Business Diversification

Phase 1 project, November 2024

DSF-PT09 CLASS

Overview

The company in concern, intends to diversify its business portfolio to purchasing and operating airplanes for commercial and private enterprises.

Due to the high capital investment required this project intends to determine the lowest risk aircraft, the company can purchase to start this new business endeavor.

The project, intends to translate the findings into actionable insights that the head of the new aviation division can use to help decide which aircraft to purchase.

Business Undestanding

Objectives

1. Develop an accident risk frequency for the different aircraft makes and models from historical accident records
2. Develop an acquisition strategy based on potential risk portfolio for different make of aircraft
3. Evaluate and determine lowest risk aircraft for purchase for private and commercial enterprises

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

```
#importing the dataset
df=pd.read_csv('/content/AviationData.csv', encoding='latin-1', low_memory=False )
```

```
df.head()
```

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

5 rows × 31 columns

```
df.tail()
```

	Event.Id	Investigation.Type	Accident.Number	Event.Date	Location	Country	Latitude	Longitude	Airport.Code	A
71718	20120522X53206	Accident	ERA12CA351	2012-05-05	Umatilla, FL	United States	285527N	0081397W	X23	
71719	20120506X80918	Accident	ERA12CA321	2012-05-06	Eagleville, TN	United States	354140N	0086370W	50M	
71720	20120507X35707	Accident	CEN12CA279	2012-05-06	Hallsville, MO	United States	039118N	0922746W	NaN	
71721	20120507X51002	Accident	ERA12CA322	2012-05-06	Newport, VT	United States	445259N	0721328W	EFK	N
71722	20120517X05337	Accident	WPR12CA215	2012-05-06	Brigham City, UT	United States	413315N	0112344W	BMC	

5 rows × 31 columns

Data Understanding

1. Extract the few first and last rows in the dataset to View content
2. View the % of missing values per column and datatypes in the dataset
3. Do a statistical analysis for the numerical columns
4. View the size of the dataset
5. Visualize raw data

```
df.shape
```

```
(71723, 31)
```

```
df.columns
```

```
Index(['Event.Id', 'Investigation.Type', 'Accident.Number', 'Event.Date',
       'Location', 'Country', 'Latitude', 'Longitude', 'Airport.Code',
       'Airport.Name', 'Injury.Severity', 'Aircraft.damage',
       'Aircraft.Category', 'Registration.Number', 'Make', 'Model',
       'Amateur.Built', 'Number.of.Engines', 'Engine.Type', 'FAR.Description',
       'Schedule', 'Purpose.of.flight', 'Air.carrier', 'Total.Fatal.Injuries',
       'Total.Serious.Injuries', 'Total.Minor.Injuries', 'Total.Uninjured',
       'Weather.Condition', 'Broad.phase.of.flight', 'Report.Status',
       'Publication.Date'],
      dtype='object')
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 71723 entries, 0 to 71722
Data columns (total 31 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Event.Id                71723 non-null  object
 1   Investigation.Type      71723 non-null  object
 2   Accident.Number         71723 non-null  object
 3   Event.Date              71723 non-null  object
 4   Location                71671 non-null  object
 5   Country                 71497 non-null  object
 6   Latitude                19263 non-null  object
 7   Longitude               19254 non-null  object
 8   Airport.Code            39910 non-null  object
 9   Airport.Name            42530 non-null  object
```

```
 10  Injury.Severity         71557 non-null  object 
 11  Aircraft.damage         69729 non-null  object 
 12  Aircraft.Category       15416 non-null  object 
 13  Registration.Number     70349 non-null  object 
 14  Make                    71669 non-null  object 
 15  Model                   71645 non-null  object 
 16  Amateur.Built           71620 non-null  object 
 17  Number.of.Engines       68730 non-null  float64
 18  Engine.Type             69421 non-null  object 
 19  FAR.Description         15559 non-null  object 
 20  Schedule                10905 non-null  object 
 21  Purpose.of.flight       68997 non-null  object 
 22  Air.carrier             8501 non-null   object 
 23  Total.Fatal.Injuries    60321 non-null  float64
 24  Total.Serious.Injuries  59212 non-null  float64
 25  Total.Minor.Injuries    59789 non-null  float64
 26  Total.Uninjured         65810 non-null  float64
 27  Weather.Condition       70562 non-null  object 
 28  Broad.phase.of.flight   61724 non-null  object 
 29  Report.Status           70951 non-null  object 
 30  Publication.Date        58964 non-null  object 
dtypes: float64(5), object(26)
memory usage: 17.0+ MB
```

