Final Project Submission

I started with importing the relevant pandas that will be needed for this project.

Project Goal

This project will help look at various aspects to mitigate risk of purchasing and operating aircrafts.

In [1]:

```
#First we Import pandas, numpy and matplotlib
#We will use pandas for data manipulation
import pandas as pd
#We will use numpy for some mathematical operations
import numpy as np
#We will use matplotlib for visualization
import matplotlib.pyplot as plt
%matplotlib inline
#We will use seaborn for visualization
import seaborn as sns
```

Reading the dataset from the CSV file

The dataset that will be loaded will contain aviation accident data from 1962 to 2023 about civil aviation accidents and selected incidents in the United States and international waters

In [2]:

```
#Load the dataset
df = pd.read_csv("AviationData.csv", encoding="ISO-8859-1")
```

```
C:\Users\Nick\AppData\Local\Temp\ipykernel_22904\1175090061.py:2: DtypeWarning: Columns
(6,7,28) have mixed types. Specify dtype option on import or set low_memory=False.
  df = pd.read_csv("AviationData.csv", encoding="ISO-8859-1")
```

Previewing the dataset

In [3]:

```
#setting the default data view. Just to check whether all the columns are visible
pd.set_option("display.max_columns", 500)
```

In [4]:

```
#Let as look at the first 10 rows of the dataset
df.head(10)
```

Out[4]:

	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	36.922223
3	20001218X45448	Accident	LAX96LA321	1977-06-19	EUREKA, CA	United States	NaN
4	20041105X01764	Accident	CHI79FA064	1979-08-02	Canton, OH	United States	NaN

	Event.Id	Investigation.Type	Accident.Number	Event.Date	Location	Country	Latitude
5	20170710X52551	Accident	NYC79AA106	1979-09-17	BOSTON, MA	United States	42.445277
6	20001218X45446	Accident	CHI81LA106	1981-08-01	COTTON, MN	United States	NaN
7	20020909X01562	Accident	SEA82DA022	1982-01-01	PULLMAN, WA	United States	NaN
8	20020909X01561	Accident	NYC82DA015	1982-01-01	EAST HANOVER, NJ	United States	NaN
9	20020909X01560	Accident	MIA82DA029	1982-01-01	JACKSONVILLE, FL	United States	NaN

In [5]:

```
#Now lets look at the last 10 rows
df.tail(10)
```

Out[5]:

	Event.Id	Investigation.Type	Accident.Number	Event.Date	Location	Country	Latitude
88879	20221219106472	Accident	DCA23LA096	2022-12-18	Kahului, HI	United States	NaN
88880	20221219106477	Accident	WPR23LA071	2022-12-18	San Manual, AZ	United States	NaN
88881	20221221106483	Accident	CEN23LA067	2022-12-21	Auburn Hills, MI	United States	NaN
88882	20221222106486	Accident	CEN23LA068	2022-12-21	Reserve, LA	United States	NaN
88883	20221228106502	Accident	GAA23WA046	2022-12-22	Brasnorte,	Brazil	NaN
88884	20221227106491	Accident	ERA23LA093	2022-12-26	Annapolis, MD	United States	NaN
88885	20221227106494	Accident	ERA23LA095	2022-12-26	Hampton, NH	United States	NaN
88886	20221227106497	Accident	WPR23LA075	2022-12-26	Payson, AZ	United States	341525N
88887	20221227106498	Accident	WPR23LA076	2022-12-26	Morgan, UT	United States	NaN
88888	20221230106513	Accident	ERA23LA097	2022-12-29	Athens, GA	United States	NaN

In [6]:

```
#Random sampling
df.sample(10)
```

Out[6]:

	Event.Id	Investigation.Type	Accident.Number	Event.Date	Location	Country	Latitu
68443	20100708X84529	Accident	CEN10CA377	2010-07-03	Jacksonville, AR	United States	345816
75488	20140731X45443	Accident	CEN14LA397	2014-07-25	Questa, NM	United States	036320
83956	20190918X94902	Accident	ERA19FA275	2019-09-17	Stroudsburg, PA	United States	403412
5166	20001214X43550	Accident	SEA83LA145	1983-06-24	FORT HALL, ID	United States	Na
60381	20060405X00394	Accident	MIA06LA074	2006-03-29	DAYTONA BEACH, FL	United States	29.
55943	20031230X02103	Accident	LAX04FA057	2003-12-04	Rosamond, CA	United States	34.8447
68861	20100907X43340	Incident	DCA10WA093	2010-09-02	Taipei, Taiwan (Province of China)	Taiwan	Na
63429	20071012X01584	Accident	SEA07CA269	2007-09-21	BEND, OR	United States	44.1916
74279	20130925X14546	Accident	WPR13CA422	2013-09-21	Seattle, WA	United States	473227
85255	20200825X23153	Accident	WPR20LA301	2020-08-25	Rawlins, WY	United States	41482

Accessing the information in the dataset

This process is to show a summary of all the available columns we have in the dataset

In [7]:

```
df.info()
```

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

Looking through various aspects of the Data

In [8]:

```
df.columns
```

Out[8]:

```
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 [9]:

```
#Checking the number of columns
len(df.columns)
```

Out[9]:

```
31
```

In [10]:

```
#Checking the number of rows
len(df)
```

Out[10]:

```
88889
```

In [11]:

```
#Checking the shape
df.shape
```

Out[11]:

```
(88889, 31)
```

In [12]:

```
#Checking Descriptive statistics for numerical variables.
#I will transpose the Data frame for better readability
df.describe().T
```

Out[12]:

	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
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

In [13]:

```
#Checking the unique values
df.nunique()
```

Out[13]:

```
Event.Id                  87951
Investigation.Type            2
Accident.Number           88863
Event.Date                14782
Location                  27758
Country                     219
Latitude                  25592
Longitude                 27156
Airport.Code              10374
Airport.Name              24870
Injury.Severity             109
Aircraft.damage               4
Aircraft.Category            15
Registration.Number       79104
Make                       8237
Model                     12318
Amateur.Built                 2
Number.of.Engines             7
Engine.Type                  12
FAR.Description              31
Schedule                      3
Purpose.of.flight            26
Air.carrier               13590
Total.Fatal.Injuries        125
Total.Serious.Injuries       50
Total.Minor.Injuries         57
Total.Uninjured             379
Weather.Condition             4
Broad.phase.of.flight        12
Report.Status             17074
Publication.Date           2924
dtype: int64
```

In [14]:

```
#Use unique to see the unique values in the columns.
unique_values = df['Make'].unique()
unique_values
```

Out[14]:

```
array(['Stinson', 'Piper', 'Cessna', ..., 'JAMES R DERNOVSEK',
       'ORLICAN S R O', 'ROYSE RALPH L'], dtype=object)
```

In [15]:

```
unique_values = df['Model'].unique()
unique_values
```

Out[15]:

```
array(['108-3', 'PA24-180', '172M', ..., 'ROTORWAY EXEC 162-F',
       'KITFOX S5', 'M-8 EAGLE'], dtype=object)
```

In [16]:

```
unique_values = df['Investigation.Type'].unique()
unique_values
```

Out[16]:

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

In [18]:

```
unique_values = df['Aircraft.damage'].unique()
unique_values
```

Out[18]:

```
array(['Destroyed', 'Substantial', 'Minor', nan, 'Unknown'], dtype=object)
```

In [19]:

```
unique_values = df['Engine.Type'].unique()
unique_values
```

Out[19]:

```
array(['Reciprocating', nan, 'Turbo Fan', 'Turbo Shaft', 'Unknown',
       'Turbo Prop', 'Turbo Jet', 'Electric', 'Hybrid Rocket',
       'Geared Turbofan', 'LR', 'NONE', 'UNK'], dtype=object)
```

In [20]:

```
#We can filter the data based on the investigation type and look at the accidents
df[df['Investigation.Type'] == 'Accident']
```

Out[20]:

	Event.Id	Investigation.Type	Accident.Number	Event.Date	Location	Country	Latitu
0	20001218X45444	Accident	SEA87LA080	1948-10-24	MOOSE CREEK, ID	United States	N
1	20001218X45447	Accident	LAX94LA336	1962-07-19	BRIDGEPORT, CA	United States	N
2	20061025X01555	Accident	NYC07LA005	1974-08-30	Saltville, VA	United States	36.9222
3	20001218X45448	Accident	LAX96LA321	1977-06-19	EUREKA, CA	United States	N
4	20041105X01764	Accident	CHI79FA064	1979-08-02	Canton, OH	United States	N
...	...	...	...	...	...	...	
88884	20221227106491	Accident	ERA23LA093	2022-12-26	Annapolis, MD	United States	N
88885	20221227106494	Accident	ERA23LA095	2022-12-26	Hampton, NH	United States	N
88886	20221227106497	Accident	WPR23LA075	2022-12-26	Payson, AZ	United States	34152
88887	20221227106498	Accident	WPR23LA076	2022-12-26	Morgan, UT	United States	N
88888	20221230106513	Accident	ERA23LA097	2022-12-29	Athens, GA	United States	N

85015 rows × 31 columns

In [21]:

```
#We can also check on the incidents.
df[df['Investigation.Type'] == 'Incident']
```

Out[21]:

	Event.Id	Investigation.Type	Accident.Number	Event.Date	Location	Country	Lati
23	20020917X02333	Incident	LAX82IA034	1982-01-03	VAN NUYS, CA	United States	
40	20020917X01764	Incident	ATL82IA029	1982-01-05	PENSACOLA, FL	United States	
79	20020917X01897	Incident	CHI82IA026	1982-01-12	CHICAGO, IL	United States	
80	20020917X01765	Incident	ATL82IA034	1982-01-12	CLARKSBURG, WV	United States	
119	20020917X01766	Incident	ATL82IA038	1982-01-19	WASHINGTON, DC	United States	
...	...	...	...	...	...	...	

	Event.Id	Investigation.Type	Accident.Number	Event.Date	Location	Country	Lati
88809	20221125106356	Incident	DCA23WA074	2022-11-21	Maturin,	Venezuela	
88819	20221125106362	Incident	DCA23WA076	2022-11-24	Maiquetía,	Venezuela	
88821	20221125106357	Incident	DCA23WA075	2022-11-25	Breslau,	Canada	
88826	20221222106484	Incident	DCA23WA099	2022-11-26	Bangkok,	Thailand	
88851	20221222106485	Incident	DCA23WA100	2022-12-05	Bangkok,	Thailand	

3874 rows × 31 columns

Data Cleaning

In this process, I will be checking on and removing null or missing values and duplicates. We will also be dropping columns that wont be needed in the analysis and changing certain aspects within the data.

In [22]:

```
#Checking for missing values in the data.
def identify_missing_values(df):
    """Identify if the data has missing values."""
    if df.isnull().any().any():
        print("The Data has missing values.")
    else:
        print("The Data has no missing values.")

identify_missing_values(df)
```

```
The Data has missing values.
```

In [23]:

```
df.isnull()
```

Out[23]:

	Event.Id	Investigation.Type	Accident.Number	Event.Date	Location	Country	Latitude	Longituc
0	False	False	False	False	False	False	True	Tru
1	False	False	False	False	False	False	True	Tru
2	False	False	False	False	False	False	False	Fals
3	False	False	False	False	False	False	True	Tru
4	False	False	False	False	False	False	True	Tru
...	...	...	...	...	...	...	...	
88884	False	False	False	False	False	False	True	Tru
88885	False	False	False	False	False	False	True	Tru
88886	False	False	False	False	False	False	False	Fals

	Event.Id	Investigation.Type	Accident.Number	Event.Date	Location	Country	Latitude	Longituc
88887	False	False	False	False	False	False	True	Tru
88888	False	False	False	False	False	False	True	Tru

88889 rows × 31 columns

In [24]:
```
#Find the number of missing values in each column
df.isnull().sum()
```

Out[24]:
```
Event.Id                      0
Investigation.Type            0
Accident.Number               0
Event.Date                    0
Location                     52
Country                     226
Latitude                  54507
Longitude                 54516
Airport.Code              38757
Airport.Name              36185
Injury.Severity            1000
Aircraft.damage            3194
Aircraft.Category         56602
Registration.Number        1382
Make                         63
Model                        92
Amateur.Built               102
Number.of.Engines          6084
Engine.Type                7096
FAR.Description           56866
Schedule                  76307
Purpose.of.flight          6192
Air.carrier               72241
Total.Fatal.Injuries      11401
Total.Serious.Injuries    12510
Total.Minor.Injuries      11933
Total.Uninjured            5912
Weather.Condition          4492
Broad.phase.of.flight     27165
Report.Status              6384
Publication.Date          13771
dtype: int64
```

In [25]:
```
#We will now drop multiple columns
df.drop(['Event.Id', 'Investigation.Type', 'Accident.Number','Latitude', 'Longitude', '
df.head()
```

Out[25]:

	Event.Date	Location	Country	Injury.Severity	Aircraft.damage	Make	Model	Engine.Type	Tc
0	1948-10-24	MOOSE CREEK, ID	United States	Fatal(2)	Destroyed	Stinson	108-3	Reciprocating	
1	1962-07-19	BRIDGEPORT, CA	United States	Fatal(4)	Destroyed	Piper	PA24-180	Reciprocating	
2	1974-08-30	Saltville, VA	United States	Fatal(3)	Destroyed	Cessna	172M	Reciprocating	
3	1977-06-19	EUREKA, CA	United States	Fatal(2)	Destroyed	Rockwell	112	Reciprocating	

	Event.Date	Location	Country	Injury.Severity	Aircraft.damage	Make	Model	Engine.Type	To
4	1979-08-02	Canton, OH	United States	Fatal(1)	Destroyed	Cessna	501	NaN	

```
In [26]: #let us look through the columns remaining.
         df.columns
```

```
Out[26]: Index(['Event.Date', 'Location', 'Country', 'Injury.Severity',
                'Aircraft.damage', 'Make', 'Model', 'Engine.Type',
                'Total.Fatal.Injuries', 'Total.Serious.Injuries',
                'Total.Minor.Injuries', 'Total.Uninjured', 'Weather.Condition',
                'Broad.phase.of.flight', 'Report.Status'],
               dtype='object')
```

```
In [29]: #We will now check on missing values again.
         df.isna().sum().sort_values(ascending=False)
```

```
Out[29]: Broad.phase.of.flight     27165
         Total.Serious.Injuries    12510
         Total.Minor.Injuries      11933
         Total.Fatal.Injuries      11401
         Engine.Type                7096
         Report.Status              6384
         Total.Uninjured            5912
         Weather.Condition          4492
         Aircraft.damage            3194
         Injury.Severity            1000
         Country                     226
         Model                        92
         Make                         63
         Location                     52
         Event.Date                    0
         dtype: int64
```

```
In [27]: #use unique to see the unique values in the variable.
         df['Country'].unique()
```

```
Out[27]: array(['United States', nan, 'GULF OF MEXICO', 'Puerto Rico',
                'ATLANTIC OCEAN', 'HIGH ISLAND', 'Bahamas', 'MISSING', 'Pakistan',
                'Angola', 'Germany', 'Korea, Republic Of', 'Martinique',
                'American Samoa', 'PACIFIC OCEAN', 'Canada', 'Bolivia', 'Mexico',
                'Dominica', 'Netherlands Antilles', 'Iceland', 'Greece', 'Guam',
                'Australia', 'CARIBBEAN SEA', 'West Indies', 'Japan',
                'Philippines', 'Venezuela', 'Bermuda', 'San Juan Islands',
                'Colombia', 'El Salvador', 'United Kingdom',
                'British Virgin Islands', 'Netherlands', 'Costa Rica',
                'Mozambique', 'Jamaica', 'Panama', 'Guyana', 'Norway', 'Hong Kong',
                'Portugal', 'Malaysia', 'Turks And Caicos Islands',
                'Northern Mariana Islands', 'Dominican Republic', 'Suriname',
                'Honduras', 'Congo', 'Belize', 'Guatemala', 'Anguilla', 'France',
                'St Vincent And The Grenadines', 'Haiti', 'Montserrat',
                'Papua New Guinea', 'Cayman Islands', 'Sweden', 'Taiwan',
                'Senegal', 'Barbados', 'BLOCK 651A', 'Brazil', 'Mauritius',
                'Argentina', 'Kenya', 'Ecuador', 'Aruba', 'Saudi Arabia', 'Cuba',
                'Italy', 'French Guiana', 'Denmark', 'Sudan', 'Spain',
                'Federated States Of Micronesia', 'St Lucia', 'Switzerland',
                'Central African Republic', 'Algeria', 'Turkey', 'Nicaragua',
                'Marshall Islands', 'Trinidad And Tobago', 'Poland', 'Belarus',
                'Austria', 'Malta', 'Cameroon', 'Solomon Islands', 'Zambia',
                'Peru', 'Croatia', 'Fiji', 'South Africa', 'India', 'Ethiopia',
                'Ireland', 'Chile', 'Antigua And Barbuda', 'Uganda', 'China',
                'Cambodia', 'Paraguay', 'Thailand', 'Belgium', 'Gambia', 'Uruguay',
                'Tanzania', 'Mali', 'Indonesia', 'Bahrain', 'Kazakhstan', 'Egypt',
                'Russia', 'Cyprus', "Cote D'ivoire", 'Nigeria', 'Greenland',
                'Vietnam', 'New Zealand', 'Singapore', 'Ghana', 'Gabon', 'Nepal',
                'Slovakia', 'Finland', 'Liberia', 'Romania', 'Maldives',
                'Antarctica', 'Zimbabwe', 'Botswana', 'Isle of Man', 'Latvia',
                'Niger', 'French Polynesia', 'Guadeloupe', 'Ivory Coast',
                'Tunisia', 'Eritrea', 'Gibraltar', 'Namibia', 'Czech Republic',
                'Benin', 'Bosnia And Herzegovina', 'Israel', 'Estonia',
                'St Kitts And Nevis', 'Sierra Leone', 'Corsica', 'Scotland',
                'Reunion', 'United Arab Emirates', 'Afghanistan', 'Ukraine',
                'Hungary', 'Bangladesh', 'Morocco', 'Iraq', 'Jordan', 'Qatar',
                'Madagascar', 'Malawi', 'Unknown', 'Central Africa', 'South Sudan',
                'Saint Barthelemy', 'Micronesia', 'South Korea', 'Kyrgyzstan',
                'Turks And Caicos', 'Eswatini', 'Tokelau', 'Sint Maarten', 'Macao',
                'Seychelles', 'Rwanda', 'Palau', 'Luxembourg', 'Lebanon',
                'Bosnia and Herzegovina', 'Libya', 'Guinea',
                'Saint Vincent and the Grenadines', 'UN', 'Iran', 'Lithuania',
                'Malampa', 'Antigua and Barbuda', 'AY', 'Chad', 'Cayenne',
                'New Caledonia', 'Yemen', 'Slovenia', 'Nauru', 'Niue', 'Bulgaria',
                'Republic of North Macedonia', 'Virgin Islands', 'Somalia',
                'Pacific Ocean', 'Obyan', 'Mauritania', 'Albania', 'Wolseley',
                'Wallis and Futuna', 'Saint Pierre and Miquelon', 'Georgia',
                "Côte d'Ivoire", 'South Korean', 'Serbia', 'MU', 'Guernsey',
                'Great Britain', 'Turks and Caicos Islands'], dtype=object)
```

In [28]:

```
#We capitalize the first letter of each word for uniformity.
df['Country'] = df['Country'].str.capitalize()
```

In [55]:

```
df['Country'] = df['Country'].astype(str)
```

In [56]:

```
#Checking on the data types of the columns
df.dtypes
```

Out[56]:

```
Event.Date                datetime64[ns]
Location                          object
Country                           object
Injury.Severity                   object
Aircraft.Damage                   object
Make                              object
Model                             object
Engine.Type                       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
dtype: object
```

In [57]:

```
#Change 'Event.Date' to datetime
df['Event.Date'] = pd.to_datetime(df['Event.Date'])
```

In [58]:

```
#Confirming if the changes were made
df.dtypes
```

Out[58]:

```
Event.Date                datetime64[ns]
Location                          object
Country                           object
Injury.Severity                   object
```

```
Aircraft.Damage                  object
Make                             object
Model                            object
Engine.Type                      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
dtype: object
```

In [59]:

```
#We are going to change the 'Make' column contents into lower case
df['Make'] = df['Make'].str.lower()
```

In [60]:

```
#We will also change the 'Location' column contents into upper case.
df['Location'] = df['Location'].str.upper()
```

In [61]:

```
#Just for uniformity purposes, we will capitalize or rather make the columns into title
df.columns = map(lambda x: str(x).title(), df.columns)
df.head()
```

Out[61]:

	Event.Date	Location	Country	Injury.Severity	Aircraft.Damage	Make	Model	Engine.Type	To
0	1948-10-24	MOOSE CREEK, ID	United states	Fatal(2)	Destroyed	stinson	108-3	Reciprocating	
1	1962-07-19	BRIDGEPORT, CA	United states	Fatal(4)	Destroyed	piper	PA24-180	Reciprocating	
2	1974-08-30	SALTVILLE, VA	United states	Fatal(3)	Destroyed	cessna	172M	Reciprocating	
3	1977-06-19	EUREKA, CA	United states	Fatal(2)	Destroyed	rockwell	112	Reciprocating	
4	1979-08-02	CANTON, OH	United states	Fatal(1)	Destroyed	cessna	501	Unknown	

In [62]:

```
#Checking on the top 15 rows
df.head(15)
```

Out[62]:

	Event.Date	Location	Country	Injury.Severity	Aircraft.Damage	Make	Model	Engine.Ty
0	1948-10-24	MOOSE CREEK, ID	United states	Fatal(2)	Destroyed	stinson	108-3	Reciprocati
1	1962-07-19	BRIDGEPORT, CA	United states	Fatal(4)	Destroyed	piper	PA24-180	Reciprocati
2	1974-08-30	SALTVILLE, VA	United states	Fatal(3)	Destroyed	cessna	172M	Reciprocati
3	1977-06-19	EUREKA, CA	United states	Fatal(2)	Destroyed	rockwell	112	Reciprocati
4	1979-08-02	CANTON, OH	United states	Fatal(1)	Destroyed	cessna	501	Unkno

	Event.Date	Location	Country	Injury.Severity	Aircraft.Damage	Make	Model	Engine.Ty
5	1979-09-17	BOSTON, MA	United states	Non-Fatal	Substantial	mcdonnell douglas	DC9	Turbo F
6	1981-08-01	COTTON, MN	United states	Fatal(4)	Destroyed	cessna	180	Reciprocati
7	1982-01-01	PULLMAN, WA	United states	Non-Fatal	Substantial	cessna	140	Reciprocati
8	1982-01-01	EAST HANOVER, NJ	United states	Non-Fatal	Substantial	cessna	401B	Reciprocati
9	1982-01-01	JACKSONVILLE, FL	United states	Non-Fatal	Substantial	north american	NAVION L-17B	Reciprocati
10	1982-01-01	HOBBS, NM	United states	Non-Fatal	Substantial	piper	PA-28-161	Reciprocati
11	1982-01-01	TUSKEGEE, AL	United states	Non-Fatal	Substantial	beech	V35B	Reciprocati
12	1982-01-02	HOMER, LA	United states	Non-Fatal	Destroyed	bellanca	17-30A	Reciprocati
13	1982-01-02	HEARNE, TX	United states	Fatal(1)	Destroyed	cessna	R172K	Reciprocati
14	1982-01-02	CHICKASHA, OK	United states	Fatal(1)	Destroyed	navion	A	Reciprocati

In [63]:

```python
#Because we are looking to purchase aircrafts, we will drop the missing values in both 
df.dropna(subset=['Make', 'Model'], inplace=True)
```

In [64]:

```python
#Let us check whether the changes were made
df[['Make', 'Model']].isna().sum()
```

Out[64]:

```
Make     0
Model    0
dtype: int64
```

In [65]:

```python
#Checking on the median values of the numerical variables
df[['Total.Fatal.Injuries', 'Total.Serious.Injuries', 'Total.Minor.Injuries', 'Total.Un
```

Out[65]:

```
Total.Fatal.Injuries      0.0
Total.Serious.Injuries    0.0
Total.Minor.Injuries      0.0
Total.Uninjured           1.0
dtype: float64
```

In [66]:

```python
#Now we will fill the missing values in the numerical variables
df['Total.Fatal.Injuries'] = df['Total.Fatal.Injuries'].fillna(0)
```

In [67]:

```python
df['Total.Serious.Injuries'] = df['Total.Serious.Injuries'].fillna(0)
```

In [68]:

```python
df['Total.Minor.Injuries'] = df['Total.Minor.Injuries'].fillna(0)
```

In [69]:

```python
df['Total.Uninjured'] = df['Total.Uninjured'].fillna(0)
```

In [70]:

```
#Let us check whether the changes were made
df[['Total.Fatal.Injuries', 'Total.Serious.Injuries', 'Total.Minor.Injuries', 'Total.Un
```

Out[70]:

```
Total.Fatal.Injuries      0
Total.Serious.Injuries    0
Total.Minor.Injuries      0
Total.Uninjured           0
dtype: int64
```

In [71]:

```
df.head(15)
```

Out[71]:

	Event.Date	Location	Country	Injury.Severity	Aircraft.Damage	Make	Model	Engine.Ty
0	1948-10-24	MOOSE CREEK, ID	United states	Fatal(2)	Destroyed	stinson	108-3	Reciprocati
1	1962-07-19	BRIDGEPORT, CA	United states	Fatal(4)	Destroyed	piper	PA24-180	Reciprocati
2	1974-08-30	SALTVILLE, VA	United states	Fatal(3)	Destroyed	cessna	172M	Reciprocati
3	1977-06-19	EUREKA, CA	United states	Fatal(2)	Destroyed	rockwell	112	Reciprocati
4	1979-08-02	CANTON, OH	United states	Fatal(1)	Destroyed	cessna	501	Unkno
5	1979-09-17	BOSTON, MA	United states	Non-Fatal	Substantial	mcdonnell douglas	DC9	Turbo F
6	1981-08-01	COTTON, MN	United states	Fatal(4)	Destroyed	cessna	180	Reciprocati
7	1982-01-01	PULLMAN, WA	United states	Non-Fatal	Substantial	cessna	140	Reciprocati
8	1982-01-01	EAST HANOVER, NJ	United states	Non-Fatal	Substantial	cessna	401B	Reciprocati
9	1982-01-01	JACKSONVILLE, FL	United states	Non-Fatal	Substantial	north american	NAVION L-17B	Reciprocati
10	1982-01-01	HOBBS, NM	United states	Non-Fatal	Substantial	piper	PA-28-161	Reciprocati
11	1982-01-01	TUSKEGEE, AL	United states	Non-Fatal	Substantial	beech	V35B	Reciprocati
12	1982-01-02	HOMER, LA	United states	Non-Fatal	Destroyed	bellanca	17-30A	Reciprocati
13	1982-01-02	HEARNE, TX	United states	Fatal(1)	Destroyed	cessna	R172K	Reciprocati
14	1982-01-02	CHICKASHA, OK	United states	Fatal(1)	Destroyed	navion	A	Reciprocati

In [72]:

```
#Lets check the ststistics of the data
df.describe().T
```

Out[72]:

	count	mean	min	25%	50%	75%	max	st
Event.Date	88777	1999-09-16 06:32:24.260337664	1948-10-24 00:00:00	1989-01-14 00:00:00	1998-07-16 00:00:00	2009-06-28 00:00:00	2022-12-29 00:00:00	Na
Total.Fatal.Injuries	88777.0	0.564493	0.0	0.0	0.0	0.0	349.0	5.12924
Total.Serious.Injuries	88777.0	0.240445	0.0	0.0	0.0	0.0	161.0	1.43494
Total.Minor.Injuries	88777.0	0.309258	0.0	0.0	0.0	0.0	380.0	2.08482
Total.Uninjured	88777.0	4.968145	0.0	0.0	1.0	2.0	699.0	27.00309

In [73]:

```
df.isna().sum()
```

Out[73]:

```
Event.Date                0
Location                  0
Country                   0
Injury.Severity           0
Aircraft.Damage           0
Make                      0
Model                     0
Engine.Type               0
Total.Fatal.Injuries      0
Total.Serious.Injuries    0
Total.Minor.Injuries      0
Total.Uninjured           0
Weather.Condition         0
Broad.Phase.Of.Flight     0
Report.Status             0
dtype: int64
```

In [74]:

```
#We still have missing values. so we will replace them with the replace function.
#The function is going to replace the missing values with 'Unavailable', 'Unknown' and
df['Injury.Severity'] = df['Injury.Severity'].fillna('Unavailable')
df['Aircraft.Damage'] = df['Aircraft.Damage'].fillna('Unknown')
df['Engine.Type'] = df['Engine.Type'].fillna('Unknown')
df['Location'] = df['Location'].fillna('Unknown')
df['Weather.Condition'] = df['Weather.Condition'].fillna('UNK')
df['Broad.Phase.Of.Flight'] = df['Broad.Phase.Of.Flight'].fillna('Unknown')
df['Report.Status'] = df['Report.Status'].fillna('Unknown')
df
```

Out[74]:

	Event.Date	Location	Country	Injury.Severity	Aircraft.Damage	Make	Model	Engine.Ty
0	1948-10-24	MOOSE CREEK, ID	United states	Fatal(2)	Destroyed	stinson	108-3	Reciprocati
1	1962-07-19	BRIDGEPORT, CA	United states	Fatal(4)	Destroyed	piper	PA24-180	Reciprocati
2	1974-08-30	SALTVILLE, VA	United states	Fatal(3)	Destroyed	cessna	172M	Reciprocati
3	1977-06-19	EUREKA, CA	United states	Fatal(2)	Destroyed	rockwell	112	Reciprocati
4	1979-08-02	CANTON, OH	United states	Fatal(1)	Destroyed	cessna	501	Unkno

	Event.Date	Location	Country	Injury.Severity	Aircraft.Damage	Make	Model	Engine.Ty
...	...	...	...	...	...	...	...	
88884	2022-12-26	ANNAPOLIS, MD	United states	Minor	Unknown	piper	PA-28-151	Unkno
88885	2022-12-26	HAMPTON, NH	United states	Unavailable	Unknown	bellanca	7ECA	Unkno
88886	2022-12-26	PAYSON, AZ	United states	Non-Fatal	Substantial	american champion aircraft	8GCBC	Unkno
88887	2022-12-26	MORGAN, UT	United states	Unavailable	Unknown	cessna	210N	Unkno
88888	2022-12-29	ATHENS, GA	United states	Minor	Unknown	piper	PA-24-260	Unkno

88777 rows × 15 columns

In [75]:

```
#Let us check whether the changes were made
df.isna().sum()
```

Out[75]:

```
Event.Date                0
Location                  0
Country                   0
Injury.Severity           0
Aircraft.Damage           0
Make                      0
Model                     0
Engine.Type               0
Total.Fatal.Injuries      0
Total.Serious.Injuries    0
Total.Minor.Injuries      0
Total.Uninjured           0
Weather.Condition         0
Broad.Phase.Of.Flight     0
Report.Status             0
dtype: int64
```

Checking for Duplicates

In [76]:

```
#If the out put comes back as 'True', it means there are duplicates.
#If the output comes back as 'False', it means there are no duplicates.
df.duplicated().any()
```

Out[76]:

```
True
```

In [77]:

```
#Since we have duplicates. Now let us check the number of duplicates.
df.duplicated().sum()
```

Out[77]:

```
36
```

In [78]:

```
#Let us drop duplicates
df.drop_duplicates(inplace=True)
```

```
In [79]: df.duplicated().sum()
```

```
Out[79]: 0
```

```
In [80]: #which 'Make' has the most number of 'Total.Fatal.Injuries'
         df.groupby(['Make'])['Total.Fatal.Injuries'].sum().sort_values(ascending=False).head(1)
```

```
Out[80]: Make
         cessna    9630.0
         Name: Total.Fatal.Injuries, dtype: float64
```

```
In [81]: #which 'Location' has the most number of 'Total.Fatal.Injuries'
         df.groupby(['Location'])['Total.Fatal.Injuries'].sum().sort_values(ascending=False).hea
```

```
Out[81]: Location
         NEW DELHI, INDIA    708.0
         Name: Total.Fatal.Injuries, dtype: float64
```

```
In [82]: #in bottom 10 countries, which 'Country' has the most number of 'Total.Fatal.Injuries'
         Bottom_10_Countries =df.groupby(['Country'])['Total.Fatal.Injuries'].sum().sort_values(
         Bottom_10_Countries
```

```
Out[82]: Country
         Aruba                  0.0
         Albania                0.0
         Ay                     0.0
         Bermuda                0.0
         Martinique             0.0
         Sierra leone           0.0
         Seychelles             0.0
         Luxembourg             0.0
         Trinidad and tobago    0.0
         Obyan                  0.0
         Name: Total.Fatal.Injuries, dtype: float64
```

```
In [83]: #In top 10 countries, which 'Country' has the most number of 'Total.Fatal.Injuries'
         Top_10_Countries =df.groupby(['Country'])['Total.Fatal.Injuries'].sum().sort_values(asc
         Top_10_Countries
```

```
Out[83]: Country
         United states    30151.0
         Brazil            1242.0
         India              970.0
         Indonesia          949.0
         Canada             943.0
         France             813.0
         Russia             765.0
         Colombia           701.0
         Mexico             653.0
         Peru               490.0
         Name: Total.Fatal.Injuries, dtype: float64
```

```
In [84]: #which 'Make' has the most number of 'Total.Minor.Injuries'
         Top_10_Makes = df.groupby(['Make'])['Total.Minor.Injuries'].sum().sort_values(ascending
         Top_10_Makes
```

```
Out[84]: Make
         cessna               6874.0
         piper                3757.0
         boeing               2761.0
         mcdonnell douglas    1505.0
```

```
beech                1340.0
bell                 1115.0
airbus industrie      399.0
mooney                391.0
hughes                344.0
robinson              319.0
Name: Total.Minor.Injuries, dtype: float64
```

In [85]:

```
#in bottom 10, which 'Make' has the most number of 'Total.Minor.Injuries'
Bottom_10_Makes = df.groupby(['Make'])['Total.Minor.Injuries'].sum().sort_values(ascend
Bottom_10_Makes
```

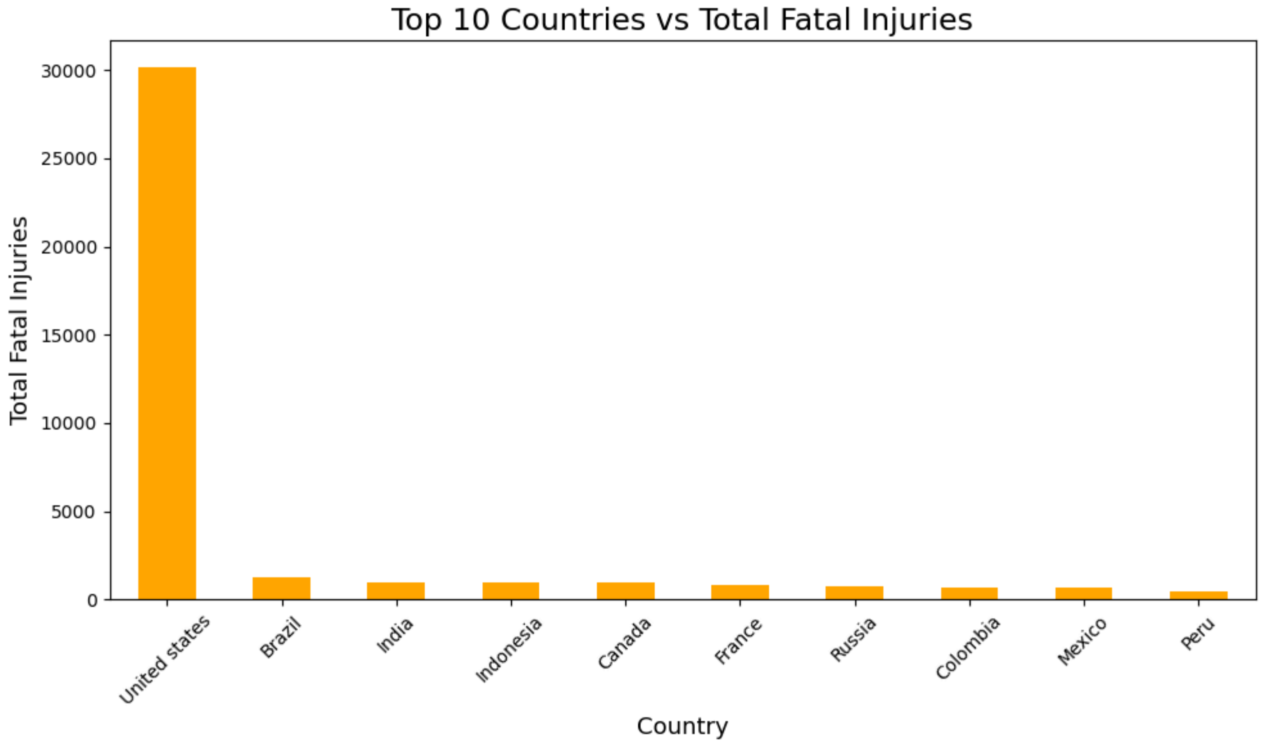
Out[85]:

```
Make
hancock                         0.0
hampson                         0.0
hammack                         0.0
hamlin john d                   0.0
hamilton                        0.0
hamer                           0.0
hamburger flugzeugbau (hfb)     0.0
ham                             0.0
halstead                        0.0
zwicker murray r                0.0
Name: Total.Minor.Injuries, dtype: float64
```

Data Visualisation

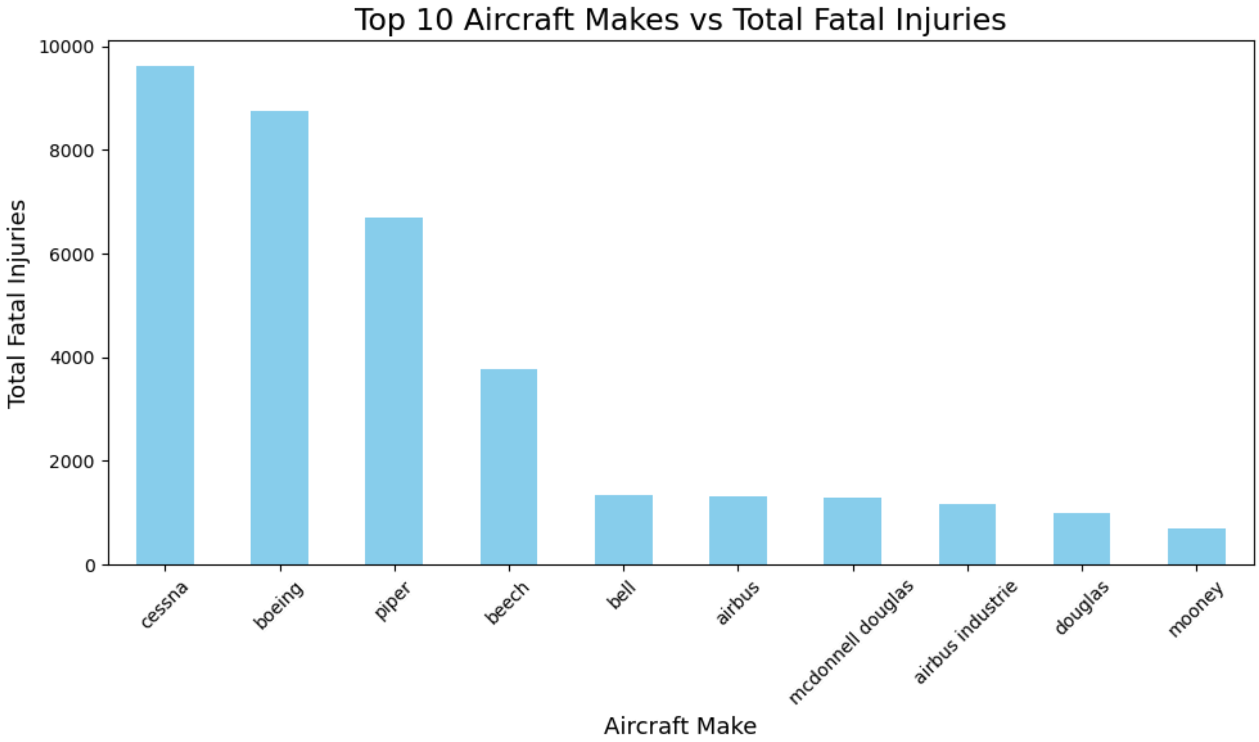
In [86]:

```
#We will plot a graph of top 10 countries
#Group by 'Country' and sum up the 'Total.Fatal.Injuries'
Top_10_Countries = df.groupby(['Country'])['Total.Fatal.Injuries'].sum().sort_values(as
# Plot the top 10 countries against total fatal injuries
plt.figure(figsize=(10, 6))
Top_10_Countries.plot(kind='bar', color='orange')

# Add title and labels
plt.title('Top 10 Countries vs Total Fatal Injuries', fontsize=17)
plt.xlabel('Country', fontsize=13)
plt.ylabel('Total Fatal Injuries', fontsize=13)

# Rotate x-axis labels for better readability
plt.xticks(rotation=45)

# Show the plot
plt.tight_layout()

plt.show()
```



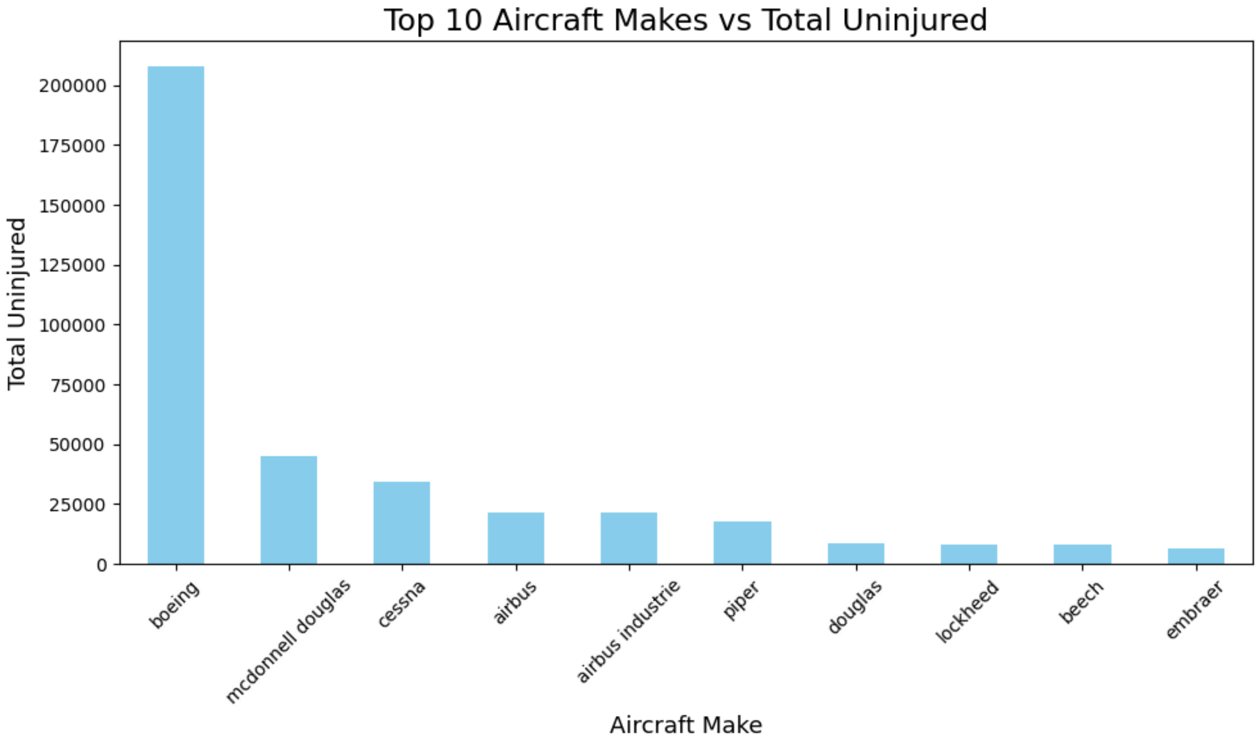
In [87]:

```
#We will plot a graph of Top 10 Makes against Total.Fatal.Injuries
# Group by 'Make' and sum up the 'Total Fatal.Injuries'
top_makes = df.groupby('Make')['Total.Fatal.Injuries'].sum().sort_values(ascending=Fals

# Plot the top 10 makes against total fatal injuries
plt.figure(figsize=(10, 6))
top_makes.plot(kind='bar', color='skyblue')

# Add title and labels
plt.title('Top 10 Aircraft Makes vs Total Fatal Injuries', fontsize=17)
plt.xlabel('Aircraft Make', fontsize=13)
plt.ylabel('Total Fatal Injuries', fontsize=13)

# Rotate x-axis labels for better readability
plt.xticks(rotation=45)

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



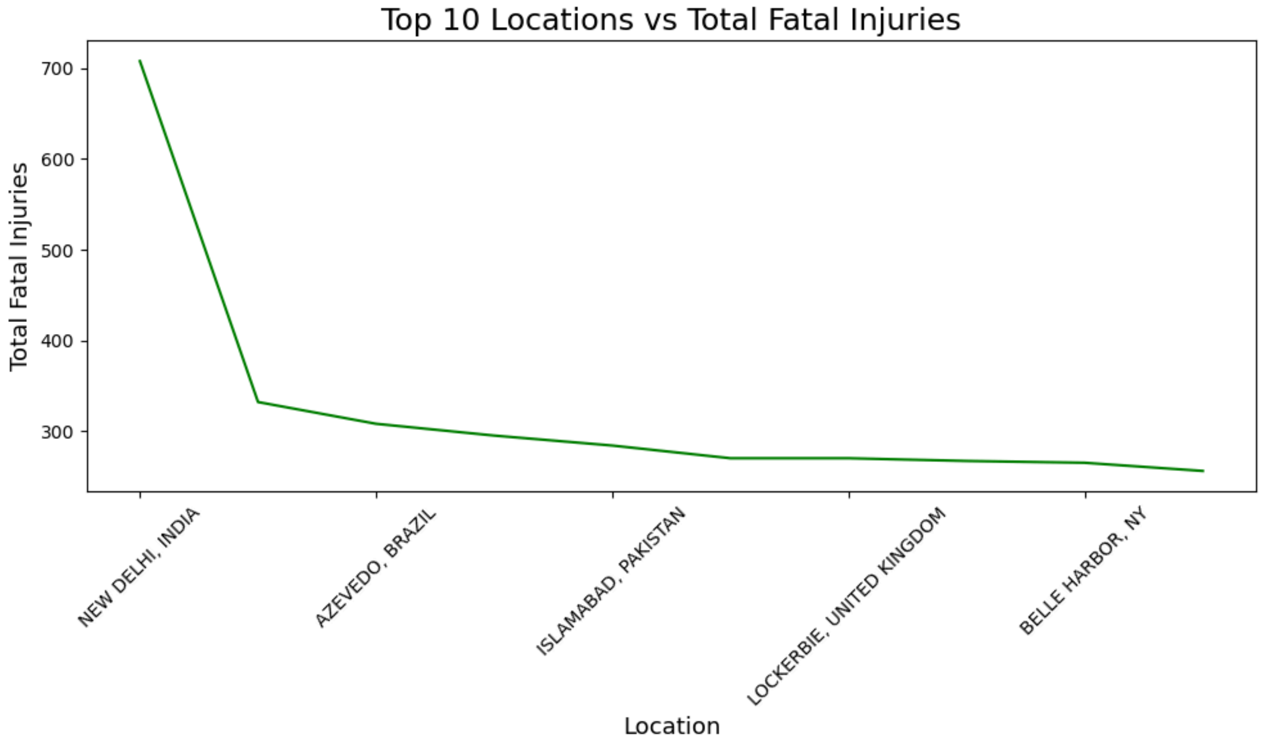
In [94]:

```
#We will plot a graph of Top 10 Makes against Total.Uninjured
# Group by 'Make' and sum up the 'Total.Uninjured'
top_makes = df.groupby('Make')['Total.Uninjured'].sum().sort_values(ascending=False).he

# Plot the top 10 makes against total uninjured.
plt.figure(figsize=(10, 6))
top_makes.plot(kind='bar', color='skyblue')

# Add title and labels
plt.title('Top 10 Aircraft Makes vs Total Uninjured', fontsize=17)
plt.xlabel('Aircraft Make', fontsize=13)
plt.ylabel('Total Uninjured', fontsize=13)

# Rotate x-axis labels for better readability
plt.xticks(rotation=45)

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



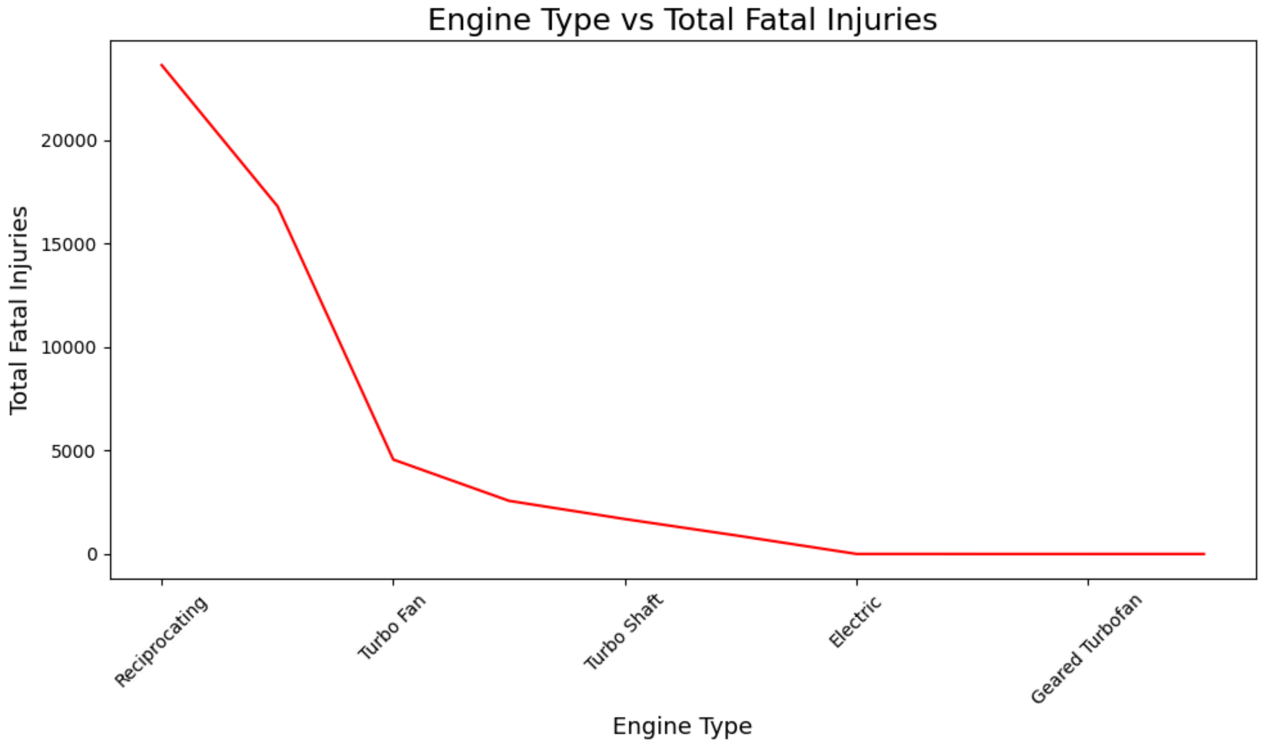
In [88]:

```
#Group by "location" and sum up the 'Total.Fatal.Injuries'
top_locations = df.groupby('Location')['Total.Fatal.Injuries'].sum().sort_values(ascend

# Plot the top 10 locations against total fatal injuries
plt.figure(figsize=(10, 6))
top_locations.plot(kind='line',color='green')

# Add title and labels
plt.title('Top 10 Locations vs Total Fatal Injuries', fontsize=17)
plt.xlabel('Location', fontsize=13)
plt.ylabel('Total Fatal Injuries', fontsize=13)

# Rotate x-axis labels for better readability
plt.xticks(rotation=45)

# Show the plot
plt.tight_layout()

plt.show()
```



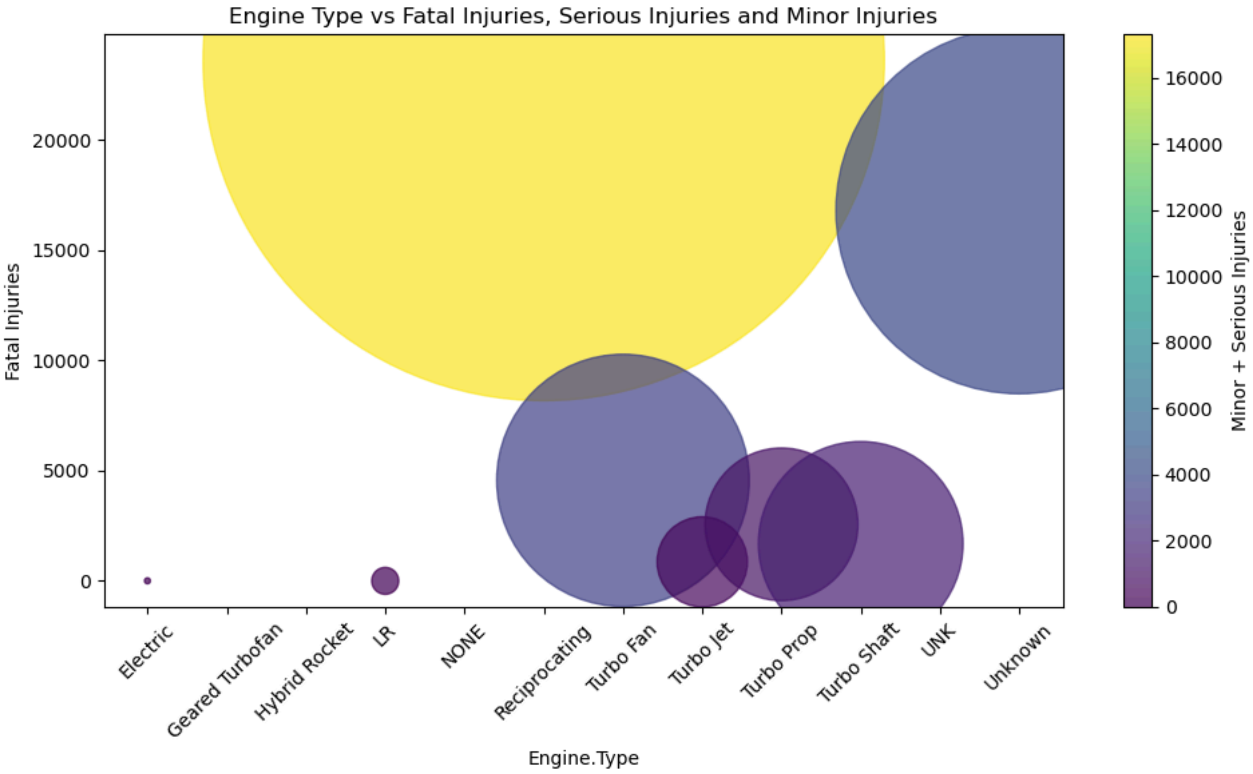
In [93]:

```
#Creating a line graph to show the relationship between engine type and total fatal inj

#Group by 'Engine Type' and sum up the 'Total.Fatal.Injuries'
engine_fatalities = df.groupby('Engine.Type')['Total.Fatal.Injuries'].sum().sort_values

# Plot the weather condition against total fatal injuries
plt.figure(figsize=(10, 6))
engine_fatalities.plot(kind='line', color='red')

# Add title and labels
plt.title('Engine Type vs Total Fatal Injuries', fontsize=17)
plt.xlabel('Engine Type', fontsize=13)
plt.ylabel('Total Fatal Injuries', fontsize=13)

# Rotate x-axis labels for better readability
plt.xticks(rotation=45)

# Show the plot
plt.tight_layout()

plt.show()
```



In [90]:

```
#We will create a scatter plot to show the relationship between engine type against tot
#Grouping the data by 'Engine.Type and summing up the 'Total.Fatal.Injuries', 'Total.Mi
injuries_by_engine = df.groupby('Engine.Type')[['Total.Fatal.Injuries', 'Total.Serious.

#Creating a figure and axis objects
fig, ax = plt.subplots(figsize=(10, 6))

#Creating a scatter plot. With 'Engine.Type' on the x-axis and 'Total.Fatal.Injuries' o
scatter = ax.scatter(
    injuries_by_engine.index, # x-axis: Engine.Type
    injuries_by_engine['Total.Fatal.Injuries'], # y-axis: Total.Fatal.Injuries
    s=injuries_by_engine['Total.Serious.Injuries'] *10,
    c=injuries_by_engine['Total.Minor.Injuries'],
    cmap='viridis',
    alpha=0.7
)
#Setting the x-axis and y-axis labels
ax.set_xlabel('Engine.Type')

ax.set_ylabel('Fatal Injuries')

#Setting the title
ax.set_title('Engine Type vs Fatal Injuries, Serious Injuries and Minor Injuries')

#Rotate the x-axis labels
plt.xticks(rotation=45)

#Add color bar to indicate what the color represents
cbar = plt.colorbar(scatter)
cbar.set_label('Minor + Serious Injuries')

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



In [91]:
```
df.head()
```

Out[91]:

	Event.Date	Location	Country	Injury.Severity	Aircraft.Damage	Make	Model	Engine.Type	To
0	1948-10-24	MOOSE CREEK, ID	United states	Fatal(2)	Destroyed	stinson	108-3	Reciprocating	
1	1962-07-19	BRIDGEPORT, CA	United states	Fatal(4)	Destroyed	piper	PA24-180	Reciprocating	
2	1974-08-30	SALTVILLE, VA	United states	Fatal(3)	Destroyed	cessna	172M	Reciprocating	
3	1977-06-19	EUREKA, CA	United states	Fatal(2)	Destroyed	rockwell	112	Reciprocating	
4	1979-08-02	CANTON, OH	United states	Fatal(1)	Destroyed	cessna	501	Unknown	

Exporting the Clean Dataset

In [92]:
```
# We will export our dataframe into a csv file.
# we use the to_csv function to create a csv file with the name
# and export it
df.to_csv("Aviation_Data.csv")
```