Final Project Submission (Phase 1)

Please fill out:

- Student name: JUDITH OTIENO
- Student pace: FULL TIME
- Instructor name: WILLIAM OKOBA
- Tableau post URL:A Data Driven Approach to Aircraft Risk Assesment

Navigating the Skies of Opportunity – A Data-Driven Approach to Aircraft Risk Assessment

Introduction

Our company is embarking on an exciting new venture, expanding its portfolio into the dynamic world of aviation which encompasses both commercial and private air enterprises. This strategic diversification aims to strengthen our market position and open new avenues for growth. However, entering an industry as complex and regulated as aviation presents unique challenges, particularly concerning inherent operational risks associated with an aircraft. To ensure a successful and secure entry into this sector, a critical first step is to thoroughly understand and mitigate potential risks. This project is specifically designed to address this imperative need.

A dataset containing civil aviation accident and incident data from 1948 to 2023, sourced from the National Transportation Safety Board via Kaggle.

Aim

Identify and assess the risk profiles of Airplanes and Helicopters in order to pinpoint an aircraft that represent the lowest o risk interms of fatalities, thereby providing a robust foundation for the head of the aviation division.

Objectives

1. Examine Purpose of Flight and Accident Frequency.
2. Determine Accident Phase Dominance.
3. Compare Aircraft structural damage and Type to Fatalities.
4. Identify Aircraft Model and Make with Lowest Fatalities.

The Tableau dashboard for this Analysis can be found here. A short presentation of the same can be found here

1.0 Overview

This study follows the outline below:

1. Data loading and Inspection

2. Data Cleaning and Processing
3. Univariate Analysis
4. Bivariate Analysis
5. Multivariate Analysis
6. Findings and Conclusion
7. Recommendations

1.1 Import Libraries

In [1]:

```
#Importing the Libraries needed
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

1.2 Data Loading and Inspection

In [2]:

```
#Loading the CSV
df = pd.read_csv('data\Aviation_Data.csv')
df
```

```
C:\Users\Achie\anaconda3\envs\learn-env\lib\site-packages\IPython\core\interactiveshe
ll.py:3145: DtypeWarning: Columns (6,7,28) have mixed types.Specify dtype option on i
mport or set low_memory=False.
  has_raised = await self.run_ast_nodes(code_ast.body, cell_name,
```

Out[2]:

	Event.Id	Investigation.Type	Accident.Number	Event.Date	Location	Country	Lat
0	20001218X45444	Accident	SEA87LA080	1948-10-24	MOOSE CREEK, ID	United States	
1	20001218X45447	Accident	LAX94LA336	1962-07-19	BRIDGEPORT, CA	United States	
2	20061025X01555	Accident	NYC07LA005	1974-08-30	Saltville, VA	United States	36
3	20001218X45448	Accident	LAX96LA321	1977-06-19	EUREKA, CA	United States	
4	20041105X01764	Accident	CHI79FA064	1979-08-02	Canton, OH	United States	
...	...	...	...	...	...	...	
90343	20221227106491	Accident	ERA23LA093	2022-12-26	Annapolis, MD	United States	
90344	20221227106494	Accident	ERA23LA095	2022-12-26	Hampton, NH	United States	
90345	20221227106497	Accident	WPR23LA075	2022-12-26	Payson, AZ	United States	341
90346	20221227106498	Accident	WPR23LA076	2022-12-26	Morgan, UT	United States	
90347	20221230106513	Accident	ERA23LA097	2022-12-29	Athens, GA	United States	

90348 rows × 31 columns

In [3]:
```
# To check that the csv loaded successfullysplay the first few rows of the DataFrame
print("DataFrame loaded successfully. These are the first 5 rows:")
df.head()
```

```
DataFrame loaded successfully. These are the first 5 rows:
```

Out[3]:

	Event.Id	Investigation.Type	Accident.Number	Event.Date	Location	Country	Latitude
0	20001218X45444	Accident	SEA87LA080	1948-10-24	MOOSE CREEK, ID	United States	NaN
1	20001218X45447	Accident	LAX94LA336	1962-07-19	BRIDGEPORT, CA	United States	NaN
2	20061025X01555	Accident	NYC07LA005	1974-08-30	Saltville, VA	United States	36.9222
3	20001218X45448	Accident	LAX96LA321	1977-06-19	EUREKA, CA	United States	NaN
4	20041105X01764	Accident	CHI79FA064	1979-08-02	Canton, OH	United States	NaN

5 rows × 31 columns

In [4]:
```
print("DataFrame loaded successfully. These are the last 5 rows:")
df.tail()
```

```
DataFrame loaded successfully. These are the last 5 rows:
```

Out[4]:

	Event.Id	Investigation.Type	Accident.Number	Event.Date	Location	Country	Latitu
90343	20221227106491	Accident	ERA23LA093	2022-12-26	Annapolis, MD	United States	N
90344	20221227106494	Accident	ERA23LA095	2022-12-26	Hampton, NH	United States	N
90345	20221227106497	Accident	WPR23LA075	2022-12-26	Payson, AZ	United States	34152
90346	20221227106498	Accident	WPR23LA076	2022-12-26	Morgan, UT	United States	N
90347	20221230106513	Accident	ERA23LA097	2022-12-29	Athens, GA	United States	N

5 rows × 31 columns

In [5]:
```
#Checking info about the dataframe icluding the datatypes and non-values
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 90348 entries, 0 to 90347
Data columns (total 31 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Event.Id                88889 non-null  object
 1   Investigation.Type      90348 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        73659 non-null  object
dtypes: float64(5), object(26)
memory usage: 21.4+ MB
```

Obeservation: The dataframe contains floats and objects. There are 31 columns and 90348 rows including the column names.

In [6]:
```python
#Change Column names
df.columns = df.columns.str.replace('.', '_', regex=False)
df.columns
```

Out[6]:
```
Index(['Event_Id', 'Investigation_Type', 'Accident_Number', 'Event_Date',
       'Location', 'Country', 'Latitude', 'Longitude', 'Airport_Code',
       'Airport_Name', 'Injury_Severity', 'Aircraft_damage',
       'Aircraft_Category', 'Registration_Number', 'Make', 'Model',
       'Amateur_Built', 'Number_of_Engines', 'Engine_Type', 'FAR_Description',
       'Schedule', 'Purpose_of_flight', 'Air_carrier', 'Total_Fatal_Injuries',
       'Total_Serious_Injuries', 'Total_Minor_Injuries', 'Total_Uninjured',
       'Weather_Condition', 'Broad_phase_of_flight', 'Report_Status',
       'Publication_Date'],
      dtype='object')
```

In [7]:
```python
print("DataFrame dimension is a tuple of:")
df.shape
```

```
DataFrame dimension is a tuple of:
```

Out[7]:
```
(90348, 31)
```

In [8]:
```python
print(f"The datatypes in the Dataframe are:{df.dtypes}")
```

```
The datatypes in the Dataframe are: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
```

In [9]:

```
#Checking for null values before droping rows
df.isnull().sum()
```

Out[9]:

```
Event_Id                   1459
Investigation_Type            0
Accident_Number            1459
Event_Date                 1459
Location                   1511
Country                    1685
Latitude                  55966
Longitude                 55975
Airport_Code              40099
Airport_Name              37558
Injury_Severity            2459
Aircraft_damage            4653
Aircraft_Category         58061
Registration_Number        2776
Make                       1522
Model                      1551
Amateur_Built              1561
Number_of_Engines          7543
Engine_Type                8536
FAR_Description           58325
Schedule                  77766
Purpose_of_flight          7651
Air_carrier               73700
Total_Fatal_Injuries      12860
Total_Serious_Injuries    13969
Total_Minor_Injuries      13392
Total_Uninjured            7371
Weather_Condition          5951
Broad_phase_of_flight     28624
Report_Status              7840
Publication_Date          16689
dtype: int64
```

In [10]:

