Compiling Data on Aviation with the Least Amount of Risk

Project Goals

Our company is interested in purchasing and operating aircrafts in order to expand and diversify our portfolio. Our goal is to determine which aircrafts have the lowest risk for our new business endeavors. We decided that these business endeavors can be broken up into three categories:

- Passenger transportation
- Cargo transportation
- Private enterprises

Our main goal for this project is to recommend an aircraft with the lowest risk for each of these business endeavors.

Data

The data used to reach our conclusions was taken from the following:

https://www.kaggle.com/datasets/khsamaha/aviation-accident-database-synopses

This is a database from the NTSB (National Transportation Safety Board), which contains accidents and incidents from 1962 up to 2023. The dataset is limited to flights originating from the United States and its territories, and select events over international waters.

Start by importing libraries we will need to use.

In [1]:
```
# First need import statements

import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

pd.set_option('display.max_colwidth', None)
```

Then let's import the data.

In [2]:
```
df = pd.read_csv("./Data/AviationData.csv", encoding='latin-1', low_memory = False)
```

Methods

Exploring The Data

Let's take a look at what we've got

In [3]:
```
# Take a look

print("These are the possible inputs for Aircraft Damage")
print(df['Aircraft.damage'].value_counts().index)
print()

print("These are the possible inputs for Injury Severity (already looked at)")
print(df['Injury.Severity'].value_counts().index)
```

```python
print()

print("These are the possible inputs for Aircraft.Category")
print(df['Aircraft.Category'].value_counts().index)
print(df['Aircraft.Category'].value_counts().values)
print()

print("These are the possible inputs for investigation type")
print(df['Investigation.Type'].value_counts().index)
print()

print("These are the possible inputs for Amateur.Built")
print(df['Amateur.Built'].value_counts().index)
print(df['Amateur.Built'].value_counts().values)

print("These are the possible inputs for Report.Status")
print(df['Report.Status'].value_counts().index)
print(df['Report.Status'].value_counts().values)

print("These are the possible inputs for Weather.Condition")
print(df['Weather.Condition'].value_counts().index)
print(df['Weather.Condition'].value_counts().values)

print("These are the possible inputs for Accident.Number")
print(df['Accident.Number'].value_counts().index)
print(df['Accident.Number'].value_counts().values)


print("These are the possible inputs for Airport.Code")
print(df['Airport.Code'].value_counts().index)
print(df['Airport.Code'].value_counts().values)

print("These are the possible inputs for FAR.Description")
print(df['FAR.Description'].value_counts().index)
print(df['FAR.Description'].value_counts().values)

print()
```

```
These are the possible inputs for Aircraft Damage
Index(['Substantial', 'Destroyed', 'Minor', 'Unknown'], dtype='object')

These are the possible inputs for Injury Severity (already looked at)
Index(['Non-Fatal', 'Fatal(1)', 'Fatal', 'Fatal(2)', 'Incident', 'Fatal(3)',
       'Fatal(4)', 'Fatal(5)', 'Minor', 'Serious',
       ...
       'Fatal(265)', 'Fatal(54)', 'Fatal(256)', 'Fatal(206)', 'Fatal(96)',
       'Fatal(89)', 'Fatal(153)', 'Fatal(228)', 'Fatal(26)', 'Fatal(117)'],
      dtype='object', length=109)

These are the possible inputs for Aircraft.Category
Index(['Airplane', 'Helicopter', 'Glider', 'Balloon', 'Gyrocraft',
       'Weight-Shift', 'Powered Parachute', 'Ultralight', 'Unknown', 'WSFT',
       'Powered-Lift', 'Blimp', 'UNK', 'Rocket', 'ULTR'],
      dtype='object')
[27617  3440   508   231   173   161    91    30    14     9     5     4
     2     1     1]

These are the possible inputs for investigation type
Index(['Accident', 'Incident'], dtype='object')

These are the possible inputs for Amateur.Built
Index(['No', 'Yes'], dtype='object')
[80312  8475]
These are the possible inputs for Report.Status
Index(['Probable Cause', 'Foreign', '<br /><br />', 'Factual',
       'The pilot's failure to maintain directional control during the landing roll.',
       'A loss of engine power for undetermined reasons.',
       'The pilot's failure to maintain directional control during landing.',
       'A total loss of engine power for undetermined reasons.',
```

```
       'The loss of engine power for undetermined reasons.',
       'The pilot□s failure to maintain directional control during the landing roll.\r
\n\r',
       ...
       'The crew leader's decision to throw a chainsaw bar into the moving arc of the
main rotor blade.',
       'The student pilot□s failure to maintain adequate airspeed while maneuvering fo
r landing, which resulted in a stall/spin. Contributing to the accident were the pilo
t's distraction and anxiety from a reported failure of the airplane's electrical syste
m.',
       'Both pilots□ inadequate visual lookout and failure to see and avoid each other
while in level cruise flight, which resulted in a midair collision. Contributing to th
e accident was the obscuration of the pilot□s visibility due to sun glare. \r',
       'The student pilot's failure to maintain directional control during takeoff and
the flight instructor's inadequate supervision of the flight and delay in remedial act
ion.',
       'The brake application for unknown reasons resulting in rupture of the left mai
n landing gear tire and tube, and subsequent loss of directional control.',
       'A partial loss of engine power during cruise flight for reasons that could not
be determined because postaccident examination of the airframe and engine did not reve
al any anomalies that would have precluded normal operation. \r\n\r',
       'The pilot's failure to maintain airplane control following an engine problem f
or reasons that could not be determined because postaccident examination of both engin
es and the rudder revealed no malfunctions or anomalies that would have precluded norm
al operation.',
       'the pilot's decision to continue the flight with a known fuel imbalance condit
ion that resulted in a loss of lateral control when the imbalance exceeded the known c
apabilities of the airplane.  The fuel imbalance was due to incomplete refueling and a
n ineffective preflight inspection by the pilot.  The imbalance was further complicate
d by an incorrectly assembled fuel transfer valve and motor combination.  \r\n\r\nCont
ributing to the severity of the accident was the pilot's delayed decision to eject pri
or to exceeding the ejection seat envelope.  Also contributing to the accident was (1)
the Navy's oversight environment, which did not require airman, aircraft, and risk man
agement controls or standards expected of a commercial civil aviation operation, and
(2) ATAC's organizational environment, which did not include CRM training to promote g
ood aeronautical decision-making and ORM guidance to mitigate hazards.  Also contribut
ing to the accident were the design features of the airplane, which were typical of it
s generation, including the lack of accurate fuel quantity indications, the design of
the fuel transfer valve; and the maintenance program's lack of clearly documented proc
edures and type-specific training for the Hunter.',
       'The pilot□s improper preflight planning and in-flight fuel management, which r
esulted in fuel starvation and a total loss of engine power. Also causal to the accide
nt was the pilot□s failure to see and avoid the wires on the airport boundary.\r',
       'A collision with a deer on the runway during the takeoff roll.\r\n\r'],
      dtype='object', length=17075)
[61754  1999   167 ...     1     1     1]
These are the possible inputs for Weather.Condition
Index(['VMC', 'IMC', 'UNK', 'Unk'], dtype='object')
[77303  5976   856   262]
These are the possible inputs for Accident.Number
Index(['ERA22LA103', 'DCA22WA158', 'GAA22WA241', 'ERA22LA364', 'WPR22LA201',
       'ERA22LA119', 'DCA22WA204', 'DCA22WA130', 'DCA22LA201', 'DCA22WA089',
       ...
       'NYC92LA033', 'MIA84FA169', 'ANC88FA056', 'LAX96LA314', 'GAA16CA311',
       'MIA97LA129', 'CEN14LA147', 'CHI89DEV03', 'FTW84LA155A', 'ATL05CA160'],
      dtype='object', length=88863)
[2 2 2 ... 1 1 1]
These are the possible inputs for Airport.Code
Index(['NONE', 'PVT', 'APA', 'ORD', 'MRI', 'None', 'DEN', 'OSH', 'BJC', 'VNY',
       ...
       'LKST', 'OK63', 'TNC', '7M8', 'D02', 'KFSK', '4K5', '14M', '59J',
       'MO42'],
      dtype='object', length=10375)
[1488  485  160 ...    1    1    1]
These are the possible inputs for FAR.Description
Index(['091', 'Part 91: General Aviation', 'NUSN', 'NUSC', '137', '135', '121',
       'Part 137: Agricultural', 'UNK', 'Part 135: Air Taxi & Commuter',
       'PUBU', '129', 'Part 121: Air Carrier', '133', 'Part 129: Foreign',
       'Non-U.S., Non-Commercial', 'Non-U.S., Commercial',
       'Part 133: Rotorcraft Ext. Load', 'Unknown', 'Public Use', '091K',
       'ARMF', '125', 'Part 125: 20+ Pax,6000+ lbs', '107', 'Public Aircraft',
       '103', 'Armed Forces', 'Part 91 Subpart K: Fractional',
```

```
       'Part 91F: Special Flt Ops.', '437'],
      dtype='object')
[18221  6486  1584  1013  1010   746   679   437   371   298   253   246
   165   107   100    97    93    32    22    19    14     8     5     5
     4     2     2     1     1     1     1]
```

What is the shape of the dataframe?

This is the dimensions of the matrix, in (row, column) format

In [4]:
```
df.shape
```

Out[4]:
```
(88889, 31)
```

Now, let's look at the features (columns) of the data.

In [5]:
```
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
```

Poke around a bit more

In [6]:
```
# Take a look at the first 5
print("Take a look at the first 5")
print(df.head())
print()

print("Take a look at some statistics")
print(df.describe())
print()
```

```
Take a look at the first 5
         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]

Take a look at some statistics
       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
```

Let's take a look at how many NaN's there are:

In [7]:
```
# Now let's check out how many NaN's there are

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

Out[7]:
```
Event.Id                     0
Investigation.Type           0
Accident.Number              0
Event.Date                   0
Location                    52
Country                    226
Latitude                 54507
```

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

We will have to clean up some null values, change some names, and convert a datetime for our exploratory data analysis (EDA)

Cleaning The Data

Handling the NaN's for the Make and Model Columns

We are going to drop the rows for the missing make and model values This is because we can't get any useful data from these if we don't know the make and the model!

In [8]:
```
# First make a copy of the original data
df_clean = df.copy()
df_clean
```

Out[8]:

	Event.Id	Investigation.Type	Accident.Number	Event.Date	Location	Country	Latitude	Lo
0	20001218X45444	Accident	SEA87LA080	1948-10-24	MOOSE CREEK, ID	United States	NaN	
1	20001218X45447	Accident	LAX94LA336	1962-07-19	BRIDGEPORT, CA	United States	NaN	
2	20061025X01555	Accident	NYC07LA005	1974-08-30	Saltville, VA	United States	36.922223	-8
3	20001218X45448	Accident	LAX96LA321	1977-06-19	EUREKA, CA	United States	NaN	
4	20041105X01764	Accident	CHI79FA064	1979-08-02	Canton, OH	United States	NaN	
...	...	...	...	...	...	...	...	
88884	20221227106491	Accident	ERA23LA093	2022-12-26	Annapolis, MD	United States	NaN	
88885	20221227106494	Accident	ERA23LA095	2022-12-26	Hampton, NH	United States	NaN	
88886	20221227106497	Accident	WPR23LA075	2022-12-26	Payson, AZ	United States	341525N	1

88887	20221227106498	Accident	WPR23LA076	2022-12-26	Morgan, UT	United States	NaN
88888	20221230106513	Accident	ERA23LA097	2022-12-29	Athens, GA	United States	NaN

88889 rows × 31 columns

Drop the Missing Values from the Make and Model columns

In [9]:
```
df_clean = df_clean.dropna(subset=['Make', 'Model'])
```

Take a look and see if it worked.

In [10]:
```
df_clean.isna().sum()
```

Out[10]:
```
Event.Id                     0
Investigation.Type           0
Accident.Number              0
Event.Date                   0
Location                    52
Country                    225
Latitude                 54430
Longitude                54439
Airport.Code             38564
Airport.Name             36031
Injury.Severity            979
Aircraft.damage           3172
Aircraft.Category        56532
Registration.Number       1275
Make                         0
Model                        0
Amateur.Built               99
Number.of.Engines         6023
Engine.Type               7006
FAR.Description          56841
Schedule                 76245
Purpose.of.flight         6138
Air.carrier              72168
Total.Fatal.Injuries     11386
Total.Serious.Injuries   12490
Total.Minor.Injuries     11914
Total.Uninjured           5897
Weather.Condition         4439
Broad.phase.of.flight    27094
Report.Status             6335
Publication.Date         13765
dtype: int64
```

Let's also Convert the Name of the Columns to Something Cleaner

In [11]:
```
df_clean.columns = df_clean.columns.str.strip().str.lower().str.replace('.', '_')
```

Comparing the two Data Frames

In [12]:
```
print("This is the original DataFrame:")
print()
print(df.info())
print()
print("This is the cleaned DataFrame:")
print()
print(df_clean.info())
print()
```

```
This is the original DataFrame:

<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
None

This is the cleaned DataFrame:

<class 'pandas.core.frame.DataFrame'>
Int64Index: 88777 entries, 0 to 88888
Data columns (total 31 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   event_id                88777 non-null  object
 1   investigation_type      88777 non-null  object
 2   accident_number         88777 non-null  object
 3   event_date              88777 non-null  object
 4   location                88725 non-null  object
 5   country                 88552 non-null  object
 6   latitude                34347 non-null  object
 7   longitude               34338 non-null  object
 8   airport_code            50213 non-null  object
 9   airport_name            52746 non-null  object
 10  injury_severity         87798 non-null  object
 11  aircraft_damage         85605 non-null  object
 12  aircraft_category       32245 non-null  object
 13  registration_number     87502 non-null  object
 14  make                    88777 non-null  object
 15  model                   88777 non-null  object
```

```
 16  amateur_built           88678 non-null  object
 17  number_of_engines       82754 non-null  float64
 18  engine_type             81771 non-null  object
 19  far_description         31936 non-null  object
 20  schedule                12532 non-null  object
 21  purpose_of_flight       82639 non-null  object
 22  air_carrier             16609 non-null  object
 23  total_fatal_injuries    77391 non-null  float64
 24  total_serious_injuries  76287 non-null  float64
 25  total_minor_injuries    76863 non-null  float64
 26  total_uninjured         82880 non-null  float64
 27  weather_condition       84338 non-null  object
 28  broad_phase_of_flight   61683 non-null  object
 29  report_status           82442 non-null  object
 30  publication_date        75012 non-null  object
dtypes: float64(5), object(26)
memory usage: 21.7+ MB
None
```

Handling the Four Injury Rows

Taking the four rows and filling the Nans as 0 because therecan only be one type of injury

In [13]:
```
df_clean['total_fatal_injuries'].fillna(0, inplace=True)
df_clean['total_serious_injuries'].fillna(0, inplace=True)
df_clean['total_minor_injuries'].fillna(0, inplace=True)
df_clean['total_uninjured'].fillna(0, inplace=True)
```

```
C:\Anaconda3\envs\learn-env\lib\site-packages\pandas\core\series.py:4517: SettingWithC
opyWarning:
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/use
r_guide/indexing.html#returning-a-view-versus-a-copy
  return super().fillna(
```

Check and make sure the transforemation worked

In [14]:
```
print("Checking the total fatal injuries column")
print(df['Total.Fatal.Injuries'].isna().sum())
print(df_clean['total_fatal_injuries'].isna().sum())

print("Checking the total serious injuries column")
print(df['Total.Serious.Injuries'].isna().sum())
print(df_clean['total_serious_injuries'].isna().sum())

print("Checking the total Minor injuries column")
print(df['Total.Minor.Injuries'].isna().sum())
print(df_clean['total_minor_injuries'].isna().sum())

print("Checking the total Uninjured column")
print(df['Total.Uninjured'].isna().sum())
print(df_clean['total_uninjured'].isna().sum())

df_clean['total_uninjured']
```

```
Checking the total fatal injuries column
11401
0
Checking the total serious injuries column
12510
0
Checking the total Minor injuries column
11933
0
Checking the total Uninjured column
5912
0
```

Out[14]:
```
0        0.0
1        0.0
2        0.0
3        0.0
4        0.0
        ...
88884    0.0
88885    0.0
88886    1.0
88887    0.0
88888    1.0
Name: total_uninjured, Length: 88777, dtype: float64
```

Convert datetime from a string to a datetime object

In [15]:
```
df_clean.event_date = pd.to_datetime(df_clean["event_date"], format="%Y-%m-%d")
```

```
C:\Anaconda3\envs\learn-env\lib\site-packages\pandas\core\generic.py:5168: SettingWith
CopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/use
r_guide/indexing.html#returning-a-view-versus-a-copy
  self[name] = value
```

Check and make sure that it worked

In [16]:
```
df_clean.event_date
```

Out[16]:
```
0       1948-10-24
1       1962-07-19
2       1974-08-30
3       1977-06-19
4       1979-08-02
           ...
88884   2022-12-26
88885   2022-12-26
88886   2022-12-26
88887   2022-12-26
88888   2022-12-29
Name: event_date, Length: 88777, dtype: datetime64[ns]
```

Now the values have been converted to a datetime object.

In [17]:
```
df_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 88777 entries, 0 to 88888
Data columns (total 31 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   event_id                88777 non-null  object
 1   investigation_type      88777 non-null  object
 2   accident_number         88777 non-null  object
 3   event_date              88777 non-null  datetime64[ns]
 4   location                88725 non-null  object
 5   country                 88552 non-null  object
 6   latitude                34347 non-null  object
 7   longitude               34338 non-null  object
 8   airport_code            50213 non-null  object
 9   airport_name            52746 non-null  object
 10  injury_severity         87798 non-null  object
 11  aircraft_damage         85605 non-null  object
 12  aircraft_category       32245 non-null  object
 13  registration_number     87502 non-null  object
 14  make                    88777 non-null  object
 15  model                   88777 non-null  object
```

```
 16   amateur_built           88678 non-null  object
 17   number_of_engines       82754 non-null  float64
 18   engine_type             81771 non-null  object
 19   far_description         31936 non-null  object
 20   schedule                12532 non-null  object
 21   purpose_of_flight       82639 non-null  object
 22   air_carrier             16609 non-null  object
 23   total_fatal_injuries    88777 non-null  float64
 24   total_serious_injuries  88777 non-null  float64
 25   total_minor_injuries    88777 non-null  float64
 26   total_uninjured         88777 non-null  float64
 27   weather_condition       84338 non-null  object
 28   broad_phase_of_flight   61683 non-null  object
 29   report_status           82442 non-null  object
 30   publication_date        75012 non-null  object
dtypes: datetime64[ns](1), float64(5), object(25)
memory usage: 21.7+ MB
```

Cleaning The Engines Column

We are going to filter out all the NaN's from the engines column

In [18]:
```
# Looking at the entries with nulls
df_clean[df_clean['number_of_engines'].isna()]
```

Out[18]:

	event_id	investigation_type	accident_number	event_date	location	country	latitude	longi
4	20041105X01764	Accident	CHI79FA064	1979-08-02	Canton, OH	United States	NaN	
3600	20001214X42040	Accident	LAX83LA093	1983-01-01	ARROYO GRANDE, CA	United States	NaN	
3741	20001214X42020	Accident	LAX83FUG14	1983-01-22	NORTH SHORE, CA	United States	NaN	
3772	20001214X41990	Accident	FTW83FA102	1983-01-29	HEMPSTEAD, TX	United States	NaN	
3870	20001214X42203	Accident	FTW83LA112	1983-02-12	LITTLEFIELD, TX	United States	NaN	
...	...	...	...	...	...	...	...	
88883	20221228106502	Accident	GAA23WA046	2022-12-22	Brasnorte,	Brazil	NaN	
88884	20221227106491	Accident	ERA23LA093	2022-12-26	Annapolis, MD	United States	NaN	
88885	20221227106494	Accident	ERA23LA095	2022-12-26	Hampton, NH	United States	NaN	
88887	20221227106498	Accident	WPR23LA076	2022-12-26	Morgan, UT	United States	NaN	
88888	20221230106513	Accident	ERA23LA097	2022-12-29	Athens, GA	United States	NaN	

6023 rows × 31 columns

In [19]:
```
# Drop the null values in number_of_engines
# This is because we need to know the number of engines for the plane model
df_clean.dropna(subset=['number_of_engines'], inplace=True)

df_clean['number_of_engines'].isna().sum()
```

```
<ipython-input-19-43b379e53ea4>:3: 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/use
r_guide/indexing.html#returning-a-view-versus-a-copy
  df_clean.dropna(subset=['number_of_engines'], inplace=True)
```

Out[19]: 0

Checking out the make and model

See what cleaning needs to be done. Need to change cases

In [20]:
```
df_clean['make'] = df_clean['make'].str.lower()
```

```
<ipython-input-20-2985eca4928d>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/use
r_guide/indexing.html#returning-a-view-versus-a-copy
  df_clean['make'] = df_clean['make'].str.lower()
```

Trying to fix the names of the make column

In [21]:
```
df_clean['make'] = df_clean['make'].str.split().str[0]
```

```
<ipython-input-21-c0ea681c15c6>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/use
r_guide/indexing.html#returning-a-view-versus-a-copy
  df_clean['make'] = df_clean['make'].str.split().str[0]
```

Combine the injuries columns into a total injuries column

In [22]:
```
df_clean['total_injuries'] = df_clean['total_fatal_injuries'] + df_clean['total_seriou
```

```
<ipython-input-22-a5bcd5f2f4b8>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/use
r_guide/indexing.html#returning-a-view-versus-a-copy
  df_clean['total_injuries'] = df_clean['total_fatal_injuries'] + df_clean['total_seri
ous_injuries'] + df_clean['total_minor_injuries']
```

In [23]:
```
df_clean['total_injuries']
```

Out[23]:
```
0        2.0
1        4.0
2        3.0
3        2.0
5        1.0
        ...
88869    0.0
88873    0.0
88876    0.0
88877    1.0
88886    0.0
Name: total_injuries, Length: 82754, dtype: float64
```

Looking at broad phase of flight

There are nulls and unknowns. Change the nulls to unknown.

In [24]:
```
df_clean['broad_phase_of_flight'].isna().sum()
```

Out[24]: 21991

In [25]:
```
df_clean['broad_phase_of_flight'].value_counts()
```

Out[25]:
```
Landing        15117
Takeoff        12334
Cruise         10174
Maneuvering     8043
Approach        6363
Climb           2009
Taxi            1947
Descent         1840
Go-around       1350
Standing         930
Unknown          542
Other            114
Name: broad_phase_of_flight, dtype: int64
```

In [26]:
```
df_clean['broad_phase_of_flight'].str.contains('Unknown').sum()
```

Out[26]:
```
542
```

So there are 21991 nulls, and 542 unknowns.

We are going to change the nulls to unknowns.

In [27]:
```
# This code takes the list of columns and what to fill the null values with
df_clean['broad_phase_of_flight'] = df_clean['broad_phase_of_flight'].fillna('Unknown
```

```
<ipython-input-27-b0f81f0cdaee>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/use
r_guide/indexing.html#returning-a-view-versus-a-copy
  df_clean['broad_phase_of_flight'] = df_clean['broad_phase_of_flight'].fillna('Unknow
n')
```

In [28]:
```
df_clean['broad_phase_of_flight'].value_counts()
```

Out[28]:
```
Unknown        22533
Landing        15117
Takeoff        12334
Cruise         10174
Maneuvering     8043
Approach        6363
Climb           2009
Taxi            1947
Descent         1840
Go-around       1350
Standing         930
Other            114
Name: broad_phase_of_flight, dtype: int64
```

Worked.

Looking into aircraft damage and doing the same thing

In [29]:
```
df_clean['aircraft_damage'].isna().sum()
```

Out[29]:
```
2021
```

In [30]:
```
df_clean['aircraft_damage'].value_counts()
```

Out[30]:
```
Substantial    61465
Destroyed      16972
Minor           2235
Unknown           61
Name: aircraft_damage, dtype: int64
```

There are a total of 61 unknowns and 2021 nulls.

Let's change those nulls to unknowns.

In [31]:
```
df_clean['aircraft_damage'] = df_clean['aircraft_damage'].fillna('Unknown')
```

```
<ipython-input-31-71db3fb88d8c>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/use
r_guide/indexing.html#returning-a-view-versus-a-copy
  df_clean['aircraft_damage'] = df_clean['aircraft_damage'].fillna('Unknown')
```

In [32]:
```
df_clean['aircraft_damage'].isna().sum()
```

Out[32]: 0

Splitting The Data

Now that our data is looking pretty clean, let's split our data into relevant categories in order to solve our main business problem. These categories are airplanes, carriers, and helicopters.

We will make three new dataframes by filtering the relevant data. First, let's start with Airplanes.

Airplanes

In [33]:
```
# What are the possible categories?
df_clean['aircraft_category'].value_counts()
```

```
Out[33]: Airplane              24831
         Helicopter             2907
         Glider                  431
         Gyrocraft               168
         Balloon                 166
         Weight-Shift            154
         Powered Parachute        87
         Ultralight               26
         WSFT                      9
         Blimp                     4
         Unknown                   4
         Powered-Lift              4
         Rocket                    1
         ULTR                      1
         Name: aircraft_category, dtype: int64
```

```
In [34]: df_clean_airplanes = df_clean.loc[df_clean['aircraft_category'] == 'Airplane']
```

```
In [35]: # What are the top 10 makes of airplanes?

         top_10_makes = df_clean_airplanes['make'].value_counts().head(10)
         top_10_makes
```

```
Out[35]: cessna      7948
         piper       4495
         beech       1531
         boeing       679
         mooney       446
         air          410
         grumman      390
         cirrus       356
         bellanca     272
         aeronca      228
         Name: make, dtype: int64
```

We are going to drop

the null and unknown values from engine_type for the dataframe because it is relevant to the safety of airplanes.

Dropping Null Values From engine_type

```
In [36]: df_clean_airplanes.dropna(subset=['engine_type'], inplace=True)
         df_clean_airplanes['engine_type'].isna().sum()

<ipython-input-36-c9c9cb67ffda>:1: 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/use
r_guide/indexing.html#returning-a-view-versus-a-copy
  df_clean_airplanes.dropna(subset=['engine_type'], inplace=True)

Out[36]: 0
```

Dropping unknown values from engine_type

```
In [37]: df_clean_airplanes.drop(df_clean[(df_clean['engine_type'] == 'unknown')].index, inplac
         df_clean_airplanes

C:\Anaconda3\envs\learn-env\lib\site-packages\pandas\core\frame.py:4163: SettingWithCo
pyWarning:
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/use
r_guide/indexing.html#returning-a-view-versus-a-copy
  return super().drop(
```

Out[37]:

	event_id	investigation_type	accident_number	event_date	location	country	latitude
5	20170710X52551	Accident	NYC79AA106	1979-09-17	BOSTON, MA	United States	42.445277
7	20020909X01562	Accident	SEA82DA022	1982-01-01	PULLMAN, WA	United States	NaN
8	20020909X01561	Accident	NYC82DA015	1982-01-01	EAST HANOVER, NJ	United States	NaN
12	20020917X02148	Accident	FTW82FRJ07	1982-01-02	HOMER, LA	United States	NaN
13	20020917X02134	Accident	FTW82FRA14	1982-01-02	HEARNE, TX	United States	NaN
...	...	...	...	...	...	...	...
88639	20221011106092	Accident	CEN23LA008	2022-10-06	Iola, TX	United States	304354N
88647	20221011106098	Accident	ERA23LA014	2022-10-08	Dacula, GA	United States	034055N
88661	20221018106153	Accident	CEN23LA015	2022-10-13	Ardmore, OK	United States	034849N
88735	20221031106231	Accident	CEN23LA023	2022-10-29	Houston, TX	United States	293620N
88767	20221109106272	Accident	CEN23LA033	2022-11-09	Bridgeport, TX	United States	331026N

22700 rows × 32 columns

Filtering down the engines

Engines we don't want: Reciprocating Engines

```
In [38]: df_clean_airplanes['engine_type'].value_counts().index

Out[38]: Index(['Reciprocating', 'Turbo Prop', 'Turbo Fan', 'Turbo Jet', 'Unknown',
                'Geared Turbofan', 'Turbo Shaft', 'Electric', 'UNK'],
               dtype='object')
```

How to handle the Reciprocating engines?

Remove the rows with reciprocating engines.

```
In [39]: # This code works by:
         # Calling the drop() method and passing it the filtered rows we want to drop

         # I want to drop the rows of df_clean_airplanes where the engine_type is Reciprocatin
         df_clean_airplanes.drop( df_clean_airplanes[df_clean_airplanes['engine_type'] == 'Rec

C:\Anaconda3\envs\learn-env\lib\site-packages\pandas\core\frame.py:4163: SettingWithCo
pyWarning:
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/use
r_guide/indexing.html#returning-a-view-versus-a-copy
  return super().drop(
```

```
In [40]: # export .csv file to look at in Tableau

         df_clean_airplanes.to_csv('./Data/clean_airplanes.csv')
```

```
In [41]: ### Exploring airplanes and the differing number of engines for their makes
```

This is a filter for planes with two engines and a new dataframe consiting of data only for planes with 2 engines:

```
In [42]: df_clean_airplanes_w_2_engines = df_clean_airplanes.loc[df_clean_airplanes['number_of_
```

```
In [43]: # df to csv

         df_clean_airplanes_w_2_engines.to_csv('./Data/two_engine_planes.csv')
```

This is a filter for planes with three engines and a new dataframe consiting of data only for planes with 3 engines:

```
In [44]: df_clean_airplanes_w_3_engines = df_clean_airplanes.loc[df_clean_airplanes['number_of_
```

```
In [45]: # df to csv

         df_clean_airplanes_w_3_engines.to_csv('./Data/three_engine_plane.csv')
```

```
In [ ]:
```

```
In [ ]:
```

Investigating the Carriers

```
In [46]: df_clean_airplanes['model'].value_counts()

Out[46]: 208B                  60
         208                   44
         AT-502B               36
         737                   36
         A320                  33
                               ..
         BOEING 777-228 IGW     1
         DHC 6 200              1
         LANCAIR IV P           1
         DHC-2                  1
         CL600-2A12             1
         Name: model, Length: 875, dtype: int64
```

```
In [47]: df_737 = df_clean_airplanes.query("model == '737' | model == '747' | model == '340'| r
         df_737 = df_737.query("make == 'boeing' | make == 'airbus'")

         df_737['model']

Out[47]: 4150     747
         52199    747
         54167    747
         55630    737
         64807    737
         65457    737
         65538    320
         66356    737
         66556    737
         66965    737
         67027    737
         67151    737
         67271    737
         67743    747
         67994    737
         68146    747
         68353    737
         68738    737
         69156    737
         69358    737
         69741    737
         69916    747
         70924    737
         71199    737
         71882    747
         73233    747
         73533    320
         73586    747
         74633    737
         74747    737
         75823    747
         77087    737
         77411    747
         77705    737
         77887    737
         77906    737
         80518    747
         81107    737
         81327    747
         82042    737
         82151    737
         82165    737
         82437    737
         82655    737
         82850    737
         82857    737
         82987    737
         83134    737
         84372    737
         84538    737
         85070    737
         Name: model, dtype: object
```

```
In [48]: df_737['model'].value_counts()

Out[48]: 737    36
         747    13
         320     2
         Name: model, dtype: int64
```

