Phase1 Project: Aviation Accidents Data Analysis and Insights

Overview

The project seeks to explore data on different aircrafts accidents in order to advise company stakeholders on which type of aircraft to choose for business as they would like to join this industry. The dataset⇒, from the National Transportation Safety Board that includes aviation accident data and selected incidents covering United States, its territories and international waters from 1962 and later.

Business

The company is seeking to start a new business endeavor, operating aircrafts for commercial and private enterprises, in which they are novice to it. The stakeholders are seeking to understand what type of aircrafts have low risk to accidents from prevoius occurences.

The dataset has various incidents since 1962. The information on the database is on continual update once an incident happens. Using this data would help identify the patterns/trends on these occurences.

At the end of the data analysis, getting aircraft types which are low-risk we need to answer questions such as:

```
what factors should we consider for low risk. This may include looking
at the number of injuries, level of damage, etc.

What make have minimal or no accidents, which models specifically?

what level of damage did they acquire during those incidences?
```

This would aid in making decisions on which type of aircrafts to purchase.

Data

Let's check what the data is about. We will use python methods and its libraries to explore the data.

```python
# Importing Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

```python
#Open and reeding the datasets
aviation_df = pd.read_csv("AviationData.csv", encoding='cp1252',
engine='python')# or use 'low_memory=False'
usstate_df = pd.read_csv("USState_Codes.csv", encoding='cp1252')

#get top 5 rows of aviation data
aviation_df.head()
```

```
         Event.Id Investigation.Type Accident.Number  Event.Date  \
0  20001218X45444           Accident      SEA87LA080  1948-10-24
1  20001218X45447           Accident      LAX94LA336  1962-07-19
2  20061025X01555           Accident      NYC07LA005  1974-08-30
3  20001218X45448           Accident      LAX96LA321  1977-06-19
4  20041105X01764           Accident      CHI79FA064  1979-08-02

          Location        Country   Latitude   Longitude Airport.Code
\
0  MOOSE CREEK, ID  United States        NaN         NaN          NaN

1   BRIDGEPORT, CA  United States        NaN         NaN          NaN

2    Saltville, VA  United States  36.922223  -81.878056          NaN

3        EUREKA, CA  United States        NaN         NaN          NaN

4        Canton, OH  United States        NaN         NaN          NaN


  Airport.Name  ... Purpose.of.flight Air.carrier Total.Fatal.Injuries
\
0          NaN  ...          Personal         NaN                  2.0

1          NaN  ...          Personal         NaN                  4.0

2          NaN  ...          Personal         NaN                  3.0

3          NaN  ...          Personal         NaN                  2.0

4          NaN  ...          Personal         NaN                  1.0


  Total.Serious.Injuries Total.Minor.Injuries Total.Uninjured  \
0                    0.0                  0.0             0.0
1                    0.0                  0.0             0.0
2                    NaN                  NaN             NaN
3                    0.0                  0.0             0.0
4                    2.0                  NaN             0.0

  Weather.Condition  Broad.phase.of.flight   Report.Status
Publication.Date
0               UNK                 Cruise  Probable Cause
```

```
NaN
1                UNK                Unknown  Probable Cause       19-
09-1996
2                IMC                 Cruise  Probable Cause       26-
02-2007
3                IMC                 Cruise  Probable Cause       12-
09-2000
4                VMC               Approach  Probable Cause       16-
04-1980

[5 rows x 31 columns]
```

From the first five records, we see that there are some null values.

```
#check the last five records
aviation_df.tail()

             Event.Id Investigation.Type Accident.Number
Event.Date  \
88884  20221227106491           Accident      ERA23LA093  2022-12-26

88885  20221227106494           Accident      ERA23LA095  2022-12-26

88886  20221227106497           Accident      WPR23LA075  2022-12-26

88887  20221227106498           Accident      WPR23LA076  2022-12-26

88888  20221230106513           Accident      ERA23LA097  2022-12-29


            Location        Country Latitude Longitude Airport.Code  \
88884  Annapolis, MD  United States      NaN       NaN          NaN
88885    Hampton, NH  United States      NaN       NaN          NaN
88886     Payson, AZ  United States  341525N  1112021W          PAN
88887     Morgan, UT  United States      NaN       NaN          NaN
88888     Athens, GA  United States      NaN       NaN          NaN

      Airport.Name  ... Purpose.of.flight        Air.carrier  \
88884          NaN  ...          Personal                NaN
88885          NaN  ...               NaN                NaN
88886       PAYSON  ...          Personal                NaN
88887          NaN  ...          Personal  MC CESSNA 210N LLC
88888          NaN  ...          Personal                NaN

      Total.Fatal.Injuries Total.Serious.Injuries Total.Minor.Injuries
\
88884                  0.0                    1.0                  0.0

88885                  0.0                    0.0                  0.0

```

```
88886                  0.0                      0.0                     0.0

88887                  0.0                      0.0                     0.0

88888                  0.0                      1.0                     0.0


      Total.Uninjured Weather.Condition  Broad.phase.of.flight
Report.Status  \
88884              0.0               NaN                    NaN
NaN
88885              0.0               NaN                    NaN
NaN
88886              1.0               VMC                    NaN
NaN
88887              0.0               NaN                    NaN
NaN
88888              1.0               NaN                    NaN
NaN

      Publication.Date
88884       29-12-2022
88885              NaN
88886       27-12-2022
88887              NaN
88888       30-12-2022

[5 rows x 31 columns]
```

```
# Get the shape of the data, rows and columns
aviation_df.shape
```

```
(88889, 31)
```

```
#get information about the data, including the datatypes of respective
columns
aviation_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            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
```

```
aviation_df.describe()
```

```
       Number.of.Engines  Total.Fatal.Injuries  Total.Serious.Injuries  
\
count       82805.000000          77488.000000            76379.000000   

mean            1.146585              0.647855                0.279881   

std             0.446510              5.485960                1.544084   

min             0.000000              0.000000                0.000000   

25%             1.000000              0.000000                0.000000   

50%             1.000000              0.000000                0.000000   

75%             1.000000              0.000000                0.000000   

max             8.000000            349.000000              161.000000   


       Total.Minor.Injuries  Total.Uninjured  
count          76956.000000     82977.000000  
mean               0.357061         5.325440  
std                2.235625        27.913634  
```

```
min                 0.000000        0.000000
25%                 0.000000        0.000000
50%                 0.000000        1.000000
75%                 0.000000        2.000000
max               380.000000      699.000000
```

From the above data, we can see aviation dataset has 88,889 rows with 35 columns. It gives info on aircraft type, location, country where the incident occured, information about the aircraft, dates, etc.

There are missing data from the null values seen in the data. The dataset has numbers on injuries - fatal, seroius or minor ones, and number of engines of the aircraft.

```
##get top 5 rows of USState Codes data
usstate_df.head()
```

```
     US_State Abbreviation
0     Alabama           AL
1      Alaska           AK
2     Arizona           AZ
3    Arkansas           AR
4  California           CA
```

```
usstate_df.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 data from USState codes contains 62 rows, showing 62 states and 2 columns showing the States in the data and the respective abbreviation. There are no null values

```
usstate_df.shape
```

```
(62, 2)
```

```
#Check the columns in the aviation publish_display_data
aviation_df.columns
```

```
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')
```

```
#let us check if the data has columns that have similar content on
both aviation and usstate codes
usstate_df['US_State'].isin(aviation_df['Country']).value_counts()
```

```
False    54
True      8
Name: US_State, dtype: int64
```

```
len(aviation_df.columns)
```

```
31
```

Data Preparation

Let us prepare the data for analysis. This will/may involve handling missing values in the data, dropping unnecessary columns, selecting needed data, etc.

1. We start by checking the essential columns and dropping the unnecessary ones.

2. The company needs 'Airplane'. If you check the AircraftCategory, there are aircraft of the Airplane Category. So we will create a dataset of that category.

3. To determine the low risk aircraft, it needs specific kind of data such as the model, make, category, engine type, level of damage, number of injuries, etc. Columns such as Event Id, Investigation Type, Accident Number, etc. may not be useful. So we will drop these columns

4. The data may be having spaces at the edges, we will strip the empty spaces in case the are present.

5. Then we rename the columns to be easy handling by removing dots on column names

6. Check null values and see how to handle them, drop, replace by mode , mean or median. As noticed earlier, there are only 4 columns without null values in the aviation data i.e.:

```
 1. 'Event.Id'
 2. 'Investigation.Type'
```

```
3. 'Accident.Number'
4. 'Event.Date'
```

```
#First let us make a copy of the data
aviation_df_copy = aviation_df.copy(deep=True)
```

```
#Checking the columns of aviation data
aviation_df.columns
```

```
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')
```

```
#Check Aircraft Category
aviation_df['Aircraft.Category'].value_counts()
```

```
Airplane             27617
Helicopter            3440
Glider                 508
Balloon                231
Gyrocraft              173
Weight-Shift           161
Powered Parachute       91
Ultralight              30
Unknown                 14
WSFT                     9
Powered-Lift             5
Blimp                    4
UNK                      2
ULTR                     1
Rocket                   1
Name: Aircraft.Category, dtype: int64
```

```
#creating Airplane dataset
airplane_df = aviation_df.loc[aviation_df["Aircraft.Category"] ==
"Airplane"].reset_index(drop=True)
airplane_df.head()
```

```
         Event.Id Investigation.Type Accident.Number  Event.Date  \
0  20170710X52551           Accident      NYC79AA106  1979-09-17
1  20020909X01562           Accident      SEA82DA022  1982-01-01
2  20020909X01561           Accident      NYC82DA015  1982-01-01
3  20020917X02148           Accident      FTW82FRJ07  1982-01-02
4  20020917X02134           Accident      FTW82FRA14  1982-01-02

           Location        Country   Latitude   Longitude Airport.Code
\
0        BOSTON, MA  United States  42.445277  -70.758333          NaN

1       PULLMAN, WA  United States        NaN         NaN          NaN

2  EAST HANOVER, NJ  United States        NaN         NaN          N58

3         HOMER, LA  United States        NaN         NaN          NaN

4        HEARNE, TX  United States        NaN         NaN          T72


         Airport.Name  ... Purpose.of.flight Air.carrier
Total.Fatal.Injuries  \
0                 NaN  ...               NaN  Air Canada
NaN
1  BLACKBURN AG STRIP  ...          Personal         NaN
0.0
2             HANOVER  ...          Business         NaN
0.0
3                 NaN  ...          Personal         NaN
0.0
4    HEARNE MUNICIPAL  ...          Personal         NaN
1.0

  Total.Serious.Injuries Total.Minor.Injuries Total.Uninjured  \
0                    NaN                  1.0            44.0
1                    0.0                  0.0             2.0
2                    0.0                  0.0             2.0
3                    0.0                  1.0             0.0
4                    0.0                  0.0             0.0

  Weather.Condition  Broad.phase.of.flight   Report.Status
Publication.Date
0               VMC                  Climb  Probable Cause       19-
09-2017
1               VMC                Takeoff  Probable Cause       01-
01-1982
2               IMC                Landing  Probable Cause       01-
01-1982
3               IMC                 Cruise  Probable Cause       02-
01-1983
4               IMC                Takeoff  Probable Cause       02-
```

```
01-1983

[5 rows x 31 columns]
```

```python
#Confirming if the aircraft is  "Airplane" type only
airplane_df['Aircraft.Category'].value_counts()
```

```
Airplane    27617
Name: Aircraft.Category, dtype: int64
```

```python
#dropping columns as they will not be needed for anaylsis. Note
"Broadphaseofflight" has no much data
airplane_df.drop(["Event.Id","Investigation.Type", "Accident.Number",
"Airport.Code", "Airport.Name", "Air.carrier", "Schedule",
"Report.Status", "Publication.Date"], axis=1,inplace = True)
airplane_df.head()
```

```
   Event.Date          Location        Country   Latitude   Longitude
\
0  1979-09-17        BOSTON, MA  United States  42.445277  -70.758333

1  1982-01-01       PULLMAN, WA  United States        NaN         NaN

2  1982-01-01  EAST HANOVER, NJ  United States        NaN         NaN

3  1982-01-02         HOMER, LA  United States        NaN         NaN

4  1982-01-02        HEARNE, TX  United States        NaN         NaN


  Injury.Severity Aircraft.damage Aircraft.Category
Registration.Number  \
0       Non-Fatal     Substantial          Airplane              CF-
TLU
1       Non-Fatal     Substantial          Airplane
N2482N
2       Non-Fatal     Substantial          Airplane
N7967Q
3       Non-Fatal       Destroyed          Airplane
N14779
4        Fatal(1)       Destroyed          Airplane
N758SK

                Make  ... Number.of.Engines    Engine.Type  \
0  Mcdonnell Douglas  ...               2.0      Turbo Fan
1             Cessna  ...               1.0  Reciprocating
2             Cessna  ...               2.0  Reciprocating
3           Bellanca  ...               1.0  Reciprocating
4             Cessna  ...               1.0  Reciprocating

             FAR.Description Purpose.of.flight Total.Fatal.Injuries  \
```

```
0          Part 129: Foreign               NaN                  NaN
1  Part 91: General Aviation          Personal                  0.0
2  Part 91: General Aviation          Business                  0.0
3  Part 91: General Aviation          Personal                  0.0
4  Part 91: General Aviation          Personal                  1.0

  Total.Serious.Injuries  Total.Minor.Injuries  Total.Uninjured  \
0                    NaN                   1.0             44.0
1                    0.0                   0.0              2.0
2                    0.0                   0.0              2.0
3                    0.0                   1.0              0.0
4                    0.0                   0.0              0.0

   Weather.Condition  Broad.phase.of.flight
0                VMC                  Climb
1                VMC                Takeoff
2                IMC                Landing
3                IMC                 Cruise
4                IMC                Takeoff

[5 rows x 22 columns]
```

```
airplane_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 27617 entries, 0 to 27616
Data columns (total 22 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Event.Date              27617 non-null  object
 1   Location                27610 non-null  object
 2   Country                 27610 non-null  object
 3   Latitude                22092 non-null  object
 4   Longitude               22083 non-null  object
 5   Injury.Severity         26803 non-null  object
 6   Aircraft.damage         26335 non-null  object
 7   Aircraft.Category       27617 non-null  object
 8   Registration.Number     27391 non-null  object
 9   Make                    27608 non-null  object
 10  Model                   27586 non-null  object
 11  Amateur.Built           27600 non-null  object
 12  Number.of.Engines       24863 non-null  float64
 13  Engine.Type             23391 non-null  object
 14  FAR.Description         27118 non-null  object
 15  Purpose.of.flight       23878 non-null  object
 16  Total.Fatal.Injuries    24452 non-null  float64
 17  Total.Serious.Injuries  24393 non-null  float64
 18  Total.Minor.Injuries    24739 non-null  float64
 19  Total.Uninjured         26717 non-null  float64
 20  Weather.Condition       24564 non-null  object
```

```
 21  Broad.phase.of.flight   6408 non-null   object
dtypes: float64(5), object(17)
memory usage: 4.6+ MB
```

```
airplane_df.columns
```

```
Index(['Event.Date', 'Location', 'Country', 'Latitude', 'Longitude',
       'Injury.Severity', 'Aircraft.damage', 'Aircraft.Category',
       'Registration.Number', 'Make', 'Model', 'Amateur.Built',
       'Number.of.Engines', 'Engine.Type', 'FAR.Description',
       'Purpose.of.flight', 'Total.Fatal.Injuries',
'Total.Serious.Injuries',
       'Total.Minor.Injuries', 'Total.Uninjured', 'Weather.Condition',
       'Broad.phase.of.flight'],
      dtype='object')
```

If you check, the column names have a '.' separator between 2 names. We can remove the dot for each column name. Then we can strip of any empty spaces in the data.

```
#Let us remove the dot(.) in the column names
#First create a dictionary with key-value pairs consisting of old
names as keys and new names as keys
#create keys append them to an empty list
colmn_names_orig = []
for colmn in airplane_df.columns:
    colmn_names_orig.append(colmn)
len(colmn_names_orig)
```

```
22
```

```
#Create values
colmn_names_new = []
#remove the '.' from column names and append to an empty list
for x in range(len(airplane_df.columns)):
    new_column_name = airplane_df.columns[x].replace('.', '')
    colmn_names_new.append(new_column_name)

#type(colmn_names_new)
len(colmn_names_new)
```

```
22
```

```
#Create a dictionary using zip() method
colmn_names_dict = dict(zip(colmn_names_orig, colmn_names_new))
#colmn_names_dict

#Rename the columns using rename() method and a dictionary as an
argument
airplane_df.rename(colmn_names_dict, axis='columns', inplace=True)

airplane_df.dtypes
```

```
EventDate                    object
Location                     object
Country                      object
Latitude                     object
Longitude                    object
InjurySeverity               object
Aircraftdamage               object
AircraftCategory             object
RegistrationNumber           object
Make                         object
Model                        object
AmateurBuilt                 object
NumberofEngines             float64
EngineType                   object
FARDescription               object
Purposeofflight              object
TotalFatalInjuries          float64
TotalSeriousInjuries        float64
TotalMinorInjuries          float64
TotalUninjured              float64
WeatherCondition             object
Broadphaseofflight           object
dtype: object
```

```python
#Checking the Make and see if there is uniformity
airplane_df['Make'].value_counts()#.head(20)
```

```
CESSNA          4867
Cessna          3608
PIPER           2805
Piper           1910
BOEING          1037
                ... 
Palmer             1
Newcomer           1
Walling            1
Ralph Sanders      1
Helmetag           1
Name: Make, Length: 3874, dtype: int64
```

If you check the 'Make" for example, there are Cessna and CESSNA. The make is the same but they keyed in with different letter cases. We will make it uniform by captitalizing each or having it in small letters.

```python
#Capitalizing the Make text for uniformity
airplane_df['Make']= airplane_df['Make'].str.capitalize()

#Confirming if the change is effected
airplane_df['Make'].value_counts().head(20)
```

```
Cessna                  8475
Piper                   4715
Beech                   1692
Boeing                  1324
Mooney                   419
Bellanca                 282
Grumman                  251
Airbus                   245
Maule                    232
Aeronca                  229
Air tractor              224
Cirrus design corp       220
Air tractor inc          219
Champion                 170
Luscombe                 164
Embraer                  155
Stinson                  146
Cirrus                   137
Vans                     125
North american           118
Name: Make, dtype: int64
```

Dealing with missing values

Let us check null values on different columns

```
airplane_df.isna().sum()
```

```
EventDate                  0
Location                   7
Country                    7
Latitude                5525
Longitude               5534
InjurySeverity           814
Aircraftdamage          1282
AircraftCategory           0
RegistrationNumber       226
Make                       9
Model                     31
AmateurBuilt              17
NumberofEngines         2754
EngineType              4226
FARDescription           499
Purposeofflight         3739
TotalFatalInjuries      3165
TotalSeriousInjuries    3224
TotalMinorInjuries      2878
TotalUninjured           900
WeatherCondition        3053
```

```
Broadphaseofflight      21209
dtype: int64
```

If you check the 'Make' and the 'Model', there are missing data. Without these values, we cannot determine what aircraft to purchase. You may not fill in the data as this concerns accidents and incidents that occured and may distort the information. So, we will drop the missing values on Make amd Model.

```
airplane_df.dropna(subset=['Make','Model'], inplace=True)

airplane_df['Broadphaseofflight'].value_counts()

Landing        2253
Takeoff        1278
Cruise          838
Approach        638
Maneuvering     511
Taxi            241
Descent         168
Go-around       154
Climb           153
Standing         75
Unknown          62
Other            14
Name: Broadphaseofflight, dtype: int64
```

If you check the "BroadPhaseoflight" it has a lot of null values, '21,209'. The phases are crucial as they show what phase the incident occured and may help in risk management for the airplane. Dropping null values in this column will result in a lot of loss of data. And if you check further, there is a value of 'Unknown'. This can fill up the missing values.

```
#Fill nulls with 'Unknown' in the 'Broadphaseofflight' column.
airplane_df['Broadphaseofflight'].fillna("Unknown", inplace=True)

#confirm
airplane_df['Broadphaseofflight'].isna().value_counts()

False    27580
Name: Broadphaseofflight, dtype: int64

airplane_df.isna().sum()

EventDate                0
Location                 7
Country                  7
Latitude              5498
Longitude             5507
InjurySeverity         812
Aircraftdamage        1279
AircraftCategory         0
```

```
RegistrationNumber       223
Make                       0
Model                      0
AmateurBuilt              17
NumberofEngines         2749
EngineType              4213
FARDescription           499
Purposeofflight         3730
TotalFatalInjuries      3159
TotalSeriousInjuries    3216
TotalMinorInjuries      2871
TotalUninjured           894
WeatherCondition        3044
Broadphaseofflight         0
dtype: int64
```

Most of the remaining factors may contribute to the analysis and others may be used for identification and event dates if needed later. Since we don't have the information, we may not know the values and cannot be filled in. So, let us drop the nulls.

```
#airplane_df.dropna(subset = ['Latitude', 'Longitude'], inplace=True)

airplane_df.dropna(axis=0, inplace=True)

airplane_df.isna().sum()
```

```
EventDate               0
Location                0
Country                 0
Latitude                0
Longitude               0
InjurySeverity          0
Aircraftdamage          0
AircraftCategory        0
RegistrationNumber      0
Make                    0
Model                   0
AmateurBuilt            0
NumberofEngines         0
EngineType              0
FARDescription          0
Purposeofflight         0
TotalFatalInjuries      0
TotalSeriousInjuries    0
TotalMinorInjuries      0
TotalUninjured          0
WeatherCondition        0
Broadphaseofflight      0
dtype: int64
```

```python
len(airplane_df)
```

```
14744
```

```python
airplane_df.columns
```

```
Index(['EventDate', 'Location', 'Country', 'Latitude', 'Longitude',
       'InjurySeverity', 'Aircraftdamage', 'AircraftCategory',
       'RegistrationNumber', 'Make', 'Model', 'AmateurBuilt',
       'NumberofEngines', 'EngineType', 'FARDescription',
'Purposeofflight',
       'TotalFatalInjuries', 'TotalSeriousInjuries',
'TotalMinorInjuries',
       'TotalUninjured', 'WeatherCondition', 'Broadphaseofflight'],
      dtype='object')
```

```python
#Checking the number of makes, they are 7,587
airplane_df['Make'].value_counts()
#Picking the top 20 makes
airplane_df['Make'].value_counts().head(20)
```

```python
#Match the make with the use of the flight. The client needs flights
for commercial and private enterprises
use_make = airplane_df[["Make","Purposeofflight"]].value_counts()
use_make.head(100)
```

```
Make                                Purposeofflight
Cessna                              Personal                   2994
Piper                               Personal                   1969
Cessna                              Instructional               925
Beech                               Personal                    657
Piper                               Instructional               458
                                                               ...
Diamond                             Personal                     11
Cessna                              Public Aircraft - State      11
Luscombe                            Instructional                11
Aeropro cz                          Personal                     11
Costruzioni aeronautiche tecna      Instructional                11
Length: 100, dtype: int64
```

The company intends to use aircraft for business and private enterprises. Aircraft such as instructional, public aircraft (federal, state, local), firefighting, and others, are more specialized and often not classified under general commercial or private enterprise use.

Let us form a dataset with aircraft for the purpose of business and private enterprises: Personal, Business, Executive/Corporate, Aerial Application, Banner Tow, Aerial Observation, Skydiving, Ferry, Flight Test, Positioning.

```python
#Select aircraft for business and private enterprises
selectcraft = ["Personal", "Business", "Executive/corporate", "Aerial
```

```
Application", "Banner Tow", "Aerial Observation", "Skydiving",
"Ferry", "Flight Test", "Positioning"]
#Confirm if the purpose selected is in the dataset purpose of flight
selectcraftSample = airplane_df['Purposeofflight'].isin(selectcraft)

#Create the dataset
airplaneCommPrivUse_df =airplane_df[selectcraftSample]
#Reset index for uniformity
airplaneCommPrivUse_df.reset_index(drop=True, inplace=True)
#Check the top 5 rows
airplaneCommPrivUse_df.head()
```

```
    EventDate          Location        Country   Latitude    Longitude
\
0  2001-06-03   LYTLE CREEK, CA  United States  34.241389  -117.539722

1  2003-06-21       Cushing, OK  United States  35.935833   -96.779167

2  2006-11-04     Yuba City, CA  United States  38.967778  -121.626945

3  2006-12-07  Summersville, WV  United States  38.248611   -80.976111

4  2007-01-15      ADJUNTAS, PR  United States  18.147222   -66.798333


  InjurySeverity Aircraftdamage AircraftCategory RegistrationNumber  \
0       Fatal(1)    Substantial         Airplane             N8253W
1       Fatal(1)      Destroyed         Airplane             N8548S
2       Fatal(2)      Destroyed         Airplane             N158MD
3       Fatal(1)      Destroyed         Airplane             N9165T
4       Fatal(2)    Substantial         Airplane              N90KB

                            Make  ... NumberofEngines
EngineType  \
0                          Piper  ...             1.0  Reciprocating

1                         Cessna  ...             1.0  Reciprocating

2  Aircraft mfg & dev. co. (amd)  ...             1.0  Reciprocating

3                         Mooney  ...             1.0  Reciprocating

4                     Partenavia  ...             2.0  Reciprocating


              FARDescription Purposeofflight TotalFatalInjuries  \
0  Part 91: General Aviation        Personal                1.0
1  Part 91: General Aviation       Skydiving                1.0
2  Part 91: General Aviation        Personal                2.0
3  Part 91: General Aviation        Personal                1.0
4  Part 91: General Aviation        Personal                2.0
```

```
  TotalSeriousInjuries  TotalMinorInjuries  TotalUninjured
WeatherCondition  \
0                  0.0                 0.0             0.0
VMC
1                  2.0                 2.0             1.0
VMC
2                  0.0                 0.0             0.0
VMC
3                  0.0                 0.0             0.0
IMC
4                  0.0                 0.0             0.0
IMC

   Broadphaseofflight
0         Maneuvering
1         Maneuvering
2              Cruise
3              Cruise
4             Descent

[5 rows x 22 columns]
```

```
#Check duplicates and drop them if they exist
airplaneCommPrivUse_df.duplicated().sum()
```

```
0
```

```
#Add statecodes and state to the data.
airplaneCommPrivUse_df['Country'].value_counts()
airplaneCommPrivUse_df['Location'][0][-2:]
#Create the abbreviation column and add null values which will be
edited below
airplaneCommPrivUse_df.insert(21, "Abbreviation", "NaN")

#Loop through the dataset and add abbreviations using the Location
column's last 2 characters
for i in range(len(airplaneCommPrivUse_df['Location'])):
    airplaneCommPrivUse_df["Abbreviation"][i] =
airplaneCommPrivUse_df['Location'][i][-2:]
    i=+1
```

```
<ipython-input-129-5f15448f2f48>:9: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  airplaneCommPrivUse_df["Abbreviation"][i] =
airplaneCommPrivUse_df['Location'][i][-2:]
c:\Users\amerc\anaconda3\envs\learn-env\lib\site-packages\IPython\
```

```
core\interactiveshell.py:3417: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  exec(code_obj, self.user_global_ns, self.user_ns)
```

```
#Confirm the column has been added
airplaneCommPrivUse_df.columns
```

```
Index(['EventDate', 'Location', 'Country', 'Latitude', 'Longitude',
       'InjurySeverity', 'Aircraftdamage', 'AircraftCategory',
       'RegistrationNumber', 'Make', 'Model', 'AmateurBuilt',
       'NumberofEngines', 'EngineType', 'FARDescription',
'Purposeofflight',
       'TotalFatalInjuries', 'TotalSeriousInjuries',
'TotalMinorInjuries',
       'TotalUninjured', 'WeatherCondition', 'Abbreviation',
       'Broadphaseofflight'],
      dtype='object')
```

```
#Merge data from the 2 datasets using a left join as we only need to
fill in the state from the us_state dataset
airplaneCommPrivUse_df_merged
=airplaneCommPrivUse_df.merge(usstate_df, how='left',
on='Abbreviation')
```

```
#confirm by checking the top 5 rows
airplaneCommPrivUse_df_merged.head()
```

```
    EventDate           Location        Country   Latitude    Longitude
\
0  2001-06-03   LYTLE CREEK, CA  United States  34.241389  -117.539722

1  2003-06-21       Cushing, OK  United States  35.935833   -96.779167

2  2006-11-04     Yuba City, CA  United States  38.967778  -121.626945

3  2006-12-07  Summersville, WV  United States  38.248611   -80.976111

4  2007-01-15      ADJUNTAS, PR  United States  18.147222   -66.798333


  InjurySeverity Aircraftdamage AircraftCategory RegistrationNumber  \
0       Fatal(1)    Substantial         Airplane             N8253W
1       Fatal(1)      Destroyed         Airplane             N8548S
2       Fatal(2)      Destroyed         Airplane             N158MD
3       Fatal(1)      Destroyed         Airplane             N9165T
4       Fatal(2)    Substantial         Airplane              N90KB
```

```
                            Make  ...              FARDescription  \
0                          Piper  ...  Part 91: General Aviation
1                         Cessna  ...  Part 91: General Aviation
2  Aircraft mfg & dev. co. (amd)  ...  Part 91: General Aviation
3                         Mooney  ...  Part 91: General Aviation
4                     Partenavia  ...  Part 91: General Aviation

  Purposeofflight  TotalFatalInjuries TotalSeriousInjuries
TotalMinorInjuries  \
0        Personal                 1.0                  0.0
0.0
1       Skydiving                 1.0                  2.0
2.0
2        Personal                 2.0                  0.0
0.0
3        Personal                 1.0                  0.0
0.0
4        Personal                 2.0                  0.0
0.0

  TotalUninjured  WeatherCondition  Abbreviation
Broadphaseofflight  \
0            0.0               VMC            CA         Maneuvering

1            1.0               VMC            OK         Maneuvering

2            0.0               VMC            CA              Cruise

3            0.0               IMC            WV              Cruise

4            0.0               IMC            PR             Descent


        US_State
0     California
1       Oklahoma
2     California
3  West Virginia
4    Puerto Rico

[5 rows x 24 columns]
```

```
#Check information about the merged dataset
airplaneCommPrivUse_df_merged.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 12376 entries, 0 to 12375
Data columns (total 24 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   EventDate             12376 non-null  object
```

```
 1   Location               12376 non-null  object
 2   Country                12376 non-null  object
 3   Latitude               12376 non-null  object
 4   Longitude              12376 non-null  object
 5   InjurySeverity         12376 non-null  object
 6   Aircraftdamage         12376 non-null  object
 7   AircraftCategory       12376 non-null  object
 8   RegistrationNumber     12376 non-null  object
 9   Make                   12376 non-null  object
 10  Model                  12376 non-null  object
 11  AmateurBuilt           12376 non-null  object
 12  NumberofEngines        12376 non-null  float64
 13  EngineType             12376 non-null  object
 14  FARDescription         12376 non-null  object
 15  Purposeofflight        12376 non-null  object
 16  TotalFatalInjuries     12376 non-null  float64
 17  TotalSeriousInjuries   12376 non-null  float64
 18  TotalMinorInjuries     12376 non-null  float64
 19  TotalUninjured         12376 non-null  float64
 20  WeatherCondition       12376 non-null  object
 21  Abbreviation           12376 non-null  object
 22  Broadphaseofflight     12376 non-null  object
 23  US_State               12290 non-null  object
dtypes: float64(5), object(19)
memory usage: 2.4+ MB
```

```
#Confirm if there is missing data after merging
airplaneCommPrivUse_df_merged.isna().sum()
```

```
EventDate                0
Location                 0
Country                  0
Latitude                 0
Longitude                0
InjurySeverity           0
Aircraftdamage           0
AircraftCategory         0
RegistrationNumber       0
Make                     0
Model                    0
AmateurBuilt             0
NumberofEngines          0
EngineType               0
FARDescription           0
Purposeofflight          0
TotalFatalInjuries       0
TotalSeriousInjuries     0
TotalMinorInjuries       0
TotalUninjured           0
WeatherCondition         0
```

```
Abbreviation                 0
Broadphaseofflight           0
US_State                    86
dtype: int64
```

Note the merge created some null values. So we fill in 'Unknown" as we cannot drop the rows

```
#Fiil in the missing values with 'unknown'
airplaneCommPrivUse_df_merged['US_State'].fillna("Unknown",
inplace=True)
len(airplaneCommPrivUse_df_merged)
#Reset index for uniformity
airplaneCommPrivUse_df_merged.reset_index(drop=True, inplace=True)
airplaneCommPrivUse_df_merged.head()
```

```
    EventDate          Location        Country   Latitude    Longitude
\
0  2001-06-03   LYTLE CREEK, CA  United States  34.241389  -117.539722

1  2003-06-21       Cushing, OK  United States  35.935833   -96.779167

2  2006-11-04     Yuba City, CA  United States  38.967778  -121.626945

3  2006-12-07  Summersville, WV  United States  38.248611   -80.976111

4  2007-01-15      ADJUNTAS, PR  United States  18.147222   -66.798333


  InjurySeverity Aircraftdamage AircraftCategory RegistrationNumber  \
0       Fatal(1)    Substantial         Airplane             N8253W
1       Fatal(1)      Destroyed         Airplane             N8548S
2       Fatal(2)      Destroyed         Airplane             N158MD
3       Fatal(1)      Destroyed         Airplane             N9165T
4       Fatal(2)    Substantial         Airplane              N90KB

                            Make  ...              FARDescription  \
0                          Piper  ...  Part 91: General Aviation
1                         Cessna  ...  Part 91: General Aviation
2  Aircraft mfg & dev. co. (amd)  ...  Part 91: General Aviation
3                         Mooney  ...  Part 91: General Aviation
4                     Partenavia  ...  Part 91: General Aviation

  Purposeofflight  TotalFatalInjuries TotalSeriousInjuries
TotalMinorInjuries  \
0        Personal                 1.0                  0.0
0.0
1       Skydiving                 1.0                  2.0
2.0
2        Personal                 2.0                  0.0
0.0
```

```
3        Personal                 1.0                    0.0
0.0
4        Personal                 2.0                    0.0
0.0

  TotalUninjured  WeatherCondition  Abbreviation
Broadphaseofflight  \
0             0.0               VMC            CA         Maneuvering

1             1.0               VMC            OK         Maneuvering

2             0.0               VMC            CA              Cruise

3             0.0               IMC            WV              Cruise

4             0.0               IMC            PR             Descent


        US_State
0     California
1       Oklahoma
2     California
3  West Virginia
4    Puerto Rico

[5 rows x 24 columns]
```

EDA

Checking history data

Let us check on the make and model of the aircrafts. This will show which ones are used/bought often.

Later we will examine Aircraft Damage by reviewing the historical damage records of each aircraft model. Models with frequent or severe damage might indicate higher risk

Assess Injury Severity: Analyze the severity of injuries reported for each aircraft model. Lower injury severity can indicate better safety performance.

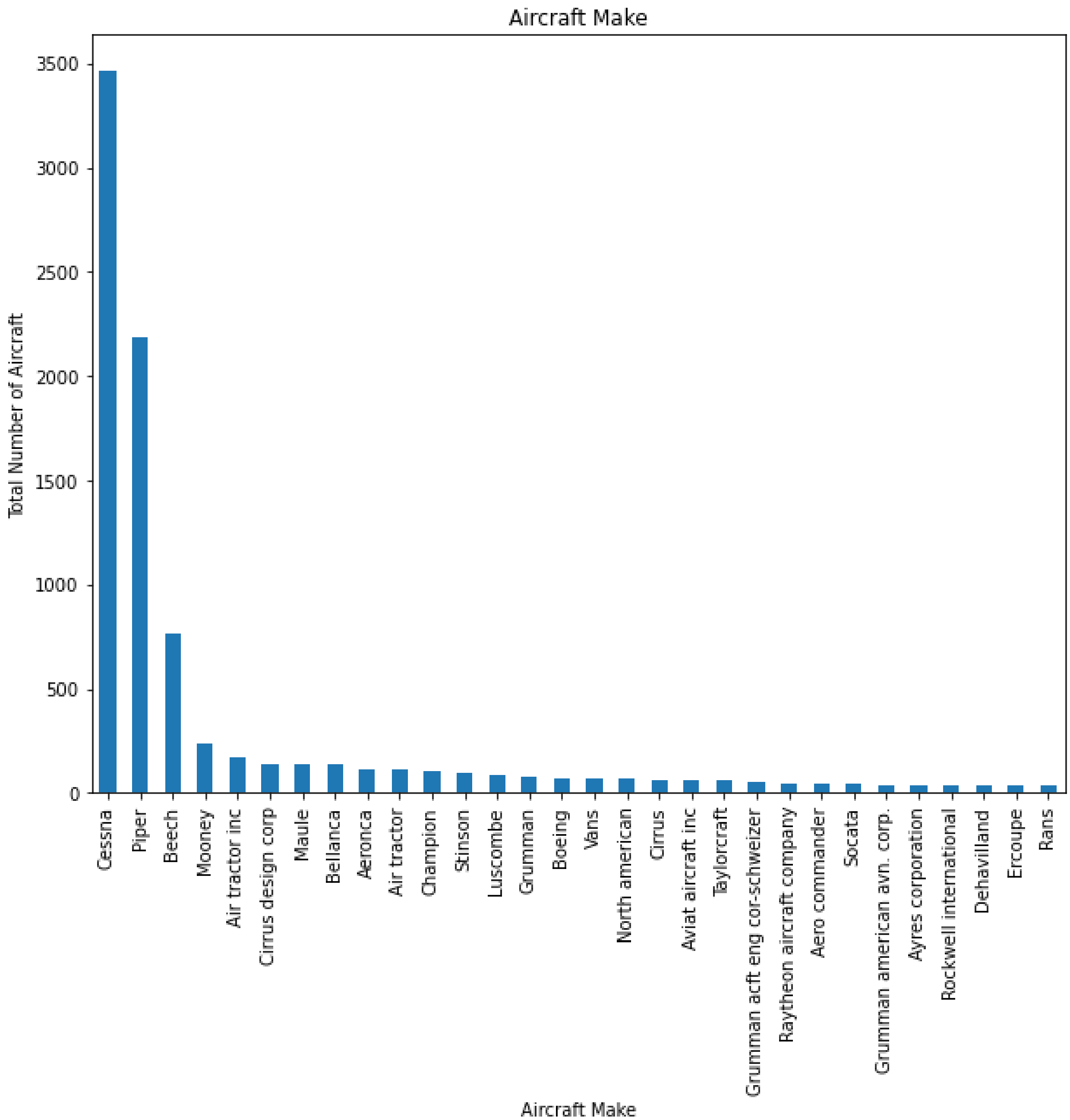
Review Total Fatal and Serious Injuries: High numbers of fatal or serious injuries might be red flags. Compare these numbers across different aircraft models to identify safer options

```
#Which 'Make' are so common: Cessna, Piper, Beech, Mooney
airplaneCommPrivUse_df_merged['Make'].value_counts()
```

```
Cessna              3463
Piper               2184
Beech                762
Mooney               233
```

```
Air tractor inc      167
                     ...
Freeman                1
Riffel jerris l        1
Cortesy, john e        1
Richmond jim r         1
Woolston glenn e       1
Name: Make, Length: 2454, dtype: int64
```

```
#Plot the graph showing Make
df = airplaneCommPrivUse_df_merged.Make.value_counts()
#Plotting the top 30 'Makes"
df[:30].plot(kind='bar', figsize=(10,8))
plt.title("Aircraft Make")
plt.xlabel("Aircraft Make")
plt.ylabel("Total Number of Aircraft")
plt.show()
```




```
#Create a clean dataset
airplaneCommPrivUse_df_merged.head()
#Save the new cleaned data
airplaneCommPrivUse_df_merged.to_csv("clean_aircraft.csv",
index=False)

#Load and read the cleaned dataset
clean_aircraft_df = pd.read_csv("clean_aircraft.csv")
clean_aircraft_df.head()
```

```
   EventDate          Location        Country   Latitude    Longitude  \
0  2001-06-03   LYTLE CREEK, CA  United States  34.241389  -117.539722

1  2003-06-21       Cushing, OK  United States  35.935833   -96.779167

2  2006-11-04     Yuba City, CA  United States  38.967778  -121.626945

3  2006-12-07  Summersville, WV  United States  38.248611   -80.976111

4  2007-01-15      ADJUNTAS, PR  United States  18.147222   -66.798333


  InjurySeverity Aircraftdamage AircraftCategory RegistrationNumber  \
0       Fatal(1)    Substantial         Airplane             N8253W
1       Fatal(1)      Destroyed         Airplane             N8548S
2       Fatal(2)      Destroyed         Airplane             N158MD
3       Fatal(1)      Destroyed         Airplane             N9165T
4       Fatal(2)    Substantial         Airplane              N90KB

                            Make  ...             FARDescription  \
0                          Piper  ...  Part 91: General Aviation
1                         Cessna  ...  Part 91: General Aviation
2  Aircraft mfg & dev. co. (amd)  ...  Part 91: General Aviation
3                         Mooney  ...  Part 91: General Aviation
4                     Partenavia  ...  Part 91: General Aviation

  Purposeofflight  TotalFatalInjuries TotalSeriousInjuries
TotalMinorInjuries  \
0        Personal                 1.0                  0.0
0.0
1       Skydiving                 1.0                  2.0
2.0
2        Personal                 2.0                  0.0
0.0
3        Personal                 1.0                  0.0
0.0
4        Personal                 2.0                  0.0
0.0

  TotalUninjured  WeatherCondition  Abbreviation
Broadphaseofflight  \
0             0.0               VMC            CA        Maneuvering

1             1.0               VMC            OK        Maneuvering

2             0.0               VMC            CA             Cruise

3             0.0               IMC            WV             Cruise

4             0.0               IMC            PR            Descent
```

```
        US_State
0     California
1       Oklahoma
2     California
3  West Virginia
4    Puerto Rico

[5 rows x 24 columns]
```

```
#Check data types for clarity during visualization
clean_aircraft_df.dtypes
```

```
EventDate                  object
Location                   object
Country                    object
Latitude                   object
Longitude                  object
InjurySeverity             object
Aircraftdamage             object
AircraftCategory           object
RegistrationNumber         object
Make                       object
Model                      object
AmateurBuilt               object
NumberofEngines           float64
EngineType                 object
FARDescription             object
Purposeofflight            object
TotalFatalInjuries        float64
TotalSeriousInjuries      float64
TotalMinorInjuries        float64
TotalUninjured            float64
WeatherCondition           object
Abbreviation               object
Broadphaseofflight         object
US_State                   object
dtype: object
```

```
#Change the datatypes from float to integer to avoid decimal places as
it represents people
clean_aircraft_df["TotalFatalInjuries"] =
clean_aircraft_df["TotalFatalInjuries"].astype("int")
clean_aircraft_df["TotalSeriousInjuries"] =
clean_aircraft_df["TotalSeriousInjuries"].astype("int")
clean_aircraft_df["TotalMinorInjuries"] =
clean_aircraft_df["TotalMinorInjuries"].astype("int")
clean_aircraft_df["TotalUninjured"] =
clean_aircraft_df["TotalUninjured"].astype("int")
```

```
clean_aircraft_df["NumberofEngines"] =
clean_aircraft_df["NumberofEngines"].astype("int")

#Save the cleaned copy to a csv file
clean_aircraft_df.to_csv("clean_aircraft.csv", index=False)

#Access the cleaned dataset for use
clean_aircraft_df = pd.read_csv("clean_aircraft.csv")
clean_aircraft_df.head()
```

```
    EventDate           Location        Country   Latitude    Longitude
\
0  2001-06-03   LYTLE CREEK, CA  United States  34.241389  -117.539722

1  2003-06-21       Cushing, OK  United States  35.935833   -96.779167

2  2006-11-04     Yuba City, CA  United States  38.967778  -121.626945

3  2006-12-07  Summersville, WV  United States  38.248611   -80.976111

4  2007-01-15      ADJUNTAS, PR  United States  18.147222   -66.798333


  InjurySeverity Aircraftdamage AircraftCategory RegistrationNumber  \
0       Fatal(1)    Substantial         Airplane             N8253W
1       Fatal(1)      Destroyed         Airplane             N8548S
2       Fatal(2)      Destroyed         Airplane             N158MD
3       Fatal(1)      Destroyed         Airplane             N9165T
4       Fatal(2)    Substantial         Airplane              N90KB

                            Make  ...              FARDescription  \
0                          Piper  ...  Part 91: General Aviation
1                         Cessna  ...  Part 91: General Aviation
2  Aircraft mfg & dev. co. (amd)  ...  Part 91: General Aviation
3                         Mooney  ...  Part 91: General Aviation
4                     Partenavia  ...  Part 91: General Aviation

  Purposeofflight  TotalFatalInjuries TotalSeriousInjuries
TotalMinorInjuries  \
0        Personal                   1                    0
0
1       Skydiving                   1                    2
2
2        Personal                   2                    0
0
3        Personal                   1                    0
0
4        Personal                   2                    0
0

  TotalUninjured  WeatherCondition  Abbreviation
```

```
Broadphaseofflight  \
0                0               VMC            CA         Maneuvering

1                1               VMC            OK         Maneuvering

2                0               VMC            CA              Cruise

3                0               IMC            WV              Cruise

4                0               IMC            PR             Descent


        US_State
0     California
1       Oklahoma
2     California
3  West Virginia
4    Puerto Rico

[5 rows x 24 columns]
```

```
clean_aircraft_df.dtypes
```

```
EventDate                object
Location                 object
Country                  object
Latitude                 object
Longitude                object
InjurySeverity           object
Aircraftdamage           object
AircraftCategory         object
RegistrationNumber       object
Make                     object
Model                    object
AmateurBuilt             object
NumberofEngines           int64
EngineType               object
FARDescription           object
Purposeofflight          object
TotalFatalInjuries        int64
TotalSeriousInjuries      int64
TotalMinorInjuries        int64
TotalUninjured            int64
WeatherCondition         object
Abbreviation             object
Broadphaseofflight       object
US_State                 object
dtype: object
```

```python
#Check which model are common
model_df = clean_aircraft_df.Model.value_counts()
model_df.head()
```

```
172     315
182     186
180     156
SR22    154
PA28    132
Name: Model, dtype: int64
```

```python
#Check Make or Model against damage, grouping by the damage
clean_aircraft_df.groupby(['Aircraftdamage', 'Make'])['Make'].count()
```

```
Aircraftdamage  Make
Destroyed       Aero commander        4
                Aero vodochody        2
                Aerofab inc.          1
                Aeronca               6
                Aeropro cz            1
                                     ..
Substantial     Zwicker murray r      1
Unknown         Aero commander        1
                Cessna                1
                Piper aircraft inc    1
                Swann lynn j          1
Name: Make, Length: 2633, dtype: int64
```

```python
#How are the Aircraft associated with injuries
type(clean_aircraft_df)
clean_aircraft_df.groupby(['TotalFatalInjuries', 'Make'])
['Make'].count()
```

```
TotalFatalInjuries  Make
0                   177mf llc               1
                    2007 savage air llc     1
                    2021fx3 llc             1
                    781569 inc              1
                    Aardema robert john     1
                                           ..
9                   Pilatus                 1
10                  Beech                   1
                    Textron aviation        1
11                  Beech                   1
14                  Pilatus                 1
Name: Make, Length: 2744, dtype: int64
```

```python
#Check if the aircrafts follow Federal Aviation Regulations (FAR)
clean_aircraft_df['FARDescription'].isna().sum()
```

```
0
```

```python
#check how the engine type and number of engines are correlated with
safety
x = clean_aircraft_df['NumberofEngines']
y = clean_aircraft_df['TotalFatalInjuries']
np.corrcoef(x,y)
```

```
array([[1.        , 0.14041973],
       [0.14041973, 1.        ]])
```

```python
clean_aircraft_df.columns
```

```
Index(['EventDate', 'Location', 'Country', 'Latitude', 'Longitude',
       'InjurySeverity', 'Aircraftdamage', 'AircraftCategory',
       'RegistrationNumber', 'Make', 'Model', 'AmateurBuilt',
       'NumberofEngines', 'EngineType', 'FARDescription',
'Purposeofflight',
       'TotalFatalInjuries', 'TotalSeriousInjuries',
'TotalMinorInjuries',
       'TotalUninjured', 'WeatherCondition', 'Abbreviation',
       'Broadphaseofflight', 'US_State'],
      dtype='object')
```

```python
#Check  Make vs Injury severity
clean_aircraft_df.groupby(['InjurySeverity', 'Make'])['Make'].count()
```

```
InjurySeverity  Make
Fatal           Adams donald l                    1
                Advertising mgmt & consulting     1
                Aero adventure                    1
                Aero commander                   11
                Aero sp z o o                     1
                                                 ..
Serious         Globe                             2
                Miller roger                      1
                Piper                             1
                Quicksilver aircraft co           1
                Sawby scott                       1
Name: Make, Length: 2679, dtype: int64
```

```python
clean_aircraft_df.corr(method='pearson')
```

```
                      NumberofEngines  TotalFatalInjuries  \
NumberofEngines              1.000000            0.140420
TotalFatalInjuries           0.140420            1.000000
TotalSeriousInjuries        -0.018796           -0.139954
TotalMinorInjuries          -0.011290           -0.029562
TotalUninjured               0.104715           -0.188698

                      TotalSeriousInjuries  TotalMinorInjuries
TotalUninjured
NumberofEngines                  -0.018796           -0.011290
```

```
0.104715
TotalFatalInjuries                          -0.139954            -0.029562        -
0.188698
TotalSeriousInjuries                         1.000000             0.007739        -
0.151834
TotalMinorInjuries                           0.007739             1.000000        -
0.151278
TotalUninjured                              -0.151834            -0.151278
1.000000
```

The Number of Engines has weak relationships with the number of injuries whether be serious, minor or fatal injury. The same applies to the uninjured totals.

This can also be seen in the graphs below.

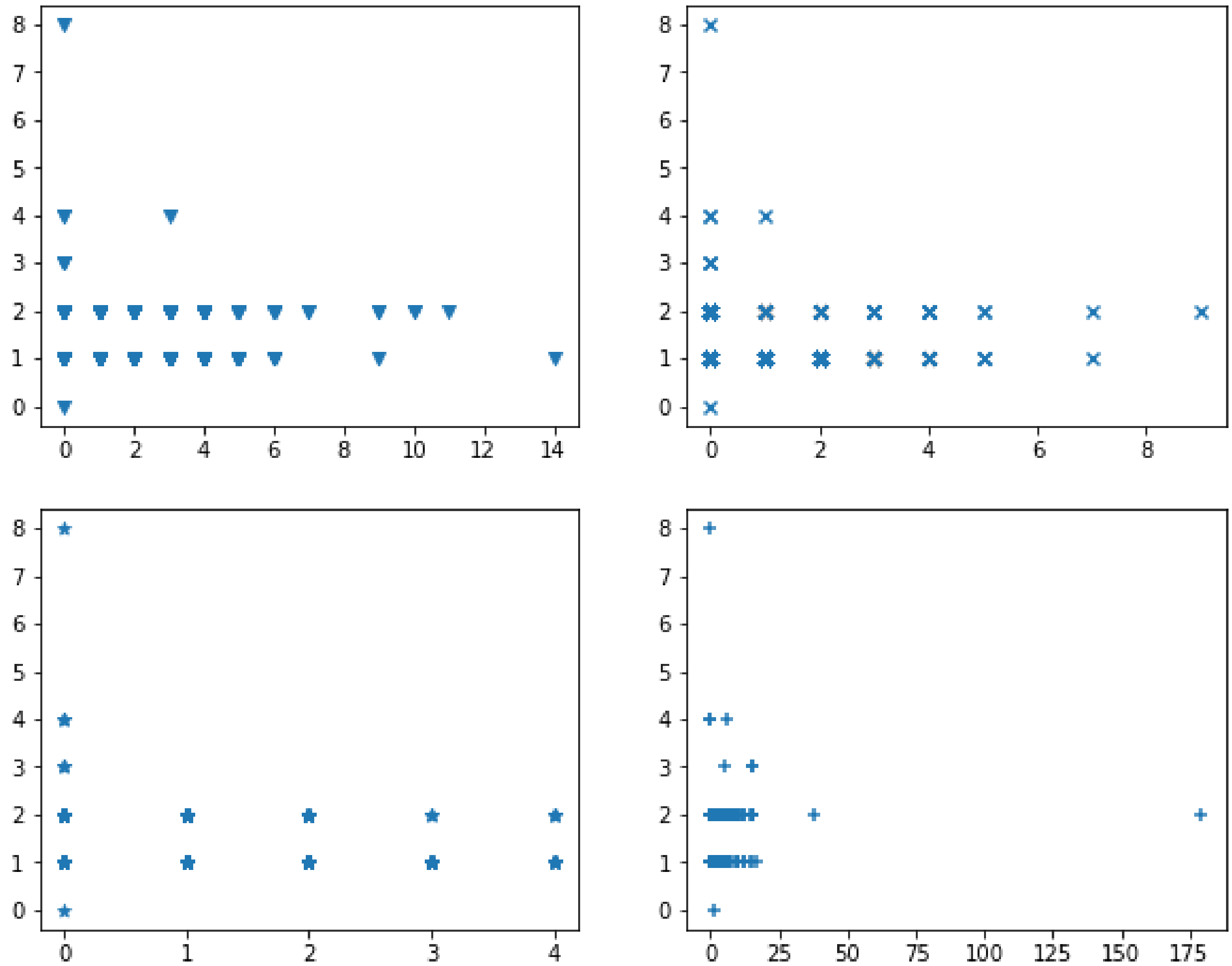
```
len(clean_aircraft_df['Make'].unique())

2454

x = clean_aircraft_df['NumberofEngines']
y = clean_aircraft_df['TotalFatalInjuries']
r = np.corrcoef(x,y)
r

array([[1.        , 0.14041973],
       [0.14041973, 1.        ]])

#Using scatter plot to chek relationship between number of Engines and
injuries
fig, axs = plt.subplots(2,2, figsize =(10,8))
axs[0,0].scatter(x=clean_aircraft_df['TotalFatalInjuries'],
y=clean_aircraft_df['NumberofEngines'], marker='v')
axs[0,1].scatter(x=clean_aircraft_df['TotalSeriousInjuries'],
y=clean_aircraft_df['NumberofEngines'], marker='x')
axs[1,0].scatter(x=clean_aircraft_df['TotalMinorInjuries'],
y=clean_aircraft_df['NumberofEngines'], marker='*')
axs[1,1].scatter(x=clean_aircraft_df['TotalUninjured'],
y=clean_aircraft_df['NumberofEngines'], marker='+')
plt.show()
```



```
clean_aircraft_df['Model'].value_counts().tail()
clean_aircraft_df['Make'].value_counts().tail()

Freeman              1
Riffel jerris l      1
Cortesy, john e      1
Richmond jim r       1
Woolston glenn e     1
Name: Make, dtype: int64
```

From the analysis, The Cessna aircraft Make is more common. It has the highest number of injuries but also leads on uninjured.

On the comparing model and Severity of injury we find '172' Make having the highest Non-Fatal accident/incidents. But we also see Cessna Make having more models that have Non-Fatal incidents/accidents.

Some other data cleaning challenges realised later. Which need more time for cleaning

```
#Some key attributes records are keyed in using different text
clean_aircraft_df['Make'].isin(['Air tractor inc.']).value_counts()
#clean_aircraft.loc[clean_aircraft['Model'], clean_aircraft['Make'] ==
"Air tractor inc"]#.value_counts()
(clean_aircraft_df['Make']=="Air tractor").value_counts()
```

```
False    12268
True       108
Name: Make, dtype: int64
```

```
(clean_aircraft_df['Make']=="Air tractor inc").value_counts()
```

```
False    12209
True       167
Name: Make, dtype: int64
```

```
(clean_aircraft_df['Make']=="Air tractor inc.").value_counts()
```

```
False    12373
True         3
Name: Make, dtype: int64
```

Recommendations

1. We would recomend the top 3 showing more safety in terms of non-Fatal accidents and Level of Damage. i.e.: Cessna, Piper and Beech Make.

2. The model that has low-risk are the 172 models (of Cessna), with the most non-fatal injuries. This is followed by Piper and Beech. Cessna has the highest number of model count for non-fatal injuries. Thus showing more safety.

3. More research to be done based on other factors before making a decision on the chose of Airplane.

```
clean_aircraft_df.loc[clean_aircraft_df['Model'] == '172'].head()
```

```
      EventDate         Location        Country Latitude Longitude  \
185  2008-03-01  Apple River, IL  United States  042303N  0090521W
210  2008-03-13   Indiantown, FL  United States  027102N  0080361W
374  2008-05-03     Anchorge, AK  United States  611113N  1495755W
405  2008-05-10      Chugiak, AK  United States  612459N  1493026W
408  2008-05-10   Big Timber, MT  United States  454824N  1095854W

    InjurySeverity Aircraftdamage AircraftCategory RegistrationNumber
Make  \
185      Non-Fatal    Substantial         Airplane             N8366B
Cessna
210          Fatal    Substantial         Airplane             N284SP
Cessna
374      Non-Fatal    Substantial         Airplane             N7886G
```

```
Cessna
405      Non-Fatal    Substantial         Airplane            N7598T
Cessna
408      Non-Fatal    Substantial         Airplane            N8103E
Cessna

     ... FARDescription     Purposeofflight  TotalFatalInjuries  \
185  ...            091            Personal                   0
210  ...            091  Aerial Observation                   4
374  ...            091            Personal                   0
405  ...            091            Personal                   0
408  ...            091            Personal                   0

    TotalSeriousInjuries TotalMinorInjuries TotalUninjured
WeatherCondition  \
185                    0                  0              3
VMC
210                    0                  0              0
VMC
374                    0                  0              2
VMC
405                    0                  0              1
VMC
408                    0                  0              1
VMC

     Abbreviation  Broadphaseofflight  US_State
185            IL             Unknown  Illinois
210            FL             Unknown   Florida
374            AK             Unknown    Alaska
405            AK             Unknown    Alaska
408            MT             Unknown   Montana

[5 rows x 24 columns]
```