```
#Statistical Summary of Numerical Columns
df.describe().T
```

Out[10]:

	count	mean	std	min	25%	50%	75%	max
Number_of_Engines	82805.0	1.146585	0.446510	0.0	1.0	1.0	1.0	8.0
Total_Fatal_Injuries	77488.0	0.647855	5.485960	0.0	0.0	0.0	0.0	349.0
Total_Serious_Injuries	76379.0	0.279881	1.544084	0.0	0.0	0.0	0.0	161.0

	count	mean	std	min	25%	50%	75%	max
Total_Minor_Injuries	76956.0	0.357061	2.235625	0.0	0.0	0.0	0.0	380.0
Total_Uninjured	82977.0	5.325440	27.913634	0.0	0.0	1.0	2.0	699.0

1.3 Data Cleaning and Processing

1.3.1 Duplicates

In [11]:
```
#Check for duplicated rows using the event id.
duplicates = df[df.duplicated(keep=False)].sort_values(by='Event_Id')
duplicates
```

Out[11]:

	Event_Id	Investigation_Type	Accident_Number	Event_Date	Location	Country	Latitude	Lon
64030	NaN	25-09-2020	NaN	NaN	NaN	NaN	NaN	
64050	NaN	25-09-2020	NaN	NaN	NaN	NaN	NaN	
64052	NaN	25-09-2020	NaN	NaN	NaN	NaN	NaN	
64388	NaN	25-09-2020	NaN	NaN	NaN	NaN	NaN	
64541	NaN	25-09-2020	NaN	NaN	NaN	NaN	NaN	
...	...	...	...	...	...	...	...	
90004	NaN	15-12-2022	NaN	NaN	NaN	NaN	NaN	
90010	NaN	15-12-2022	NaN	NaN	NaN	NaN	NaN	
90031	NaN	15-12-2022	NaN	NaN	NaN	NaN	NaN	
90090	NaN	20-12-2022	NaN	NaN	NaN	NaN	NaN	
90097	NaN	20-12-2022	NaN	NaN	NaN	NaN	NaN	

1447 rows × 31 columns

In [12]:
```
#Check for duplicates in the dataframe
df.duplicated().value_counts()
```

Out[12]:
```
False    88958
True      1390
dtype: int64
```

In [13]:
```
#droping dupicated Entries
df = df.drop_duplicates()
df
```

Out[13]:

	Event_Id	Investigation_Type	Accident_Number	Event_Date	Location	Country	La
0	20001218X45444	Accident	SEA87LA080	1948-10-24	MOOSE CREEK, ID	United States	
1	20001218X45447	Accident	LAX94LA336	1962-07-19	BRIDGEPORT, CA	United States	
2	20061025X01555	Accident	NYC07LA005	1974-08-30	Saltville, VA	United States	3
3	20001218X45448	Accident	LAX96LA321	1977-06-19	EUREKA, CA	United States	

	Event_Id	Investigation_Type	Accident_Number	Event_Date	Location	Country	La
4	20041105X01764	Accident	CHI79FA064	1979-08-02	Canton, OH	United States	
...	...	...	...	...	...	...	
90343	20221227106491	Accident	ERA23LA093	2022-12-26	Annapolis, MD	United States	
90344	20221227106494	Accident	ERA23LA095	2022-12-26	Hampton, NH	United States	
90345	20221227106497	Accident	WPR23LA075	2022-12-26	Payson, AZ	United States	34
90346	20221227106498	Accident	WPR23LA076	2022-12-26	Morgan, UT	United States	
90347	20221230106513	Accident	ERA23LA097	2022-12-29	Athens, GA	United States	

88958 rows × 31 columns

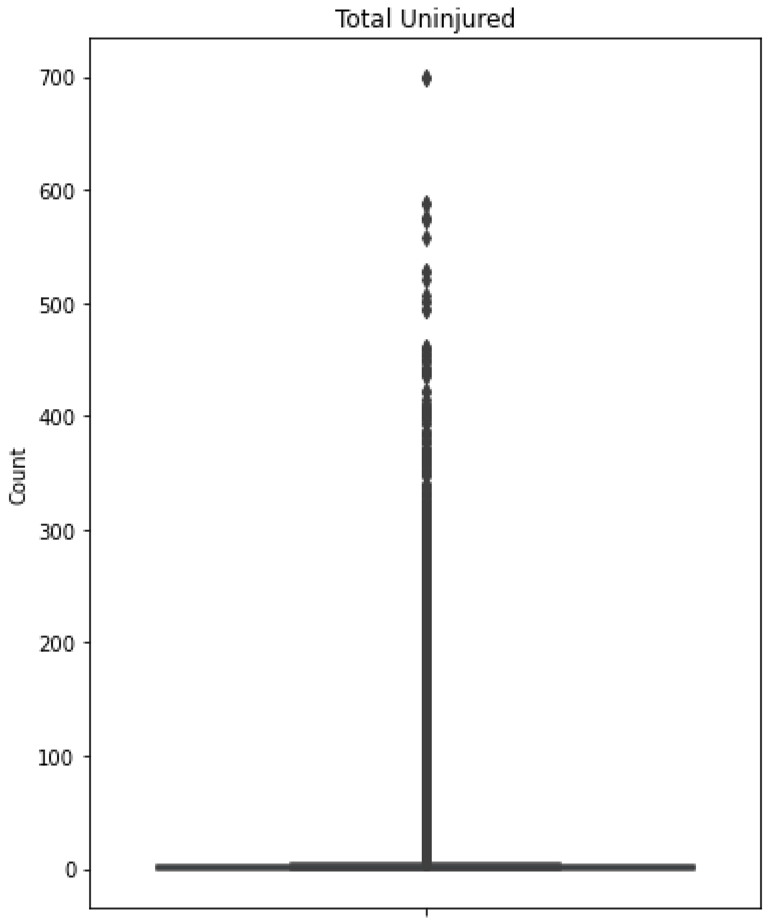
In [14]:
```
#Check if duplicated rows have been dropped from the dataframe.
df.duplicated().value_counts()
```

Out[14]:
```
False    88958
dtype: int64
```

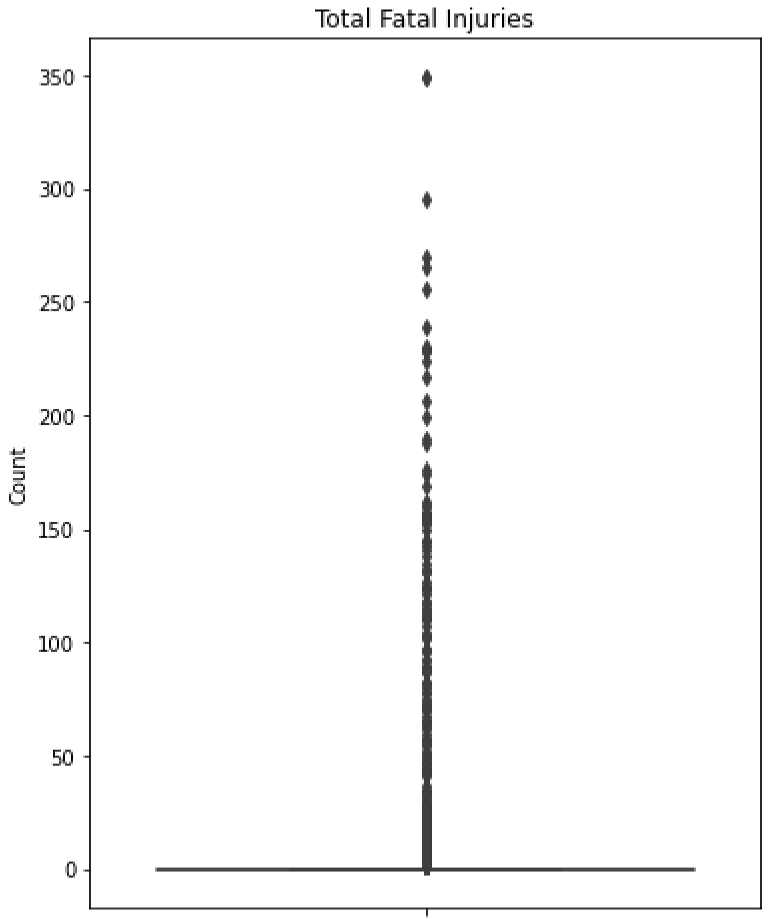
All rows with identical information have been dropped.

1.3.2 Outliers

In [15]:
```
#Checking for outliers in Total Uninjured
plt.figure(figsize=(6, 8))
sns.boxplot(y = df['Total_Uninjured'])
plt.title('Total Uninjured')
plt.ylabel('Count')
plt.show()
```

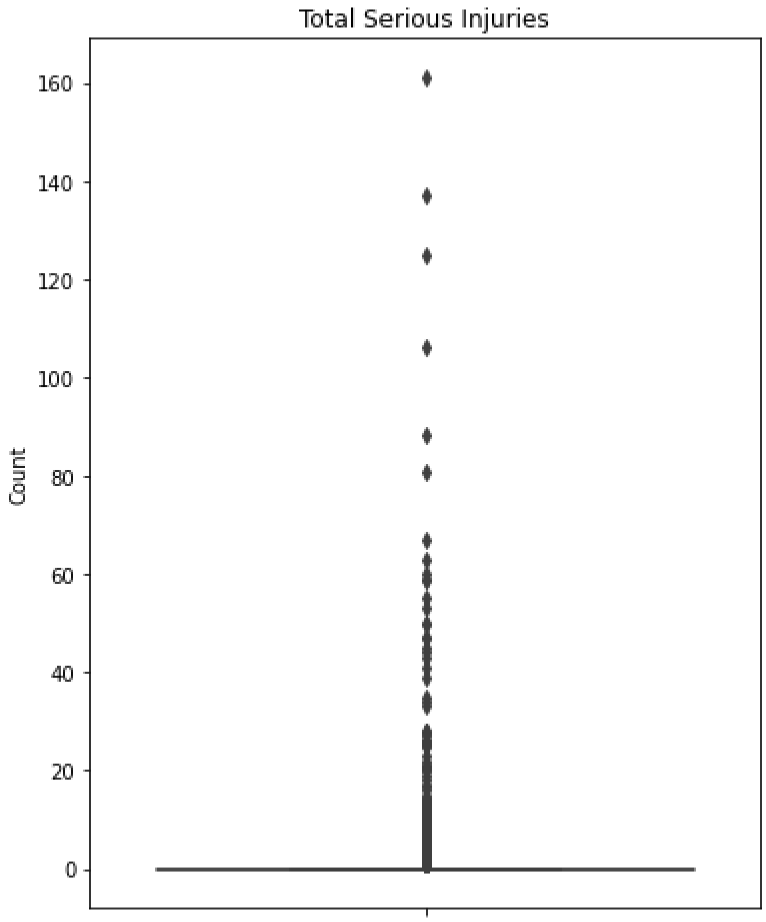


In [16]:
```
#Total Fatal Injuries
plt.figure(figsize=(6, 8))
sns.boxplot(y= df['Total_Fatal_Injuries'])
plt.title('Total Fatal Injuries')
plt.ylabel('Count')
plt.show()
```



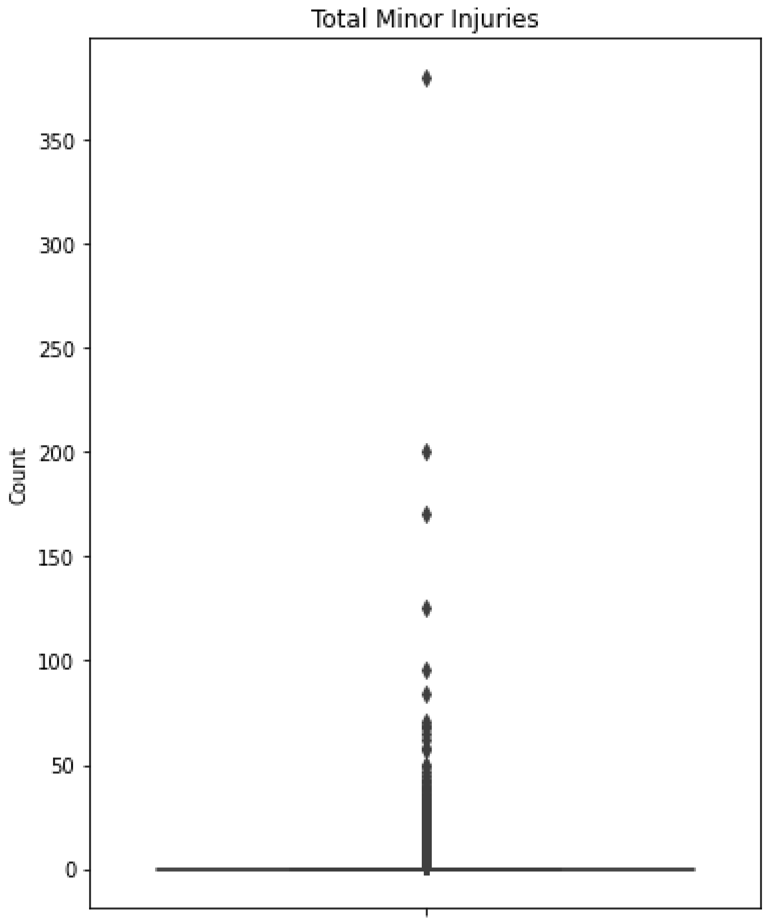
In [17]:

```
#Total serious but non fatal injuries
plt.figure(figsize=(6, 8))
sns.boxplot(y= df['Total_Serious_Injuries'])
plt.title('Total Serious Injuries')
plt.ylabel('Count')
plt.show()
```



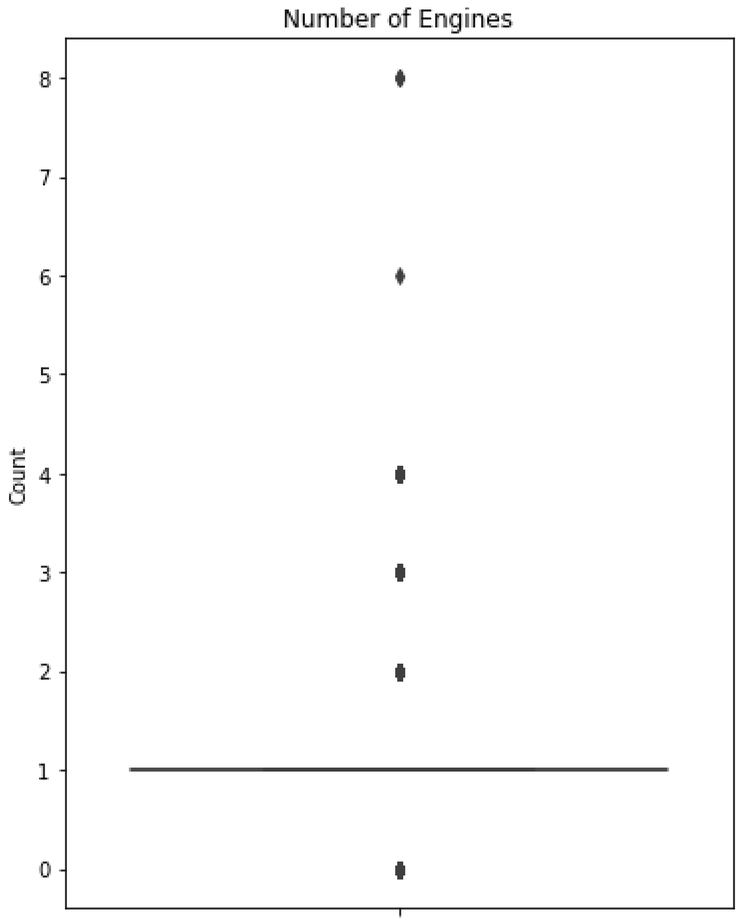
In [18]:

```
#Total Minor Injuries
plt.figure(figsize=(6, 8))
sns.boxplot(y= df['Total_Minor_Injuries'])
plt.title('Total Minor Injuries')
plt.ylabel('Count')
plt.show()
print(f'The mean number of total minor injuries is: {df["Total_Minor_Injuries"].mean
```




```
The mean number of total minor injuries is: 0.3570611778158948
```

In [19]:
```
plt.figure(figsize=(6, 8))
sns.boxplot(y= df['Number_of_Engines'])
plt.title('Number of Engines')
plt.ylabel('Count')
plt.show()
print(f'The mean number of Engines is: {df["Number_of_Engines"].mean()}')
df['Number_of_Engines'].unique()
```




```
The mean number of Engines is: 1.1465853511261397
```

Out[19]:
```
array([ 1., nan,  2.,  0.,  3.,  4.,  8.,  6.])
```

NB

For the purpose of this investigation, anomalies within the passenger classification data (fatal, serious, minor, and uninjured categories) will be preserved. The rationale for this decision is rooted in the potential for these outliers to reveal significant determinants of aircraft safety.

In [20]:
```
#Convert every value in (df) into a string datatype and store it in object_df
object_df = df.applymap(lambda x: str(x))
object_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 88958 entries, 0 to 90347
Data columns (total 31 columns):
 #   Column                  Non-Null Count  Dtype 
---  ------                  --------------  ----- 
 0   Event_Id                88958 non-null  object
 1   Investigation_Type      88958 non-null  object
 2   Accident_Number         88958 non-null  object
 3   Event_Date              88958 non-null  object
 4   Location                88958 non-null  object
 5   Country                 88958 non-null  object
 6   Latitude                88958 non-null  object
 7   Longitude               88958 non-null  object
 8   Airport_Code            88958 non-null  object
 9   Airport_Name            88958 non-null  object
 10  Injury_Severity         88958 non-null  object
 11  Aircraft_damage         88958 non-null  object
 12  Aircraft_Category       88958 non-null  object
 13  Registration_Number     88958 non-null  object
 14  Make                    88958 non-null  object
 15  Model                   88958 non-null  object
 16  Amateur_Built           88958 non-null  object
 17  Number_of_Engines       88958 non-null  object
```

```
 18  Engine_Type            88958 non-null  object
 19  FAR_Description        88958 non-null  object
 20  Schedule               88958 non-null  object
 21  Purpose_of_flight      88958 non-null  object
 22  Air_carrier            88958 non-null  object
 23  Total_Fatal_Injuries   88958 non-null  object
 24  Total_Serious_Injuries 88958 non-null  object
 25  Total_Minor_Injuries   88958 non-null  object
 26  Total_Uninjured        88958 non-null  object
 27  Weather_Condition      88958 non-null  object
 28  Broad_phase_of_flight  88958 non-null  object
 29  Report_Status          88958 non-null  object
 30  Publication_Date       88958 non-null  object
dtypes: object(31)
memory usage: 21.7+ MB
```

1.3.3 Null Values

In [21]:
```
#Null values after dropping duplicates
null_percentage = (df.isnull().sum() / len(df)) * 100
null_percentage
```

Out[21]:
```
Event_Id                   0.077565
Investigation_Type         0.000000
Accident_Number            0.077565
Event_Date                 0.077565
Location                   0.136019
Country                    0.331617
Latitude                  61.350300
Longitude                 61.360417
Airport_Code              43.513793
Airport_Name              40.657389
Injury_Severity            1.201691
Aircraft_damage            3.668023
Aircraft_Category         63.705344
Registration_Number        1.558039
Make                       0.148385
Model                      0.180984
Amateur_Built              0.192226
Number_of_Engines          6.916747
Engine_Type                8.033004
FAR_Description           64.002113
Schedule                  85.856247
Purpose_of_flight          7.038153
Air_carrier               81.285550
Total_Fatal_Injuries      12.893725
Total_Serious_Injuries    14.140381
Total_Minor_Injuries      13.491760
Total_Uninjured            6.723398
Weather_Condition          5.127139
Broad_phase_of_flight     30.614447
Report_Status              7.250613
Publication_Date          17.198004
dtype: float64
```

Observation:

Percentage Missing Data for columns with over 40% missing data

1. 61.4% of Longitudes and Latitudes
2. 43.5% of Airport Code
3. 40.7% of Airport Name
4. 63.7% of Aircraft Category
5. 64.0% of FAR Description
6. 85.9% of Schedule

7. 81.3% of Air carrier

Function to get categorical modes

In [22]:
```
def get_categorical_modes(df):
    categorical_modes = {}

    for col in df.columns:

