BUSINESS UNDERSTANDING

The following data analysis aims to assist the Head of Aviation in minimizing risks while exploring a new industry to diversify the company's portfolio. Specifically, the company is interested in acquiring and operating airplanes for both commercial and private enterprises. However, it currently lacks insight into the potential risks associated with aircraft operations.

Objectives

1. Determine which aircraft has the lowest risk for the company to start this new business endeavor.
2. Translate the findings into actionable insights that drive the choice of which aircraft to purchase.

Success Criteria

Atleast three concrete business recommendations for how the business should move forward with the new aviation opportunity.

Questions to guide the analysis

- What defines a low risk aircraft according to the data provided?
- Where can we implement strategies in anticipation of risk within the variables in the data?

DATA UNDERSTANDING

The data used in the analysis is from NTSB aviation accident database and 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.

In [38]:
```
# Importing libraries
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
```

In [39]:
```
# Loading the data using pandas
df = pd.read_csv('AviationData.csv', encoding='latin-1')
```

In [40]:
```
# Preview of the data set
df.head()
```

Out[40]:

	Event.Id	Investigation.Type	Accident.Number	Event.Date	Location	Country	Latitude	Longitude	Airport.Code	Airport.Name	...	Purpose.
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	36.9222	-81.8781	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 [41]:
```
df.tail()
```

Out[41]:

	Event.Id	Investigation.Type	Accident.Number	Event.Date	Location	Country	Latitude	Longitude	Airport.Code	Airport.Name	...	Purpo
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 [42]:
```
# More info on the dataset
df.info()
```

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

- From the data, we can define a low risk aircraft by the number of events they have been involved in. Are a specific type of aircraft more prone to accidents?
- Immediately, we can see some columns that we can term as irrelevant to our analysis i.e 'Airport.Code', 'Airport.Name', 'Latitude','Longitude','Accident.Number' and others. We can focus majorly on attributes relating to type of aircraft and outcomes of events.
- We can also see some columns whose values are majorly null values.
- Most of the columns have categorical data, a few have continuous data. Investigating the continuous data:

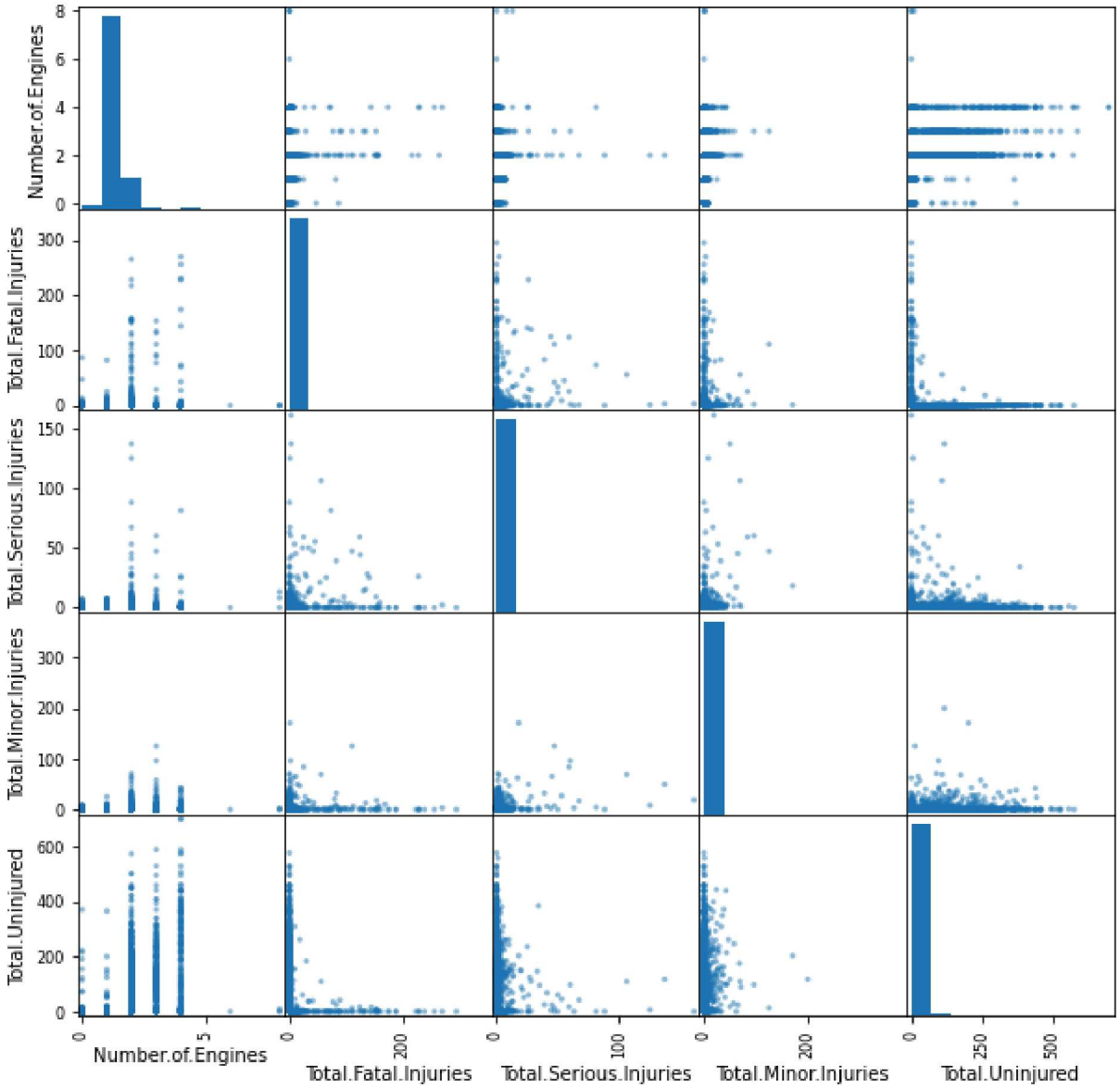
In [43]:
```
# Summary of the continuous data
df.describe()
```

Out[43]:

	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

We can see that the number of serious, minor and fatal injuries are almost at per.

In [44]:
```
# Relationship between the continuous data
pd.plotting.scatter_matrix(df, figsize=(10,10));
```



- There is no correlation between number of engines, and how many people get hurt
- There is no correlation between total fatal injuries and the total number of uninjured people, indicating that in the case of a fatal accident, all passengers must be classified as either seriously or mildly injured except in one off situations.

There are some columns that will not be helpful in our analysis. We can remove them from the data.

In [45]:
```
# Dropping irrelevant columns
columns_to_drop = ['Accident.Number','Latitude','Longitude','Airport.Code','Airport.Name','FAR.Description',
                   'Schedule','Air.carrier','Publication.Date','Report.Status','Registration.Number']
df = df.drop(columns_to_drop, axis = 1)
```

DATA PREPARATION

Data cleaning

We will start by checking for duplicates before proceeding to dealing with null values.

In [46]:
```
# Checking for duplicates
duplicates = df[df.duplicated()]
duplicates.head()
```

Out[46]:

	Event.Id	Investigation.Type	Event.Date	Location	Country	Injury.Severity	Aircraft.damage	Aircraft.Category	Make	Model	Am
1371	20020917X02935	Accident	1982-05-28	EVANSVILLE, IN	United States	Non-Fatal	Substantial	Airplane	Cessna	172	
3082	20020917X04638	Accident	1982-10-18	GULF OF MEXICO	GULF OF MEXICO	Fatal(3)	Destroyed	Helicopter	Bell	206B	
4761	20001214X43016	Accident	1983-05-22	BRIDGEPORT, CA	United States	Fatal(1)	Substantial	NaN	Schempp-hirth	STANDARD CIRRUS	
7941	20001214X39345	Accident	1984-04-13	DELAND, FL	United States	Non-Fatal	Substantial	NaN	Cessna	172P	
8661	20001214X40069	Accident	1984-06-18	PORTLAND, AR	United States	Non-Fatal	Substantial	NaN	Snow	AT400	

There are some duplicated values in the dataframe which we can eliminate.

In [47]:
```
# Dropping duplicates
df = df.drop_duplicates()
```

In [48]:
```
# Checking for null values as a %
df.isna().sum()/len(df) * 100
```

Out[48]:
```
Event.Id                  0.000000
Investigation.Type        0.000000
Event.Date                0.000000
Location                  0.058518
Country                   0.254330
Injury.Severity           1.124228
Aircraft.damage           3.591002
Aircraft.Category        63.672477
Make                      0.070897
Model                     0.103532
Amateur.Built             0.114786
Number.of.Engines         6.843272
Engine.Type               7.961873
Purpose.of.flight         6.965936
Total.Fatal.Injuries     12.826774
Total.Serious.Injuries   14.070290
Total.Minor.Injuries     13.420961
Total.Uninjured           6.648586
Weather.Condition         5.053961
Broad.phase.of.flight    30.566840
dtype: float64
```

We can see that 64% of the data in 'Aircraft.Category' is null. Let us investigate that column further.

In [49]:
```
# Checking values in 'Aircraft.Category'
df['Aircraft.Category'].value_counts()
```

Out[49]:
```
Airplane            27613
Helicopter           3438
Glider                508
Balloon               231
Gyrocraft             173
Weight-Shift          161
Powered Parachute      91
Ultralight             30
Unknown                14
WSFT                    9
Powered-Lift            5
Blimp                   4
UNK                     2
ULTR                    1
Rocket                  1
Name: Aircraft.Category, dtype: int64
```

Since this data could be valuable in our analysis, we shall replace the null values with the text 'UFO' meaning unidentified flying object

In [50]:
```
# Replacing null values in 'Aircraft.Category'
df['Aircraft.Category'] = df['Aircraft.Category'].fillna('UFO')
```

In [51]:
```
# Replacing Unknown values in 'Aircraft.Category' with UFO
df['Aircraft.Category'] = df['Aircraft.Category'].replace({'Unknown':'UFO'})
```

In [52]:
```
# Re-checking for null values as a %
df.isna().sum()/len(df) * 100
```

Out[52]:
```
Event.Id                   0.000000
Investigation.Type         0.000000
Event.Date                 0.000000
Location                   0.058518
Country                    0.254330
Injury.Severity            1.124228
Aircraft.damage            3.591002
Aircraft.Category          0.000000
Make                       0.070897
Model                      0.103532
Amateur.Built              0.114786
Number.of.Engines          6.843272
Engine.Type                7.961873
Purpose.of.flight          6.965936
Total.Fatal.Injuries      12.826774
Total.Serious.Injuries    14.070290
Total.Minor.Injuries      13.420961
Total.Uninjured            6.648586
Weather.Condition          5.053961
Broad.phase.of.flight     30.566840
dtype: float64
```

Our next column with 30 % null values is 'Broad.phase.of.flight'. Investigating this column:

In [53]:
```
# Checking count of unique values in 'Broad.phase.of.flight'
df['Broad.phase.of.flight'].value_counts()
```

Out[53]:
```
Landing        15427
Takeoff        12493
Cruise         10265
Maneuvering     8135
Approach        6543
Climb           2033
Taxi            1953
Descent         1887
Go-around       1353
```

```
Standing          943
Unknown           548
Other             119
Name: Broad.phase.of.flight, dtype: int64
```

Since this data could be valuable in our analysis, we will keep the data. However we can spot some more missing values in place holder term 'Unknown'. We shall add the rest of the missing values under this category.

In [54]:
```
# Filling null values in 'Broad.phase.of.flight' with the value 'Unknown'
df['Broad.phase.of.flight'] = df['Broad.phase.of.flight'].fillna('Unknown')
```

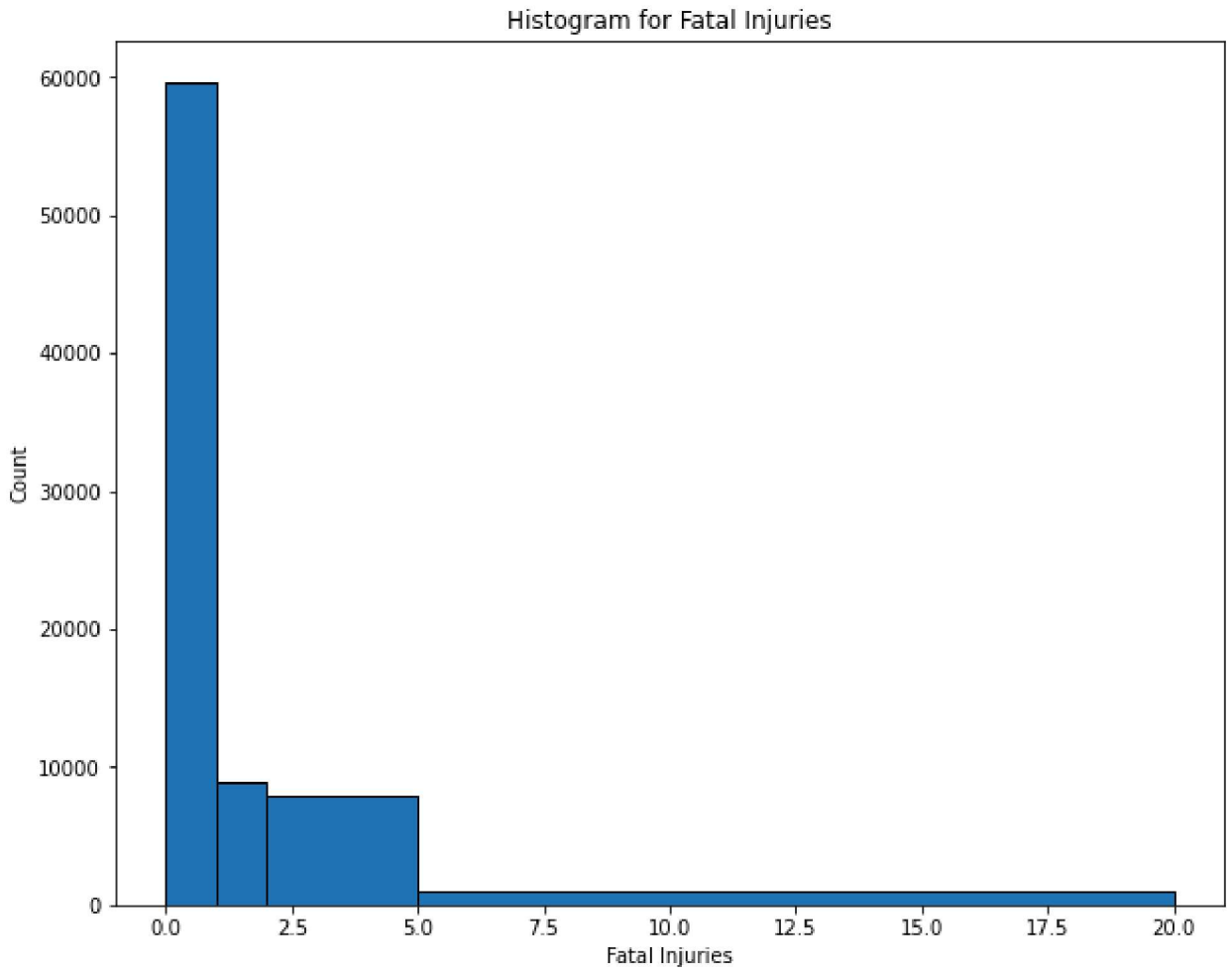
In [55]:
```
# Re-checking for null values as a %
df.isna().sum()/len(df) * 100
```

Out[55]:
```
Event.Id                  0.000000
Investigation.Type        0.000000
Event.Date                0.000000
Location                  0.058518
Country                   0.254330
Injury.Severity           1.124228
Aircraft.damage           3.591002
Aircraft.Category         0.000000
Make                      0.070897
Model                     0.103532
Amateur.Built             0.114786
Number.of.Engines         6.843272
Engine.Type               7.961873
Purpose.of.flight         6.965936
Total.Fatal.Injuries     12.826774
Total.Serious.Injuries   14.070290
Total.Minor.Injuries     13.420961
Total.Uninjured           6.648586
Weather.Condition         5.053961
Broad.phase.of.flight     0.000000
dtype: float64
```

Checking the distributions for 'Total.Fatal.Injuries', 'Total.Serious.Injuries','Total.Minor.Injuries' to gauge how to best fill the null values

In [56]:
```
# Visualizing Total.Fatal.Injuries
fig, ax = plt.subplots(figsize = (10,8))
ax.hist(x = df['Total.Fatal.Injuries'], bins = [0, 1, 2, 5, 20], edgecolor = 'black')
ax.set_title('Histogram for Fatal Injuries')
# x-axis label
ax.set_xlabel('Fatal Injuries')
```

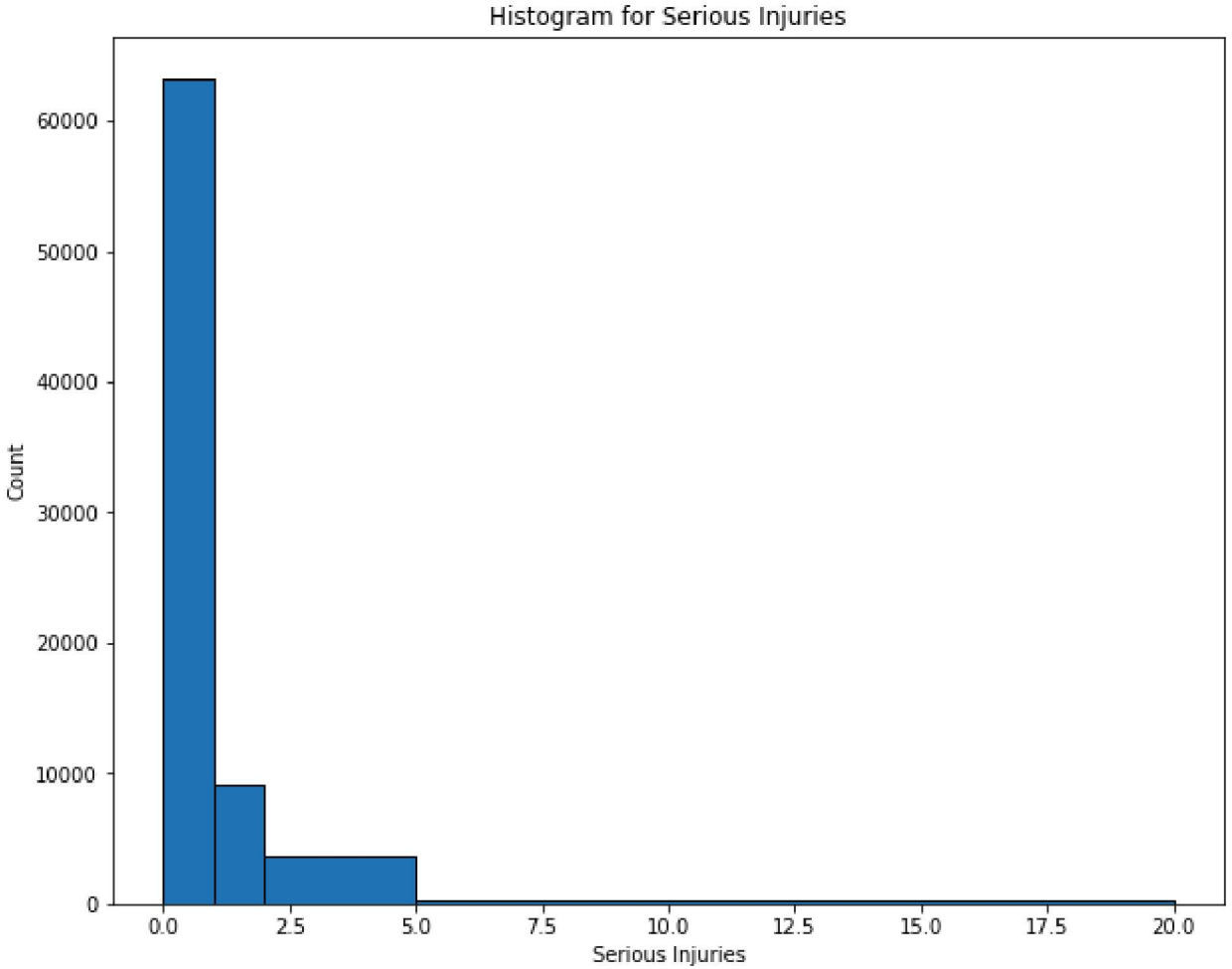
```
# y-axis label
ax.set_ylabel('Count');
```



We can see that the number of fatal injuries per accident are almost always 0. For this column, we shall replace the missing values with the mode.

In [57]:
```
# Calculating mode
fatal_injuries_mode = df['Total.Fatal.Injuries'].mode()[0]

# Replacing null values with mode
df['Total.Fatal.Injuries'] = df['Total.Fatal.Injuries'].fillna(fatal_injuries_mode)
```

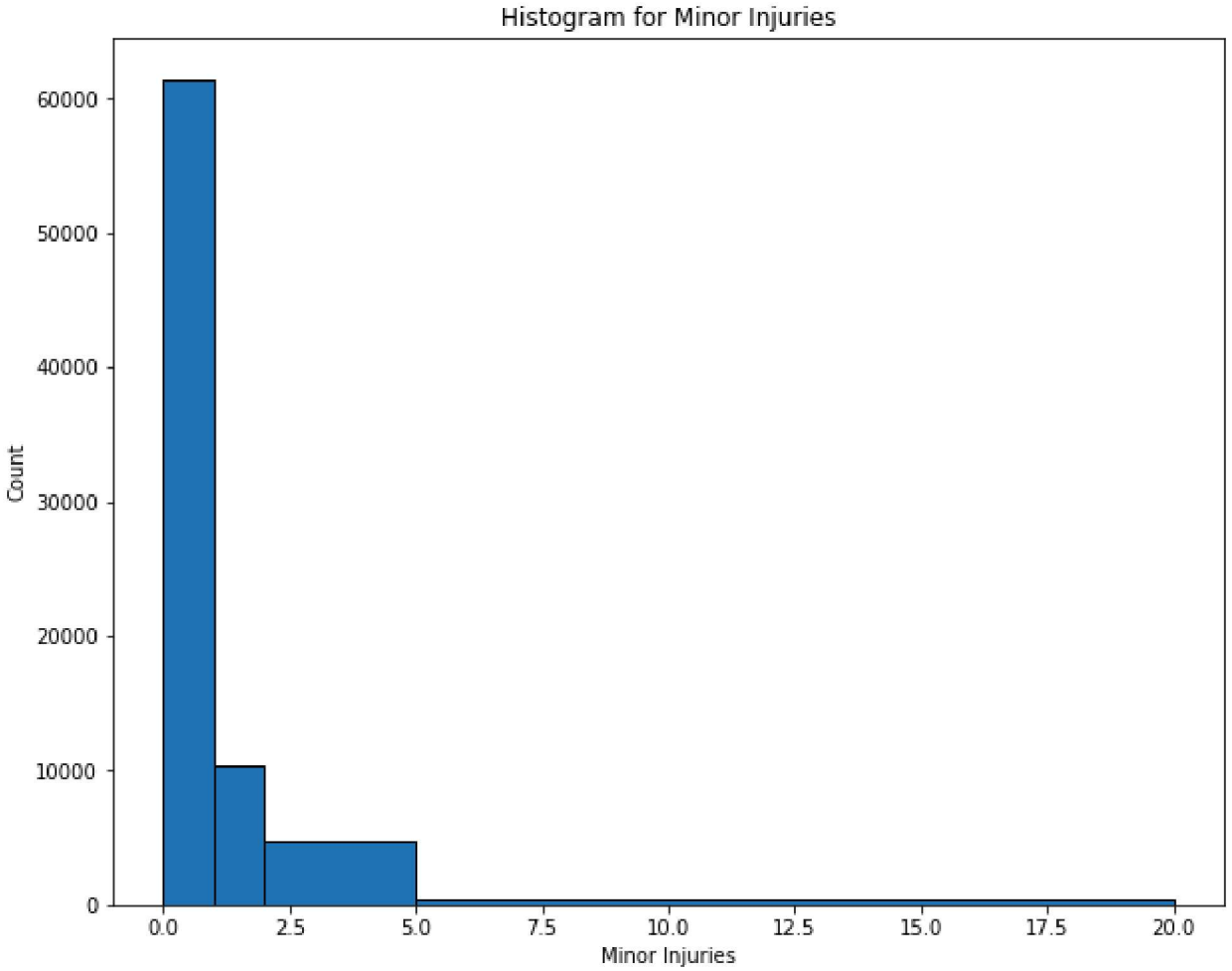
In [58]:
```
# Visualizing Total.Serious.Injuries
fig, ax = plt.subplots(figsize = (10,8))
ax.hist(x = df['Total.Serious.Injuries'], bins = [0, 1, 2, 5, 20], edgecolor = 'black')
ax.set_title('Histogram for Serious Injuries')

# Labelling
ax.set_xlabel('Serious Injuries')
ax.set_ylabel('Count');
```



As was the case with fatal injuries, the mode for this column is between 0-1. This means that cases of serious injuries per aircraft accident is almost always 0. Replacing nan values with the mode makes sense here too.

In [59]:
```
# Calculating mode
serious_injuries_mode = df['Total.Serious.Injuries'].mode()[0]

# Replacing null values with mode
df['Total.Serious.Injuries'] = df['Total.Serious.Injuries'].fillna(serious_injuries_mode)
```

In [60]:
```
# Total.Minor.Injuries
fig, ax = plt.subplots(figsize = (10,8))
ax.hist(x = df['Total.Minor.Injuries'], bins = [0, 1, 2, 5, 20], edgecolor = 'black')
ax.set_title('Histogram for Minor Injuries')

# Labelling
ax.set_xlabel('Minor Injuries')
ax.set_ylabel('Count');
```

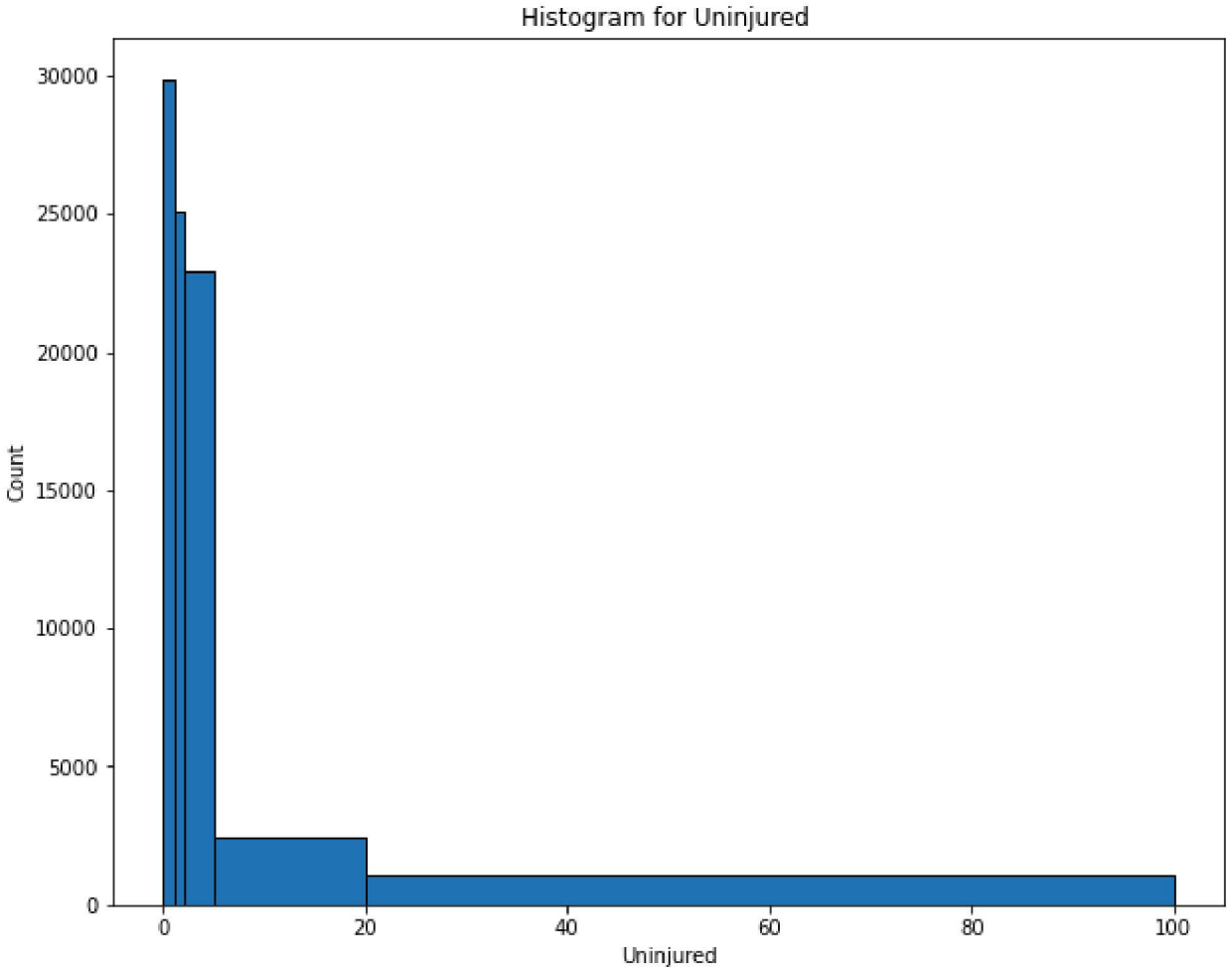


As we did with the serious and fatal columns, the same case applies here. The mode is a sensible placeholder.

In [61]:
```
# Calculating mode
minor_injuries_mode = df['Total.Minor.Injuries'].mode()[0]

# Replacing null values with mode
df['Total.Minor.Injuries'] = df['Total.Minor.Injuries'].fillna(minor_injuries_mode)
```

In [62]:
```
# Total.Uninjured
fig, ax = plt.subplots(figsize = (10,8))
ax.hist(x = df['Total.Uninjured'], bins = [0, 1, 2, 5, 20,100], edgecolor = 'black')
ax.set_title('Histogram for Uninjured ')
```

```
# Labelling
ax.set_xlabel('Uninjured')
ax.set_ylabel('Count');
```



Mode is a sensible placeholder

In [63]:
```
# Calculating mode
uninjured_mode = df['Total.Uninjured'].mode()[0]

# Replacing null values with mode
df['Total.Uninjured'] = df['Total.Uninjured'].fillna(uninjured_mode)
```

In [64]:
```
# Re-checking for null values as a %
df.isna().sum()/len(df) * 100
```

Out[64]:
```
Event.Id                0.000000
Investigation.Type      0.000000
Event.Date              0.000000
Location                0.058518
Country                 0.254330
Injury.Severity         1.124228
Aircraft.damage         3.591002
Aircraft.Category       0.000000
Make                    0.070897
Model                   0.103532
Amateur.Built           0.114786
Number.of.Engines       6.843272
Engine.Type             7.961873
Purpose.of.flight       6.965936
Total.Fatal.Injuries    0.000000
Total.Serious.Injuries  0.000000
Total.Minor.Injuries    0.000000
Total.Uninjured         0.000000
Weather.Condition       5.053961
Broad.phase.of.flight   0.000000
dtype: float64
```

In [65]:
```
# Check how many number of rows still have null values
null_present = df.isna().any(axis=1).sum()
rows_in_data = len(df)

print (f'{null_present} out of {rows_in_data} still have null values \n\n')
print (f'That is {(null_present / rows_in_data * 100).round(2)} % of the data')
```

```
13534 out of 88861 still have null values


That is 15.23 % of the data
```

We can drop a few more columns that seem redundant to avoid losing relevant data when clearing all rows with null values. We can also notice some redundant values in the 'Make' column, where we have the same values repeated in different cases.

In [66]:
```
# Dropping the 'Injury.Severity' column as we can get the same data from the 'Total.Fatal/Serious/Minor.Injuries' columns
df = df.drop('Injury.Severity', axis = 1)
```

In [67]:
```
# Filling missing values for the 'Engine.Type' column with existing placeholder 'None'
```

```
df['Engine.Type'] = df['Engine.Type'].fillna('None')
```

In [68]:
```
# Combining all placeholders values in the 'Engine.Type' column into 'None'

df['Engine.Type'] = df['Engine.Type'].replace({'NONE':'None'})
```

In [69]:
```
# Check how many number of rows still have null values

df.isna().any(axis=1).sum()/len(df)*100
```

Out[69]:
```
12.985449184681693
```

In [70]:
```
# We can now drop all the rest of the missing values.

df = df.dropna()

# Check if we still have any missing values
df.isna().sum()
```

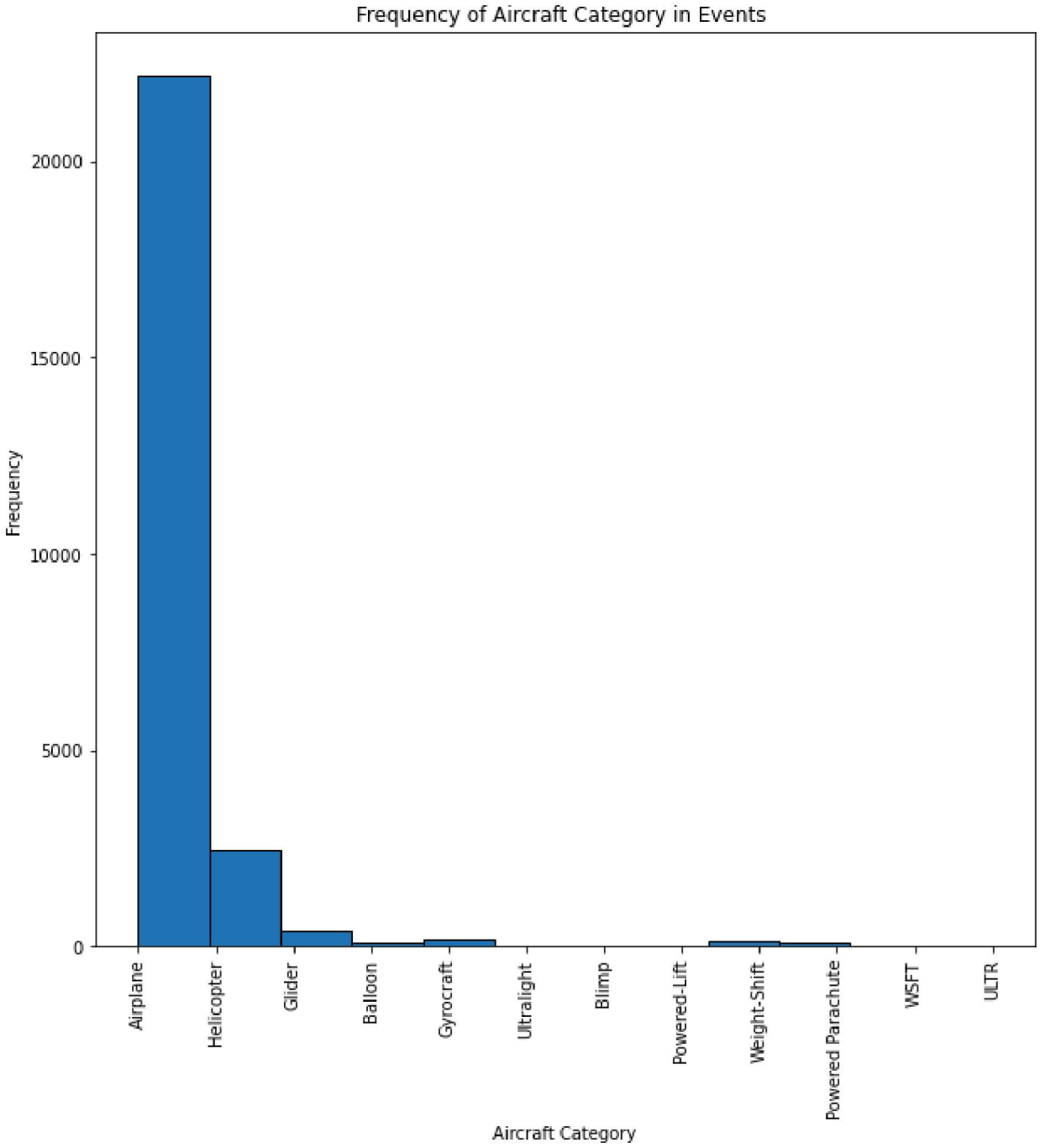
Out[70]:
```
Event.Id                0
Investigation.Type      0
Event.Date              0
Location                0
Country                 0
Aircraft.damage         0
Aircraft.Category       0
Make                    0
Model                   0
Amateur.Built           0
Number.of.Engines       0
Engine.Type             0
Purpose.of.flight       0
Total.Fatal.Injuries    0
Total.Serious.Injuries  0
Total.Minor.Injuries    0
Total.Uninjured         0
Weather.Condition       0
Broad.phase.of.flight   0
dtype: int64
```

In [71]:
```
# Standardizing values in the 'Make' column

df['Make'] = df['Make'].str.title()
```

ANALYSIS

To determine which aircraft are the lowest risk for the company we can start by looking at some correlations of some of the attributes relating to the type of aircraft and comparing them with the accidents extents. Which 'Aircraft.Category' is more likely to be involved in incidents and accidents?

In [72]:
```
# Visualizing the count of 'Aircraft.Category'
fig, ax = plt.subplots(figsize = (10,10))

# Filtering out the rows containing 'UFO'
aircrafts = df[df['Aircraft.Category'] != 'UFO']

# Histogram plot for the 'Aircraft Category'
ax.hist(aircrafts['Aircraft.Category'], bins = 12, edgecolor = 'black')
ax.set_title('Frequency of Aircraft Category in Events ')
ax.set_xlabel('Aircraft Category')
ax.set_ylabel('Frequency')
ax.tick_params(axis='x', labelrotation=90);
```


Frequency of Aircraft Category in Events
Frequency
20000
15000
10000
5000
0
Airplane
Helicopter
Glider
Balloon
Gyrocraft
Ultralight
Blimp
Powered-Lift
Weight-Shift
Powered Parachute
WSFT
ULTR
Aircraft Category

Airplanes appear to be the most common type of aircraft in the dataset for events. This may not necessarily suggest they are the least safe, but rather that they are the most widely used. To test this hypothesis, we will compare injury data across different aircraft categories to determine which has the lowest risk.

In [73]:
```
# Top 5 aircraft categories
top_5_aircraft_category = df['Aircraft.Category'].value_counts().head(5).index.tolist()

# Applying filters to access injury data and 'Aircraft.Category'
category_filtered = df[((df['Aircraft.Category'].isin(top_5_aircraft_category)&(df['Aircraft.Category']!= 'UFO')))
                       &(df[['Total.Uninjured','Total.Minor.Injuries','Total.Serious.Injuries','Total.Fatal.Injuries']].sum(axis=1)

# Create pivot table
pivot = category_filtered.pivot_table(index = 'Aircraft.Category', values = ['Total.Uninjured','Total.Minor.Injuries','Total.Serio
pivot
```

Out[73]:

Aircraft.Category	Total.Fatal.Injuries	Total.Minor.Injuries	Total.Serious.Injuries	Total.Uninjured
Airplane	7021	4169	4727	31050
Glider	64	92	95	288
Gyrocraft	42	29	53	84
Helicopter	698	580	734	2675

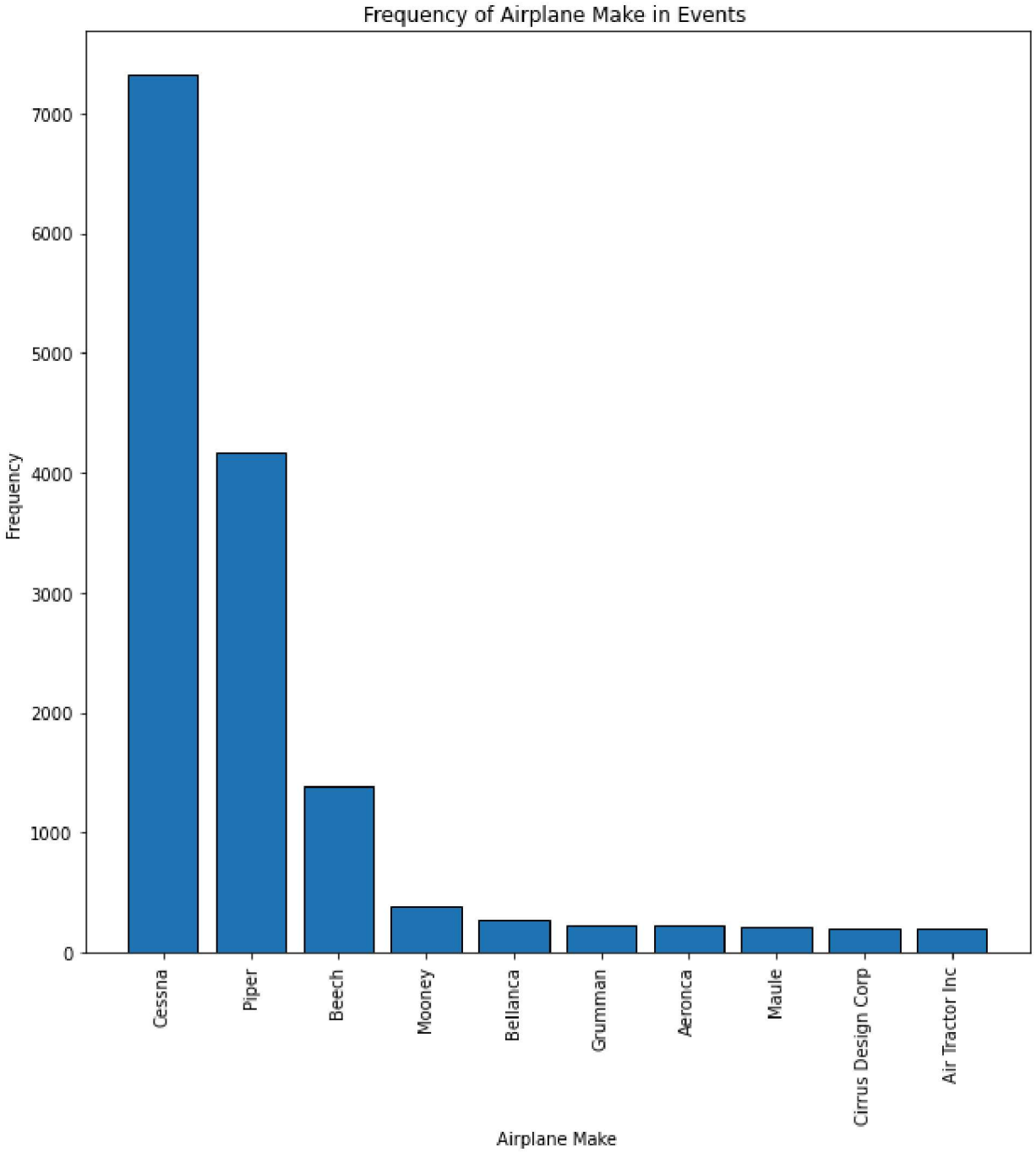
In [74]:
```
# Normalising the above table
pivot.div(pivot.sum(axis=1), axis=0)*100
```

Out[74]:

Aircraft.Category	Total.Fatal.Injuries	Total.Minor.Injuries	Total.Serious.Injuries	Total.Uninjured
Airplane	14.948794	8.876445	10.064513	66.110248
Glider	11.873840	17.068646	17.625232	53.432282
Gyrocraft	20.192308	13.942308	25.480769	40.384615
Helicopter	14.892255	12.374653	15.660337	57.072754

The analysis above clearly shows that in 100 airplane events, 62% of passengers remain uninjured, the highest percentage among all aircraft types. This indicates that airplanes have more safety measures in place to prevent injuries compared to other types of aircraft. Let us investigating airplane 'Makes' and find out which is the most preferred in the market.

In [75]:
```
# Filtering out airplanes from 'Aircraft.Category'
airplanes = df[df['Aircraft.Category'] == 'Airplane']

# Defining x and y
top10_airplane_make = airplanes['Make'].value_counts().head(10).index.tolist()
count_top10_airplane_make = airplanes['Make'].value_counts().head(10).values.tolist()

# Bar plot for the 'Aircraft Category' against counts
fig, ax = plt.subplots(figsize = (10,10))
ax.bar(top10_airplane_make, count_top10_airplane_make, edgecolor = 'black')
ax.set_title('Frequency of Airplane Make in Events ')
ax.set_xlabel('Airplane Make')
ax.set_ylabel('Frequency')
ax.tick_params(axis='x', labelrotation=90)
```


Frequency of Airplane Make in Events
Frequency
0
1000
2000
3000
4000
5000
6000
7000
Cessna
Piper
Beech
Mooney
Bellanca
Grumman
Aeronca
Maule
Cirrus Design Corp
Air Tractor Inc
Airplane Make

The 'Cessna' make seems to be the most common, as it appears most frequently in the plotted events. However, this doesn't necessarily suggest a problem with the make itself; it could simply reflect the widespread presence of Cessnas in the industry. How can we confirm if this is the case? Let us evaluate the extent of the damage for each make.

Replacing the values in Aircraft damage as follows: 'Destroyed'-'3','Substantial'-'2','Minor'-'1'

In [76]:
```
# Replacing categorical data in column 'Aircraft.damage' with continuous data for purposes of analysis
airplanes['Aircraft.damage'] = airplanes['Aircraft.damage'].replace({'Destroyed':'3','Substantial':'2','Minor':'1'})
airplanes.head()
```

Out[76]:

	Event.Id	Investigation.Type	Event.Date	Location	Country	Aircraft.damage	Aircraft.Category	Make	Model	Amateur.Built	Number.of.En
7	20020909X01562	Accident	1982-01-01	PULLMAN, WA	United States	2	Airplane	Cessna	140	No	
8	20020909X01561	Accident	1982-01-01	EAST HANOVER, NJ	United States	2	Airplane	Cessna	401B	No	
12	20020917X02148	Accident	1982-01-02	HOMER, LA	United States	3	Airplane	Bellanca	17-30A	No	
13	20020917X02134	Accident	1982-01-02	HEARNE, TX	United States	3	Airplane	Cessna	R172K	No	
14	20020917X02119	Accident	1982-01-02	CHICKASHA, OK	United States	3	Airplane	Navion	A	No	

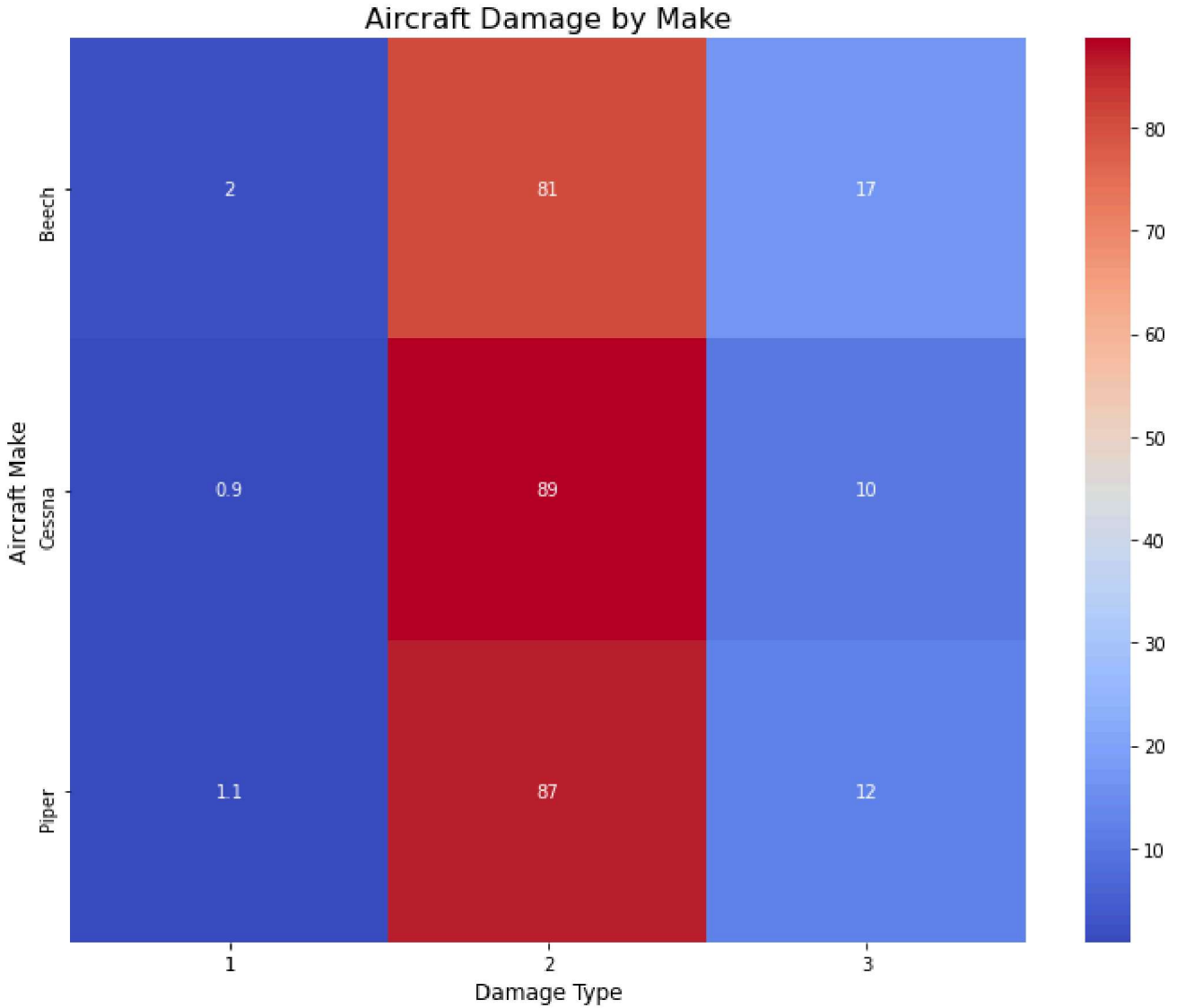
Is there any correlation between the Aircraft damage and Make of the airplane?

In [77]:
```
# Filtering top 3 makes
top_3_makes = airplanes['Make'].value_counts().head(3).index.tolist()

# Applying filters to get the column 'Make' and 'Aircraft.damage'
make_filtered = airplanes[
    (airplanes['Make'].isin(top_3_makes)) &
    (airplanes['Aircraft.damage'] != 'Unknown')
]

# Creating pivot table
pivot = make_filtered.pivot_table(index='Make', columns='Aircraft.damage', aggfunc='size', fill_value=0)
```

```
# Normalising the pivot table
pivot = pivot.div(pivot.sum(axis=1), axis=0)*100
```

In [78]:

```
# Visualizing the above data
fig, ax = plt.subplots(figsize=(10, 8))

# Plotting a heatmap
sns.heatmap(pivot, annot=True, cmap='coolwarm', cbar=True, ax=ax)

# Labelling
ax.set_title('Aircraft Damage by Make', fontsize=16)
ax.set_xlabel('Damage Type', fontsize=12)
ax.set_ylabel('Aircraft Make', fontsize=12)

# Showing the plot
plt.tight_layout()
plt.show()
```

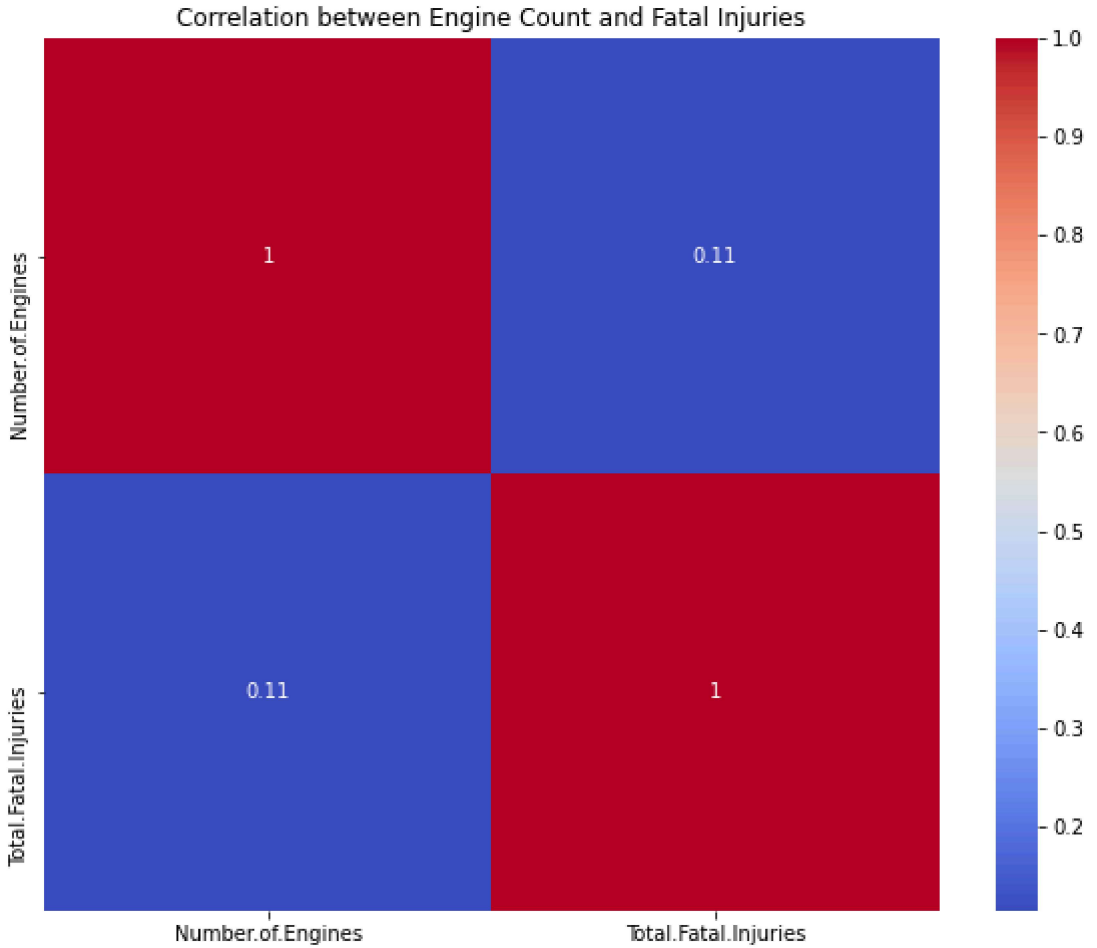


The map above shows that while Cessna has the highest percentage of substantial damages, it lowest percentage of severe damages. This suggests that, overall, Cessna performs relatively well compared to its competitors. Looking at the kind and number of engines and whether it has any relationship with fatal injuries

In [86]:
```
# Calculate correlation
correlation = airplanes[['Number.of.Engines', 'Total.Fatal.Injuries']].corr()
print(correlation)
```

```
# Heatmap of the correlation
fig, ax = plt.subplots(figsize=(10, 8))
sns.heatmap(correlation, annot=True, cmap='coolwarm')
ax.set_title('Correlation between Engine Count and Fatal Injuries')
plt.show()
```

```
                     Number.of.Engines  Total.Fatal.Injuries
Number.of.Engines             1.000000              0.114844
Total.Fatal.Injuries          0.114844              1.000000
```



We can see that there is very low correlation between number of engines and safety of the airplane. Let us investigate engine types and how they relate with injury counts.

In [80]:
```
# Filtering out unneccessary values
filtered_engine_types = df[~df['Engine.Type'].isin(['UNK','Unknown','None'])]

# Looking at the engine type against total fatal injuries
engine_fatal_summary = filtered_engine_types.groupby('Engine.Type')['Total.Fatal.Injuries'].agg(['sum', 'mean', 'count']).reset_in

# Rename columns for clarity
engine_fatal_summary.columns = ['Engine.Type', 'Total Fatal Injuries', 'Average Fatal Injuries', 'Count of Records']
engine_fatal_summary
```

Out[80]:

	Engine.Type	Total Fatal Injuries	Average Fatal Injuries	Count of Records
0	Electric	1.0	0.142857	7
1	LR	0.0	0.000000	1
2	Reciprocating	22230.0	0.329026	67563
3	Turbo Fan	3081.0	3.176289	970
4	Turbo Jet	630.0	1.425339	442
5	Turbo Prop	1950.0	0.726257	2685
6	Turbo Shaft	1217.0	0.408115	2982

The reciprocating engine is the most widely used in the aviation market. It also has the second-lowest average number of fatal injuries, following the electric engine. This indicates that it is less likely prone to engine failures. The relatively low number of electric engines could indicate potential drawbacks, such as higher prices or maintenance costs. Given these factors, the reciprocating engine emerges as the optimal choice. Now let us investigate the impact of weather conditions to events.

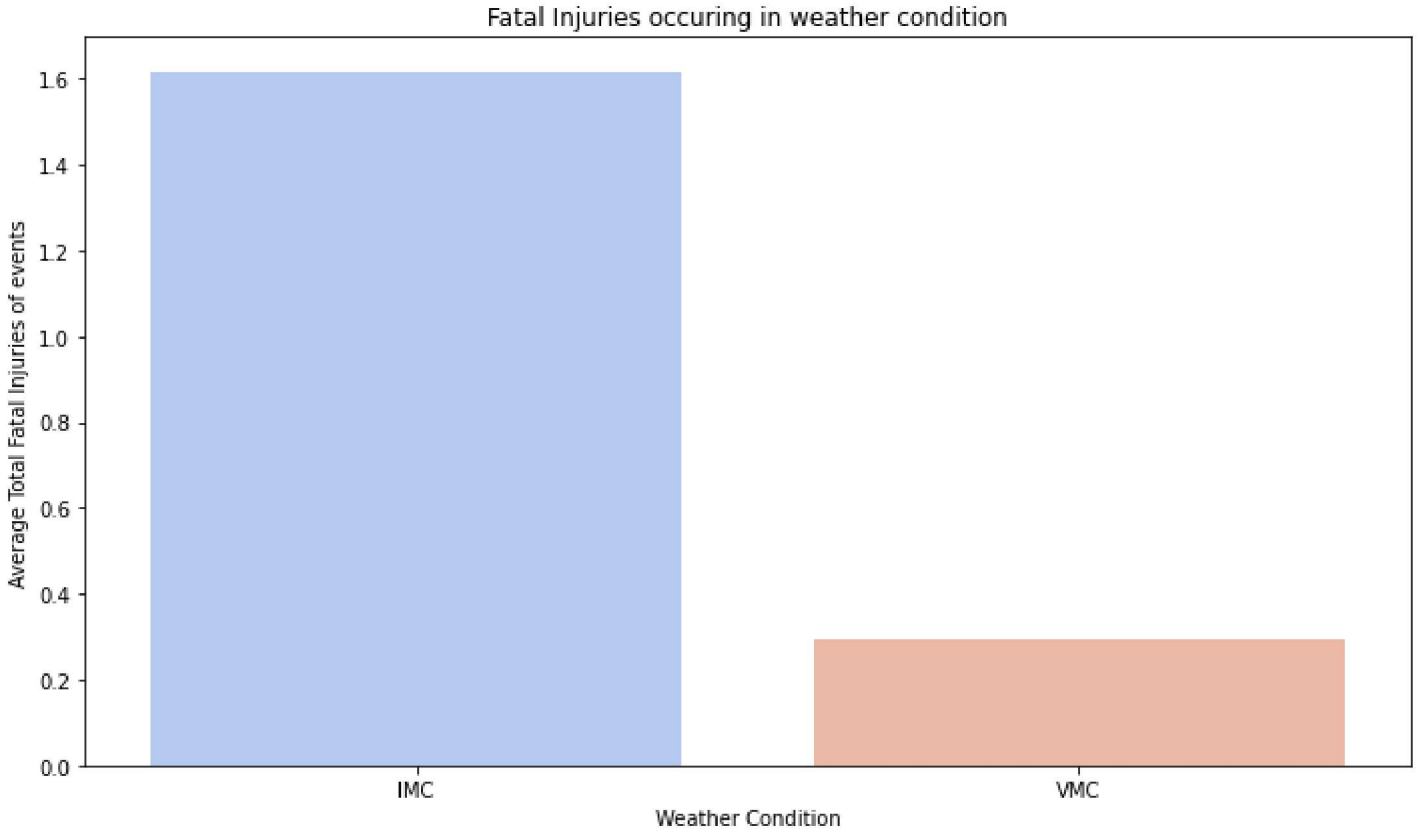
In [81]:
```
# Standardizing the values in the column
df['Weather.Condition'] = df['Weather.Condition'].str.upper()
```

In [82]:
```
# Filtering out unneccessary values
filtered_weather_condition = df[~df['Weather.Condition'].isin(['UNK'])]

# Looking at the engine type against total fatal injuries
weather_fatal_summary = filtered_weather_condition.groupby('Weather.Condition')['Total.Fatal.Injuries'].agg('mean')
weather_fatal_summary
```

Out[82]:
```
Weather.Condition
IMC    1.617315
```

```
VMC    0.291487
Name: Total.Fatal.Injuries, dtype: float64
```

In [83]:
```
# Visualizing the above data
fig, ax = plt.subplots(figsize=(10, 6))
sns.barplot(x=weather_fatal_summary.index, y=weather_fatal_summary.values, palette='coolwarm')

# Add title and labels
ax.set_title('Fatal Injuries occuring in weather condition')
ax.set_xlabel('Weather Condition')
ax.set_ylabel('Average Total Fatal Injuries of events')

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



It is clear that majority of the events occur in the IMC (Instrument Meteorological Conditions) weather conditions that are not suitable for visual flight and require the use of instruments for navigation and control. The client could think about scheduling flights in the VMC (Visual Meteorological Conditions), where weather conditions allow pilots to navigate using visual references or mobilise resources to put safety measures in place during flights occuring in the IMC weather conditions. Let us finally look at phase of flight compared to fatal injuries.

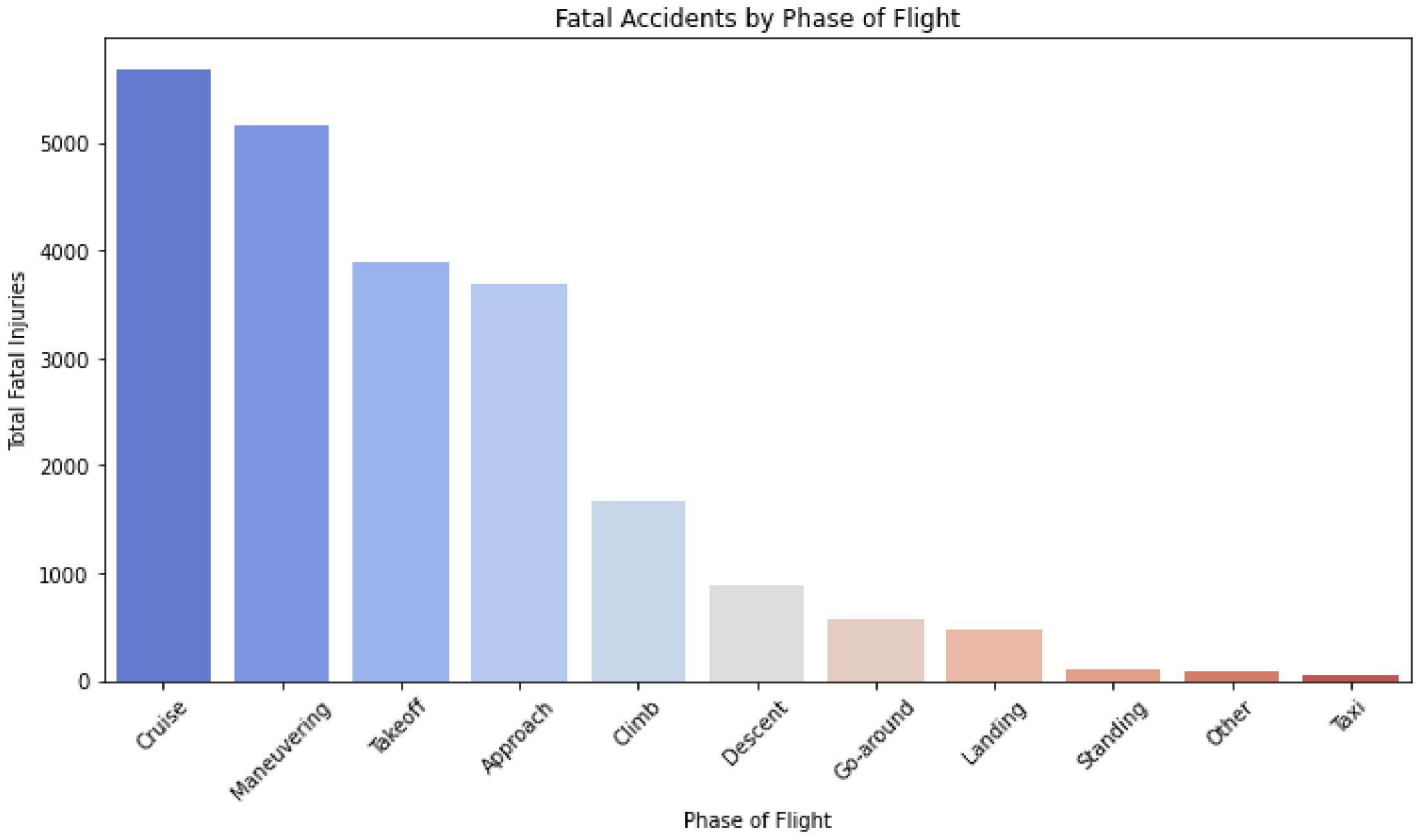
In [84]:
```
# Grouping phase of flight and count of fatal injuries
fatal_phase = df.groupby('Broad.phase.of.flight')['Total.Fatal.Injuries'].agg(['sum', 'count']).reset_index()
fatal_phase = fatal_phase[fatal_phase['Broad.phase.of.flight'] != 'Unknown']
fatal_phase
```

Out[84]:

	Broad.phase.of.flight	sum	count
0	Approach	3686.0	6119
1	Climb	1679.0	1834
2	Cruise	5691.0	9656
3	Descent	892.0	1659
4	Go-around	575.0	1334
5	Landing	478.0	14783
6	Maneuvering	5162.0	7925
7	Other	84.0	108
8	Standing	98.0	670
9	Takeoff	3893.0	12001
10	Taxi	42.0	1732

In [85]:
```
# Sorting by fatal injuries in descending order
fatal_phase = fatal_phase.sort_values(by='sum', ascending=False)

# Visualising the data
fig, ax = plt.subplots(figsize=(10, 6))
sns.barplot(data=fatal_phase, x='Broad.phase.of.flight', y='sum', palette='coolwarm', ax=ax)

# Labelling
ax.set_title('Fatal Accidents by Phase of Flight')
ax.set_xlabel('Phase of Flight')
```

```
ax.set_ylabel('Total Fatal Injuries')

# Rotate x-ticks
ax.set_xticklabels(ax.get_xticklabels(), rotation=45)

# Showing the plot
plt.tight_layout()
plt.show()
```



The 'Cruise,' 'Maneuvering,' 'Takeoff,' and 'Approach' phases are clearly the most critical. The client should explore strategies to ensure these phases proceed without incidents and invest in robust precautionary measures to enhance safety.

CONCLUSIONS

The following are the actionable insights from the analysis:

1. Opt for the Cessna airplane model, as it has proven to be the safest with the lowest extent of damage, despite its high prevalence. This indicates that it is resilient and can withstand risks, even though it is the most exposed to them.
2. Choose the reciprocating engine, which has the lowest number of fatal injuries, indicating a lower likelihood of engine failures.
3. Consider scheduling flights during VMC weather conditions and implement enhanced safety measures for flights in IMC conditions. This will help manage risk and improve safety during adverse weather scenarios.
4. Focus on enforcing additional safety measures during critical flight phases such as 'Cruise,' 'Maneuvering,' 'Takeoff,' and 'Approach.' The client should explore ways to ensure smooth operations during these phases and invest in robust precautionary measures to minimize mishaps.