Overview

NEW HORIZONS 2025.

Following recent trends in the macroeconomic environment and volatility within the nation and also globally, we seek to restrategize our business by looking into sectors we could venture into as we try to diversify our existing portfolio to enhance our balance sheet position and market dominance.
Having said that the board agreed on venturing in the aviation sector by purchasing and operating airplanes for commercial and private enterprises.

Business Understanding

Objectives

Below are the key stakeholder and key business questions;
1. To determine low risk aircrafts.
2. To find out the locations that are less prone to aircraft accidents

Data Understanding and Analysis

Source of data

The Data was obtained from the National Transportation Safety Board that includes aviation accident data from 1962 to 2023 about civil aviation accidents and selected incidents in the United States and international waters.

Description of data

To understand the dataset further, it is important to drill down further and get information on the dataset, and this was achieved by running the below codes:

In [13]:
```
# Importing Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [27]:
```
# Importing the datasets and checking on the information it contains.
# US States dataset
States = pd.read_csv('USState_Codes.csv')
States.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 62 entries, 0 to 61
Data columns (total 2 columns):
 #   Column        Non-Null Count  Dtype 
---  ------        --------------  ----- 
 0   US_State      62 non-null     object
 1   Abbreviation  62 non-null     object
dtypes: object(2)
memory usage: 1.1+ KB
```

The USState_Codes dataset has 2 columns and 62 rows with data type as an object.

In [26]:
```
# AviationData dataset
aviation_data = pd.read_csv('AviationData.csv', encoding='latin-1',low_memory=False)
aviation_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            50249 non-null  object 
 9   Airport.Name            52790 non-null  object 
 10  Injury.Severity         87889 non-null  object 
 11  Aircraft.damage         85695 non-null  object 
 12  Aircraft.Category       32287 non-null  object 
 13  Registration.Number     87572 non-null  object 
 14  Make                    88826 non-null  object 
 15  Model                   88797 non-null  object 
 16  Amateur.Built           88787 non-null  object 
 17  Number.of.Engines       82805 non-null  float64
 18  Engine.Type             81812 non-null  object 
 19  FAR.Description         32023 non-null  object 
 20  Schedule                12582 non-null  object 
 21  Purpose.of.flight       82697 non-null  object 
 22  Air.carrier             16648 non-null  object 
 23  Total.Fatal.Injuries    77488 non-null  float64
 24  Total.Serious.Injuries  76379 non-null  float64
 25  Total.Minor.Injuries    76956 non-null  float64
 26  Total.Uninjured         82977 non-null  float64
 27  Weather.Condition       84397 non-null  object 
 28  Broad.phase.of.flight   61724 non-null  object 
 29  Report.Status           82508 non-null  object 
 30  Publication.Date        75118 non-null  object 
dtypes: float64(5), object(26)
memory usage: 21.0+ MB
```

The AviationData dataset has 31 columns and 88889 rows. The columns have various data types where 5 are floats and 26 are objects.

In [29]:
```
# Statistical description of the Aviation dataset including the objects
aviation_data.describe(include='object')
```

Out[29]:

	Event.Id	Investigation.Type	Accident.Number	Event.Date	Location	Country	Latitude	Longitude	Airport.Code	Airport.Na
count	88889	88889	88889	88889	88837	88663	34382	34373	50249	52
unique	87951	2	88863	14782	27758	219	25589	27154	10375	24
top	20001214X45071	Accident	CEN23MA034	2000-07-08	ANCHORAGE, AK	United States	332739N	0112457W	NONE	Pri
freq	3	85015	2	25	434	82248	19	24	1488	

4 rows × 26 columns

In [ ]:
```
From the above preview, various columns will need to be transformed to the correct formats: Event.Date, Publication
```

In [30]:
```
# Statistical description of the Aviation dataset minus the objects
aviation_data.describe()
```

Out[30]:

	Number.of.Engines	Total.Fatal.Injuries	Total.Serious.Injuries	Total.Minor.Injuries	Total.Uninjured
count	82805.000000	77488.000000	76379.000000	76956.000000	82977.000000
mean	1.146585	0.647855	0.279881	0.357061	5.325440
std	0.446510	5.485960	1.544084	2.235625	27.913634
min	0.000000	0.000000	0.000000	0.000000	0.000000
25%	1.000000	0.000000	0.000000	0.000000	0.000000
50%	1.000000	0.000000	0.000000	0.000000	1.000000
75%	1.000000	0.000000	0.000000	0.000000	2.000000
max	8.000000	349.000000	161.000000	380.000000	699.000000

The above descriptions give us more insights into our dataset and the various variables that can be correlated.

In [81]:
```
# To check on correlation in the columns

aviation_data.corr()
```

Out[81]:

	Number.of.Engines	Total.Fatal.Injuries	Total.Serious.Injuries	Total.Minor.Injuries	Total.Uninjured
Number.of.Engines	1.000000	0.064947	0.040530	0.042068	0.343908
Total.Fatal.Injuries	0.064947	1.000000	0.064569	0.010095	-0.028091
Total.Serious.Injuries	0.040530	0.064569	1.000000	0.310394	0.048144
Total.Minor.Injuries	0.042068	0.010095	0.310394	1.000000	0.071928
Total.Uninjured	0.343908	-0.028091	0.048144	0.071928	1.000000

The above analysis shows us correlation values on the numeric columns.
From the above data, values greater than zero will indicate a positive relationship whereas the values less than zero indicate a negative relationship. This has been demonstrated in a visual on the analysis part.

In [37]:
```
#checking for duplicates
aviation_data.duplicated()
```

Out[37]:
```
0        False
1        False
2        False
3        False
4        False
         ...
88884    False
88885    False
88886    False
88887    False
88888    False
Length: 88889, dtype: bool
```

In [31]:
```
# Checking for missing values.
aviation_data.isnull().sum()
```

Out[31]:
```
Event.Id                      0
Investigation.Type            0
Accident.Number               0
Event.Date                    0
Location                     52
Country                     226
Latitude                  54507
Longitude                 54516
Airport.Code              38640
Airport.Name              36099
Injury.Severity            1000
Aircraft.damage            3194
Aircraft.Category         56602
Registration.Number        1317
Make                         63
Model                        92
Amateur.Built               102
Number.of.Engines          6084
Engine.Type                7077
FAR.Description           56866
Schedule                  76307
Purpose.of.flight          6192
Air.carrier               72241
Total.Fatal.Injuries      11401
Total.Serious.Injuries    12510
Total.Minor.Injuries      11933
Total.Uninjured            5912
Weather.Condition          4492
Broad.phase.of.flight     27165
Report.Status              6381
Publication.Date          13771
dtype: int64
```

As indicated above, various columns have missing values in different rows, and cleanup has been done in the data preparation stage.

DATA PREPARATION

In data preparation our data story and structure will begin to change as we seek to get to our goals and objectives.

In [44]:
```
#Data Cleaning
## Removing duplicated values.

aviation_data = aviation_data.drop_duplicates()
aviation_data.duplicated()
```

Out[44]:
```
0        False
1        False
2        False
3        False
4        False
         ...
88884    False
88885    False
88886    False
88887    False
88888    False
Length: 88889, dtype: bool
```

From the above we see the duplicates have been removed.

In [45]:
```
## Replacing null values in the floats with 0 as they are numeric in nature
numeric_columns = ['Total.Fatal.Injuries', 'Total.Serious.Injuries', 'Total.Minor.Injuries', 'Total.Uninjured']
aviation_data[numeric_columns] = aviation_data[numeric_columns].fillna(0)
aviation_data.isnull().sum()
```

Out[45]:
```
Event.Id                      0
Investigation.Type            0
Accident.Number               0
Event.Date                    0
Location                     52
Country                     226
Latitude                  54507
Longitude                 54516
Airport.Code              38640
Airport.Name              36099
Injury.Severity            1000
Aircraft.damage            3194
Aircraft.Category         56602
Registration.Number        1317
Make                         63
Model                        92
Amateur.Built               102
Number.of.Engines          6084
Engine.Type                7077
FAR.Description           56866
Schedule                  76307
Purpose.of.flight          6192
Air.carrier               72241
Total.Fatal.Injuries          0
Total.Serious.Injuries        0
Total.Minor.Injuries          0
Total.Uninjured               0
Weather.Condition          4492
Broad.phase.of.flight     27165
Report.Status              6381
Publication.Date          13771
dtype: int64
```

From the above code, we have replaced null values in the numeric columns with '0'

In [46]:
```
# Dropping null values in the Latitude and Longitudes columns for precise analysis and judgement
aviation_data = aviation_data.dropna(subset=['Longitude', 'Latitude'])
```

In [47]:
```
aviation_data.isnull().sum()
```

Out[47]:
```
Event.Id                    0
Investigation.Type          0
Accident.Number             0
Event.Date                  0
Location                    5
Country                     1
Latitude                    0
Longitude                   0
Airport.Code            11957
Airport.Name            11733
Injury.Severity           233
Aircraft.damage           944
Aircraft.Category        8414
Registration.Number       384
Make                       22
Model                      28
Amateur.Built              31
Number.of.Engines        2228
Engine.Type              4298
FAR.Description          8515
Schedule                31410
Purpose.of.flight        3301
Air.carrier             21652
Total.Fatal.Injuries        0
Total.Serious.Injuries      0
Total.Minor.Injuries        0
Total.Uninjured             0
Weather.Condition        1736
Broad.phase.of.flight   22882
Report.Status            3994
Publication.Date          605
dtype: int64
```

In [48]:
```
# Converting Event.Date and Publication.Date columns to the correct date and time formats.

aviation_data['Event.Date'] = pd.to_datetime(aviation_data['Event.Date'], errors='coerce')
aviation_data['Publication.Date'] = pd.to_datetime(aviation_data['Publication.Date'], errors='coerce')
aviation_data.head(10)
```

Out[48]:

	Event.Id	Investigation.Type	Accident.Number	Event.Date	Location	Country	Latitude	Longitude	Airport.Code	Airport.Na
2	20061025X01555	Accident	NYC07LA005	1974-08-30	Saltville, VA	United States	36.922223	-81.878056	NaN	N
5	20170710X52551	Accident	NYC79AA106	1979-09-17	BOSTON, MA	United States	42.445277	-70.758333	NaN	N
593	20080417X00504	Accident	MIA08CA076	1982-03-16	MOBILE, AL	United States	30.757778	-88.355555	MOB	MOB REGION
3654	20051208X01953	Accident	SEA83LA209	1983-01-08	Goldendale, WA	United States	46.041111	-120.849722	NaN	N
6202	20020904X01525	Accident	SEA83FA208	1983-09-09	Kalispell, MT	United States	48.12	-113.8875	NaN	N
22096	20001213X27446	Accident	LAX89LA068	1988-12-23	Midway Islands, PO	United States	38.54	-173.24	NONE	N
24567	20021022X05356	Accident	CHI90LA280	1989-12-01	ENGADINE, MI	United States	46.154444	-85.663611	NaN	N
26826	20030411X00484	Accident	ANC91GAMS1	1990-10-11	Deadhorse, AK	United States	70.333333	-150.933333	NaN	N
31353	20170710X10920	Accident	FTW92FA224	1992-09-05	Alpine, TX	United States	30.383611	-103.783334	NaN	N
38740	20011127X02295	Accident	NYC96FA192	1995-11-28	Marlinton, WV	United States	38.335	-80.28	48I	Brax Cou

10 rows × 31 columns

As demonstrated in the above 10 rows, the data type for Event.Date and Publication.Date columns have been converted to the correct date and time formats.

Data Analysis

In [69]:
```
# To determine low risk aircrafts.

aviation_risk_analysis = aviation_data.groupby('Aircraft.Category').agg(
    total_accidents=('Aircraft.Category', 'count'), #category has been grouped by count as it is not numerical.
    total_fatal_injuries=('Total.Fatal.Injuries', 'sum'),#Fatal injuries has been grouped by sum as it is numerical
    total_serious_injuries=('Total.Serious.Injuries', 'sum'), #Serious injuries has been grouped by sum as it is nu
    total_minor_injuries=('Total.Minor.Injuries', 'sum') #Minor injuries has been grouped by sum as it is numerical
).reset_index()

aviation_risk_analysis['risk_score'] = (
    aviation_risk_analysis['total_fatal_injuries'] * 3 +
    aviation_risk_analysis['total_serious_injuries'] * 2 +
    aviation_risk_analysis['total_minor_injuries']
) / aviation_risk_analysis['total_accidents']

aviation_low_risk_categories = aviation_risk_analysis.sort_values(by='risk_score').fillna(0)

aviation_low_risk_categories.head(100)
```

Out[69]:

	Aircraft.Category	total_accidents	total_fatal_injuries	total_serious_injuries	total_minor_injuries	risk_score
10	UNK	1	0.0	0.0	0.0	0.000000
7	Powered-Lift	3	0.0	1.0	0.0	0.666667
2	Blimp	4	0.0	0.0	3.0	0.750000
9	ULTR	1	0.0	0.0	1.0	1.000000
3	Glider	452	88.0	94.0	106.0	1.234513
4	Gyrocraft	158	42.0	52.0	26.0	1.620253
0	Airplane	22080	8051.0	5722.0	4131.0	1.799275
11	Ultralight	25	5.0	11.0	8.0	1.800000
5	Helicopter	2760	1197.0	833.0	668.0	2.146739
6	Powered Parachute	91	15.0	40.0	73.0	2.175824
14	Weight-Shift	161	67.0	58.0	50.0	2.279503
1	Balloon	200	36.0	160.0	173.0	3.005000
13	WSFT	9	10.0	1.0	2.0	3.777778
8	Rocket	1	1.0	0.0	1.0	4.000000
12	Unknown	7	10.0	7.0	1.0	6.428571

From the above risk assesment it answers one of the objectives set towards identifying a suitable aircraft category to venture in.
Based on the above data, Gliders, Gyrocraft, Airplane, Helicopter are to avoided when considering the aircraft categories to purchase since they recorded a high number of accidents and injuries hence the high risk score.
For starters, the company can venture into Blimp, Powered-lift and UNK aircraft categories as they recorded a low risk score.

In [71]:
```
# To find out the locations that are less prone to aircraft accidents.

aviation_location_analysis = aviation_data.groupby('Location').agg(
    total_accidents=('Location', 'count') #location has been grouped by count as it is not numerical.
).reset_index()

aviation_safe_locations = aviation_location_analysis[aviation_location_analysis['total_accidents'] > 0].sort_values

aviation_safe_locations.head(10)
```

Out[71]:

	Location	total_accidents
14503	helena, MT	1
12712	Symrna, TN	1
7321	Laughlin, NV	1
7322	Laupahoehoe, HI	1
7323	Laural, MT	1
7324	Laurel Bloomery, TN	1
7325	Laurel Hill, FL	1
7326	Laurel, DE	1
7319	Laton, CA	1
12711	Sylvester, GA	1

As we drill down to focus on States in the United States where we can venture in, it is important to identify locations that are less prone to accidents as it will be a safe gamble to start by flying to those locations as we grow to other states.
From the above analysis, below are the States that recorded rather low accidents:

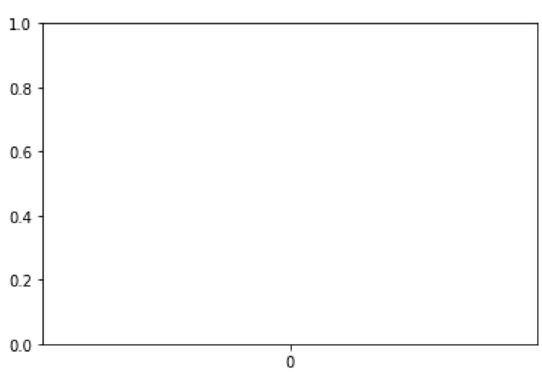
1) California.
2) Delaware.
3) Florida.
4) Georgia.
5) Hawaii.
6) Montana.
7) Nevada.
8) Tennessee.

Visualizations

In [73]:
```
# A boxplot to check on outliers.

sns.boxplot(aviation_data=aviation_data)
```

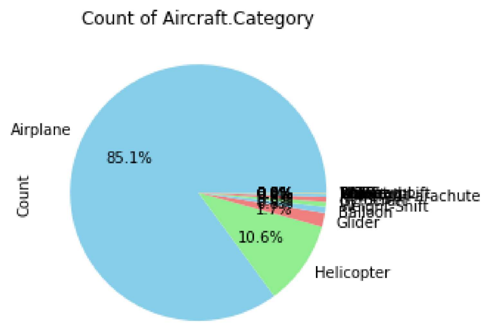
Out[73]: <AxesSubplot:>



We do not have outliers in our aviation_data.

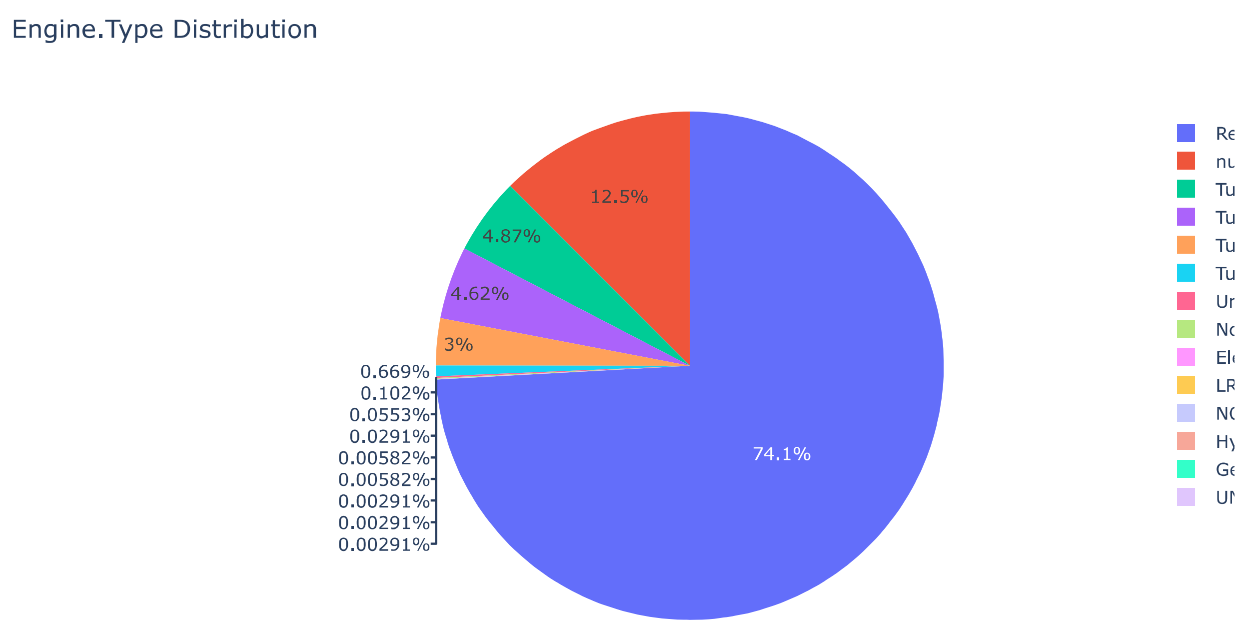
In [92]:
```
# Piechart showing the Aircraft.Category by count

aviation_data['Aircraft.Category'].value_counts().plot(kind='pie', autopct='%1.1f%%', colors=['skyblue', 'lightgree
plt.title('Count of Aircraft.Category')
plt.ylabel('Count')
plt.show()
```



From the above visual, the common type of Aircraft Category is the Airplane that accounts for more than 85% of our data, followed closely by the Helicopter at almost 11%

In [97]:
```
# Pichart showing the distribution by Engine Type
import plotly.express as px
fig = px.pie(aviation_data, names='Engine.Type', title='Engine.Type Distribution')
fig.show()
```

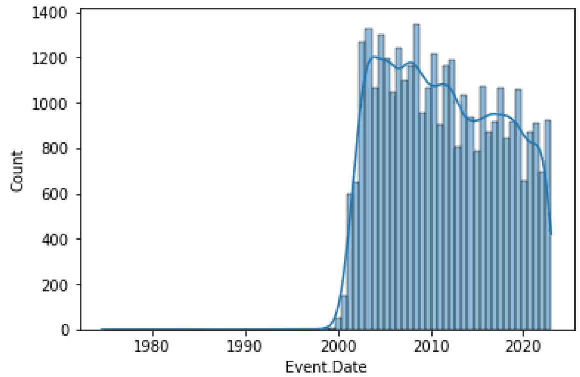


While considering the Engine Type to purchase, it is important to point out to you that majority of the Airplane categories flights are powered by the Reciprocating Engine Type that accounts for 74%.

In [78]:
```
# Histogram to check on Event Dates on when they began.

sns.histplot(aviation_data['Event.Date'], kde=True)
```

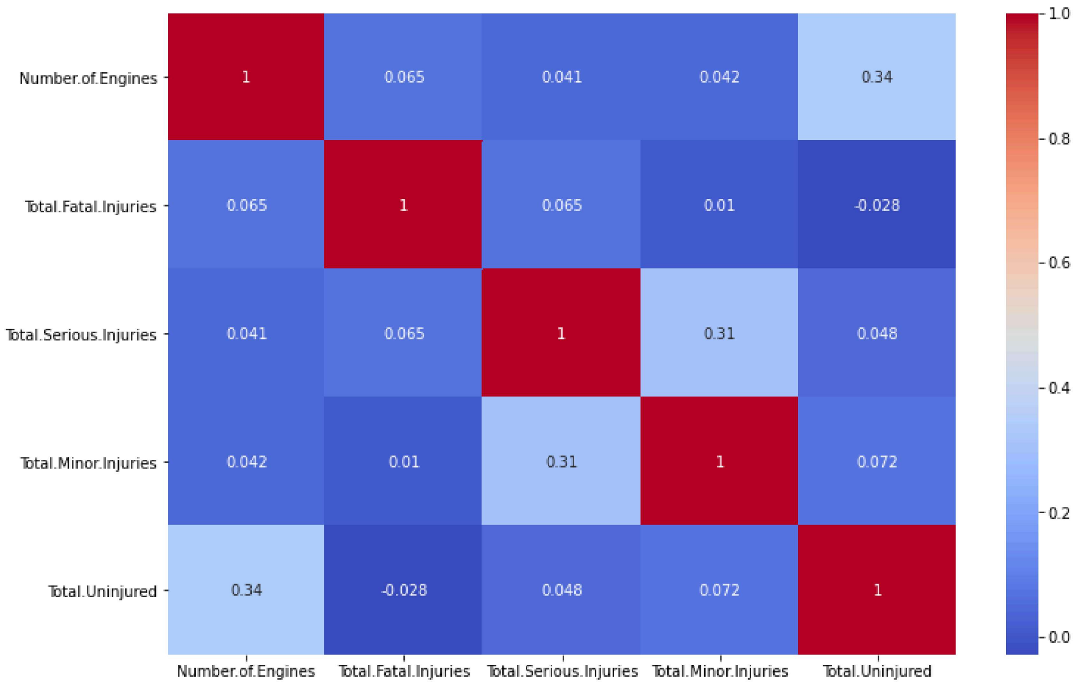
Out[78]: <AxesSubplot:xlabel='Event.Date', ylabel='Count'>



In [80]:
```
# Heatmap to show correlation on Number.of.Engines, Total.Fatal.Injuries,Total.Serious.Injuries,Total.Minor.Injurie:

plt.figure(figsize=(12,8))
sns.heatmap(aviation_data.corr(), annot=True, cmap='coolwarm')
```

Out[80]: <AxesSubplot:>



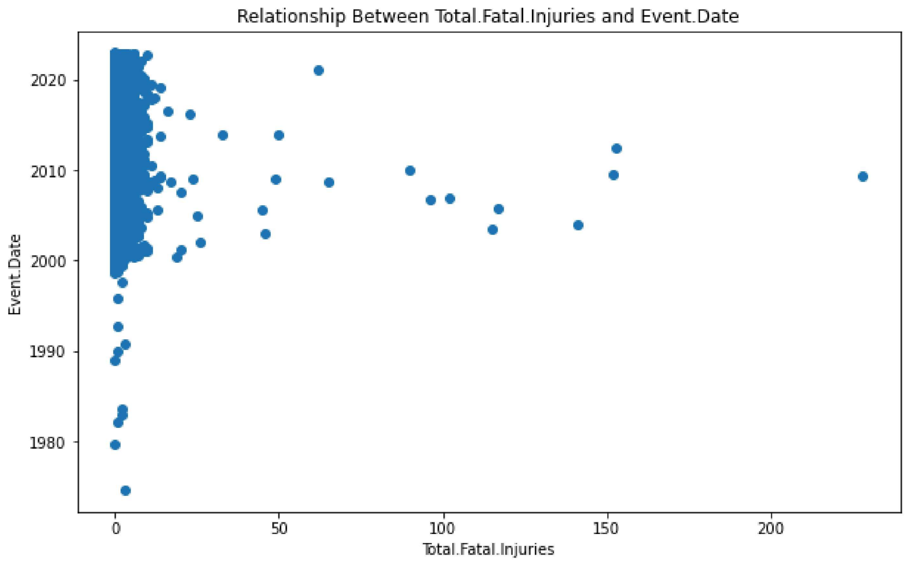
In [86]:
```
# Scatterplot to show the Relationship between Total.Fatal.Injuries and Event.Date

scatter_plot_title = 'Relationship Between Total.Fatal.Injuries and Event.Date'
Total_Fatal_Injuries_label = 'Total.Fatal.Injuries'
Event_Date_label = 'Event.Date'

tackle_figure, ax = plt.subplots(figsize=(10, 6))

# Your code here
ax.scatter(aviation_data['Total.Fatal.Injuries'], aviation_data['Event.Date'])
plt.title('Relationship Between Total.Fatal.Injuries and Event.Date')
plt.xlabel('Total.Fatal.Injuries')
plt.ylabel('Event.Date')
```

Out[86]:
```
Text(0, 0.5, 'Event.Date')
```



Conclusion

Following the above analysis from the aviation_data I am of the view that the organisation can move ahead with the proposed venture into the aviation industry sector.