        # Check if the column's dtype is 'object'
        if df[col].dtype == 'object':
            mode_value = df[col].mode()
            categorical_modes[col] = mode_value

    return categorical_modes
```

In [23]:
```
get_categorical_modes(df)
```

Out[23]:
```
{'Event_Id': 0    20001212X19172
 1    20001214X45071
 dtype: object,
 'Investigation_Type': 0    Accident
 dtype: object,
 'Accident_Number': 0     CEN22FA424
 1     CEN22LA149
 2     CEN22LA346
 3     CEN23MA034
 4     DCA22LA135
 5     DCA22LA201
 6     DCA22WA089
 7     DCA22WA130
 8     DCA22WA158
 9     DCA22WA167
 10    DCA22WA172
 11    DCA22WA204
 12    DCA22WA214
 13    DCA23WA071
 14    ERA22FA318
 15    ERA22FA338
 16    ERA22LA103
 17    ERA22LA119
 18    ERA22LA364
 19    ERA22LA379
 20    GAA22WA241
 21    WPR22FA309
 22    WPR22LA143
 23    WPR22LA201
 24    WPR23LA041
 25    WPR23LA045
 dtype: object,
 'Event_Date': 0    1982-05-16
 1    1984-06-30
 2    2000-07-08
 dtype: object,
 'Location': 0    ANCHORAGE, AK
 dtype: object,
 'Country': 0    United States
 dtype: object,
 'Latitude': 0    332739N
 dtype: object,
 'Longitude': 0    0112457W
 dtype: object,
 'Airport_Code': 0    NONE
 dtype: object,
 'Airport_Name': 0    Private
 dtype: object,
 'Injury_Severity': 0    Non-Fatal
```

```
dtype: object,
'Aircraft_damage': 0    Substantial
dtype: object,
'Aircraft_Category': 0    Airplane
dtype: object,
'Registration_Number': 0    NONE
dtype: object,
'Make': 0    Cessna
dtype: object,
'Model': 0    152
dtype: object,
'Amateur_Built': 0    No
dtype: object,
'Engine_Type': 0    Reciprocating
dtype: object,
'FAR_Description': 0    091
dtype: object,
'Schedule': 0    NSCH
dtype: object,
'Purpose_of_flight': 0    Personal
dtype: object,
'Air_carrier': 0    Pilot
dtype: object,
'Weather_Condition': 0    VMC
dtype: object,
'Broad_phase_of_flight': 0    Landing
dtype: object,
'Report_Status': 0    Probable Cause
dtype: object,
'Publication_Date': 0    25-09-2020
dtype: object}
```

In [24]:

```
#count the number of non-missing (non-NaN) values in each column ie. filled in value
df.notna().sum()
```

Out[24]:

```
Event_Id                  88889
Investigation_Type        88958
Accident_Number           88889
Event_Date                88889
Location                  88837
Country                   88663
Latitude                  34382
Longitude                 34373
Airport_Code              50249
Airport_Name              52790
Injury_Severity           87889
Aircraft_damage           85695
Aircraft_Category         32287
Registration_Number       87572
Make                      88826
Model                     88797
Amateur_Built             88787
Number_of_Engines         82805
Engine_Type               81812
FAR_Description           32023
Schedule                  12582
Purpose_of_flight         82697
Air_carrier               16648
Total_Fatal_Injuries      77488
Total_Serious_Injuries    76379
Total_Minor_Injuries      76956
Total_Uninjured           82977
Weather_Condition         84397
Broad_phase_of_flight     61724
Report_Status             82508
Publication_Date          73659
dtype: int64
```

In [25]:
```
df['Broad_phase_of_flight'].value_counts()
```

Out[25]:
```
Landing        15428
Takeoff        12493
Cruise         10269
Maneuvering     8144
Approach        6546
Climb           2034
Taxi            1958
Descent         1887
Go-around       1353
Standing         945
Unknown          548
Other            119
Name: Broad_phase_of_flight, dtype: int64
```

Observation:

- Most accidents occur in the Landing phase of flight followed by Take-off then lastly during cruise.

In [26]:
```
df['Purpose_of_flight'].value_counts()
```

Out[26]:
```
Personal                      49448
Instructional                 10601
Unknown                        6802
Aerial Application             4712
Business                       4018
Positioning                    1646
Other Work Use                 1264
Ferry                           812
Aerial Observation              794
Public Aircraft                 720
Executive/corporate             553
Flight Test                     405
Skydiving                       182
External Load                   123
Public Aircraft - Federal       105
Banner Tow                      101
Air Race show                    99
Public Aircraft - Local          74
Public Aircraft - State          64
Air Race/show                    59
Glider Tow                       53
Firefighting                     40
Air Drop                         11
ASHO                              6
PUBS                              4
PUBL                              1
Name: Purpose_of_flight, dtype: int64
```

Observation

- Aircrafts that seem to involved in more accidents seem to be used mostly for Personal and Instructional use.

In [27]:
```
df['Injury_Severity'].value_counts()
```

Out[27]:
```
Non-Fatal     67357
Fatal(1)       6167
Fatal          5262
Fatal(2)       3711
Incident       2219
              ...
Fatal(115)        1
Fatal(270)        1
```

```
Fatal(189)        1
Fatal(45)         1
Fatal(43)         1
Name: Injury_Severity, Length: 109, dtype: int64
```

Observation:

- Most accidents in the dataset were of non-fatal injury severity.

Filling in NaN Values in Columns of Interest.

In [28]:
```
#df2 = drop Aircraft_Category,Latitude,Longitude,Airport_Code,Airport_Name,Report_St
df
```

Out[28]:

	Event_Id	Investigation_Type	Accident_Number	Event_Date	Location	Country	La
0	20001218X45444	Accident	SEA87LA080	1948-10-24	MOOSE CREEK, ID	United States	
1	20001218X45447	Accident	LAX94LA336	1962-07-19	BRIDGEPORT, CA	United States	
2	20061025X01555	Accident	NYC07LA005	1974-08-30	Saltville, VA	United States	3
3	20001218X45448	Accident	LAX96LA321	1977-06-19	EUREKA, CA	United States	
4	20041105X01764	Accident	CHI79FA064	1979-08-02	Canton, OH	United States	
...	...	...	...	...	...	...	
90343	20221227106491	Accident	ERA23LA093	2022-12-26	Annapolis, MD	United States	
90344	20221227106494	Accident	ERA23LA095	2022-12-26	Hampton, NH	United States	
90345	20221227106497	Accident	WPR23LA075	2022-12-26	Payson, AZ	United States	34
90346	20221227106498	Accident	WPR23LA076	2022-12-26	Morgan, UT	United States	
90347	20221230106513	Accident	ERA23LA097	2022-12-29	Athens, GA	United States	

88958 rows × 31 columns

In [29]:
```
#Forming a new dataframe with the columns to be used in this analysis df1

df1 = df[['Investigation_Type','Make','Purpose_of_flight', 'Model', 'Total_Fatal_Inj
df1
```

Out[29]:

	Investigation_Type	Make	Purpose_of_flight	Model	Total_Fatal_Injuries	Total_Serious_Inj
0	Accident	Stinson	Personal	108-3	2.0	
1	Accident	Piper	Personal	PA24-180	4.0	
2	Accident	Cessna	Personal	172M	3.0	

	Investigation_Type	Make	Purpose_of_flight	Model	Total_Fatal_Injuries	Total_Serious_Inj
3	Accident	Rockwell	Personal	112	2.0	
4	Accident	Cessna	Personal	501	1.0	
...	...	...	...	...	...	
90343	Accident	PIPER	Personal	PA-28-151	0.0	
90344	Accident	BELLANCA	NaN	7ECA	0.0	
90345	Accident	AMERICAN CHAMPION AIRCRAFT	Personal	8GCBC	0.0	
90346	Accident	CESSNA	Personal	210N	0.0	
90347	Accident	PIPER	Personal	PA-24-260	0.0	

88958 rows × 16 columns

1.3.4 Filtering

For this study focus was solely on powered, traditional aircraft types i.e. those with engines. The other categories that might not be relevant to the research questions.

To analyze only incidents involving "Airplanes" and "Helicopters," filtering was used in order to isolate those records and prevent irrelevant data from skewing the results..

In [30]:
```
df2 = df1.loc[(df1['Aircraft_Category'] == 'Airplane') | (df1['Aircraft_Category'] =
df2
```

Out[30]:

	Investigation_Type	Make	Purpose_of_flight	Model	Total_Fatal_Injuries	Total_Seriou
5	Accident	Mcdonnell Douglas	NaN	DC9	NaN	
7	Accident	Cessna	Personal	140	0.0	
8	Accident	Cessna	Business	401B	0.0	
12	Accident	Bellanca	Personal	17-30A	0.0	
13	Accident	Cessna	Personal	R172K	1.0	
...	...	...	...	...	...	
90328	Accident	PIPER	NaN	PA42	0.0	
90332	Accident	CIRRUS DESIGN CORP	Personal	SR22	0.0	
90335	Accident	SWEARINGEN	NaN	SA226TC	0.0	
90336	Accident	CESSNA	Personal	R172K	0.0	
90345	Accident	AMERICAN CHAMPION AIRCRAFT	Personal	8GCBC	0.0	

31057 rows × 16 columns

In [31]:
```
df2.columns
```

Out[31]:
```
Index(['Investigation_Type', 'Make', 'Purpose_of_flight', 'Model',
       'Total_Fatal_Injuries', 'Total_Serious_Injuries',
       'Total_Minor_Injuries', 'Total_Uninjured', 'Weather_Condition',
       'Broad_phase_of_flight', 'Injury_Severity', 'Aircraft_damage',
       'Aircraft_Category', 'Registration_Number', 'Number_of_Engines',
       'Engine_Type'],
      dtype='object')
```

In [32]:
```
df2 = df2.replace('', np.nan)
df2= df2.replace(['None', 'NONE', 'N/A','NaN','nan'], np.nan)
df2
```

Out[32]:

	Investigation_Type	Make	Purpose_of_flight	Model	Total_Fatal_Injuries	Total_Seriou
5	Accident	Mcdonnell Douglas	NaN	DC9	NaN	
7	Accident	Cessna	Personal	140	0.0	
8	Accident	Cessna	Business	401B	0.0	
12	Accident	Bellanca	Personal	17-30A	0.0	
13	Accident	Cessna	Personal	R172K	1.0	
...	...	...	...	...	...	
90328	Accident	PIPER	NaN	PA42	0.0	
90332	Accident	CIRRUS DESIGN CORP	Personal	SR22	0.0	
90335	Accident	SWEARINGEN	NaN	SA226TC	0.0	
90336	Accident	CESSNA	Personal	R172K	0.0	
90345	Accident	AMERICAN CHAMPION AIRCRAFT	Personal	8GCBC	0.0	

31057 rows × 16 columns

1.4 Univariate Analysis

Categorical Culumns with Null Values

In [33]:
```
#PURPOSE OF FLIGHT
mode_value = df['Purpose_of_flight'].mode()[0]
df2['Purpose_of_flight'].fillna(mode_value, inplace=True) #inplace=True modifies the
df2['Purpose_of_flight'].isnull().sum()
```

Out[33]:
```
0
```

In [34]:
```
#PHASE OF FLIGHT
mode_value = df['Broad_phase_of_flight'].mode()[0]
df2['Broad_phase_of_flight'].fillna(mode_value, inplace=True)
df2['Broad_phase_of_flight'].isnull().sum()
```

Out[34]: 0

In [35]:
```
#INJURY SEVERITY
mode_value = df['Injury_Severity'].mode()[0]
df2['Injury_Severity'].fillna(mode_value, inplace=True)
df2['Injury_Severity'].isnull().sum()
```

Out[35]: 0

In [36]:
```
#ENGINE TYPE
mode_value = df['Engine_Type'].mode()[0]
df2['Engine_Type'].fillna(mode_value, inplace=True)
df2['Engine_Type'].isnull().sum()
```

Out[36]: 0

In [37]:
```
#AIRCRAFT CATEGORY
mode_value = df['Aircraft_Category'].mode()[0]
df2['Aircraft_Category'].fillna(mode_value, inplace=True)
df2['Aircraft_Category'].isnull().sum()
```

Out[37]: 0

In [38]:
```
#PHASE OF FLIGHT
mode_value = df['Broad_phase_of_flight'].mode()[0]
df2['Broad_phase_of_flight'].fillna(mode_value, inplace=True)
df2['Broad_phase_of_flight'].isnull().sum()
```

Out[38]: 0

In [39]:
```
#INVESTIGATION TYPE
mode_value = df['Investigation_Type'].mode()[0]
df2['Investigation_Type'].fillna(mode_value, inplace=True)
df2['Investigation_Type'].isnull().sum()
```

Out[39]: 0

In [40]:
```
#AIRCRAFT DAMAGE
mode_value = df['Aircraft_damage'].mode()[0]
df2['Aircraft_damage'].fillna(mode_value, inplace=True)
df2['Aircraft_damage'].isnull().sum()
```

Out[40]: 0

In [41]:
```
df2['Weather_Condition'] = df['Weather_Condition'].replace('UNK', 'Unk')
print(df2['Weather_Condition'].value_counts())
```