```
In [49]: df_737['make'].value_counts()

Out[49]: boeing    49
         airbus     2
         Name: make, dtype: int64
```

```
In [50]: number_of_planes_737 = df_737['model'].value_counts().values[0]
         number_of_planes_747 = df_737['model'].value_counts().values[1]
```

```
In [51]: df_737_accidents = df_737[df_737['model'] == '737']
         df_747_accidents = df_737[df_737['model'] == '747']

         df_737_substantial = df_737_accidents[df_737_accidents['aircraft_damage'] == 'Substant

         df_747_substantial = df_747_accidents[df_747_accidents['aircraft_damage'] == 'Substant

         df_737_minor = df_737_accidents[df_737_accidents['aircraft_damage'] == 'Minor'].shape

         df_747_minor = df_747_accidents[df_747_accidents['aircraft_damage'] == 'Minor'].shape
```

```
In [52]: # export .csv file to look at further in Tableau
         df_737.to_csv('./Data/carriers.csv')
```

Helicopters

```
In [53]: df_clean_helicopters = df_clean.loc[df_clean['aircraft_category'] == 'Helicopter']
```

```
In [54]: df_clean_helicopters[df_clean_helicopters['engine_type'] == 'Reciprocating'].sort_valu
```

Out[54]:

	event_id	investigation_type	accident_number	event_date	location	country	latitude
88414	20220831105841	Accident	WPR22LA329	2022-08-26	Deadhorse, AK	United States	684126N
88341	20220811105721	Accident	WPR22LA299	2022-08-11	Palmer, AK	United States	612618N
88309	20220808105687	Accident	CEN22LA366	2022-08-05	Maynard, IA	United States	424912N
88281	20220801105634	Accident	CEN22LA350	2022-07-31	Marshall, MO	United States	039633N
88256	20220729105620	Accident	CEN22LA345	2022-07-28	Ithaca, NE	United States	041946N
...	...	...	...	...	...	...	...
151	20020917X02578	Accident	SEA82DA020	1982-01-22	NEAR KINGSTON, ID	United States	NaN
137	20020917X01915	Accident	DEN82DA021	1982-01-20	BONDURANT, WY	United States	NaN
91	20020917X02577	Accident	SEA82DA019	1982-01-15	S.E. OF MALTA, ID	United States	NaN
19	20020917X02339	Accident	MIA82DA028	1982-01-02	MIAMI, FL	United States	NaN
16	20020917X01962	Accident	DEN82DTM08	1982-01-02	MIDWAY, UT	United States	NaN

1375 rows × 32 columns

It looks like reciprocating engines are still being used for helicopters today, so we will not drop those engines for this dataframe.

```
In [55]: # What models have the highest average of uninjured passengers?

         df_clean_helicopters.groupby('model')['total_uninjured'].mean().sort_values(ascending

Out[55]: model
         92              18.0
         S-92A           14.0
         SA-330J         13.0
         214ST           11.0
         412 - EP        11.0
         S 76B           10.0
         AS 350 BA FX1    7.0
         BV234            7.0
         S-76C++          7.0
         EC 130B4         6.0
         Name: total_uninjured, dtype: float64
```

```
In [56]: # turn into .csv file to visualize in Tableau
         df_clean_helicopters.to_csv('./Data/clean_helicopters.csv')
```

Results

Question Statement:

Which aircrafts are the lowest risk?

What are our three concrete business recommendations

After anaylising our data we are able to come to the results that;

Passenger Transportation

- The Boeing 727 model is ideal for passenger transportaion because out of all 2407 plane crashes only 345 of them were a Boeing make
- Out of all 345 Boeing crashes, only 16 of them were with the Boeing 727 model the only Boeing make with three engines.
- Out of all 16 Boeing crashes only 3 had any injuries related none of them being fatal.
- We were able to reach this conclusion by analyzing the data set and filtering it down o specific compenents we were interested in looking at and then comparing the data sets with information like total crashes for makes or total crashes for a specific number of engined plane.

Cargo Transportation

- The Boeing 747 model is ideal for cargo transportation.
- We came to this conclusion by making a ratio of number of accidents to number of planes.
- When comparing the aircraft damage severity between the Boeings the 737 is smaller than the 747.
- The Boeing 747 has 107 ton of capacity.
- The nose of the 747 opens (really cool).
- The 747 has a wide body which means more space.
- The 747 also has 4 engines making it much safer.

Private Enterprises

- The Sikorsky S-92A is ideal for private enterprises
- Since certain makes and models have more crash occurrences than others
- Used the mean to show the ratio of uninjured passengers to the number of accidents.

```
In [ ]:
```