Flying High: Choosing Safe Aircraft for Growth

Business Understanding

The business is pursuing a new strategic venture in the aviation sector. The objective is to **assist the head of the newly established aviation division** in identifying the **risks associated with aircraft operations**, including **potential operational mishaps** for both **commercial and private aircraft**. The aim is to provide **evidence-based** insights to support informed **decision-making** based on the gathered data.

Problem Statement

The company is **expanding into the aviation sector**, focusing on the purchase and operation of both commercial and private aircraft. However, there is a **lack of comprehensive understanding regarding the associated risks**, **operational mishaps**, and **safety concerns** related to these aircraft types. The challenge is to **identify the aircraft models** that pose the **lowest risk** and provide **data-driven insights** that will enable the head of the new aviation division to make informed, evidence-based decisions on which aircraft to purchase and operate for the business's success and growth

Objectives

1. Identifying the safest and unsafest aircrafts and their Characteristics
2. Identifying the major cause of accidents
3. Identifying the effect of weather on accidents
4. Does the type of build affect the number of fatalities
5. Which year had the most casualties

Methodology

The data is obtained from the Kaggle Repository. The below is the methods pursued to achieve the reccomendations below;

1. **Data Cleaning**

- Removing null columns and rows
- Harmonising data to prevent duplication

2. **Descriptive Statistics**

- Identifying correlations
- Introducing columns in relation to existing data to enrich the data set

3. **Visualisation**

- Using different plots **focusing on the 5 key objectives set** with the support of matplotlib and seaborn to achieve relational data and insights

4. **Conclusion and Recommendations**

- Developing actionable insights from the analysis achieved

Success Criteria

Ranking Aircraft Safety:

Rank aircraft based on accident rates and fatalities. Identify key risk factors.

Categorizing Accident Causes:

Classify and quantify major accident causes Provide actionable recommendations for risk mitigation

Weather Impact on Accidents:

Quantify how weather conditions correlate with accident frequency and severity

Fatalities by Aircraft Type:

Identify differences in fatalities between commercial and private aircraft. Offer recommendations on prioritizing aircraft types based on safety.

Identifying Peak Years for Casualties:

Provide time-based insights into trends and potential external factors influencing accidents.

Actionable Insights and Recommendations:

Provide clear, data-driven recommendations for aircraft purchasing and operations. Focus on safety, risk mitigation, and cost-effectiveness in decisions.

Limitations and Assumptions

Limitations

1. The data had some missing items and some named as unknown which reduced the accuracy of the dataset

Assumptions

1. The total classification of injuries summed up gave the total number of passengers in one flight
2. Year 2020 had the highest number of fatalities. This could be due to the world-wide lockdown in relation to Covid19 and therefore a high increase in recreational activities to reduce boredom
3. Personal flights were related to recreational purposes

Data Understanding

Data Discovery

```
#importing libraries
#importing data

import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
aviation_df = pd.read_csv('AviationData.csv', encoding='latin1')
```

```
<ipython-input-1-b9f7e159fc09>:8: DtypeWarning: Columns (6,7,28) have mixed types. Speci
  aviation_df = pd.read_csv('AviationData.csv', encoding='latin1')
```

```
aviation_df.shape
```

```
(88889, 31)
```

```
aviation_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')
```

```
#Understanding the dataframe -
#1.no of rows and columns
#2.data type
#3.number of non-null values per column
aviation_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
```

```
#Generating descriptive analysis of the data
aviation_df.describe()
```

	Number.of.Engines	Total.Fatal.Injuries	Total.Serious.Injuries	Total.Minor.Inj
count	82805.000000	77488.000000	76379.000000	76956.0(
mean	1.146585	0.647855	0.279881	0.3!
std	0.446510	5.485960	1.544084	2.2:
min	0.000000	0.000000	0.000000	0.0(
25%	1.000000	0.000000	0.000000	0.0(
50%	1.000000	0.000000	0.000000	0.0(
75%	1.000000	0.000000	0.000000	0.0(
max	8.000000	349.000000	161.000000	380.0(

Data Cleaning

```
#Identifying number of null values per column
aviation_df.isna().sum()
```

	0
Event.Id	0
Investigation.Type	0
Accident.Number	0
Event.Date	0
Location	52
Country	226
Latitude	54507
Longitude	54516
Airport.Code	38757
Airport.Name	36185
Injury.Severity	1000
Aircraft.damage	3194
Aircraft.Category	56602
Registration.Number	1382
Make	63
Model	92
Amateur.Built	102
Number.of.Engines	6084
Engine.Type	7096
FAR.Description	56866
Schedule	76307
Purpose.of.flight	6192
Air.carrier	72241
Total.Fatal.Injuries	11401
Total.Serious.Injuries	12510
Total.Minor.Injuries	11933
Total.Uninjured	5912
Weather.Condition	4492
Broad.phase.of.flight	27165
Report.Status	6384

Publication.Date	13771

dtype: int64

```
#Dropping columns with high number of null values
aviation_df_cln = aviation_df.drop(columns=['Airport.Code','Airport.Name','Latitude','Longit
```

```
#8 columns dropped
aviation_df_cln.shape
```

```
(88889, 23)
```

```
#verification of the dropped columns
aviation_df_cln.isna().sum()
```

	0
Event.Id	0
Investigation.Type	0
Accident.Number	0
Event.Date	0
Location	52
Country	226
Injury.Severity	1000
Aircraft.damage	3194
Registration.Number	1382
Make	63
Model	92
Amateur.Built	102
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
Weather.Condition	4492
Broad.phase.of.flight	27165
Report.Status	6384
Publication.Date	13771

dtype: int64

```
#Identifying unique values to harmonise data
aviation_df_cln.Make.unique().tolist()
```

```
['Stinson',
 'Piper',
 'Cessna',
 'Rockwell',
 'Mcdonnell Douglas',
```

```
'North American',
'Beech',
'Bellanca',
'Navion',
'Enstrom',
'Smith',
'Bell',
'Grumman',
'Beechcraft',
'Maule',
'Air Tractor',
'Aerospatiale',
'Mooney',
'Boeing',
'Curtis',
'Schleicher',
'Quickie',
'Lockheed',
'Embraer',
'Hughes',
'Swearingen',
'De Havilland',
'Bell Helicopter',
'Bede Aircraft',
'Convair',
'Beachner',
'Canadair',
'Douglas',
'Sons Mustang',
'Dassault/sud',
'Sikorsky',
'Bell/textron',
'Robertson',
'Aeronca',
'Smith Miniplane',
'Mitsubishi',
'Mcdonnell-douglas',
'Taylorcraft',
'Ted Smith',
'Robinson',
'Raven',
'Ercoupe',
'Rockwell Comdr',
'Howard Aircraft Corp.',
'Porterfield',
'Nihon',
'Great Lakes',
'Balloon Works',
'Pitts',
'Fairchild Hiller',
'Kaman',
'Weatherly',
'Eagle'
```

```
#Changing all values in column 'Make' to upper case
#This will remove the case sensitivity
```

```
aviation_df_cln['Make']= aviation_df_cln['Make'].str.upper()
aviation_df_cln['Make']
```

	Make
0	STINSON
1	PIPER
2	CESSNA
3	ROCKWELL
4	CESSNA
...	...
88884	PIPER
88885	BELLANCA
88886	AMERICAN CHAMPION AIRCRAFT
88887	CESSNA
88888	PIPER

88889 rows × 1 columns

dtype: object

```
#checking for duplicate values with reference to accident number as the unique identifier
aviation_df_cln['Accident.Number'].duplicated().sum()
```

```
26
```

```
#checking for duplicated values in the accident number as the unique identifier
duplicates = aviation_df_cln[aviation_df_cln['Accident.Number'].duplicated()]
print(duplicates)
```

```
             Event.Id Investigation.Type Accident.Number  Event.Date  \
87305  20220111104514           Accident      ERA22LA103  2022-01-08
87331  20220309104747           Incident      DCA22WA089  2022-01-15
87348  20220801105632           Incident      DCA22WA167  2022-01-22
87412  20220212104630           Accident      ERA22LA119  2022-02-11
87549  20220323104818           Accident      CEN22LA149  2022-03-18
87623  20220406104897           Accident      WPR22LA143  2022-04-02
87917  20220614105258           Incident      DCA22WA130  2022-06-05
87932  20220608105217           Accident      WPR22LA201  2022-06-07
87991  20220623105317           Accident      DCA22LA135  2022-06-18
88085  20220726105577           Incident      DCA22WA158  2022-07-02
88096  20220804105661           Incident      DCA22WA172  2022-07-04
88168  20220802105640           Accident      GAA22WA241  2022-07-16
88175  20220718105496           Accident      ERA22FA318  2022-07-17
88241  20220727105589           Accident      ERA22FA338  2022-07-26
```

```
88258  20220730105623         Accident     CEN22LA346  2022-07-28
88315  20220808105682         Accident     ERA22LA364  2022-08-06
88372  20220818105763         Accident     WPR22FA309  2022-08-18
88387  20220822105776         Accident     ERA22LA379  2022-08-20
88513  20220915105950         Accident     DCA22LA201  2022-09-11
88528  20220921105978         Incident     DCA22WA204  2022-09-14
88538  20220918105957         Accident     CEN22FA424  2022-09-17
88593  20220929106019         Accident     DCA22WA214  2022-09-28
88777  20221112106276         Accident     CEN23MA034  2022-11-12
88796  20221121106336         Accident     WPR23LA041  2022-11-18
88798  20221122106340         Incident     DCA23WA071  2022-11-18
88814  20221123106354         Accident     WPR23LA045  2022-11-22

             Location        Country Injury.Severity Aircraft.damage  \
87305    Knoxville, TN  United States       Non-Fatal     Substantial
87331        Sukkur, OF       Pakistan             NaN             NaN
87348          Kigali,          Rwanda             NaN             NaN
87412       Naples, FL  United States       Non-Fatal           Minor
87549    Grapevine, TX  United States       Non-Fatal     Substantial
87623     Van Nuys, CA  United States       Non-Fatal     Substantial
87917     Peshawar, OF  United States             NaN             NaN
87932    Hawthorne, CA  United States       Non-Fatal           Minor
87991     New York, NY  United States             NaN     Substantial
88085       Barcelona,           Spain             NaN             NaN
88096             TBD,              MU             NaN             NaN
88168         Aaachen,         Georgia             NaN     Substantial
88175    Las Vegas, NV  United States           Fatal       Destroyed
88241      Portland, AR  United States          Fatal     Substantial
88258      Oshkosh, WI  United States       Non-Fatal     Substantial
88315      Erwinna, PA  United States       Non-Fatal     Substantial
88372  Watsonville, CA  United States           Fatal       Destroyed
88387     Bealeton, VA  United States           Minor     Substantial
88513      Chicago, IL  United States             NaN             NaN
88528          Mumbai,           India             NaN             NaN
88538     Longmont, CO  United States           Fatal       Destroyed
88593          London,   Great Britain       Non-Fatal             NaN
88777       Dallas, TX  United States           Fatal       Destroyed
88796    Las Vegas, NV  United States       Non-Fatal     Substantial
88798       Marrakech,         Morocco             NaN             NaN
88814    San Diego, CA  United States       Non-Fatal     Substantial

      Registration.Number            Make  ...    Engine.Type  \
87305               N1308U          CESSNA        Reciprocating
```

```
#dropping duplicated rows with reference to Accident Number
aviation_df_drpd =aviation_df_cln.drop_duplicates(subset='Accident.Number', keep='first')
```

```
#confirming the number of resultant rows
aviation_df_drpd.shape
```

```
(88863, 23)
```

```
#confirming if duplicated rows are dropped
aviation_df_drpd['Accident.Number'].duplicated().sum()
```

```
0
```

```
aviation_df_drpd.to_csv('AviationDataCleaned.csv')
```

Data Manipulation

```
#Adding a column that represents the total number of passengers by adding all total injured
aviation_df_drpd['Total.Passengers']=aviation_df_drpd['Total.Fatal.Injuries'] + aviation_df_
aviation_df_drpd['Total.Passengers']
```

```
<ipython-input-18-e8a396bc4763>:2: 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_
  aviation_df_drpd['Total.Passengers']=aviation_df_drpd['Total.Fatal.Injuries'] + aviati
```

	Total.Passengers
0	2.0
1	4.0
2	NaN
3	2.0
4	NaN
...	...
88884	1.0
88885	0.0
88886	1.0
88887	0.0
88888	2.0

88863 rows × 1 columns

dtype: float64

```
#Getting statistical data on additional row
aviation_df_drpd['Total.Passengers'].describe()
```

	Total.Passengers
count	74333.000000
mean	5.843690
std	26.744801
min	0.000000
25%	1.000000
50%	2.000000
75%	2.000000
max	576.000000

dtype: float64

```
#Top 10 Aircrafts in the dataset
top_10_Make = aviation_df_drpd['Make'].value_counts().head(10)
top_10_Make
```

	count
Make	
CESSNA	27141
PIPER	14870
BEECH	5372
BOEING	2734
BELL	2722
MOONEY	1334
ROBINSON	1229
GRUMMAN	1172
BELLANCA	1045
HUGHES	932

dtype: int64

```
#Identifying the aircrafts that had the top 10 total number of passengers
top_10_carriers = aviation_df_drpd.groupby('Make')['Total.Passengers'].sum().sort_values(asc
top_10_carriers
```

Make	Total.Passengers
BOEING	177211.0
CESSNA	47057.0
MCDONNELL DOUGLAS	37573.0
PIPER	26895.0
AIRBUS	20447.0
AIRBUS INDUSTRIE	13842.0
BEECH	11768.0
DOUGLAS	8674.0
LOCKHEED	7075.0
BELL	5206.0

dtype: float64

```
#viewing correlation between int and float data
aviation_df.select_dtypes(include=['int', 'float']).corr()
```

	Number.of.Engines	Total.Fatal.Injuries	Total.Serious.Injuries	To
Number.of.Engines	1.000000	0.098505	0.046157	
Total.Fatal.Injuries	0.098505	1.000000	0.135724	
Total.Serious.Injuries	0.046157	0.135724	1.000000	
Total.Minor.Injuries	0.098162	0.073559	0.326849	
Total.Uninjured	0.406058	-0.015214	0.052869	

Visualisation

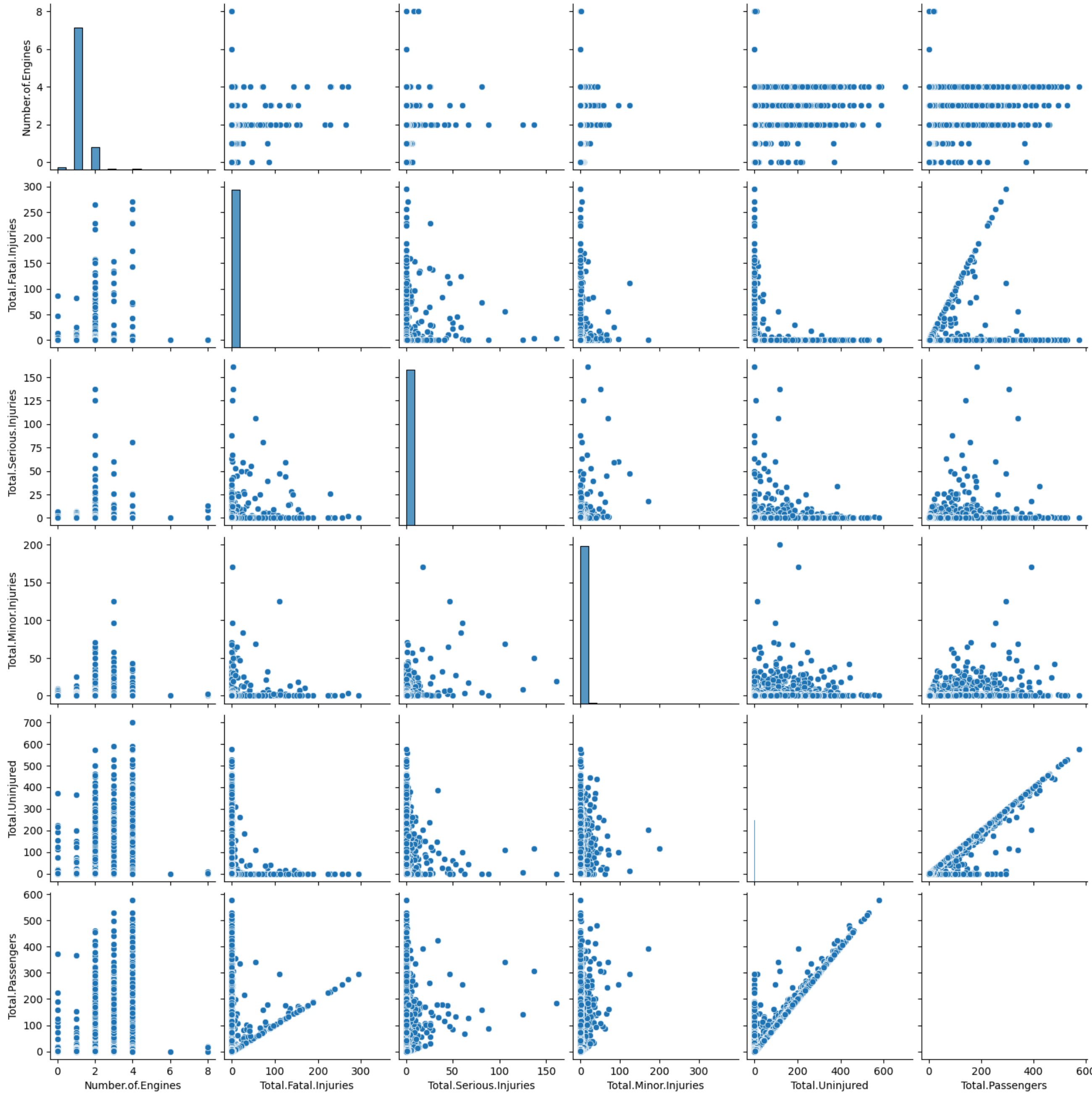
Answering objectives through visualisation

Descriptive Visualisations

Pairplot

```
# Creating a pairplot
sns.pairplot(aviation_df_drpd)
```

<seaborn.axisgrid.PairGrid at 0x7f74be466320>



Observation: There are some relations viewed between number of engines and variables such total fatal, serious injuries and uninjured,total passengers

Heatmap

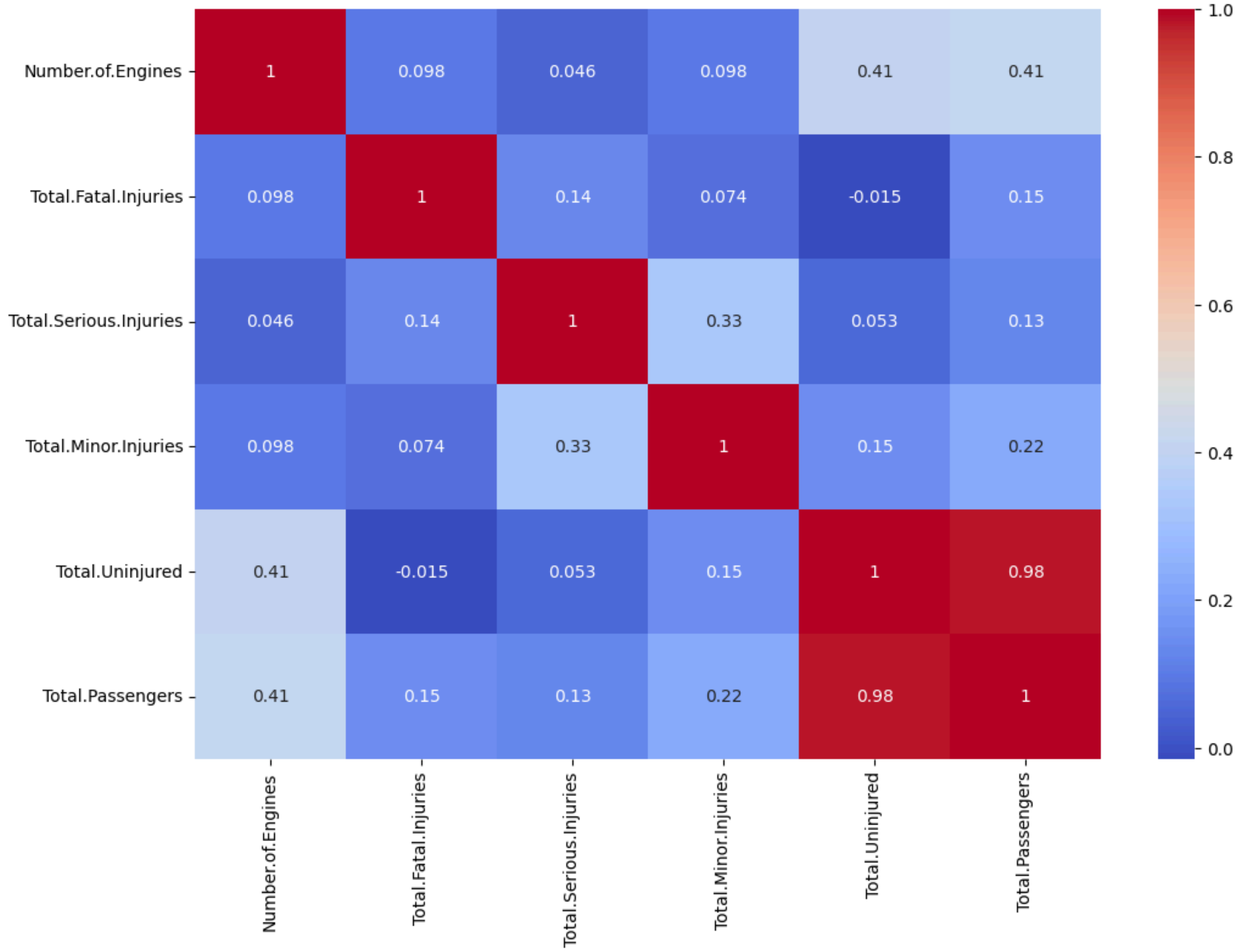
Checking for correlation between the int and float variables

```
#viewing correlation between int and float data
aviation_df.select_dtypes(include=['int', 'float']).corr()
```

	Number.of.Engines	Total.Fatal.Injuries	Total.Serious.Injuries	To
Number.of.Engines	1.000000	0.098505	0.046157	
Total.Fatal.Injuries	0.098505	1.000000	0.135724	
Total.Serious.Injuries	0.046157	0.135724	1.000000	
Total.Minor.Injuries	0.098162	0.073559	0.326849	
Total.Uninjured	0.406058	-0.015214	0.052869	

```
# Visualization heatmap
plt.figure(figsize=(12,8))
sns.heatmap(aviation_df_drpd.select_dtypes(include=['int', 'float']).corr(),annot=True, cmap
```

```
<Axes: >
```



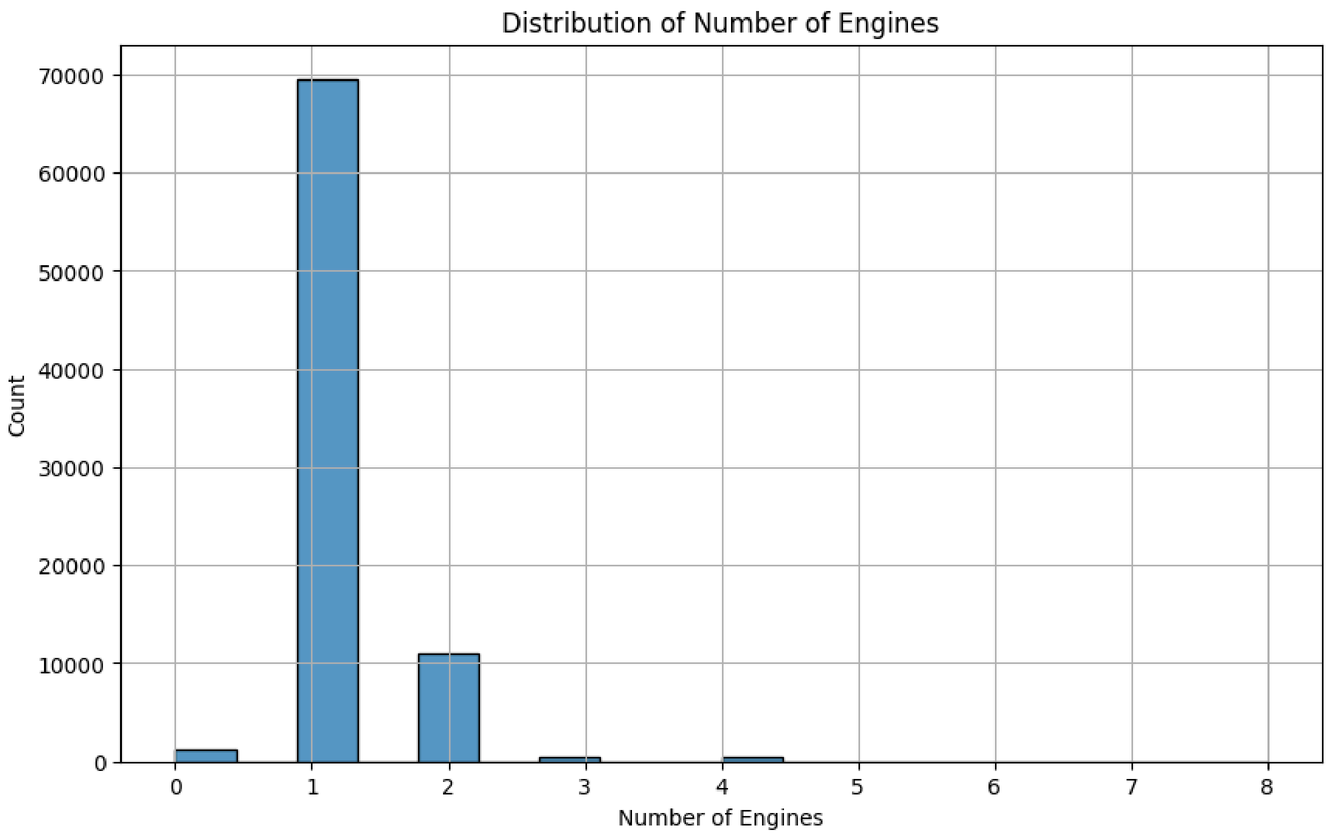
Observation

- Not much correlation between the int and float columns

Histograms

Identifying the distribution of number of engines in the accident log

```
# Histogram
plt.figure(figsize=(10, 6))
sns.histplot(aviation_df_drpd['Number.of.Engines'])
plt.title('Distribution of Number of Engines')
plt.xlabel('Number of Engines')
plt.ylabel('Count')
plt.grid(True)
plt.show()
```



Observation

- Majority of aircrafts that were in accidents had 1 engine
- Majority of aircrafts had 1 or 2 engines

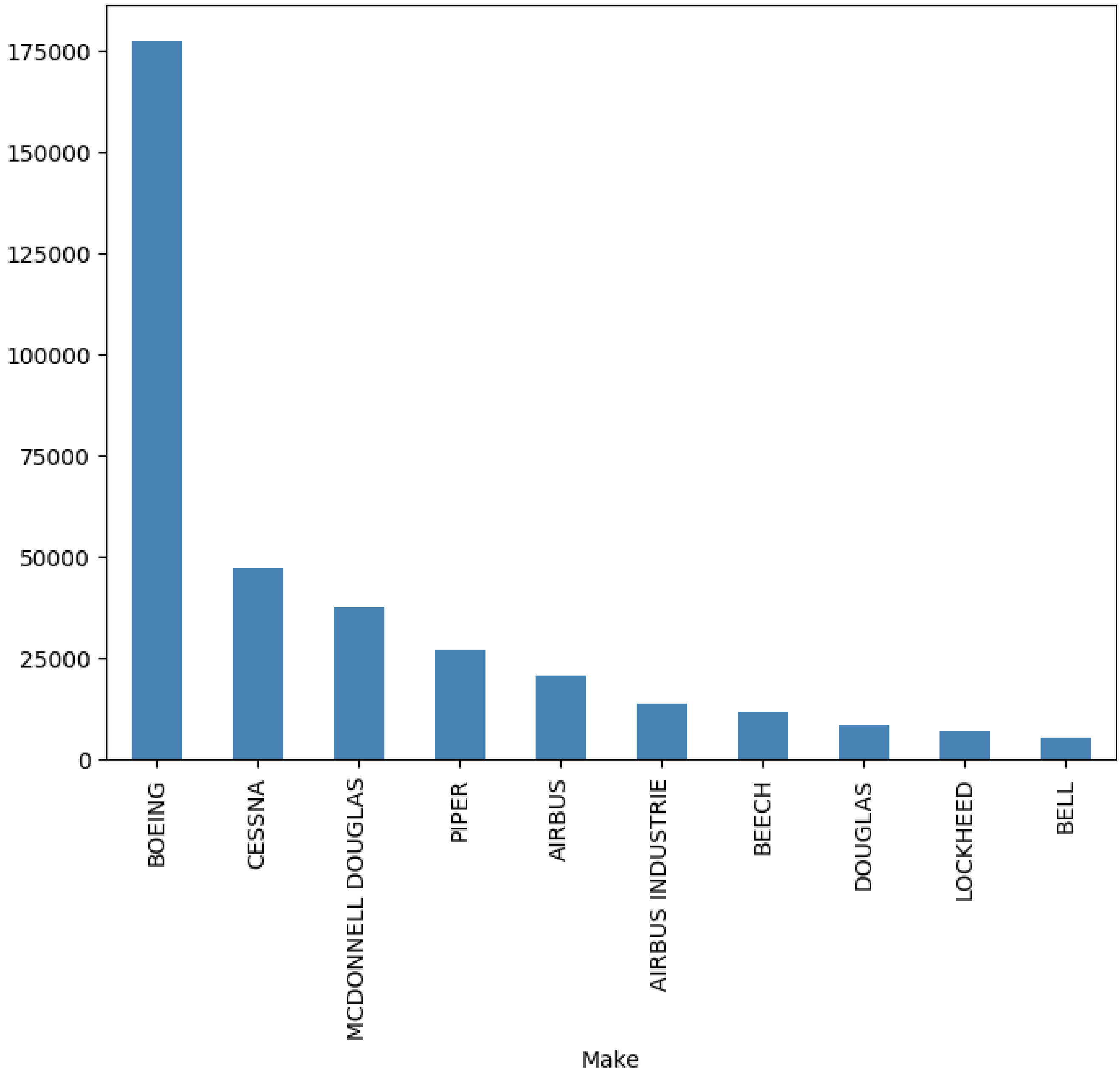
BarGraphs

BarGraph 1

Visualising the top 10 carriers

```
# Creating a bar graph for total passengers per make (top 10)
plt.figure(figsize=(8, 6))  # Set figure size
top_10_carriers.plot(kind='bar', color='steelblue')
```

```
<Axes: xlabel='Make'>
```



Observation:

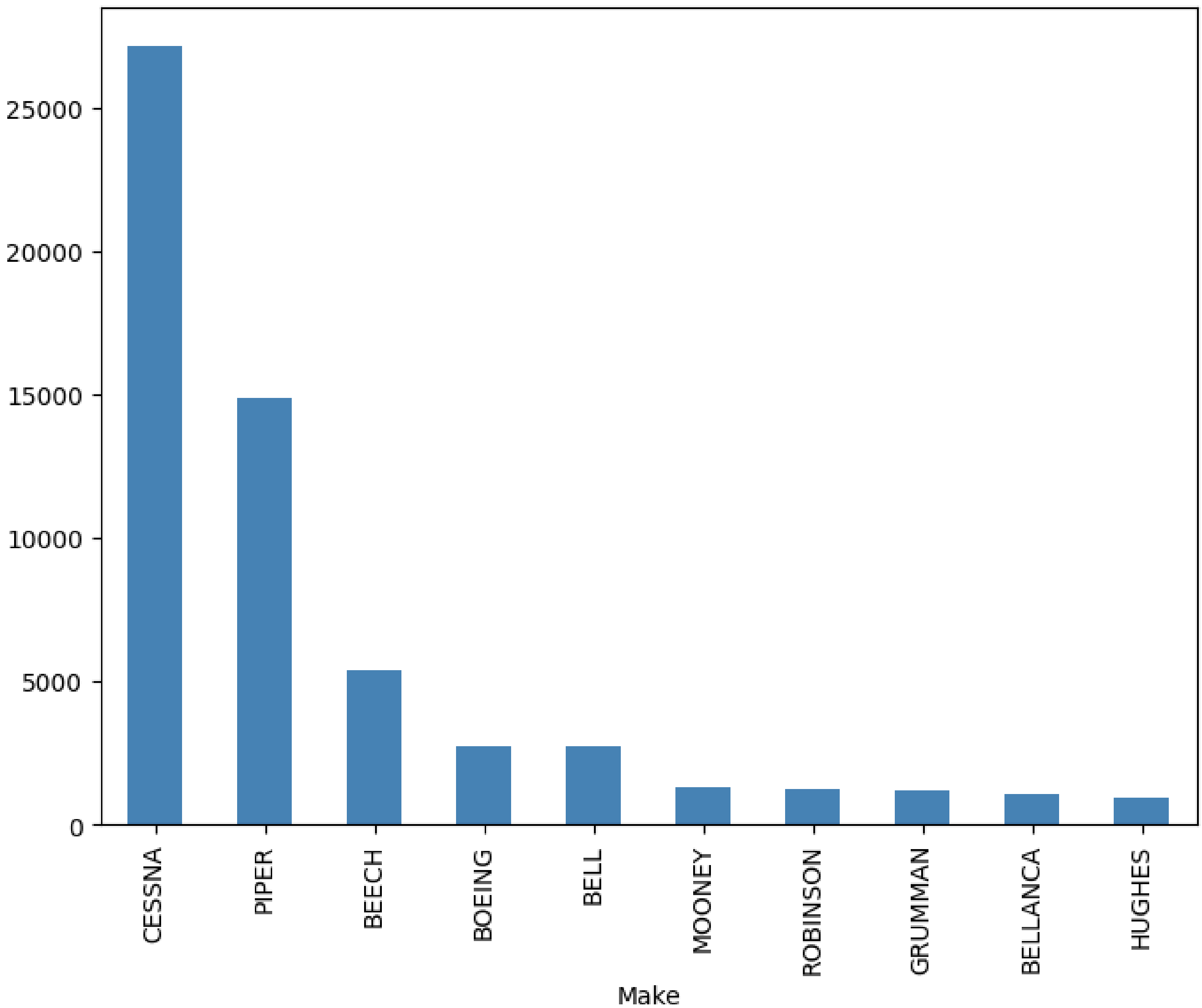
- This shows that Boeing has the highest number of passengers per aircraft

BarGraph2

Largest number of a particular on the Dataset

```
# Creating a bar graph for total count of airraft data per make (top 10)
plt.figure(figsize=(8, 6))  # Set figure size
top_10_Make.plot(kind='bar', color='steelblue')
```

```
<Axes: xlabel='Make'>
```



Observation:

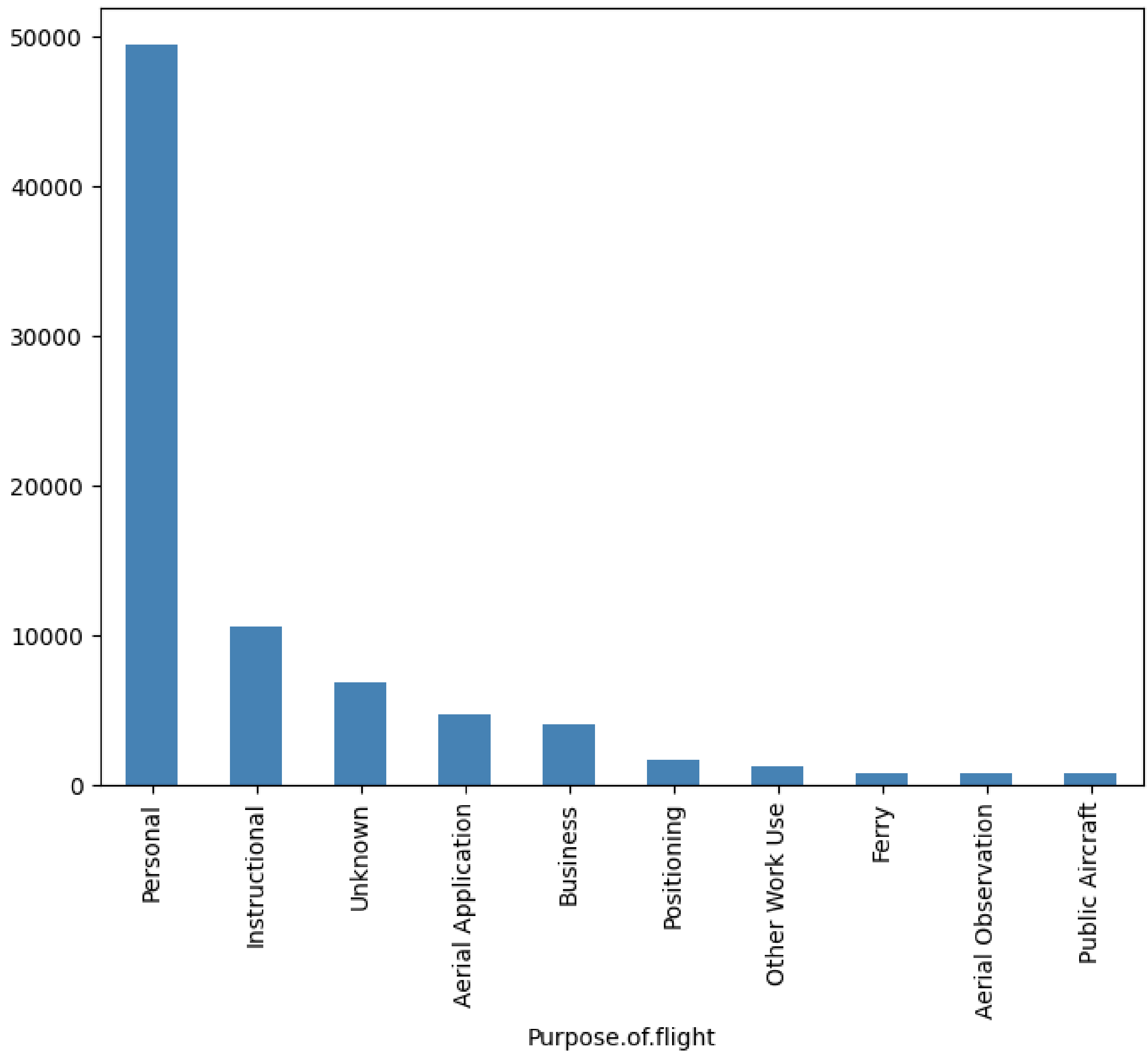
- This shows that the CESSNA aircraft has high tendencies to have accidents as its the most populated on the dataset