```
VMC    25484
IMC     1522
Unk      435
Name: Weather_Condition, dtype: int64
```

NB

In aviation weather, VMC stands for Visual Meteorological Conditions.

Essentially, VMC refers to the weather conditions under which a pilot can operate an aircraft primarily by visual reference to the ground, water, and other landmarks, as well as by visually avoiding obstacles and other aircraft. These are the conditions that allow for Visual Flight Rules (VFR) flight.

IMC (Instrument Meteorological Conditions): The opposite of VMC. These are weather conditions where visibility, cloud clearance, or ceiling are below the VMC minima. In IMC, pilots must rely on their aircraft's instruments for navigation and control, and must operate under Instrument Flight Rules (IFR).

In [42]:
```
df2['Make'] = df['Make'].str.title()
df2['Make']
```

Out[42]:
```
5                   Mcdonnell Douglas
7                             Cessna
8                             Cessna
12                          Bellanca
13                            Cessna
                      ...
90328                          Piper
90332            Cirrus Design Corp
90335                     Swearingen
90336                         Cessna
90345     American Champion Aircraft
Name: Make, Length: 31057, dtype: object
```

In [43]:
```
df2['Aircraft_Category'].value_counts()
```

Out[43]:
```
Airplane      27617
Helicopter     3440
Name: Aircraft_Category, dtype: int64
```

In [44]:
```
df2['Purpose_of_flight'].value_counts()
```

Out[44]:
```
Personal                     21266
Instructional                 3740
Aerial Application            1386
Unknown                       1119
Business                       915
Positioning                    551
Other Work Use                 334
Aerial Observation             313
Flight Test                    266
Ferry                          195
Executive/corporate            172
Skydiving                      166
External Load                  105
Banner Tow                      89
Public Aircraft - Federal       83
Air Race show                   77
Public Aircraft - Local         65
Public Aircraft                 65
Public Aircraft - State         54
Glider Tow                      35
Firefighting                    34
Air Race/show                    8
Air Drop                         8
ASHO                             6
PUBS                             4
PUBL                             1
Name: Purpose_of_flight, dtype: int64
```

Numerical Columns with Null Values

In [45]:
```
mean_fatal_injuries = df2['Total_Fatal_Injuries'].mean().round()
df2['Total_Fatal_Injuries_Filled'] = df2['Total_Fatal_Injuries'].fillna(mean_fatal_i
df2['Total_Fatal_Injuries_Filled'].isnull().sum()
```

Out[45]:
```
0
```

In [46]:
```
mean_serious_injuries = (df2['Total_Serious_Injuries'].mean().round()+1)
df2['Total_Serious_Injuries'] = df2['Total_Serious_Injuries'].fillna(mean_serious_in
df2['Total_Serious_Injuries'].isnull().sum()
```

Out[46]:
```
0
```

In [47]:
```
mean_minor_injuries = (df2['Total_Minor_Injuries'].mean().round()+1)
df2['Total_Minor_Injuries'] = df2['Total_Minor_Injuries'].fillna(mean_minor_injuries
df2['Total_Minor_Injuries'].isnull().sum()
```

Out[47]:
```
0
```

In [48]:
```
#For Saftey purposes the median(1) was used as opposed to mean(6) which gives a
median_uninjured = df2['Total_Uninjured'].median().round()
df2['Total_Uninjured'] = df2['Total_Uninjured'].fillna(median_uninjured)
df2['Total_Uninjured'].isnull().sum()
```

Out[48]:
```
0
```

NB

- For Saftey purposes the median(1) was used as opposed to mean(6) to fill in the missing values in the total uninjured passengers.

In [49]:
```
#Replace PIPER with Piper
df2['Make'] = df['Make'].replace('PIPER', 'Piper')
df2['Make'].value_counts()
```

Out[49]:
```
CESSNA                             4867
Piper                              4715
Cessna                             3608
BOEING                             1039
BEECH                              1018
                                   ...
HENDERSON                             1
FREEMAN HERITAGE COLLECTION           1
CHAPMAN MARK A                        1
Advertising MGMT & Consulting         1
American General Aircraft             1
Name: Make, Length: 4171, dtype: int64
```

In [50]:
```
#Clean Weather to title case
df2['Weather_Condition'] = df['Weather_Condition'].replace('UNK', 'Unk')
df2['Weather_Condition'].value_counts()
```

Out[50]:
```
VMC    25484
IMC     1522
Unk      435
Name: Weather_Condition, dtype: int64
```

In [51]:
```
#Clean Make by replacing Cessna and Piper to title case
# Replace 'CESSNA' with 'Cessna' in the 'Make' column
df2['Make'] = df['Make'].replace('CESSNA', 'Cessna')
df2['Make'].value_counts()
```

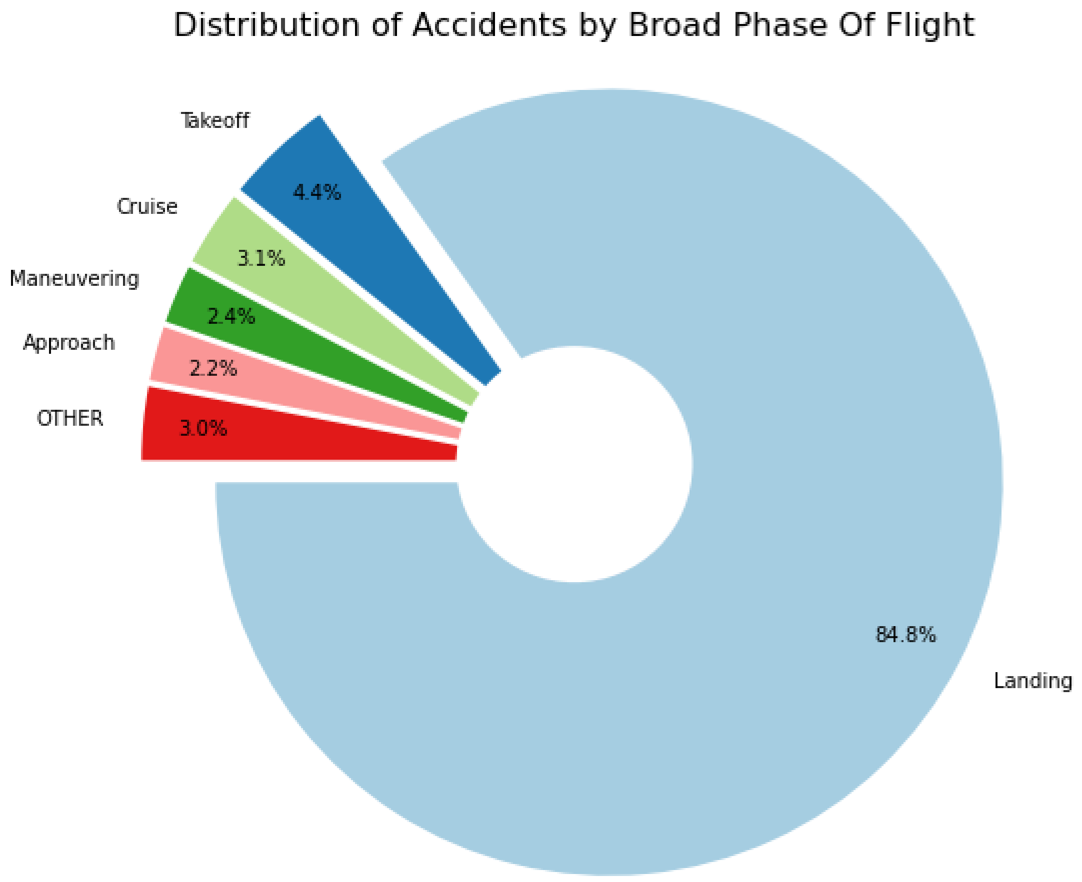
Out[51]:
```
Cessna                             8475
PIPER                              2805
Piper                              1910
BOEING                             1039
BEECH                              1018
                                   ...
HENDERSON                             1
FREEMAN HERITAGE COLLECTION           1
```

```
CHAPMAN MARK A                    1
Advertising MGMT & Consulting     1
American General Aircraft         1
Name: Make, Length: 4171, dtype: int64
```

1.4.1 Accidents per Flight Phase [Broad_phase_of_flight]

In [52]:

```
#Column
category_column = 'Broad_phase_of_flight'

#counts of each
category_counts = df2['Broad_phase_of_flight'].value_counts(dropna=False) # Keep NaN


# Show categories that make up more than 1%
threshold = 0.01 * category_counts.sum()
main_categories = category_counts[category_counts >= threshold]

#combine smaller categories(<1%) into an "Other" slice
other_count = category_counts[category_counts < threshold].sum()

if other_count > 0:
    plot_data = pd.concat([main_categories, pd.Series({'OTHER': other_count})])
else:
        plot_data = main_categories

labels = plot_data.index
sizes = plot_data.values


#Explode" a slice by 0.1
explode = (0.1, 0, 0, 0)
explode = [0.1 for _ in range(len(labels))]


# Donut Chart
plt.figure(figsize=(10, 8))

plt.pie(sizes,labels=labels,autopct='%1.1f%%', startangle=180,
            colors=plt.cm.Paired.colors,
            explode=explode,
            pctdistance=0.85)


centre_circle = plt.Circle((0,0), 0.30, fc='white')
fig = plt.gcf()
fig.gca().add_artist(centre_circle)

plt.title(f'Distribution of Accidents by {category_column.replace("_", " ").title()}
plt.axis('equal')
plt.show()
```



Most entries recorded involed Airplanes an Helicopters within the Landing phase of flight followed by Take off then cruise.

In [53]:
```
#Dropping Total Fatal Injuries to use the Total_injuries_ filled column.
df2 = df2.drop(columns=['Total_Fatal_Injuries'])
df2
```

Out[53]:

	Investigation_Type	Make	Purpose_of_flight	Model	Total_Serious_Injuries	Total_Min
5	Accident	Mcdonnell Douglas	Personal	DC9	1.0	
7	Accident	Cessna	Personal	140	0.0	
8	Accident	Cessna	Business	401B	0.0	
12	Accident	Bellanca	Personal	17-30A	0.0	
13	Accident	Cessna	Personal	R172K	0.0	
...	...	...	...	...	...	
90328	Accident	PIPER	Personal	PA42	0.0	
90332	Accident	CIRRUS DESIGN CORP	Personal	SR22	0.0	
90335	Accident	SWEARINGEN	Personal	SA226TC	0.0	
90336	Accident	Cessna	Personal	R172K	1.0	
90345	Accident	AMERICAN CHAMPION AIRCRAFT	Personal	8GCBC	0.0	

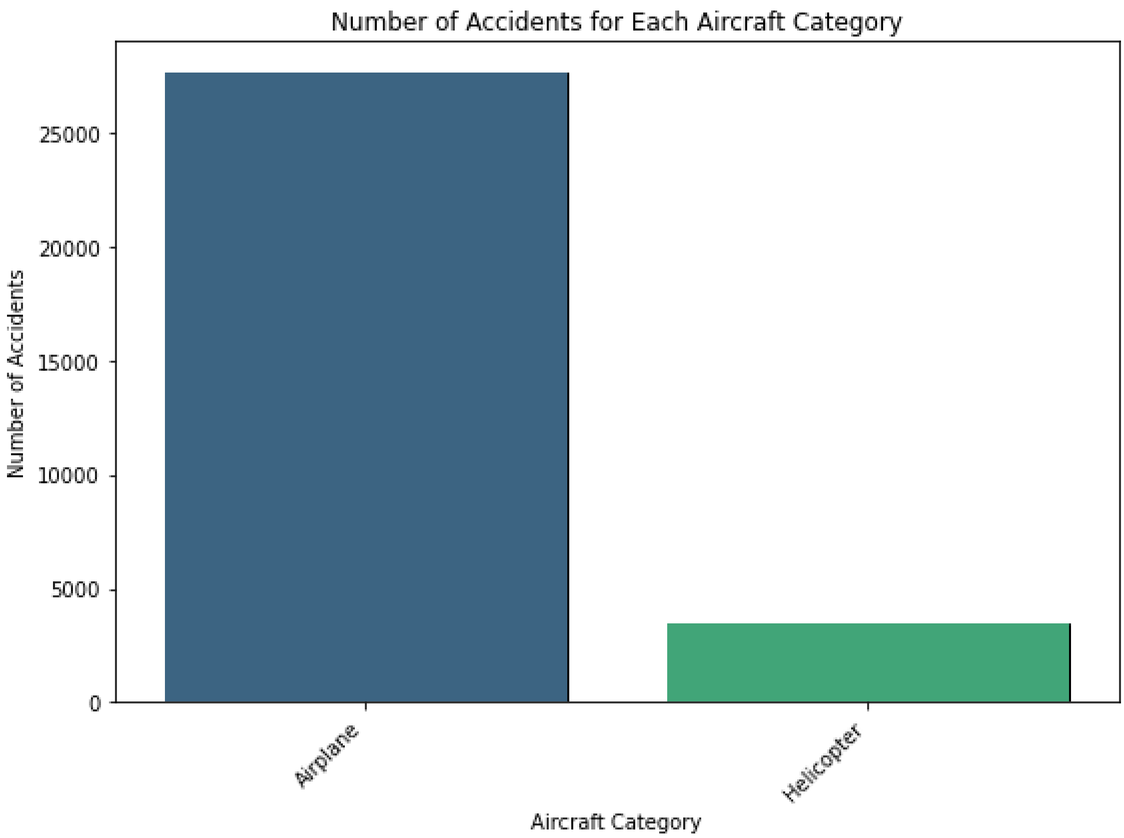
31057 rows × 16 columns

1.5 Bivariate Analysis

1.5.1 Analysis by Aircraft Category

In [54]:

```
make_counts = df2['Aircraft_Category'].value_counts()

plt.figure(figsize=(8, 6))

plt.bar(make_counts.index, make_counts.values, color='skyblue', edgecolor='black')

sns.barplot(x=make_counts.index, y=make_counts.values, palette='viridis')

plt.xlabel('Aircraft Category')
plt.ylabel('Number of Accidents')
plt.title('Number of Accidents for Each Aircraft Category')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```



Observation:

- Helicopters are involved in less accidents with less than 5000 reported.
- Airplanes have over 25,000 accidents recorded.

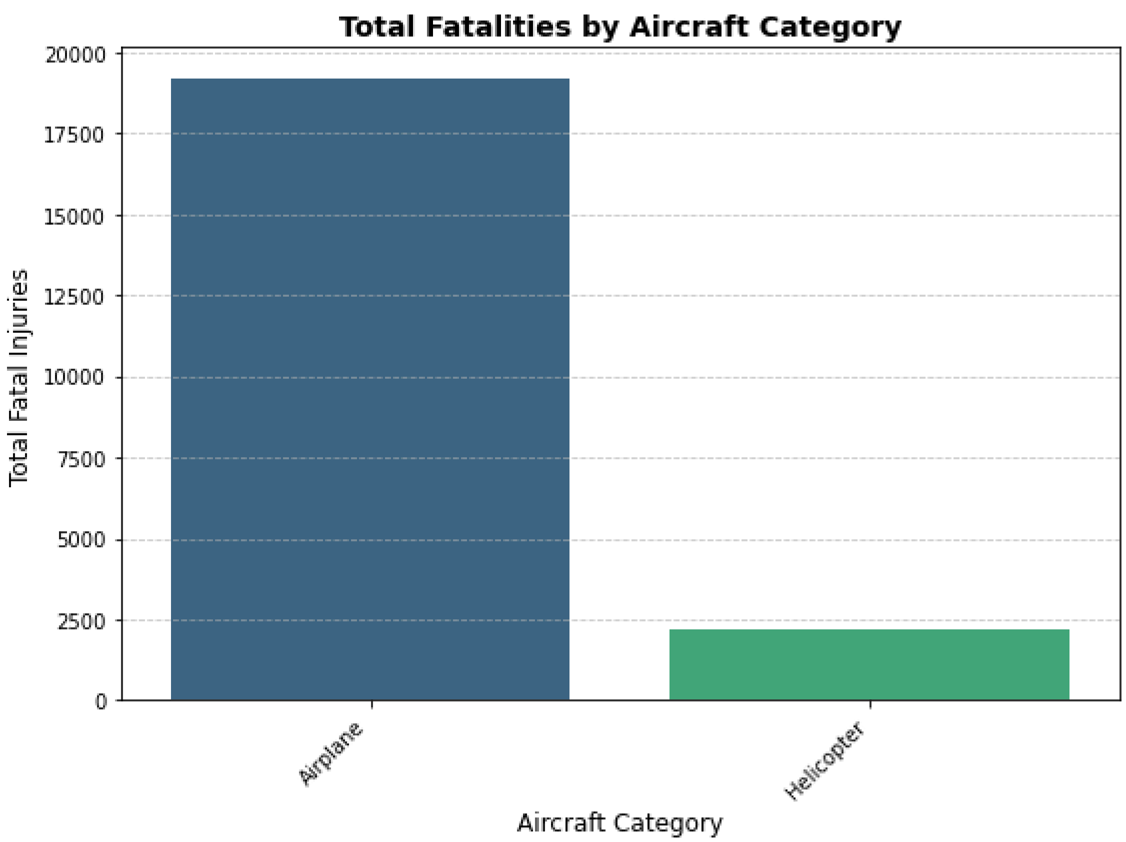
In [55]:

```
total_fatalities_by_category = df2.groupby('Aircraft_Category')['Total_Fatal_Injurie
total_fatalities_by_category
```

Out[55]:

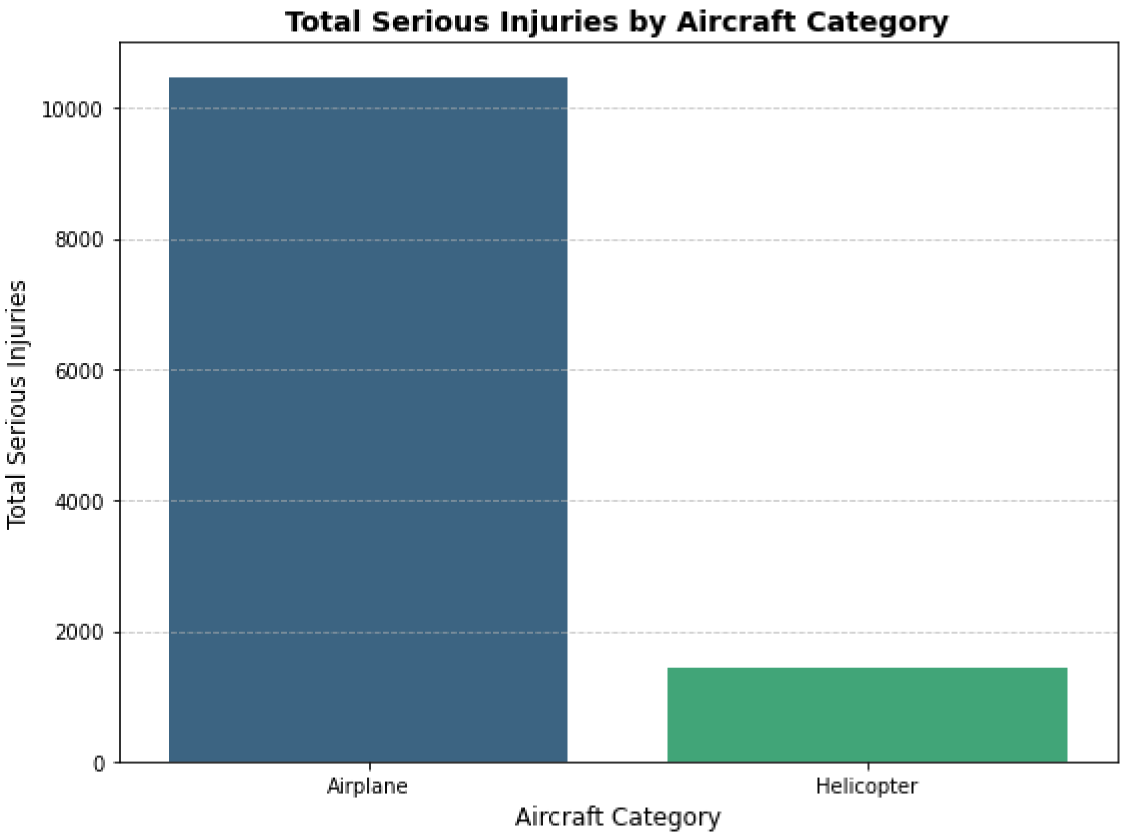
```
Aircraft_Category
Airplane      19194.0
Helicopter     2166.0
Name: Total_Fatal_Injuries_Filled, dtype: float64
```

In [56]:
```
#Total fatalities per aircraft category

total_fatalities_by_category = df2.groupby('Aircraft_Category')['Total_Fatal_Injurie


plt.figure(figsize=(8, 6))

sns.barplot(x=total_fatalities_by_category.index,
            y=total_fatalities_by_category.values,
            palette='viridis')

plt.xlabel('Aircraft Category', fontsize=12)
plt.ylabel('Total Fatal Injuries', fontsize=12)
plt.title('Total Fatalities by Aircraft Category', fontsize=14, fontweight='bold')

# Rotate x-axis
plt.xticks(rotation=45, ha='right', fontsize=10)
plt.yticks(fontsize=10)

plt.grid(axis='y', linestyle='--', alpha=0.7)

plt.tight_layout()
plt.show()
```



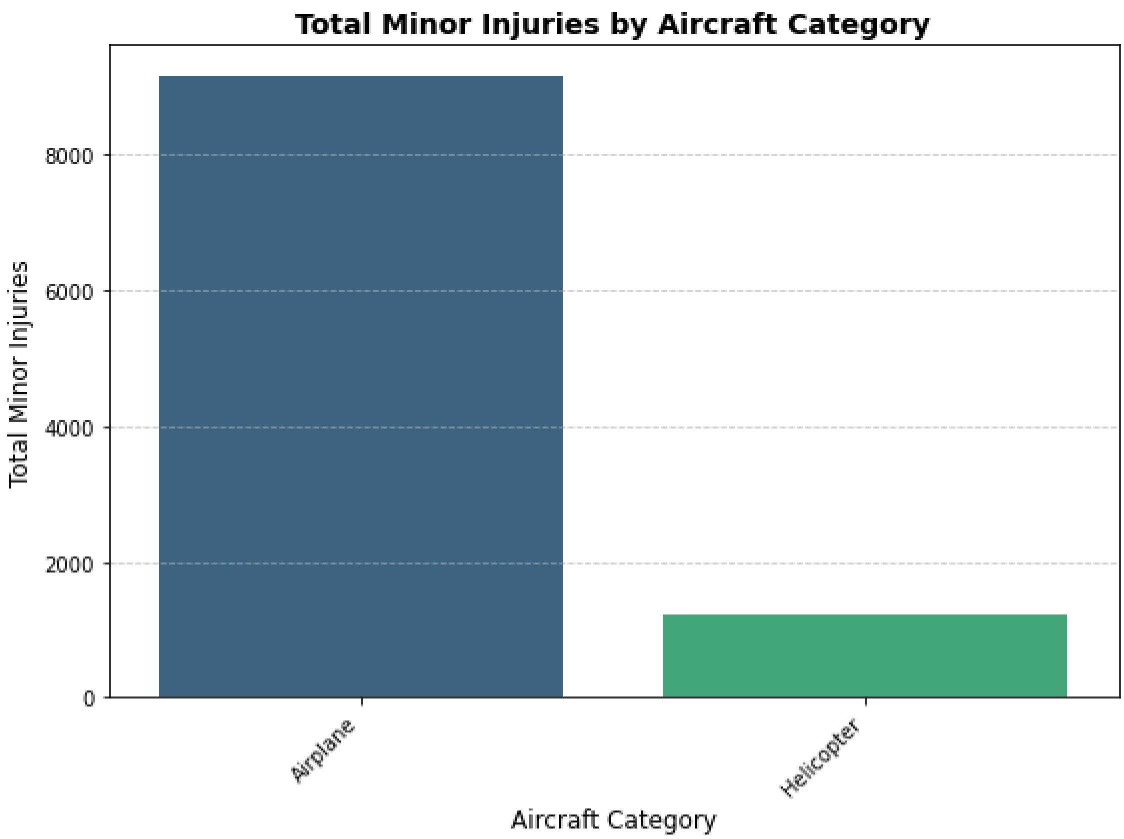
In [57]:
```
total_serious_by_category = df2.groupby('Aircraft_Category')['Total_Serious_Injuries


plt.figure(figsize=(8, 6))

sns.barplot(x=total_serious_by_category.index,
            y=total_serious_by_category.values,
            palette='viridis')

plt.xlabel('Aircraft Category', fontsize=12)
plt.ylabel('Total Serious Injuries', fontsize=12)
plt.title('Total Serious Injuries by Aircraft Category', fontsize=14, fontweight='bo
```

```
# Rotate x-axis
#plt.xticks(rotation=45, ha='right', fontsize=10)
plt.yticks(fontsize=10)

plt.grid(axis='y', linestyle='--', alpha=0.7)

plt.tight_layout()
plt.show()
```



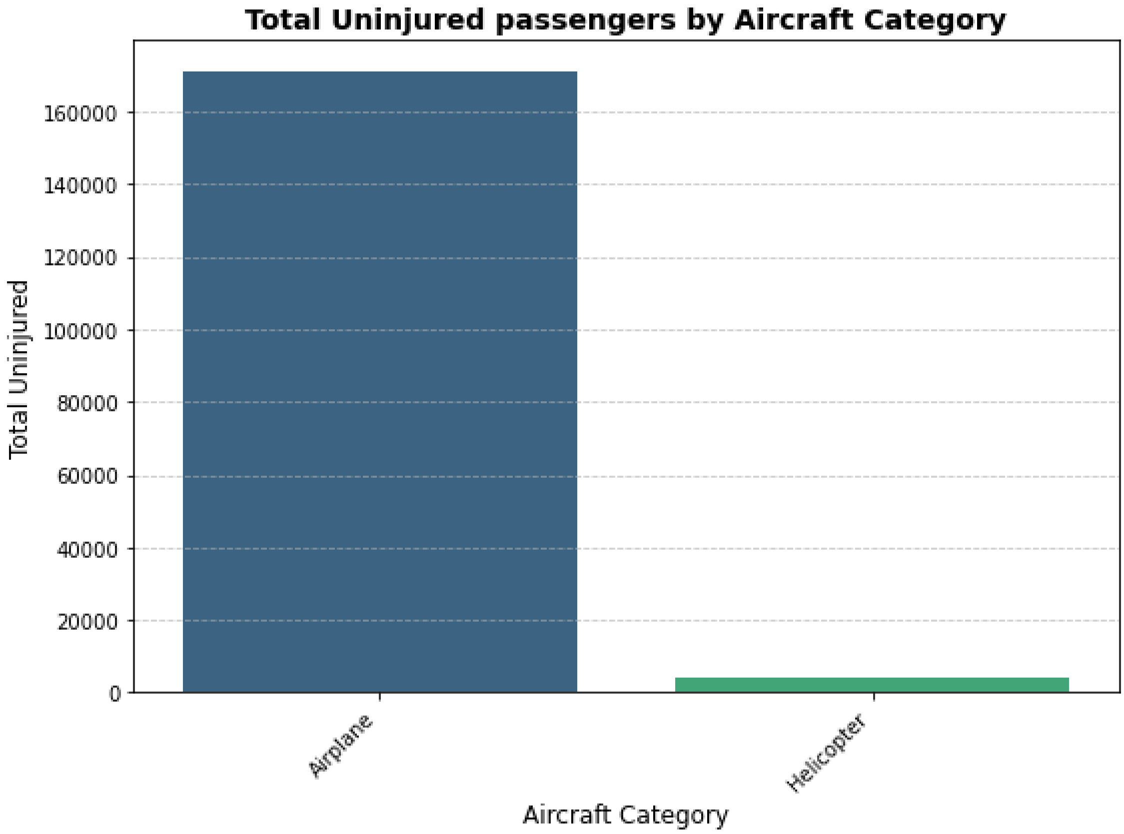
In [58]:

```
total_minor_by_category = df2.groupby('Aircraft_Category')['Total_Minor_Injuries'].s


plt.figure(figsize=(8, 6))

sns.barplot(x=total_minor_by_category.index,
            y=total_minor_by_category.values,
            palette='viridis')

plt.xlabel('Aircraft Category', fontsize=12)
plt.ylabel('Total Minor Injuries', fontsize=12)
plt.title('Total Minor Injuries by Aircraft Category', fontsize=14, fontweight='bold

# Rotate x-axis
plt.xticks(rotation=45, ha='right', fontsize=10)
plt.yticks(fontsize=10)

plt.grid(axis='y', linestyle='--', alpha=0.7)

plt.tight_layout()
plt.show()
```



In [59]:

```
total_uninjured_by_category = df2.groupby('Aircraft_Category')['Total_Uninjured'].su


plt.figure(figsize=(8, 6))

sns.barplot(x=total_uninjured_by_category.index,
            y=total_uninjured_by_category.values,
            palette='viridis')

plt.xlabel('Aircraft Category', fontsize=12)
plt.ylabel('Total Uninjured', fontsize=12)
plt.title('Total Uninjured passengers by Aircraft Category', fontsize=14, fontweight

# Rotate x-axis
plt.xticks(rotation=45, ha='right', fontsize=10)
plt.yticks(fontsize=10)

plt.grid(axis='y', linestyle='--', alpha=0.7)

plt.tight_layout()
plt.show()
```



In [60]:
```
total_uninjured_by_category
```

Out[60]:
```
Aircraft_Category
Airplane      171011.0
Helicopter      4008.0
Name: Total_Uninjured, dtype: float64
```

In [61]:
```
total_serious_by_category
```

Out[61]:
```
Aircraft_Category
Airplane      10486.0
Helicopter     1424.0
Name: Total_Serious_Injuries, dtype: float64
```

In [62]:
```
total_minor_by_category
```

Out[62]:
```
Aircraft_Category
Airplane      9164.0
Helicopter    1226.0
Name: Total_Minor_Injuries, dtype: float64
```

In [63]:
```
total_fatalities_by_category
```

Out[63]:
```
Aircraft_Category
Airplane      19194.0
Helicopter     2166.0
Name: Total_Fatal_Injuries_Filled, dtype: float64
```

In [64]:
```
df2['Aircraft_Category'].value_counts()
```

Out[64]:
```
Airplane      27617
Helicopter     3440
Name: Aircraft_Category, dtype: int64
```

Observation:

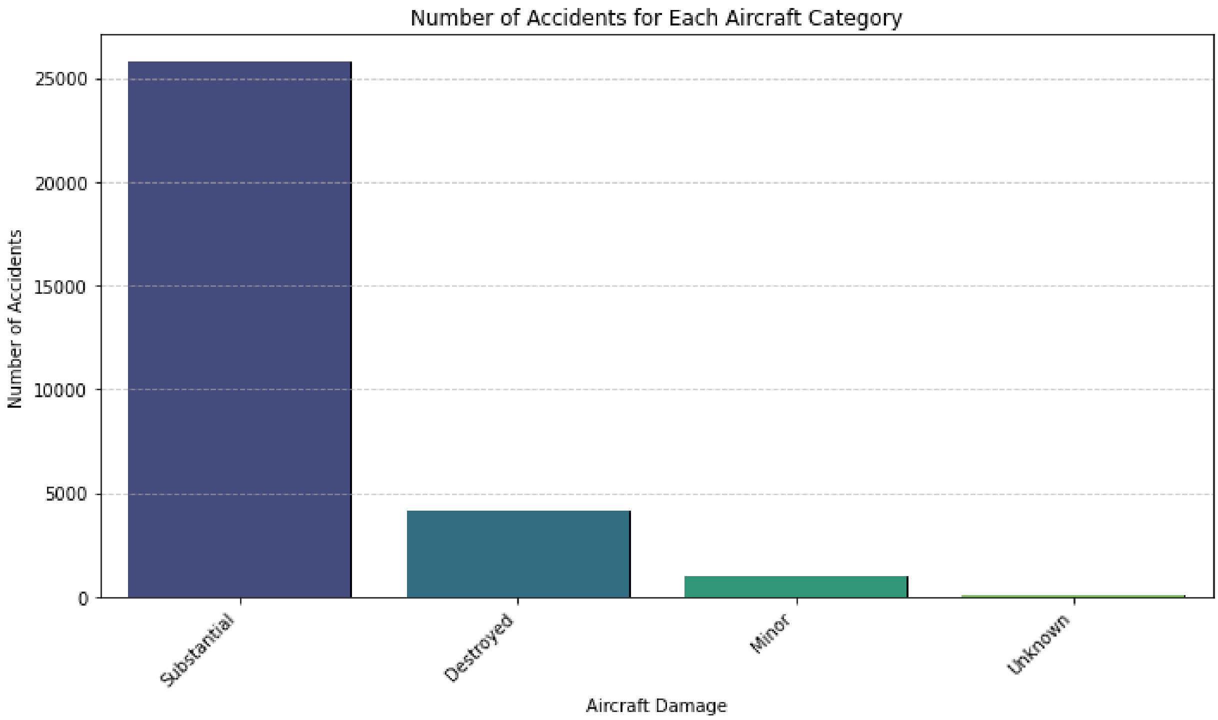
- Airplane (27617 Entries)
  - 171011 uninjured passengers

- 10486 seriously injured passengers
- 9164 passengers with minor injuries
- 19194 fatally injured passengers

- Helicopter(3440 Entries)
  - 4008 uninjured passengers
  - 1424 seriously injured passengers
  - 1226 passengers with minor injuries
  - 2166 fatally injured passengers

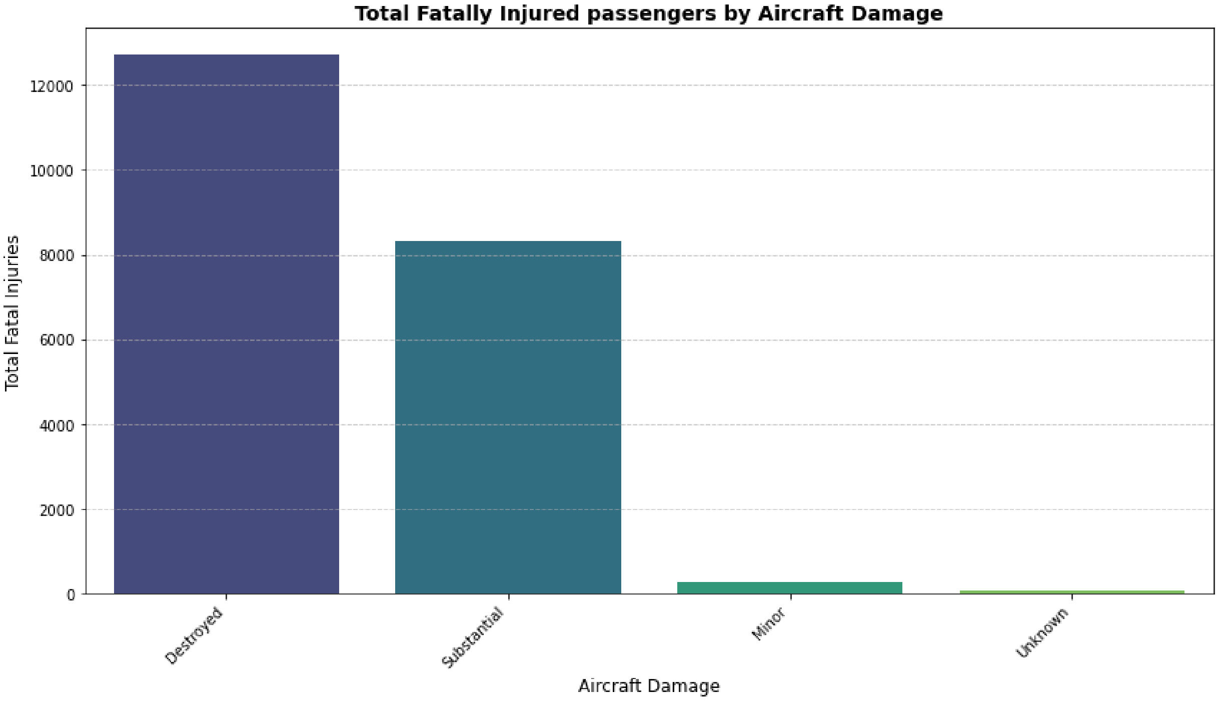
1.5.2 Analysis by Aircraft Damage

In [65]:
```
make_counts = df2['Aircraft_damage'].value_counts()

plt.figure(figsize=(10, 6))

plt.bar(make_counts.index, make_counts.values, color='skyblue', edgecolor='black')

sns.barplot(x=make_counts.index, y=make_counts.values, palette='viridis')

plt.xlabel('Aircraft Damage')
plt.ylabel('Number of Accidents')
plt.title('Number of Accidents for Each Aircraft Category')
plt.xticks(rotation=45, ha='right')
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()
```



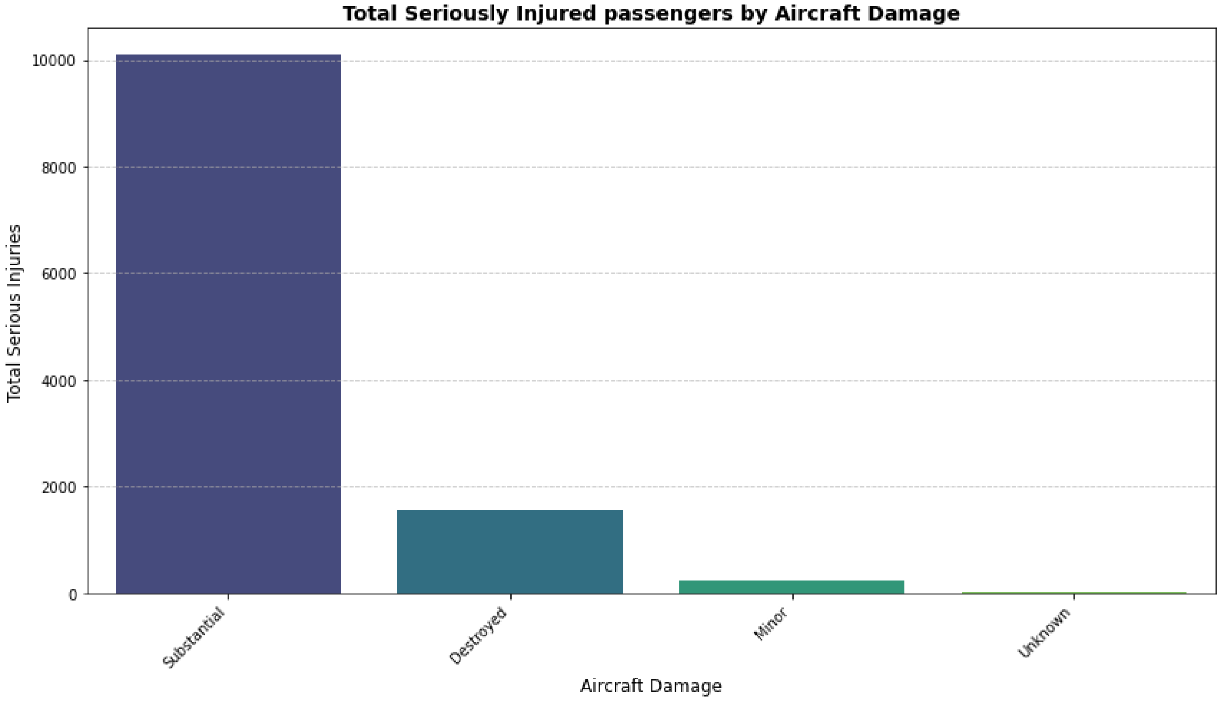
In [66]:
```
total_fatalities_by_damage = df2.groupby('Aircraft_damage')['Total_Fatal_Injuries_Fi


plt.figure(figsize=(12, 7))

sns.barplot(x=total_fatalities_by_damage.index,
            y=total_fatalities_by_damage.values,
            palette='viridis')

plt.xlabel('Aircraft Damage', fontsize=12)
```

```
plt.ylabel('Total Fatal Injuries', fontsize=12)
plt.title('Total Fatally Injured passengers by Aircraft Damage', fontsize=14, fontwe

# Rotate x-axis
plt.xticks(rotation=45, ha='right', fontsize=10)
plt.yticks(fontsize=10)

plt.grid(axis='y', linestyle='--', alpha=0.7)

plt.tight_layout()
plt.show()
```



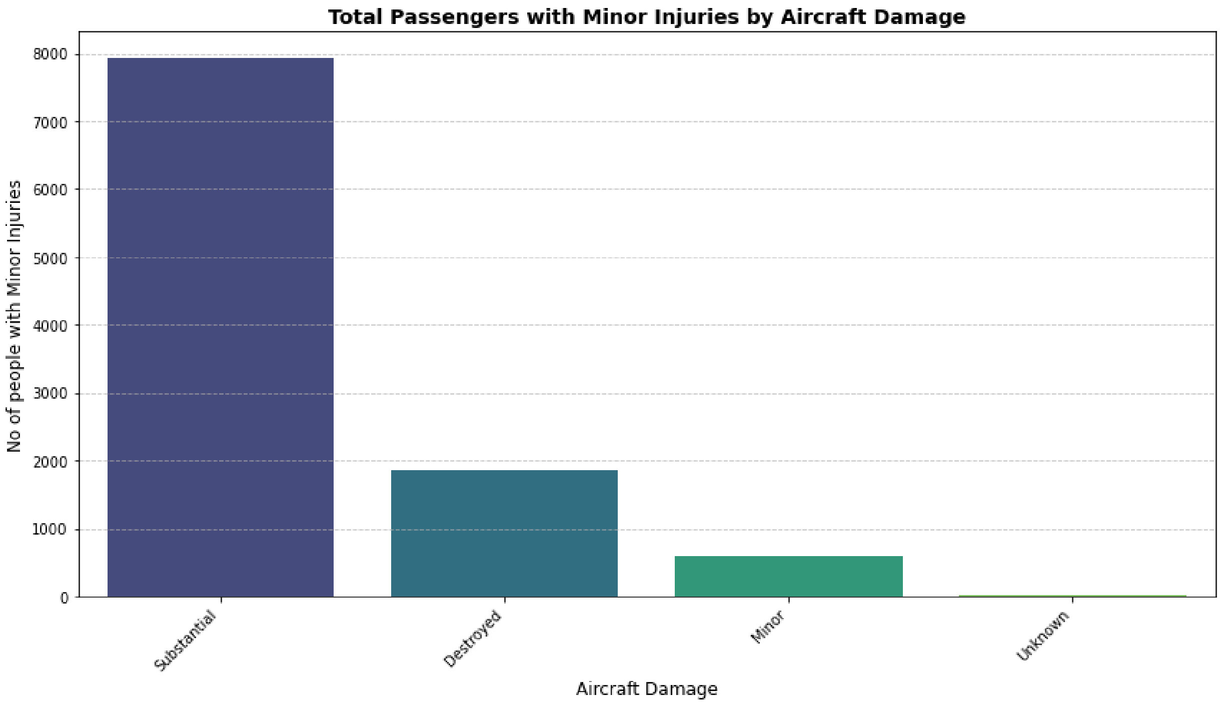
In [67]:

```
total_serious_by_damage = df2.groupby('Aircraft_damage')['Total_Serious_Injuries'].s


plt.figure(figsize=(12, 7))

sns.barplot(x=total_serious_by_damage.index,
            y=total_serious_by_damage.values,
            palette='viridis')

plt.xlabel('Aircraft Damage', fontsize=12)
plt.ylabel('Total Serious Injuries', fontsize=12)
plt.title('Total Seriously Injured passengers by Aircraft Damage', fontsize=14, font

# Rotate x-axis
plt.xticks(rotation=45, ha='right', fontsize=10)
plt.yticks(fontsize=10)

plt.grid(axis='y', linestyle='--', alpha=0.7)

plt.tight_layout()
plt.show()
```

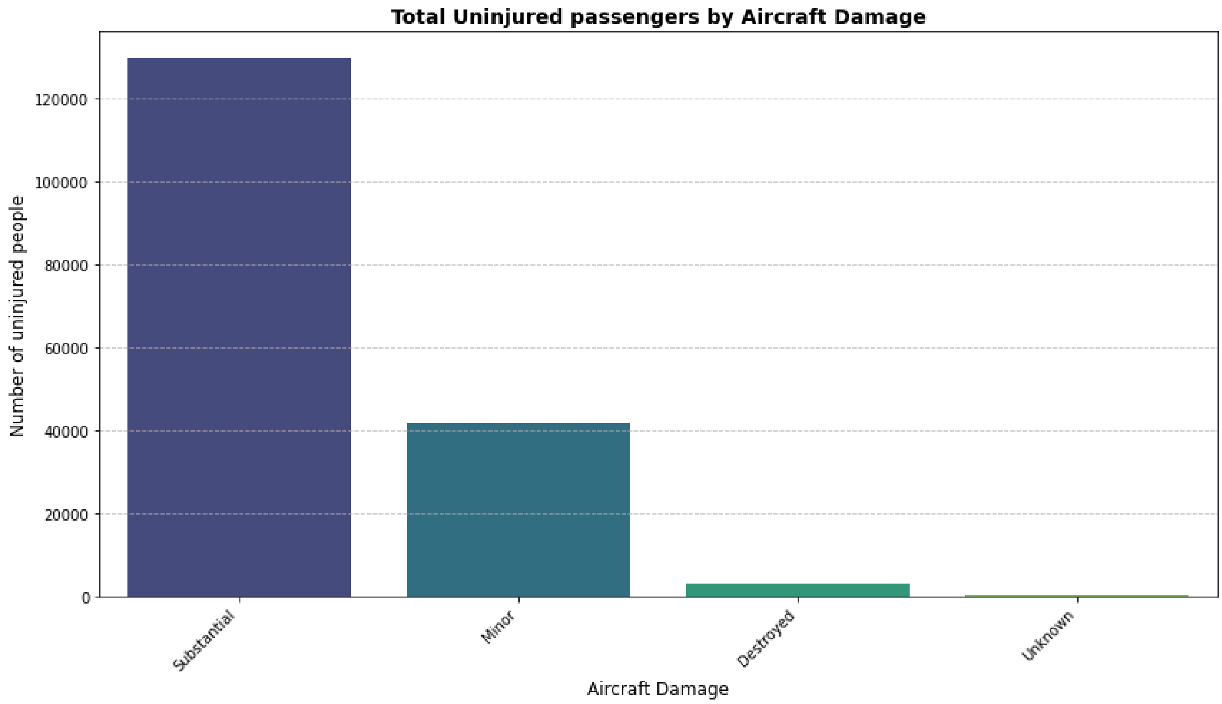


In [68]:

```
total_minor_by_damage = df2.groupby('Aircraft_damage')['Total_Minor_Injuries'].sum()


plt.figure(figsize=(12, 7))

sns.barplot(x=total_minor_by_damage.index,
            y=total_minor_by_damage.values,
            palette='viridis')

plt.xlabel('Aircraft Damage', fontsize=12)
plt.ylabel('No of people with Minor Injuries', fontsize=12)
plt.title('Total Passengers with Minor Injuries by Aircraft Damage', fontsize=14, fo

# Rotate x-axis
plt.xticks(rotation=45, ha='right', fontsize=10)
plt.yticks(fontsize=10)

plt.grid(axis='y', linestyle='--', alpha=0.7)

plt.tight_layout()
plt.show()
```



In [69]:
```python
total_uninjured_by_damage = df2.groupby('Aircraft_damage')['Total_Uninjured'].sum().


plt.figure(figsize=(12, 7))

sns.barplot(x=total_uninjured_by_damage.index,
            y=total_uninjured_by_damage.values,
            palette='viridis')

plt.xlabel('Aircraft Damage', fontsize=12)
plt.ylabel('Number of uninjured people', fontsize=12)
plt.title('Total Uninjured passengers by Aircraft Damage', fontsize=14, fontweight='

# Rotate x-axis
plt.xticks(rotation=45, ha='right', fontsize=10)
plt.yticks(fontsize=10)

plt.grid(axis='y', linestyle='--', alpha=0.7)

plt.tight_layout()
plt.show()
```



In [70]:
```
total_uninjured_by_damage
```

Out[70]:
```
Aircraft_damage
Substantial    129693.0
Minor           41866.0
Destroyed        2983.0
Unknown           477.0
Name: Total_Uninjured, dtype: float64
```

In [71]:
```
total_serious_by_damage
```

Out[71]:
```
Aircraft_damage
Substantial    10089.0
Destroyed       1558.0
Minor            247.0
Unknown           16.0
Name: Total_Serious_Injuries, dtype: float64
```

In [72]:
```
total_minor_by_damage
```

Out[72]:
```
Aircraft_damage
Substantial    7923.0
Destroyed      1850.0
Minor           592.0
Unknown          25.0
Name: Total_Minor_Injuries, dtype: float64
```

In [73]:
```
total_fatalities_by_damage
```

Out[73]:
```
Aircraft_damage
Destroyed      12716.0
Substantial     8320.0
Minor            269.0
Unknown           55.0
Name: Total_Fatal_Injuries_Filled, dtype: float64
```

In [74]:
```
df2['Aircraft_damage'].value_counts()
```

Out[74]:
```
Substantial    25792
Destroyed       4121
Minor           1029
Unknown          115
Name: Aircraft_damage, dtype: int64
```

Observation

1. Substantial Damage
    - 129,693 uninjured
    - 10,089 seriously Injured
    - 7923 minor Injuries
    - 5015 fatal injuries

1. Destroyed Damage
    - 2983 uninjured
    - 1558 Serious
    - 1850 Minor Injuries
    - 12558 fatal injuries

1. Minor Damage
    - 41866 uninjured
    - 247 Serious
    - 592 Minor Injuries
    - 179 fatal injuries

In [75]:
```
df2.columns
```

Out[75]:
```
Index(['Investigation_Type', 'Make', 'Purpose_of_flight', 'Model',
       'Total_Serious_Injuries', 'Total_Minor_Injuries', 'Total_Uninjured',
       'Weather_Condition', 'Broad_phase_of_flight', 'Injury_Severity',
       'Aircraft_damage', 'Aircraft_Category', 'Registration_Number',
       'Number_of_Engines', 'Engine_Type', 'Total_Fatal_Injuries_Filled'],
      dtype='object')
```

1.6 Multivariate Analysis

1.6.1 Analysis by Aircraft Model and Make

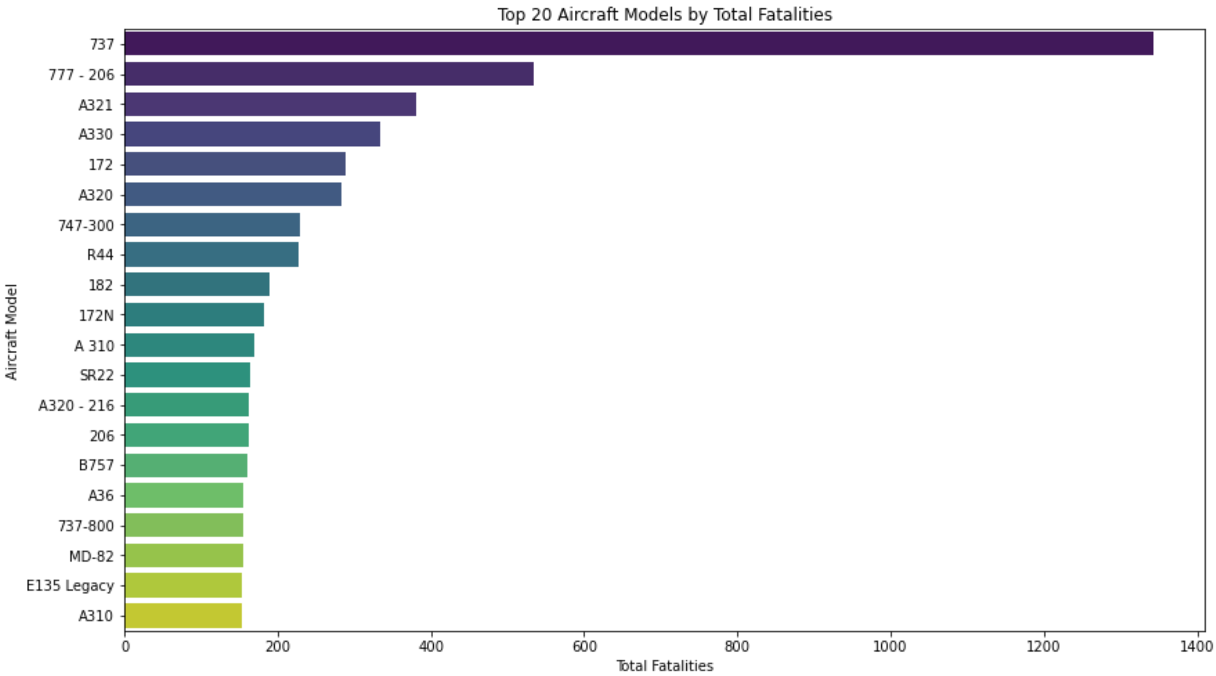
In [76]:
```
#Total Fatalities per Model
fatalities_by_model = df2.groupby('Model')['Total_Fatal_Injuries_Filled'].sum().sort
df_fatal_accidents = df2[df2['Total_Fatal_Injuries_Filled'] > 0].copy()
print("\n--- Safest Aircraft by Total Fatalities ---")
print(fatalities_by_model.head(15))
```

```
--- Safest Aircraft by Total Fatalities ---
Model
Gemini Remos          0.0
PULSAR XP SERIES I    0.0
PULSAR XP             0.0
DA20 - C1             0.0
PULSAR SPORT 150      0.0
PULSAR SERIES III     0.0
DA20C1                0.0
PULSAR   912XP        0.0
PTUNDRADACTYL         0.0
PT2                   0.0
DA40F                 0.0
DA42                  0.0
PT17                  0.0
DAKATO HAWK           0.0
DAKOTA HAWK           0.0
Name: Total_Fatal_Injuries_Filled, dtype: float64
```

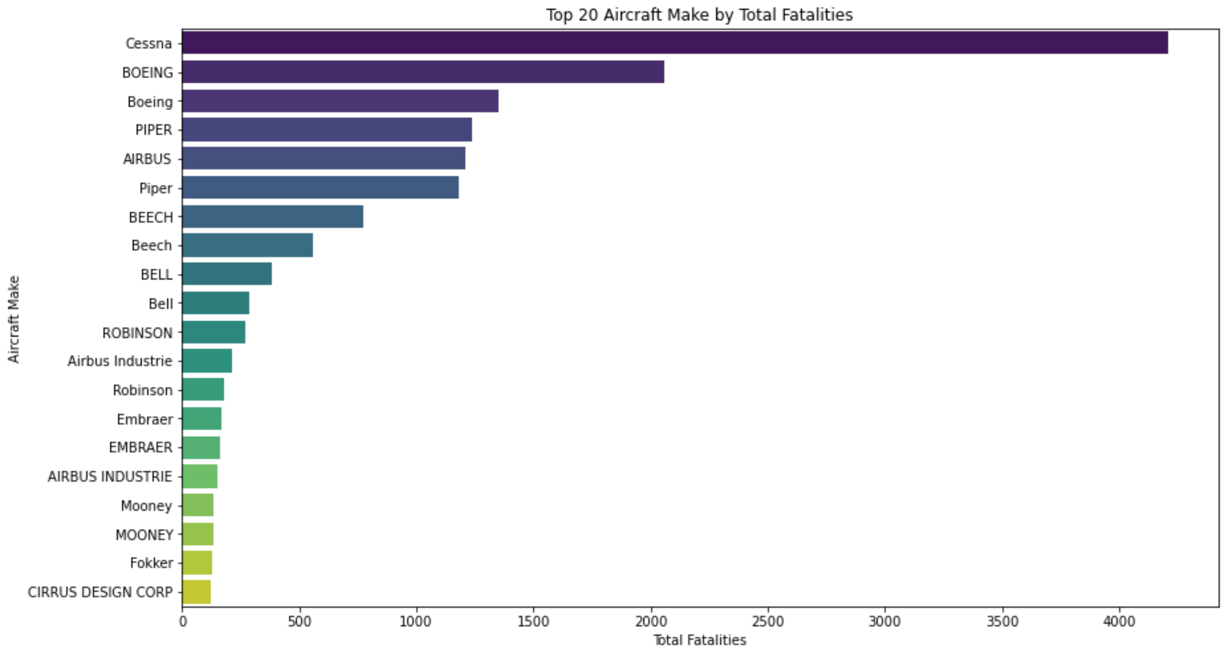
In [77]:
```
fatalities_by_model.tail(10)
```

Out[77]:
```
Model
172N          182.0
182           189.0
R44           228.0
747-300       230.0
A320          284.0
172           289.0
A330          334.0
A321          381.0
777 - 206     534.0
737          1343.0
Name: Total_Fatal_Injuries_Filled, dtype: float64
```

In [78]:
```
top_models_by_fatalities = fatalities_by_model.sort_values(ascending=False).head(20)
plt.figure(figsize=(14, 8))
sns.barplot(x=top_models_by_fatalities.values, y=top_models_by_fatalities.index, pal
plt.xlabel('Total Fatalities')
plt.ylabel('Aircraft Model')
plt.title('Top 20 Aircraft Models by Total Fatalities')
plt.show()
```



In [79]:
```
fatalities_by_make = df2.groupby('Make')['Total_Fatal_Injuries_Filled'].sum().sort_v
df_fatal_accidents = df2[df2['Total_Fatal_Injuries_Filled'] > 0].copy()
top_make_by_fatalities = fatalities_by_make.sort_values(ascending=False).head(20)
plt.figure(figsize=(14, 8))
sns.barplot(x=top_make_by_fatalities.values, y=top_make_by_fatalities.index, palette
plt.xlabel('Total Fatalities')
plt.ylabel('Aircraft Make')
plt.title('Top 20 Aircraft Make by Total Fatalities')
plt.show()
```



In [80]:

```
#Total Accidents (Fatal + Non-Fatal) per Aircraft Model
# Useful for context, as some aircraft might have many non-fatal incidents but few f
df2['Total_sum_injuries'] = df2['Total_Fatal_Injuries_Filled'] + df2['Total_Serious_

df2
```

Out[80]:

	Investigation_Type	Make	Purpose_of_flight	Model	Total_Serious_Injuries	Total_Min
5	Accident	Mcdonnell Douglas	Personal	DC9	1.0	
7	Accident	Cessna	Personal	140	0.0	
8	Accident	Cessna	Business	401B	0.0	
12	Accident	Bellanca	Personal	17-30A	0.0	
13	Accident	Cessna	Personal	R172K	0.0	
...	...	...	...	...	...	
90328	Accident	PIPER	Personal	PA42	0.0	
90332	Accident	CIRRUS DESIGN CORP	Personal	SR22	0.0	
90335	Accident	SWEARINGEN	Personal	SA226TC	0.0	
90336	Accident	Cessna	Personal	R172K	1.0	
90345	Accident	AMERICAN CHAMPION AIRCRAFT	Personal	8GCBC	0.0	

31057 rows × 17 columns

In [81]:

```
#Export Data to use on Tableau.
excel_file_name = 'cleaned_aviation_data.xlsx'
df2.to_excel(excel_file_name,index=False, sheet_name='AviationData_clean')
```

2.0 Findings and Conclusion

Based on the exploratory data analysis of the NTSB Air Accident records, the following key insights were identified:

1. **Accident Phase Dominance:** The Landing phase of flight accounts for the majority of recorded air accidents. This suggests a critical period for pilot focus, aircraft performance, and environmental factors.

2. **Purpose of Flight and Accident Frequency:** Aircraft utilized for personal and instructional purposes exhibit the highest number of recorded accidents. This indicates a potential area for targeted safety improvements, pilot training enhancements, or a closer look at the experience levels of pilots involved in these flight types.

3. **Aircraft Category and Fatality Rates:** Airplanes show a higher fatality rate at 69.5% when compared to helicopters at 63.0%. This difference warrants further investigation into the nature of accidents for each aircraft type that leads to these outcomes (e.g., impact forces, escape opportunities).

4. **Aircraft Damage vs. Fatalities:** Substantial Damage is the most frequently recorded outcome regarding aircraft damage. However, it's crucial to note that more fatalities are associated with aircraft classified as "Destroyed." This highlights that while substantial damage is common, complete destruction of the aircraft is a stronger indicator of a fatal outcome.

5. **Aircraft Model with Lowest Fatalities:** Among the analyzed aircraft models, the A310 Aircraft Model stands out for having the lowest number of recorded fatalities. This could suggest inherent safety features, operational procedures, or a lower accident rate for this specific model, though further context on its prevalence in the dataset would be beneficial.

6. **Aircraft Make with Lowest Fatalities:** Cirrus Design Corp and Fokker aircraft makes are associated with the lowest number of fatalities. Similar to the A310 model, this finding could point to design safety, operational practices, or potentially a lower exposure to high-risk scenarios for these manufacturers. Further investigation into specific models under these makes could be insightful.

3.0 Recommendations

1. **Enhance Safety Protocols and Training for Landing Operations:** Developing and implementing more rigorous safety protocols and advanced training modules specifically for the landing phase of flight for all aircraft types.

   Actionable Steps:

- Review existing landing procedures and identify common failure points.
- Implement enhanced simulator training scenarios emphasizing complex landing conditions (e.g., crosswinds, short runways, engine failures on approach).
- Promote best practices for go-around procedures and decision-making.
- Investigate and address any infrastructure issues at airfields that contribute to landing accidents.

1. **Improve Safety Oversight and Education for Personal and Instructional Flights:** Develop and disseminate tailored safety guidelines, educational programs, and potentially more stringent oversight for aircraft used in personal and instructional capacities.

   Actionable Steps:

- Launch awareness campaigns emphasizing risk management for private pilots.
- Encourage or mandate more frequent recurrent training or check-rides for private pilot licenses.
- Investigate if specific types of instructional maneuvers or training environments contribute disproportionately to accidents.
- Consider the role of flight school oversight and curriculum.

1. **Investigate Discrepancies in Aircraft Type Fatality Rates:** Conduct a deeper comparative analysis into the types of accidents and their causal factors that lead to the differing fatality rates between airplanes (69.5%) and helicopters (63.0%). While both are high, the distinct difference suggests varying mechanisms of injury and survivability. Understanding these differences can lead to aircraft-specific safety improvements.

   Actionable Steps:

- Analyze accident reports for airplanes vs. helicopters focusing on impact dynamics, post-crash fires, structural integrity, and occupant restraint systems.
- Evaluate emergency egress capabilities and survival equipment for each aircraft type.
- Assess if specific operational environments or mission profiles contribute to the higher airplane fatality rate.

1. **Prioritize Research into Crashworthiness for "Destroyed" Aircraft:** Focus engineering and design efforts on improving crashworthiness, particularly to prevent aircraft from being classified as "Destroyed," given the strong correlation with fatalities. Although "Substantial Damage" is more common, the finding that "more fatalities were recorded for Destroyed aircraft" highlights the critical nature of preventing catastrophic structural failure.

   Actionable Steps:

- Research advancements in materials and structural design to enhance energy absorption during impact.
- Promote research into advanced fire suppression systems that activate quickly post-impact.
- Review and update international crashworthiness standards based on recent accident data.

1. **Study Best Practices from Low-Fatality Aircraft Models/Makes:** Conduct case studies and detailed analyses of the A310 Aircraft Model, Cirrus Design Corp, and Fokker aircraft makes to identify design features, operational philosophies, or safety innovations that contribute to their lower fatality rates. These entities have demonstrated relatively better safety outcomes ("A310 Aircraft Model has the lowest number of fatalities," "Cirrus Design Cop and Fokker have the lowest fatalities"). Understanding their success can inform industry-wide improvements.

   Actionable Steps:

- Collaborate with manufacturers to understand their safety design principles and testing methodologies.
- Analyze the specific accident profiles of these aircraft to understand why severe outcomes are less frequent.
- Disseminate findings from these case studies across the aviation industry as best practices. (Crucially, normalize these findings by fleet size and total flight hours to ensure fair comparison).

By acting on these recommendations, a better informed choice on the way forward can be made.

In [1]:
```
pip install nbconvert
```

```
Requirement already satisfied: nbconvert in c:\users\achie\anaconda3\envs\learn-env\l
ib\site-packages (6.0.7)Note: you may need to restart the kernel to use updated packa
ges.
Requirement already satisfied: pandocfilters>=1.4.1 in c:\users\achie\anaconda3\envs
\learn-env\lib\site-packages (from nbconvert) (1.4.2)
Requirement already satisfied: mistune<2,>=0.8.1 in c:\users\achie\anaconda3\envs\lea
rn-env\lib\site-packages (from nbconvert) (0.8.4)
Requirement already satisfied: jinja2>=2.4 in c:\users\achie\anaconda3\envs\learn-env
\lib\site-packages (from nbconvert) (2.11.2)
Requirement already satisfied: traitlets>=4.2 in c:\users\achie\anaconda3\envs\learn-
env\lib\site-packages (from nbconvert) (5.0.5)
Requirement already satisfied: nbformat>=4.4 in c:\users\achie\anaconda3\envs\learn-e
nv\lib\site-packages (from nbconvert) (5.0.8)
Requirement already satisfied: entrypoints>=0.2.2 in c:\users\achie\anaconda3\envs\le
arn-env\lib\site-packages (from nbconvert) (0.3)
Requirement already satisfied: pygments>=2.4.1 in c:\users\achie\anaconda3\envs\learn
-env\lib\site-packages (from nbconvert) (2.7.1)
Requirement already satisfied: bleach in c:\users\achie\anaconda3\envs\learn-env\lib
\site-packages (from nbconvert) (3.2.1)
Requirement already satisfied: jupyter-core in c:\users\achie\anaconda3\envs\learn-en
v\lib\site-packages (from nbconvert) (4.6.3)
Requirement already satisfied: jupyterlab-pygments in c:\users\achie\anaconda3\envs\l
earn-env\lib\site-packages (from nbconvert) (0.1.2)
Requirement already satisfied: testpath in c:\users\achie\anaconda3\envs\learn-env\li
b\site-packages (from nbconvert) (0.4.4)
Requirement already satisfied: nbclient<0.6.0,>=0.5.0 in c:\users\achie\anaconda3\env
s\learn-env\lib\site-packages (from nbconvert) (0.5.1)
Requirement already satisfied: defusedxml in c:\users\achie\anaconda3\envs\learn-env
\lib\site-packages (from nbconvert) (0.6.0)
Requirement already satisfied: MarkupSafe>=0.23 in c:\users\achie\anaconda3\envs\lear
n-env\lib\site-packages (from jinja2>=2.4->nbconvert) (1.1.1)
Requirement already satisfied: ipython-genutils in c:\users\achie\anaconda3\envs\lear
n-env\lib\site-packages (from traitlets>=4.2->nbconvert) (0.2.0)
Requirement already satisfied: jsonschema!=2.5.0,>=2.4 in c:\users\achie\anaconda3\en
vs\learn-env\lib\site-packages (from nbformat>=4.4->nbconvert) (3.2.0)
Requirement already satisfied: webencodings in c:\users\achie\anaconda3\envs\learn-en
v\lib\site-packages (from bleach->nbconvert) (0.5.1)
Requirement already satisfied: packaging in c:\users\achie\anaconda3\envs\learn-env\l
ib\site-packages (from bleach->nbconvert) (20.4)
Requirement already satisfied: six>=1.9.0 in c:\users\achie\anaconda3\envs\learn-env
\lib\site-packages (from bleach->nbconvert) (1.15.0)
Requirement already satisfied: pywin32>=1.0; sys_platform == "win32" in c:\users\achi
e\anaconda3\envs\learn-env\lib\site-packages (from jupyter-core->nbconvert) (227)
Requirement already satisfied: nest-asyncio in c:\users\achie\anaconda3\envs\learn-en
v\lib\site-packages (from nbclient<0.6.0,>=0.5.0->nbconvert) (1.4.1)
Requirement already satisfied: async-generator in c:\users\achie\anaconda3\envs\learn
-env\lib\site-packages (from nbclient<0.6.0,>=0.5.0->nbconvert) (1.10)
Requirement already satisfied: jupyter-client>=6.1.5 in c:\users\achie\anaconda3\envs
\learn-env\lib\site-packages (from nbclient<0.6.0,>=0.5.0->nbconvert) (6.1.7)
Requirement already satisfied: attrs>=17.4.0 in c:\users\achie\anaconda3\envs\learn-e
nv\lib\site-packages (from jsonschema!=2.5.0,>=2.4->nbformat>=4.4->nbconvert) (20.2.
0)
Requirement already satisfied: setuptools in c:\users\achie\anaconda3\envs\learn-env
```

```
\lib\site-packages (from jsonschema!=2.5.0,>=2.4->nbformat>=4.4->nbconvert) (50.3.0.p
ost20201103)
Requirement already satisfied: pyrsistent>=0.14.0 in c:\users\achie\anaconda3\envs\le
arn-env\lib\site-packages (from jsonschema!=2.5.0,>=2.4->nbformat>=4.4->nbconvert)
(0.17.3)
Requirement already satisfied: pyparsing>=2.0.2 in c:\users\achie\anaconda3\envs\lear
n-env\lib\site-packages (from packaging->bleach->nbconvert) (2.4.7)
Requirement already satisfied: python-dateutil>=2.1 in c:\users\achie\anaconda3\envs
\learn-env\lib\site-packages (from jupyter-client>=6.1.5->nbclient<0.6.0,>=0.5.0->nbc
onvert) (2.8.1)
Requirement already satisfied: pyzmq>=13 in c:\users\achie\anaconda3\envs\learn-env\l
ib\site-packages (from jupyter-client>=6.1.5->nbclient<0.6.0,>=0.5.0->nbconvert) (19.
0.2)
Requirement already satisfied: tornado>=4.1 in c:\users\achie\anaconda3\envs\learn-en
v\lib\site-packages (from jupyter-client>=6.1.5->nbclient<0.6.0,>=0.5.0->nbconvert)
(6.0.4)
```

In [ ]: