index

June 17, 2024

1 Aviation Accident Database & Synopses, up to 2023

1.1 About Dataset

1.1.1 Content

The NTSB aviation accident database contains information from 1962 and later about civil aviation accidents and selected incidents within the United States, its territories and possessions, and in international waters.

1.1.2 Acknowledgements

Generally, a preliminary report is available online within a few days of an accident. Factual information is added when available, and when the investigation is completed, the preliminary report is replaced with a final description of the accident and its probable cause. Full narrative descriptions may not be available for dates before 1993, cases under revision, or where NTSB did not have primary investigative responsibility.

1.1.3 Inspiration

Hope it will teach us how to improve the quality and safety of traveling by Airplane.

> **Note:** We are using the CRISP DM methodology to help use meet our requirements

1.1.4 Business Problem

Your company is expanding in to new industries to diversify its portfolio. Specifically, they are interested in purchasing and operating airplanes for commercial and private enterprises, but do not know anything about the potential risks of aircraft. You are charged with determining which aircraft are the lowest risk for the company to start this new business endeavor. You must then translate your findings into actionable insights that the head of the new aviation division can use to help decide which aircraft to purchase.

1.1.5 Deliverables

There are three deliverables for this project:

- A **non-technical presentation**
- A **Jupyter Notebook**
- A **GitHub repository**
- An **Interactive Dashboard**

1.2 Understanding the data

1.2.1 Loading the data

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

[ ]:
```
df = pd.read_excel('./data/AviationData.xlsx')
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  \
0               UNK                Cruise  Probable Cause
1               UNK               Unknown  Probable Cause
2               IMC                Cruise  Probable Cause
3               IMC                Cruise  Probable Cause
4               VMC              Approach  Probable Cause
```

```
      Publication.Date
0                  NaN
1           19-09-1996
2           26-02-2007
3  2000-12-09 00:00:00
4           16-04-1980

[5 rows x 31 columns]
```

Note: This dataset is quite large so load just once. Had to load an excel file because csv results to an error

[ ]:
```
df.shape
```

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

[ ]:
```
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]
```

[ ]:
```
df.describe()
```

[ ]:
```
                          Event.Date  Number.of.Engines  Total.Fatal.Injuries  \
count                          88889       82805.000000          77488.000000
mean   1999-09-17 17:13:39.354475904           1.146585              0.647855
min              1948-10-24 00:00:00           0.000000              0.000000
25%              1989-01-15 00:00:00           1.000000              0.000000
50%              1998-07-18 00:00:00           1.000000              0.000000
75%              2009-07-01 00:00:00           1.000000              0.000000
max              2022-12-29 00:00:00           8.000000            349.000000
std                              NaN           0.446510              5.485960

       Total.Serious.Injuries  Total.Minor.Injuries  Total.Uninjured
count            76379.000000          76956.000000     82977.000000
mean                 0.279881              0.357061         5.325440
min                  0.000000              0.000000         0.000000
25%                  0.000000              0.000000         0.000000
50%                  0.000000              0.000000         1.000000
75%                  0.000000              0.000000         2.000000
max                161.000000            380.000000       699.000000
std                  1.544084              2.235625        27.913634
```

Isolate just the **interesting columns** that seem relevant to my research and contribute to business understanding here

[ ]:
```
interesting_columns = [
    'Injury.Severity',
    'Aircraft.damage',
    'Aircraft.Category',
    'Make',
    'Model',
    'Number.of.Engines',
```

```
    'Engine.Type',
    'Weather.Condition',
    'Broad.phase.of.flight',
    'Total.Fatal.Injuries',
    'Total.Serious.Injuries',
    'Total.Minor.Injuries',
    'Total.Uninjured'
]
```

These ***variables*** will help me analyze and compare the safety records, damage extent, and injury outcomes of different aircraft models and manufacturers, ultimately leading to informed recommendations for aircraft purchase decisions.

[ ]:
```
df2 = df[interesting_columns].copy()  # it's crutial you use copy to avoid␣
↪`SettingWithCopyWarning`
df2.head()
```

[ ]:
```
  Injury.Severity Aircraft.damage Aircraft.Category      Make     Model  \
0       Fatal(2)       Destroyed               NaN   Stinson     108-3
1       Fatal(4)       Destroyed               NaN     Piper  PA24-180
2       Fatal(3)       Destroyed               NaN    Cessna      172M
3       Fatal(2)       Destroyed               NaN  Rockwell       112
4       Fatal(1)       Destroyed               NaN    Cessna       501

   Number.of.Engines    Engine.Type Weather.Condition Broad.phase.of.flight  \
0                1.0  Reciprocating               UNK                Cruise
1                1.0  Reciprocating               UNK               Unknown
2                1.0  Reciprocating               IMC                Cruise
3                1.0  Reciprocating               IMC                Cruise
4                NaN            NaN               VMC              Approach

   Total.Fatal.Injuries  Total.Serious.Injuries  Total.Minor.Injuries  \
0                   2.0                     0.0                   0.0
1                   4.0                     0.0                   0.0
2                   3.0                     NaN                   NaN
3                   2.0                     0.0                   0.0
4                   1.0                     2.0                   NaN

   Total.Uninjured
0              0.0
1              0.0
2              NaN
3              0.0
4              0.0
```

[ ]:
```
df2.shape
```

[ ]: (88889, 13)

1.3 Data Cleaning

1.3.1 Handling missing values

[ ]:
```
df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 88889 entries, 0 to 88888
Data columns (total 13 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Injury.Severity         87889 non-null  object
 1   Aircraft.damage         85695 non-null  object
 2   Aircraft.Category       32287 non-null  object
 3   Make                    88826 non-null  object
 4   Model                   88797 non-null  object
 5   Number.of.Engines       82805 non-null  float64
 6   Engine.Type             81793 non-null  object
 7   Weather.Condition       84397 non-null  object
 8   Broad.phase.of.flight   61724 non-null  object
 9   Total.Fatal.Injuries    77488 non-null  float64
 10  Total.Serious.Injuries  76379 non-null  float64
 11  Total.Minor.Injuries    76956 non-null  float64
 12  Total.Uninjured         82977 non-null  float64
dtypes: float64(5), object(8)
memory usage: 8.8+ MB
```

[ ]:
```
missing_values_series = df2.isna().sum()
missing_values_series
```

[ ]:
```
Injury.Severity            1000
Aircraft.damage            3194
Aircraft.Category         56602
Make                         63
Model                        92
Number.of.Engines          6084
Engine.Type                7096
Weather.Condition          4492
Broad.phase.of.flight     27165
Total.Fatal.Injuries      11401
Total.Serious.Injuries    12510
Total.Minor.Injuries      11933
Total.Uninjured            5912
dtype: int64
```

Create a dataframe to show percentage missing values for each column so we can know

how to deal with them

[ ]:
```
missing_values_indexes = list(missing_values_series.index)
missing_values_values = list(missing_values_series.values)
```

[ ]:
```
missing_values_percentage = pd.DataFrame({"indexes": missing_values_indexes,␣
 ↪"values": missing_values_values, "percentage_missing": list(np.round(((np.
 ↪array(missing_values_values) * 100) / len(df2)), 2))})
missing_values_percentage.set_index("indexes", inplace=True)
missing_values_percentage
```

[ ]:

indexes	values	percentage_missing
Injury.Severity	1000	1.12
Aircraft.damage	3194	3.59
Aircraft.Category	56602	63.68
Make	63	0.07
Model	92	0.10
Number.of.Engines	6084	6.84
Engine.Type	7096	7.98
Weather.Condition	4492	5.05
Broad.phase.of.flight	27165	30.56
Total.Fatal.Injuries	11401	12.83
Total.Serious.Injuries	12510	14.07
Total.Minor.Injuries	11933	13.42
Total.Uninjured	5912	6.65

Some columns have increasing alot of missing values and therefore they need to be **dropped** entirely based on their relevance in this research

