Final Project Submission

Please fill out:

- Student name: Monicah Wanjiru
- Student pace: part time
- Scheduled project review date/time: 8/9/2024
- Instructor name: William Okomba
- Blog post URL:

BUSINESS UNDERSTANDING

PROJECT OVERVIEW

Risk is assesed by the probability of adverse results emanating from a hazard. it is the likelyhold that the hazard will cause harm. some harzards in the aviaton industry include and not limited to adverse weather condition, runway hazardous condition, equipment malfuction, fuel contamination, poor maintance practices among others. once the hazardsoccur they could lead to the risk of loss of life, damage of properties and damage of the aircraft etc. Risk in the aviation industry can be looked at as those that have a low probabilty of occurence but the imapct is high.

BUSINESS PROBLEM

The business problem is identfy the aircraft that has the lowest risk to help the company invest in it so as it can diversify its portfolio.

PROJECT OBJECTIVE

The objective of the project is to evaluate the existing data on risks in the aviation industry and how they have previously affected different makes/models of aircrafts. The output of this analysis will help the new head of aviation division to settle on which aircrafts the organization can purchase.This will expand/diversfy the oragnizaton portfolio whichwill in return increase the revenue of the company.

DATA SOURCE

The project utilized data obtained from Kaggle its from National Transportation Safety Board that includes aviation accident data from 1962 to 2023 about civil aviation accidents and selected incidents in the United States and international waters.

Importing libraries and reading our csv dataset

```
# importing libaries
import pandas as pd
import numpy as np
import seaborn as sns
```

```
import matplotlib.pyplot as plt
%matplotlib inline

# reading our dataset
aviation = pd.read_csv("AviationData.csv", encoding =('ISO-8859-1'),
low_memory=False)
aviation

             Event.Id Investigation.Type Accident.Number
Event.Date  \
0      20001218X45444           Accident      SEA87LA080  1948-10-24

1      20001218X45447           Accident      LAX94LA336  1962-07-19

2      20061025X01555           Accident      NYC07LA005  1974-08-30

3      20001218X45448           Accident      LAX96LA321  1977-06-19

4      20041105X01764           Accident      CHI79FA064  1979-08-02

...               ...                ...             ...          ...

88884  20221227106491           Accident      ERA23LA093  2022-12-26

88885  20221227106494           Accident      ERA23LA095  2022-12-26

88886  20221227106497           Accident      WPR23LA075  2022-12-26

88887  20221227106498           Accident      WPR23LA076  2022-12-26

88888  20221230106513           Accident      ERA23LA097  2022-12-29


              Location        Country   Latitude   Longitude
Airport.Code  \
0      MOOSE CREEK, ID  United States        NaN         NaN
NaN
1       BRIDGEPORT, CA  United States        NaN         NaN
NaN
2        Saltville, VA  United States  36.922223  -81.878056
NaN
3            EUREKA, CA  United States        NaN         NaN
NaN
4            Canton, OH  United States        NaN         NaN
NaN
...                ...            ...        ...         ...
...
88884    Annapolis, MD  United States        NaN         NaN
NaN
88885      Hampton, NH  United States        NaN         NaN
NaN
```

```
88886       Payson, AZ  United States    341525N    1112021W
PAN
88887       Morgan, UT  United States        NaN         NaN
NaN
88888       Athens, GA  United States        NaN         NaN
NaN

      Airport.Name  ... Purpose.of.flight         Air.carrier  \
0              NaN  ...          Personal                 NaN
1              NaN  ...          Personal                 NaN
2              NaN  ...          Personal                 NaN
3              NaN  ...          Personal                 NaN
4              NaN  ...          Personal                 NaN
...            ...  ...               ...                 ...
88884          NaN  ...          Personal                 NaN
88885          NaN  ...               NaN                 NaN
88886       PAYSON  ...          Personal                 NaN
88887          NaN  ...          Personal  MC CESSNA 210N LLC
88888          NaN  ...          Personal                 NaN

      Total.Fatal.Injuries Total.Serious.Injuries Total.Minor.Injuries
\
0                      2.0                    0.0                  0.0

1                      4.0                    0.0                  0.0

2                      3.0                    NaN                  NaN

3                      2.0                    0.0                  0.0

4                      1.0                    2.0                  NaN

...                    ...                    ...                  ...

88884                  0.0                    1.0                  0.0

88885                  0.0                    0.0                  0.0

88886                  0.0                    0.0                  0.0

88887                  0.0                    0.0                  0.0

88888                  0.0                    1.0                  0.0


      Total.Uninjured Weather.Condition  Broad.phase.of.flight  \
0                 0.0               UNK                 Cruise
1                 0.0               UNK                Unknown
2                 NaN               IMC                 Cruise
3                 0.0               IMC                 Cruise
4                 0.0               VMC               Approach
```

```
...                  ...                  ...                        ...
88884                0.0                  NaN                        NaN
88885                0.0                  NaN                        NaN
88886                1.0                  VMC                        NaN
88887                0.0                  NaN                        NaN
88888                1.0                  NaN                        NaN

        Report.Status Publication.Date
0       Probable Cause              NaN
1       Probable Cause       19-09-1996
2       Probable Cause       26-02-2007
3       Probable Cause       12-09-2000
4       Probable Cause       16-04-1980
...                ...              ...
88884              NaN       29-12-2022
88885              NaN              NaN
88886              NaN       27-12-2022
88887              NaN              NaN
88888              NaN       30-12-2022

[88889 rows x 31 columns]
```

Data Understanding

Our datataset is from kaggle The NTSB aviation accident database contains information from 1962 and later about civil aviation accidents and selected incidents within the United States, its territories and possessions, and in international waters.

```
#  checking the shape of the dataset
aviation.shape
```

```
(88889, 31)
```

```
# checking the lenghth the length
len(aviation)
```

```
88889
```

```
# checking the type of the dataset
type(aviation)
```

```
pandas.core.frame.DataFrame
```

```
# prints the first 3 rows of our dataset
aviation.head(3)
```

```
         Event.Id Investigation.Type Accident.Number  Event.Date  \
0  20001218X45444           Accident      SEA87LA080  1948-10-24
1  20001218X45447           Accident      LAX94LA336  1962-07-19
```

```
2  20061025X01555          Accident      NYC07LA005  1974-08-30

          Location        Country   Latitude   Longitude Airport.Code
\
0  MOOSE CREEK, ID  United States        NaN         NaN          NaN

1   BRIDGEPORT, CA  United States        NaN         NaN          NaN

2    Saltville, VA  United States  36.922223  -81.878056          NaN


  Airport.Name  ... Purpose.of.flight Air.carrier Total.Fatal.Injuries
\
0          NaN  ...          Personal         NaN                  2.0

1          NaN  ...          Personal         NaN                  4.0

2          NaN  ...          Personal         NaN                  3.0


  Total.Serious.Injuries Total.Minor.Injuries Total.Uninjured  \
0                    0.0                  0.0             0.0
1                    0.0                  0.0             0.0
2                    NaN                  NaN             NaN

  Weather.Condition  Broad.phase.of.flight   Report.Status
Publication.Date
0               UNK                 Cruise  Probable Cause
NaN
1               UNK                Unknown  Probable Cause       19-
09-1996
2               IMC                 Cruise  Probable Cause       26-
02-2007

[3 rows x 31 columns]
```

```
# prints the last 3 rows of our dataset
aviation.tail(3)
```

```
             Event.Id Investigation.Type Accident.Number
Event.Date  \
88886  20221227106497          Accident      WPR23LA075  2022-12-26

88887  20221227106498          Accident      WPR23LA076  2022-12-26

88888  20221230106513          Accident      ERA23LA097  2022-12-29


         Location        Country Latitude Longitude Airport.Code
Airport.Name  \
88886  Payson, AZ  United States  341525N  1112021W          PAN
```

```
PAYSON
88887  Morgan, UT  United States      NaN       NaN          NaN
NaN
88888  Athens, GA  United States      NaN       NaN          NaN
NaN

       ... Purpose.of.flight         Air.carrier Total.Fatal.Injuries
\
88886  ...          Personal                 NaN                  0.0

88887  ...          Personal  MC CESSNA 210N LLC                  0.0

88888  ...          Personal                 NaN                  0.0


      Total.Serious.Injuries Total.Minor.Injuries Total.Uninjured  \
88886                    0.0                  0.0             1.0
88887                    0.0                  0.0             0.0
88888                    1.0                  0.0             1.0

      Weather.Condition  Broad.phase.of.flight Report.Status
Publication.Date
88886               VMC                    NaN           NaN       27-
12-2022
88887               NaN                    NaN           NaN
NaN
88888               NaN                    NaN           NaN       30-
12-2022

[3 rows x 31 columns]
```

```
# shows the descriptive statistics of the dataset
aviation.describe()
```

```
       Number.of.Engines  Total.Fatal.Injuries  Total.Serious.Injuries
\
count       82805.000000          77488.000000            76379.000000

mean            1.146585              0.647855                0.279881

std             0.446510              5.485960                1.544084

min             0.000000              0.000000                0.000000

25%             1.000000              0.000000                0.000000

50%             1.000000              0.000000                0.000000

75%             1.000000              0.000000                0.000000

max             8.000000            349.000000              161.000000
```

```
       Total.Minor.Injuries  Total.Uninjured
count          76956.000000     82977.000000
mean               0.357061         5.325440
std                2.235625        27.913634
min                0.000000         0.000000
25%                0.000000         0.000000
50%                0.000000         1.000000
75%                0.000000         2.000000
max              380.000000       699.000000
```

```
aviation.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
```

```
# checking columns of the dataset
aviation.columns
```

```
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')
```

Some of the columns in the data set will not be needed for the anaysis in the business problem at hand, therefore they will be dropped. below are relevant columns we will use in our analysis.

1. Injury.Severity: Indicates the severity of injuries in each accident (e.g., Fatal, Serious, Minor, Uninjured).
2. Aircraft.damage: Provides information about the extent of damage to the aircraft.
3. Aircraft.Category: Specifies the category of the aircraft (e.g., commercial, private).
4. Make and Model: Identifies the manufacturer and model of the aircraft.
5. Number.of.Engines:
6. Engine.Type:
7. FAR.Description:
8. Total.Fatal.Injuries, Total.Serious.Injuries, Total.Minor.Injuries: Quantifies the number of injuries in each category.
9. Weather.Condition: Provides context on weather conditions during the accident.
10. Broad.phase.of.flight: Helps to identify during which phase of flight accidents occurred (e.g., takeoff, cruising, landing).

```
irelevant_col = ['Event.Id', 'Investigation.Type', 'Accident.Number',
'Event.Date', 'Location',
                 'Country', 'Latitude', 'Longitude', 'Airport.Code',
'Airport.Name', 'Registration.Number',
                  'Amateur.Built', 'Schedule',
'Air.carrier','Report.Status']

aviation1 = aviation.drop(columns=irelevant_col)
aviation1.head(3)
```

```
  Injury.Severity Aircraft.damage Aircraft.Category     Make     Model
\
0        Fatal(2)       Destroyed               NaN  Stinson     108-3

1        Fatal(4)       Destroyed               NaN    Piper  PA24-180

2        Fatal(3)       Destroyed               NaN   Cessna      172M


   Number.of.Engines    Engine.Type FAR.Description Purpose.of.flight
\
0                1.0  Reciprocating             NaN          Personal

1                1.0  Reciprocating             NaN          Personal

2                1.0  Reciprocating             NaN          Personal


   Total.Fatal.Injuries  Total.Serious.Injuries  Total.Minor.Injuries
\
0                   2.0                     0.0                   0.0

1                   4.0                     0.0                   0.0

2                   3.0                     NaN                   NaN


   Total.Uninjured Weather.Condition Broad.phase.of.flight
Publication.Date
0              0.0               UNK                Cruise
NaN
1              0.0               UNK               Unknown       19-
09-1996
2              NaN               IMC                Cruise       26-
02-2007
```

```
aviation1.columns
```

```
Index(['Injury.Severity', 'Aircraft.damage', 'Aircraft.Category',
'Make',
       'Model', 'Number.of.Engines', 'Engine.Type', 'FAR.Description',
       'Purpose.of.flight', 'Total.Fatal.Injuries',
'Total.Serious.Injuries',
       'Total.Minor.Injuries', 'Total.Uninjured', 'Weather.Condition',
       'Broad.phase.of.flight', 'Publication.Date'],
      dtype='object')
```

Data Preparation

1. check for missing values either to fill, or drop them
2. check for duplicates, drop them and keep first

3. check for outliers and drop them

```
# checking missing values
aviation1.isna().sum()

Injury.Severity             1000
Aircraft.damage             3194
Aircraft.Category          56602
Make                          63
Model                         92
Number.of.Engines           6084
Engine.Type                 7096
FAR.Description            56866
Purpose.of.flight           6192
Total.Fatal.Injuries       11401
Total.Serious.Injuries     12510
Total.Minor.Injuries       11933
Total.Uninjured             5912
Weather.Condition           4492
Broad.phase.of.flight      27165
Publication.Date           13771
dtype: int64

aviation1.shape

(88889, 16)

aviation1.dtypes

Injury.Severity             object
Aircraft.damage             object
Aircraft.Category           object
Make                        object
Model                       object
Number.of.Engines          float64
Engine.Type                 object
FAR.Description             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
Publication.Date            object
dtype: object
```

Drop missing values from object data types columns

```
# droping missing in object datatype columns
aviation2 = aviation1.dropna(subset=['Injury.Severity',
```

```
'Aircraft.damage', 'Aircraft.Category', 'Make',
                                     'Model', 'Engine.Type',
'FAR.Description', 'Broad.phase.of.flight',
                                     'Weather.Condition',
'Purpose.of.flight'])
```

Explanatory data analysis

using fillna to fill the numerical columns we will use the mean

```
import warnings
warnings.filterwarnings('ignore')

engine_mean = aviation2["Number.of.Engines"].mean()
aviation2['Number.of.Engines'] =
aviation2['Number.of.Engines'].fillna(engine_mean)

tot_inj_mean = aviation2["Total.Fatal.Injuries"].mean()
aviation2['Total.Fatal.Injuries'] =
aviation2['Total.Fatal.Injuries'].fillna(engine_mean)

Total_Se_Inj = aviation2["Total.Serious.Injuries"].mean()
aviation2['Total.Serious.Injuries'] =
aviation2['Total.Serious.Injuries'].fillna(Total_Se_Inj)

Total_Mino_inj = aviation2["Total.Minor.Injuries"].mean()
aviation2['Total.Minor.Injuries'] =
aviation2['Total.Minor.Injuries'].fillna(Total_Mino_inj)

total_unj = aviation2["Total.Uninjured"].mean()
aviation2['Total.Uninjured'] =
aviation2['Total.Uninjured'].fillna(total_unj)

aviation2.isna().sum()
```

```
Injury.Severity           0
Aircraft.damage           0
Aircraft.Category         0
Make                      0
Model                     0
Number.of.Engines         0
Engine.Type               0
FAR.Description           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
```

```
Publication.Date           0
dtype: int64
```

```
aviation2.shape
```

```
(6975, 16)
```

checking for duplicates

```
# checking for duplicates
aviation2.duplicated().sum()
```

```
63
```

```
# droping duplicates
aviation3 = aviation2.drop_duplicates()
```

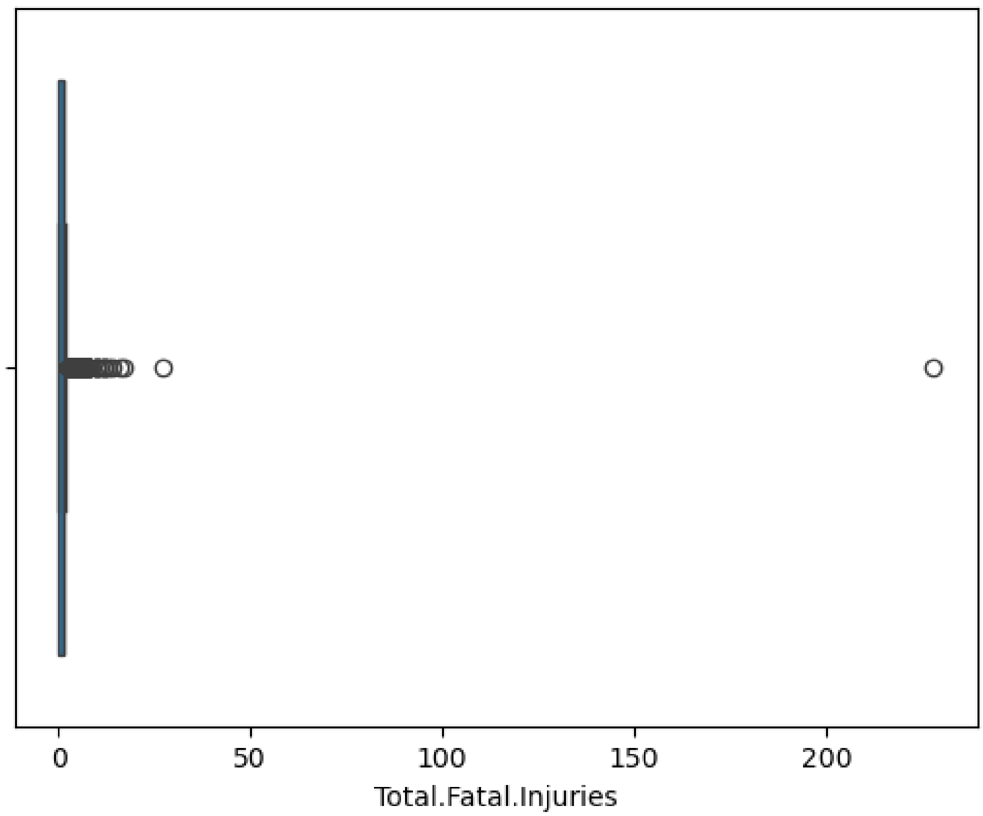
```
# confiriming if duplicates were dropped
aviation3.duplicated().sum()
```

```
0
```

Checking for outliers

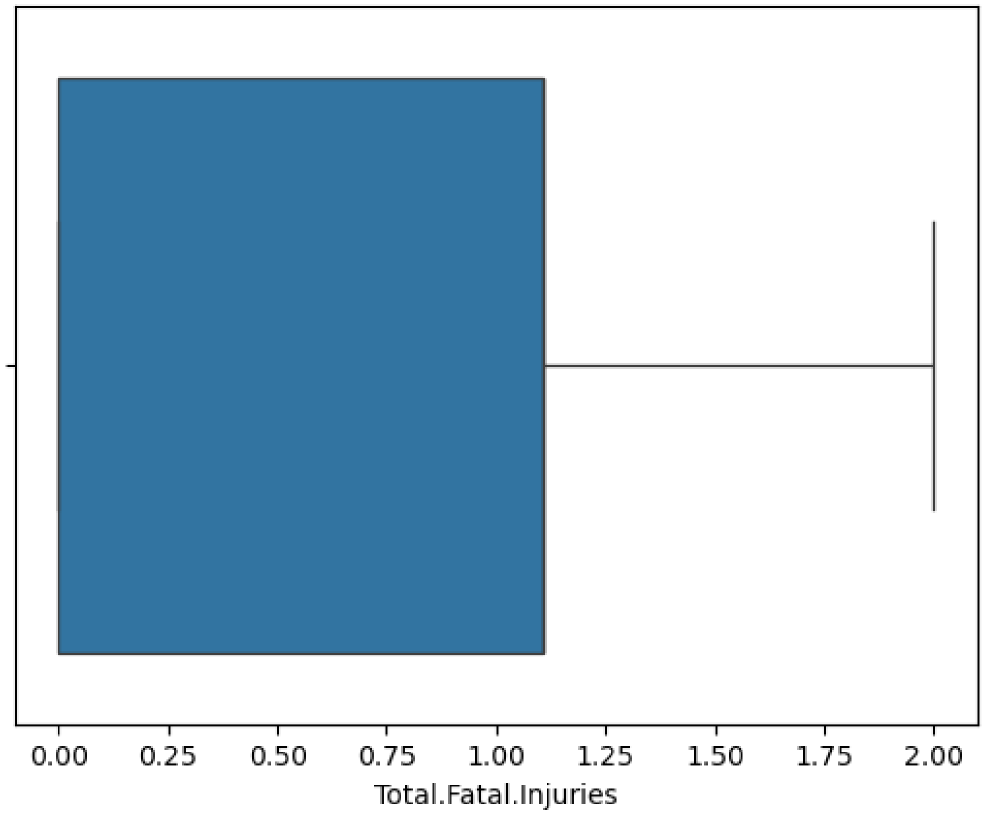
```
sns.boxplot(x=aviation3["Total.Fatal.Injuries"])
```

```
<Axes: xlabel='Total.Fatal.Injuries'>
```



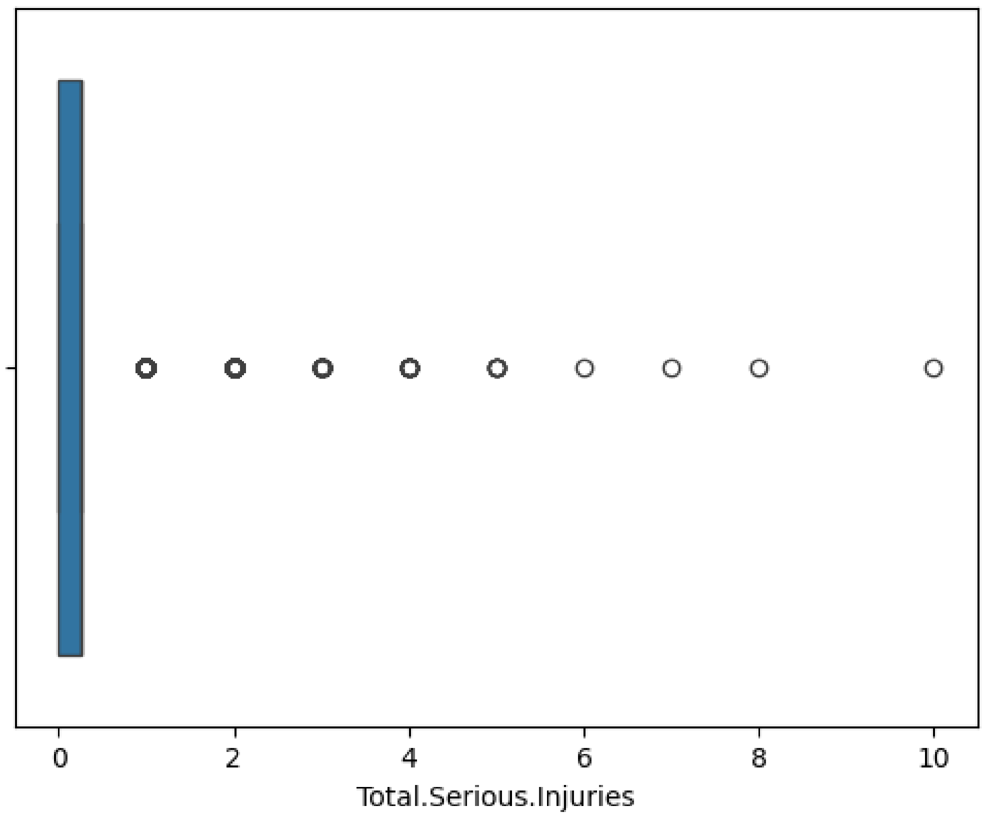

```
# i will use the interquatile range to calculate and filter out
outliers
# Calculate the IQR
Q1 = aviation3['Total.Fatal.Injuries'].quantile(0.25)
Q3 = aviation3['Total.Fatal.Injuries'].quantile(0.75)
IQR = Q3 - Q1

# Define the bounds for outliers
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

# Filter out outliers
aviation4 = aviation3[(aviation3['Total.Fatal.Injuries'] >=
lower_bound) & (aviation3['Total.Fatal.Injuries'] <= upper_bound)]
aviation4.reset_index(drop=True, inplace=True)

sns.boxplot(x=aviation4["Total.Fatal.Injuries"])
```

```
<Axes: xlabel='Total.Fatal.Injuries'>
```



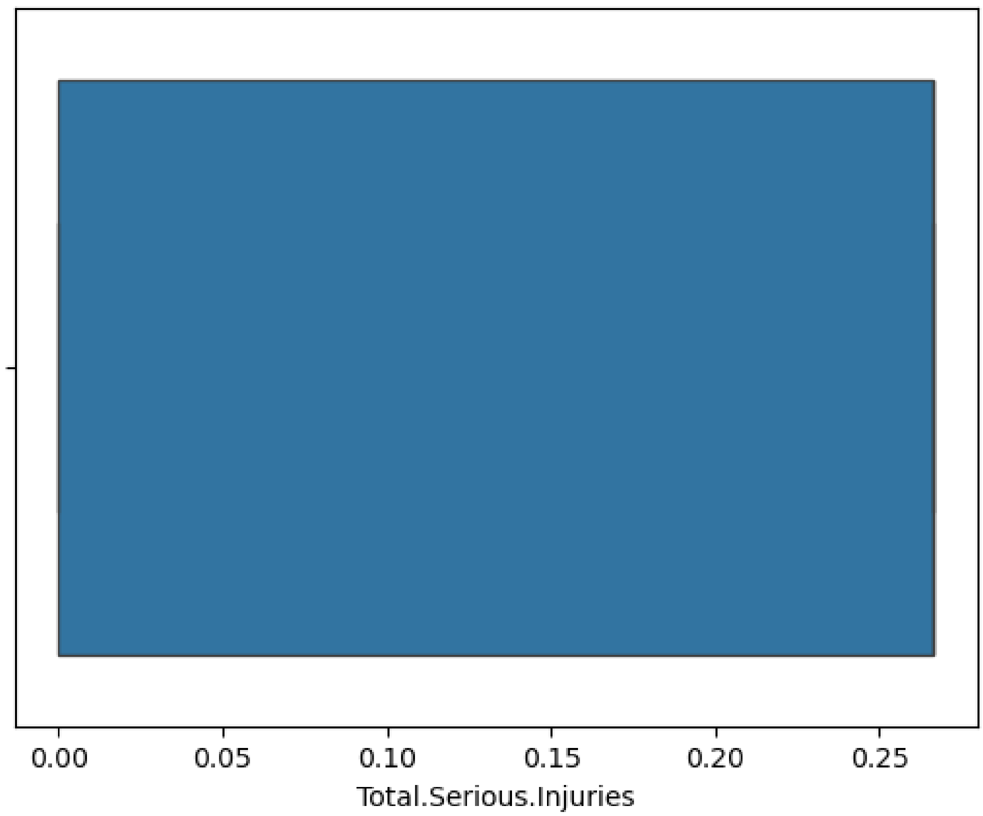

```
sns.boxplot(x=aviation4["Total.Serious.Injuries"])
```

```
<Axes: xlabel='Total.Serious.Injuries'>
```



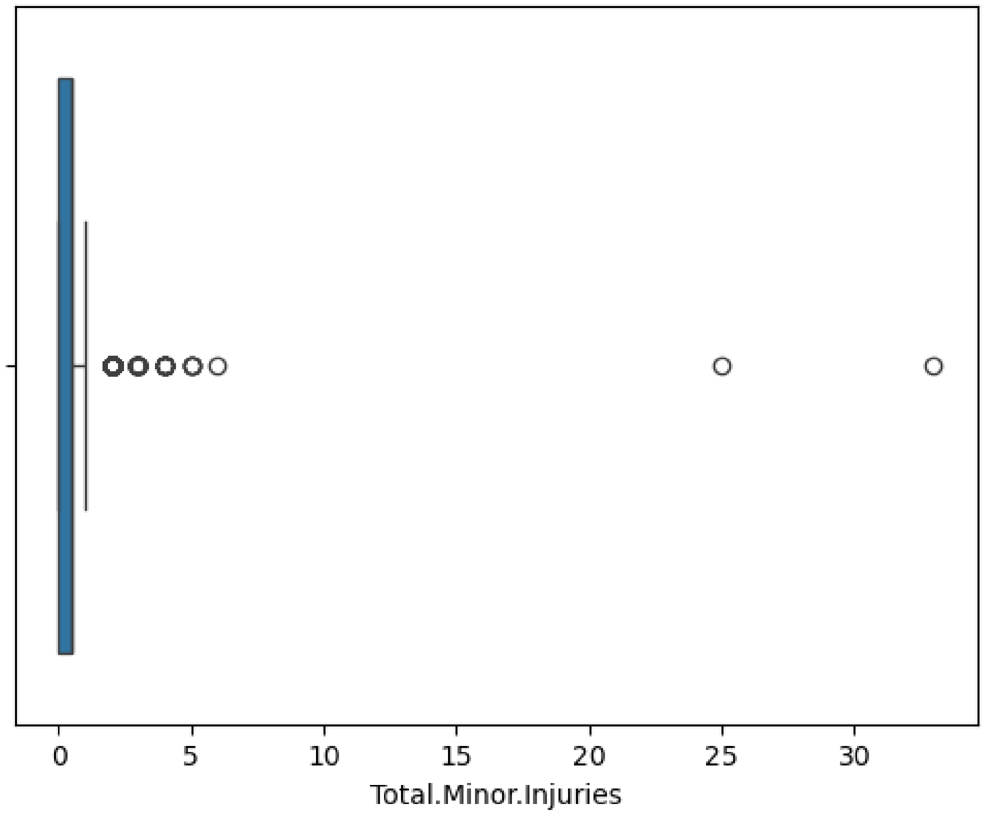

```
#### i will use the interquatile range to calculate and filter out
outliers
# Calculate the IQR
Q1 = aviation4['Total.Serious.Injuries'].quantile(0.25)
Q3 = aviation4['Total.Serious.Injuries'].quantile(0.75)
IQR = Q3 - Q1

# Define the bounds for outliers
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

# Filter out outliers
aviation5 = aviation4[(aviation4['Total.Serious.Injuries'] >=
lower_bound) & (aviation4['Total.Serious.Injuries'] <= upper_bound)]
aviation5.reset_index(drop=True, inplace=True)

sns.boxplot(x=aviation5["Total.Serious.Injuries"])
```

```
<Axes: xlabel='Total.Serious.Injuries'>
```



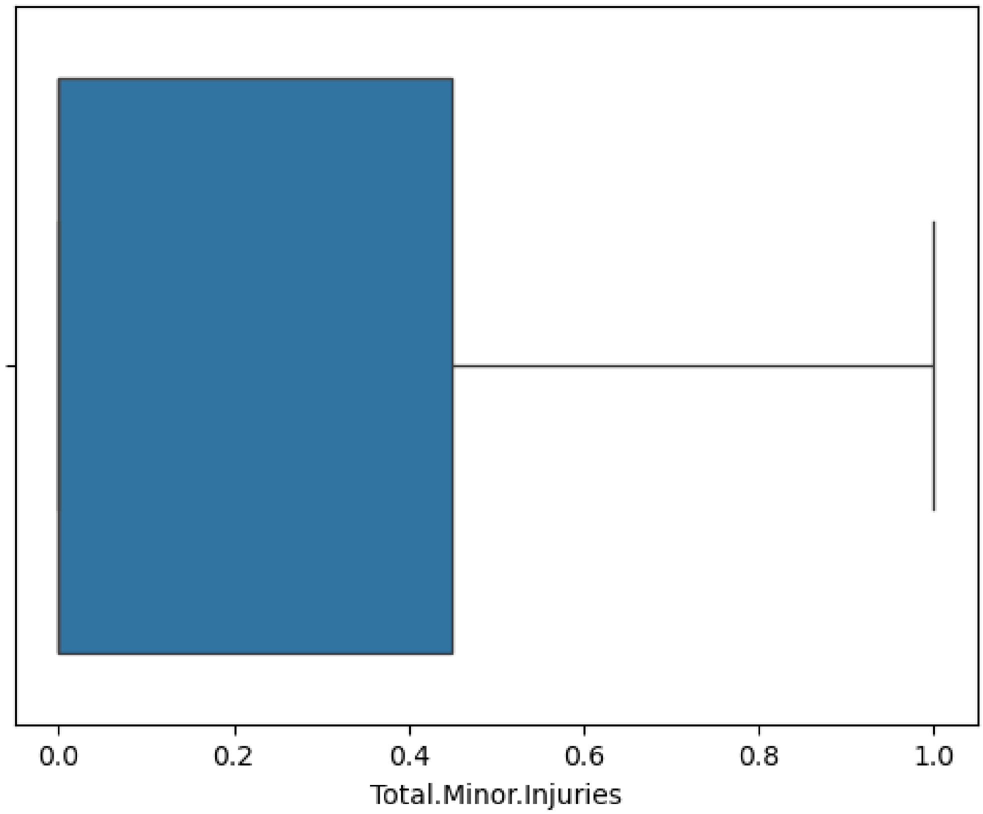

```
sns.boxplot(x=aviation5["Total.Minor.Injuries"])
```

```
<Axes: xlabel='Total.Minor.Injuries'>
```



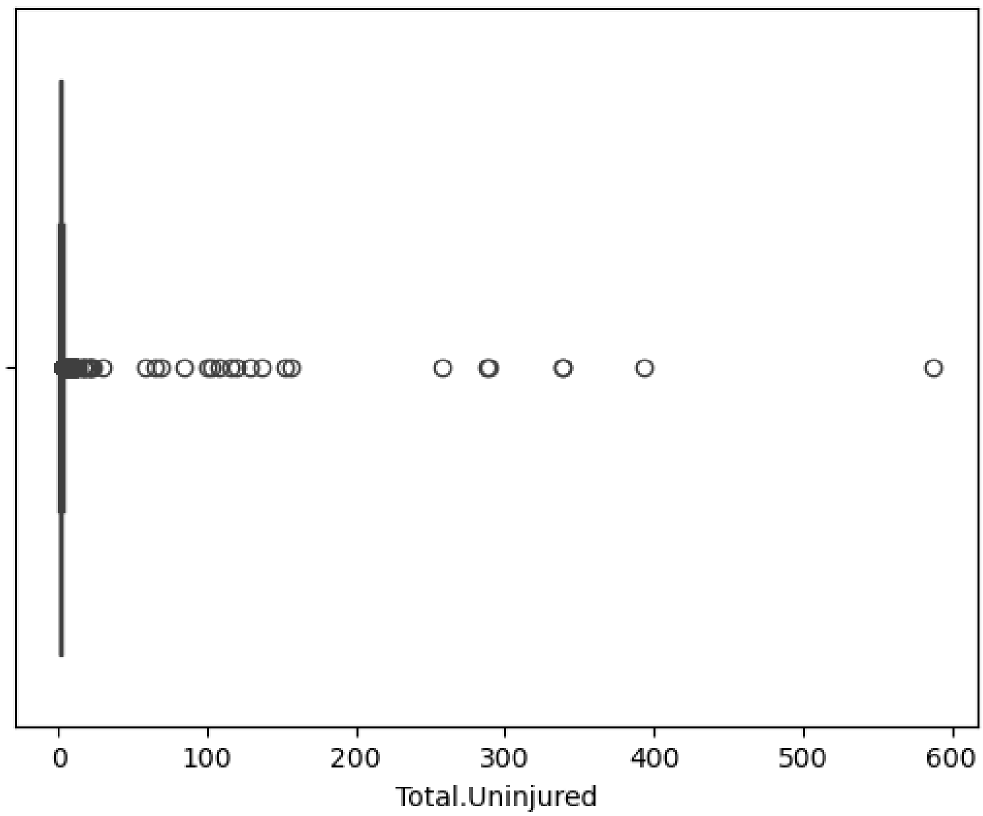

```
# i will use the interquatile range to calculate and filter out
outliers
# Calculate the IQR
Q1 = aviation5['Total.Minor.Injuries'].quantile(0.25)
Q3 = aviation5['Total.Minor.Injuries'].quantile(0.75)
IQR = Q3 - Q1

# Define the bounds for outliers
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

# Filter out outliers
aviation6 = aviation5[(aviation3['Total.Minor.Injuries'] >=
lower_bound) & (aviation5['Total.Minor.Injuries'] <= upper_bound)]
aviation6.reset_index(drop=True, inplace=True)

sns.boxplot(x=aviation6["Total.Minor.Injuries"])
```

```
<Axes: xlabel='Total.Minor.Injuries'>
```

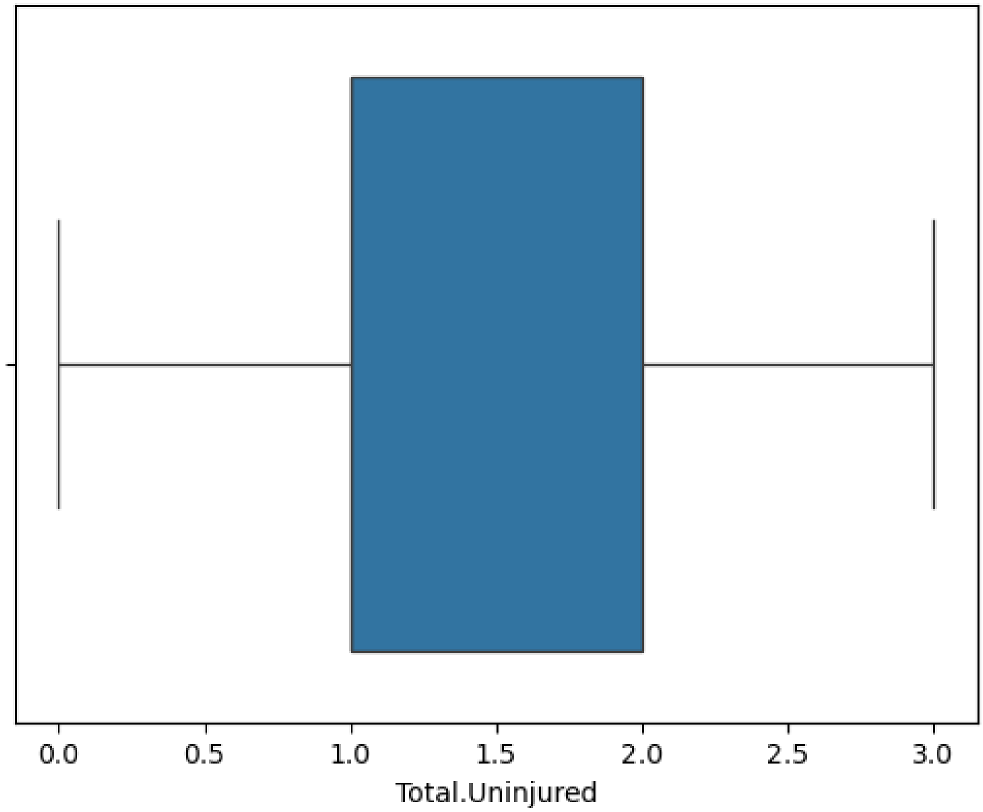



```
sns.boxplot(x=aviation6["Total.Uninjured"])
```

```
<Axes: xlabel='Total.Uninjured'>
```

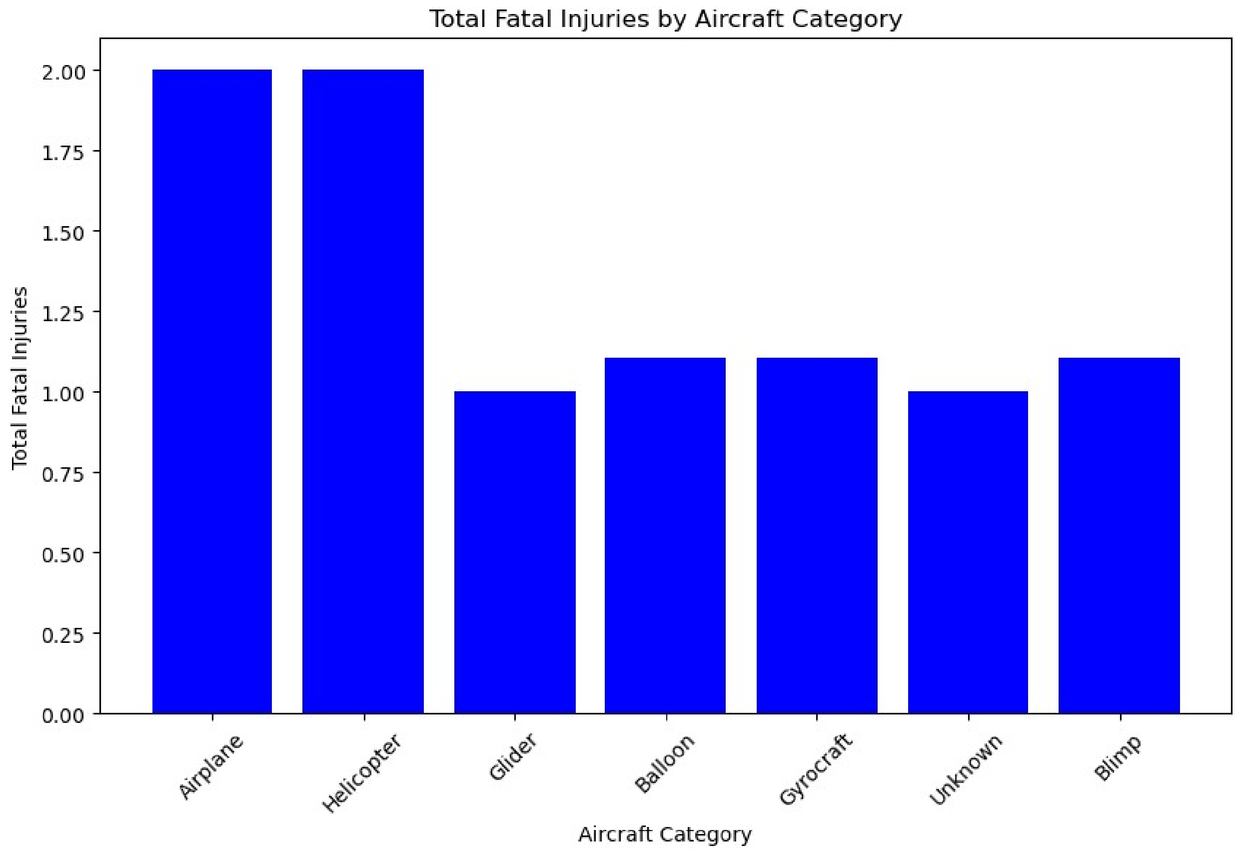



```
# i will use the interquatile range to calculate and filter out
outliers
# Calculate the IQR
Q1 = aviation6['Total.Uninjured'].quantile(0.25)
Q3 = aviation6['Total.Uninjured'].quantile(0.75)
IQR = Q3 - Q1

# Define the bounds for outliers
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

# Filter out outliers
aviation7 = aviation6[(aviation6['Total.Uninjured'] >= lower_bound) &
(aviation6['Total.Uninjured'] <= upper_bound)]
aviation7.reset_index(drop=True, inplace=True)

sns.boxplot(x=aviation7["Total.Uninjured"])

<Axes: xlabel='Total.Uninjured'>
```

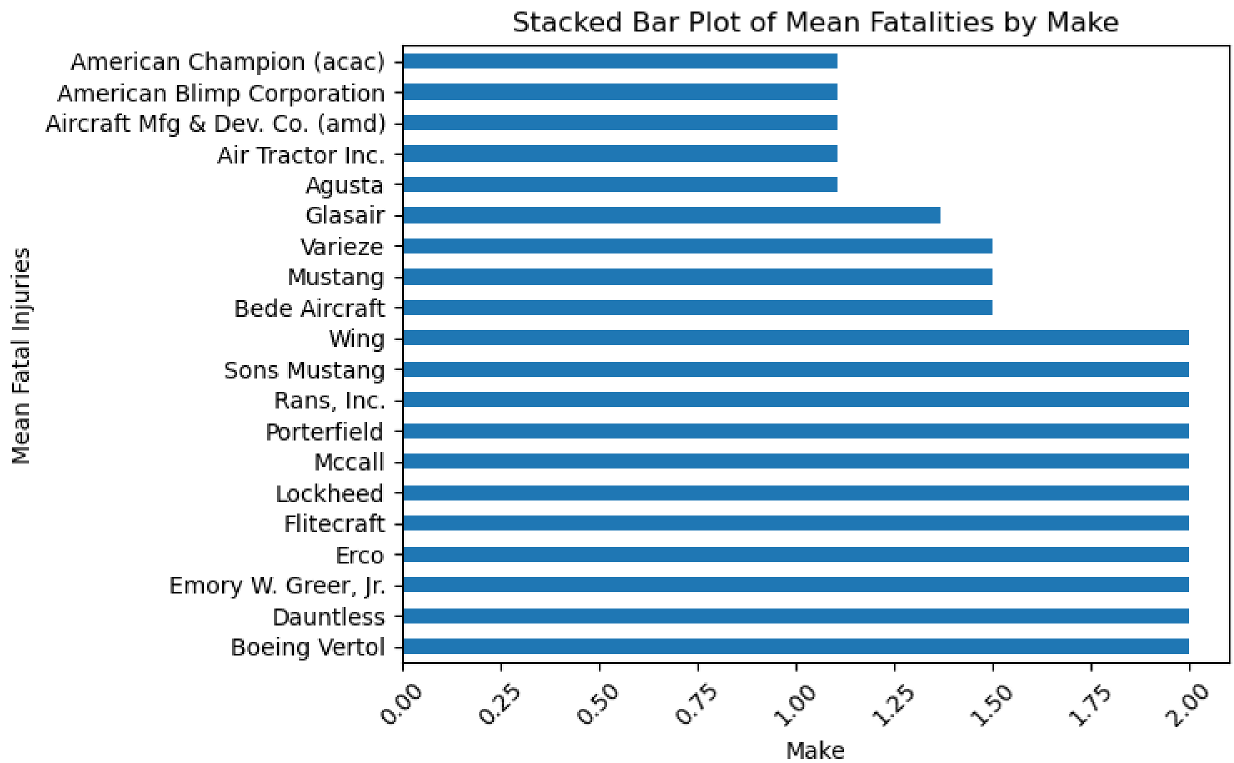


Data Visualization

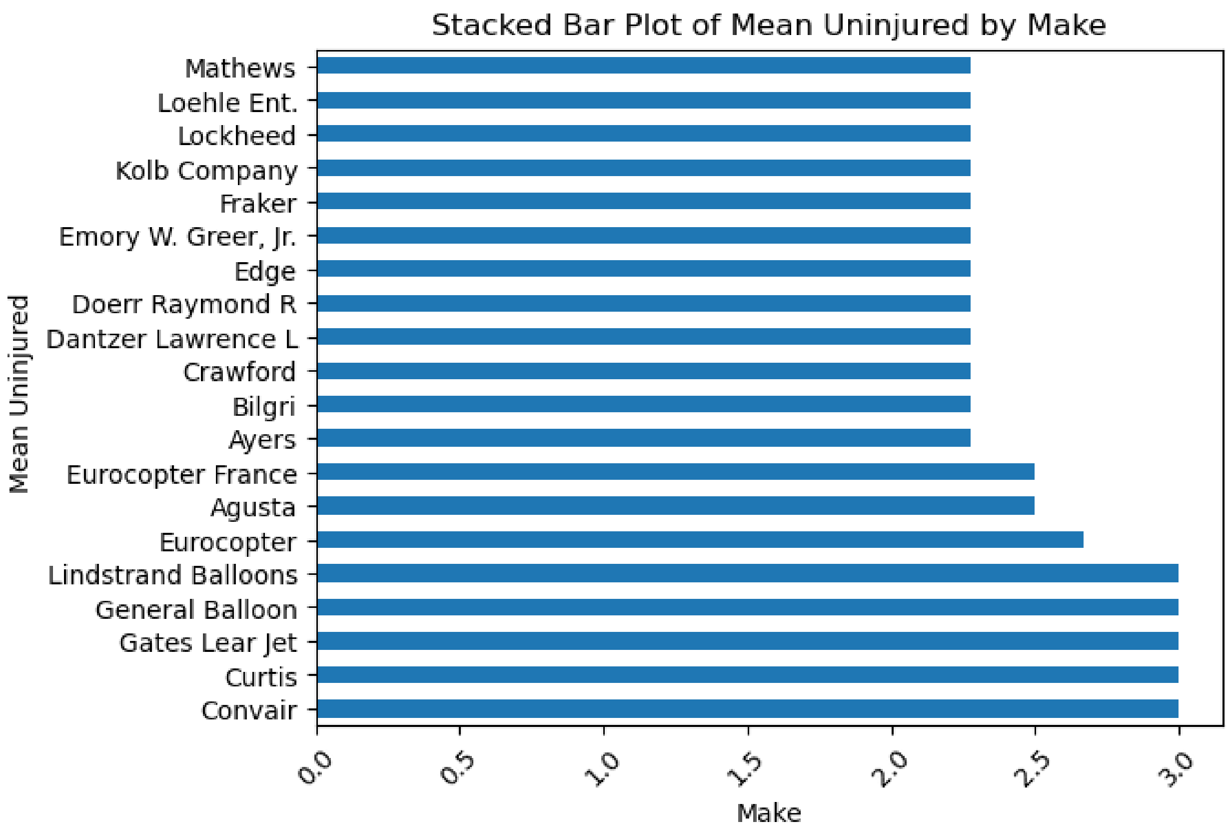
```
# Data preparation for plotting
categories = aviation7['Aircraft.Category']
fatal_injuries = aviation7['Total.Fatal.Injuries']

# Plotting the Bar Chart
plt.figure(figsize=(10, 6))
plt.bar(categories, fatal_injuries, color='blue')

# Add labels and title
plt.xlabel('Aircraft Category')
plt.ylabel('Total Fatal Injuries')
plt.title('Total Fatal Injuries by Aircraft Category')
plt.xticks(rotation=45)
plt.show()
```




```
# Aggregate the mean number of fatalities by make
mean_fatalities_by_make = aviation7.groupby('Make')
['Total.Fatal.Injuries'].mean()
# Get the top 20 makes by mean fatalities as the makes are too many to
plot visibly on the notebook
top_10_mean_makes = mean_fatalities_by_make.nlargest(20)
# Create a DataFrame for the top 20 makes
top_10_mean_df = top_10_mean_makes.reset_index()

# Plot the bar plot
top_10_mean_df.plot(kind='barh', x='Make', y='Total.Fatal.Injuries',
stacked=True, legend=False)
plt.title('Stacked Bar Plot of Mean Fatalities by Make')
plt.xlabel('Make')
plt.ylabel('Mean Fatal Injuries')
plt.xticks(rotation=45)
plt.show()
```




```
# Aggregate the mean number of fatalities by make
mean_fatalities_by_make = aviation7.groupby('Make')
['Total.Uninjured'].mean()
top_10_mean_makes = mean_fatalities_by_make.nlargest(20)
# Create a DataFrame for the top 20 makes
top_10_mean_df = top_10_mean_makes.reset_index()

# Plot the bar plot
top_10_mean_df.plot(kind='barh', x='Make', y='Total.Uninjured',
stacked=True, legend=False)
plt.title('Stacked Bar Plot of Mean Uninjured by Make')
plt.xlabel('Make')
plt.ylabel('Mean Uninjured ')
plt.xticks(rotation=45)
plt.show()
```




```
# Define the file path and name
file_path = r"C:\Users\MONICAH\Documents\Flatiron\new_aviation7.csv"
# Save DataFrame to CSV
aviation7.to_csv(file_path, encoding ='UTF-8', index=False)
```

CONCULUSION

The head of aviation divison should consider purchasing the aircrafts that recorded the highest number of uninjured such as Mathews, lockheed etc. Crafts that registred the highest number of fatal accidents should be avoided since they pose highest risk to property and life.