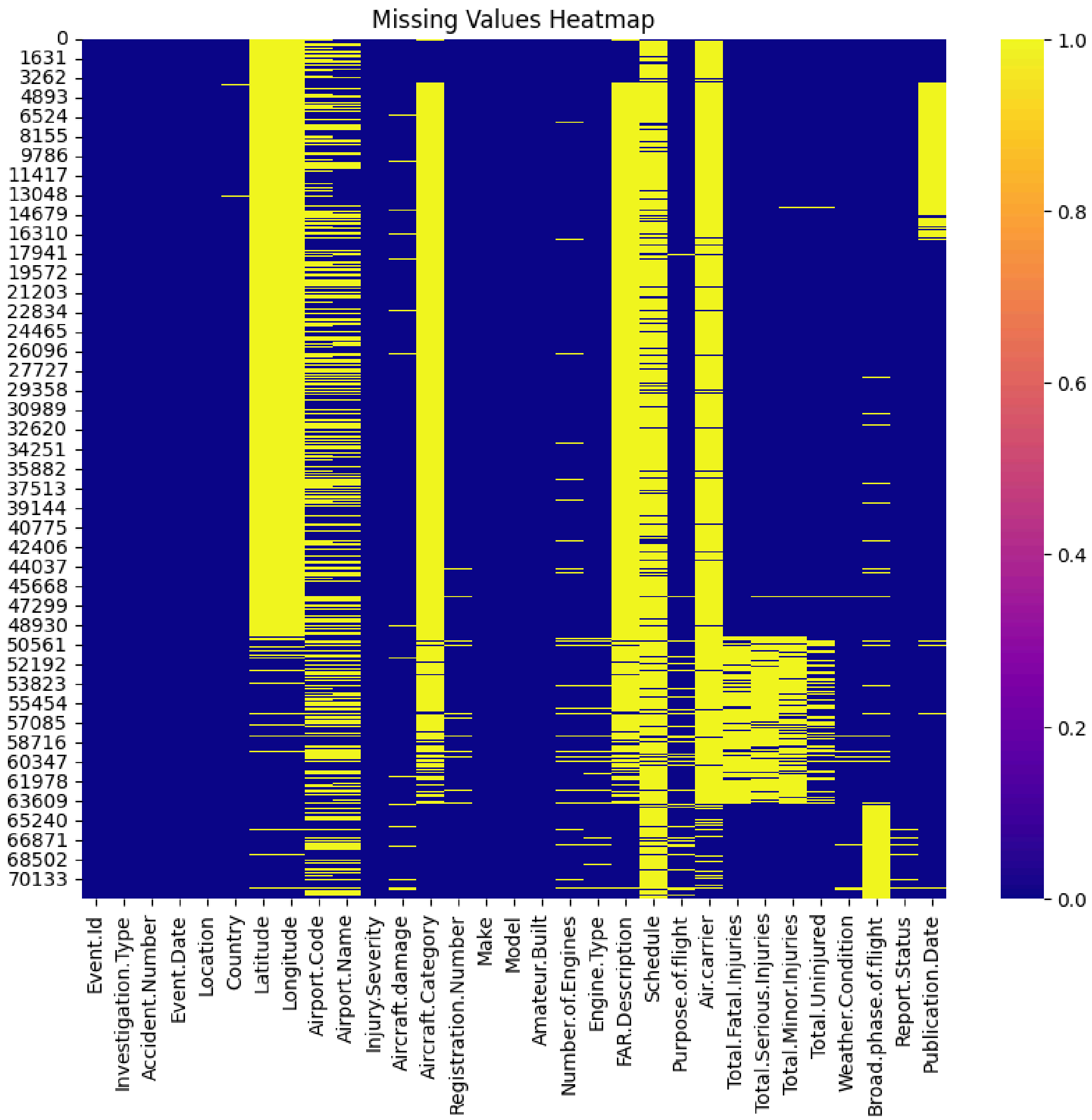
```
df.describe()
```

	Number.of.Engines	Total.Fatal.Injuries	Total.Serious.Injuries	Total.Minor.Injuries	Total.Uninjured
count	68730.000000	60321.000000	59212.000000	59789.000000	65810.000000
mean	1.150560	0.673795	0.276143	0.404188	5.191992
std	0.454254	5.594691	1.352885	2.504597	27.485822
min	0.000000	0.000000	0.000000	0.000000	0.000000
25%	1.000000	0.000000	0.000000	0.000000	0.000000
50%	1.000000	0.000000	0.000000	0.000000	1.000000
75%	1.000000	0.000000	0.000000	0.000000	2.000000
max	4.000000	349.000000	106.000000	380.000000	699.000000

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

	0
Event.Id	0
Investigation.Type	0
Accident.Number	0
Event.Date	0
Location	52
Country	226
Latitude	52460
Longitude	52469
Airport.Code	31813
Airport.Name	29193
Injury.Severity	166
Aircraft.damage	1994
Aircraft.Category	56307
Registration.Number	1374
Make	54
Model	78
Amateur.Built	103
Number.of.Engines	2993
Engine.Type	2302
FAR.Description	56164
Schedule	60818
Purpose.of.flight	2726
Air.carrier	63222
Total.Fatal.Injuries	11402
Total.Serious.Injuries	12511
Total.Minor.Injuries	11934
Total.Uninjured	5913
Weather.Condition	1161
Broad.phase.of.flight	9999
Report.Status	772
Publication.Date	12759

```
# Visualize the missing values
plt.figure(figsize=(10, 8))
sns.heatmap(df.isnull(), cbar=True, cmap="plasma")
plt.title("Missing Values Heatmap")
plt.show()
```



Data Preparation

The data cleaning process will be as follows;

1. Drop columns with more than 70% missing values
2. Drop of all columns that are not of immediate concern to the objective of low risk airplanes. For example

```
*   Investigation.Type
*   Registration.Number
*   Publication.Date
*   Airport.Code
*   Airport.Name
```

3. Substitute the object type column with mode and float64 columns with mean
4. Remove fuzzy duplicates and aliases in Make, Model and Weather conditions column
5. Define the target market by country with the greatest % of available data
6. Import changes to CSV for onward processing in Tableau and visualization
7. As per the business problem a risk matrix best answers the hypothesis. In this case therefore;

- Develop risk metrics
- Assign severity as per the string values provided in the dataset
- Aggregate the risk metrics
- Assign risk scores
- Develop a risk matrix

```
# Drop columns with more than 70% of missing data
df = df.dropna(axis=1, thresh=0.7 * df.shape[0])
print(df.columns)
```

```
Index(['Event.Id', 'Investigation.Type', 'Accident.Number', 'Event.Date',
       'Location', 'Country', 'Injury.Severity', 'Aircraft.damage',
       'Registration.Number', 'Make', 'Model', 'Amateur.Built',
       'Number.of.Engines', 'Engine.Type', 'Purpose.of.flight',
       'Total.Fatal.Injuries', 'Total.Serious.Injuries',
       'Total.Minor.Injuries', 'Total.Uninjured', 'Weather.Condition',
       'Broad.phase.of.flight', 'Report.Status', 'Publication.Date'],
      dtype='object')
```

```
# Drop more columns
df = df.drop(columns=['Investigation.Type', 'Publication.Date', 'Registration.Number'])
```

```
# View new dataset
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 71723 entries, 0 to 71722
Data columns (total 20 columns):
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   Event.Id                71723 non-null  object 
 1   Accident.Number         71723 non-null  object 
 2   Event.Date              71723 non-null  object 
 3   Location                71671 non-null  object 
 4   Country                 71497 non-null  object 
 5   Injury.Severity         71557 non-null  object 
 6   Aircraft.damage         69729 non-null  object 
 7   Make                    71669 non-null  object 
 8   Model                   71645 non-null  object 
 9   Amateur.Built           71620 non-null  object 
 10  Number.of.Engines       68730 non-null  float64
 11  Engine.Type             69421 non-null  object 
 12  Purpose.of.flight       68997 non-null  object 
 13  Total.Fatal.Injuries    60321 non-null  float64
 14  Total.Serious.Injuries  59212 non-null  float64
 15  Total.Minor.Injuries    59789 non-null  float64
 16  Total.Uninjured         65810 non-null  float64
 17  Weather.Condition       70562 non-null  object 
 18  Broad.phase.of.flight   61724 non-null  object 
 19  Report.Status           70951 non-null  object 
dtypes: float64(5), object(15)
memory usage: 10.9+ MB
```

```
#Dropping data before Year 1982. Too few details available in prior years
df= df[df['Event.Date'] >= '1982-01-01']
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 71716 entries, 7 to 71722
Data columns (total 20 columns):
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   Event.Id                71716 non-null  object 
 1   Accident.Number         71716 non-null  object 
 2   Event.Date              71716 non-null  object 
 3   Location                71664 non-null  object 
 4   Country                 71490 non-null  object 
 5   Injury.Severity         71550 non-null  object 
 6   Aircraft.damage         69722 non-null  object 
 7   Make                    71662 non-null  object 
 8   Model                   71638 non-null  object 
 9   Amateur.Built           71613 non-null  object 
 10  Number.of.Engines       68724 non-null  float64
```

```
 11  Engine.Type             69415 non-null  object
 12  Purpose.of.flight       68991 non-null  object
 13  Total.Fatal.Injuries    60315 non-null  float64
 14  Total.Serious.Injuries  59207 non-null  float64
 15  Total.Minor.Injuries    59784 non-null  float64
 16  Total.Uninjured         65804 non-null  float64
 17  Weather.Condition       70555 non-null  object
 18  Broad.phase.of.flight   61717 non-null  object
 19  Report.Status           70944 non-null  object
dtypes: float64(5), object(15)
memory usage: 11.5+ MB
```

```
# fill the missing values in columns with object data type with mode
for column in df.select_dtypes(include='object'):
    df[column].fillna(df[column].mode()[0], inplace=True)
```

```
<ipython-input-18-2d2f287ce685>:3: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chaine
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are set

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = d


  df[column].fillna(df[column].mode()[0], inplace=True)
```

```
# fill float64 data type columns with mean as integer if respective columns
for column in df.select_dtypes(include='float64'):
    df[column].fillna(int(df[column].mean()), inplace=True)
```

```
<ipython-input-19-eb1defcb7aad>:3: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chaine
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are set

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = d


  df[column].fillna(int(df[column].mean()), inplace=True)
```

```
# Relevant data in Purpose.of.flight
df['Purpose.of.flight'].value_counts()
# relevant rows = Personal, Business and Executive/Corporate
df = df[df['Purpose.of.flight'].isin(['Personal', 'Business', 'Executive/Corporate'])]
print(df['Purpose.of.flight'].value_counts())
```

```
Purpose.of.flight
Personal    43114
Business     3618
Name: count, dtype: int64
```

```
# view new data set
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 46732 entries, 7 to 71722
Data columns (total 20 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Event.Id                46732 non-null  object
 1   Accident.Number         46732 non-null  object
 2   Event.Date              46732 non-null  object
 3   Location                46732 non-null  object
 4   Country                 46732 non-null  object
 5   Injury.Severity         46732 non-null  object
 6   Aircraft.damage         46732 non-null  object
 7   Make                    46732 non-null  object
 8   Model                   46732 non-null  object
 9   Amateur.Built           46732 non-null  object
 10  Number.of.Engines       46732 non-null  float64
 11  Engine.Type             46732 non-null  object
 12  Purpose.of.flight       46732 non-null  object
 13  Total.Fatal.Injuries    46732 non-null  float64
 14  Total.Serious.Injuries  46732 non-null  float64
 15  Total.Minor.Injuries    46732 non-null  float64
 16  Total.Uninjured         46732 non-null  float64
 17  Weather.Condition       46732 non-null  object
 18  Broad.phase.of.flight   46732 non-null  object
```

```
 19  Report.Status           46732 non-null  object
dtypes: float64(5), object(15)
memory usage: 7.5+ MB
```

```
# Check duplicates
df.duplicated().sum()
# View duplicated data
df[df.duplicated()]
# Remove the duplicated data
df = df.drop_duplicates()
```

```
# all strings to upper casing in dataset
df = df.apply(lambda col: col.str.upper() if col.dtype == 'object' else col)
```

```
# For pilot purposes by start up, the target market USA is chosen because of the available data count
df['Country'].value_counts()
print(df['Country'].value_counts())
```

```
Country
UNITED STATES     44800
BAHAMAS             124
CANADA              122
MEXICO              117
UNITED KINGDOM       90
                  ...
KAZAKHSTAN            1
MAURITIUS             1
ALGERIA               1
CAMEROON              1
UN                    1
Name: count, Length: 166, dtype: int64
```

```
# Extract the US dataset
df = df[df['Country'].str.strip().eq('UNITED STATES')]
```

```
# export to CSV the cleaned dataset for onward processing in tableau
df.to_csv('cleaned_aviation_data.csv', index=False)

# Assigning the cleaned dataset correctly
df_clean = df.copy()
```

```
#  A risk Matrix best answers the business problem in question
# Group the make and Model columns and specifiy  the funtion to apply to each risk column and regularize new column

risk_metrics = df.groupby(['Make', 'Model']).agg(
    Total_Incidents=('Aircraft.damage', 'count'),
    Average_Fatal_Injuries=('Total.Fatal.Injuries', 'mean'),
    Average_Serious_Injuries=('Total.Serious.Injuries', 'mean'),
    Average_Minor_Injuries=('Total.Minor.Injuries', 'mean'),
    Damage_Frequency=('Aircraft.damage', lambda x: x.value_counts().to_dict())
).reset_index()
```

```
# Quantify the severity of damage. create new column to store frequencies
df['Severe.Damage.Frequency'] = df['Aircraft.damage'].apply(lambda x: x.count('Destroyed') + x.count('Substantial') if isinstance(x, str) else 0)
```

```
# calculate the severity of score frequency for each row as string and Create new 'Severe.Damage.Frequency' to store the damage scores as int
df['Severe_Damage_Frequency'] = df['Aircraft.damage'].str.count('Destroyed') + df['Aircraft.damage'].str.count('Substantial')
```

```
 # Aggregate identified risk metrics by Make and Model
 # This provides incident counts, injury trends and damage severity
 # Allows therefore risk analysis and comparisons

aggregated_metrics = df.groupby(['Make', 'Model']).agg(
    Total_Incidents=('Event.Id', 'count'),
    Average_Fatal_Injuries=('Total.Fatal.Injuries', 'mean'),
    Average_Serious_Injuries=('Total.Serious.Injuries', 'mean'),
    Average_Minor_Injuries=('Total.Minor.Injuries', 'mean'),
    # Changed from 'Severe.Damage.Frequency' to 'Severe_Damage_Frequency'
    Severe_Damage_Frequency=('Severe_Damage_Frequency', 'sum')
).reset_index()
```

```
# Assign weight to each risk factor
weights = {
    'Total_Incidents': 0.4,
    'Average_Fatal_Injuries': 0.3,
    'Average_Serious_Injuries': 0.2,
    'Severe_Damage_Frequency': 0.1
}

# Calculate overall risk score for each aircraft based on weighted score
aggregated_metrics['Risk_Score'] = (
    weights['Total_Incidents'] * aggregated_metrics['Total_Incidents'] +
    weights['Average_Fatal_Injuries'] * aggregated_metrics['Average_Fatal_Injuries'].fillna(0) +
    weights['Average_Serious_Injuries'] * aggregated_metrics['Average_Serious_Injuries'].fillna(0) +
    weights['Severe_Damage_Frequency'] * aggregated_metrics['Severe_Damage_Frequency']
)

# Sort by Risk Score in ascending order
low_risk_aircraft = aggregated_metrics.sort_values(by='Risk_Score', ascending=True)

df_new=low_risk_aircraft
```

```
df.to_csv('aggregated_metrics.csv', index=False)
df_metrics=aggregated_metrics
```

```
df_new.head()
```

	Make	Model	Total_Incidents	Average_Fatal_Injuries	Average_Serious_Injuries	Average_Minor_Injuries	Risk
5336	HOMER DAVIS	RV4	1	0.0	0.0	0.0	
5756	KAUFFMAN	BEDE IV	1	0.0	0.0	0.0	
5754	KASHPUREFF	NIEUPORT II	1	0.0	0.0	0.0	
5751	KARMY	ROTORWAY EXEC	1	0.0	0.0	1.0	
5750	KARL & DOT, INC.	COMP AIR 7SL	1	0.0	0.0	0.0	

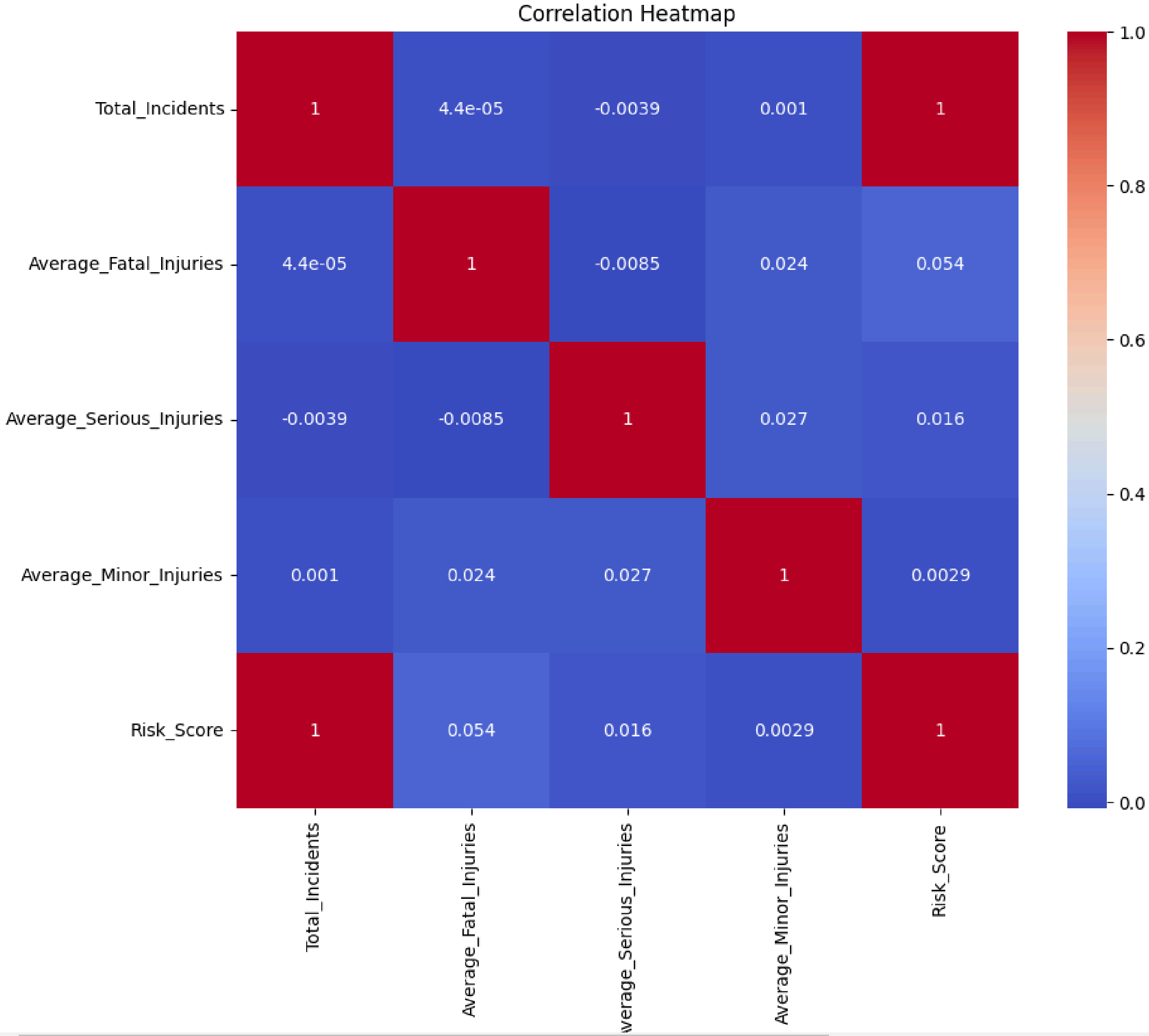
Next steps: Generate code with df_new | View recommended plots | New interactive sheet

Data Visualization

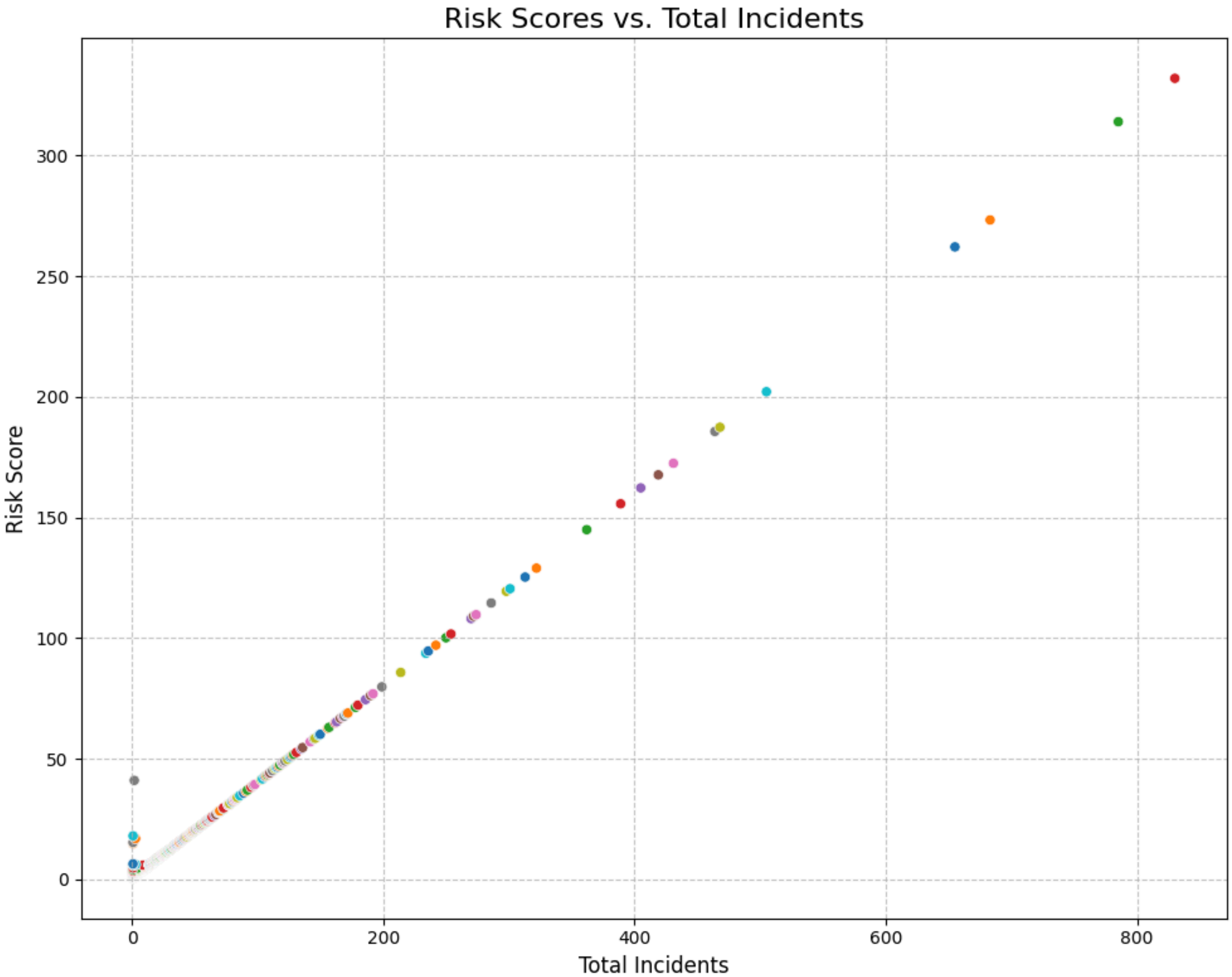
```
#Drop severe damage frequency from aggregated_metrics
df_new.drop(columns=['Severe_Damage_Frequency'], inplace=True)
```

```
# Calculate the correlation matrix based on the aggregated metrics without 'Severe.Damage.Frequency'
correlation_matrix = df_new.select_dtypes(include=['float64', 'int64']).corr()
```

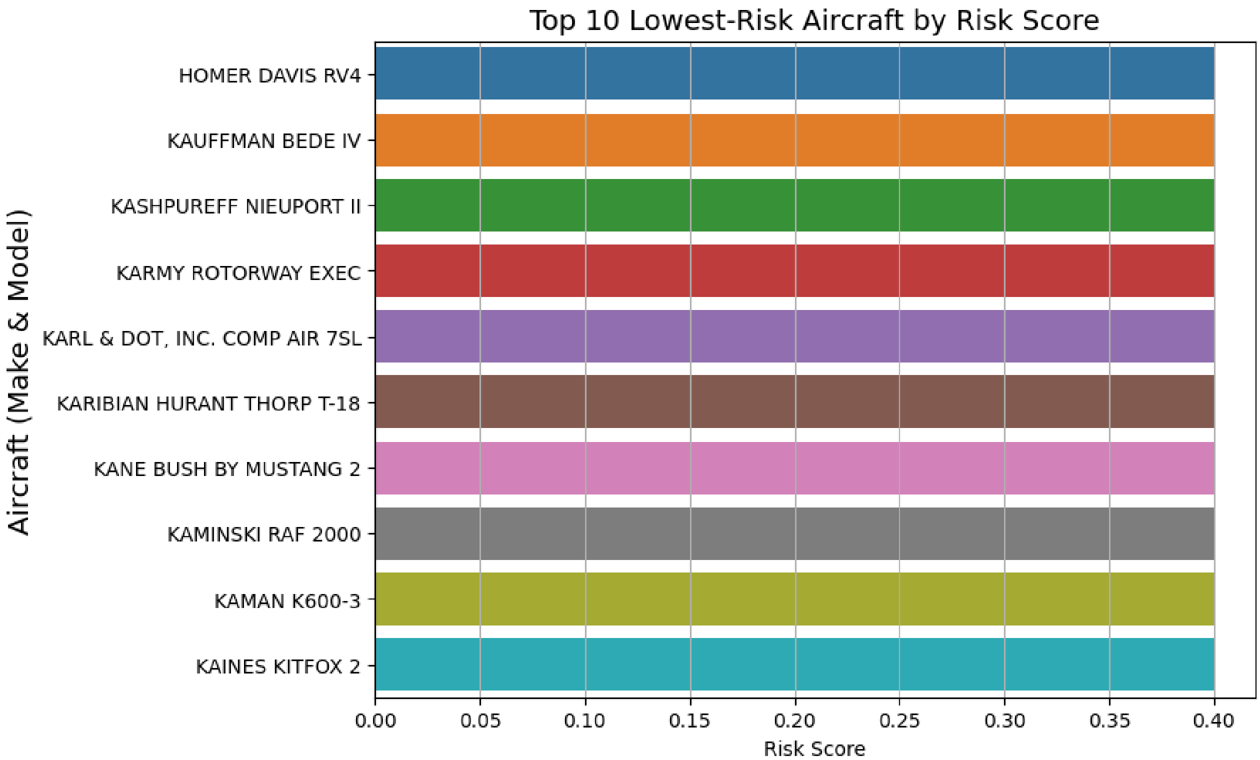
```
# visualization of a heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(df_new.select_dtypes(include=['float64', 'int64']).corr(), annot=True, cmap='coolwarm')
plt.title('Correlation Heatmap')
plt.show()
```



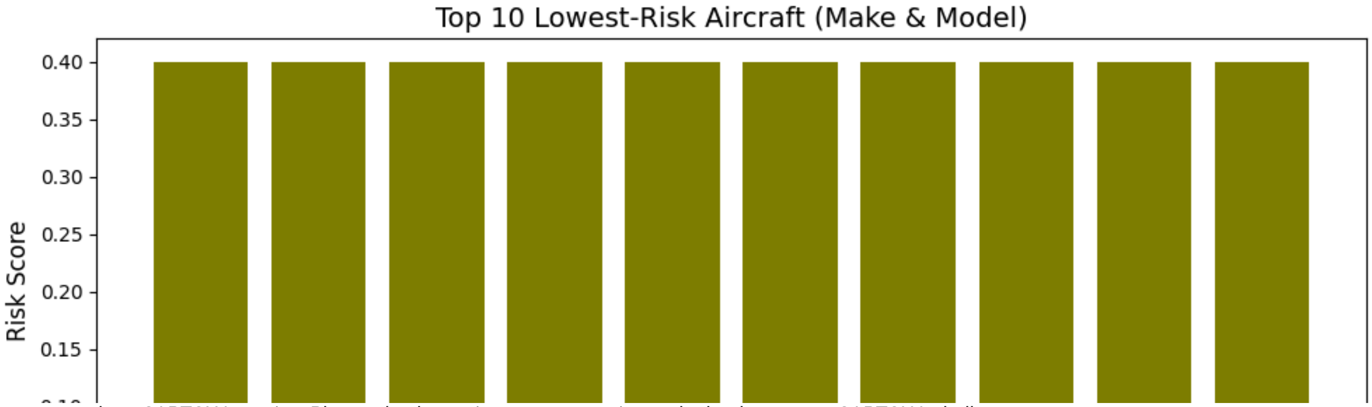

```
#
low_risk_aircraft['Make_Model'] = low_risk_aircraft['Make'] + " " + low_risk_aircraft['Model']

# Create a figure with specific size
plt.figure(figsize=(10, 8))

# Create the scatter plot using seaborn
sns.scatterplot(
    data=low_risk_aircraft,
    x='Total_Incidents',
    y='Risk_Score',
    hue='Make_Model',
    palette='tab10',
    legend=False
)

# Title and labels with customized font sizes
plt.title('Risk Scores vs. Total Incidents', fontsize=16)
plt.xlabel('Total Incidents', fontsize=12)
plt.ylabel('Risk Score', fontsize=12)

# Add gridlines for better readability
plt.grid(linestyle='--', alpha=0.7)

# Ensure everything fits within the plot
plt.tight_layout()

# Show the plot
plt.show()
```




```
# combine the make and aircraft model data in to one column
low_risk_aircraft['Make_Model'] = low_risk_aircraft['Make'] + " " + low_risk_aircraft['Model']

# View the top 10 lowest risk aircraft
top_low_risk = low_risk_aircraft.head(10)

plt.figure(figsize=(8, 6))
sns.barplot(
    data=top_low_risk,
    x='Risk_Score',
    y='Make_Model',
    hue='Make_Model',
    palette='tab10',
    order=top_low_risk.sort_values('Risk_Score')['Make_Model']
)
plt.title('Top 10 Lowest-Risk Aircraft by Risk Score', fontsize=14)
plt.xlabel('Risk Score', fontsize=10)
plt.ylabel('Aircraft (Make & Model)', fontsize=14)
plt.grid(axis='x', linestyle='-')
plt.show()
```




```
# Sort aircraft by Risk Score in ascending order
lowest_risk_aircraft = aggregated_metrics[['Make', 'Model', 'Risk_Score']].sort_values(by='Risk_Score', ascending=True)

# Select top 5 lowest-risk models
top_low_risk_aircraft = lowest_risk_aircraft.head(10)

# Plot the Risk Score for the top 5 lowest-risk aircraft
plt.figure(figsize=(10, 6))
plt.bar(
    top_low_risk_aircraft['Make'] + " " + top_low_risk_aircraft['Model'],
    top_low_risk_aircraft['Risk_Score'],
    color='olive'
)
plt.xlabel('Aircraft (Make and Model)', fontsize=12)
plt.ylabel('Risk Score', fontsize=12)
plt.title('Top 10 Lowest-Risk Aircraft (Make & Model)', fontsize=14)
plt.xticks(rotation=45, ha='right')
plt.tight_layout()

# Show plot
plt.show()

# Display acquisition strategy
print("Acquisition Strategy:")
for index, row in top_low_risk_aircraft.iterrows():
    print(f"Make: {row['Make']}, Model: {row['Model']}, Risk Score: {row['Risk_Score']:.2f}")
```



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