BarGraph3

Major Purpose of flight

```
#Identifying the major purpose of flights
plt.figure(figsize=(8, 6))  # Set figure size
aviation_df_drpd['Purpose.of.flight'].value_counts().head(10).plot(kind='bar', color='steelt
```

```
<Axes: xlabel='Purpose.of.flight'>
```



Observation Most aircraft accidents are from personal use. This could be due to training or recreational activities

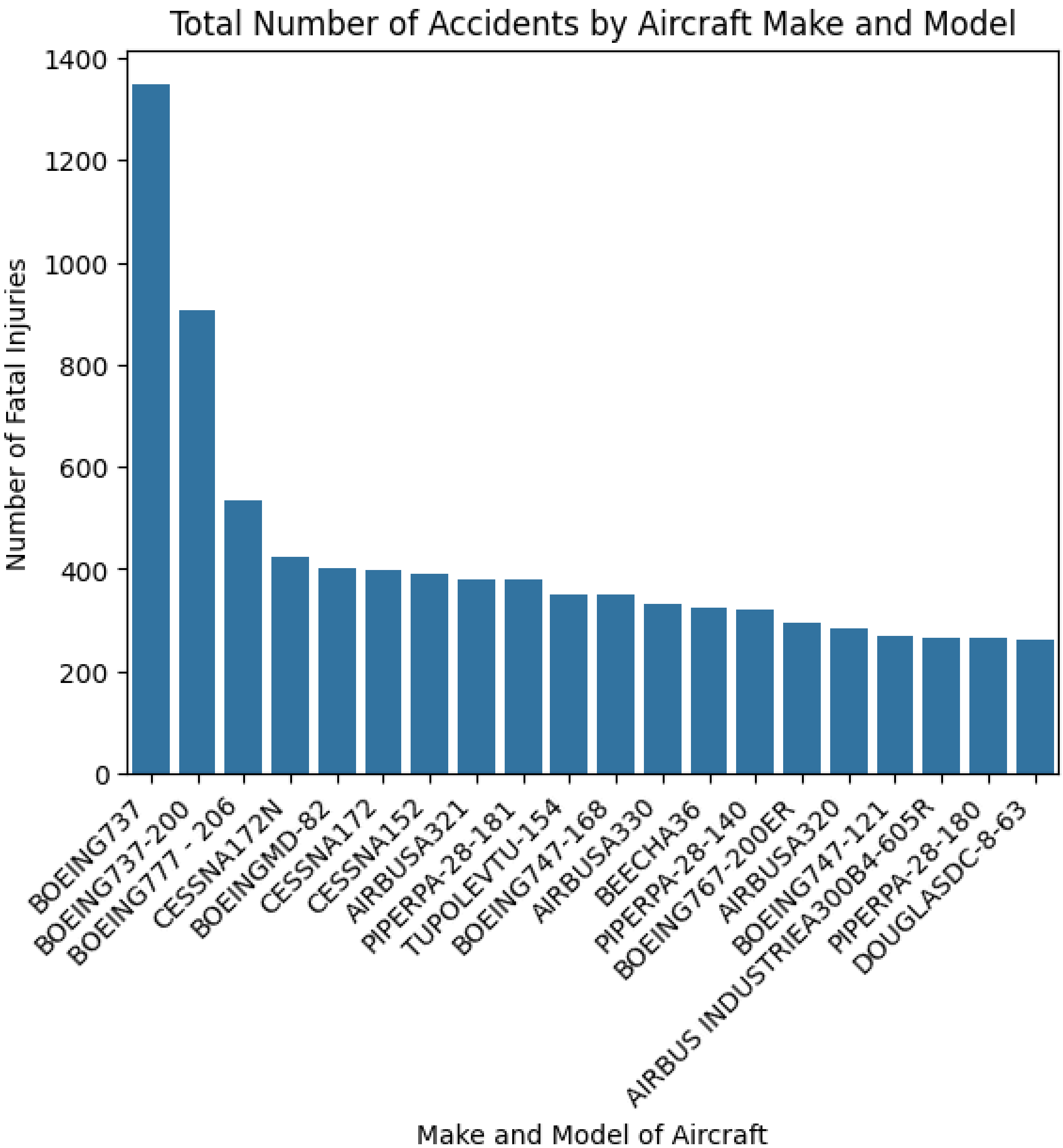
Objective 1

Which is the safest or most unsafe aircrafts and why

Aircraft with sum of the highest fatal injuries

```
# Combining 'make' and 'model' into a single column for grouping
aviation_df_drpd['Make_Model'] = aviation_df_drpd['Make'].str.upper() + aviation_df_drpd['Mc
# Group by the combined 'make_model' and sum based on fatal injuries
accident_sum = aviation_df_drpd.groupby('Make_Model')['Total.Fatal.Injuries'].sum().reset_ir

top_10_sum_accidents = accident_sum.sort_values(by='Accident_sum', ascending=False).head(20)
# Plotting the bar graph
sns.barplot(x='Make_Model', y='Accident_sum', data=top_10_sum_accidents)

plt.title('Total Number of Accidents by Aircraft Make and Model')
plt.xlabel('Make and Model of Aircraft')
plt.ylabel('Number of Fatal Injuries')


plt.tight_layout()
#orienting the labels on the axis rotated to 45 degrees
plt.xticks(rotation=45, ha='right')
plt.show()
```

```
<ipython-input-43-be699623beda>:2: 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_
  aviation_df_drpd['Make_Model'] = aviation_df_drpd['Make'].str.upper() + aviation_df_dr
```



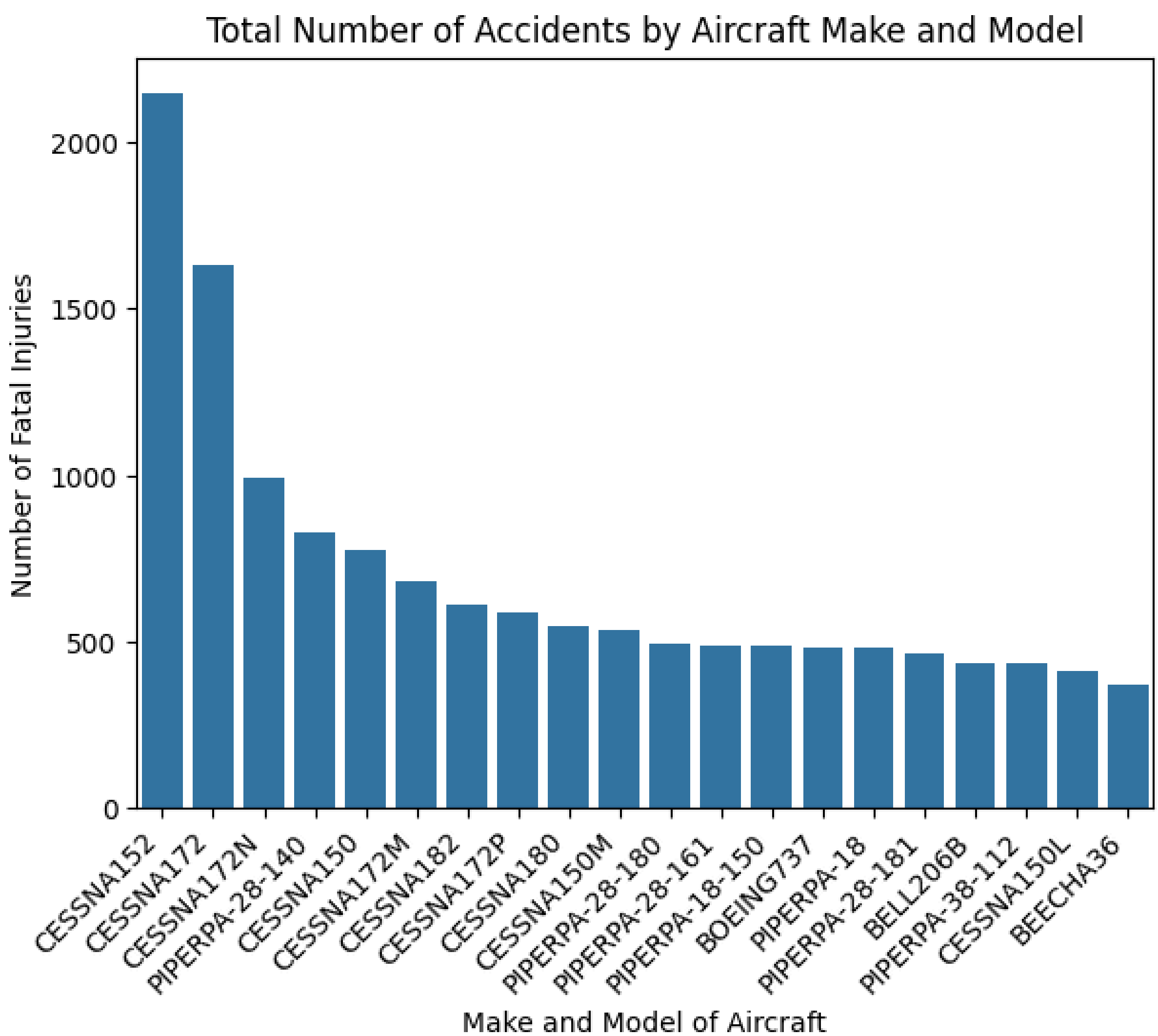
Double-click (or enter) to edit

Observation

1. Boeing 737 aircrafts have the highest number of casulaties - Boeing typically has more passengers and therefore it is expected to have more casualities

Aircraft with count of the highest fatal injuries

```
# Group by the combined 'make_model' and count based on fatal injuries
accident_counts = aviation_df_drpd.groupby('Make_Model')['Total.Fatal.Injuries'].count().res

top_10_count_accidents = accident_counts.sort_values(by='Accident_count', ascending=False).h
# Plotting the bar graph
sns.barplot(x='Make_Model', y='Accident_count', data=top_10_count_accidents)

plt.title('Total Number of Accidents by Aircraft Make and Model')
plt.xlabel('Make and Model of Aircraft')
plt.ylabel('Number of Fatal Injuries')


plt.tight_layout()
#orienting the labels on the axis rotated to 45 degrees
plt.xticks(rotation=45, ha='right')
plt.show()
```



Double-click (or enter) to edit

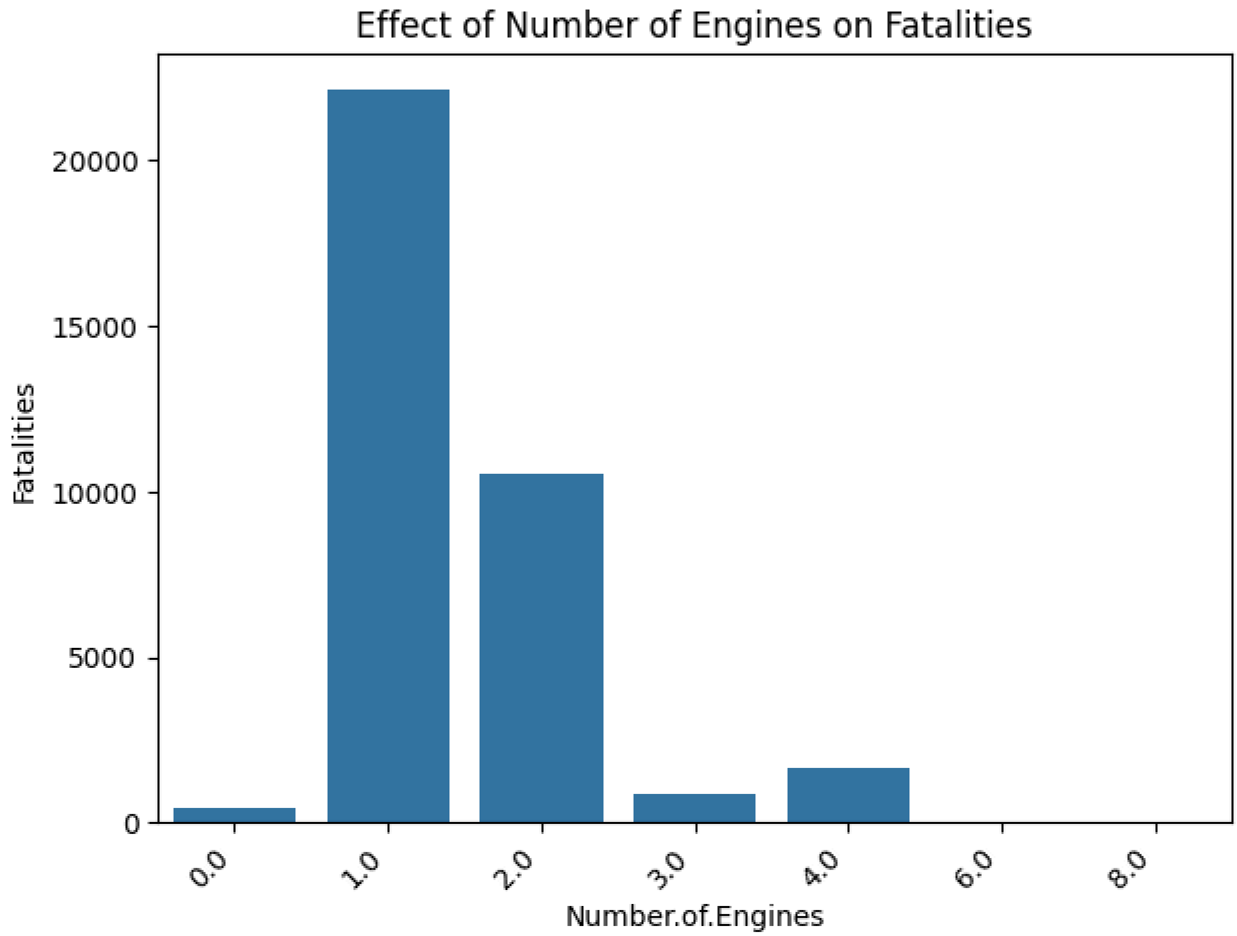
Observation CESSNA aircrafts have a higher count on accidents this is expected since they are smaller aircrafts used for training and personal use as seen in BarGraph3 above

Correlation between number of engines and total fatalities

```
#Identifying the correlation between the number of fatalities
engine_corr = aviation_df_drpd.groupby('Number.of.Engines')['Total.Fatal.Injuries'].sum().re
engine_corr.sort_values(by='Fatalities',ascending=False).head(10)
```

	Number.of.Engines	Fatalities
1	1.0	22123.0
2	2.0	10518.0
4	4.0	1660.0
3	3.0	878.0
0	0.0	409.0
5	6.0	0.0
6	8.0	0.0

```
# Creating a barplot of number of engines vs Total Fatal Injuries
sns.barplot(x='Number.of.Engines', y='Fatalities', data = engine_corr)

plt.title('Effect of Number of Engines on Fatalities')



plt.tight_layout()
#orienting the labels on the axis rotated to 45 degrees
plt.xticks(rotation=45, ha='right')
plt.show()
```



Observation:

- Aircrafts with one engine had a higher number of fatalities then 2 engines

1. This is because for personal flights, which have one engine(typically fewer passengers), there is higher likelihood all passengers to have fatal injuries
2. For aircraft with 2 engines, the likelihood to have high fatalities is because the aircraft has more pssengers and hence increased fatalities

Make of aircraft vs Number of engines

```
#This shows the breakdown of the Makes vs Number of engines
counts = aviation_df_drpd.groupby(['Number.of.Engines', 'Make']).size().reset_index(name='co

# Sort by count in descending order and take top 20
top_20 = counts.sort_values('count', ascending=False).head(20)

# Print the results in a formatted way
print("\nTop 20 Aircraft by Make and Engine Count:")
print("========================================")
```

```
# Ranking list
for idx, row in top_20.iterrows():
    rank = idx + 1
    print(f"{rank:2d}. {row['Make']} ({row['Number.of.Engines']} Number.of.Engines) - {row['
```

```
Top 20 Aircraft by Make and Engine Count:
========================================
1322. CESSNA (1.0 Number.of.Engines) - 23845 aircraft
5048. PIPER (1.0 Number.of.Engines) - 12219 aircraft
766. BEECH (1.0 Number.of.Engines) - 3107 aircraft
7265. CESSNA (2.0 Number.of.Engines) - 2299 aircraft
783. BELL (1.0 Number.of.Engines) - 2270 aircraft
7445. PIPER (2.0 Number.of.Engines) - 2133 aircraft
7228. BEECH (2.0 Number.of.Engines) - 1985 aircraft
4559. MOONEY (1.0 Number.of.Engines) - 1301 aircraft
2802. GRUMMAN (1.0 Number.of.Engines) - 1056 aircraft
812. BELLANCA (1.0 Number.of.Engines) - 1027 aircraft
5496. ROBINSON (1.0 Number.of.Engines) - 1016 aircraft
7235. BOEING (2.0 Number.of.Engines) - 882 aircraft
3269. HUGHES (1.0 Number.of.Engines) - 881 aircraft
351. AIR TRACTOR (1.0 Number.of.Engines) - 649 aircraft
301. AERONCA (1.0 Number.of.Engines) - 632 aircraft
4251. MAULE (1.0 Number.of.Engines) - 574 aircraft
1344. CHAMPION (1.0 Number.of.Engines) - 502 aircraft
6245. STINSON (1.0 Number.of.Engines) - 437 aircraft
4104. LUSCOMBE (1.0 Number.of.Engines) - 409 aircraft
6395. TAYLORCRAFT (1.0 Number.of.Engines) - 376 aircraft
5806. SCHWEIZER (1.0 Number.of.Engines) - 372 aircraft
941. BOEING (1.0 Number.of.Engines) - 366 aircraft
4778. NORTH AMERICAN (1.0 Number.of.Engines) - 346 aircraft
3128. HILLER (1.0 Number.of.Engines) - 340 aircraft
2189. ENSTROM (1.0 Number.of.Engines) - 287 aircraft
7524. BOEING (3.0 Number.of.Engines) - 264 aircraft
314. AEROSPATIALE (1.0 Number.of.Engines) - 261 aircraft
5508. ROCKWELL (1.0 Number.of.Engines) - 259 aircraft
1807. DE HAVILLAND (1.0 Number.of.Engines) - 257 aircraft
7411. MCDONNELL DOUGLAS (2.0 Number.of.Engines) - 231 aircraft
5497. ROBINSON HELICOPTER (1.0 Number.of.Engines) - 228 aircraft
7547. BOEING (4.0 Number.of.Engines) - 226 aircraft
628. AYRES (1.0 Number.of.Engines) - 224 aircraft
2808. GRUMMAN AMERICAN (1.0 Number.of.Engines) - 221 aircraft
1416. CIRRUS DESIGN CORP (1.0 Number.of.Engines) - 221 aircraft
277. AERO COMMANDER (1.0 Number.of.Engines) - 217 aircraft
192. SCHWEIZER (0.0 Number.of.Engines) - 211 aircraft
352. AIR TRACTOR INC (1.0 Number.of.Engines) - 205 aircraft
5501. ROBINSON HELICOPTER COMPANY (1.0 Number.of.Engines) - 191 aircraft
7189. AERO COMMANDER (2.0 Number.of.Engines) - 187 aircraft
2220. EUROCOPTER (1.0 Number.of.Engines) - 178 aircraft
7306. EMBRAER (2.0 Number.of.Engines) - 167 aircraft
7203. AIRBUS (2.0 Number.of.Engines) - 167 aircraft
2193. ERCOUPE (ENG & RESEARCH CORP.) (1.0 Number.of.Engines) - 160 aircraft
7499. SWEARINGEN (2.0 Number.of.Engines) - 160 aircraft
5071. PITTS (1.0 Number.of.Engines) - 148 aircraft
3863. LAKE (1.0 Number.of.Engines) - 142 aircraft
```

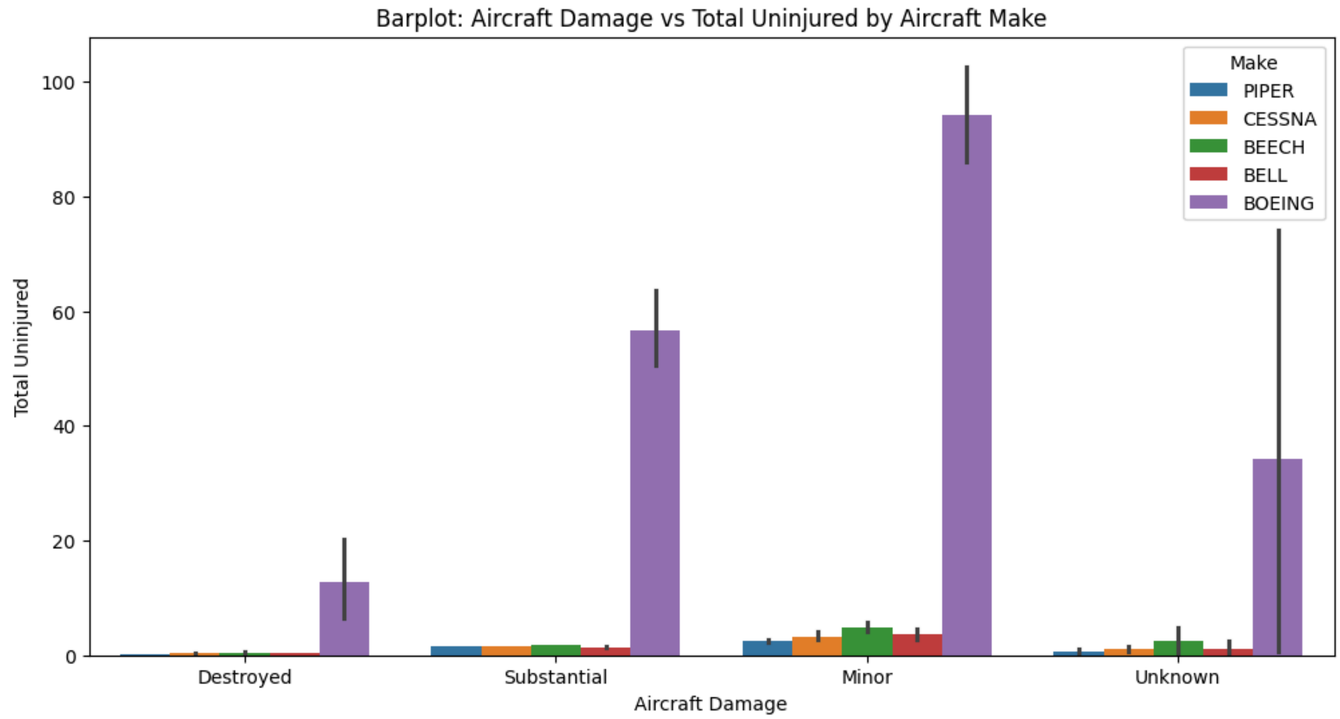
```
6801. WACO (1.0 Number.of.Engines) - 140 aircraft
7299. DOUGLAS (2.0 Number.of.Engines) - 138 aircraft
606. AVIAT (1.0 Number.of.Engines) - 138 aircraft
```

Observation:

- From the above data we see that the BEECH, then BOEING aircrafts have the highest number of aircrafts with 2 engines and from Bar graph 1, they both have the highest number of passengers for a 2 engine aircraft that are large carriers

Aircraft with the most damage and with the least uninjured passengers

```
# Getting the top 10 aircraft makes based on frequency
top_5_makes = aviation_df_drpd['Make'].value_counts().head(5).index

# Filtering the data to include only the top 10 makes
df_top_5 = aviation_df_drpd[aviation_df_drpd['Make'].isin(top_5_makes)]

# Creating a boxplot showing Aircraft Damage vs Total Uninjured, colored by Aircraft Make (h
plt.figure(figsize=(12, 6))
sns.barplot(x="Aircraft.damage", y="Total.Uninjured", hue="Make", data=df_top_5)

# Title and labels
plt.title("Barplot: Aircraft Damage vs Total Uninjured by Aircraft Make")
plt.xlabel("Aircraft Damage")
plt.ylabel("Total Uninjured")

#Plot
plt.show()
```



Observation:

- From the above graph we see that BOEING followed by BEECH have the highest number of uninjured passengers relative to the aircraft damage

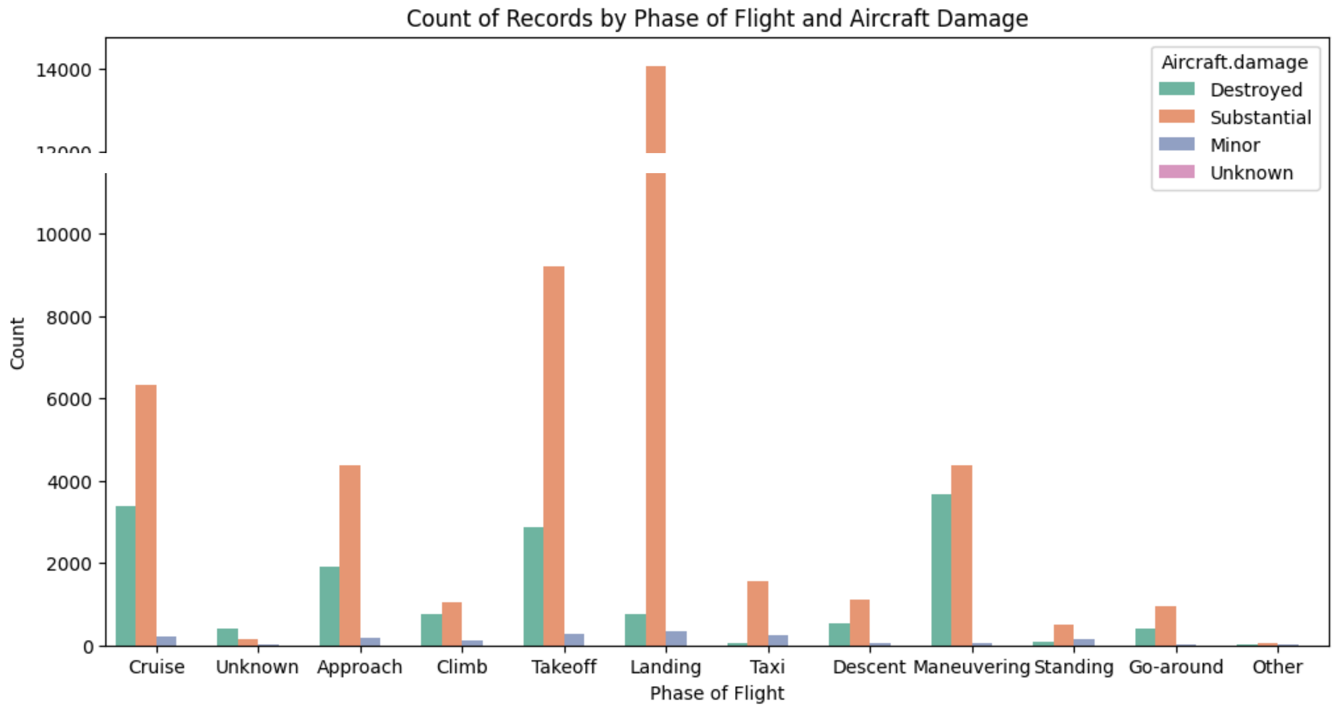
Objective 2

What is the major cause of accidents

Phase of flight that causes most accidents

```
# Creating a countplot showing the count of records by Phase of Flight, with aircraft damage
plt.figure(figsize=(12, 6))
sns.countplot(x="Broad.phase.of.flight", hue="Aircraft.damage", data=aviation_df_drpd, palet

# Title and labels
```

```
plt.title("Count of Records by Phase of Flight and Aircraft Damage")
plt.xlabel("Phase of Flight")
plt.ylabel("Count")

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



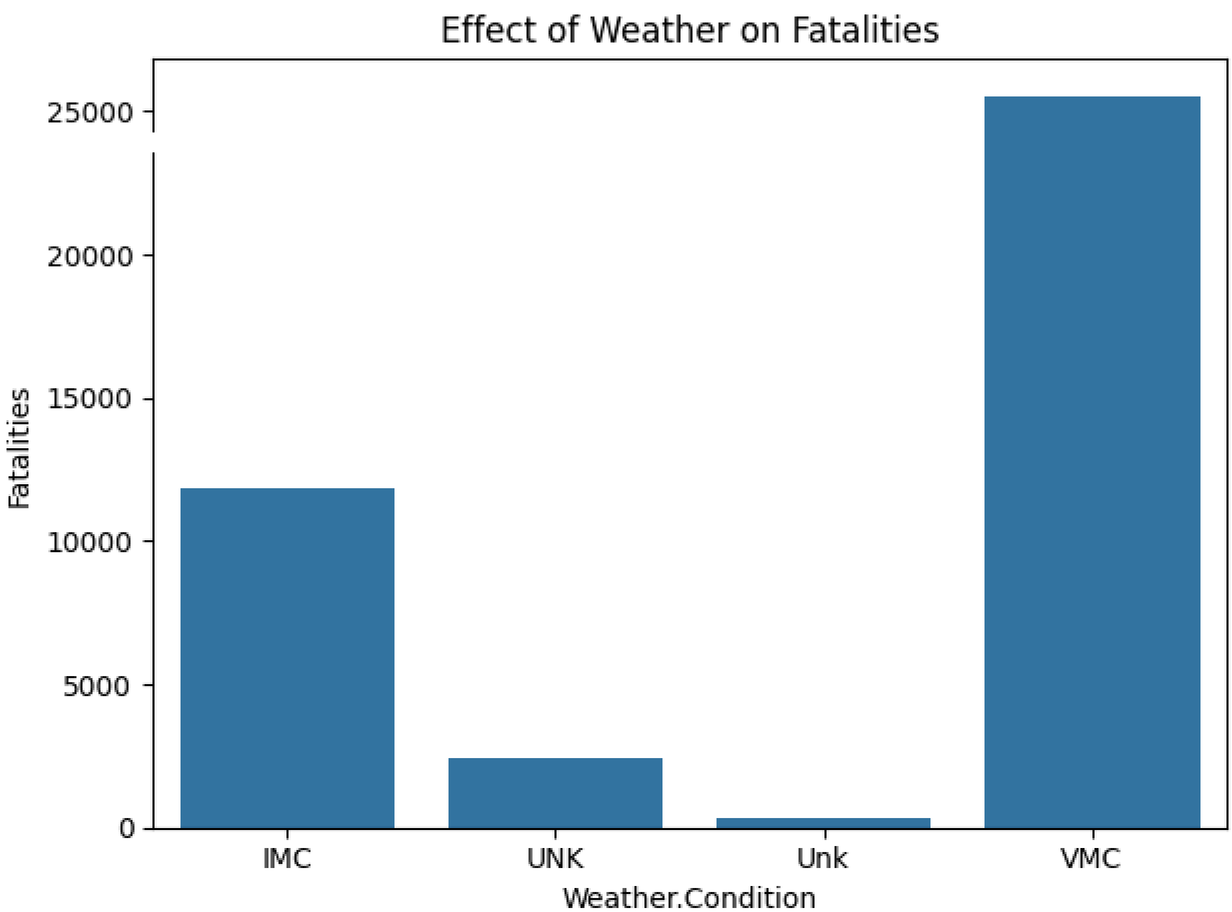
Observation:

- The above data shows that landing is the highest cause of accidents in relation to aircraft damage

Effect of weather on fatalities

```
#Grouping fatalities by wwether condition
weather_eff = aviation_df_drpd.groupby('Weather.Condition')['Total.Fatal.Injuries'].sum().re

weather_eff.sort_values(by='Fatalities',ascending=False).head(10)
```

```
# Creating a barplot of number of engines vs Total Fatal Injuries
sns.barplot(x='Weather.Condition', y='Fatalities', data = weather_eff)

plt.title('Effect of Weather on Fatalities')

plt.tight_layout()

plt.show()
```



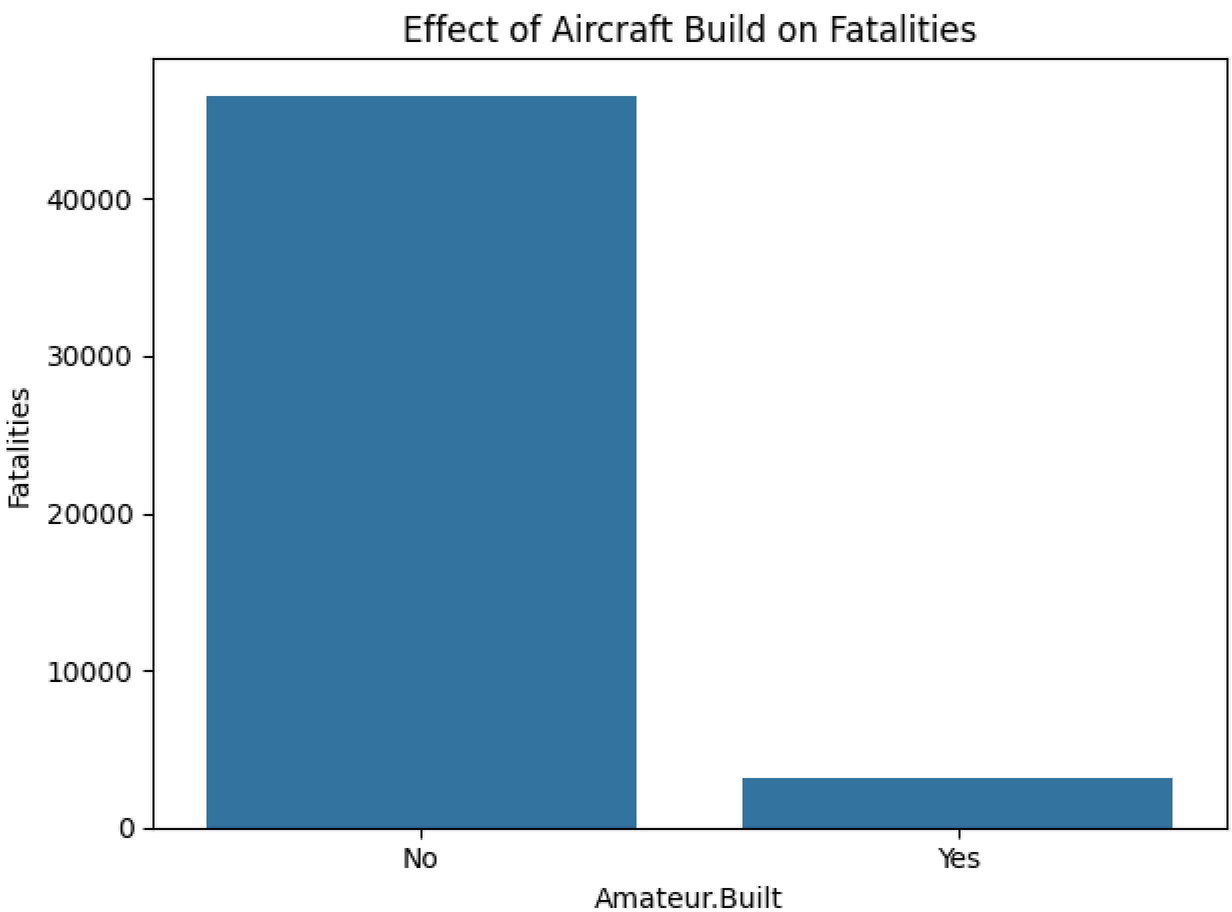
Observation:

- VNC - Visual Navigation Conditions -conditions that are suitable for visual navigation.
- IMC - Instrument Meteorological Conditions -conditions that are below the minimums required for visual flight.
- UNK - UNK is a shorthand for "unknown", and it is used when the data about a certain condition cannot be determined.

From the above we can see that the weather had little to no effect on the fatalities. Fatalities were realised even in perfect weather conditions

Effect of build on Fatalities

```
#Grouping fatalities by aircraft build
aircraft_build = aviation_df_drpd.groupby('Amateur.Built')['Total.Fatal.Injuries'].sum().res

aircraft_build.sort_values(by='Fatalities',ascending=False).head(10)
# Creating a barplot of number of engines vs Total Fatal Injuries
sns.barplot(x='Amateur.Built', y='Fatalities', data = aircraft_build)

plt.title('Effect of Aircraft Build on Fatalities')

plt.tight_layout()

plt.show()
```



Observation: