1 DSF PHASE PROJECT ON AVIATION DATASET ANALYSIS

1.1 Overview of the dataset

This project outlines the analysis based on a dataset provided by the National Safety and Transport Board that relayed data on aviation accidents over the past decades (1962-2022).

1.2 Business Understanding

Over the past decades, the rates of aviation accidents have been influenced by various factors, such as aircraft make and model. Understanding how these factors correlate provides actionable insights into improving aviation safety, identifying design flaws, and implementing better operational protocols. This analysis can guide manufacturers and stakeholders to make data-driven decisions for risk regulation and enhance safety.

2 Data Cleaning

After looking into my dataset and reviewing what categorical and numerical data I will require for my analysis, I will start by cleaning the dataset by:

1. Dropping columns that are unnecessary for my analysis
2. Checking for missing values
3. Rectifying column arrangement for uniformity
4. Checking for outliers and eliminating them
5. Checking and dropping duplicates
6. Changing data types for uniformity

I will start by importing the necessary libraries for data cleaning. They are pre-written codes that provide a specific functionality in our case data cleaning and analysis.

In [1]:
```
# Importing the various libraries used for data cleaning and analysis
# We assign each library an alias for instance the alias for pandas is pd
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
```

After importing the various libraries I will load the dataset to my notebook.

In [2]:
```
#Loading the dataset
#Assign the notebook a variable df:
df = pd.read_csv("AviationData.csv", encoding='latin1')
#To confirm the dataset has been uploaded in the notebook we either use df(it will upload the first and la
df
#Similarly we can select the first five rows only by using df.head() and the last five rows only by runnin
```

```
C:\Users\123\AppData\Local\Temp\ipykernel_23448\692569652.py:3: 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='latin1')
```

Out[2]:

	Event.Id	Investigation.Type	Accident.Number	Event.Date	Location	Country	Latitude	Longitude	A
0	20001218X45444	Accident	SEA87LA080	1948-10-24	MOOSE CREEK, ID	United States	NaN	NaN	
1	20001218X45447	Accident	LAX94LA336	1962-07-19	BRIDGEPORT, CA	United States	NaN	NaN	
2	20061025X01555	Accident	NYC07LA005	1974-08-30	Saltville, VA	United States	36.922223	-81.878056	
3	20001218X45448	Accident	LAX96LA321	1977-06-19	EUREKA, CA	United States	NaN	NaN	
4	20041105X01764	Accident	CHI79FA064	1979-08-02	Canton, OH	United States	NaN	NaN	
...	...	...	...	...	...	...	...	...	
88884	20221227106491	Accident	ERA23LA093	2022-12-26	Annapolis, MD	United States	NaN	NaN	
88885	20221227106494	Accident	ERA23LA095	2022-12-26	Hampton, NH	United States	NaN	NaN	
88886	20221227106497	Accident	WPR23LA075	2022-12-26	Payson, AZ	United States	341525N	1112021W	
88887	20221227106498	Accident	WPR23LA076	2022-12-26	Morgan, UT	United States	NaN	NaN	
88888	20221230106513	Accident	ERA23LA097	2022-12-29	Athens, GA	United States	NaN	NaN	

88889 rows × 31 columns

When loading the dataset in my code you notice encoding='latin1'in some cases the code will still run without including latin1 but in my case I had to include it to ensure python correctly interpretes the file content.

Before cleaning we will create a copy to retain the contents of the origin for further analysis if need be

In [3]:
```
#I will run the code below to create a copy by assigning the variable df1
df1=df.copy(deep=True)
#To confirm the copy has been upload run df1
df1
```

Out[3]:

	Event.Id	Investigation.Type	Accident.Number	Event.Date	Location	Country	Latitude	Longitude	A
0	20001218X45444	Accident	SEA87LA080	1948-10-24	MOOSE CREEK, ID	United States	NaN	NaN	
1	20001218X45447	Accident	LAX94LA336	1962-07-19	BRIDGEPORT, CA	United States	NaN	NaN	
2	20061025X01555	Accident	NYC07LA005	1974-08-30	Saltville, VA	United States	36.922223	-81.878056	
3	20001218X45448	Accident	LAX96LA321	1977-06-19	EUREKA, CA	United States	NaN	NaN	
4	20041105X01764	Accident	CHI79FA064	1979-08-02	Canton, OH	United States	NaN	NaN	
...	...	...	...	...	...	...	...	...	
88884	20221227106491	Accident	ERA23LA093	2022-12-26	Annapolis, MD	United States	NaN	NaN	
88885	20221227106494	Accident	ERA23LA095	2022-12-26	Hampton, NH	United States	NaN	NaN	
88886	20221227106497	Accident	WPR23LA075	2022-12-26	Payson, AZ	United States	341525N	1112021W	
88887	20221227106498	Accident	WPR23LA076	2022-12-26	Morgan, UT	United States	NaN	NaN	
88888	20221230106513	Accident	ERA23LA097	2022-12-29	Athens, GA	United States	NaN	NaN	

88889 rows × 31 columns

2.1 Dropping the columns

Here I will drop some the columns that are not necessary for my analysis, to easen the analysis process

In [4]:
```
#I will use the code df1.drop to drop the columns
df1.drop(["Event.Id","Accident.Number","Airport.Code"],axis=1,inplace=True)
#We use axis=1 when dealing with columns
#Inplace modifies the objects without creating a new copy
```

In [5]:
```
#To confirm the columns have been removed I use df1.columns
df1.columns
```

Out[5]:
```
Index(['Investigation.Type', 'Event.Date', 'Location', 'Country', 'Latitude',
       'Longitude', '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')
```

2.2 Changing column format :

For uniformity I will change the format to title

In [6]:
```
#I will use the code below to change the column format to title we assign df1.columns as a variable and us
df1.columns=df1.columns.str.title()
#I will run the code df1.columns to confirm the change
df1.columns
```

Out[6]:
```
Index(['Investigation.Type', 'Event.Date', 'Location', 'Country', 'Latitude',
       'Longitude', '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 our dataset we notice the punctuation is incorrect we will replace the fullstops(.) with whitespaces(" ")

In [7]:
```
#To replace the fullstops(.) with whitespaces(" ") we run the code below
df1.columns=df1.columns.str.replace("."," ")
#Use df1.columns to confirm changes
df1.columns
```

Out[7]:
```
Index(['Investigation Type', 'Event Date', 'Location', 'Country', 'Latitude',
       'Longitude', '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')
```

2.3 Checking for missing values:

We need to check for any missing values and rectify them to avoid problems during analysis. We do this by running the codes below:

In [8]:
```
#Checking for missing values for the entire dataset we use the .isnull().sum().any()
df1.isnull().sum().any()
#If the result is true it indicates that there are missing values from the data set
#However if it indicates false it means the dataset lacks missing values
```

Out[8]:
```
True
```

For our case the dataset has missing values and hence we can check the number of missing values in each column by running the code below

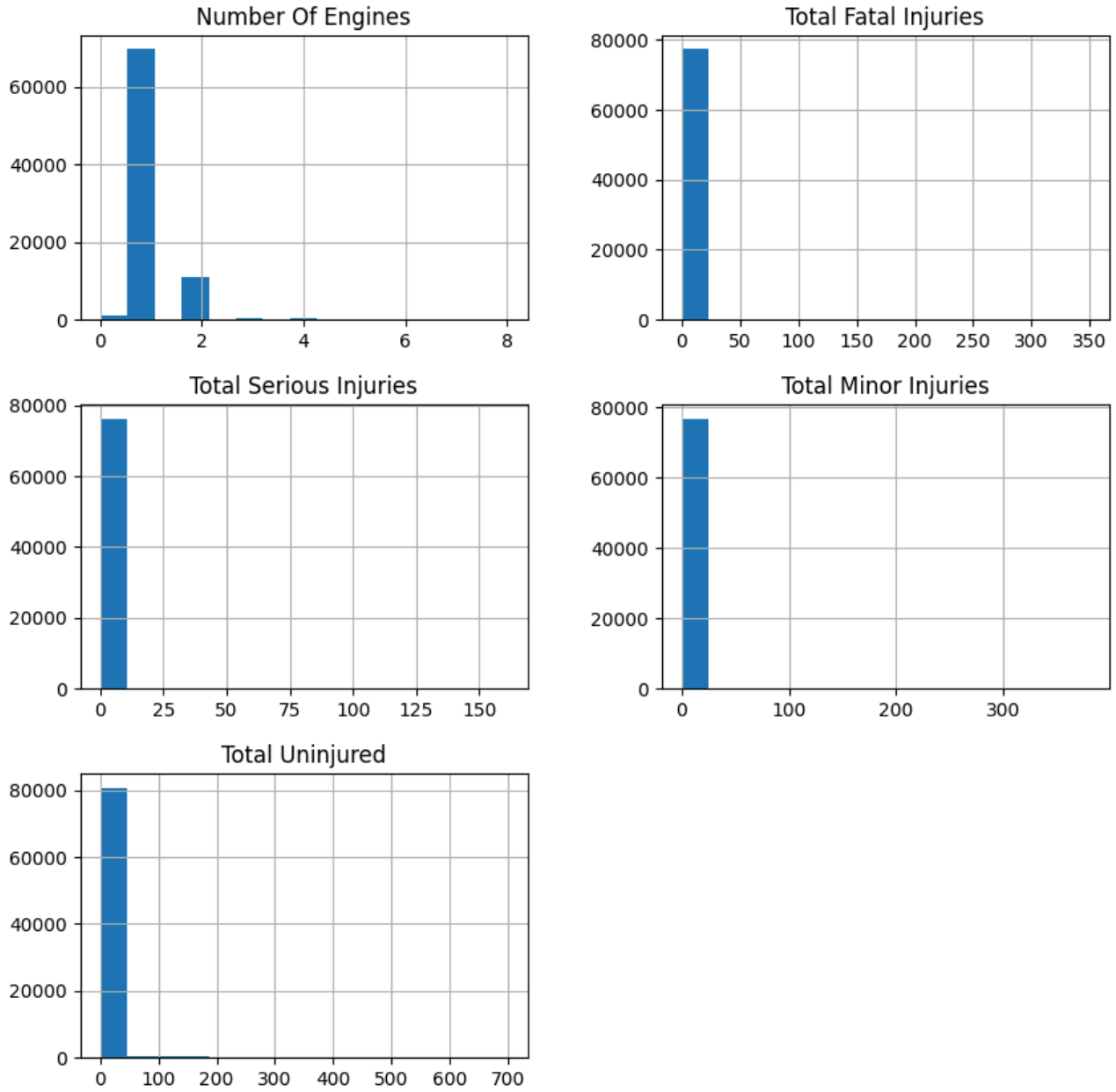
In [9]:
```
#checking for missing vaues for each columns we use .isna().sum()
df1.isna().sum()
```

Out[9]:
```
Investigation Type          0
Event Date                  0
Location                   52
Country                   226
Latitude                54507
Longitude               54516
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
```

In our case we have a lot of missing values. We can rectify the missing values for numerical values filling in statistical measures such mean, mode or median

We start by checking the skewness of the numerical information by plotting a histogram or kdeplot

In [10]:
```
#plotting the histogram
df1.hist(bins=15,figsize=(10,10));
```



Since the plots show the distribution is positively skewed(right skewed) mode will be the most appropriate method to use, as it indicates the most frequent value.

In [11]:
```
#To input the missing values for all numeric data we will use for loop for easier manipulation:
for column in df1.select_dtypes(include=["number"]).columns:
        df1[column].fillna(df1[column].mode()[0],inplace=True)
```

In [12]:
```
#To confirm that all numeric values lack missing data we will run
df1.isna().sum()
```

Out[12]:
```
Investigation Type            0
Event Date                    0
Location                     52
Country                     226
Latitude                  54507
Longitude                 54516
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             0
Engine Type                7096
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              6384
Publication Date          13771
dtype: int64
```

Confirmed that all numeric missing values have been cartered for. Hence will proceed by filling object category by inputing the word ("unknown")

In [13]:
```
#We also run the code using for loop
for column in df1.select_dtypes(include=["object"]).columns:
    df1[column].fillna("Unknown",inplace=True)
```

In [14]:
```
#for confirmation that all missing values have been erased for the entire dataset
df1.isnull().sum().any()
```

Out[14]:
```
False
```

In [15]:
```
df1
```

Out[15]:

	Investigation Type	Event Date	Location	Country	Latitude	Longitude	Airport Name	Injury Severity	Aircraft Damage	Aircraft Category	...
0	Accident	1948-10-24	MOOSE CREEK, ID	United States	Unknown	Unknown	Unknown	Fatal(2)	Destroyed	Unknown	
1	Accident	1962-07-19	BRIDGEPORT, CA	United States	Unknown	Unknown	Unknown	Fatal(4)	Destroyed	Unknown	
2	Accident	1974-08-30	Saltville, VA	United States	36.922223	-81.878056	Unknown	Fatal(3)	Destroyed	Unknown	
3	Accident	1977-06-19	EUREKA, CA	United States	Unknown	Unknown	Unknown	Fatal(2)	Destroyed	Unknown	
4	Accident	1979-08-02	Canton, OH	United States	Unknown	Unknown	Unknown	Fatal(1)	Destroyed	Unknown	
...	...	...	...	...	...	...	...	...	...	...	
88884	Accident	2022-12-26	Annapolis, MD	United States	Unknown	Unknown	Unknown	Minor	Unknown	Unknown	
88885	Accident	2022-12-26	Hampton, NH	United States	Unknown	Unknown	Unknown	Unknown	Unknown	Unknown	
88886	Accident	2022-12-26	Payson, AZ	United States	341525N	1112021W	PAYSON	Non-Fatal	Substantial	Airplane	
88887	Accident	2022-12-26	Morgan, UT	United States	Unknown	Unknown	Unknown	Unknown	Unknown	Unknown	
88888	Accident	2022-12-29	Athens, GA	United States	Unknown	Unknown	Unknown	Minor	Unknown	Unknown	

88889 rows × 28 columns

The result will confirm that all the missing values have been rectified

2.4 Checking for duplicates

We can check for the duplicated values by running the code below:

In [16]:
```
#We will use .duplicated().sum()
df1.duplicated().sum()
```

Out[16]:
```
0
```

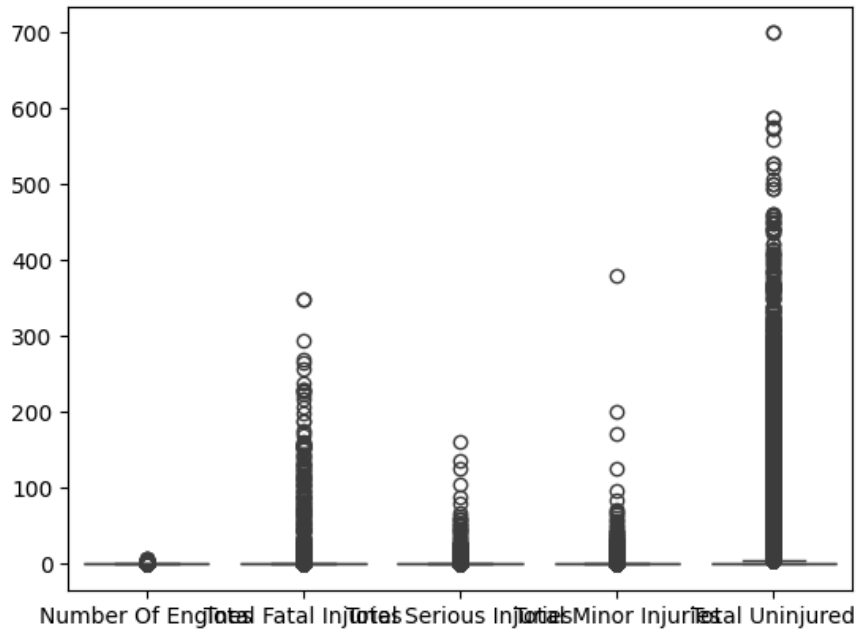
The result is (0) meaning the dataset is free of any duplicated value.

2.5 Checking for outliers

To check for outliers will start by plotting a boxplot

In [17]:
```
#we will use the seaborn library to plot
sns.boxplot(df1)
```

Out[17]: <Axes: >



It is evident that the numeric values have a few outliers we will Inter-quantile Range(IQR) to remove the outliers

In [18]:
```
# Select only numeric columns from the dataframe
numeric_columns = df1.select_dtypes(include=['number']).columns

# Calculate Q1, Q3, and IQR for numeric columns
Q1 = df1[numeric_columns].quantile(0.25)
Q3 = df1[numeric_columns].quantile(0.75)
IQR = Q3 - Q1

# Define the lower and upper bounds for outlier removal
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

# Filter the data by keeping only values within the IQR bounds
df_no_outliers = df1[(df1[numeric_columns] >= lower_bound) & (df1[numeric_columns] <= upper_bound)]

df_no_outliers = df_no_outliers.dropna()
```

In [19]:
```
df1
```

Out[19]:

	Investigation Type	Event Date	Location	Country	Latitude	Longitude	Airport Name	Injury Severity	Aircraft Damage	Aircraft Category
0	Accident	1948-10-24	MOOSE CREEK, ID	United States	Unknown	Unknown	Unknown	Fatal(2)	Destroyed	Unknown
1	Accident	1962-07-19	BRIDGEPORT, CA	United States	Unknown	Unknown	Unknown	Fatal(4)	Destroyed	Unknown
2	Accident	1974-08-30	Saltville, VA	United States	36.922223	-81.878056	Unknown	Fatal(3)	Destroyed	Unknown
3	Accident	1977-06-19	EUREKA, CA	United States	Unknown	Unknown	Unknown	Fatal(2)	Destroyed	Unknown
4	Accident	1979-08-02	Canton, OH	United States	Unknown	Unknown	Unknown	Fatal(1)	Destroyed	Unknown

2.6 Cleaning each column being used for analysis:

Here I will be creating uniformity for each non-numeric column I will be using for my analysis to make analysis less complicated. For instance some objects in the same column are in upper case while others in lower case, hence I will correct that

In [20]:
```
#checking uniformity in location
df1["Location"].value_counts()
```

Out[20]:
```
Location
ANCHORAGE, AK       434
MIAMI, FL           200
ALBUQUERQUE, NM     196
HOUSTON, TX         193
CHICAGO, IL         184
                   ...
MALLARDS LDG, GA      1
LODGEPOLE, MT         1
VERNILLION, SD        1
MCMECHEN, WV          1
Brasnorte,            1
Name: count, Length: 27758, dtype: int64
```

The last location is missing its state name compared to the other location hence I will replace its value with Unknown

In [21]:
```
#Replacing the location value
df1["Location"]=df1["Location"].replace("Brasnorte,","Unknown",regex=True)
#regex is a regular expression used to replace texts
```

In [22]:
```
#To confirm that all the changes have been made
df1["Location"].value_counts()
```

Out[22]:
```
Location
ANCHORAGE, AK       434
MIAMI, FL           200
ALBUQUERQUE, NM     196
HOUSTON, TX         193
CHICAGO, IL         184
                   ...
MALLARDS LDG, GA      1
LODGEPOLE, MT         1
VERNILLION, SD        1
MCMECHEN, WV          1
Unknown               1
Name: count, Length: 27758, dtype: int64
```

For uniformity I will change the cases to title form

In [23]:
```
#Changing the cases to title
df1["Location"]=df1["Location"].str.title()
```

In [24]:
```
#Confirming the changes
df1["Location"].value_counts()
```

Out[24]:
```
Location
Anchorage, Ak       548
Miami, Fl           275
Houston, Tx         271
Albuquerque, Nm     265
Chicago, Il         256
                   ...
Medina, Mn            1
Circle Pines, Mn      1
Pine Island, Fl       1
Churchtown, Oh        1
Unknown               1
Name: count, Length: 21978, dtype: int64
```

In [25]:
```
#Checking for uniformity in the country column we run the code below
df1["Country"].value_counts()
```

Out[25]:
```
Country
United States                       82248
Brazil                                374
Canada                                359
Mexico                                358
United Kingdom                        344
                                    ...
Saint Vincent and the Grenadines        1
Cambodia                                1
Malampa                                 1
AY                                      1
Turks and Caicos Islands                1
Name: count, Length: 219, dtype: int64
```

We notice that the country AY was wrongly input or is incomplete hence we replace with unknown

In [26]:
```
#Replacing AY with Unknown
df1["Country"]=df1["Country"].replace("AY","Unknown",regex=True)
```

In [27]:
```
#Confirming the changes
df1["Country"].value_counts()
```

Out[27]:
```
Country
United States                       82248
Brazil                                374
Canada                                359
Mexico                                358
United Kingdom                        344
                                    ...
Saint Vincent and the Grenadines        1
Cambodia                                1
Malampa                                 1
Belarus                                 1
Turks and Caicos Islands                1
Name: count, Length: 218, dtype: int64
```

In [28]:
```
#Checking uniformity for Airport Name
df1["Airport Name"].value_counts()
```

Out[28]:
```
Airport Name
Unknown                           36192
Private                             240
PRIVATE                             224
Private Airstrip                    153
NONE                                146
                                  ...
GEORGE CAMPERT MEMORIAL               1
WESTCHESTER COUNTY ARPT               1
IL VALLEY PARACHUTE CLUB              1
LAUGHLIN/BULLHEAD                     1
WICHITA DWIGHT D EISENHOWER NT        1
Name: count, Length: 24870, dtype: int64
```

As mentioned earlier, we notice there is a repetition of the word private in upper case and title case. We will change this to title case and the Private Airstrip to Private Strip to Private as well.

In [29]:
```
#Changing the texts to title case
df1["Airport Name"]=df1["Airport Name"].str.title()
```

In [30]:
```
#Confirming the changes
df1["Airport Name"].value_counts()
```

Out[30]:
```
Airport Name
Unknown                           36261
Private                             471
Private Airstrip                    266
Private Strip                       161
None                                146
                                  ...
Starbuck Muni                         1
Southwest Airpark                     1
Columbia Downtown                     1
Laama[A;O                             1
Wichita Dwight D Eisenhower Nt        1
Name: count, Length: 21566, dtype: int64
```

In [31]:
```
#Changing Private Strip, Private Airstrip to just private
df1["Airport Name"]=df1["Airport Name"].replace(["Private Strip","Private Airstrip"],"Private",regex=True)
```

In [32]:
```
#Changing None to Unknown
df1["Airport Name"]=df1["Airport Name"].replace("None","Unknown",regex=True)
```

In [33]:
```
#Confirming the changes
df1["Airport Name"].value_counts()
```

Out[33]:
```
Airport Name
Unknown                           36407
Private                             898
Merrill Field                       109
Centennial                          102
Van Nuys                             97
                                  ...
Roy'S Strip                           1
Greater Wilmingon                     1
Bell Helicopter Training              1
Stinson Fld.                          1
Wichita Dwight D Eisenhower Nt        1
Name: count, Length: 21563, dtype: int64
```

In [34]:
```
#Checking for uniformity in Injury Severity
df1["Injury Severity"].value_counts()
```

Out[34]:
```
Injury Severity
Non-Fatal     67357
Fatal(1)       6167
Fatal          5262
Fatal(2)       3711
Incident       2219
               ...
Fatal(80)         1
Fatal(217)        1
Fatal(169)        1
Fatal(88)         1
Fatal(189)        1
Name: count, Length: 110, dtype: int64
```

In [35]:
```
df1['Injury Severity'] = df1['Injury Severity'].str.replace(r'\(.*\)', '', regex=True)
```

In [36]:
```
df1["Injury Severity"].value_counts()
```

Out[36]:
```
Injury Severity
Non-Fatal      67357
Fatal          17826
Incident        2219
Unknown         1000
Minor            218
Serious          173
Unavailable       96
Name: count, dtype: int64
```

In [37]:
```
#Checking for any uniformity in this category
df1["Aircraft Damage"].value_counts()
```

Out[37]:
```
Aircraft Damage
Substantial    64148
Destroyed      18623
Unknown         3313
Minor           2805
Name: count, dtype: int64
```

Uniformity is okay

In [38]:
```
#Checking for uniformity
df1['Make'].value_counts()
```

Out[38]:
```
Make
Cessna             22227
Piper              12029
CESSNA              4922
Beech               4330
PIPER               2841
                   ...
Leonard Walters        1
Maule Air Inc.         1
Motley Vans            1
Perlick                1
ROYSE RALPH L          1
Name: count, Length: 8237, dtype: int64
```

We notice some texts are in upper while others in title case. I will change the texts to title case

In [39]:
```
#Changing all the texts to title case
df1["Make"]=df1["Make"].str.title()
```

In [40]:
```
#Confirming the changes
df1['Make'].value_counts()
```

Out[40]:
```
Make
Cessna           27149
Piper            14870
Beech             5372
Boeing            2745
Bell              2722
                 ...
Cohen                1
Kitchens             1
Lutes                1
Izatt                1
Royse Ralph L        1
Name: count, Length: 7587, dtype: int64
```

In [41]:
```
#Confirm uniformity
df1["Model"].value_counts()
```

Out[41]:
```
Model
152              2367
172              1756
172N             1164
PA-28-140         932
150               829
                 ...
GC-1-A              1
737-3S3             1
MBB-BK117-B2        1
GLASSAIR GL25       1
M-8 EAGLE           1
Name: count, Length: 12318, dtype: int64
```

No changes are required

In [42]:
```
#Checking for uniformity
df1["Engine Type"].value_counts()
```

Out[42]:
```
Engine Type
Reciprocating      69530
Unknown             9147
Turbo Shaft         3609
Turbo Prop          3391
Turbo Fan           2481
Turbo Jet            703
Geared Turbofan       12
Electric              10
LR                     2
NONE                   2
Hybrid Rocket          1
UNK                    1
Name: count, dtype: int64
```

We notice some values are in upper case while others in title case. I will change all texts to title case. I will also replace none with unknown

In [43]:
```
#Changing the texts to title case
df1["Engine Type"]=df1["Engine Type"].str.title()
```

In [44]:
```
#Changing the text None to Unknown
df1["Engine Type"]=df1["Engine Type"].replace("None","Unknown",regex=True)
```

In [45]:
```
#Confirming the changes
df1["Engine Type"].value_counts()
```

Out[45]:
```
Engine Type
Reciprocating      69530
Unknown             9149
Turbo Shaft         3609
Turbo Prop          3391
Turbo Fan           2481
Turbo Jet            703
Geared Turbofan       12
Electric              10
Lr                     2
Hybrid Rocket          1
Unk                    1
Name: count, dtype: int64
```

In [46]:
```
#Confirming the uniformity
df1["Purpose Of Flight"].value_counts()
```

Out[46]:
```
Purpose Of Flight
Personal                  49448
Unknown                   12994
Instructional             10601
Aerial Application         4712
Business                   4018
Positioning                1646
Other Work Use             1264
Ferry                       812
Aerial Observation          794
Public Aircraft             720
Executive/corporate         553
Flight Test                 405
Skydiving                   182
External Load               123
Public Aircraft - Federal   105
Banner Tow                  101
Air Race show                99
Public Aircraft - Local      74
Public Aircraft - State      64
Air Race/show                59
Glider Tow                   53
Firefighting                 40
Air Drop                     11
ASHO                          6
PUBS                          4
PUBL                          1
Name: count, dtype: int64
```

We will change all cases to title case.

In [47]:
```
#Changing the cases to title
df1["Purpose Of Flight"]=df1["Purpose Of Flight"].str.title()
```

In [48]:
```
#Confirming the changes
df1["Purpose Of Flight"].value_counts()
```

Out[48]:
```
Purpose Of Flight
Personal                  49448
Unknown                   12994
Instructional             10601
Aerial Application         4712
Business                   4018
Positioning                1646
Other Work Use             1264
Ferry                       812
Aerial Observation          794
Public Aircraft             720
Executive/Corporate         553
Flight Test                 405
Skydiving                   182
External Load               123
Public Aircraft - Federal   105
Banner Tow                  101
Air Race Show                99
Public Aircraft - Local      74
Public Aircraft - State      64
Air Race/Show                59
Glider Tow                   53
Firefighting                 40
Air Drop                     11
Asho                          6
Pubs                          4
Publ                          1
Name: count, dtype: int64
```

In [49]:
```
#Checking for uniformity
df1["Weather Condition"].value_counts()
```

Out[49]:
```
Weather Condition
VMC        77303
IMC         5976
Unknown     4492
UNK          856
Unk          262
Name: count, dtype: int64
```

Most cases are in upper cases perhaps they are initials because if they were to be written in full it would have bombarred the data set hence I will change to upper for all values for uniformity during analysis when dealing with this category.

In [50]:
```
#Changing all cases to uppercase
df1["Weather Condition"]=df1["Weather Condition"].str.upper()
```

In [51]:
```
#Confirming the changes
df1["Weather Condition"].value_counts()
```

Out[51]:
```
Weather Condition
VMC        77303
IMC         5976
UNKNOWN     4492
UNK         1118
Name: count, dtype: int64
```

In [52]:
```
#Checking for uniformity in this category
df1["Air Carrier"].value_counts()
```

Out[52]:
```
Air Carrier
Unknown                      72255
Pilot                          258
American Airlines               90
United Airlines                 89
Delta Air Lines                 53
                               ...
WOODY CONTRACTING INC            1
Rod Aviation LLC                 1
Paul D Franzon                   1
TRAINING SERVICES INC DBA        1
MC CESSNA 210N LLC               1
Name: count, Length: 13590, dtype: int64
```

We will change it to title case for uniformity

In [53]:
```
#Changing to title case
df1["Air Carrier"]=df1["Air Carrier"].str.title()
```

In [54]:
```
#Confirming the changes
df1["Air Carrier"].value_counts()
```

Out[54]:
```
Air Carrier
Unknown                      72258
Pilot                          258
American Airlines               90
United Airlines                 89
Delta Air Lines                 53
                               ...
Richard L. Mcglashan             1
Inflight Pilot Traning Llc       1
Mills & Daughters Inc            1
Beery Douglas W                  1
Mc Cessna 210N Llc               1
Name: count, Length: 13208, dtype: int64
```

Since most of the air carriers are unknown I will change the unknown values to "other carriers" for easier interpretation.

In [55]:
```
#Changing unknown to other carriers
df1["Air Carrier"]=df1["Air Carrier"].replace("Unknown","Other Carriers")
```

In [56]:
```
#Checking for uniformity
df1["Event Date"].value_counts()
```

Out[56]:
```
Event Date
1984-06-30    25
1982-05-16    25
2000-07-08    25
1983-08-05    24
1984-08-25    24
              ..
2014-03-16     1
2014-03-15     1
2014-03-12     1
2014-03-10     1
2022-12-29     1
Name: count, Length: 14782, dtype: int64
```

In [57]:
```
#Checking for uniformity
df1["Investigation Type"].value_counts()
```

Out[57]:
```
Investigation Type
Accident    85015
Incident     3874
Name: count, dtype: int64
```

In [58]:
```
#Checking for uniormity on this column
df1["Latitude"].value_counts()
```

Out[58]:
```
Latitude
Unknown      54507
332739N         19
335219N         18
32.815556       17
334118N         17
              ...
345832N          1
31.991666        1
444947N          1
034358N          1
373829N          1
Name: count, Length: 25593, dtype: int64
```

Since latitudes deal with numeric values I will change the unknown values to zero. I will apply the same changes for the longitude column

In [59]:
```
#Changing unknown to zero for latitude
df1["Latitude"]=df1["Latitude"].replace("Unknown",0,regex=True)
```

In [60]:
```
#Changing unknown to zero for longitude
df1["Longitude"]=df1["Longitude"].replace("Unknown",0,regex=True)
```

In [61]:
```
#Checking for uniformity in aircraft category
df1["Aircraft Category"].value_counts().head(10)
```

Out[61]:
```
Aircraft Category
Unknown              56616
Airplane             27617
Helicopter            3440
Glider                 508
Balloon                231
Gyrocraft              173
Weight-Shift           161
Powered Parachute       91
Ultralight              30
WSFT                     9
Name: count, dtype: int64
```

Since most values are unknown I will change it to "other" assuming the unknown values were categories that were not entered to the dataset.

In [62]:
```
#Changing from unknown to other
df1["Aircraft Category"]=df1["Aircraft Category"].replace("Unknown","Other")
```

In [63]:
```
#Confirming the changes
df1.columns
```

Out[63]:
```
Index(['Investigation Type', 'Event Date', 'Location', 'Country', 'Latitude',
       'Longitude', '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 [64]:
```
#Adding Total Injuries Column
df1["Total Injuries"]=df1["Total Fatal Injuries"]+df1["Total Minor Injuries"]+df1["Total Serious Injuries"
```

In [65]:
```
df1.columns
```

Out[65]:
```
Index(['Investigation Type', 'Event Date', 'Location', 'Country', 'Latitude',
       'Longitude', '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', 'Total Injuries'],
      dtype='object')
```

After cleaning the dataset we save the clean dataset

In [66]:
```
#Saving the clean and updated dataset
df1.to_csv("updated clean Aviation.csv")
```

In [67]:
```
df1.columns
```

Out[67]:
```
Index(['Investigation Type', 'Event Date', 'Location', 'Country', 'Latitude',
       'Longitude', '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', 'Total Injuries'],
      dtype='object')
```

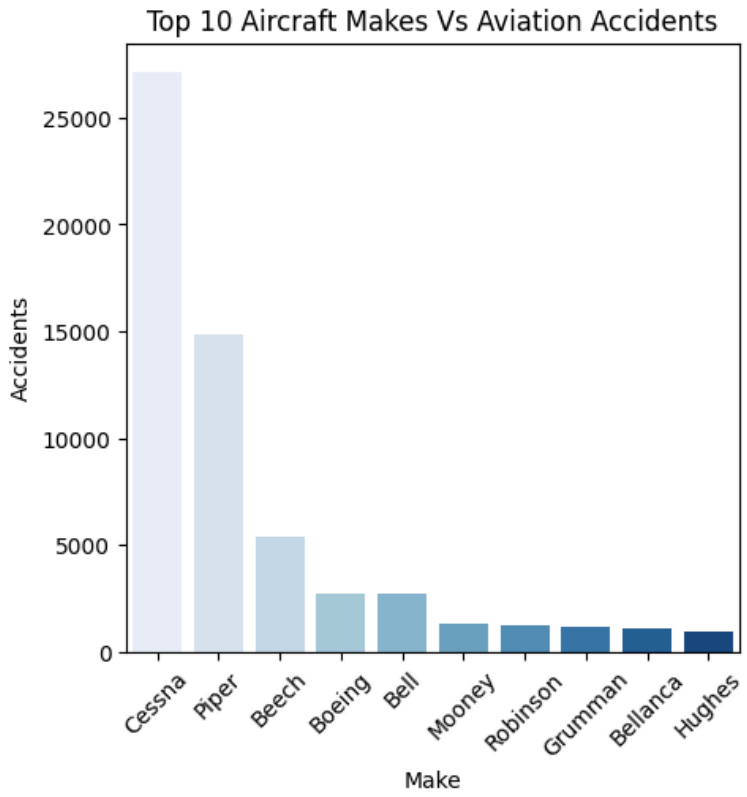
3 Data Preparation and Analysis

3.0.1 Analysis based on aircraft make

I will start my analysis based on the Aircraft by making a visualization that will help me determine aircrafts makes with highest number of accidents. To easen my analysis and make visualization interpretable I will select the top 10 makes and make a visualization based on them by running the code below:

In [68]:
```
#The code below will plot and display a visualization based on top 10 Aircraft makes with the most acciden
make_counts = df1['Make'].value_counts().head(10)#Selecting the top 10 makes
plt.figure(figsize=(5,5 ))#Selecting the figure size of the visualiztion
sns.barplot(x=make_counts.index, y=make_counts.values,palette="Blues")#Selecting the x-axis and y-axis for
plt.title('Top 10 Aircraft Makes Vs Aviation Accidents')#Selecting the title of the visualization
plt.xlabel('Make')#Selecting the x-axis title
plt.ylabel('Accidents')#Selecting the y-axis title
plt.xticks(rotation=45)#selecting the label format for each category on the x-axis(This to avoid overlappi
plt.show()#plots and visualizes the image
```

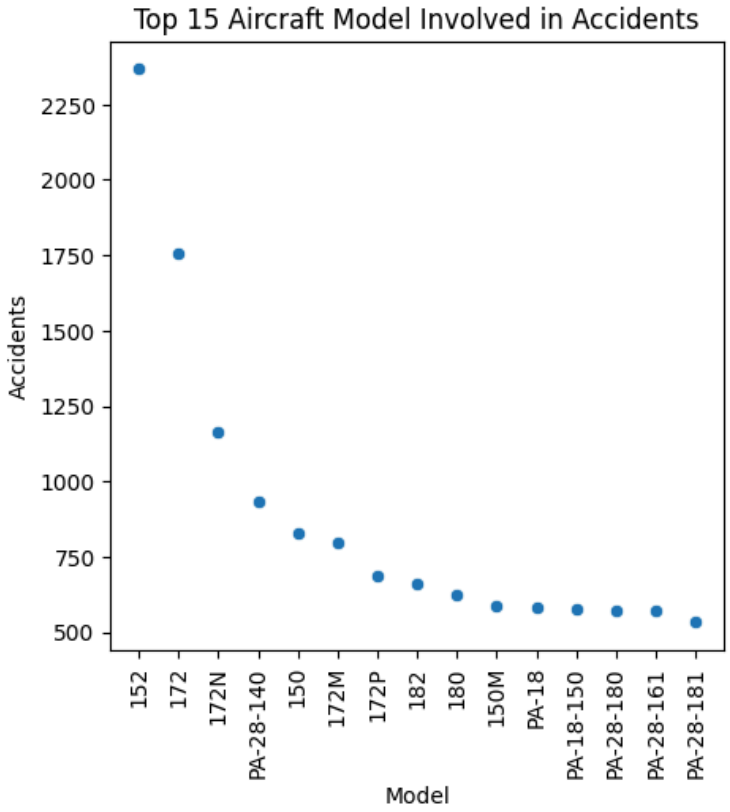
```
C:\Users\123\AppData\Local\Temp\ipykernel_23448\860493965.py:4: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` va
riable to `hue` and set `legend=False` for the same effect.

  sns.barplot(x=make_counts.index, y=make_counts.values,palette="Blues")#Selecting the x-axis and y-axis
for a bargraph
```



3.0.2 Analysis based on model

We can further our analysis by visualizing models of aircrafts and checking how prone they are to aircraft accidents. I wrote a code based on top fifteen models that have recorded the most aircraft accidents. I did so by developing the code below to make the appropriate visualizations for analysis

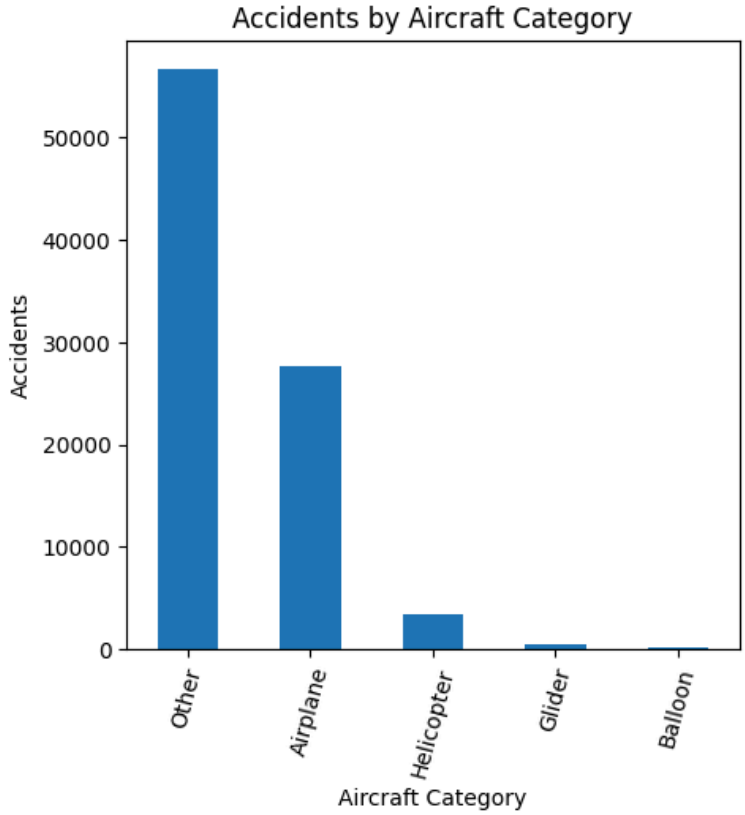
In [69]:
```
#The code below will plot and display a visualization based on top 10 Aircraft makes with the most acciden
model_counts = df1['Model'].value_counts().head(15)#Selecting the top 15 models
plt.figure(figsize=(5,5))#Selecting the figure size for the visualization
sns.scatterplot(x=model_counts.index, y=model_counts.values)#Selecting the x-axis and y-axis for my scatte
plt.title('Top 15 Aircraft Model Involved in Accidents')#Title for the visualization
plt.xlabel('Model')#Title for the x-axis
plt.ylabel('Accidents')#Title for the y-axis
plt.xticks(rotation=90)##selecting the label format for each category on the x-axis(This to avoid overlapp
plt.show()#plots and visualizes the image
```



3.0.3 Analysis based on aircraft category:

We can further our analysis by visualizating which category of aircrafts have the most aviation accident by following the procedures below:

In [70]:
```
# The code below will plot and display a visualization based on Aircraft category with the most accident
df1['Aircraft Category'].value_counts().head(5).plot(kind='bar', figsize=(5, 5))#An alternative direct cod
plt.title('Accidents by Aircraft Category')#title of the visualization
plt.ylabel('Accidents')#y axis label
plt.xlabel('Aircraft Category')#x axis label
plt.xticks(rotation=75)#Selects the label format for labels on the x-axis
plt.show()#Image display
```

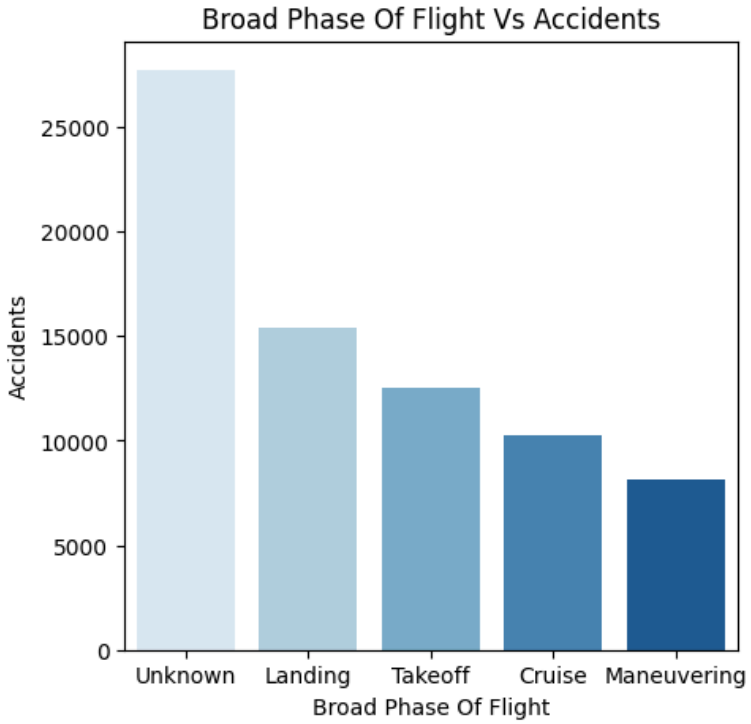


3.0.4 Analysis based on Broad Phase of Flight

Analyzing using Broad Phase of Flight is of great importance for our analysis to be familiar under what phase do accidents occur is it during take-off, approach or landing. We can see when, by plotting our visualization below:

In [71]:
```
#Code for analysis for top 5 broad of flights where accidents occur
phase_flight=df1["Broad Phase Of Flight"].value_counts().head(5)#selecting the top five
plt.figure(figsize=(5,5))#Selecting the figure size
sns.barplot(x=phase_flight.index, y=phase_flight.values,palette="Blues")#plotting the bargraph
plt.title("Broad Phase Of Flight Vs Accidents")#title for the visualization
plt.xlabel("Broad Phase Of Flight")#x-axis title
plt.ylabel("Accidents")#y-axis title
plt.show()#isualization
```

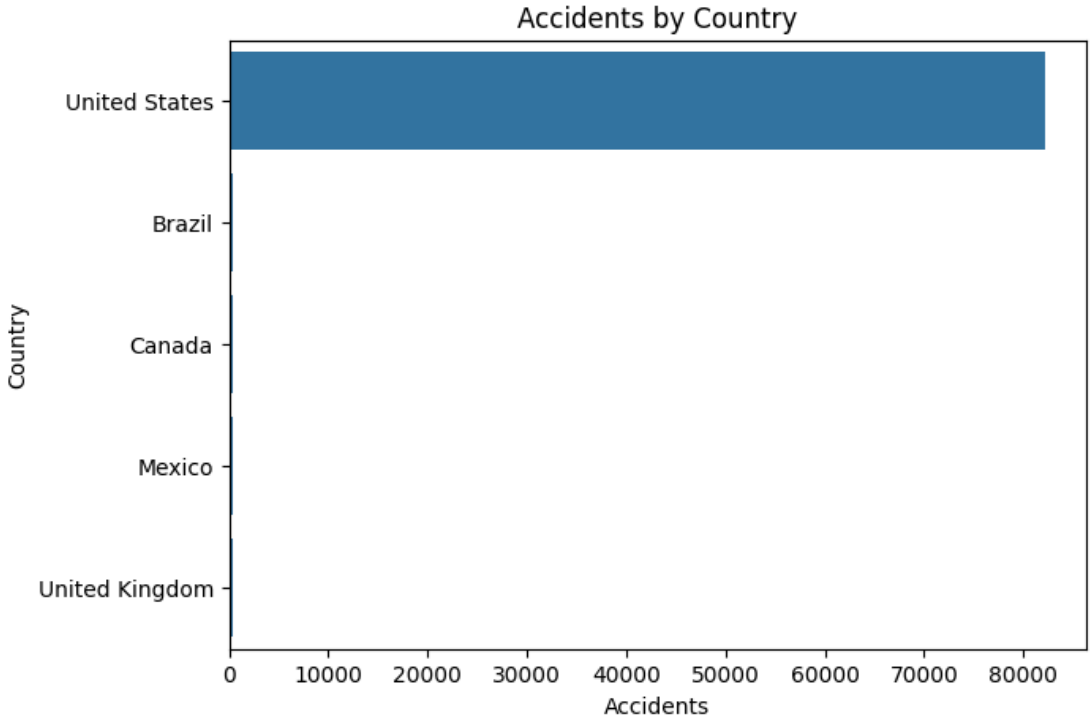
```
C:\Users\123\AppData\Local\Temp\ipykernel_23448\1229906056.py:4: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` va
riable to `hue` and set `legend=False` for the same effect.

  sns.barplot(x=phase_flight.index, y=phase_flight.values,palette="Blues")#plotting the bargraph
```



3.0.5 Analysis based on countries:

We can further our analysis by checking which top five countries has the most number of accidendt as recorded by the NTSB.

In [72]:
```
#Accidents by the top five countries
country_accidents = df1['Country'].value_counts().head(5)#Selecting top 5 countries with most number of ac
plt.figure(figsize=(7,5))#inputing the figure size
sns.barplot(x=country_accidents.values, y=country_accidents.index, orient='h')#plotting the barplot assigi
plt.title('Accidents by Country')#title
plt.xlabel('Accidents')# x axis label
plt.ylabel('Country');# y axis label
```

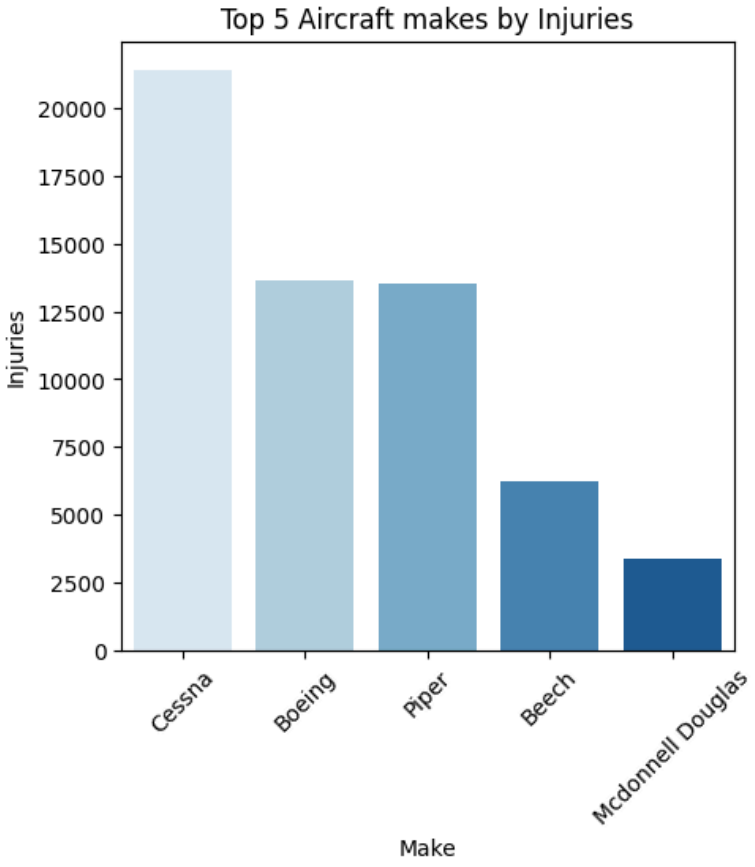


3.0.6 Analysis based on Make and total injuries

The visualization below will indicate which top 5 makes have recorded the most injuries.

In [73]:
```
#Group top 5 makes with most number of injuries as shown below
total_injuries = df1.groupby('Make')['Total Injuries'].sum().sort_values(ascending=False).head(5)#grouping
plt.figure(figsize=(5,5))#selecting the figure size
sns.barplot(x=total_injuries.index, y=total_injuries.values,palette="Blues")#plotting the barplot appying
plt.title('Top 5 Aircraft makes by Injuries')#Title for the visualizatio
plt.xlabel('Make')#x-axis title
plt.ylabel('Injuries')#y-axis title
plt.xticks(rotation=45)#label format for the x-axis
plt.show()#display the visualization
```

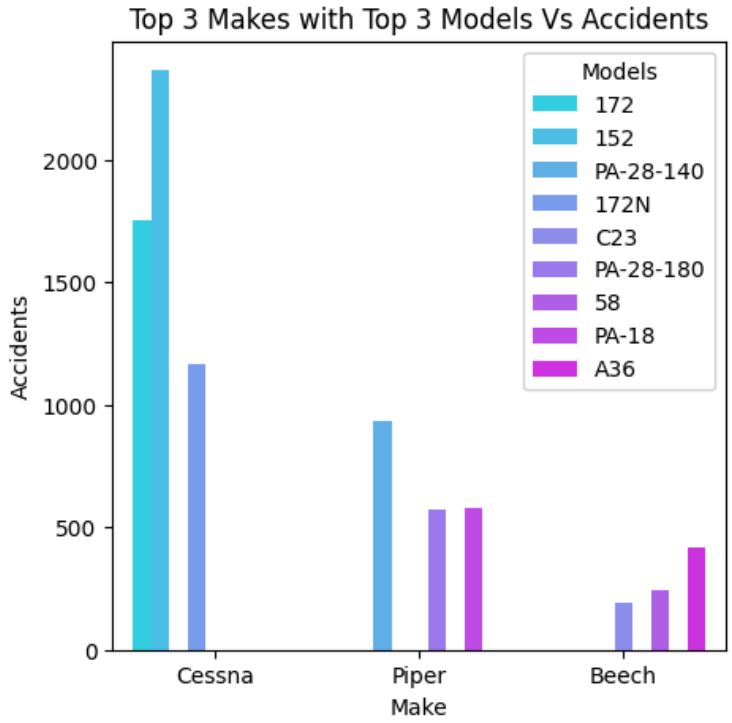
```
C:\Users\123\AppData\Local\Temp\ipykernel_23448\839726315.py:4: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` va
riable to `hue` and set `legend=False` for the same effect.

  sns.barplot(x=total_injuries.index, y=total_injuries.values,palette="Blues")#plotting the barplot appyi
ng the approriate color
```



3.0.7 Analysis based on make and mode vs the accidents

I went further with my analysis by selecting the top three models and makes that have recorded the most accidents

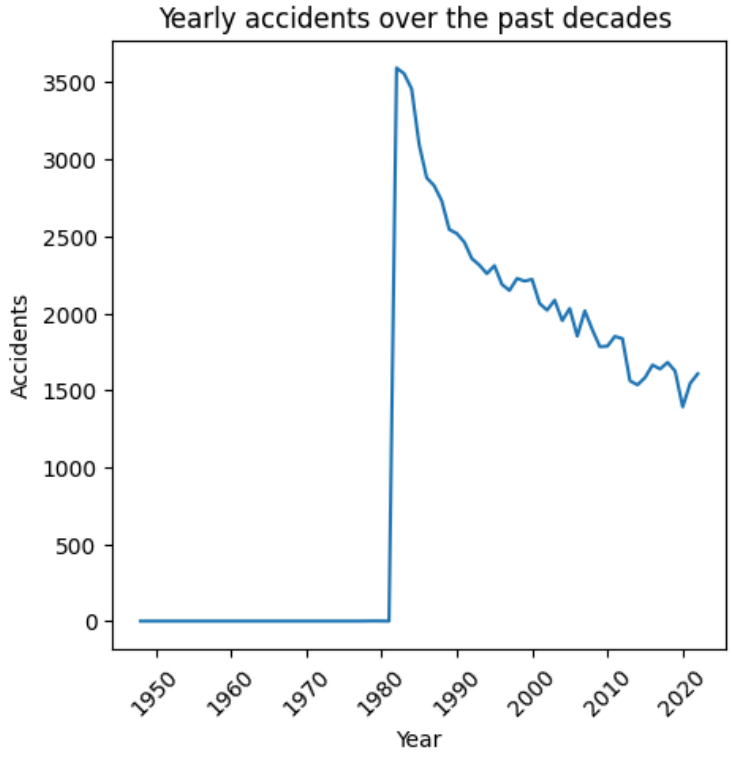
In [74]:
```
# Get top 3 makes
top_three_makes = df1['Make'].value_counts().head(3).index

# Filter data for top 3 makes
filtered_data = df1[df1['Make'].isin(top_three_makes)]

# top 3 models for each make
top_models_per_make = (
    filtered_data.groupby('Make')['Model']
    .value_counts()
    .groupby(level=0).nlargest(3)
    .reset_index(level=0, drop=True)
    .index.get_level_values(1)
)#We groupby the data and select the top three value makes by using the nlargest( method) and retrieving t

# Filter data for top 3 models
filtered_data = filtered_data[filtered_data['Model'].isin(top_models_per_make)]

# Plotting the visualization
plt.figure(figsize=(5, 5))#select figure size
sns.countplot(data=filtered_data, x='Make', hue='Model',palette="cool")#selecting type of graph(countplot)
plt.title('Top 3 Makes with Top 3 Models Vs Accidents')#title for the graph
plt.xlabel('Make')#x-axis label
plt.ylabel('Accidents')#y-axis label
plt.legend(title='Models')#legend title
plt.show()#visualize the graph
```



3.0.8 Analysing accidents yearly

We can further do analysis and analyize how accidents have been fairing on over the past decades

In [75]:
```
#Accidents occuring yearly
df1['Year'] = pd.to_datetime(df1['Event Date'].str.strip()).dt.year#selecting the dates based on the years
yearly_accidents = df1['Year'].value_counts().sort_index()#sorting in index form
plt.figure(figsize=(5, 5))#Selecting the figure size
sns.lineplot(x=yearly_accidents.index, y=yearly_accidents.values)#plotting a line plot
plt.title('Yearly accidents over the past decades ')#title
plt.xlabel('Year')#x-axis label
plt.ylabel('Accidents')#y-axis label
plt.xticks(rotation=45);
```



In [ ]: