1 Aviation Risk Analysis

The objective of this project is to analyse aviation accident data from 1962 to 2023 to identify the safest aircraft model. As the company wants to venture in purchasing and operating airplanes for commercial and private enterprises, this will help to decide which aircraft to purchase.

This analysis leverages data from National Transportation and Safety Board to;

1. Help decide which aircraft to purchase.

Key Questions

1. Which aircraft models have the lowest accidents and injuries?
2. How can this analysis help strategic business desicions?

Libraries used for data analysis and visualization

1. Numpy
2. Pandas
3. Matplotlib
4. Seaborn.

In [70]:
```
#import necessary libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

1.1 Loading the dataset

In [71]:
```
#loding the dataset
data = pd.read_csv('AviationData.csv',encoding='latin1',low_memory=False)

#to display all columns in the dataset
pd.set_option('display.max_columns',40)

#show first 5 rows to understand the dataset
data.head()
```

Out[71]:

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

1.2 Initial Data Understanding and Exploration

In [72]:
```
#checking the dataset information
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 88889 entries, 0 to 88888
Data columns (total 31 columns):
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   Event.Id                88889 non-null  object 
 1   Investigation.Type      88889 non-null  object 
 2   Accident.Number         88889 non-null  object 
 3   Event.Date              88889 non-null  object 
 4   Location                88837 non-null  object 
 5   Country                 88663 non-null  object 
 6   Latitude                34382 non-null  object 
 7   Longitude               34373 non-null  object 
 8   Airport.Code            50132 non-null  object 
 9   Airport.Name            52704 non-null  object 
 10  Injury.Severity         87889 non-null  object 
 11  Aircraft.damage         85695 non-null  object 
 12  Aircraft.Category       32287 non-null  object 
 13  Registration.Number     87507 non-null  object 
 14  Make                    88826 non-null  object 
 15  Model                   88797 non-null  object 
 16  Amateur.Built           88787 non-null  object 
 17  Number.of.Engines       82805 non-null  float64
 18  Engine.Type             81793 non-null  object 
 19  FAR.Description         32023 non-null  object 
 20  Schedule                12582 non-null  object 
 21  Purpose.of.flight       82697 non-null  object 
 22  Air.carrier             16648 non-null  object 
 23  Total.Fatal.Injuries    77488 non-null  float64
 24  Total.Serious.Injuries  76379 non-null  float64
 25  Total.Minor.Injuries    76956 non-null  float64
 26  Total.Uninjured         82977 non-null  float64
 27  Weather.Condition       84397 non-null  object 
 28  Broad.phase.of.flight   61724 non-null  object 
 29  Report.Status           82505 non-null  object 
 30  Publication.Date        75118 non-null  object 
dtypes: float64(5), object(26)
memory usage: 21.0+ MB
```

In [73]:
```
#checking the concise info 2
data.info(verbose= False)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 88889 entries, 0 to 88888
Columns: 31 entries, Event.Id to Publication.Date
dtypes: float64(5), object(26)
memory usage: 21.0+ MB
```

In [74]:
```
#cheking data shape
data.shape
```

Out[74]: (88889, 31)

In [75]:
```
#cheking data types
data.dtypes
```

Out[75]:
```
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 [76]:
```
#checking columns
data.columns
```

Out[76]:
```
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')
```

1.3 Descriptive Statistics

In [77]:
```
#checking summary statistics
data.describe()
```

Out[77]:

	Number.of.Engines	Total.Fatal.Injuries	Total.Serious.Injuries	Total.Minor.Injuries	To
count	82805.000000	77488.000000	76379.000000	76956.000000	
mean	1.146585	0.647855	0.279881	0.357061	
std	0.446510	5.485960	1.544084	2.235625	
min	0.000000	0.000000	0.000000	0.000000	
25%	1.000000	0.000000	0.000000	0.000000	
50%	1.000000	0.000000	0.000000	0.000000	
75%	1.000000	0.000000	0.000000	0.000000	
max	8.000000	349.000000	161.000000	380.000000	

In [78]:
```
#Cheking categorical columns
data.describe(include='object')
```

Out[78]:

	Event.Id	Investigation.Type	Accident.Number	Event.Date	Location	Co
count	88889	88889	88889	88889	88837	
unique	87951	2	88863	14782	27758	
top	20001212X19172	Accident	CEN22LA149	1984-06-30	ANCHORAGE, AK	
freq	3	85015	2	25	434	

1.4 Data Preparation

In [79]:
```
#Subsetting columns to only use whats necessary
data = data[['Event.Id', 'Investigation.Type', 'Accident.Number', 'Event.D
       '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.Descrip
#        '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'
           ]]
```

In [80]:
```
#checking shape again
data.shape
```

Out[80]: (88889, 15)

Observe we have dropped the columns I do not need

1.5 Data Cleaning

In [81]:
```
#create a dataframe copy for data cleaning
data1 = data.copy(deep=True)
```

1.5.1 Changing Data types

While checking on the data types above, I noticed the Event.Date column object is an object. It supposed to be a date time column

In [82]:
```
#using date time format
data1['Event.Date'] = pd.to_datetime(data1['Event.Date'], errors='coerce')
```

In [83]:
```
#confirm
data1.dtypes
```

Out[83]:
```
Event.Id                          object
Investigation.Type                object
Accident.Number                   object
Event.Date                datetime64[ns]
Location                          object
Country                           object
Injury.Severity                   object
Aircraft.damage                   object
Aircraft.Category                 object
Make                              object
Model                             object
Total.Fatal.Injuries             float64
Total.Serious.Injuries           float64
Total.Minor.Injuries             float64
Total.Uninjured                  float64
dtype: object
```

1.5.2 Checking for duplicates

In [84]:
```
data1.duplicated(keep=False).sum()
```

Out[84]:
```
6
```

In [85]:
```
#inspecting the duplicates
duplicates = data1[data1.duplicated(keep=False)]
print(duplicates)
```

```
            Event.Id Investigation.Type Accident.Number Event.Date  \
87548  20220323104818           Accident      CEN22LA149 2022-03-18
87549  20220323104818           Accident      CEN22LA149 2022-03-18
88386  20220822105776           Accident      ERA22LA379 2022-08-20
88387  20220822105776           Accident      ERA22LA379 2022-08-20
88527  20220921105978           Incident      DCA22WA204 2022-09-14
88528  20220921105978           Incident      DCA22WA204 2022-09-14

           Location        Country Injury.Severity Aircraft.damage  \
87548  Grapevine, TX  United States       Non-Fatal     Substantial
87549  Grapevine, TX  United States       Non-Fatal     Substantial
88386   Bealeton, VA  United States           Minor     Substantial
88387   Bealeton, VA  United States           Minor     Substantial
88527       Mumbai,           India             NaN             NaN
88528       Mumbai,           India             NaN             NaN

      Aircraft.Category    Make  Model  Total.Fatal.Injuries  \
87548          Airplane  CESSNA   208B                   0.0
87549          Airplane  CESSNA   208B                   0.0
88386          Airplane  BOEING  A75N1                   0.0
88387          Airplane  BOEING  A75N1                   0.0
88527          Airplane  BOEING    787                   0.0
88528          Airplane  BOEING    787                   0.0

       Total.Serious.Injuries  Total.Minor.Injuries  Total.Uninjured
87548                     0.0                   0.0              2.0
87549                     0.0                   0.0              2.0
88386                     2.0                   0.0              2.0
88387                     2.0                   0.0              2.0
88527                     0.0                   0.0              0.0
88528                     0.0                   0.0              0.0
```

In [86]:
```
#dropping duplicates
data1 = data1.drop_duplicates()
```

In [87]:
```
#veryfing
data1.duplicated(keep=False).sum()
```

Out[87]:
```
0
```

1.5.3 Standardizing Columns

In [88]:
```
data1.columns
```

Out[88]:
```
Index(['Event.Id', 'Investigation.Type', 'Accident.Number', 'Event.Date',
       'Location', 'Country', 'Injury.Severity', 'Aircraft.damage',
       'Aircraft.Category', 'Make', 'Model', 'Total.Fatal.Injuries',
       'Total.Serious.Injuries', 'Total.Minor.Injuries', 'Total.Uninjured'],
      dtype='object')
```

In [89]:
```
#Standardize columns
data1.columns = data1.columns.str.strip().str.title().str.replace('.','_')
data1.columns
```

Out[89]:
```
Index(['Event_Id', 'Investigation_Type', 'Accident_Number', 'Event_Date',
       'Location', 'Country', 'Injury_Severity', 'Aircraft_Damage',
       'Aircraft_Category', 'Make', 'Model', 'Total_Fatal_Injuries',
       'Total_Serious_Injuries', 'Total_Minor_Injuries', 'Total_Uninjured'],
      dtype='object')
```

1.5.4 Checking for missing values

In [90]:
```
data1.isna().sum()
```

Out[90]:
```
Event_Id                   0
Investigation_Type         0
Accident_Number            0
Event_Date                 0
Location                  52
Country                  226
Injury_Severity          999
Aircraft_Damage         3193
Aircraft_Category      56602
Make                      63
Model                     92
Total_Fatal_Injuries   11401
Total_Serious_Injuries 12510
Total_Minor_Injuries   11933
Total_Uninjured         5912
dtype: int64
```

In [91]:
```
# adds the number of missing values in each column
missing_values = data1.isnull().sum()

# Getting percantage of missing values
total_rows = len(data1)
percent_missing_values = (missing_values/total_rows)*100

#using a dataframe
missing_values_df = pd.DataFrame({"missing_values": missing_values, "Perce

# arranges from highest percantage of missing values to lowest
missing_values_df = missing_values_df.sort_values(by='Percentage',ascendin
missing_values_df
```

Out[91]:

	missing_values	Percentage
Aircraft_Category	56602	63.679320
Total_Serious_Injuries	12510	14.074207
Total_Minor_Injuries	11933	13.425061
Total_Fatal_Injuries	11401	12.826542
Total_Uninjured	5912	6.651216
Aircraft_Damage	3193	3.592242
Injury_Severity	999	1.123912
Country	226	0.254258
Model	92	0.103503
Make	63	0.070877
Location	52	0.058502
Event_Id	0	0.000000
Investigation_Type	0	0.000000
Accident_Number	0	0.000000
Event_Date	0	0.000000

1.5.5 Dropping colums with more than 50% missing values

1. This improves the quality and usability of the dataset for analysis
2. This also helps us focus on columns directly related to the key questions

In [92]:
```
#dropping Aircraft_Category column because it has over 50% missing values
data1 = data1.drop(columns=['Aircraft_Category'])
```

In [93]:
```
1 data1.columns
```

Out[93]:
```
Index(['Event_Id', 'Investigation_Type', 'Accident_Number', 'Event_Date',
       'Location', 'Country', 'Injury_Severity', 'Aircraft_Damage', 'Make',
       'Model', 'Total_Fatal_Injuries', 'Total_Serious_Injuries',
       'Total_Minor_Injuries', 'Total_Uninjured'],
      dtype='object')
```

1.5.6 Input missing values

In [94]:
```
1 # Inputing missing categorical columns with unknown
2 categorical_columns= ['Injury_Severity', 'Aircraft_Damage', 'Make','Model'
3 for column in categorical_columns:
4     data1[column] = data1[column].fillna('unknown')
5
6 # Inputing missing values with the median for each column
7 numeric_columns = ['Total_Fatal_Injuries', 'Total_Serious_Injuries', 'Tota
8 for column in numeric_columns:
9     data1[column] = data1[column].fillna(data1[column].median())
```

1.5.7 Check for missing vaues

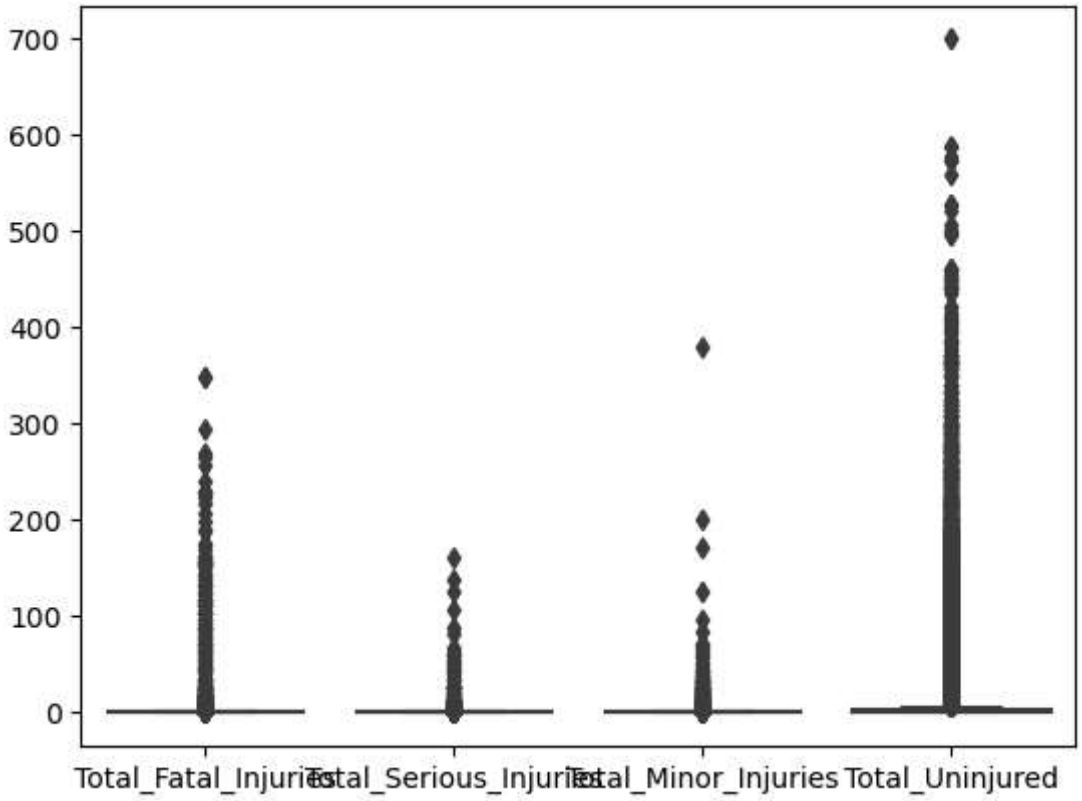
In [95]:
```
1 #pass in a list of columns we want to check
2 check_columns = ['Injury_Severity', 'Aircraft_Damage', 'Make','Model','Tot
3 #Check the sum missing values
4 data1[check_columns].isna().sum()
```

Out[95]:
```
Injury_Severity           0
Aircraft_Damage           0
Make                      0
Model                     0
Total_Fatal_Injuries      0
Total_Serious_Injuries    0
Total_Minor_Injuries      0
Total_Uninjured           0
dtype: int64
```

1.5.8 Checking for outliers

In [96]:
```
sns.boxplot(data1[['Total_Fatal_Injuries', 'Total_Serious_Injuries', 'Tota
```

Out[96]: <Axes: >



Outliers are not that bad

1.5.9 Convert Data types

This is to ensure all the numerical columns defined are of numeric types

In [97]:
```
numerical_columns = ['Total_Fatal_Injuries', 'Total_Serious_Injuries', 'To
data1[numerical_columns] = data1[numerical_columns].apply(pd.to_numeric)
```

In [98]:
```
# Convert make and model column to categorical type
categorical_columns2 = ['Make', 'Model']
for column in categorical_columns2:
    data1[column] = data1[column].astype('category')
```

1.5.10 Checking unique values

In [99]:
```
1 print(data1['Model'].unique())
```

```
['108-3', 'PA24-180', '172M', '112', '501', ..., 'XLT-RG', 'MH-60R', 'ROTORWA
Y EXEC 162-F', 'KITFOX S5', 'M-8 EAGLE']
Length: 12318
Categories (12318, object): ['&GCBC', '(EX) RV-6', '(MODIFIED)', '(SOLOY CONV
ERSION)', ..., 'none', 'sportstar', 'unk', 'unknown']
```

In [100]:
```
1 print(data1['Make'].unique())
```

```
['Stinson', 'Piper', 'Cessna', 'Rockwell', 'Mcdonnell Douglas', ..., 'PHANTO
M', 'GREG HOBBS', 'JAMES R DERNOVSEK', 'ORLICAN S R O', 'ROYSE RALPH L']
Length: 8237
Categories (8237, object): ['107.5 Flying Corporation', '1200', '177MF LLC',
'1977 Colfer-chan', ..., 'Zwart', 'de Havilland', 'drone', 'unknown']
```

In [101]:
```
1 data1['Make'] = data1['Make'].str.lower().str.title()
```

In [102]:
```
1 print(data1['Aircraft_Damage'].unique())
```

```
['Destroyed' 'Substantial' 'Minor' 'unknown' 'Unknown']
```

In [103]:
```
1 # Standardize Text Values- Convert all values to lowercase to ensure consi
2 data1['Aircraft_Damage'] = data1['Aircraft_Damage'].str.lower()
```

In [104]:
```
1 data1['Aircraft_Damage']
```

Out[104]:
```
0          destroyed
1          destroyed
2          destroyed
3          destroyed
4          destroyed
            ...
88884        unknown
88885        unknown
88886    substantial
88887        unknown
88888        unknown
Name: Aircraft_Damage, Length: 88886, dtype: object
```

In [105]:
```
1 data1['Injury_Severity'].unique()
```

Out[105]:
```
array(['Fatal(2)', 'Fatal(4)', 'Fatal(3)', 'Fatal(1)', 'Non-Fatal',
       'Incident', 'Fatal(8)', 'Fatal(78)', 'Fatal(7)', 'Fatal(6)',
       'Fatal(5)', 'Fatal(153)', 'Fatal(12)', 'Fatal(14)', 'Fatal(23)',
       'Fatal(10)', 'Fatal(11)', 'Fatal(9)', 'Fatal(17)', 'Fatal(13)',
       'Fatal(29)', 'Fatal(70)', 'Unavailable', 'Fatal(135)', 'Fatal(31)',
       'Fatal(256)', 'Fatal(25)', 'Fatal(82)', 'Fatal(156)', 'Fatal(28)',
       'Fatal(18)', 'Fatal(43)', 'Fatal(15)', 'Fatal(270)', 'Fatal(144)',
       'Fatal(174)', 'Fatal(111)', 'Fatal(131)', 'Fatal(20)', 'Fatal(73)',
       'Fatal(27)', 'Fatal(34)', 'Fatal(87)', 'Fatal(30)', 'Fatal(16)',
       'Fatal(47)', 'Fatal(56)', 'Fatal(37)', 'Fatal(132)', 'Fatal(68)',
       'Fatal(54)', 'Fatal(52)', 'Fatal(65)', 'Fatal(72)', 'Fatal(160)',
       'Fatal(189)', 'Fatal(123)', 'Fatal(33)', 'Fatal(110)',
       'Fatal(230)', 'Fatal(97)', 'Fatal(349)', 'Fatal(125)', 'Fatal(35)',
       'Fatal(228)', 'Fatal(75)', 'Fatal(104)', 'Fatal(229)', 'Fatal(80)',
       'Fatal(217)', 'Fatal(169)', 'Fatal(88)', 'Fatal(19)', 'Fatal(60)',
       'Fatal(113)', 'Fatal(143)', 'Fatal(83)', 'Fatal(24)', 'Fatal(44)',
       'Fatal(64)', 'Fatal(92)', 'Fatal(118)', 'Fatal(265)', 'Fatal(26)',
       'Fatal(138)', 'Fatal(206)', 'Fatal(71)', 'Fatal(21)', 'Fatal(46)',
       'Fatal(102)', 'Fatal(115)', 'Fatal(141)', 'Fatal(55)',
       'Fatal(121)', 'Fatal(45)', 'Fatal(145)', 'Fatal(117)',
       'Fatal(107)', 'Fatal(124)', 'Fatal(49)', 'Fatal(154)', 'Fatal(96)',
       'Fatal(114)', 'Fatal(199)', 'Fatal(89)', 'Fatal(57)', 'Fatal',
       'unknown', 'Minor', 'Serious'], dtype=object)
```

1.5.11 Filtering Relevant Data

FIltering for 'Accident' to ensure the analysis aligns with the objective of focusing on high-risk events.

In [106]:
```
1 data1['Investigation_Type'].unique()
```

Out[106]:
```
array(['Accident', 'Incident'], dtype=object)
```

In [107]:
```
1 # Focus only on accidents
2 data1 = data1[data1['Investigation_Type'] == 'Accident']
```

1.5.12 Feature engineering

1.5.12.1 Creating a column of Fatalities

1. The categories has value counts indicating a fatal incident with the number of fatalities in parentheses.
2. I will split and strip to separate them
3. Also create a new column

In [108]:
```
# Define a function to clean the data
def clean_injury_severity(value):
    if 'Fatal' in value:
        # Extract the number in parentheses if it exists
        if '(' in value:
            return int(value.split('(')[1].strip(')'))
        else:
            return 1  # Default for 'Fatal' without a number
    elif value in ['Non-Fatal', 'Minor', 'Serious']:
        return 0  # Non-fatal categories treated as 0 fatalities
    else:
        return np.nan  # Handle 'unknown' and 'Unavailable' as missing val

# Apply the cleaning function
data1['Fatalities'] = data1['Injury_Severity'].apply(clean_injury_severity

# View the cleaned data
print(data1[['Injury_Severity', 'Fatalities']].head())
```

```
  Injury_Severity  Fatalities
0        Fatal(2)         2.0
1        Fatal(4)         4.0
2        Fatal(3)         3.0
3        Fatal(2)         2.0
4        Fatal(1)         1.0
```

1.5.12.2 Creating a column of total injuries

In [109]:
```
 # Create a total injuries column
data1['Total_Injuries'] = (data1['Total_Fatal_Injuries'] + data1['Total_Se
data1[['Total_Injuries']]
```

Out[109]:

	Total_Injuries
0	2.0
1	4.0
2	3.0
3	2.0
4	3.0
...	...
88884	1.0
88885	0.0
88886	0.0
88887	0.0
88888	1.0

85013 rows × 1 columns

1.5.12.3 Creating a column of damage level

To transform categorical data ('Aircraft_Damage') into numerical values ('Damage_Level')

In [110]:
```
# To get the level of damage represented by a number
damage_mapping = {'destroyed': 3, 'substantial': 2, 'minor': 1,'unknown':
data1['Damage_Level'] = data1['Aircraft_Damage'].map(damage_mapping)
data1.head()
```

Out[110]:

	Event_Id	Investigation_Type	Accident_Number	Event_Date	Location	Cou
0	20001218X45444	Accident	SEA87LA080	1948-10-24	MOOSE CREEK, ID	
1	20001218X45447	Accident	LAX94LA336	1962-07-19	BRIDGEPORT, CA	
2	20061025X01555	Accident	NYC07LA005	1974-08-30	Saltville, VA	
3	20001218X45448	Accident	LAX96LA321	1977-06-19	EUREKA, CA	
4	20041105X01764	Accident	CHI79FA064	1979-08-02	Canton, OH	

In [111]:
```
1 data1
```

Out[111]:

	Event_Id	Investigation_Type	Accident_Number	Event_Date	Location	C
0	20001218X45444	Accident	SEA87LA080	1948-10-24	MOOSE CREEK, ID	
1	20001218X45447	Accident	LAX94LA336	1962-07-19	BRIDGEPORT, CA	
2	20061025X01555	Accident	NYC07LA005	1974-08-30	Saltville, VA	
3	20001218X45448	Accident	LAX96LA321	1977-06-19	EUREKA, CA	
4	20041105X01764	Accident	CHI79FA064	1979-08-02	Canton, OH	
...	...	...	...	...	...	
88884	20221227106491	Accident	ERA23LA093	2022-12-26	Annapolis, MD	
88885	20221227106494	Accident	ERA23LA095	2022-12-26	Hampton, NH	
88886	20221227106497	Accident	WPR23LA075	2022-12-26	Payson, AZ	
88887	20221227106498	Accident	WPR23LA076	2022-12-26	Morgan, UT	
88888	20221230106513	Accident	ERA23LA097	2022-12-29	Athens, GA	

85013 rows × 17 columns

1.5.12.4 Creating a column of Make and Model

In [112]:
```
1 # Combining Make and Model columns to get full name of aircraft
2 data1['Aircraft'] = data1['Make'].astype(str) + ' ' + data1['Model'].astyp
```

In [113]:
```
1 data1.columns
```

Out[113]:
```
Index(['Event_Id', 'Investigation_Type', 'Accident_Number', 'Event_Date',
       'Location', 'Country', 'Injury_Severity', 'Aircraft_Damage', 'Make',
       'Model', 'Total_Fatal_Injuries', 'Total_Serious_Injuries',
       'Total_Minor_Injuries', 'Total_Uninjured', 'Fatalities',
       'Total_Injuries', 'Damage_Level', 'Aircraft'],
      dtype='object')
```

In [114]:
```
1 data1[['Aircraft']]
```

Out[114]:

	Aircraft
0	Stinson 108-3
1	Piper PA24-180
2	Cessna 172M
3	Rockwell 112
4	Cessna 501
...	...
88884	Piper PA-28-151
88885	Bellanca 7ECA
88886	American Champion Aircraft 8GCBC
88887	Cessna 210N
88888	Piper PA-24-260

85013 rows × 1 columns

1.5.12.5 Creating a column of total metrics of Damages, Fatalties and Injuries

In [115]:
```
1 # This adds all to get the objective
2 data1['Total_Metric'] = data1['Fatalities'] + data1['Damage_Level'] + data
```

In [116]:
```
1 data1.head()
```

Out[116]:

	Event_Id	Investigation_Type	Accident_Number	Event_Date	Location	Cou
0	20001218X45444	Accident	SEA87LA080	1948-10-24	MOOSE CREEK, ID	
1	20001218X45447	Accident	LAX94LA336	1962-07-19	BRIDGEPORT, CA	
2	20061025X01555	Accident	NYC07LA005	1974-08-30	Saltville, VA	
3	20001218X45448	Accident	LAX96LA321	1977-06-19	EUREKA, CA	
4	20041105X01764	Accident	CHI79FA064	1979-08-02	Canton, OH	

1.5.13 Validate before saving the clean copy

In [117]:
```
# Inspect the cleaned dataset
print(data1.info())
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 85013 entries, 0 to 88888
Data columns (total 19 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Event_Id                85013 non-null  object
 1   Investigation_Type      85013 non-null  object
 2   Accident_Number         85013 non-null  object
 3   Event_Date              85013 non-null  datetime64[ns]
 4   Location                84973 non-null  object
 5   Country                 84807 non-null  object
 6   Injury_Severity         85013 non-null  object
 7   Aircraft_Damage         85013 non-null  object
 8   Make                    85013 non-null  object
 9   Model                   85013 non-null  category
 10  Total_Fatal_Injuries    85013 non-null  float64
 11  Total_Serious_Injuries  85013 non-null  float64
 12  Total_Minor_Injuries    85013 non-null  float64
 13  Total_Uninjured         85013 non-null  float64
 14  Fatalities              84590 non-null  float64
 15  Total_Injuries          85013 non-null  float64
 16  Damage_Level            85013 non-null  int64
 17  Aircraft                85013 non-null  object
 18  Total_Metric            84590 non-null  float64
dtypes: category(1), datetime64[ns](1), float64(7), int64(1), object(9)
memory usage: 12.8+ MB
None
```

In [118]:
```
data1.head()
```

Out[118]:

	Event_Id	Investigation_Type	Accident_Number	Event_Date	Location	Cou
0	20001218X45444	Accident	SEA87LA080	1948-10-24	MOOSE CREEK, ID	
1	20001218X45447	Accident	LAX94LA336	1962-07-19	BRIDGEPORT, CA	
2	20061025X01555	Accident	NYC07LA005	1974-08-30	Saltville, VA	
3	20001218X45448	Accident	LAX96LA321	1977-06-19	EUREKA, CA	
4	20041105X01764	Accident	CHI79FA064	1979-08-02	Canton, OH	

1.5.14 Saving the clean copy

In [119]:
```
#save the new dataframe in csv format
data1.to_csv('CLEAN_AviationData.csv', index=False)
```

1.5.15 Loading the clean Dataset

In [120]:
```
avidata = pd.read_csv('CLEAN_AviationData.csv',low_memory=False)
print(avidata.shape)
avidata.head()
```

```
(85013, 19)
```

Out[120]:

	Event_Id	Investigation_Type	Accident_Number	Event_Date	Location	Cou
0	20001218X45444	Accident	SEA87LA080	1948-10-24	MOOSE CREEK, ID	
1	20001218X45447	Accident	LAX94LA336	1962-07-19	BRIDGEPORT, CA	
2	20061025X01555	Accident	NYC07LA005	1974-08-30	Saltville, VA	
3	20001218X45448	Accident	LAX96LA321	1977-06-19	EUREKA, CA	
4	20041105X01764	Accident	CHI79FA064	1979-08-02	Canton, OH	

1.6 Exploratory Data Analysis

To analyze and investigate data set to summarize their main characteristics in line with our objective

In [121]:
```
1 avidata.head()
```

Out[121]:

	Event_Id	Investigation_Type	Accident_Number	Event_Date	Location	Cou
0	20001218X45444	Accident	SEA87LA080	1948-10-24	MOOSE CREEK, ID	
1	20001218X45447	Accident	LAX94LA336	1962-07-19	BRIDGEPORT, CA	
2	20061025X01555	Accident	NYC07LA005	1974-08-30	Saltville, VA	
3	20001218X45448	Accident	LAX96LA321	1977-06-19	EUREKA, CA	
4	20041105X01764	Accident	CHI79FA064	1979-08-02	Canton, OH	

In [122]:
```
1 # Group data by aircraft make and model
2 grouped_data = avidata.groupby(['Aircraft']).agg({ 'Damage_Level': 'sum','
3 grouped_data
```

Out[122]:

	Aircraft	Damage_Level	Fatalities	Total_Injuries	Total_Metric
0	107.5 Flying Corporation One Design DR 107	3	1.0	1.0	5.0
1	1200 G103	2	1.0	1.0	4.0
2	177Mf Llc PITTS MODEL 12	2	1.0	2.0	5.0
3	1977 Colfer-Chan STEEN SKYBOLT	2	1.0	1.0	4.0
4	1St Ftr Gp FOCKE-WULF 190	3	1.0	1.0	5.0
...	...	...	...	...	...
17785	Zubair S Khan RAVEN	2	1.0	1.0	4.0
17786	Zuber Thomas P ZUBER SUPER DRIFTER	2	1.0	0.0	3.0
17787	Zukowski EAA BIPLANE	2	1.0	0.0	3.0
17788	Zwart KIT FOX VIXEN	2	1.0	0.0	3.0
17789	Zwicker Murray R GLASTAR	2	1.0	0.0	3.0

17790 rows × 5 columns

1.6.1 Checking top 10 Aircraft models with highest injuries

In [123]:
```
# Sort by Highest total injuries
highest_injuries_df = grouped_data.sort_values(by='Total_Injuries', ascend

# Top 10 aircraft with highest injuries
highest_injuries_df
```

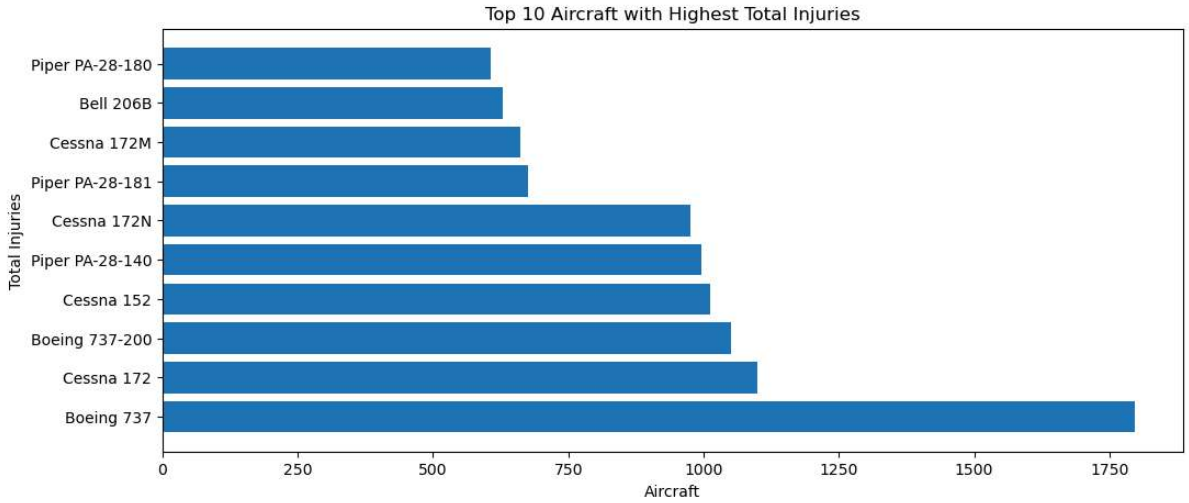
Out[123]:

	Aircraft	Damage_Level	Fatalities	Total_Injuries	Total_Metric
3063	Boeing 737	184	100.0	1796.0	2035.0
4301	Cessna 172	3649	1818.0	1100.0	6547.0
3080	Boeing 737-200	61	925.0	1050.0	2028.0
4277	Cessna 152	5022	2468.0	1012.0	8495.0
12967	Piper PA-28-140	2069	1052.0	996.0	4114.0
4352	Cessna 172N	2531	1342.0	975.0	4842.0
12984	Piper PA-28-181	1187	752.0	676.0	2612.0
4350	Cessna 172M	1721	876.0	662.0	3256.0
2465	Bell 206B	1157	620.0	629.0	2402.0
12977	Piper PA-28-180	1283	681.0	607.0	2571.0

Ploting for visualisation

In [124]:

```
plt.figure(figsize=(12, 5))  # Set the figure size for better readability

# Create a horizontal bar chart
plt.barh(highest_injuries_df['Aircraft'], highest_injuries_df['Total_Injur

# Add titles and labels
plt.xlabel('Aircraft')
plt.ylabel('Total Injuries')
plt.title('Top 10 Aircraft with Highest Total Injuries')

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



Observation, Boeing 737 has the highest total injuries

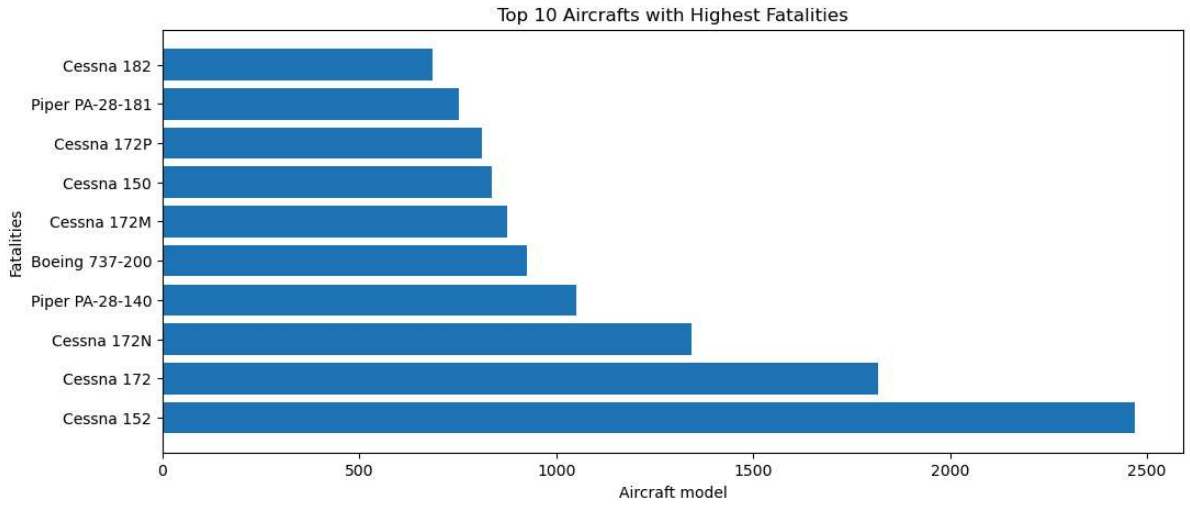
1.6.2 Checking top 10 Aircrafts with highest fatalities

In [125]:
```
# Sort by highest fatalities
Highest_fatalities_df = grouped_data.sort_values(by='Fatalities', ascendin

#  Top 10 aircraft with highest fatalities
Highest_fatalities_df
```

Out[125]:

	Aircraft	Damage_Level	Fatalities	Total_Injuries	Total_Metric
4277	Cessna 152	5022	2468.0	1012.0	8495.0
4301	Cessna 172	3649	1818.0	1100.0	6547.0
4352	Cessna 172N	2531	1342.0	975.0	4842.0
12967	Piper PA-28-140	2069	1052.0	996.0	4114.0
3080	Boeing 737-200	61	925.0	1050.0	2028.0
4350	Cessna 172M	1721	876.0	662.0	3256.0
4250	Cessna 150	1755	836.0	447.0	3035.0
4355	Cessna 172P	1502	812.0	565.0	2874.0
12984	Piper PA-28-181	1187	752.0	676.0	2612.0
4409	Cessna 182	1408	687.0	530.0	2620.0

In [126]:
```
plt.figure(figsize=(12,5))

plt.barh(Highest_fatalities_df['Aircraft'], Highest_fatalities_df['Fatalit

# Add titles and labels
plt.title('Top 10 Aircrafts with Highest Fatalities')
plt.xlabel('Aircraft model')
plt.ylabel('Fatalities')
plt.show()
```



Observation, Cessna 152 has the highest Fatalities

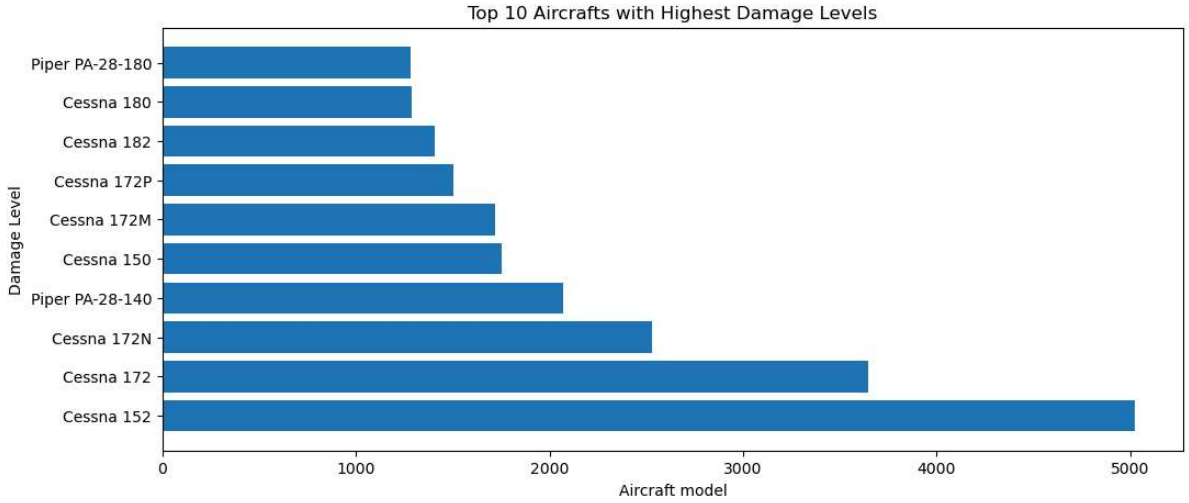
1.6.3 Checking top 10 Aircrafts with highest damage levels

In [127]:
```
# Sort by highest damage levels
Highest_damage_df = grouped_data.sort_values(by='Damage_Level', ascending=

#  Top 10 aircraft with highest damage levels
Highest_damage_df
```

Out[127]:

	Aircraft	Damage_Level	Fatalities	Total_Injuries	Total_Metric
4277	Cessna 152	5022	2468.0	1012.0	8495.0
4301	Cessna 172	3649	1818.0	1100.0	6547.0
4352	Cessna 172N	2531	1342.0	975.0	4842.0
12967	Piper PA-28-140	2069	1052.0	996.0	4114.0
4250	Cessna 150	1755	836.0	447.0	3035.0
4350	Cessna 172M	1721	876.0	662.0	3256.0
4355	Cessna 172P	1502	812.0	565.0	2874.0
4409	Cessna 182	1408	687.0	530.0	2620.0
4385	Cessna 180	1285	666.0	247.0	2198.0
12977	Piper PA-28-180	1283	681.0	607.0	2571.0

In [128]:
```
plt.figure(figsize=(12,5))

plt.barh(Highest_damage_df['Aircraft'], Highest_damage_df['Damage_Level'])

# Add titles and labels
plt.title('Top 10 Aircrafts with Highest Damage Levels')
plt.xlabel('Aircraft model')
plt.ylabel('Damage Level')
plt.show()
```



Observation, Cessna 152 has the highest damage levels

1.6.4 Checking which model has the HIGHEST total metrics

1.6.4.1 Top 10 Aircrafts with highest metrics

In [129]:
```
# Sort by the Total_Metric to find the highest
grouped_data['Total_Metric'] = grouped_data['Damage_Level'] + grouped_data
Hmetrics = grouped_data.sort_values(by='Total_Metric', ascending=False).he
Hmetrics
```

Out[129]:

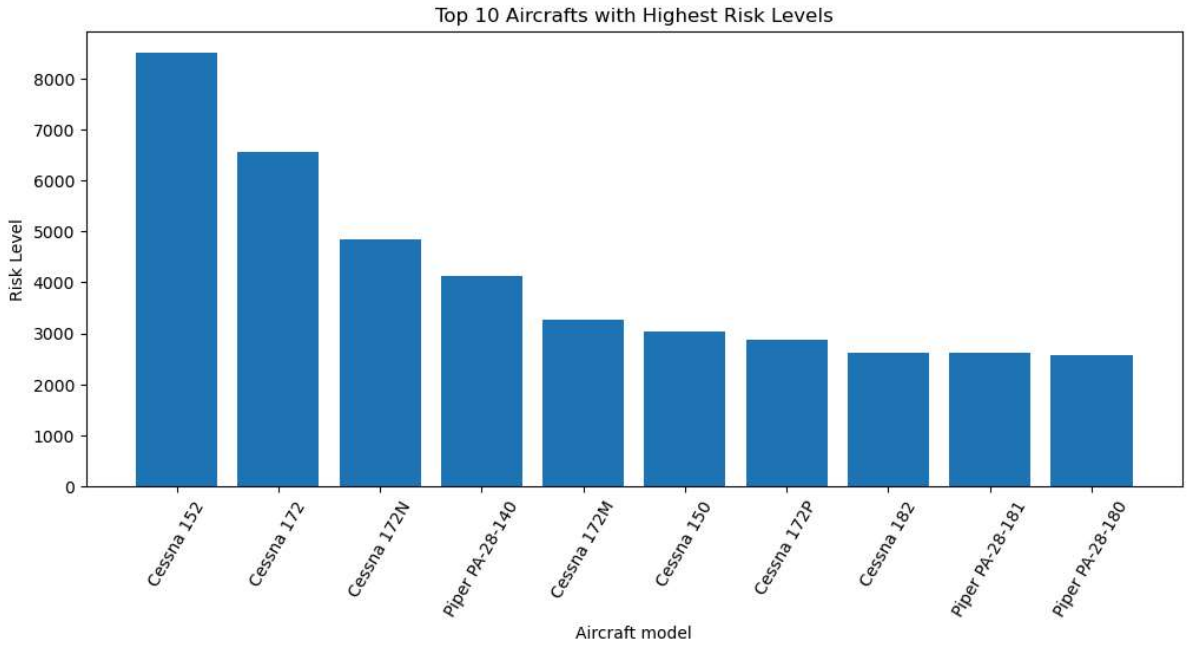
	Aircraft	Damage_Level	Fatalities	Total_Injuries	Total_Metric
4277	Cessna 152	5022	2468.0	1012.0	8502.0
4301	Cessna 172	3649	1818.0	1100.0	6567.0
4352	Cessna 172N	2531	1342.0	975.0	4848.0
12967	Piper PA-28-140	2069	1052.0	996.0	4117.0
4350	Cessna 172M	1721	876.0	662.0	3259.0
4250	Cessna 150	1755	836.0	447.0	3038.0
4355	Cessna 172P	1502	812.0	565.0	2879.0
4409	Cessna 182	1408	687.0	530.0	2625.0
12984	Piper PA-28-181	1187	752.0	676.0	2615.0
12977	Piper PA-28-180	1283	681.0	607.0	2571.0

In [130]:
```
# Sort by highest metrics
Highest_metric_df = grouped_data.sort_values(by='Total_Metric', ascending=

#  Top 10 aircraft with highest metric
Highest_metric_df
```

Out[130]:

	Aircraft	Damage_Level	Fatalities	Total_Injuries	Total_Metric
4277	Cessna 152	5022	2468.0	1012.0	8502.0
4301	Cessna 172	3649	1818.0	1100.0	6567.0
4352	Cessna 172N	2531	1342.0	975.0	4848.0
12967	Piper PA-28-140	2069	1052.0	996.0	4117.0
4350	Cessna 172M	1721	876.0	662.0	3259.0
4250	Cessna 150	1755	836.0	447.0	3038.0
4355	Cessna 172P	1502	812.0	565.0	2879.0
4409	Cessna 182	1408	687.0	530.0	2625.0
12984	Piper PA-28-181	1187	752.0	676.0	2615.0
12977	Piper PA-28-180	1283	681.0	607.0	2571.0

In [131]:
```
plt.figure(figsize=(12,5))

plt.bar(Highest_metric_df['Aircraft'], Highest_metric_df['Total_Metric'])

# Add titles and labels
plt.title('Top 10 Aircrafts with Highest Risk Levels')
plt.xlabel('Aircraft model')
plt.ylabel('Risk Level')
plt.xticks(rotation=60)# Rotate the x-axis labels to avoid overlap
plt.show()
```



1.6.5 Checking which model has the LEAST total metrics

1.6.5.1 Top 10 Aircrafts with least metrics

In [132]:
```
# Sort by the Total_Metric to find the least
grouped_data['Total_Metric'] = grouped_data['Damage_Level'] + grouped_data
Lmetrics = grouped_data.sort_values(by='Total_Metric', ascending=True).hea
Lmetrics
```

Out[132]:

	Aircraft	Damage_Level	Fatalities	Total_Injuries	Total_Metric
4387	Cessna 180 - UNDESIGNAT	0	0.0	0.0	0.0
8408	Gulfstream Aerospace Lp Gulfstream G150	0	0.0	0.0	0.0
3416	Bombardier BD 700 1A10	0	0.0	0.0	0.0
8726	Hawker Siddeley 800XP	0	0.0	0.0	0.0
16218	Syracuse V F/Syracuse C D KITFOX	0	0.0	0.0	0.0
3241	Boeing 777 - 200	0	0.0	0.0	0.0
7581	Found Acft Canada Inc FBA-2C1	0	0.0	0.0	0.0
16851	Uvify IFO	0	0.0	0.0	0.0
1172	American Legend Aircraft Compa AL18	0	0.0	0.0	0.0
3113	Boeing 737-4B6	0	0.0	0.0	0.0
836	Airbus A330-243	0	0.0	0.0	0.0
4099	Canadair CL604	0	0.0	0.0	0.0
3205	Boeing 767 - 200	0	0.0	0.0	0.0
6543	Drone Viper Pro	0	0.0	0.0	0.0
16794	Unknown A330	0	0.0	0.0	0.0

- These are the top 10 aircrafts to choose

1.6.6 Total Injuries, Fatalities, and Damage Level over time

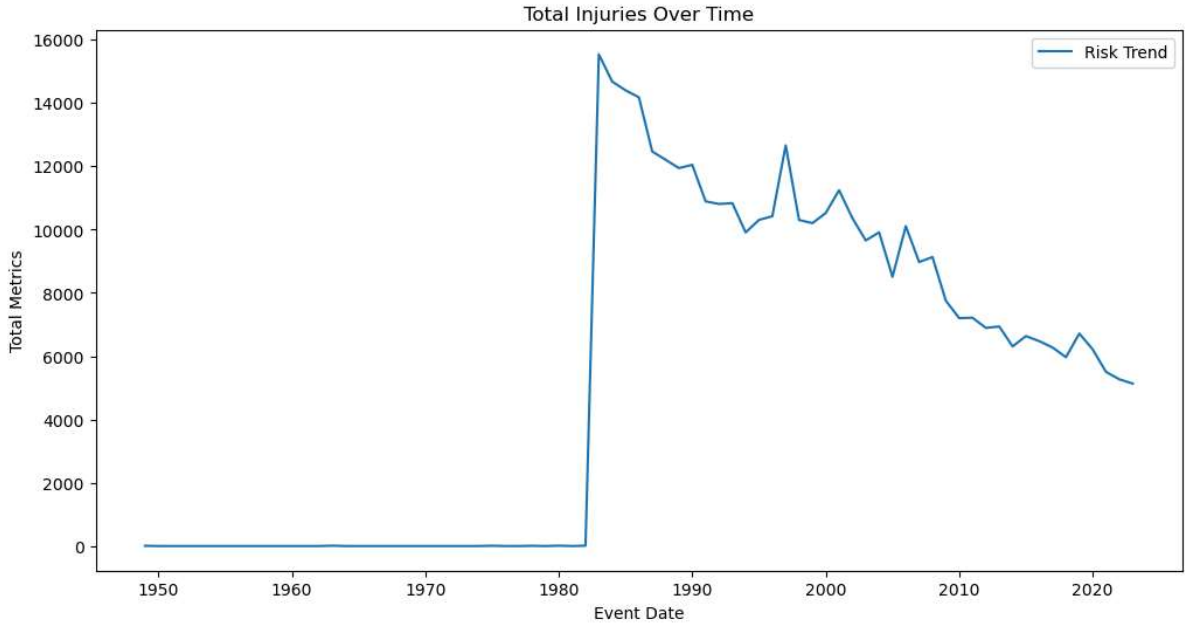
In [133]:
```
# convert the column Event_Date to a datetime type:
avidata['Event_Date'] = pd.to_datetime(avidata['Event_Date'], errors='coer

# Set the 'Event_Date' column as the index
avidata.set_index('Event_Date', inplace=True, drop=False)

# resample the data at a yearly frequency
yearly_data = avidata.resample('Y').agg({
    'Total_Injuries': 'sum',
    'Fatalities': 'sum',
    'Damage_Level': 'sum',
    'Total_Metric': 'sum'
}).reset_index()

yearly_data.head()
```

Out[133]:

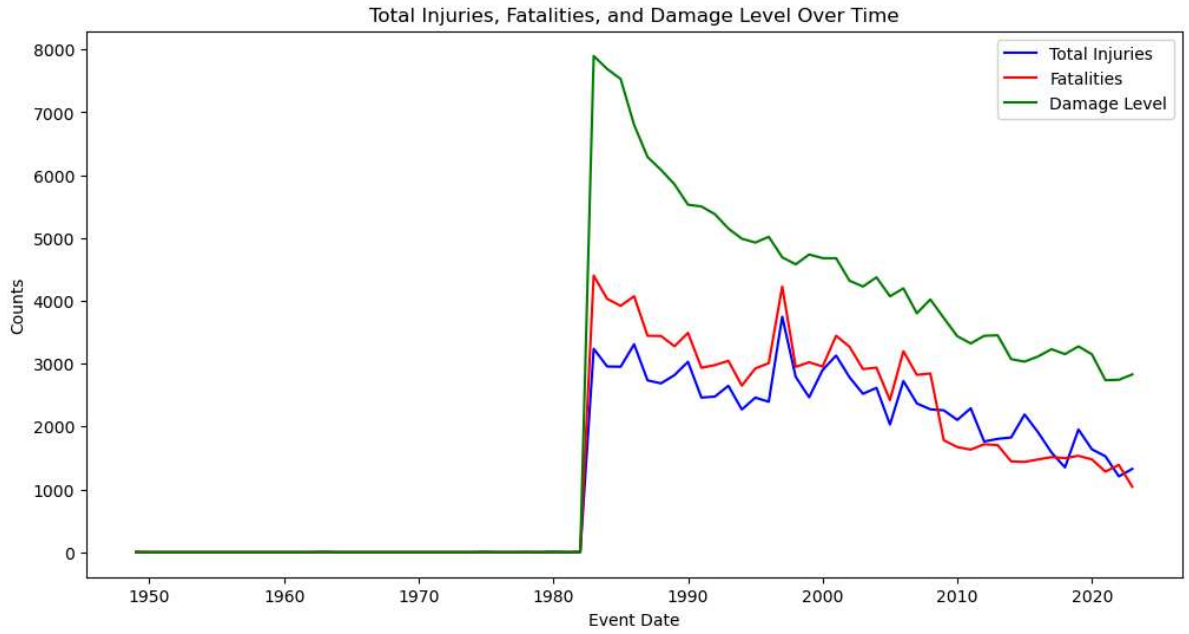
	Event_Date	Total_Injuries	Fatalities	Damage_Level	Total_Metric
0	1948-12-31	2.0	2.0	3	7.0
1	1949-12-31	0.0	0.0	0	0.0
2	1950-12-31	0.0	0.0	0	0.0
3	1951-12-31	0.0	0.0	0	0.0
4	1952-12-31	0.0	0.0	0	0.0

In [134]:

```
# Plotting Total Injuries over time
plt.figure(figsize=(12, 6))
plt.plot(yearly_data['Event_Date'], yearly_data['Total_Metric'], label='Ri

plt.title('Total Injuries Over Time')
plt.xlabel('Event Date')
plt.ylabel('Total Metrics')
plt.legend()
plt.show()
```



In [135]:
```
# Plotting Total Injuries, Fatalities, and Damage Level over time
plt.figure(figsize=(12, 6))
plt.plot(yearly_data['Event_Date'], yearly_data['Total_Injuries'], label='
plt.plot(yearly_data['Event_Date'], yearly_data['Fatalities'], label='Fata
plt.plot(yearly_data['Event_Date'], yearly_data['Damage_Level'], label='Da

# Add titles and labels
plt.title('Total Injuries, Fatalities, and Damage Level Over Time')
plt.xlabel('Event Date')
plt.ylabel('Counts')
plt.legend()
plt.show()

```



- Buy new aircrafts since Total Injuries, Fatalities, and Damage levels have been steadily declining by each year