[ ]:
```
# Imputing missing values
df2['Injury.Severity'].fillna(df2['Injury.Severity'].mode()[0], inplace=True)
df2['Aircraft.damage'].fillna(df2['Aircraft.damage'].mode()[0], inplace=True)
df2['Aircraft.Category'].fillna('Unknown', inplace=True)
df2['Make'].fillna(df2['Make'].mode()[0], inplace=True)
df2['Model'].fillna(df2['Model'].mode()[0], inplace=True)
df2['Number.of.Engines'].fillna(df2['Number.of.Engines'].mode()[0],␣
 ↪inplace=True)
df2['Engine.Type'].fillna(df2['Engine.Type'].mode()[0], inplace=True)
df2['Weather.Condition'].fillna(df2['Weather.Condition'].mode()[0],␣
 ↪inplace=True)
df2['Broad.phase.of.flight'].fillna('Unknown', inplace=True)
df2['Total.Fatal.Injuries'].fillna(0, inplace=True)
df2['Total.Serious.Injuries'].fillna(0, inplace=True)
df2['Total.Minor.Injuries'].fillna(0, inplace=True)
df2['Total.Uninjured'].fillna(df2['Total.Uninjured'].mode()[0], inplace=True)
df2.isna().sum()
```

```
Injury.Severity           0
Aircraft.damage           0
Aircraft.Category         0
Make                      0
Model                     0
Number.of.Engines         0
Engine.Type               0
Weather.Condition         0
Broad.phase.of.flight     0
Total.Fatal.Injuries      0
Total.Serious.Injuries    0
Total.Minor.Injuries      0
Total.Uninjured           0
dtype: int64
```

```
df2['Weather.Condition'].loc[(df2['Weather.Condition'] == 'Unk')] = 'UNK'
df2['Weather.Condition'].unique()
```

```
array(['UNK', 'IMC', 'VMC'], dtype=object)
```

```
df2['Engine.Type'].loc[(df2['Engine.Type'] == 'UNK')] = 'Unknown'
df2['Engine.Type'].unique()
```

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

```
df2['Aircraft.Category'].loc[(df2['Aircraft.Category'] == 'UNK')] = 'Unknown'
df2['Aircraft.Category'].unique()
```

```
array(['Unknown', 'Airplane', 'Helicopter', 'Glider', 'Balloon',
       'Gyrocraft', 'Ultralight', 'Blimp', 'Powered-Lift', 'Weight-Shift',
       'Powered Parachute', 'Rocket', 'WSFT', 'ULTR'], dtype=object)
```

1.3.2 Handle Duplicates

```
# Check for duplicates in the dataframe
duplicates = df2.duplicated()

# Display the number of duplicate rows
num_duplicates = duplicates.sum()
print(f'Number of duplicate rows: {num_duplicates}')

# Display the duplicate rows, if any
if num_duplicates > 0:
    duplicate_rows = df2[duplicates]
    print(duplicate_rows)
```

```
else:
    print('No duplicate rows found.')
```

```
Number of duplicate rows: 27203
      Injury.Severity Aircraft.damage Aircraft.Category  \
59          Non-Fatal     Substantial          Airplane
121         Non-Fatal     Substantial          Airplane
136         Non-Fatal     Substantial          Airplane
175         Non-Fatal     Substantial          Airplane
180         Non-Fatal     Substantial          Airplane
…                 …               …                …
88873       Non-Fatal     Substantial          Airplane
88874       Non-Fatal     Substantial           Unknown
88877           Minor     Substantial          Airplane
88883           Fatal     Substantial           Unknown
88886       Non-Fatal     Substantial          Airplane

                            Make   Model  Number.of.Engines    Engine.Type  \
59                         Cessna     152                1.0  Reciprocating
121                        Cessna     152                1.0  Reciprocating
136                        Cessna     152                1.0  Reciprocating
175                        Cessna     152                1.0  Reciprocating
180                        Cessna     152                1.0  Reciprocating
…                             …     …                  …            …
88873          CIRRUS DESIGN CORP    SR22                1.0  Reciprocating
88874                        BELL  206-L4                1.0  Reciprocating
88877                      CESSNA   R172K                1.0  Reciprocating
88883                  AIR TRACTOR   AT502                1.0  Reciprocating
88886  AMERICAN CHAMPION AIRCRAFT   8GCBC                1.0  Reciprocating

      Weather.Condition Broad.phase.of.flight  Total.Fatal.Injuries  \
59                  VMC               Takeoff                   0.0
121                 VMC                Cruise                   0.0
136                 VMC               Landing                   0.0
175                 VMC               Landing                   0.0
180                 VMC               Landing                   0.0
…                   …                       …                   …
88873               VMC               Unknown                   0.0
88874               VMC               Unknown                   0.0
88877               VMC               Unknown                   0.0
88883               VMC               Unknown                   1.0
88886               VMC               Unknown                   0.0

       Total.Serious.Injuries  Total.Minor.Injuries  Total.Uninjured
59                        0.0                   0.0              1.0
121                       0.0                   0.0              1.0
136                       0.0                   0.0              1.0
```

```
175                     0.0                   0.0              1.0
180                     0.0                   0.0              1.0
…                       …                     …                …
88873                   0.0                   0.0              1.0
88874                   0.0                   0.0              0.0
88877                   1.0                   0.0              0.0
88883                   0.0                   0.0              0.0
88886                   0.0                   0.0              1.0

[27203 rows x 13 columns]
```

[ ]:
```
# Remove duplicates from the dataframe
df2_cleaned = df2.drop_duplicates()

# Verify that duplicates have been removed
print(f'Number of rows after removing duplicates: {df2_cleaned.shape[0]}')
```

```
Number of rows after removing duplicates: 61686
```

[ ]:
```
df2_cleaned.head()
```

[ ]:
```
  Injury.Severity Aircraft.damage Aircraft.Category      Make     Model  \
0        Fatal(2)       Destroyed           Unknown   Stinson     108-3
1        Fatal(4)       Destroyed           Unknown     Piper  PA24-180
2        Fatal(3)       Destroyed           Unknown    Cessna      172M
3        Fatal(2)       Destroyed           Unknown  Rockwell       112
4        Fatal(1)       Destroyed           Unknown    Cessna       501

   Number.of.Engines    Engine.Type Weather.Condition Broad.phase.of.flight  \
0                1.0        Unknown               UNK                Cruise
1                1.0        Unknown               UNK               Unknown
2                1.0  Reciprocating               IMC                Cruise
3                1.0  Reciprocating               IMC                Cruise
4                1.0  Reciprocating               VMC              Approach

   Total.Fatal.Injuries  Total.Serious.Injuries  Total.Minor.Injuries  \
0                   2.0                     0.0                   0.0
1                   4.0                     0.0                   0.0
2                   3.0                     0.0                   0.0
3                   2.0                     0.0                   0.0
4                   1.0                     2.0                   0.0

   Total.Uninjured
0              0.0
1              0.0
2              0.0
3              0.0
```

```
4                0.0
```

1.3.3 Type Conversion

[ ]:
```
df2_cleaned.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 61686 entries, 0 to 88888
Data columns (total 13 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Injury.Severity         61686 non-null  object
 1   Aircraft.damage         61686 non-null  object
 2   Aircraft.Category       61686 non-null  object
 3   Make                    61686 non-null  object
 4   Model                   61686 non-null  object
 5   Number.of.Engines       61686 non-null  float64
 6   Engine.Type             61686 non-null  object
 7   Weather.Condition       61686 non-null  object
 8   Broad.phase.of.flight   61686 non-null  object
 9   Total.Fatal.Injuries    61686 non-null  float64
 10  Total.Serious.Injuries  61686 non-null  float64
 11  Total.Minor.Injuries    61686 non-null  float64
 12  Total.Uninjured         61686 non-null  float64
dtypes: float64(5), object(8)
memory usage: 6.6+ MB
```

[ ]:
```
df2_cleaned['Number.of.Engines'] = df2_cleaned['Number.of.Engines'].map(int)
df2_cleaned.loc[:, ['Total.Fatal.Injuries', 'Total.Serious.Injuries', 'Total.
 ↪Minor.Injuries', 'Total.Uninjured']] = df2_cleaned[['Total.Fatal.Injuries',␣
 ↪'Total.Serious.Injuries', 'Total.Minor.Injuries', 'Total.Uninjured']].
 ↪astype(int)
df2_cleaned.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 61686 entries, 0 to 88888
Data columns (total 13 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Injury.Severity         61686 non-null  object
 1   Aircraft.damage         61686 non-null  object
 2   Aircraft.Category       61686 non-null  object
 3   Make                    61686 non-null  object
 4   Model                   61686 non-null  object
 5   Number.of.Engines       61686 non-null  int64
 6   Engine.Type             61686 non-null  object
 7   Weather.Condition       61686 non-null  object
```

```
 8   Broad.phase.of.flight   61686 non-null  object
 9   Total.Fatal.Injuries    61686 non-null  float64
 10  Total.Serious.Injuries  61686 non-null  float64
 11  Total.Minor.Injuries    61686 non-null  float64
 12  Total.Uninjured         61686 non-null  float64
dtypes: float64(4), int64(1), object(8)
memory usage: 6.6+ MB
```

```
/tmp/ipykernel_75697/177581856.py: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/user_guide/indexing.html#returning-a-view-versus-a-copy
  df2_cleaned['Number.of.Engines'] = df2_cleaned['Number.of.Engines'].map(int)
```

1.3.4 Export Data

[ ]:
```
df2_cleaned.to_csv('./data/AviationDataCleaned.csv', index=False)
```

1.4 Data Analysis

[ ]:
```
# Descriptive statistics for numerical variables
df2_cleaned[['Total.Fatal.Injuries', 'Total.Serious.Injuries', 'Total.Minor.
↪Injuries', 'Total.Uninjured']].describe()
```

[ ]:

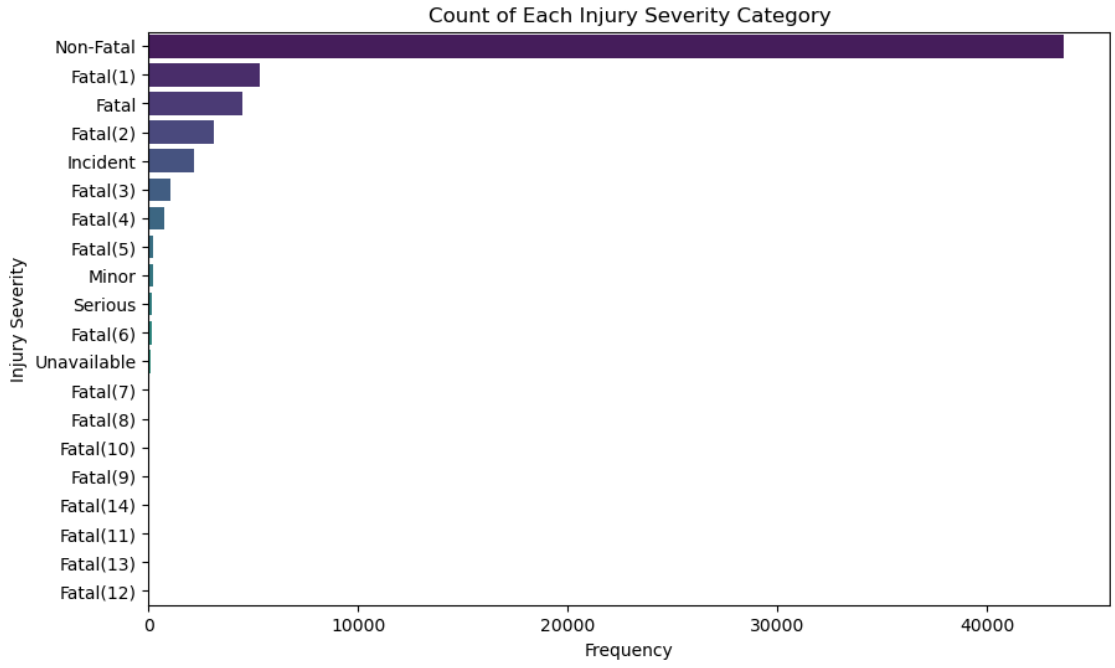
	Total.Fatal.Injuries	Total.Serious.Injuries	Total.Minor.Injuries	Total.Uninjured
count	61686.000000	61686.000000	61686.000000	61686.000000
mean	0.748079	0.316701	0.397951	6.574231
std	6.129929	1.704701	2.482730	32.178140
min	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.000000	0.000000	0.000000
50%	0.000000	0.000000	0.000000	1.000000
75%	0.000000	0.000000	0.000000	2.000000
max	349.000000	161.000000	380.000000	699.000000

1.4.1 Visualizations

Bar chart for Injury Severity

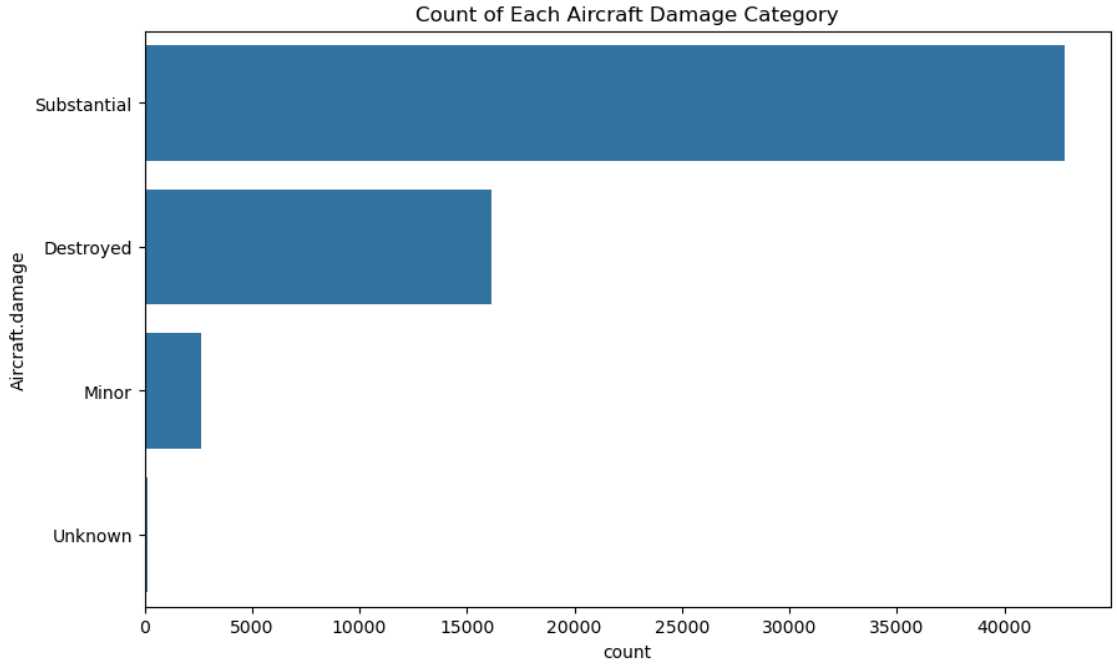
[ ]:
```
# Counting frequencies of each severity category
severity_counts = df2_cleaned['Injury.Severity'].value_counts()[:20]

# Filtering out categories with zero counts
severity_counts = severity_counts[severity_counts > 1]

# Bar chart for Injury Severity
plt.figure(figsize=(10, 6))
ax = sns.barplot(y=severity_counts.index, x=severity_counts.values,␣
 ↪palette='viridis')
ax.set_title('Count of Each Injury Severity Category')
ax.set_xlabel('Frequency')
ax.set_ylabel('Injury Severity')
plt.show()
```

```
/tmp/ipykernel_75697/1056378067.py:9: FutureWarning:

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

  ax = sns.barplot(y=severity_counts.index, x=severity_counts.values,
palette='viridis')
```



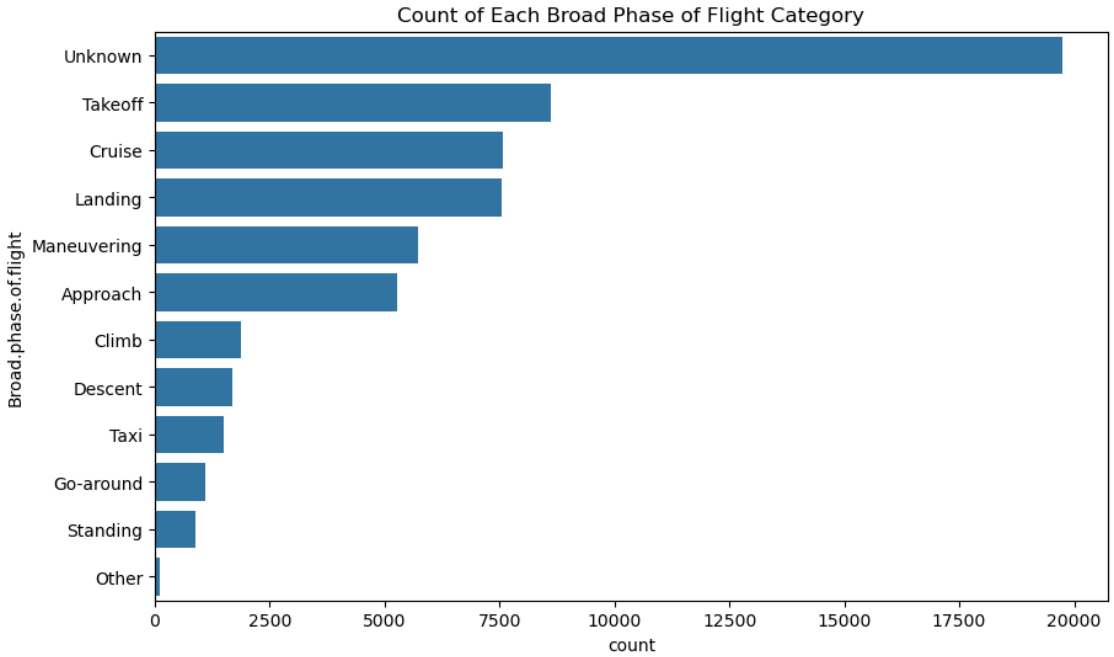
Bar chart for Aircraft Damage

```
[ ]: # Bar chart for Aircraft Damage
plt.figure(figsize=(10, 6))
sns.countplot(y='Aircraft.damage', data=df2_cleaned,␣
 ↪order=df2_cleaned['Aircraft.damage'].value_counts().index)
plt.title('Count of Each Aircraft Damage Category')
plt.show()
```



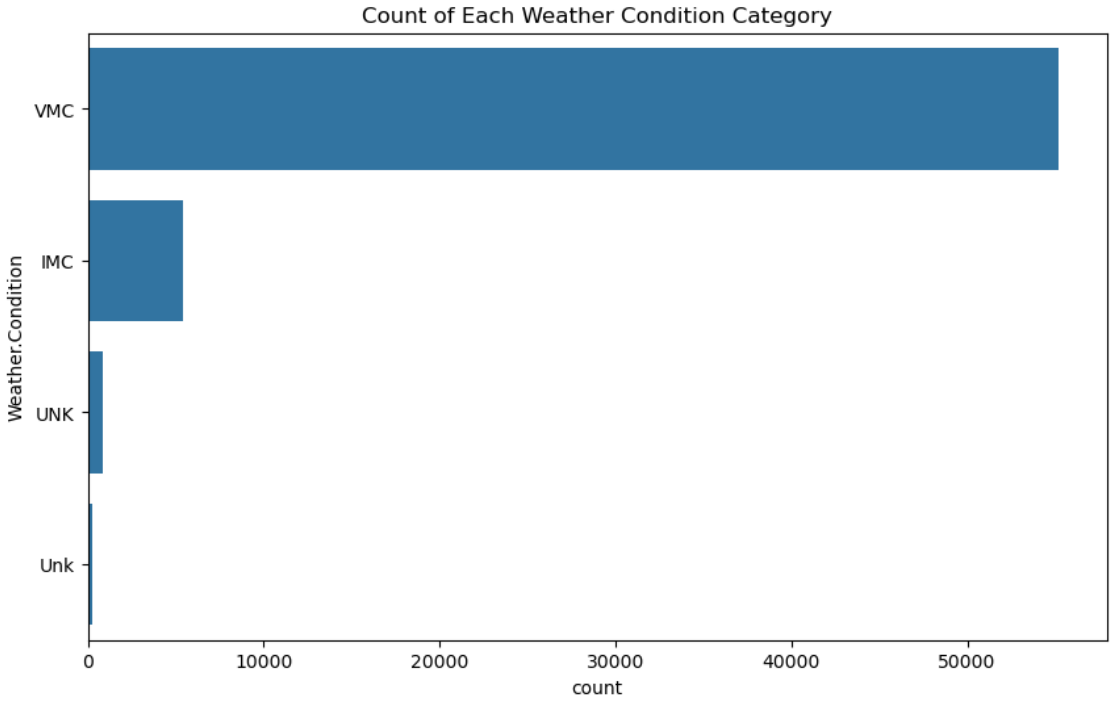
Bar chart for Broad Phase of Flight

```
[ ]: # Bar chart for Broad Phase of Flight
plt.figure(figsize=(10, 6))
sns.countplot(y='Broad.phase.of.flight', data=df2_cleaned,␣
 ↪order=df2_cleaned['Broad.phase.of.flight'].value_counts().index)
plt.title('Count of Each Broad Phase of Flight Category')
plt.show()
```



