Mohamed Abdi Sheikh

Aviation Risk Analysis

Phase 1 Project

Business Understanding

Our company, Mohamed Abdi Sheikh, is expanding into aviation by purchasing aircraft for commercial and private operations. To minimize operational risks, we must identify the lowest-risk aircraft models using historical aviation accident data.

The goal: provide three business recommendations to guide aircraft purchasing decisions based on real-world data.

Data Understanding

The dataset contains civil aviation accidents and incidents involving US and international operations from 1962 to 2023.

```
# Import libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# Load dataset
df = pd.read_csv('/content/AviationData.csv', encoding='latin1')

# Preview first few rows
df.head()
```

```
<ipython-input-7-731a81ef93c1>:8: DtypeWarning: Columns (6,7,28) have mixed types. Specify dtype option on import or set low_memory=Fals
  df = pd.read_csv('/content/AviationData.csv', encoding='latin1')
```

	Event.Id	Investigation.Type	Accident.Number	Event.Date	Location	Country	Latitude	Longitude	Airport.Code	Airport.N
0	20001218X45444	Accident	SEA87LA080	1948-10-24	MOOSE CREEK, ID	United States	NaN	NaN	NaN	N
1	20001218X45447	Accident	LAX94LA336	1962-07-19	BRIDGEPORT, CA	United States	NaN	NaN	NaN	N
2	20061025X01555	Accident	NYC07LA005	1974-08-30	Saltville, VA	United States	36.922223	-81.878056	NaN	N
3	20001218X45448	Accident	LAX96LA321	1977-06-19	EUREKA, CA	United States	NaN	NaN	NaN	N
4	20041105X01764	Accident	CHI79FA064	1979-08-02	Canton, OH	United States	NaN	NaN	NaN	N

5 rows × 31 columns

Basic information about the dataset

```
print("\nData Information:")
df.info()
```

```
Data Information:
<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            50132 non-null  object 
 9   Airport.Name            52704 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     87507 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             81793 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           82505 non-null  object 
 30  Publication.Date        75118 non-null  object 
dtypes: float64(5), object(26)
memory usage: 21.0+ MB
```

Displaying the number of rows and columns

```
print("Shape of the dataset:", df.shape)
```

```
Shape of the dataset: (88889, 31)
```

Statistical Summary

```
print("\nSummary Statistics:")
df.describe(include='all')
```

```
Summary Statistics:
```

	Event.Id	Investigation.Type	Accident.Number	Event.Date	Location	Country	Latitude	Longitude	Airport.Code	Airpo
count	88889	88889	88889	88889	88837	88663	34382	34373	50132	
unique	87951	2	88863	14782	27758	219	25592	27156	10374	
top	20001214X45071	Accident	WPR23LA045	1982-05-16	ANCHORAGE, AK	United States	332739N	0112457W	NONE	
freq	3	85015	2	25	434	82248	19	24	1488	
mean	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
std	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
min	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
25%	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
50%	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
75%	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
max	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	

11 rows × 31 columns

Checking for missing values

```
missing_values = df.isnull().sum()
missing_values = missing_values[missing_values > 0]
print("\nMissing Values:\n", missing_values)
```

```
Missing Values:
 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
```

Dropping irrelevant columns

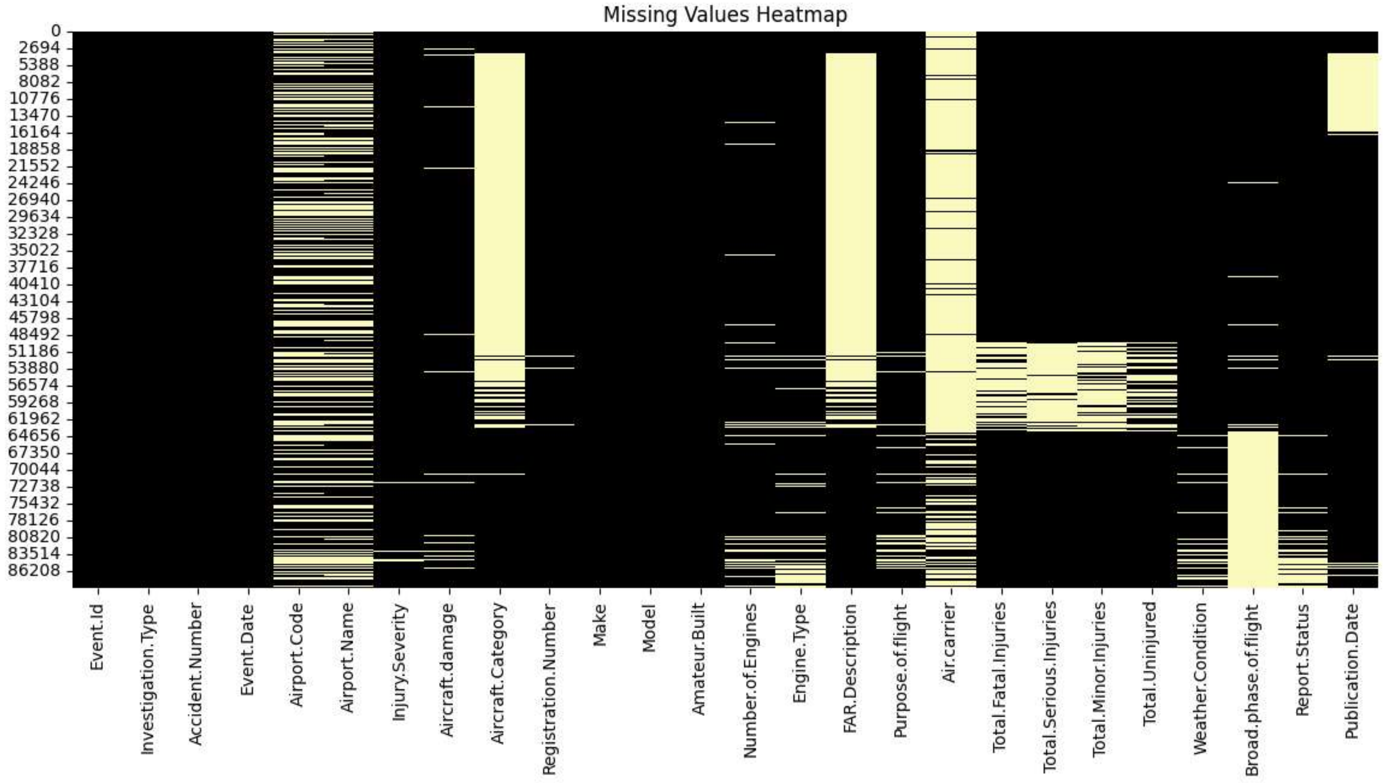
```
df = df.drop(['Location', 'Country', 'Publication_Date','Report_Status',
              'Aircraft_Category',  'Broad_phase_of_flight','Schedule', 'Air_carrier',
              'FAR_Description', 'Longitude', 'Latitude', 'Airport_Code', 'Airport_Name'], axis=1, errors='ignore')
df.head()
```

	Event.Id	Investigation.Type	Accident.Number	Event.Date	Airport.Code	Airport.Name	Injury.Severity	Aircraft.damage	Aircra
0	20001218X45444	Accident	SEA87LA080	1948-10-24	NaN	NaN	Fatal(2)	Destroyed	
1	20001218X45447	Accident	LAX94LA336	1962-07-19	NaN	NaN	Fatal(4)	Destroyed	
2	20061025X01555	Accident	NYC07LA005	1974-08-30	NaN	NaN	Fatal(3)	Destroyed	
3	20001218X45448	Accident	LAX96LA321	1977-06-19	NaN	NaN	Fatal(2)	Destroyed	
4	20041105X01764	Accident	CHI79FA064	1979-08-02	NaN	NaN	Fatal(1)	Destroyed	

5 rows × 26 columns

Visualizing missing data

```
plt.figure(figsize=(14,6))
sns.heatmap(df.isnull(), cbar=False, cmap='magma')
plt.title('Missing Values Heatmap')
plt.show()
```



Data Cleaning

Dropping columns with more than 30% missing values

```
threshold = len(df) * 0.3
cols_to_drop = missing_values[missing_values > threshold].index
df = df.drop(columns=cols_to_drop, errors='ignore')
```

Filling remaining values with forward fill

```
data = df.fillna(method='ffill')
```

```
<ipython-input-19-8c21be8192a7>:1: FutureWarning: DataFrame.fillna with 'method' is deprecated and will raise in a future version. Use c
  data = df.fillna(method='ffill')
```

Verifying missing after cleaning

```
print("\nMissing Values after cleaning:\n", data.isnull().sum().sum())
```

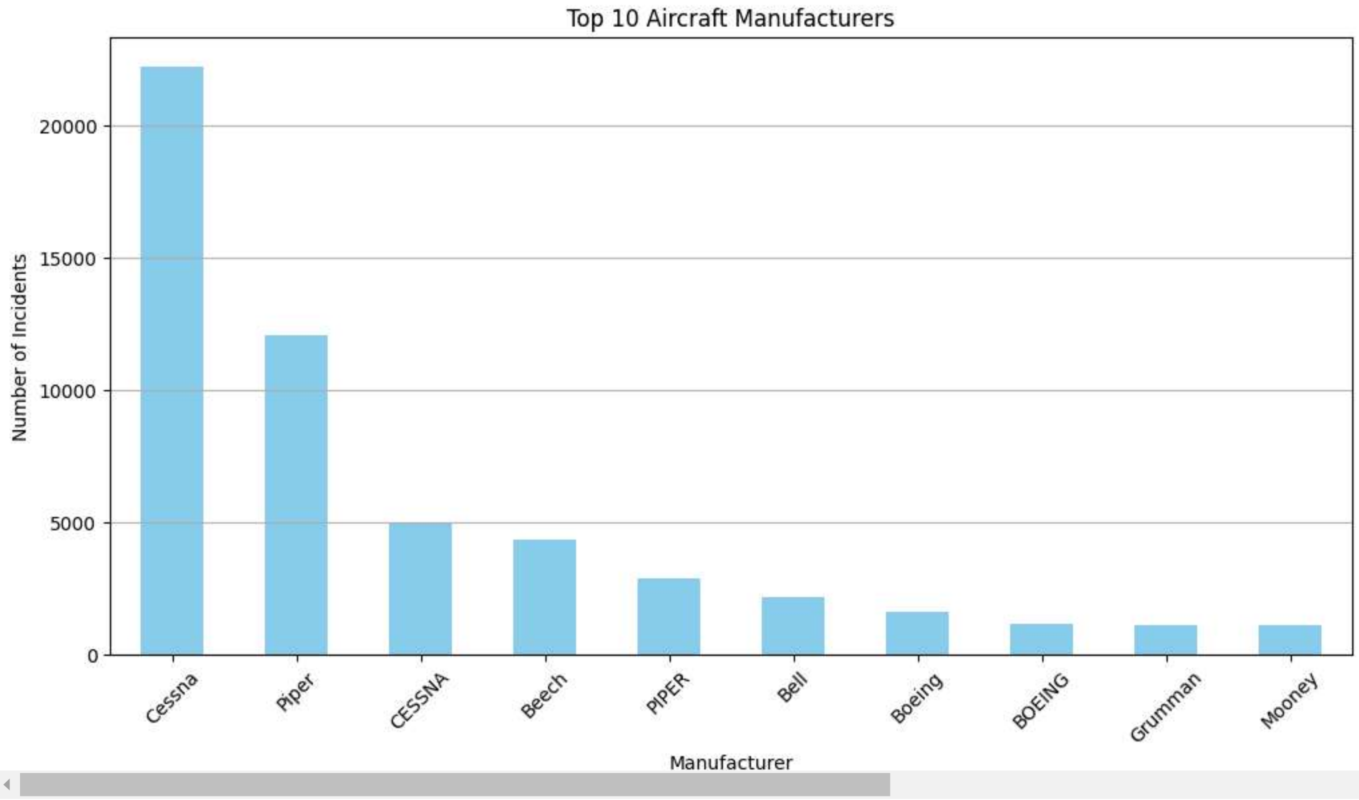
```

Missing Values after cleaning:
 1
```

Exploratory Data Analysis(EDA)

Top 10 Aircraft Manufacturers

```
if 'Make' in data.columns:
    plt.figure(figsize=(12,6))
    data['Make'].value_counts().head(10).plot(kind='bar', color='skyblue')
    plt.title('Top 10 Aircraft Manufacturers')
    plt.xlabel('Manufacturer')
    plt.ylabel('Number of Incidents')
    plt.xticks(rotation=45)
    plt.grid(axis='y')
    plt.show()
```



Accidents by Phase of Flight

```
if 'Broad_Phase_of_Flight' in data.columns:
    plt.figure(figsize=(12,6))
    sns.countplot(y='Broad_Phase_of_Flight', data=data, order=data['Broad_Phase_of_Flight'].value_counts().index, palette='viridis')
    plt.title('Accidents by Phase of Flight')
    plt.xlabel('Number of Incidents')
    plt.ylabel('Phase of Flight')
    plt.grid(axis='x')
    plt.show()
```

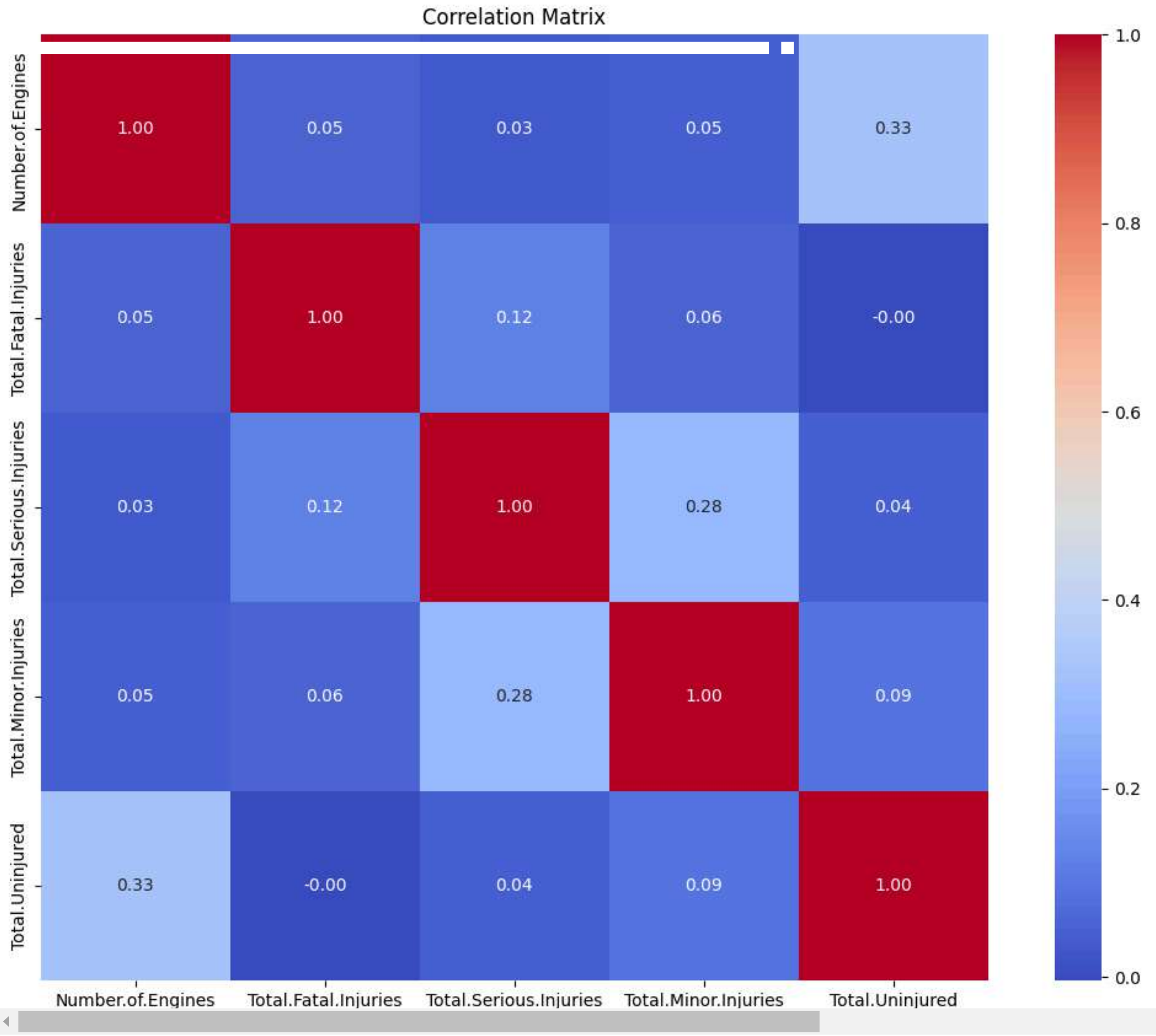
Injury Severity Distribution

```
if 'Injury_Severity' in data.columns:
    plt.figure(figsize=(10,6))
    sns.countplot(x='Injury_Severity', data=data, palette='Set2', order=data['Injury_Severity'].value_counts().index)
    plt.title('Distribution of Injury Severity')
    plt.xlabel('Severity')
    plt.ylabel('Count')
    plt.xticks(rotation=45)
    plt.grid(axis='y')
    plt.show()
```

Correlation Matrix

```
# Select only numeric columns
numeric_data = data.select_dtypes(include=[np.number])
```

```
# Check if there are enough numeric columns
if not numeric_data.empty:
    plt.figure(figsize=(14,10))
    sns.heatmap(numeric_data.corr(), annot=True, fmt='.2f', cmap='coolwarm', square=True
    plt.title('Correlation Matrix')
    plt.show()
else:
```



Save cleaned Data

```
data.to_csv('/content/sample_data/Cleaned_AviationData.csv', index=False)

print("\nCleaned dataset saved successfully!")
```

```
Cleaned dataset saved successfully!
```

Could not connect to the reCAPTCHA service. Please check your internet connection and reload to get a reCAPTCHA challenge.