Cleaning 'AviationData.csv' from NTSB

The dataset can be found here (url). Our goal is to identifying high-risk aircraft models to guide purchasing decisions.

Our goals are:

- to identifying high-risk aircraft models to guide purchasing decisions;
- focus on aircraft safety to reduce costs and improve brand reputation;
- explore weather conditions and their impact on accidents.

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# Custom date parser functions
date_parser_1 = lambda x: pd.to_datetime(x, format='%Y-%m-%d')
date_parser_2 = lambda x: pd.to_datetime(x, format='%d-%m-%Y')

df = pd.read_csv('data/AviationData.csv', encoding='ISO-8859-1', conve
df.head()
```

Out[1]:

	Investigation.Type	Accident.Number	Event.Date	Location	Country	La
Event.Id						
20001218X45444	Accident	SEA87LA080	1948-10-24	MOOSE CREEK, ID	United States	
20001218X45447	Accident	LAX94LA336	1962-07-19	BRIDGEPORT, CA	United States	
20061025X01555	Accident	NYC07LA005	1974-08-30	Saltville, VA	United States	36.9
20001218X45448	Accident	LAX96LA321	1977-06-19	EUREKA, CA	United States	
20041105X01764	Accident	CHI79FA064	1979-08-02	Canton, OH	United States	

5 rows × 30 columns

Let's explore the dataset

In [2]:
```
df.shape
```

Out[2]:
```
(88889, 30)
```

In [3]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 88889 entries, 20001218X45444 to 20221230106513
Data columns (total 30 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Investigation.Type      88889 non-null  object
 1   Accident.Number         88889 non-null  object
 2   Event.Date              88889 non-null  datetime64[ns]
 3   Location                88837 non-null  object
 4   Country                 88663 non-null  object
 5   Latitude                34382 non-null  object
 6   Longitude               34373 non-null  object
 7   Airport.Code            50132 non-null  object
 8   Airport.Name            52704 non-null  object
 9   Injury.Severity         87889 non-null  object
 10  Aircraft.damage         85695 non-null  object
 11  Aircraft.Category       32287 non-null  object
 12  Registration.Number     87507 non-null  object
 13  Make                    88826 non-null  object
 14  Model                   88797 non-null  object
 15  Amateur.Built           88787 non-null  object
 16  Number.of.Engines       82805 non-null  float64
 17  Engine.Type             81793 non-null  object
 18  FAR.Description         32023 non-null  object
 19  Schedule                12582 non-null  object
 20  Purpose.of.flight       82697 non-null  object
 21  Air.carrier             16648 non-null  object
 22  Total.Fatal.Injuries    77488 non-null  float64
 23  Total.Serious.Injuries  76379 non-null  float64
 24  Total.Minor.Injuries    76956 non-null  float64
 25  Total.Uninjured         82977 non-null  float64
 26  Weather.Condition       84397 non-null  object
 27  Broad.phase.of.flight   61724 non-null  object
 28  Report.Status           82505 non-null  object
 29  Publication.Date        75118 non-null  datetime64[ns]
dtypes: datetime64[ns](2), float64(5), object(23)
memory usage: 21.0+ MB
```

In [4]:
```
# Check for duplicate rows
df.duplicated().sum()
```

Out[4]:
```
0
```

In [5]:

```
# Count missing values in each column
df.isnull().sum()
```

Out[5]:

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

In [6]:

```
# Calculate the percentage of missing values in each column
(df.isnull().sum() * 100 / len(df)).round(2)
```

Out[6]:

```
Investigation.Type         0.00
Accident.Number            0.00
Event.Date                 0.00
Location                   0.06
Country                    0.25
Latitude                  61.32
Longitude                 61.33
Airport.Code              43.60
Airport.Name              40.71
Injury.Severity            1.12
Aircraft.damage            3.59
Aircraft.Category         63.68
Registration.Number        1.55
Make                       0.07
Model                      0.10
Amateur.Built              0.11
Number.of.Engines          6.84
Engine.Type                7.98
FAR.Description           63.97
Schedule                  85.85
Purpose.of.flight          6.97
Air.carrier               81.27
Total.Fatal.Injuries      12.83
Total.Serious.Injuries    14.07
Total.Minor.Injuries      13.42
Total.Uninjured            6.65
Weather.Condition          5.05
Broad.phase.of.flight     30.56
Report.Status              7.18
Publication.Date          15.49
dtype: float64
```

Let's drop columns with 40%+ missing values

In [7]:

```
rt.Code', 'Airport.Name', 'Aircraft.Category', 'FAR.Description', 'Sch
```

Let's get unique values and their count for each columns

```
In [8]: unique_values_dict = {col: df[col].unique() for col in df.columns}

        for column, unique_vals in unique_values_dict.items():
            unique_count = len(unique_vals)
            print(f"Unique values in column '{column}' ({unique_count}): {unic
```

```
 'Fatal(135)' 'Fatal(31)' 'Fatal(256)' 'Fatal(25)' 'Fatal(82)'
 'Fatal(156)' 'Fatal(28)' 'Fatal(18)' 'Fatal(43)' 'Fatal(15)' 'Fatal(
270)'
 'Fatal(144)' 'Fatal(174)' 'Fatal(111)' 'Fatal(131)' 'Fatal(20)'
 'Fatal(73)' 'Fatal(27)' 'Fatal(34)' 'Fatal(87)' 'Fatal(30)' 'Fatal(1
6)'
 'Fatal(47)' 'Fatal(56)' 'Fatal(37)' 'Fatal(132)' 'Fatal(68)' 'Fatal(
54)'
 'Fatal(52)' 'Fatal(65)' 'Fatal(72)' 'Fatal(160)' 'Fatal(189)'
 'Fatal(123)' 'Fatal(33)' 'Fatal(110)' 'Fatal(230)' 'Fatal(97)'
 'Fatal(349)' 'Fatal(125)' 'Fatal(35)' 'Fatal(228)' 'Fatal(75)'
 'Fatal(104)' 'Fatal(229)' 'Fatal(80)' 'Fatal(217)' 'Fatal(169)'
 'Fatal(88)' 'Fatal(19)' 'Fatal(60)' 'Fatal(113)' 'Fatal(143)' 'Fata
l(83)'
 'Fatal(24)' 'Fatal(44)' 'Fatal(64)' 'Fatal(92)' 'Fatal(118)' 'Fatal(
265)'
 'Fatal(26)' 'Fatal(138)' 'Fatal(206)' 'Fatal(71)' 'Fatal(21)' 'Fata
l(46)'
 'Fatal(102)' 'Fatal(115)' 'Fatal(141)' 'Fatal(55)' 'Fatal(121)'
 'Fatal(45)' 'Fatal(145)' 'Fatal(117)' 'Fatal(107)' 'Fatal(124)'
```

Let's explore 'Country' variable

```
In [9]: df['Country'].value_counts()
```

```
Out[9]: Country
        United States                       82248
        Brazil                                374
        Canada                                359
        Mexico                                358
        United Kingdom                        344
                                            ...
        Seychelles                              1
        Palau                                   1
        Libya                                   1
        Saint Vincent and the Grenadines        1
        Turks and Caicos Islands                1
        Name: count, Length: 219, dtype: int64
```

```
In [10]: # Drop all rows where 'Country' is not 'United States'
         df = df[df['Country']=='United States']
         df['Country'].value_counts()
```

```
Out[10]: Country
         United States    82248
         Name: count, dtype: int64
```

Let's explore 'Amateur.Built' column

```
In [11]: df['Amateur.Built'].value_counts()
```

```
Out[11]: Amateur.Built
         No     73906
         Yes     8321
         Name: count, dtype: int64
```

```
In [12]: # Remove all aircraft unless not amateur built
         df = df[df['Amateur.Built']=='No']
```

Let's explore 'Number.of.Engines' column

```
In [13]: df['Number.of.Engines'].value_counts()
```

```
Out[13]: Number.of.Engines
         1.0    60409
         2.0    10010
         0.0     1045
         3.0      430
         4.0      339
         8.0        3
         6.0        1
         Name: count, dtype: int64
```

```
In [14]: # Replace missing values in "Number.of.Engines" with -1
         df['Number.of.Engines'] = df['Number.of.Engines'].fillna(-1)
```

In [15]:
```
df['Engine.Type'] = df['Engine.Type'].replace(['UNK'], 'Unknown')
df['Engine.Type'] = df['Engine.Type'].fillna('Unknown')

df['Engine.Type'].value_counts()
```

Out[15]:
```
Engine.Type
Reciprocating    60672
Unknown           4046
Turbo Shaft       3303
Turbo Prop        3130
Turbo Fan         2087
Turbo Jet          654
Electric            10
LR                   2
Hybrid Rocket        1
NONE                 1
Name: count, dtype: int64
```

Drop rows with missing values for 'Model' and 'Make'

In [16]:
```
df.dropna(subset=['Model'], inplace=True)
```

In [17]:
```
df.dropna(subset=['Make'], inplace=True)
```

Let's explore 'Purpose.of.Flight' variable

In [18]:
```
df['Purpose.of.flight'].value_counts()
```

Out[18]:
```
Purpose.of.flight
Personal                     41128
Instructional                10169
Unknown                       5451
Aerial Application            4611
Business                      3779
Positioning                   1546
Other Work Use                1179
Aerial Observation             704
Ferry                          693
Public Aircraft                668
Executive/corporate            502
Flight Test                    233
Skydiving                      171
External Load                  111
Banner Tow                     101
Public Aircraft - Federal       97
Public Aircraft - Local         74
Public Aircraft - State         62
Air Race show                   58
Glider Tow                      52
Air Race/show                   38
Firefighting                    29
Air Drop                         8
ASHO                             5
PUBS                             4
PUBL                             1
Name: count, dtype: int64
```

In [19]:
```
# 'Purpose,of.flight'
df['Purpose.of.flight'] = df['Purpose.of.flight'].fillna('Unknown')
df['Purpose.of.flight'] = df['Purpose.of.flight'].replace('Air Race sh
```

In [20]:
```
# Replace values in 'Purpose.of.flight'
df['Purpose.of.flight'] = df['Purpose.of.flight'].replace(['Public Air
                                                            'Public Air
```

In [21]:
```
# Replace nan values with 'Unknown'
df['Purpose.of.flight'] = df['Purpose.of.flight'].fillna('Unknown')
```

Replace missing values with 'Unknown' in 'Broad.phase.of.flight'

In [22]:
```
df['Broad.phase.of.flight'] = df['Broad.phase.of.flight'].fillna('Unkr

df['Broad.phase.of.flight'].value_counts()
```

Out[22]:
```
Broad.phase.of.flight
Unknown        18731
Landing        14441
Takeoff        10953
Cruise          9180
Maneuvering     7104
Approach        5806
Taxi            1860
Climb           1800
Descent         1741
Go-around       1270
Standing         904
Other             99
Name: count, dtype: int64
```

Let's explore 'Weather.Condition' variable

In [23]:
```
# 'Weather.Condition' (5): ['UNK' 'IMC' 'VMC' nan 'Unk']
df['Weather.Condition'].value_counts()
```

Out[23]:
```
Weather.Condition
VMC    67162
IMC     5484
UNK      509
Unk      117
Name: count, dtype: int64
```

In [24]:
```
# Replace "UNK" and "Unk" in 'Weather.Condition' column with 'Unknown'
df['Weather.Condition'] = df['Weather.Condition'].replace(['UNK', 'Unk

# Replace nan values with 'Unknown'
df['Weather.Condition'] = df['Weather.Condition'].fillna('Unknown')
```

In [25]:
```
df['Weather.Condition'].value_counts()
```

Out[25]:
```
Weather.Condition
VMC        67162
IMC         5484
Unknown     1243
Name: count, dtype: int64
```

In [26]:
```
# Let's replace missing values in 'Report.Status' with 'Unknown'
```

In [27]:

```
df['Report.Status'] = df['Report.Status'].fillna('Unknown')
```

We can remove irrelevant columns that willl not be usefull in our analysis

In [28]:

```
df.drop(['Accident.Number', 'Location', 'Country', 'Registration.Numbe
```

```
  Cell In[28], line 1
    df.drop(['Accident.Number', 'Location', 'Country', 'Registration.
Number', 'Publication.Date', 'Amateur.Built' ], axis=1, inplace=True

^
SyntaxError: incomplete input
```

Let's explore 'Aircraft.damage' variable

In [ ]:

```
df['Aircraft.damage'].value_counts()
```

In [ ]:

```
# Replace nan values with 'Unknown' in 'Aircraft.damage'
df['Aircraft.damage'] = df['Aircraft.damage'].fillna('Unknown')
```

There are only a few instances before January 1, 1982. Let' s remove them

In [ ]:

```
# Remove instances before 1982
df = df[df['Event.Date'] >= '1982-01-01']
```

We can remove rows with missing values for 'Injury.Severity' (no info on any number of people)

In [ ]:

```
df.dropna(subset=['Injury.Severity'], inplace=True)
```

Let's reexamine the data

In [ ]:

```
df.info()
```

In [ ]:

```
# Calculate the percentage of missing values in each column
(df.isnull().sum() * 100 / len(df)).round(2)
```

Missing values in columns Total.Fatal.Injuries, Total.Serious.Injuries, Total.Minor.Injuries and Total.Uninjured are in the years 2001 to 2007. Replace them with 0.

In [ ]:
```
columns_to_fill = ['Total.Fatal.Injuries', 'Total.Serious.Injuries', '

df[columns_to_fill] = df[columns_to_fill].fillna(0)
```

In [ ]:
```
(df.isnull().sum() * 100 / len(df)).round(2)
```

Cleaning 'Make' variable

In [ ]:
```
# Set option to display all columns
pd.set_option('display.max_rows', None)
df['Make'].value_counts()
```

In [ ]:
```
# Convert all values in 'Make to lower case'
df['Make'] = df['Make'].str.lower()
```

In [ ]:
```
df['Make'].value_counts().nunique()
```

In [ ]:
```
# Find rows where 'cessna' is a substring but not exactly 'cessna'
# cessna_rows = df[df['Make'].str.contains('socata', case=False, na=Fa
# cessna_rows
```

Let's standardize all manufacturers' names and focus on those with at least 100 instances int he dataset

In [ ]:
```
def replace_make(df, make_list):
    """
    Replace all values in the 'Make' column that contain any of the sp

    Parameters:
    df (pd.DataFrame): The DataFrame containing the 'Make' column.
    make_list (list of str): A list of names to standardize.

    Returns:
    pd.DataFrame: The DataFrame with standardized 'Make' values.
    """
    for maker in make_list:
        df.loc[df['Make'].str.contains(maker, case=False, na=False), '
    return df

make_list = ['cessna', 'piper', 'beech', 'boeing', 'mooney', 'grumman'
             'maule', 'champion', 'mcdonnell douglas', 'stinson', 'lusc
             'rockwell', 'enstrom', 'ayres', 'cirrus', 'eurocopter', 's
             'schleicher', 'waco', 'burkhart grob', 'airbus', 'socata',

df = replace_make(df, make_list)
```

Some companies have undergone transitions and are known by more than one name. Let's address those

In [ ]:
```
# Replace all string containing 'bell' that are not 'bellanca' with 'b
df['Make'] = df['Make'].apply(lambda x: 'bell' if isinstance(x, str) a
```

In [ ]:
```
df['Make'] = df['Make'].apply(lambda x: 'schweizer' if isinstance(x, s
```

In [ ]:
```
df.loc[df['Make'].str.contains('havilland', case=False, na=False), 'Ma
```

In [ ]:
```
df.loc[df['Make'].str.contains('hiller', case=False, na=False), 'Make'
```

In [ ]:
```
df.loc[df['Make'].str.contains('fairchild', case=False, na=False), 'Ma
```

In [ ]:
```
df.loc[df['Make'].str.contains('douglas', case=False, na=False), 'Make
```

In [ ]:
```
# the same company
df.loc[df['Make'].str.contains('firefly balloons', case=False, na=Fals
```

In [ ]:
```
df.loc[df['Make'].str.contains('aviat aircraft', case=False, na=False)
```

In [ ]:

```
df.loc[df['Make'].str.contains('ryan aeronautical', case=False, na=Fal
df.loc[df['Make'].str.contains('ryan aeronautics', case=False, na=Fals
df.loc[df['Make'].str.contains('ryan-navion', case=False, na=False), '
```

In [ ]:

```
df.loc[df['Make'].str.contains('helio aircraft ltd', case=False, na=Fa
```

In [ ]:

```
df['Make'].value_counts().nunique()
```

Now let's standardize all 'Model' varieble values

In [ ]:

```
# Convert all values in 'Model' to lower case
df.loc[:,'Model'] = df['Model'].str.lower()

import re
# Remove all whitespaces
df.loc[:, 'Model'] = df['Model'].str.replace(r'\s+', '', regex=True)

# Remove all symbols, leaving only letters and digits
df.loc[:,'Model'] = df['Model'].str.replace(r'[^a-zA-Z0-9]', '', regex

df.head()
```

Let's create new column 'Maker_Model' where we combine the name of the maker and the model

In [ ]:

```
df.loc[:, 'Maker_Model'] = df['Make'] + '_' + df['Model']
```

In [ ]:

```
df['Maker_Model'].value_counts()
```

In [ ]:

```
df['Maker_Model'].value_counts().nunique()
```

In [ ]:

```
df.shape
```

In [ ]:

```
# Calculate the percentage of missing values in each column
(df.isnull().sum() * 100 / len(df)).round(2)
```

Filter out rows where 'Make' value counts are at least 20

In [ ]:

```
df = df.groupby('Make').filter(lambda x: len(x) >= 20)
```

Filter rows where 'Maker_Model' value counts are at least 20

In [ ]:

```
df = df.groupby('Maker_Model').filter(lambda x: len(x) >= 20)
```

The number of distinct Maker_Model's is too large. We used CatGPT to group models based on similarity

In [ ]:

```
mapping_dict = {
    'aero commander_100': 'aero_commander_100',
    'aero commander_500b': 'aero_commander_500',
    'aero commander_s2r': 'aero_commander_S2',
    'aeronca_11ac': 'aeronca_11',
    'aeronca_15ac': 'aeronca_15',
    'aeronca_7ac': 'aeronca_7',
    'aeronca_7bcm': 'aeronca_7',
    'aeronca_7ec': 'aeronca_7',
    'aerospatiale_as350b': 'aerospatiale_as350',
    'aerospatiale_as350d': 'aerospatiale_as350',
    'aerospatiale_sa315b': 'aerospatiale_sa315',
    'aerospatiale_sa316b': 'aerospatiale_sa316',
    'air tractor_at301': 'air_tractor_at300',
    'air tractor_at400': 'air_tractor_at400',
    'air tractor_at401': 'air_tractor_at400',
    'air tractor_at402': 'air_tractor_at400',
    'air tractor_at502': 'air_tractor_at500',
    'air tractor_at502b': 'air_tractor_at500',
    'air tractor_at602': 'air_tractor_at600',
    'air tractor_at802': 'air_tractor_at800',
    'air tractor_at802a': 'air_tractor_at800',
    'alon_a2': 'alon_a2',
    'american_aa1': 'american_aa1',
    'aviat_a1': 'aviat_a1',
    'aviat_a1b': 'aviat_a1',
    'ayres_s2r': 'ayres_s2r',
    'ayres_s2rt34': 'ayres_s2r',
    'balloon works_firefly7': 'balloon_works_firefly',
    'beech_1900c': 'beech_1900',
    'beech_1900d': 'beech_1900',
    'beech_200': 'beech_200',
    'beech_23': 'beech_23',
    'beech_35': 'beech_35',
    'beech_35b33': 'beech_35',
    'beech_35c33': 'beech_35',
    'beech_36': 'beech_36',
    'beech_55': 'beech_55',
    'beech_58': 'beech_58',
    'beech_58p': 'beech_58',
    'beech_60': 'beech_60',
    'beech_76': 'beech_76',
    'beech_77': 'beech_77',
    'beech_95': 'beech_95',
    'beech_95a55': 'beech_95',
    'beech_95b55': 'beech_95',
    'beech_95b55t42a': 'beech_95',
    'beech_95c55': 'beech_95',
    'beech_99': 'beech_99',
    'beech_a23': 'beech_23',
    'beech_a2319': 'beech_23',
    'beech_a2324': 'beech_23',
    'beech_a23a': 'beech_23',
    'beech_a24r': 'beech_24',
    'beech_a35': 'beech_35',
    'beech_a36': 'beech_36',
    'beech_a36tc': 'beech_36',
    'beech_b19': 'beech_19',
    'beech_b200': 'beech_200',
    'beech_b23': 'beech_23',
    'beech_b24r': 'beech_24',
    'beech_b35': 'beech_35',
    'beech_b36tc': 'beech_36',
    'beech_b60': 'beech_60',
    'beech_b90': 'beech_90',
    'beech_be58': 'beech_58',
    'beech_c23': 'beech_23',
    'beech_c24r': 'beech_24',
    'beech_c35': 'beech_35',
    'beech_c45h': 'beech_45',
    'beech_c90': 'beech_90',
    'beech_c99': 'beech_99',
    'beech_d18s': 'beech_18',
    'beech_d35': 'beech_35',
    'beech_d55': 'beech_55',
    'beech_e18s': 'beech_18',
    'beech_e35': 'beech_35',
    'beech_e55': 'beech_55',
    'beech_e90': 'beech_90',
    'beech_f33a': 'beech_33',
    'beech_f35': 'beech_35',
    'beech_g18s': 'beech_18',
    'beech_g35': 'beech_35',
    'beech_h35': 'beech_35',
    'beech_j35': 'beech_35',
    'beech_k35': 'beech_35',
    'beech_m35': 'beech_35',
    'beech_n35': 'beech_35',
    'beech_p35': 'beech_35',
    'beech_s35': 'beech_35',
    'beech_v35': 'beech_35',
    'beech_v35a': 'beech_35',
    'beech_v35b': 'beech_35',
    'bell_206': 'bell_206',
    'bell_206b': 'bell_206',
    'bell_206b3': 'bell_206',
    'bell_206biii': 'bell_206',
    'bell_206l': 'bell_206',
    'bell_206l1': 'bell_206',
    'bell_206l3': 'bell_206',
    'bell_206l4': 'bell_206',
    'bell_212': 'bell_212',
    'bell_407': 'bell_407',
    'bell_47d1': 'bell_47',
    'bell_47g': 'bell_47',
    'bell_47g2': 'bell_47',
    'bell_47g2a': 'bell_47',
    'bell_47g3b': 'bell_47',
    'bell_47g3b1': 'bell_47',
    'bell_47g3b2': 'bell_47',
    'bell_47g4a': 'bell_47',
    'bell_47g5': 'bell_47',
    'bell_oh58a': 'bell_oh58',
    'bell_uh1b': 'bell_uh1',
    'bell_uh1h': 'bell_uh1',
    'bellanca_1730': 'bellanca_1730',
    'bellanca_1730a': 'bellanca_1730',
    'bellanca_1731atc': 'bellanca_1731',
    'bellanca_7eca': 'bellanca_7',
    'bellanca_7gcaa': 'bellanca_7',
    'bellanca_7gcbc': 'bellanca_7',
    'bellanca_7kcab': 'bellanca_7',
    'bellanca_8gcbc': 'bellanca_8',
    'bellanca_8kcab': 'bellanca_8',
    'boeing_727200': 'boeing_727',
    'boeing_737': 'boeing_737',
    'boeing_737200': 'boeing_737',
    'boeing_737300': 'boeing_737',
    'boeing_7377h4': 'boeing_737',
    'boeing_a75': 'boeing_a75',
    'boeing_a75n1': 'boeing_a75',
    'boeing_a75n1pt17': 'boeing_a75',
    'boeing_b75n1': 'boeing_b75',
    'boeing_e75': 'boeing_e75',
    'cessna_120': 'cessna_120',
    'cessna_140': 'cessna_140',
    'cessna_140a': 'cessna_140',
    'cessna_150': 'cessna_150',
    'cessna_150c': 'cessna_150',
    'cessna_150d': 'cessna_150',
    'cessna_150e': 'cessna_150',
    'cessna_150f': 'cessna_150',
    'cessna_150g': 'cessna_150',
    'cessna_150h': 'cessna_150',
    'cessna_150j': 'cessna_150',
    'cessna_150k': 'cessna_150',
    'cessna_150l': 'cessna_150',
    'cessna_150m': 'cessna_150',
    'cessna_152': 'cessna_152',
    'cessna_152ii': 'cessna_152',
    'cessna_162': 'cessna_162',
    'cessna_170': 'cessna_170',
    'cessna_170a': 'cessna_170',
    'cessna_170b': 'cessna_170',
    'cessna_172': 'cessna_172',
    'cessna_172a': 'cessna_172',
    'cessna_172b': 'cessna_172',
    'cessna_172c': 'cessna_172',
    'cessna_172d': 'cessna_172',
    'cessna_172e': 'cessna_172',
    'cessna_172f': 'cessna_172',
    'cessna_172g': 'cessna_172',
    'cessna_172h': 'cessna_172',
    'cessna_172i': 'cessna_172',
    'cessna_172k': 'cessna_172',
    'cessna_172l': 'cessna_172',
    'cessna_172m': 'cessna_172',
    'cessna_172n': 'cessna_172',
    'cessna_172p': 'cessna_172',
    'cessna_172r': 'cessna_172',
    'cessna_172rg': 'cessna_172',
    'cessna_172s': 'cessna_172',
    'cessna_172xp': 'cessna_172',
    'cessna_175': 'cessna_175',
    'cessna_175a': 'cessna_175',
    'cessna_177': 'cessna_177',
    'cessna_177a': 'cessna_177',
    'cessna_177b': 'cessna_177',
    'cessna_177rg': 'cessna_177',
    'cessna_180': 'cessna_180',
    'cessna_180a': 'cessna_180',
    'cessna_180b': 'cessna_180',
    'cessna_180h': 'cessna_180',
    'cessna_180j': 'cessna_180',
    'cessna_180k': 'cessna_180',
    'cessna_182': 'cessna_182',
    'cessna_182a': 'cessna_182',
    'cessna_182b': 'cessna_182',
    'cessna_182c': 'cessna_182',
    'cessna_182d': 'cessna_182',
    'cessna_182e': 'cessna_182',
    'cessna_182f': 'cessna_182',
    'cessna_182g': 'cessna_182',
    'cessna_182h': 'cessna_182',
    'cessna_182j': 'cessna_182',
    'cessna_182k': 'cessna_182',
    'cessna_182l': 'cessna_182',
    'cessna_182m': 'cessna_182',
    'cessna_182n': 'cessna_182',
    'cessna_182p': 'cessna_182',
    'cessna_182q': 'cessna_182',
    'cessna_182r': 'cessna_182',
    'cessna_182rg': 'cessna_182',
    'cessna_182s': 'cessna_182',
    'cessna_182t': 'cessna_182',
    'cessna_185': 'cessna_185',
    'cessna_185e': 'cessna_185',
    'cessna_185f': 'cessna_185',
    'cessna_188': 'cessna_188',
    'cessna_188a': 'cessna_188',
    'cessna_188b': 'cessna_188',
    'cessna_195': 'cessna_195',
    'cessna_195a': 'cessna_195',
    'cessna_195b': 'cessna_195',
    'cessna_205': 'cessna_205',
    'cessna_206': 'cessna_206',
    'cessna_206g': 'cessna_206',
    'cessna_207': 'cessna_207',
    'cessna_207a': 'cessna_207',
    'cessna_208': 'cessna_208',
    'cessna_208b': 'cessna_208',
    'cessna_210': 'cessna_210',
    'cessna_210a': 'cessna_210',
    'cessna_210b': 'cessna_210',
    'cessna_210d': 'cessna_210',
    'cessna_210e': 'cessna_210',
    'cessna_210l': 'cessna_210',
    'cessna_210m': 'cessna_210',
    'cessna_210n': 'cessna_210',
    'cessna_305a': 'cessna_305',
    'cessna_310': 'cessna_310',
    'cessna_310c': 'cessna_310',
    'cessna_310d': 'cessna_310',
    'cessna_310j': 'cessna_310',
    'cessna_310k': 'cessna_310',
    'cessna_310l': 'cessna_310',
    'cessna_310n': 'cessna_310',
    'cessna_310q': 'cessna_310',
    'cessna_310r': 'cessna_310',
    'cessna_320': 'cessna_320',
    'cessna_337': 'cessna_337',
    'cessna_340': 'cessna_340',
    'cessna_340a': 'cessna_340',
    'cessna_401': 'cessna_401',
    'cessna_402': 'cessna_402',
    'cessna_402b': 'cessna_402',
    'cessna_402c': 'cessna_402',
    'cessna_404': 'cessna_404',
    'cessna_414': 'cessna_414',
    'cessna_414a': 'cessna_414',
    'cessna_421': 'cessna_421',
    'cessna_421b': 'cessna_421',
    'cessna_421c': 'cessna_421',
    'cessna_425': 'cessna_425',
    'cessna_441': 'cessna_441',
    'cessna_550': 'cessna_550',
    'cessna_a150k': 'cessna_150',
    'cessna_a150l': 'cessna_150',
    'cessna_a150m': 'cessna_150',
    'cessna_a152': 'cessna_152',
    'cessna_a185': 'cessna_185',
    'cessna_a185e': 'cessna_185',
    'cessna_a185f': 'cessna_185',
    'cessna_a188': 'cessna_188',
    'cessna_a188b': 'cessna_188',
    'cessna_c152': 'cessna_152',
    'cessna_c172': 'cessna_172',
    'cessna_p210': 'cessna_210',
    'cessna_p210n': 'cessna_210',
    'cessna_r172k': 'cessna_172',
    'cessna_r182': 'cessna_182',
    'cessna_t182t': 'cessna_182',
    'cessna_t188c': 'cessna_188',
    'cessna_t206h': 'cessna_206',
    'cessna_t207a': 'cessna_207',
    'cessna_t210': 'cessna_210',
    'cessna_t210f': 'cessna_210',
    'cessna_t210l': 'cessna_210',
    'cessna_t210m': 'cessna_210',
    'cessna_t210n': 'cessna_210',
    'cessna_t303': 'cessna_303',
    'cessna_t310r': 'cessna_310',
    'cessna_t337g': 'cessna_337',
    'cessna_tr182': 'cessna_182',
    'cessna_tu206': 'cessna_206',
    'cessna_tu206f': 'cessna_206',
    'cessna_tu206g': 'cessna_206',
    'cessna_u206': 'cessna_206',
    'cessna_u206f': 'cessna_206',
    'cessna_u206g': 'cessna_206',
    'north american_t6g': 'north_american_t6',
    'champion_7ec': 'champion_7',
    'champion_7eca': 'champion_7',
    'champion_7fc': 'champion_7',
    'champion_7gcaa': 'champion_7',
    'champion_7gcbc': 'champion_7',
    'champion_7kcab': 'champion_7',
    'champion_8gcbc': 'champion_8',
    'champion_8kcab': 'champion_8',
    'cirrus_sr20': 'cirrus_sr20',
    'cirrus_sr22': 'cirrus_sr22',
    'consolidated aeronautics inc._lakela4200': 'consolidated_lakela42
    'de havilland_dhc2': 'de_havilland_dhc2',
    'de havilland_dhc3': 'de_havilland_dhc3',
    'diamond aircraft ind inc_da20c1': 'diamond_da20',
    'diamond aircraft ind inc_da40': 'diamond_da40',
    'eagle aircraft co._dw1': 'eagle_dw1',
    'enstrom_280c': 'enstrom_280',
    'enstrom_f28a': 'enstrom_f28',
    'enstrom_f28c': 'enstrom_f28',
    'enstrom_f28f': 'enstrom_f28',
    'ercoupe_415c': 'ercoupe_415',
    'ercoupe_415d': 'ercoupe_415',
    'eurocopter_as350b2': 'eurocopter_as350',
    'eurocopter_as350b3': 'eurocopter_as350',
    'fairchild hiller_fh1100': 'fairchild_fh1100',
    'fairchild hiller_sa227ac': 'fairchild_sa227',
    'fairchild hiller_uh12b': 'fairchild_uh12',
    'fairchild hiller_uh12c': 'fairchild_uh12',
    'fairchild hiller_uh12d': 'fairchild_uh12',
    'fairchild hiller_uh12e': 'fairchild_uh12',
    'flight design gmbh_ctls': 'flight_design_ctls',
    'globe_gc1b': 'globe_gc1',
    'great lakes_2t1a2': 'great_lakes_2t1',
    'grumman_aa1b': 'grumman_aa1',
    'grumman_aa1c': 'grumman_aa1',
    'grumman_aa5': 'grumman_aa5',
    'grumman_aa5a': 'grumman_aa5',
    'grumman_aa5b': 'grumman_aa5',
    'grumman_g164': 'grumman_g164',
    'grumman_g164a': 'grumman_g164',
    'grumman_g164b': 'grumman_g164',
    'gulfstream_aa5a': 'gulfstream_aa5',
    'gulfstream_aa5b': 'gulfstream_aa5',
    'gulfstream_g164b': 'gulfstream_g164',
    'helio_h295': 'helio_h295',
    'hughes_269a': 'hughes_269',
    'hughes_269b': 'hughes_269',
    'hughes_269c': 'hughes_269',
    'hughes_369': 'hughes_369',
    'hughes_369d': 'hughes_369',
    'hughes_369e': 'hughes_369',
    'hughes_369hs': 'hughes_369',
    'hughes_oh6a': 'hughes_oh6',
    'lake_la4': 'lake_la4',
    'lake_la4200': 'lake_la4200',
    'let_blanikl13': 'let_blanik_l13',
    'let_l13': 'let_l13',
    'luscombe_8': 'luscombe_8',
    'luscombe_8a': 'luscombe_8',
    'luscombe_8e': 'luscombe_8',
    'luscombe_8f': 'luscombe_8',
    'maule_m4': 'maule_m4',
    'maule_m4220c': 'maule_m4',
    'maule_m5': 'maule_m5',
    'maule_m5210c': 'maule_m5',
    'maule_m5235c': 'maule_m5',
    'maule_m6235': 'maule_m6',
    'maule_mx7': 'maule_mx7',
    'mcdonnell douglas_369e': 'mcdonnell_douglas_369',
    'mcdonnell douglas_dc1010': 'mcdonnell_douglas_dc10',
    'mcdonnell douglas_dc931': 'mcdonnell_douglas_dc9',
    'mcdonnell douglas_dc932': 'mcdonnell_douglas_dc9',
    'mcdonnell douglas_dc982': 'mcdonnell_douglas_dc9',
    'mcdonnell douglas_md88': 'mcdonnell_douglas_md88',
    'mitsubishi_mu2b60': 'mitsubishi_mu2',
    'mooney_m20': 'mooney_m20',
    'mooney_m20a': 'mooney_m20',
    'mooney_m20b': 'mooney_m20',
    'mooney_m20c': 'mooney_m20',
    'mooney_m20e': 'mooney_m20',
    'mooney_m20f': 'mooney_m20',
    'mooney_m20j': 'mooney_m20',
    'mooney_m20k': 'mooney_m20',
    'mooney_m20m': 'mooney_m20',
    'navion_a': 'navion_a',
    'north american_at6d': 'north_american_at6',
    'north american_navion': 'north_american_navion',
    'north american_p51d': 'north_american_p51',
    'north american_snj5': 'north_american_snj',
    'north american_t6': 'north_american_t6',
    'piper_j3': 'piper_j3',
    'piper_j3c': 'piper_j3',
    'piper_j3c65': 'piper_j3',
    'piper_j5a': 'piper_j5',
    'piper_pa11': 'piper_pa11',
    'piper_pa12': 'piper_pa12',
    'piper_pa14': 'piper_pa14',
    'piper_pa16': 'piper_pa16',
    'piper_pa18': 'piper_pa18',
 'piper_pa18135': 'piper_pa18',
 'piper_pa18150': 'piper_pa18',
 'piper_pa18160': 'piper_pa18',
 'piper_pa18a': 'piper_pa18',
 'piper_pa18a150': 'piper_pa18',
 'piper_pa20': 'piper_pa20',
 'piper_pa22': 'piper_pa22',
 'piper_pa22108': 'piper_pa22',
 'piper_pa22135': 'piper_pa22',
 'piper_pa22150': 'piper_pa22',
 'piper_pa22160': 'piper_pa22',
 'piper_pa23': 'piper_pa23',
 'piper_pa23150': 'piper_pa23',
 'piper_pa23160': 'piper_pa23',
 'piper_pa23250': 'piper_pa23',
 'piper_pa24': 'piper_pa24',
 'piper_pa24180': 'piper_pa24',
 'piper_pa24250': 'piper_pa24',
 'piper_pa24260': 'piper_pa24',
 'piper_pa25': 'piper_pa25',
 'piper_pa25235': 'piper_pa25',
 'piper_pa25260': 'piper_pa25',
 'piper_pa28': 'piper_pa28',
 'piper_pa28140': 'piper_pa28',
 'piper_pa28150': 'piper_pa28',
 'piper_pa28151': 'piper_pa28',
 'piper_pa28160': 'piper_pa28',
 'piper_pa28161': 'piper_pa28',
 'piper_pa28180': 'piper_pa28',
 'piper_pa28181': 'piper_pa28',
 'piper_pa28235': 'piper_pa28',
 'piper_pa28236': 'piper_pa28',
 'piper_pa28r': 'piper_pa28r',
 'piper_pa28r180': 'piper_pa28r',
 'piper_pa28r200': 'piper_pa28r',
 'piper_pa28r201': 'piper_pa28r',
 'piper_pa28r201t': 'piper_pa28r',
 'piper_pa28rt201': 'piper_pa28r',
 'piper_pa28rt201t': 'piper_pa28r',
 'piper_pa30': 'piper_pa30',
 'piper_pa31': 'piper_pa31',
 'piper_pa31310': 'piper_pa31',
 'piper_pa31325': 'piper_pa31',
 'piper_pa31350': 'piper_pa31',
 'piper_pa31p': 'piper_pa31',
 'piper_pa31t': 'piper_pa31',
 'piper_pa32': 'piper_pa32',
 'piper_pa32260': 'piper_pa32',
 'piper_pa32300': 'piper_pa32',
 'piper_pa32301': 'piper_pa32',
 'piper_pa32r': 'piper_pa32r',
 'piper_pa32r300': 'piper_pa32r',
 'piper_pa32r301': 'piper_pa32r',
 'piper_pa32r301t': 'piper_pa32r',
 'piper_pa32rt': 'piper_pa32rt',
 'piper_pa32rt300': 'piper_pa32rt',
 'piper_pa32rt300t': 'piper_pa32rt',
 'piper_pa34': 'piper_pa34',
 'piper_pa34200': 'piper_pa34',
 'piper_pa34200t': 'piper_pa34',
 'piper_pa34220t': 'piper_pa34',
 'piper_pa36': 'piper_pa36',
 'piper_pa36285': 'piper_pa36',
 'piper_pa36300': 'piper_pa36',
 'piper_pa36375': 'piper_pa36',
 'piper_pa38': 'piper_pa38',
 'piper_pa38112': 'piper_pa38',
 'piper_pa44180': 'piper_pa44',
 'piper_pa46': 'piper_pa46',
 'piper_pa46310p': 'piper_pa46',
 'piper_pa46350p': 'piper_pa46',
 'piper_pa46500tp': 'piper_pa46',
 'piper_pa60': 'piper_pa60',
 'piper_pa60601p': 'piper_pa60',
 'pitts_s2b': 'pitts_s2',
 'republic_rc3': 'republic_rc3',
 'robinson_r22': 'robinson_r22',
 'robinson_r22a': 'robinson_r22',
 'robinson_r22b': 'robinson_r22',
 'robinson_r22beta': 'robinson_r22',
 'robinson_r44': 'robinson_r44',
 'robinson_r44ii': 'robinson_r44',
 'robinson_r66': 'robinson_r66',
 'rockwell_112a': 'rockwell_112',
 'rockwell_114': 'rockwell_114',
 'rockwell_s2r': 'rockwell_s2r',
 'ryan_navion': 'ryan_navion',
 'ryan_st3kr': 'ryan_st3',
 'schleicher_asw20': 'schleicher_asw20',
 'schweizer_269c': 'schweizer_269',
 'schweizer_269c1': 'schweizer_269',
 'schweizer_g164a': 'schweizer_g164',
 'schweizer_g164b': 'schweizer_g164',
 'schweizer_sgs134': 'schweizer_sgs',
 'schweizer_sgs233': 'schweizer_sgs',
 'schweizer_sgs233a': 'schweizer_sgs',
 'sikorsky_s76a': 'sikorsky_s76',
 'socata_tbm700': 'socata_tbm700',
 'stinson_108': 'stinson_108',
 'stinson_1081': 'stinson_108',
    'stinson_1082': 'stinson_108',
    'stinson_1083': 'stinson_108',
    'swearingen_sa226tc': 'swearingen_sa226',
    'taylorcraft_bc12d': 'taylorcraft_bc12',
    'taylorcraft_bl65': 'taylorcraft_bl65',
    'taylorcraft_f19': 'taylorcraft_f19',
    'waco_upf7': 'waco_upf7',
    'weatherly_201b': 'weatherly_201',
    'wsk pzl mielec_m18a': 'wsk_pzl_m18'
}

df['General_Maker_Model'] = df['Maker_Model'].map(mapping_dict)
```

In [ ]:

```
general_unique = (df['General_Maker_Model'].unique()).tolist()
display(len(general_unique))
# general_unique = sorted(general_unique)
# general_unique
```

In [ ]:

```
df[df['Maker_Model'] == 'beech_s35']
df[df['Maker_Model'] == 'cessna_t210m']
```

Not every aircraft model is still in use. We used ChatGPT to determine the vintage models and remove ethem from the dataset

In [ ]:

```
# Aircraft that are no longer in use (based on ChatGPT)
vintage = ['aero_commander_100', 'aero_commander_500', 'aero_commander
           'aeronca_7', 'alon_a2', 'bellanca_1730', 'bellanca_1731', '
           'boeing_b75', 'boeing_e75', 'globe_gc1', 'great_lakes_2t1',
           'schweizer_sgs134', 'schweizer_sgs233', 'taylorcraft_bc12d'
           'waco_upf7']

vintage_df = df[df['General_Maker_Model'].isin(vintage)]

vintage_df.shape
```

In [ ]:

```
# Keep aircraft that are still in use in df
df = df[~df['General_Maker_Model'].isin(vintage)]

df.shape
```

Lets calculate the number of people on board each flight for future statistics calculaations

In [ ]:
```python
df['Total_On_Board'] = (df['Total.Fatal.Injuries'] +
                        df['Total.Serious.Injuries'] +
                        df['Total.Minor.Injuries'] +
                        df['Total.Uninjured'])
```

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

In [ ]:
```python
df['Total_On_Board'].value_counts().sort_index()
```

In [ ]:
```python
df[df['Total_On_Board'] == 0]
```

Remove instances where the aircraft was unoccupied when an accident happened

In [ ]:
```python
df = df[df['Total_On_Board'] != 0]
```

Not every General_Maker_Model has enough instances to conduct statistical analysis. Let's remove those with less than 20 instances

In [ ]:
```python
# Filter rows where 'General_Maker_Model' value counts are at least 20
df = df.groupby('General_Maker_Model').filter(lambda x: len(x) >= 20)
```

In [ ]:
```python
# Set display options to show all rows
pd.set_option('display.max_rows', None)

df['General_Maker_Model'].value_counts()
```

Let's scheck how many people were on board each fligh

In [ ]:
```python
df['Total_On_Board'].value_counts().sort_index()
```

In [ ]:
```python
df[df['Total_On_Board'] == 408]
```

In [ ]:
```python
# Calculate the percentage of missing values in each column
(df.isnull().sum() * 100 / len(df)).round(2)
```

In [ ]:
```python
df.shape
```

There are no missing values in the dataset. Let's save it to AviationData_CLEAN.csv

In [ ]:
```python
df.to_csv('AviationData_CLEAN.csv', index=True)
```