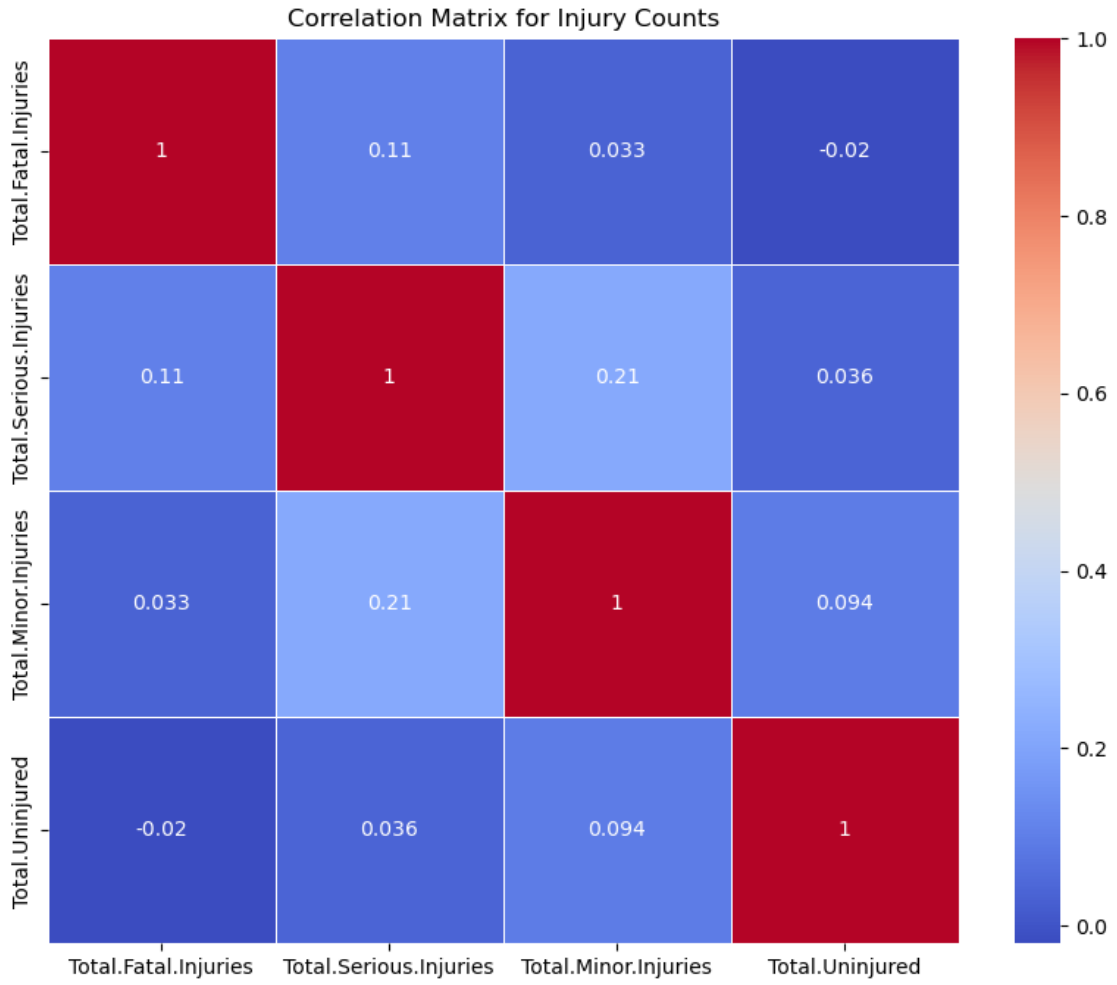
Bar chart for Weather Condition

[ ]:
```
# Bar chart for Weather Condition
plt.figure(figsize=(10, 6))
sns.countplot(y='Weather.Condition', data=df2_cleaned,␣
  ↪order=df2_cleaned['Weather.Condition'].value_counts().index)
plt.title('Count of Each Weather Condition Category')
plt.show()
```



Correlation heatmap for numerical variables

[ ]:
```
# Correlation heatmap for numerical variables
plt.figure(figsize=(10, 8))
correlation_matrix = df2_cleaned[['Total.Fatal.Injuries', 'Total.Serious.
  ↪Injuries', 'Total.Minor.Injuries', 'Total.Uninjured']].corr()
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', linewidths=0.5)
plt.title('Correlation Matrix for Injury Counts')
plt.show()
```



1.5 Interpretation and Recommendations

Invest in Aircraft with Proven Safety Records

Recommendation: Prioritize the acquisition of aircraft models and makes that have consistently demonstrated lower injury severity rates.

Rationale: By focusing on aircraft with lower fatality and injury rates, we can mitigate potential risks and enhance passenger safety.

Implement Robust Safety Protocols for Adverse Weather Conditions

Recommendation: Develop comprehensive operational guidelines and safety measures for flights operating under Instrument Meteorological Conditions (IMC).

Rationale: Our analysis indicates that weather conditions significantly impact the severity of accidents. By focusing on IMC conditions, we can reduce the likelihood of severe accidents.

Opt for Aircraft with Reliable Engine Types

Recommendation: Choose aircraft models that use Reciprocating engine types that have lower associated risks, as identified in our analysis.

Rationale: Different engine types exhibit varying safety records, with some types showing a higher propensity for severe accidents.