Final Project Submission

Please fill out:

- Student name: Winfred Karimi
- Student pace: part time
- Scheduled project review date/time: 8/09/2024
- Instructor name: William Okomba
- Blog post URL:

Business understanding

Business problem

Your company is expanding into aviation for commercial and private operations and needs to assess aircraft risks. Your task is to evaluate and identify the lowest-risk aircraft and provide practical recommendations to guide the head of the aviation division in making informed purchase decisions

Business Objective

The objective is to identify low-risk aircraft for commercial and private operations to support the company's expansion into the aviation industry, ensuring well-informed purchasing decisions.

Data and Stakeholders Questions

we will use aviation dataset from the 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. The dataset can be downloaded from kaggle

Shareholders question:

1. What criteria are being used to assess aircraft risk? How will safety records, operating costs, maintenance requirements, and regulatory compliance be evaluated to determine the lowest-risk aircraft?

Data Understanding

```
# importing libraries
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline

# reading our dataset and print the first 5 rows using head method
df = pd.read_csv("AviationData.csv", encoding =('ISO-8859-1' ),
low_memory=False)
df.head()
```

```
         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

           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


  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

3          NaN  ...          Personal         NaN                  2.0

4          NaN  ...          Personal         NaN                  1.0


  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
3                    0.0                  0.0             0.0
4                    2.0                  NaN             0.0

  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               IMC                 Cruise  Probable Cause       12-
09-2000
4               VMC               Approach  Probable Cause       16-
```

```
04-1980

[5 rows x 31 columns]
```

```
# printing the last 5 rows using tail method
df.tail()
```

```
             Event.Id Investigation.Type Accident.Number
Event.Date  \
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  \
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  \
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
\
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
Report.Status  \
88884             0.0               NaN                   NaN
NaN
```

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

      Publication.Date
88884       29-12-2022
88885              NaN
88886       27-12-2022
88887              NaN
88888       30-12-2022

[5 rows x 31 columns]
```

```
# looking at the shape of the data
shape = df.shape
length = len(df)
dataset = type(df)


print(f" The shape of the dataset is: {shape}")
print(f" The length of the dataset is: {length}")
print(f" The type of the dataset is: {dataset}")
```

```
 The shape of the dataset is: (88889, 31)
 The length of the dataset is: 88889
 The type of the dataset is: <class 'pandas.core.frame.DataFrame'>
```

```
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

# summary statistics of the dataframe
df.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
```

```
# checking the datatypes
df.dtypes
```

```
Event.Id                   object
Investigation.Type         object
Accident.Number            object
Event.Date                 object
Location                   object
Country                    object
Latitude                   object
Longitude                  object
Airport.Code               object
Airport.Name               object
Injury.Severity            object
Aircraft.damage            object
Aircraft.Category          object
Registration.Number        object
Make                       object
Model                      object
Amateur.Built              object
Number.of.Engines         float64
Engine.Type                object
FAR.Description            object
Schedule                   object
Purpose.of.flight          object
Air.carrier                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
Publication.Date           object
dtype: object
```

Data Preparation

Our Data has 31 columns,we will not need all of them to aid us in our desicion making, thus will drop the irrelevant ones for our analysis.

```
# checking our columns to determine which ones are irrelevant for our
Analysis
df.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')
```

```
# listing irrelevant columns
irre_cols = ['Event.Id','Accident.Number', 'Event.Date',
       'Location', 'Country', 'Latitude', 'Longitude', 'Airport.Code',
       'Airport.Name', 'Registration.Number','Amateur.Built',
'FAR.Description',
       'Schedule', 'Air.carrier', 'Weather.Condition',
'Report.Status',]

df.drop(columns=irre_cols, inplace=True)

# checking updated columns
col = df.columns
len_col = len(df.columns)

print(f"The columns in our dataset are: {col}")

print(f"The column length of our dataset is: {len_col}")
```

```
The columns in our dataset are: Index(['Investigation.Type',
'Injury.Severity', 'Aircraft.damage',
       'Aircraft.Category', 'Make', 'Model', 'Number.of.Engines',
       'Engine.Type', 'Purpose.of.flight', 'Total.Fatal.Injuries',
       'Total.Serious.Injuries', 'Total.Minor.Injuries',
'Total.Uninjured',
       'Broad.phase.of.flight', 'Publication.Date'],
      dtype='object')
The column length of our dataset is: 15
```

column description

1. Investigation.Type - Describes the type of investigation conducted that is if the event was a full accident investigation or an incident investigation.
2. Injury.Severity - Describes the severity of injuries resulting from the event. I.e Fatal, Non-Fatal Major , minor etc
3. Aircraft.damage - Specifies the extent of damage to the aircraft that is if Destroyed, Substantial or Minor etc.

4. Aircraft.Category - Refers to the classification of the aircraft, such as Airplane, Helicopter, Glider, Balloon etc.
5. Make - The manufacturer of the aircraft, e.g., Boeing, Airbus, Cessna, or other aircraft makers
6. Model - The specific model of the aircraft, such as 737, A320 and C172 which gives more detailed information about the aircraft's design and specifications.
7. Number.of.Engines - The total number of engines on the aircraft.
8. Engine.Type - Describes the type of engine used in the aircraft, such as Turboprop,Jet, Reciprocating or Electric.
9. Purpose.of.flight - Describes the reason or purpose for which the aircraft was in flight, such as Commercial and Private.
10. Total.Fatal.Injuries - The total number of fatalities resulting from the accident or incident.
11. Total.Serious.Injuries - The number of serious injuries reported, as defined by aviation regulations, where serious injuries are typically more severe but non-fatal.
12. Total.Minor.Injuries - The total count of minor injuries, which may include injuries that did not require hospitalization or were less severe.
13. Total.Uninjured - The number of people involved in the accident or incident who were unharmed.
14. Broad.phase.of.flight - Describes the general phase of the flight when the accident or incident occurred. Common phases include Takeoff, limb, Cruise, Descent and Landing.
15. Publication.Date - The date when the investigation report or initial findings were published. This can help track the timeline of the investigation and findings.

```
# Checking missing values in our data
df.isna().sum()
```

```
Investigation.Type           0
Injury.Severity           1000
Aircraft.damage           3194
Aircraft.Category        56602
Make                        63
Model                       92
Number.of.Engines         6084
Engine.Type               7096
Purpose.of.flight         6192
Total.Fatal.Injuries     11401
Total.Serious.Injuries   12510
Total.Minor.Injuries     11933
Total.Uninjured           5912
Broad.phase.of.flight    27165
Publication.Date         13771
dtype: int64
```

```
# dropping null values from object data columns
drop_miss =['Investigation.Type', 'Injury.Severity',
'Aircraft.damage',
       'Aircraft.Category', 'Make', 'Model', 'Engine.Type',
```

```
'Purpose.of.flight',
       'Broad.phase.of.flight', 'Publication.Date']
df.dropna(subset=drop_miss, inplace=True)

# Assign float datatype to fill using mean
df['Number.of.Engines'] =
df['Number.of.Engines'].fillna(df['Number.of.Engines'].mean())

df['Total.Fatal.Injuries'] =
df['Total.Fatal.Injuries'].fillna(df['Total.Fatal.Injuries'].mean())

df['Total.Serious.Injuries'] =
df['Total.Serious.Injuries'].fillna(df['Total.Serious.Injuries'].mean(
))

df['Total.Minor.Injuries'] =
df['Total.Minor.Injuries'].fillna(df['Total.Minor.Injuries'].mean())

df['Total.Uninjured'] =
df['Total.Fatal.Injuries'].fillna(df['Total.Uninjured'].mean())

# Confirming if there is any missing values
df.isna().sum()
```

```
Investigation.Type         0
Injury.Severity            0
Aircraft.damage            0
Aircraft.Category          0
Make                       0
Model                      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
Broad.phase.of.flight      0
Publication.Date           0
dtype: int64
```

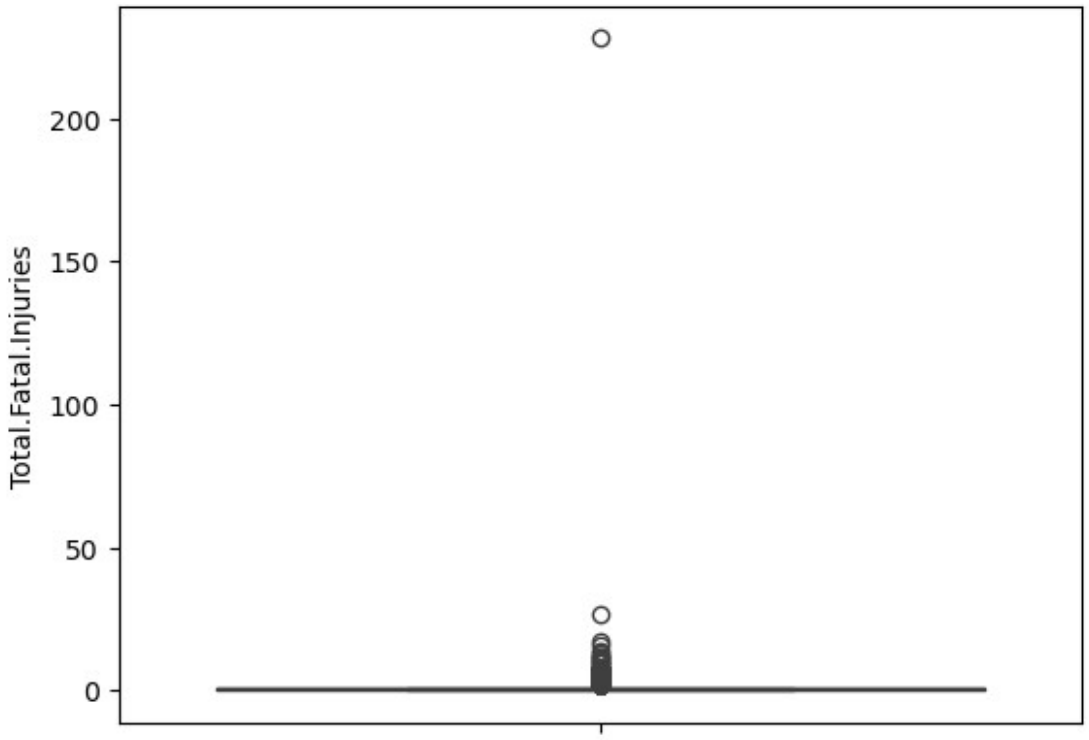
```
# checking for duplicates
df.duplicated().sum()
```

```
95
```

```
# Dropping duplicates
df.drop_duplicates(keep='first', inplace=True)

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

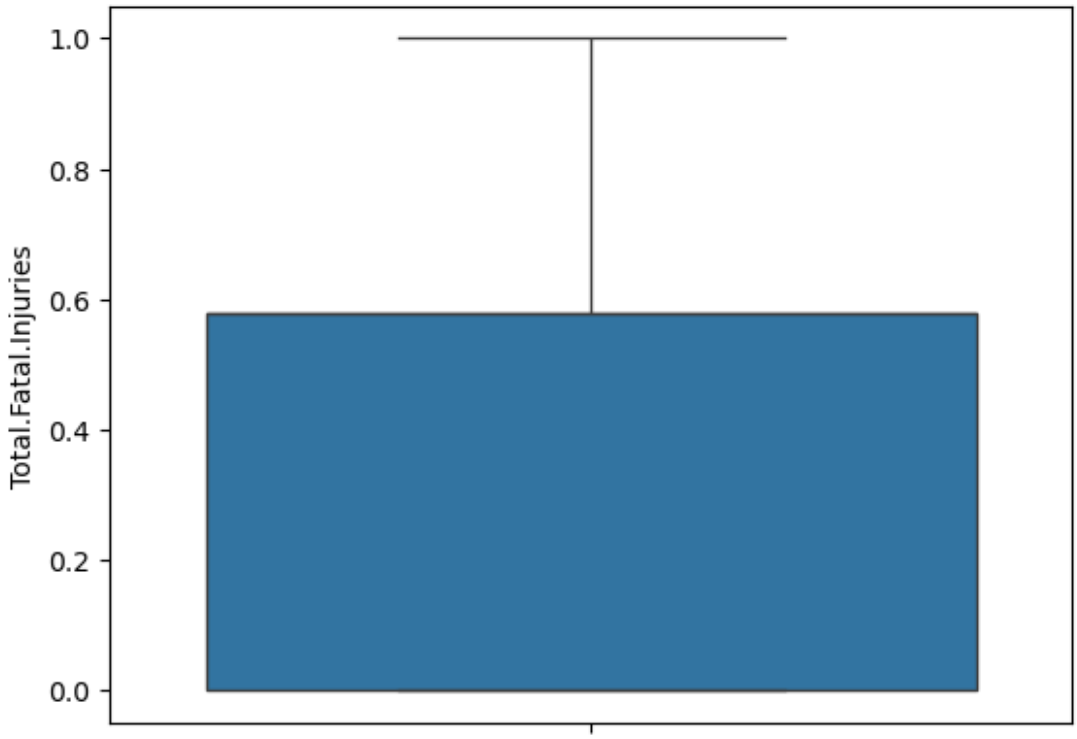
```
# Checking outliers
sns.boxplot(df['Total.Fatal.Injuries'])
```

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



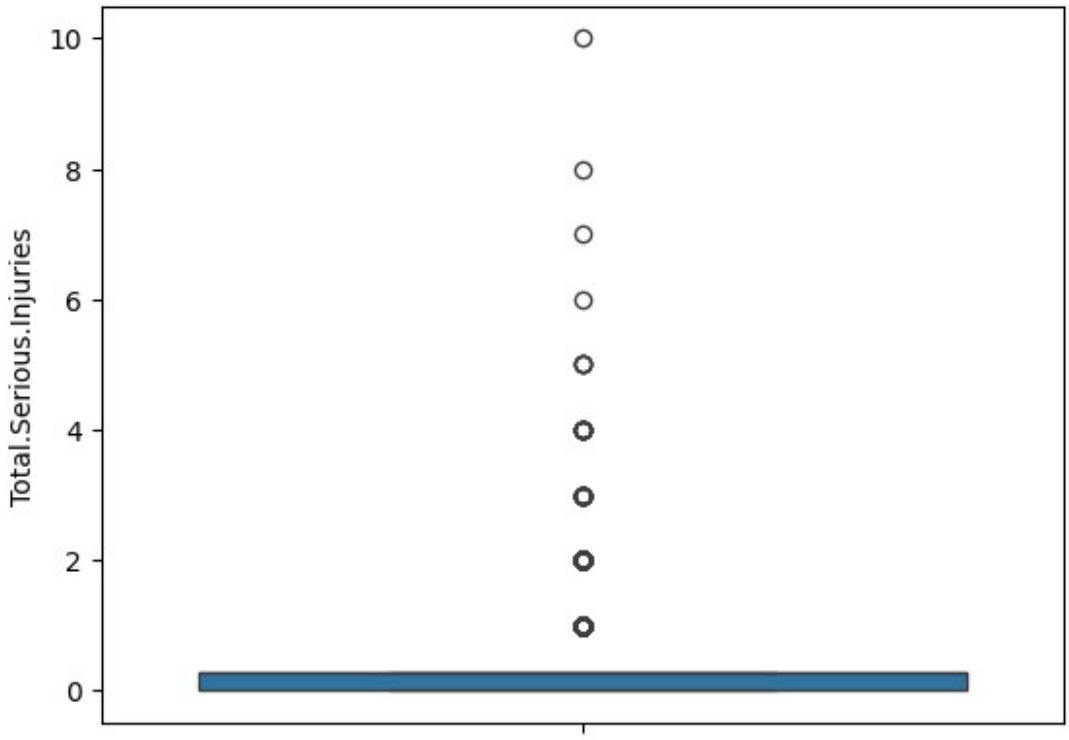

```
# using the interquatile range to drop out outliers
# Calculate the IQR
Q1 = df['Total.Fatal.Injuries'].quantile(0.25)
Q3 = df['Total.Fatal.Injuries'].quantile(0.75)
IQR = Q3 - Q1

# calculating lower bound and uppper bound
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

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

# Checking for outlier for total fatal injuries
sns.boxplot(df['Total.Fatal.Injuries'])
```

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



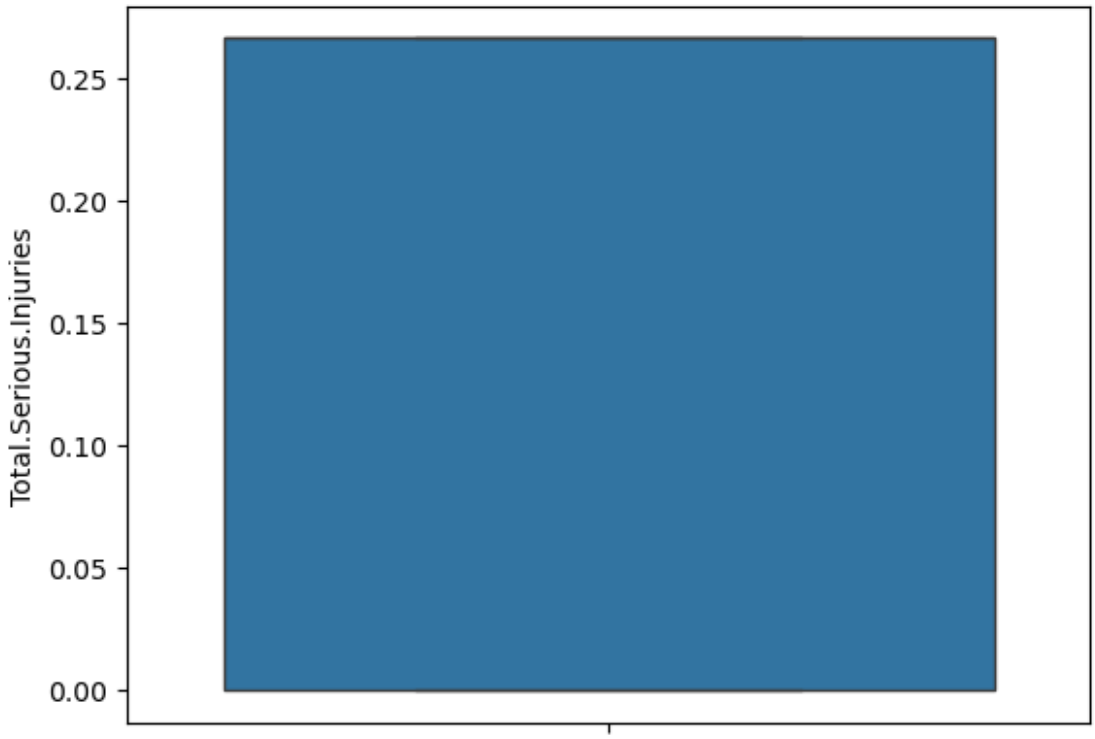

```
# Checking outliers for Total.Serious.Injuries
sns.boxplot(df['Total.Serious.Injuries'])
```

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



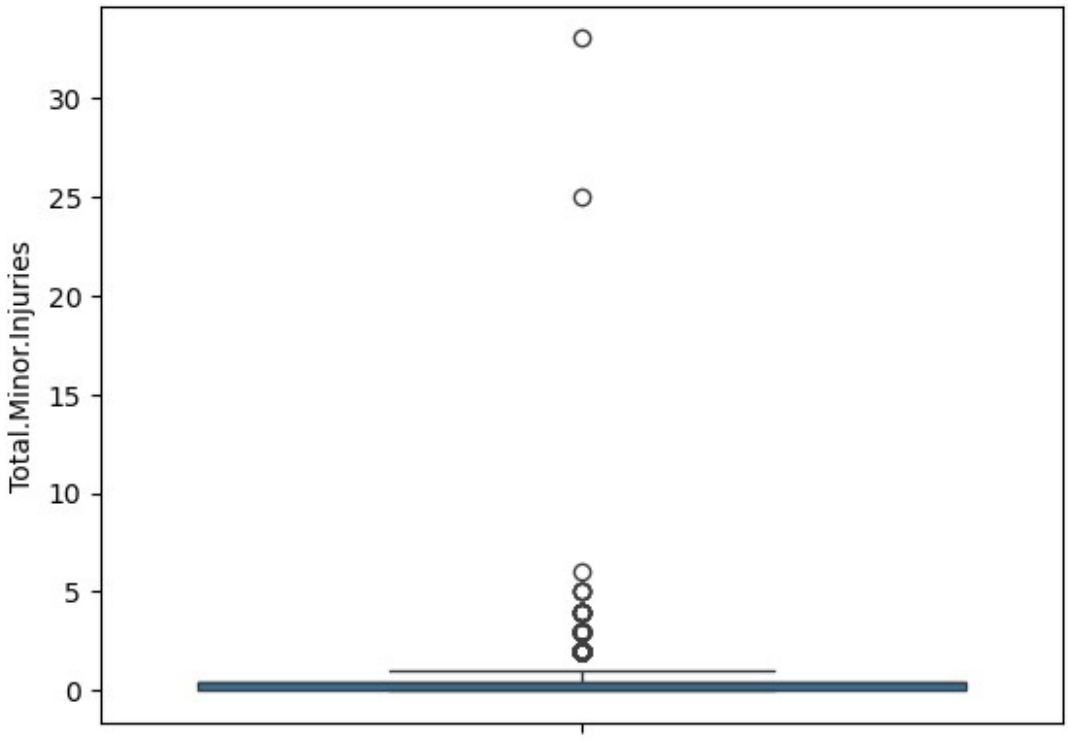

```
# using the interquatile range to drop out outliers
# Calculate the IQR
Q1 = df['Total.Serious.Injuries'].quantile(0.25)
Q3 = df['Total.Serious.Injuries'].quantile(0.75)
IQR = Q3 - Q1

# calculating lower bound and uppper bound
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

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

sns.boxplot(df['Total.Serious.Injuries'])
```

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



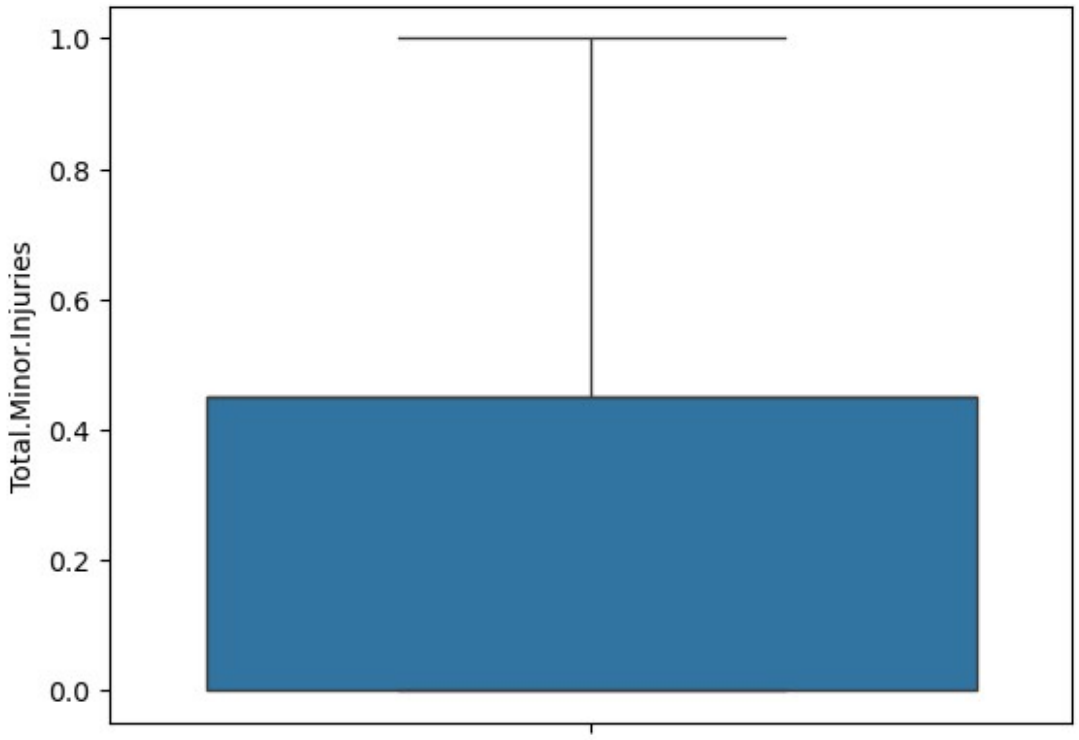

```
# Checking outliers for Total.Minor.Injuries
sns.boxplot(df['Total.Minor.Injuries'])
```

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



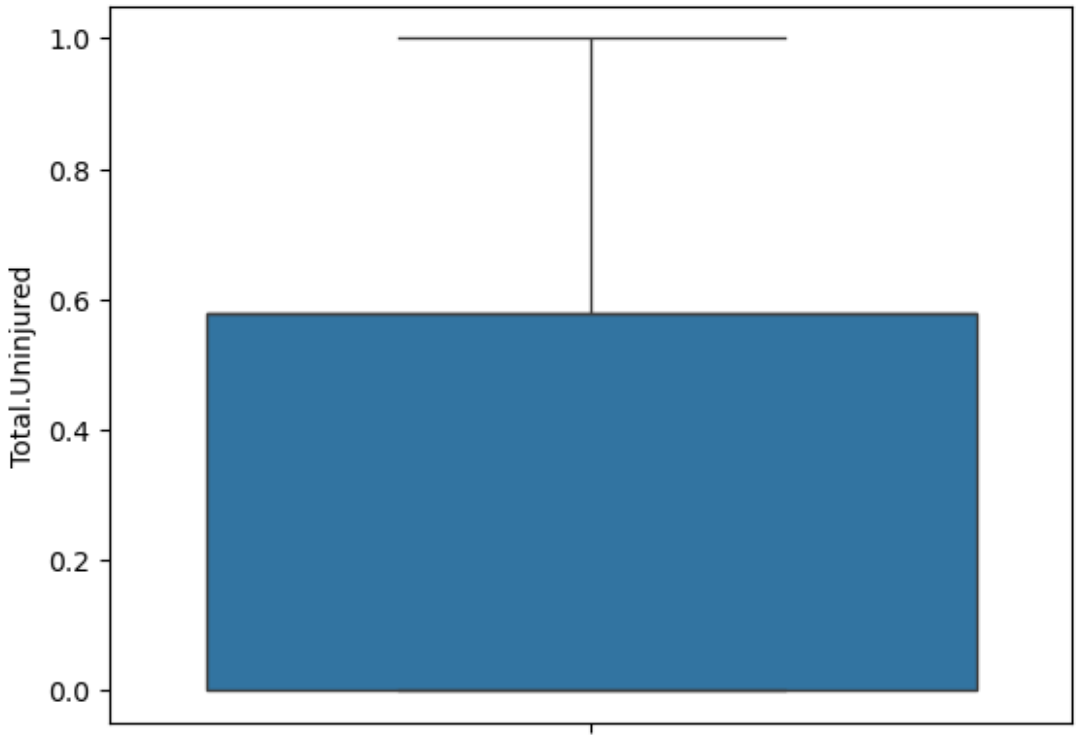

```
# using the interquatile range to drop out outliers
# Calculate the IQR
Q1 = df['Total.Minor.Injuries'].quantile(0.25)
Q3 = df['Total.Minor.Injuries'].quantile(0.75)
IQR = Q3 - Q1

# calculating lower bound and uppper bound
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

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

# Checking outliers for Total.Minor.Injuries
sns.boxplot(df['Total.Minor.Injuries'])
```

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



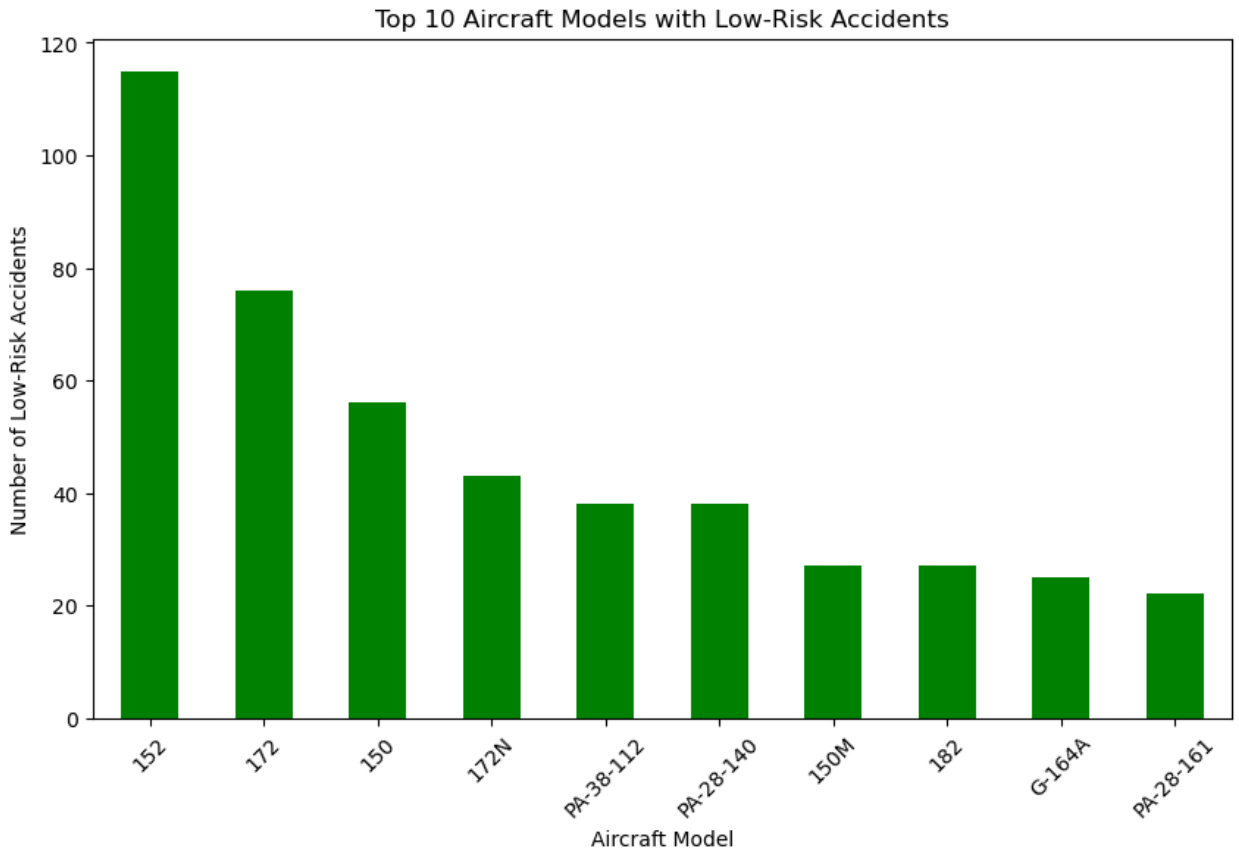

```
# Checking outliers for Total.Uninjured
sns.boxplot(df['Total.Uninjured'])
```

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



Data Visualization

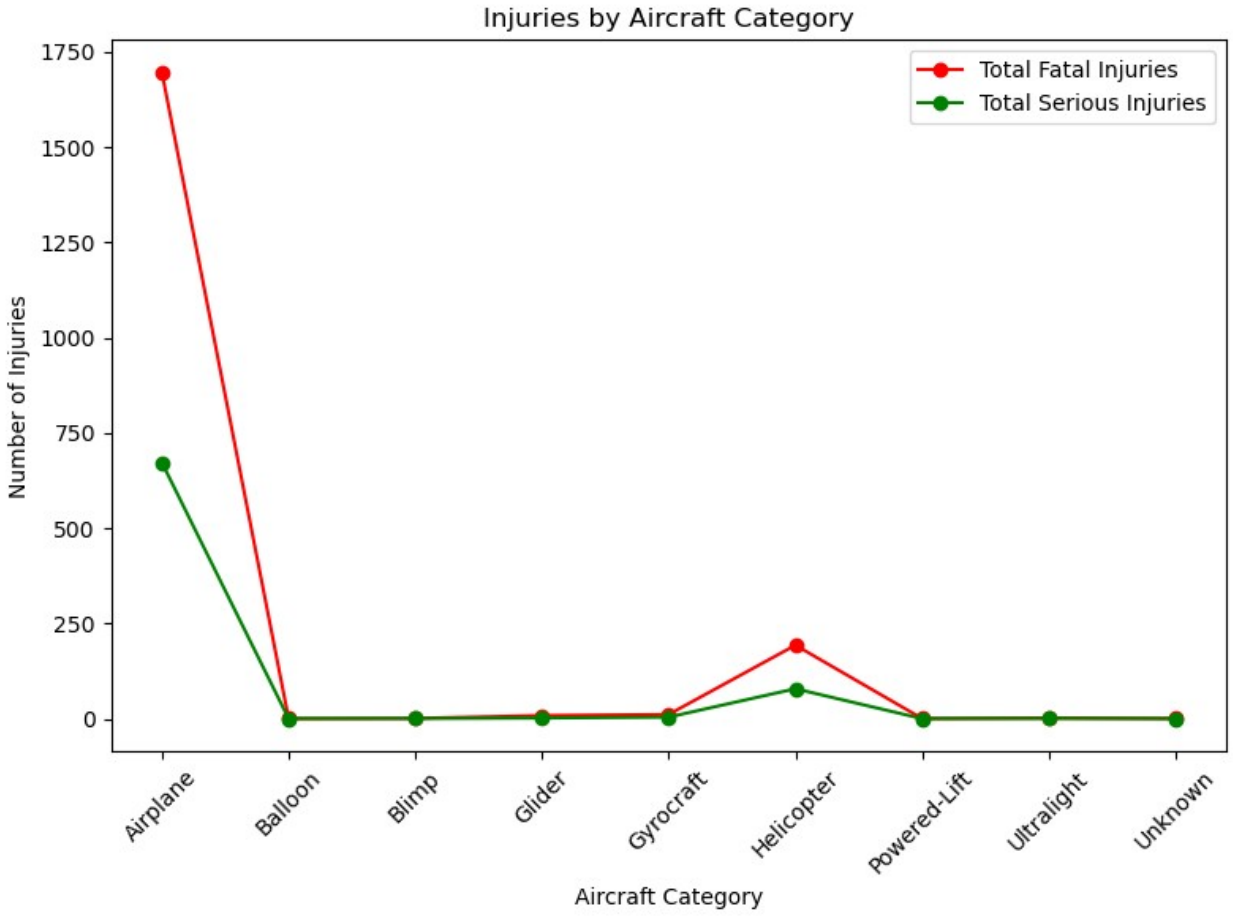
```
""" To indentify aircraft with lowest risk by filtering the data for
planes that were involved in accidents with zero fatal injuries and
fewer total serious injures, i.e TOP 10 AIRCRAFT MODELS WITH LOW-RISK
ACCIDENTS"""

# create a variable and name it low risk aircraft
low_risk_aircraft = (
    (df['Total.Fatal.Injuries'] == 0) &
    (df['Total.Serious.Injuries'] <= 1))
# Filter the dataset based on the low-risk aircraft
low_risk_planes = df[low_risk_aircraft]
# model counts of low-risk accidents by Aircraft Model
model_counts = low_risk_planes['Model'].value_counts()
# Select the top 10 models with the most low-risk accidents
top_10_models = model_counts.head(10)
# Plot the top 10 low-risk accidents by Aircraft Model
plt.figure(figsize=(10, 6))
top_10_models.plot(kind='bar', color='green')
plt.title('Top 10 Aircraft Models with Low-Risk Accidents')
plt.xlabel('Aircraft Model')
plt.ylabel('Number of Low-Risk Accidents')
plt.xticks(rotation=45)
plt.show()
plt.savefig('my_plot1.png')
```




```
<Figure size 640x480 with 0 Axes>
```

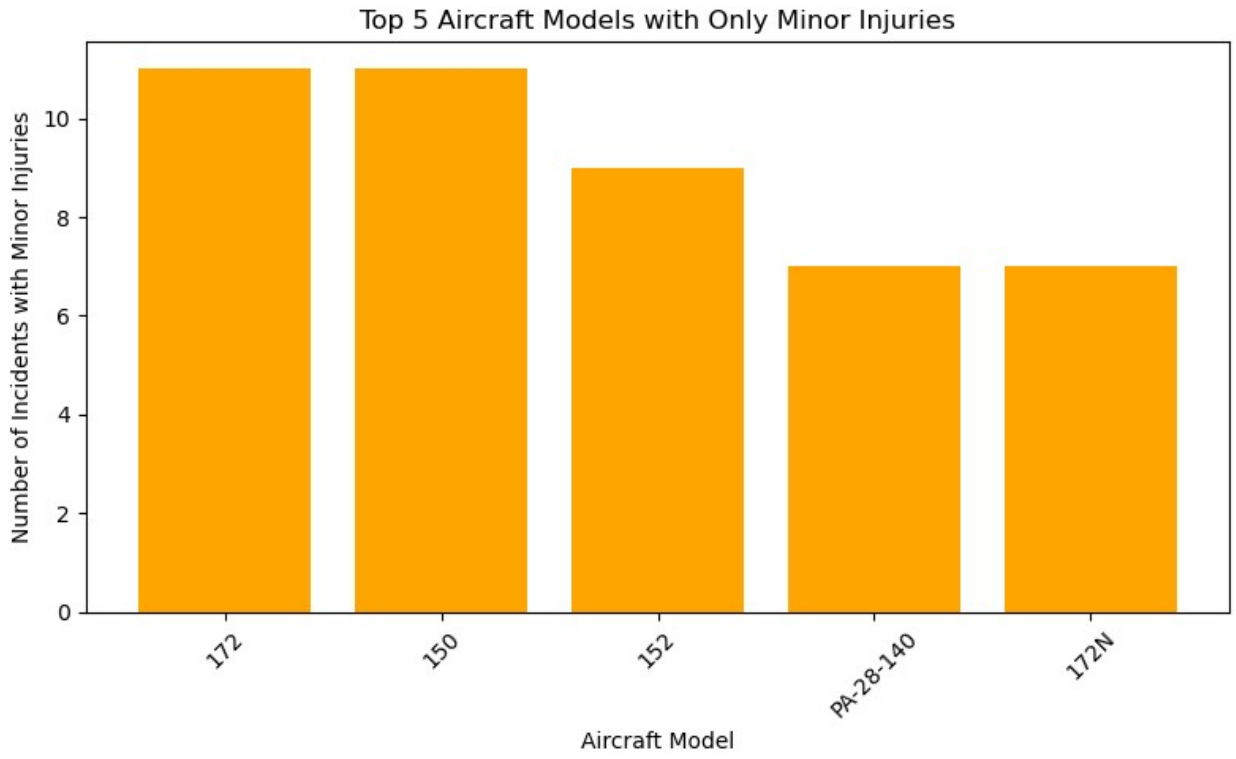
```
""" Analysis to identify Aircraft category vs fatal and serious
injuries"""
# Grouping data by Aircraft Category and summing the injury columns i
will name it
inju_by_categ = df.groupby('Aircraft.Category')
[['Total.Fatal.Injuries',
'Total.Serious.Injuries']].sum().reset_index()
# Plotting the data
plt.figure(figsize=(8,6))
# Line for Total Fatal Injuries
plt.plot(inju_by_categ['Aircraft.Category'],
inju_by_categ['Total.Fatal.Injuries'], marker='o', linestyle='-',
color='red', label='Total Fatal Injuries')
# Line for Total Serious Injuries
plt.plot(inju_by_categ['Aircraft.Category'],
inju_by_categ['Total.Serious.Injuries'], marker='o', linestyle='-',
color='green', label='Total Serious Injuries')
plt.xticks(rotation=45)
plt.xlabel('Aircraft Category')
plt.ylabel('Number of Injuries')
plt.title('Injuries by Aircraft Category')
```

```
plt.legend()

plt.tight_layout()
plt.show()
plt.savefig('my_plot2.png')
```




```
<Figure size 640x480 with 0 Axes>

""" Analysis to find out Aircraft Models with Only Minor Injuries,
(Narrowing to top five models with minor injuries)."""

# Filter the data for aircraft with only minor injuries (no fatal or
serious injuries)
data = df[(df['Total.Fatal.Injuries'] == 0) &
                                        (df['Total.Serious.Injuries']
== 0) &
                                        (df['Total.Minor.Injuries'] >
0)]
data_model_counts =
data.groupby('Model').size().reset_index(name='Count')
```

```
# Sort the data and select top 5 models
top_5_minor_injury = data_model_counts.sort_values(by='Count',
ascending=False).head(5)

# Plotting the top 5 models with only minor injuries
plt.figure(figsize=(8, 5))
plt.bar(top_5_minor_injury['Model'], top_5_minor_injury['Count'],
color='orange')

plt.title('Top 5 Aircraft Models with Only Minor Injuries')
plt.xlabel('Aircraft Model')
plt.ylabel('Number of Incidents with Minor Injuries')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```




```
<Figure size 640x480 with 0 Axes>
```

Summary of data Visualization, interplation and conclusion

Gragh 1:TOP 10 AIRCRAFT MODELS WITH LOW-RISK ACCIDENTS- The aircraft model with the highest bar (in this case, "152") is the most frequently involved in accidents that were categorized as low-risk.The accidents did not result in fatalities or serious injuries thus making it a reliable interms of safety for both private and commercial ventures. Model 172, and 150, rank second and 3rd best based on this anlysis thus could also be realible options.

Gragh 2 :INJURIES BY AIRCRAFT CATEGORY-• Based on the graph, blimps, gliders, gyrocraft, powered-lift, ultralight, and balloons appear to represent the lowest-risk aircraft categories. These categories either have no significant fatal or serious injuries, or their contribution to aviation injuries is extremely low. Airplanes and helicopters show a significantly higher risk of injuries, particularly fatal ones, and thus are not considered low-risk.

Gragh 3 :TOP 5 AIRCRAFT MODELS WITH ONLY MINOR INJURIES-• Models 172, 150, 172, have highest number of minor injuries meaning the damage in accidents has been minimal and does not pose significant risks to the safety of passengers or crew. This makes these models ideal for less risky operations where safety is important but occasional minor incidents are acceptable.

CONCLUSION: Models 152, 172, and 150 consistently show that they are involved in incidents with only minor injuries. These aircraft should be prioritized for purchase, particularly for operations that require reliability and safety.