fatal.Injuries      11401
Total.Serious.Injuries    12510
Total.Minor.Injuries      11933
Total.Uninjured            5912
dtype: int64
```

In [156]:
```
#dropping the null values
df2.dropna(how="all")
```

Out[156]:

	Injury.Severity	Aircraft.damage	Make	Purpose.of.flight	Total.Fatal.Injuries	Total.Serious.Injuries	Total.Minor.Injuries	Total.Uninjured
0	Fatal(2)	Destroyed	Stinson	Personal	2.0	0.0	0.0	0.0
1	Fatal(4)	Destroyed	Piper	Personal	4.0	0.0	0.0	0.0
2	Fatal(3)	Destroyed	Cessna	Personal	3.0	NaN	NaN	NaN
3	Fatal(2)	Destroyed	Rockwell	Personal	2.0	0.0	0.0	0.0
4	Fatal(1)	Destroyed	Cessna	Personal	1.0	2.0	NaN	0.0
...	...	...	...	...	...	...	...	...
88884	Minor	NaN	PIPER	Personal	0.0	1.0	0.0	0.0
88885	NaN	NaN	BELLANCA	NaN	0.0	0.0	0.0	0.0
88886	Non-Fatal	Substantial	AMERICAN CHAMPION AIRCRAFT	Personal	0.0	0.0	0.0	1.0
88887	NaN	NaN	CESSNA	Personal	0.0	0.0	0.0	0.0
88888	Minor	NaN	PIPER	Personal	0.0	1.0	0.0	1.0

88889 rows × 8 columns

In [157]:
```
df2.isnull().sum()
```

Out[157]:
```
Injury.Severity            1000
Aircraft.damage            3194
Make                         63
Purpose.of.flight          6192
Total.Fatal.Injuries      11401
Total.Serious.Injuries    12510
Total.Minor.Injuries      11933
Total.Uninjured            5912
dtype: int64
```

In [17]:
```
#Check unique values in the Injury.Severity column
df2['Injury.Severity'].unique()
```

Out[17]:
```
array(['Fatal(2)', 'Fatal(4)', 'Fatal(3)', 'Fatal(1)', 'Non-Fatal',
       'Incident', 'Fatal(8)', 'Fatal(78)', 'Fatal(7)', 'Fatal(6)',
       'Fatal(5)', 'Fatal(153)', 'Fatal(12)', 'Fatal(14)', 'Fatal(23)',
       'Fatal(10)', 'Fatal(11)', 'Fatal(9)', 'Fatal(17)', 'Fatal(13)',
       'Fatal(29)', 'Fatal(70)', 'Unavailable', 'Fatal(135)', 'Fatal(31)',
       'Fatal(256)', 'Fatal(25)', 'Fatal(82)', 'Fatal(156)', 'Fatal(28)',
       'Fatal(18)', 'Fatal(43)', 'Fatal(15)', 'Fatal(270)', 'Fatal(144)',
       'Fatal(174)', 'Fatal(111)', 'Fatal(131)', 'Fatal(20)', 'Fatal(73)',
       'Fatal(27)', 'Fatal(34)', 'Fatal(87)', 'Fatal(30)', 'Fatal(16)',
       'Fatal(47)', 'Fatal(56)', 'Fatal(37)', 'Fatal(132)', 'Fatal(68)',
       'Fatal(54)', 'Fatal(52)', 'Fatal(65)', 'Fatal(72)', 'Fatal(160)',
       'Fatal(189)', 'Fatal(123)', 'Fatal(33)', 'Fatal(110)',
       'Fatal(230)', 'Fatal(97)', 'Fatal(349)', 'Fatal(125)', 'Fatal(35)',
       'Fatal(228)', 'Fatal(75)', 'Fatal(104)', 'Fatal(229)', 'Fatal(80)',
       'Fatal(217)', 'Fatal(169)', 'Fatal(88)', 'Fatal(19)', 'Fatal(60)',
       'Fatal(113)', 'Fatal(143)', 'Fatal(83)', 'Fatal(24)', 'Fatal(44)',
       'Fatal(64)', 'Fatal(92)', 'Fatal(118)', 'Fatal(265)', 'Fatal(26)',
       'Fatal(138)', 'Fatal(206)', 'Fatal(71)', 'Fatal(21)', 'Fatal(46)',
       'Fatal(102)', 'Fatal(115)', 'Fatal(141)', 'Fatal(55)',
```

In [18]:
```
#To remove the number of fatal injuries
df2['Injury.Severity'] = df2['Injury.Severity'].str.split('(').str[0]
```

```
C:\Users\Sam\AppData\Local\Temp\ipykernel_1852\906973350.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  df2['Injury.Severity'] = df2['Injury.Severity'].str.split('(').str[0] #To remove the number of fatal injuries
```

In [19]:
```
df2['Injury.Severity'].value_counts()
```

Out[19]:
```
Injury.Severity
Non-Fatal      67357
Fatal          17826
Incident        2219
Minor            218
Serious          173
Unavailable       96
Name: count, dtype: int64
```

In [20]:
```
#To remove rows that dont have information about injury
df2['Injury.Severity'].fillna('Unavailable',inplace=True)
df2 = df2.drop(index=list(df2[df2['Injury.Severity'] == 'Unavailable'].index.values.tolist())).reset_index(drop=True)
```

```
C:\Users\Sam\AppData\Local\Temp\ipykernel_1852\2689738102.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  df2['Injury.Severity'].fillna('Unavailable',inplace=True)
```

In [21]:
```
#filling in missing values with the mode
df2['Total.Fatal.Injuries'].fillna(df2['Total.Fatal.Injuries'].mean(), inplace=True)
df2['Total.Serious.Injuries'].fillna(df2['Total.Serious.Injuries'].mean(), inplace=True)
df2['Total.Minor.Injuries'].fillna(df2['Total.Minor.Injuries'].mean(), inplace=True)
df2['Total.Uninjured'].fillna(df2['Total.Uninjured'].mean(), inplace=True)
```

In [22]:
```
#filling in missing values with 'Unknown'
df2['Make'].fillna('Unknown',inplace=True)
df2['Aircraft.damage'].fillna('Unknown',inplace=True)
df2['Purpose.of.flight'].fillna('Unknown',inplace=True)
```

In [23]:
```
#Remove the outlier using the maximum quantile

#a. Get the max interquantile

max_TMI = df2['Total.Minor.Injuries'].quantile(0.995)
```

In [24]:
```
#check the outliers
df2[df2["Total.Minor.Injuries"] > max_TMI]
```

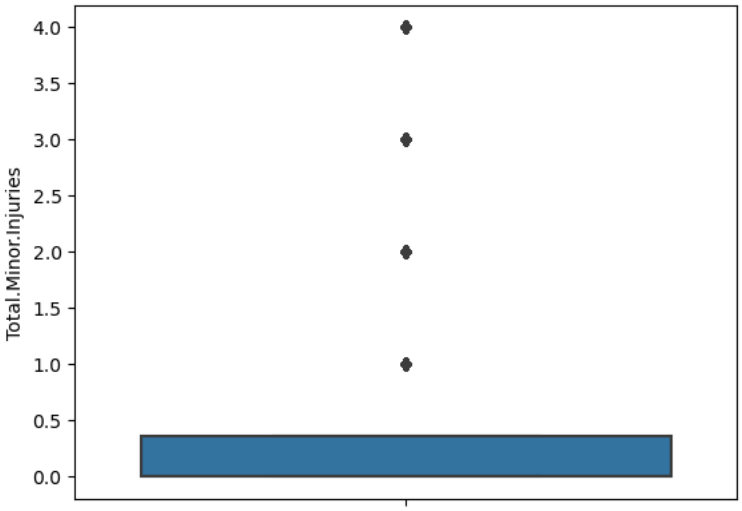
Out[24]:

	Injury.Severity	Aircraft.damage	Make	Purpose.of.flight	Total.Fatal.Injuries	Total.Serious.Injuries	Total.Minor.Injuries	Total.Uninjured
155	Fatal	Destroyed	Mcdonnell Douglas	Unknown	2.0	4.0	24.0	182.0
229	Non-Fatal	Destroyed	Mitsubishi	Business	0.0	0.0	6.0	0.0
1343	Non-Fatal	Substantial	Douglas	Unknown	0.0	0.0	25.0	113.0
1347	Incident	Unknown	Douglas	Unknown	0.0	0.0	17.0	129.0
1969	Non-Fatal	Minor	Mcdonnell Douglas	Unknown	0.0	7.0	19.0	142.0
...	...	...	...	...	...	...	...	...
83643	Fatal	Substantial	BOEING	Unknown	3.0	161.0	19.0	0.0
84612	Non-Fatal	Substantial	CESSNA	Personal	0.0	0.0	7.0	0.0
85873	Non-Fatal	Unknown	BRITTEN NORMAN	Unknown	0.0	0.0	7.0	0.0
85920	Non-Fatal	Substantial	CESSNA	Unknown	0.0	0.0	7.0	0.0
86758	Fatal	Substantial	CESSNA	Unknown	2.0	0.0	7.0	7.0

321 rows × 8 columns

In [26]:
```
#Remove the outlier by assigning the value to a new DataFrame
df3 = df2[df2["Total.Minor.Injuries"] < max_TMI]
```

In [28]:
```
#confirm removal of outlier
sns.boxplot(data=df3, y='Total.Minor.Injuries');
```



In [29]:
```
#Remove the outlier using the maximum quantile

#a. Get the max interquantile
max_TSI = df3['Total.Serious.Injuries'].quantile(0.995)
```

In [30]:
```
#check the outliers
df3[df3["Total.Serious.Injuries"] > max_TSI]
```

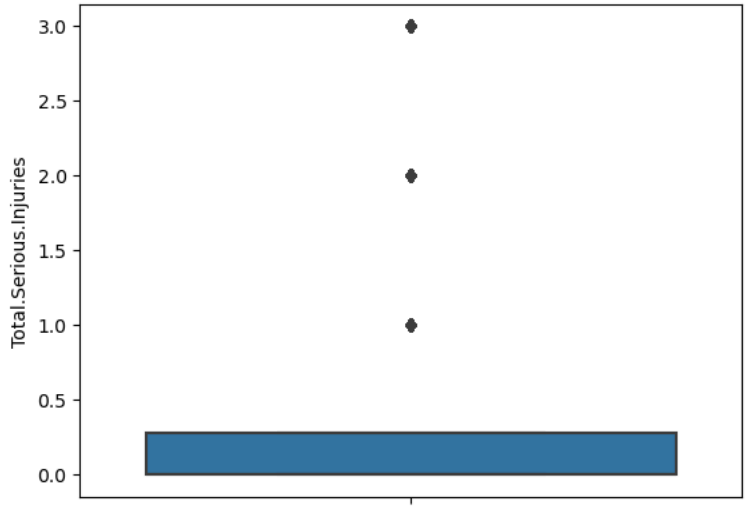
Out[30]:

	Injury.Severity	Aircraft.damage	Make	Purpose.of.flight	Total.Fatal.Injuries	Total.Serious.Injuries	Total.Minor.Injuries	Total.Uninjured
84	Fatal	Destroyed	Boeing	Unknown	78.0	6.0	3.0	0.0
214	Non-Fatal	Destroyed	Beech	Unknown	0.0	5.0	2.0	0.0
377	Fatal	Destroyed	De Havilland	Unknown	1.0	10.0	1.0	0.0
1216	Non-Fatal	Destroyed	De Havilland	Unknown	0.0	8.0	0.0	0.0
1465	Non-Fatal	Destroyed	Douglas	Unknown	0.0	5.0	0.0	0.0
...	...	...	...	...	...	...	...	...
84306	Non-Fatal	Unknown	Lindstrand	Business	0.0	13.0	2.0	1.0
85533	Non-Fatal	Unknown	AIRBUS	Unknown	0.0	8.0	1.0	185.0
86529	Fatal	Unknown	SIKORSKY	Unknown	1.0	12.0	0.0	0.0
86827	Serious	Unknown	BOEING	Unknown	0.0	6.0	1.0	121.0
87441	Fatal	Substantial	CESSNA	Unknown	1.0	5.0	0.0	0.0

213 rows × 8 columns

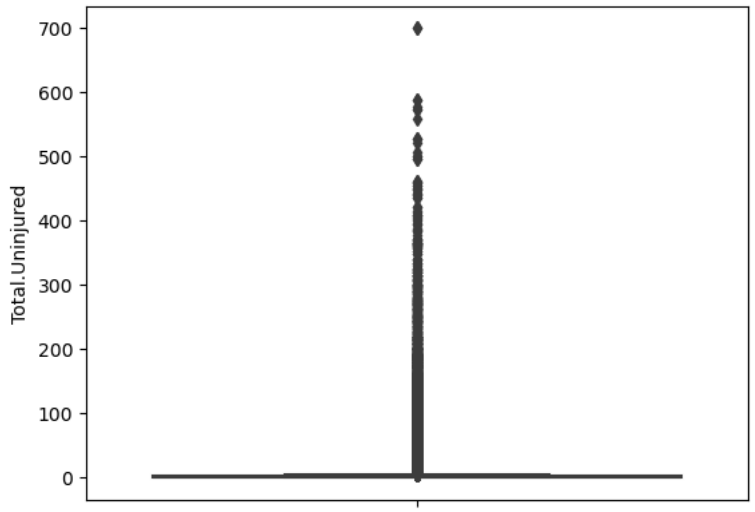
In [31]:
```
#Remove the outlier by assigning the value to a new DataFrame
df4 = df3[df3["Total.Serious.Injuries"] < max_TSI]
```

In [32]:
```
#confirm removal of outlier
sns.boxplot(data=df4, y='Total.Serious.Injuries');
```



In [33]:
```
sns.boxplot(data=df4, y='Total.Uninjured')
```

Out[33]: <Axes: ylabel='Total.Uninjured'>



In [34]:
```
#Remove the outlier using the maximum quantile

#a. Get the max interquantile
max_TU = df4['Total.Uninjured'].quantile(0.995)
max_TU
```

Out[34]: 191.0

In [35]:
```
#check the outliers
df4[df4["Total.Uninjured"] > max_TU]
```

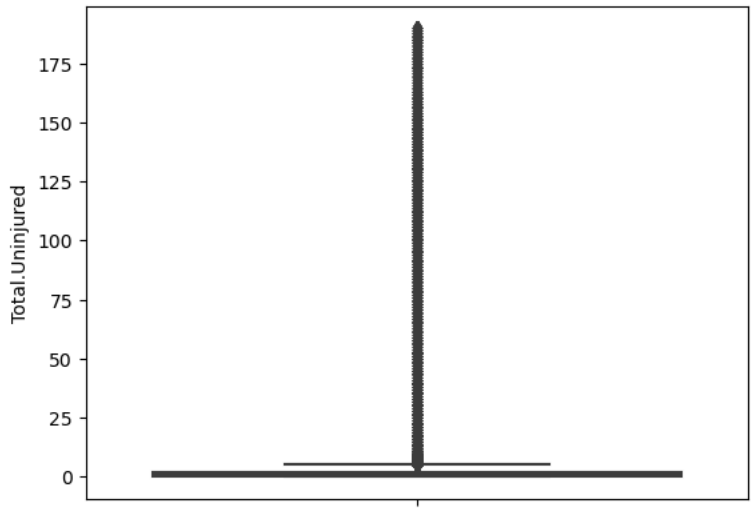
Out[35]:

	Injury.Severity	Aircraft.damage	Make	Purpose.of.flight	Total.Fatal.Injuries	Total.Serious.Injuries	Total.Minor.Injuries	Total.Uninjured
2456	Incident	Minor	Mcdonnell Douglas	Unknown	0.00000	0.000000	0.000000	393.0
3686	Incident	Minor	Mcdonnell Douglas	Unknown	0.00000	0.000000	0.000000	201.0
3702	Incident	Minor	Boeing	Personal	0.00000	0.000000	0.000000	412.0
4149	Incident	Minor	Lockheed	Unknown	0.65642	0.283635	0.361814	588.0
4150	Incident	Minor	Boeing	Unknown	0.65642	0.283635	0.361814	588.0
...	...	...	...	...	...	...	...	...
87264	Non-Fatal	Unknown	BOEING	Unknown	0.00000	0.000000	0.000000	203.0
87493	Non-Fatal	Unknown	AIRBUS	Unknown	0.00000	0.000000	0.000000	290.0
87535	Non-Fatal	Minor	BOEING	Unknown	0.00000	0.000000	0.000000	368.0
87647	Non-Fatal	Unknown	BOEING	Unknown	0.00000	0.000000	0.000000	268.0
87661	Non-Fatal	Unknown	BOEING	Unknown	0.00000	0.000000	0.000000	201.0

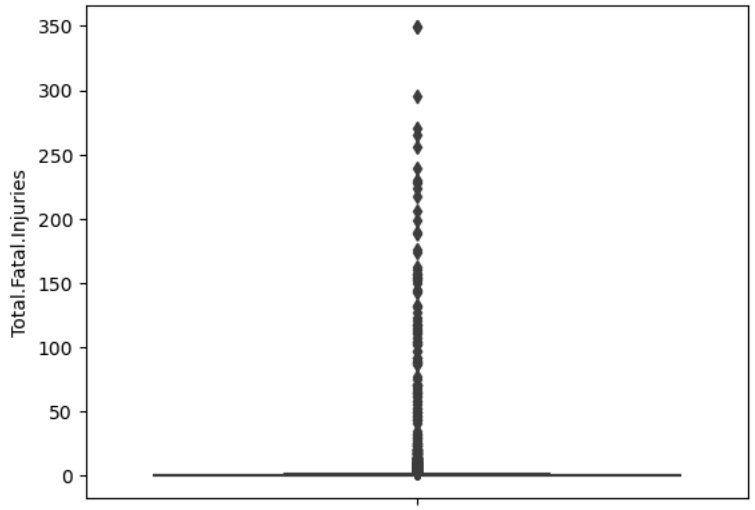
432 rows × 8 columns

In [36]:
```
#Remove the outlier by assigning the value to a new DataFrame
df5 = df4[df4["Total.Uninjured"] < max_TU]
```

In [37]:
```
#confirm removal of outlier
sns.boxplot(data=df5, y='Total.Uninjured');
```



In [38]:
```
sns.boxplot(data=df5, y='Total.Fatal.Injuries');
```



In [39]:
```
#Remove the outlier using the maximum quantile

#a. Get the max interquantile
max_TFI = df5['Total.Fatal.Injuries'].quantile(0.995)
max_TFI
```

Out[39]: 6.0

In [40]:
```
#check the outliers
df5[df5["Total.Fatal.Injuries"] > max_TFI]
```

Out[40]:

	Injury.Severity	Aircraft.damage	Make	Purpose.of.flight	Total.Fatal.Injuries	Total.Serious.Injuries	Total.Minor.Injuries	Total.Uninjured
25	Fatal	Destroyed	Cessna	Business	8.0	0.0	0.0	0.
165	Fatal	Destroyed	Robertson	Personal	7.0	0.0	0.0	0.
254	Fatal	Destroyed	Piper	Personal	8.0	0.0	0.0	0.
255	Fatal	Destroyed	Cessna	Personal	8.0	0.0	0.0	0.
334	Fatal	Destroyed	Piper	Business	8.0	0.0	0.0	0.
...	...	...	...	...	...	...	...	.
86466	Fatal	Unknown	CESSNA	Unknown	14.0	0.0	0.0	0.
86552	Fatal	Destroyed	BOEING	Unknown	132.0	0.0	0.0	0.
87405	Fatal	Substantial	DEHAVILLAND	Unknown	10.0	0.0	0.0	0.
87616	Fatal	Destroyed	BELL	Unknown	7.0	0.0	0.0	0.
87719	Fatal	Destroyed	PIPER	Unknown	8.0	0.0	0.0	0.

432 rows × 8 columns

In [41]:
```
#Remove the outlier by assigning the value to a new DataFrame
df6 = df5[df5["Total.Fatal.Injuries"] < max_TFI]
```

In [42]:
```
df6.describe()
```

Out[42]:

	Total.Fatal.Injuries	Total.Serious.Injuries	Total.Minor.Injuries	Total.Uninjured
count	85804.000000	85804.000000	85804.000000	85804.000000
mean	0.432732	0.229322	0.292856	3.650371
std	0.822342	0.503375	0.601103	15.936961
min	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.000000	0.000000	0.000000
50%	0.000000	0.000000	0.000000	1.000000
75%	0.656420	0.283635	0.361814	2.000000
max	5.000000	3.000000	4.000000	190.000000

In [67]:
```
df6['Make'] = df6['Make'].str.title() #to Capitalize the first word in Make Column
```

```
C:\Users\Sam\AppData\Local\Temp\ipykernel_1852\2754985339.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  df6['Make'] = df6['Make'].str.title() #to Capitalize the first word in Make Column
```

In [68]:
```
df6.to_csv('clean_aviation.csv', index=False)
```

1 Exploratory Data Analysis

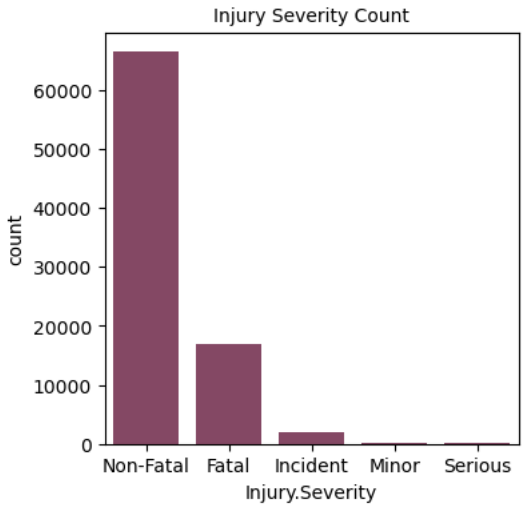
In [69]:
```
#Load the clean_aviation dataset
data = pd.read_csv('clean_aviation.csv')
data.head(20)
```

Out[69]:

	Injury.Severity	Aircraft.damage	Make	Purpose.of.flight	Total.Fatal.Injuries	Total.Serious.Injuries	Total.Minor.Injuries	Total.Uninjured
0	Fatal	Destroyed	Stinson	Personal	2.00000	0.000000	0.000000	0.000000
1	Fatal	Destroyed	Piper	Personal	4.00000	0.000000	0.000000	0.000000
2	Fatal	Destroyed	Cessna	Personal	3.00000	0.283635	0.361814	5.391191
3	Fatal	Destroyed	Rockwell	Personal	2.00000	0.000000	0.000000	0.000000
4	Fatal	Destroyed	Cessna	Personal	1.00000	2.000000	0.361814	0.000000
5	Non-Fatal	Substantial	Mcdonnell Douglas	Unknown	0.65642	0.283635	1.000000	44.000000
6	Fatal	Destroyed	Cessna	Personal	4.00000	0.000000	0.000000	0.000000
7	Non-Fatal	Substantial	Cessna	Personal	0.00000	0.000000	0.000000	2.000000
8	Non-Fatal	Substantial	Cessna	Business	0.00000	0.000000	0.000000	2.000000
9	Non-Fatal	Substantial	North American	Personal	0.00000	0.000000	3.000000	0.000000
10	Non-Fatal	Substantial	Piper	Personal	0.00000	0.000000	0.000000	1.000000
11	Non-Fatal	Substantial	Beech	Personal	0.00000	0.000000	0.000000	1.000000
12	Non-Fatal	Destroyed	Bellanca	Personal	0.00000	0.000000	1.000000	0.000000
13	Fatal	Destroyed	Cessna	Personal	1.00000	0.000000	0.000000	0.000000
14	Fatal	Destroyed	Navion	Personal	1.00000	0.000000	0.000000	0.000000
15	Fatal	Destroyed	Beech	Personal	2.00000	0.000000	0.000000	0.000000
16	Non-Fatal	Destroyed	Enstrom	Personal	0.00000	0.000000	0.000000	1.000000
17	Fatal	Destroyed	Cessna	Personal	3.00000	0.000000	0.000000	0.000000
18	Non-Fatal	Substantial	Cessna	Personal	0.00000	0.000000	0.000000	1.000000
19	Non-Fatal	Substantial	Smith	Personal	0.00000	0.000000	0.000000	2.000000

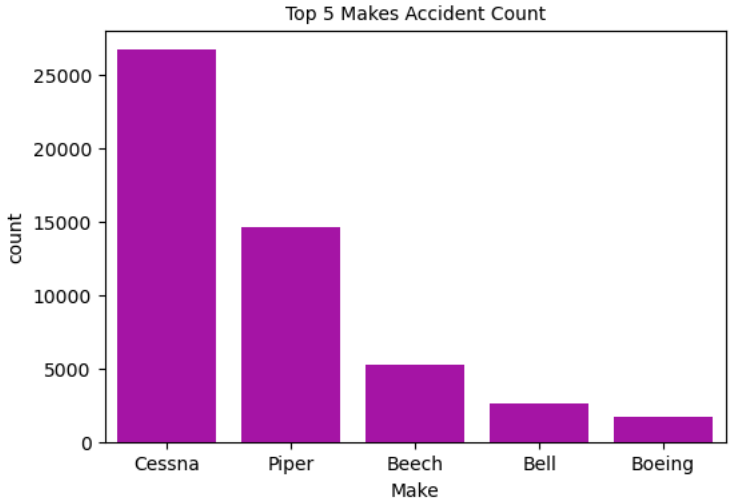
1.1 1. Univariate analysis

In [77]:
```
#count plot-Injury Severity count
plt.figure(figsize=(4,4))
sns.countplot(x='Injury.Severity', order= data['Injury.Severity'].value_counts().index, color='#8E3E63', data=data)
plt.title('Injury Severity Count', fontsize=10);
```



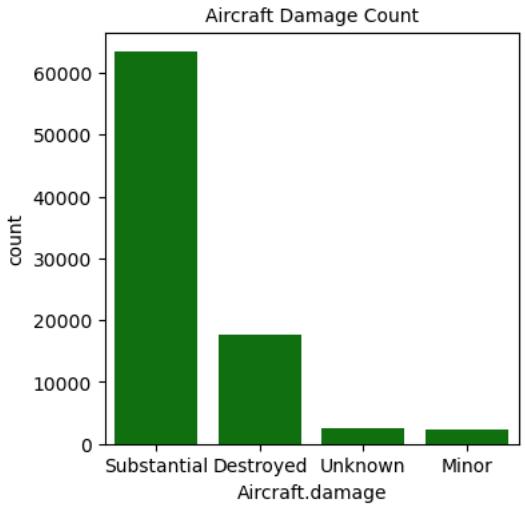
Observation: The injury severity count is very high for Non-Fatal and low for Minor and Serious

In [98]:
```
#count plot-frequency of accidents for the top 5 Makes
top_5_makes = data['Make'].value_counts().head(5).index
filtered_data = data[data['Make'].isin(top_5_makes)]

plt.figure(figsize=(6,4))
sns.countplot(x='Make', data=filtered_data, order=top_5_makes, color='m')
plt.title('Top 5 Makes Accident Count', fontsize=10);
```



Observation: the chart summarises the frequency of accidents for the top five aircraft ranked from Cessna being the highest and Boeing the lowest of the top five

In [99]:
```
#count plot-Aircraft Damage count
plt.figure(figsize=(4,4))
sns.countplot(x='Aircraft.damage', order= data['Aircraft.damage'].value_counts().index, color='g', data=data)
plt.title('Aircraft Damage Count', fontsize=10);
```

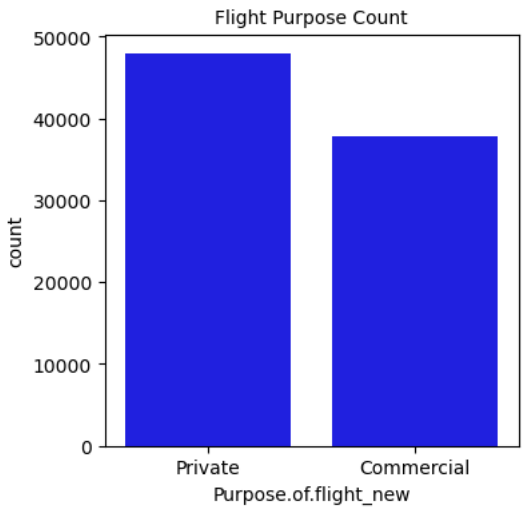


Observation: from the aviation dataset, the number of accidents with substantial damage to the aircraft is high and with minor and unknown being the lowest

In [111]:
```
#Group Purpose of flight into Commercial and Private
def personal(purpose):
    if purpose == 'Personal':
        return 'Private'

    else:
        return 'Commercial'
```

In [113]:
```
data['Purpose.of.flight_new'] = df['Purpose.of.flight'].apply(personal)
```

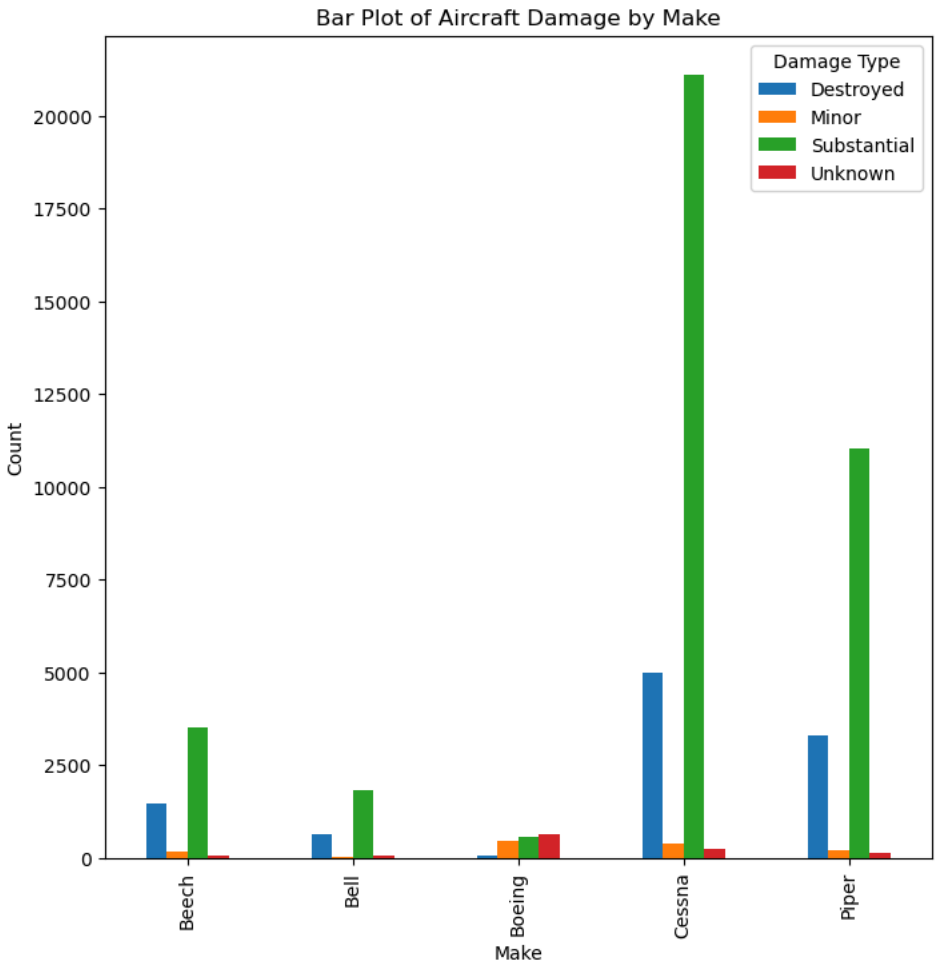
In [159]:
```
#count plot-Aircraft Damage count
plt.figure(figsize=(4,4))
sns.countplot(x='Purpose.of.flight_new', order= data['Purpose.of.flight_new'].value_counts().index, color='b', data=data)
plt.title('Flight Purpose Count', fontsize=10);
```



I summarised the Purpose of flight column into two categories i.e Private and Commercial Observation: the aircrafts in the aviation dataset were mainly used for private purposes followed closely by commercial

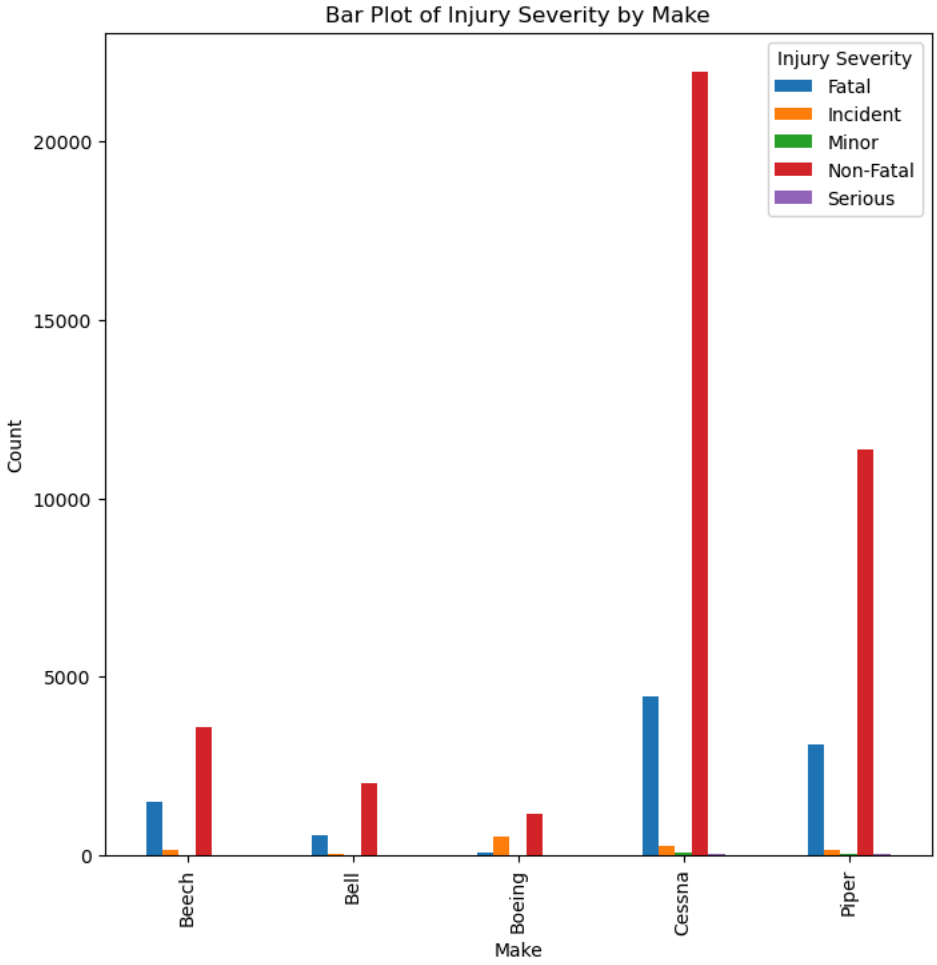
1.2 2. Bivariate Analysis

In [119]:
```
#to compare the make vs aircraft damage
make_damage = filtered_data.groupby(['Aircraft.damage','Make']).size().reset_index().pivot(columns='Aircraft.damage',index='Make'

make_damage.plot(kind='bar', figsize=(8, 8))
plt.title('Bar Plot of Aircraft Damage by Make')
plt.xlabel('Make')
plt.ylabel('Count')
plt.legend(title='Damage Type');
```



Observtion: All makes reported high substantial & destroyed damages except for Boeing

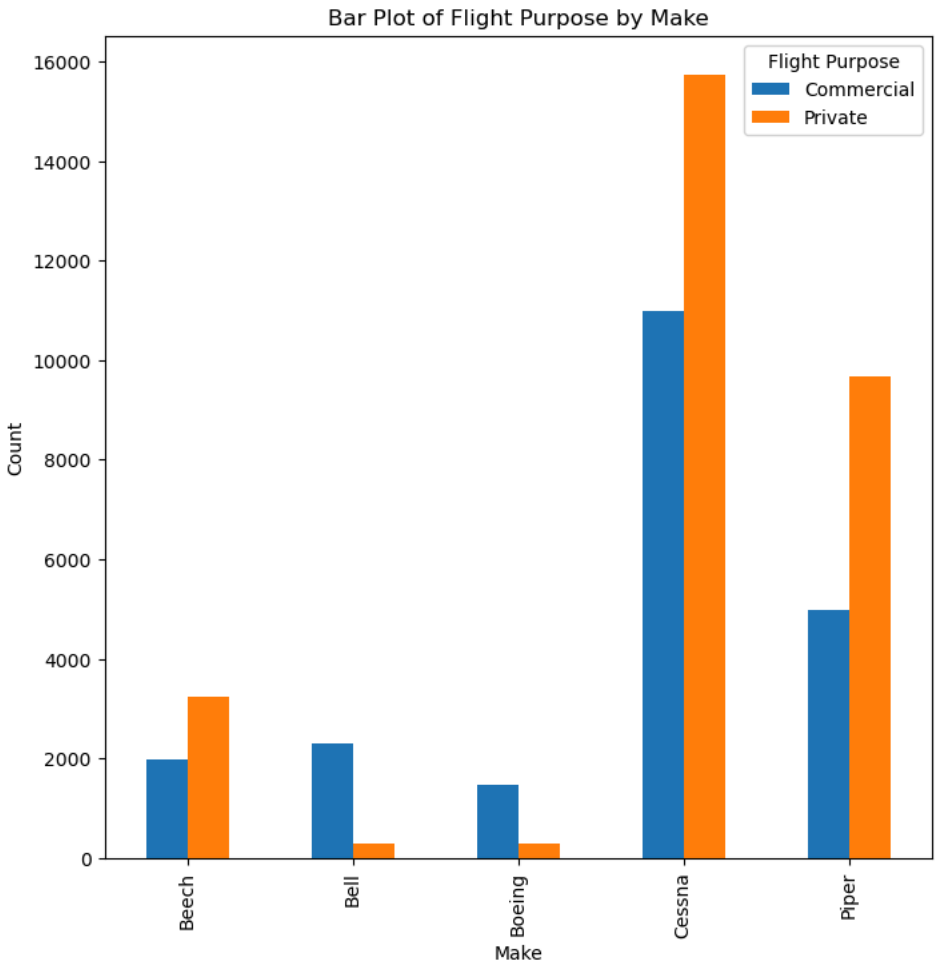
In [131]:
```
#To compare the make vs the injury severity
make_injury = filtered_data.groupby(['Injury.Severity','Make']).size().reset_index().pivot(columns='Injury.Severity',index='Make'

make_injury.plot(kind='bar', figsize=(8, 8))
plt.title('Bar Plot of Injury Severity by Make')
plt.xlabel('Make')
plt.ylabel('Count')
plt.legend(title='Injury Severity');
```



Observation: the injury severity - Non-fatal was highest for all the makes followed by Fatal. Cessna which a higher accident rate has relatively lower Fatal rates when compared to Piper

In [133]:
```
filtered_data['Purpose.of.flight_new'] = filtered_data['Purpose.of.flight'].apply(personal)
make_purpose = filtered_data.groupby(['Purpose.of.flight_new','Make']).size().reset_index().pivot(columns='Purpose.of.flight_new'

make_purpose.plot(kind='bar', figsize=(8, 8))
plt.title('Bar Plot of Flight Purpose by Make')
plt.xlabel('Make')
plt.ylabel('Count')
plt.legend(title='Flight Purpose');
```

```
C:\Users\Sam\AppData\Local\Temp\ipykernel_1852\3350530157.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-ve
rsus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  filtered_data['Purpose.of.flight_new'] = filtered_data['Purpose.of.flight'].apply(personal)
```



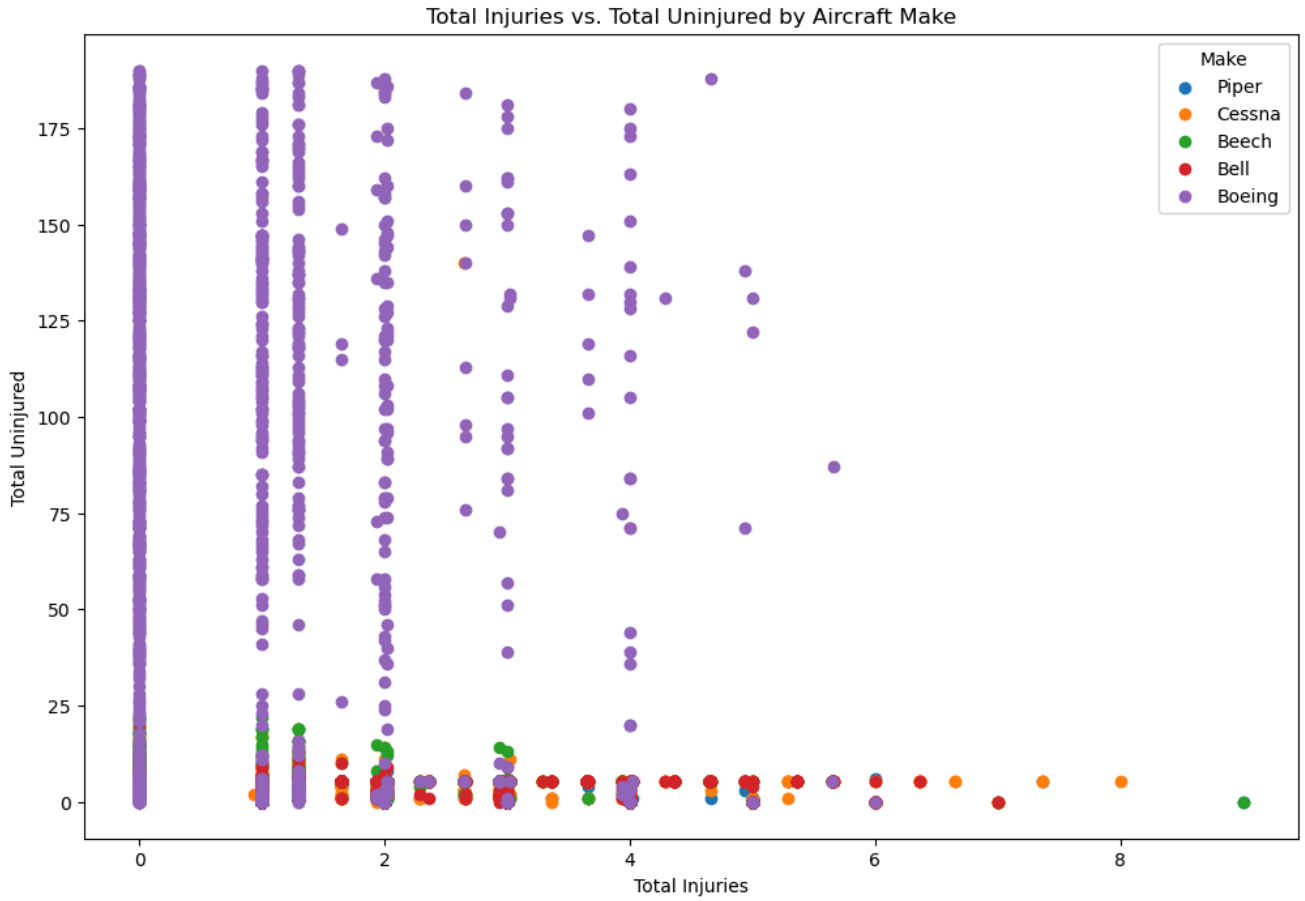
Observation: Cessna, Piper and Beach makes were mainly used for private purposes while Bell and Boeing were mainly used for commercial purpose

1.3 3. Multivariate Analayis

In [141]:
```
filtered_data['Total.Injuries'] = filtered_data['Total.Fatal.Injuries'] + filtered_data['Total.Serious.Injuries'] + filtered_data
filtered_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 50953 entries, 1 to 85803
Data columns (total 10 columns):
 #   Column                 Non-Null Count  Dtype  
---  ------                 --------------  -----  
 0   Injury.Severity        50953 non-null  object 
 1   Aircraft.damage        50953 non-null  object 
 2   Make                   50953 non-null  object 
 3   Purpose.of.flight      50953 non-null  object 
 4   Total.Fatal.Injuries   50953 non-null  float64
 5   Total.Serious.Injuries 50953 non-null  float64
 6   Total.Minor.Injuries   50953 non-null  float64
 7   Total.Uninjured        50953 non-null  float64
 8   Purpose.of.flight_new  50953 non-null  object 
 9   Total.Injuries         50953 non-null  float64
dtypes: float64(5), object(5)
memory usage: 4.3+ MB

C:\Users\Sam\AppData\Local\Temp\ipykernel_1852\4089459898.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-ve
rsus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  filtered_data['Total.Injuries'] = filtered_data['Total.Fatal.Injuries'] + filtered_data['Total.Serious.Injuries'] + filtered_
data['Total.Minor.Injuries']
```

In [148]:
```
#To check for relationship between make, total injuries and total uninjured
plt.figure(figsize=(12, 8))

# Scatter plot with different colors for each 'Make'
for make in filtered_data['Make'].unique():
    subset = filtered_data[filtered_data['Make'] == make]
    plt.scatter(subset['Total.Injuries'], subset['Total.Uninjured'], label=make)

# Add labels and title
plt.title('Total Injuries vs. Total Uninjured by Aircraft Make')
plt.xlabel('Total Injuries')
plt.ylabel('Total Uninjured')
plt.legend(title='Make');
```



Observation: Boeing has the lowest total injuries and highest total for uninjured Cessna, Piper, Bell and Beach makes have high total injuries and low numbers for total uninjured.

1.4 Conclusion

Based on the above observations I would recommend purchase of Boeing aircraft. We have that Boeing has registered low total injuries, high totals for uninjured, the injury severity and aircraft damage was relatively low for all categories when compared to other makes.

In [ ]: