Final Project Submission

Please fill out:

- Student name: David Munyiri
- Student pace: part time
- Scheduled project review date/time: 27/07/2025 23:59:59
- Instructor name: Fidelis Wanalwenge
- Blog post URL:

==

Aviation Safety Risk Analysis Report

==

Introduction

This notebook analyzes aviation accident data to provide recommendations for selecting the safest aircraft models for business, commercial, or personal purposes.

Key objectives:

- Clean and prepare the data
- Compute safety risk metrics (Fatality, Severe Injury, Damage Severity)
- Calculate a weighted Risk Score
- Identify aircraft models with best safety records
- Provide data exports for Tableau visualization #

  ## ------------------------------

Data Exploration

In [1]:

```
#Load the data into a pandas Dataframe
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

Aviation_df = pd.read_csv("data/Aviation_Data.csv")
```

```
C:\Users\david.munyiri\AppData\Local\anaconda3\envs\learn-env\lib\si
te-packages\IPython\core\interactiveshell.py:3145: DtypeWarning: Col
umns (6,7,28) have mixed types.Specify dtype option on import or set
low_memory=False.
  has_raised = await self.run_ast_nodes(code_ast.body, cell_name,
```

In [2]:

```
#Check the size of the Aviation raw data
Aviation_df.shape
```

Out[2]:

```
(90348, 31)
```

In [3]:

```
#View the all the columns of the raw data
Aviation_df.columns
```

Out[3]:

```
Index(['Event.Id', 'Investigation.Type', 'Accident.Number', 'Event.D
ate',
       'Location', 'Country', 'Latitude', 'Longitude', 'Airport.Cod
e',
       'Airport.Name', 'Injury.Severity', 'Aircraft.damage',
       'Aircraft.Category', 'Registration.Number', 'Make', 'Model',
       'Amateur.Built', 'Number.of.Engines', 'Engine.Type', 'FAR.Des
cription',
       'Schedule', 'Purpose.of.flight', 'Air.carrier', 'Total.Fatal.
Injuries',
       'Total.Serious.Injuries', 'Total.Minor.Injuries', 'Total.Unin
jured',
       'Weather.Condition', 'Broad.phase.of.flight', 'Report.Statu
s',
       'Publication.Date'],
      dtype='object')
```

In [4]:

```python
#Get information on the data types and content in different columns
Aviation_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 90348 entries, 0 to 90347
Data columns (total 31 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Event.Id                88889 non-null  object
 1   Investigation.Type      90348 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
```

In [5]:

```python
#View a snapshot of the raw data
Aviation_df.head()
```

Out[5]:

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

5 rows × 31 columns

In [6]:
```
#View statistics of columns of interest
Aviation_df[['Make', 'Model','Aircraft.Category', 'Engine.Type', 'Inj
```

Out[6]:

	Make	Model	Aircraft.Category	Engine.Type	Injury.Severity	Aircraft.damage
count	88826	88797	32287	81812	87889	8569[illegible]
unique	8237	12318	15	13	109	[illegible]
top	Cessna	152	Airplane	Reciprocating	Non-Fatal	Substantia
freq	22227	2367	27617	69530	67357	6414[illegible]

Data Cleaning

Based on a quick exploration, the dataset appears to contain records of accidents and incidents involving various aircraft types, with **airplanes** being the most frequent category.

The focus of our analysis will be on accident records and remove rows missing:

- Make, Model, Aircraft Category
- Injury counts (fatal, serious, minor, uninjured)

which are critical to our eventual recommendation. This cleaning process ensures that the dataset remains relevant, consistent, and ready for further analysis.

In [7]:

```
# Filter only 'Accident' type investigations
accidents_df = Aviation_df[Aviation_df['Investigation.Type'] == 'Acci

# Standardize Make and Model columns before grouping
accidents_df['Make'] = accidents_df['Make'].str.lower().str.strip()
accidents_df['Model'] = accidents_df['Model'].str.lower().str.strip()

# Rebuild combined make_model field
accidents_df['make_model'] = accidents_df['Make'] + ' ' + accidents_d

# Define critical columns to keep
critical_columns = [
    'Make', 'Model', 'Aircraft.Category',
    'Total.Fatal.Injuries', 'Total.Serious.Injuries',
    'Total.Minor.Injuries', 'Total.Uninjured'
]

# Drop rows with missing critical values
accidents_df.dropna(subset=critical_columns, inplace=True)

# Fill in missing aircraft damage field
accidents_df['Aircraft.damage'] = accidents_df['Aircraft.damage'].fil

# Convert injuries to numeric
injury_cols = ['Total.Fatal.Injuries', 'Total.Serious.Injuries', 'Tot
for col in injury_cols:
    accidents_df[col] = pd.to_numeric(accidents_df[col], errors='coer
```

Aggregate Accident Statistics by Aircraft Make and Model

In [8]:
```
#Define the columns that the data will be grouped by
grouped_df = accidents_df.groupby(['make_model'])

#Total risk factor counts
model_summary_df = grouped_df.agg(
    total_accidents=('Model', 'count'),
    total_fatalities=('Total.Fatal.Injuries', 'sum'),
    total_serious=('Total.Serious.Injuries', 'sum'),
    total_minor=('Total.Minor.Injuries', 'sum'),
    total_uninjured=('Total.Uninjured', 'sum'),
    total_destroyed=('Aircraft.damage', lambda x: (x == 'Destroyed').s
).reset_index()

model_summary_df['make_model'] = model_summary_df['make_model'].str.lo

# Total people onboard
model_summary_df['total_people'] = (
    model_summary_df['total_fatalities'] +
    model_summary_df['total_serious'] +
    model_summary_df['total_minor'] +
    model_summary_df['total_uninjured']
)

# Filter for valid data
model_summary_df = model_summary_df[
    (model_summary_df['total_people'] > 0) &
    (model_summary_df['total_accidents'] >= 10)
]

# Add a combined Make_Model label for easier charting
# model_summary_df['make_model'] = model_summary_df['Make'] + ' ' + mo
```

In [9]:
```
# Check for missing values in critical columns
print(model_summary_df[['total_fatalities', 'total_serious', 'total_mi

# Look at models with very few accidents or zero values in critical co
print(model_summary_df[model_summary_df['total_accidents'] < 10])
```

```
total_fatalities    0
total_serious       0
total_minor         0
total_destroyed     0
total_accidents     0
dtype: int64
Empty DataFrame
Columns: [make_model, total_accidents, total_fatalities, total_serio
us, total_minor, total_uninjured, total_destroyed, total_people]
Index: []
```

Compute Risk Indexes

Based on the available dataset, we derive indexes that help us estimate and assign a safety evaluation of each aircraft model

- **Fatality Index** = Fatalities / Total People Onboard
- **Injury Index** = (All Injuries) / Total People
- **Damage Severity Index** = Weighted damage / Total Accidents

In [10]:

```
# Define fatality index
model_summary_df['fatality_index'] = model_summary_df['total_fataliti

#Define injury index
model_summary_df['injury_index'] = (
    model_summary_df['total_serious'] + model_summary_df['total_minor
) / model_summary_df['total_people']

#Define damage severity index
model_summary_df['damage_severity_index'] = model_summary_df['total_d
print(model_summary_df.columns)
```

```
Index(['make_model', 'total_accidents', 'total_fatalities', 'total_s
erious',
       'total_minor', 'total_uninjured', 'total_destroyed', 'total_p
eople',
       'fatality_index', 'injury_index', 'damage_severity_index'],
      dtype='object')
```

Calculate Weighted Risk Score

Define weights for each index — update these anytime to change importance or client priority/preference

- **Fatality Index** = 0.5
- **Injury Index** = 0.2
- **Damage Severity Index** = 0.3

In [11]:
```
# Define damage weights
WEIGHTS = {
    'fatality_index': 0.5,
    'damage_severity_index': 0.3,
    'injury_index': 0.2
}
# Compute Risk score using weighted fatality, damage_severity and Inju
model_summary_df['risk_score'] = (
    model_summary_df['fatality_index'] * WEIGHTS['fatality_index'] +
    model_summary_df['damage_severity_index'] * WEIGHTS['damage_sever
    model_summary_df['injury_index'] * WEIGHTS['injury_index']
)
model_summary_df.tail()
```

Out[11]:

	make_model	total_accidents	total_fatalities	total_serious	total_minor	total_uni
7488	vans rv4	15	9.0	2.0	5.0	
7489	vans rv6	14	6.0	6.0	8.0	
7491	vans rv7	11	4.0	6.0	1.0	
7495	vans rv8	14	5.0	1.0	2.0	
7750	yakovlev yak 52	11	10.0	3.0	2.0	

In [12]:
```
model_summary_df.shape
```

Out[12]: (431, 12)

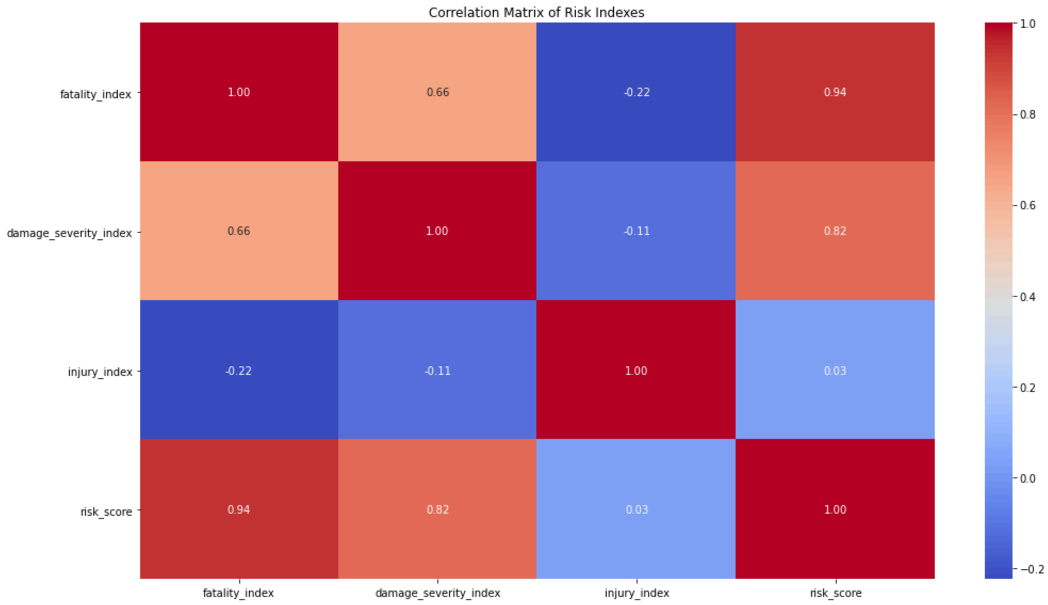
In [13]:
```
model_summary_df_cleaned = model_summary_df.dropna(subset=['risk_score
model_summary_df_cleaned.shape
```

Out[13]: (431, 12)

Visualize Risk Index Distributions

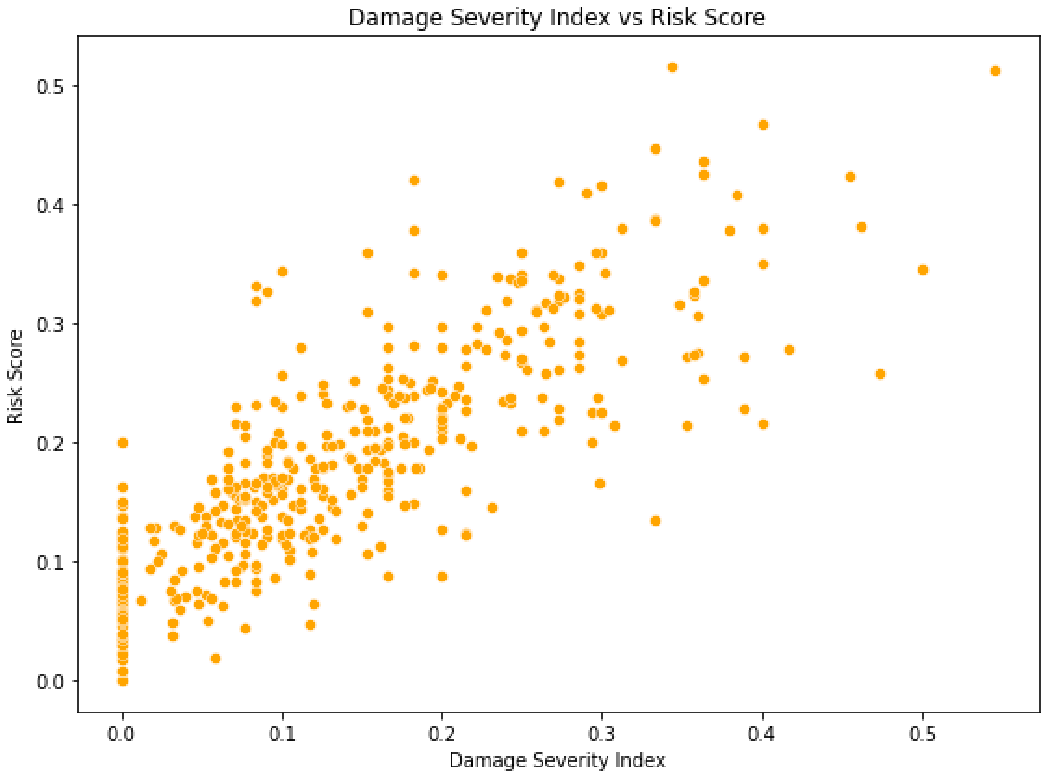
In [14]:

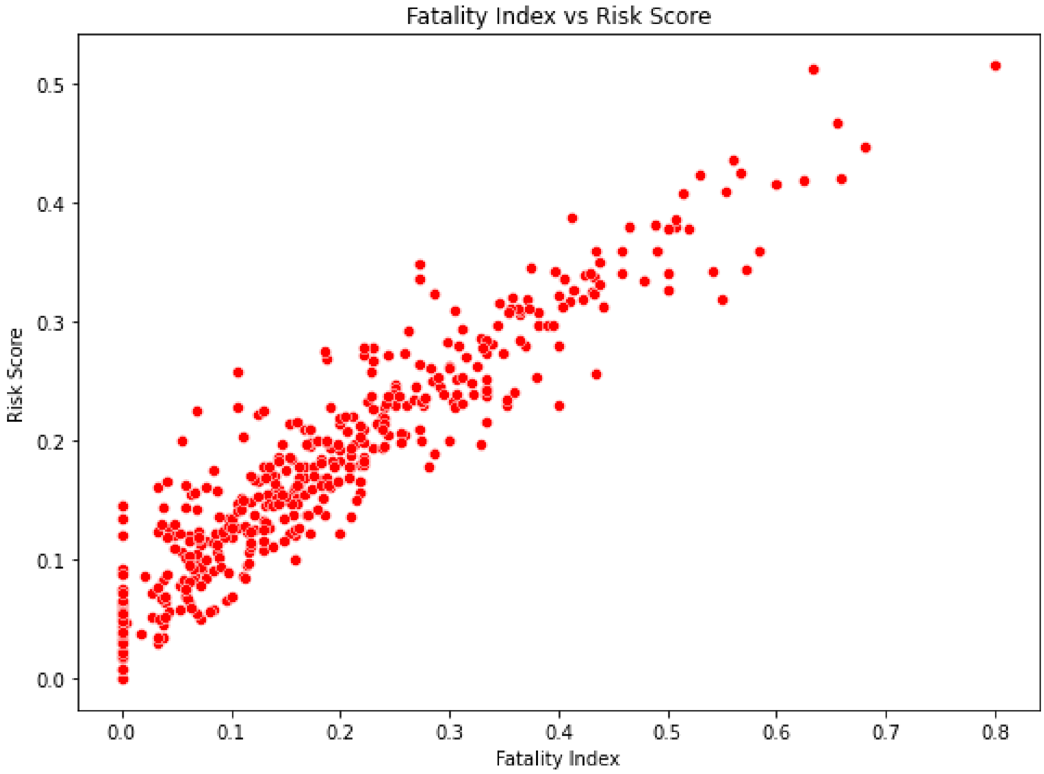
```
plt.figure(figsize=(15, 8))
sns.heatmap(
    model_summary_df[['fatality_index', 'damage_severity_index', 'inj
    annot=True,
    cmap='coolwarm',
    fmt='.2f'
)
plt.title("Correlation Matrix of Risk Indexes")
plt.tight_layout()
plt.show()
```



In [15]:

```
# Scatter plot: Damage Severity Index vs Risk Score
plt.figure(figsize=(8, 6))
sns.scatterplot(data=model_summary_df, x='damage_severity_index', y='r
plt.title('Damage Severity Index vs Risk Score')
plt.xlabel('Damage Severity Index')
plt.ylabel('Risk Score')
plt.tight_layout()
plt.show()

# Scatter plot: Fatality Index vs Risk Score
plt.figure(figsize=(8, 6))
sns.scatterplot(data=model_summary_df, x='fatality_index', y='risk_sco
plt.title('Fatality Index vs Risk Score')
plt.xlabel('Fatality Index')
plt.ylabel('Risk Score')
plt.tight_layout()
plt.show()
```

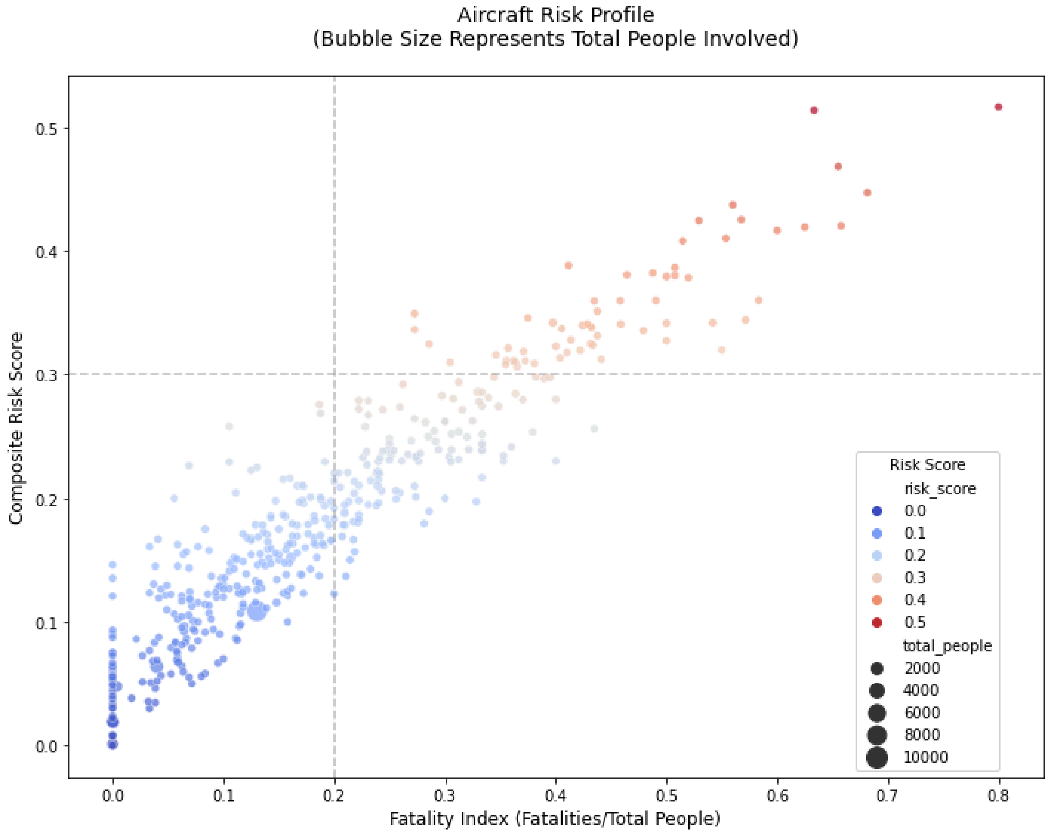




Fatality Index vs Risk Score
Risk Score
Fatality Index
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8

In [16]:

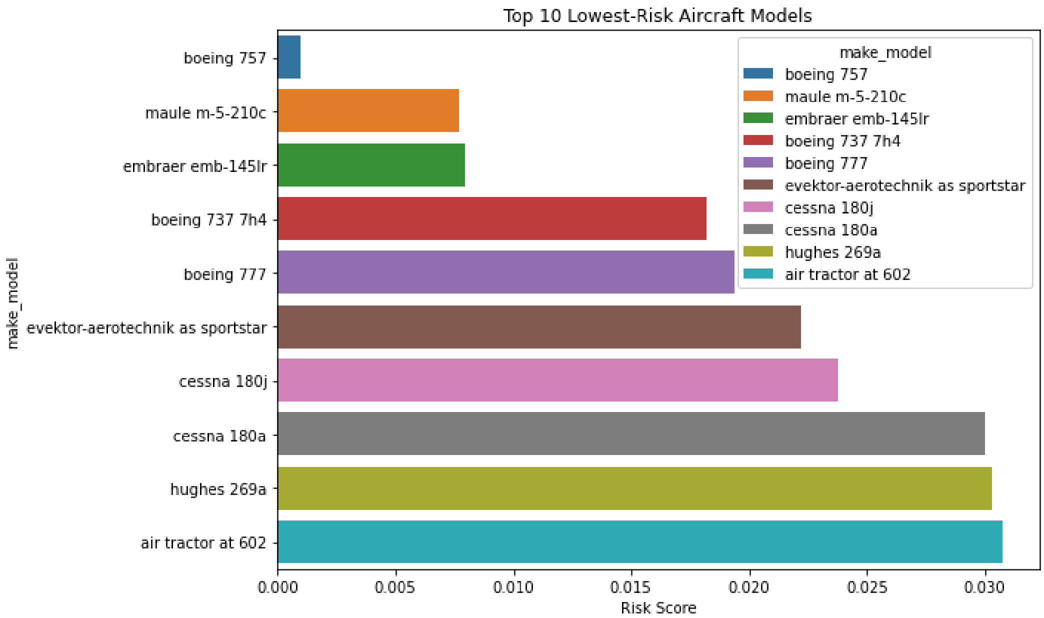
```python
plt.figure(figsize=(10, 8))  # Increased figure size

# Create the scatter plot
scatter = sns.scatterplot(
    data=model_summary_df,
    x='fatality_index',
    y='risk_score',
    size='total_people',
    hue='risk_score',
    palette='coolwarm',
    sizes=(30, 200),
    alpha=0.7
)

# Add reference lines
plt.axhline(0.3, linestyle='--', color='gray', alpha=0.5)
plt.axvline(0.2, linestyle='--', color='gray', alpha=0.5)

# Customize titles and labels
plt.title("Aircraft Risk Profile\n(Bubble Size Represents Total People
plt.xlabel("Fatality Index (Fatalities/Total People)", fontsize=12)
plt.ylabel("Composite Risk Score", fontsize=12)



# Method 2: If you really want bottom-left inside the plot
plt.legend(
     bbox_to_anchor=(0.80, 0.0),  # Inside bottom-left
     loc='lower left',
     borderaxespad=0.5,
     frameon=True,
     title='Risk Score'
 )
# Add tight_layout
plt.tight_layout()

plt.show()
```



Aircraft Risk Profile
(Bubble Size Represents Total People Involved)
Composite Risk Score
Fatality Index (Fatalities/Total People)
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
Risk Score
risk_score
0.0
0.1
0.2
0.3
0.4
0.5
total_people
2000
4000
6000
8000
10000

In [17]:

```python
# Filter out models with zero risk score
filtered_df = model_summary_df[model_summary_df['risk_score'] > 0]

# Sort top 10 lowest non-zero risk models
top_10 = filtered_df.sort_values('risk_score').head(10)

plt.figure(figsize=(10, 6))
sns.barplot(
    data=top_10,
    x='risk_score',
    y='make_model',
    hue='make_model',
    dodge=False
)
plt.title("Top 10 Lowest-Risk Aircraft Models")
plt.xlabel("Risk Score")
plt.ylabel("make_model")
plt.tight_layout()
plt.show()

# Display summary table with key stats
top_10[[
    'make_model', 'total_accidents', 'total_people',
    'fatality_index', 'damage_severity_index', 'injury_index', 'risk_s
]].round(4)
```



Out[17]:

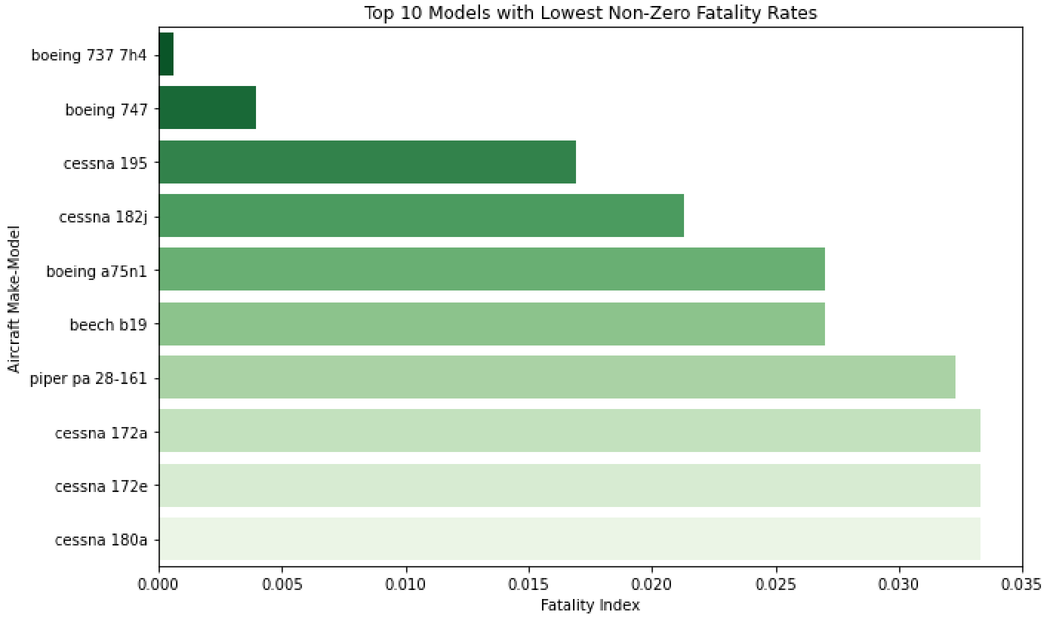
	make_model	total_accidents	total_people	fatality_index	damage_severity_index
1428	boeing 757	16	1810.0	0.0000	0.0000
4759	maule m-5-210c	12	26.0	0.0000	0.0000
3010	embraer emb-145lr	10	426.0	0.0000	0.0000
1387	boeing 737 7h4	14	1655.0	0.0006	0.0000
1460	boeing 777	17	2422.0	0.0000	0.0588
3202	evektor-aerotechnik as sportstar	20	27.0	0.0000	0.0000
1938	cessna 180j	25	42.0	0.0000	0.0000
1930	cessna 180a	14	30.0	0.0333	0.0000
4047	hughes 269a	20	33.0	0.0000	0.0000
283	air tractor at 602	12	13.0	0.0000	0.0000

In [18]:

```
# Remove models with zero risk score as indication of limited data for
model_summary_df = model_summary_df[model_summary_df['risk_score'] > (

# filter models with zero fatality, damage and injury index for respec
fatality_filtered = model_summary_df[model_summary_df['fatality_index
damage_filtered = model_summary_df[model_summary_df['damage_severity_i
injury_filtered = model_summary_df[model_summary_df['injury_index'] >
```

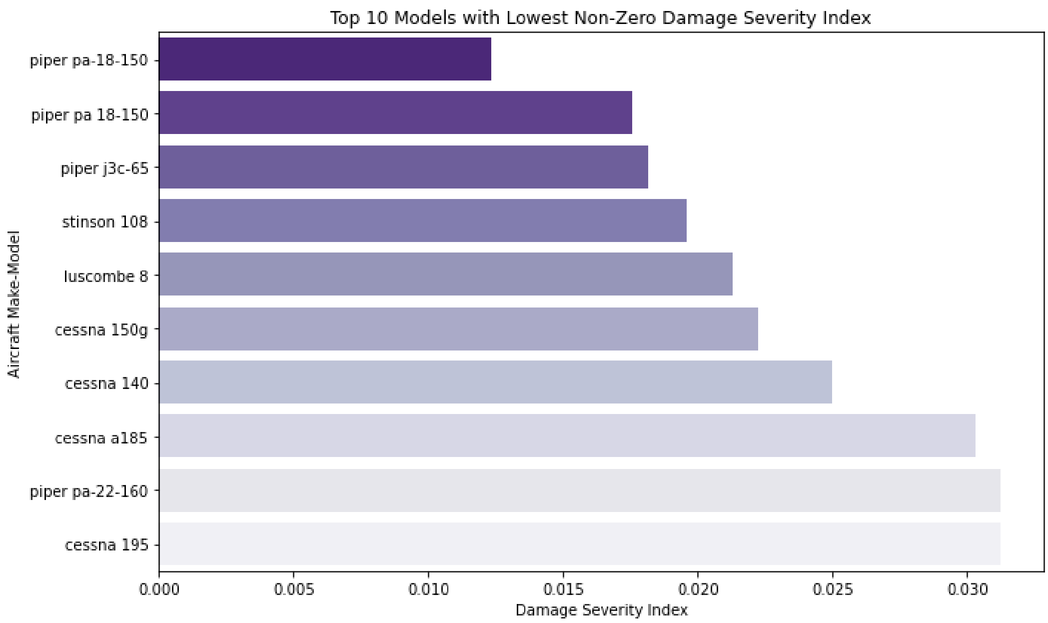
In [19]:

```
# Sort and select top 10
lowest_fatality = fatality_filtered.sort_values('fatality_index').head

# Plot
plt.figure(figsize=(10,6))
sns.barplot(data=lowest_fatality, x='fatality_index', y='make_model',
plt.title("Top 10 Models with Lowest Non-Zero Fatality Rates")
plt.xlabel("Fatality Index")
plt.ylabel("Aircraft Make-Model")
plt.tight_layout()
plt.show()
```



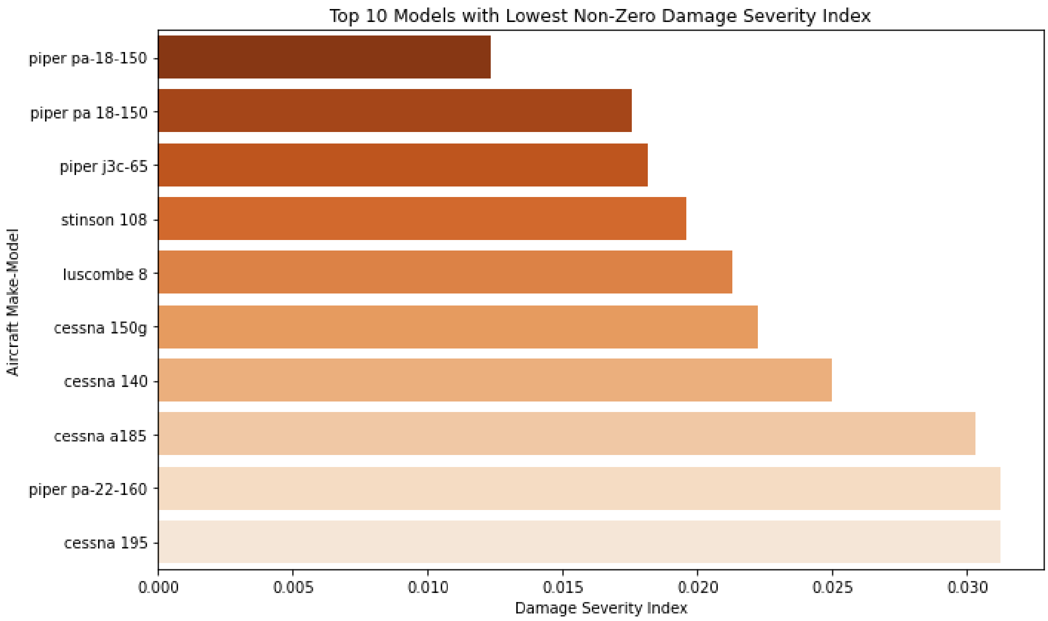
In [20]:

```
# Top 10 models with lowest non-zero Damage Severity Index
damage_filtered = model_summary_df[model_summary_df['damage_severity_i
lowest_damage = damage_filtered.sort_values('damage_severity_index').l

plt.figure(figsize=(10,6))
sns.barplot(data=lowest_damage, x='damage_severity_index', y='make_mo
plt.title("Top 10 Models with Lowest Non-Zero Damage Severity Index")
plt.xlabel("Damage Severity Index")
plt.ylabel("Aircraft Make-Model")
plt.tight_layout()
plt.show()
```



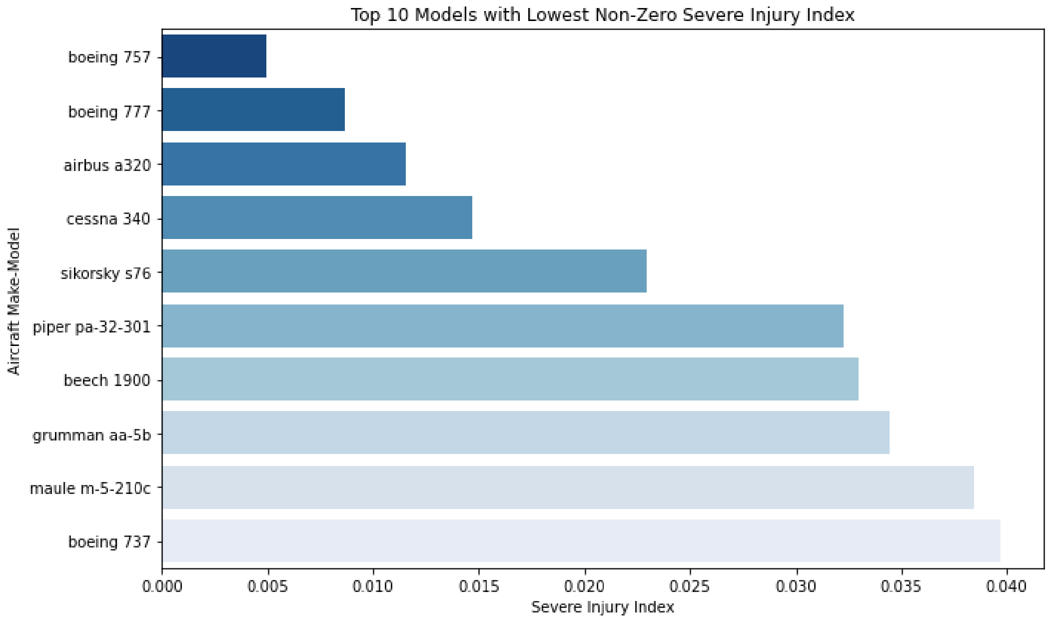
In [21]:

```python
# Top 10 models with lowest non-zero Damage Severity Index
damage_filtered = model_summary_df[model_summary_df['damage_severity_i
lowest_damage = damage_filtered.sort_values('damage_severity_index').h

plt.figure(figsize=(10,6))
sns.barplot(data=lowest_damage, x='damage_severity_index', y='make_mod
plt.title("Top 10 Models with Lowest Non-Zero Damage Severity Index")
plt.xlabel("Damage Severity Index")
plt.ylabel("Aircraft Make-Model")
plt.tight_layout()
plt.show()
```



In [22]:

```
# Top 10 models with lowest non-zero Injury Index
lowest_injury = injury_filtered.sort_values('injury_index').head(10)

plt.figure(figsize=(10,6))
sns.barplot(data=lowest_injury, x='injury_index', y='make_model', pal
plt.title("Top 10 Models with Lowest Non-Zero Severe Injury Index")
plt.xlabel("Severe Injury Index")
plt.ylabel("Aircraft Make-Model")
plt.tight_layout()
plt.show()
```



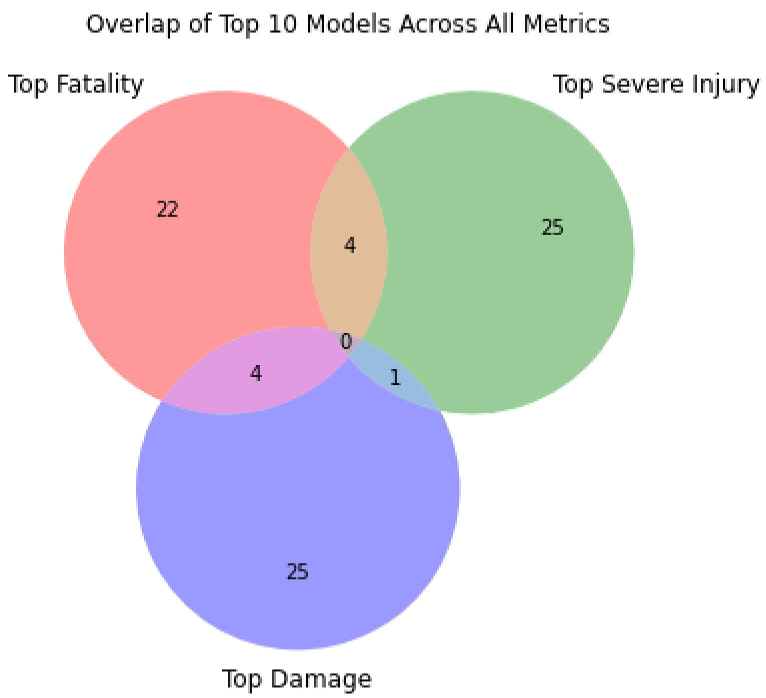
Data Analysis

Recommend the aircraft with the lowest fatality, injury, damage and overall risk i.e. the ones that itersect across all the metrics.

In [23]:

```
# Increase range
top_n = 30
top_fatality = fatality_filtered.sort_values('fatality_index').head(t
top_injury = injury_filtered.sort_values('injury_index').head(top_n)[
top_risk = model_summary_df.sort_values('risk_score').head(top_n)['ma
top_damage = damage_filtered.sort_values('damage_severity_index').hea

# Intersection
common_models = set(top_fatality) & set(top_injury) & set(top_risk)

if common_models:
    print(f"✅ Models appearing in top {top_n} for all 3 metrics:")
    print(common_models)
else:
    print(f"❌ No exact overlap in top {top_n}. Computing combined ra

# Compute combined rank
model_summary_df['rank_fatality'] = model_summary_df['fatality_index'
model_summary_df['rank_injury'] = model_summary_df['injury_index'].ra
model_summary_df['rank_risk'] = model_summary_df['risk_score'].rank(m
model_summary_df['rank_damage'] = model_summary_df['damage_severity_i

# Compute combined rank across 4 metrics
model_summary_df['combined_rank'] = (
    model_summary_df['rank_fatality'] +
    model_summary_df['rank_injury'] +
    model_summary_df['rank_risk'] +
    model_summary_df['rank_damage']
)

# Sort by combined rank
combined_top = model_summary_df.sort_values('combined_rank').head(10)
print("\n✅ Top 10 Models by Combined Safety Rank:")
print(combined_top[['make_model', 'fatality_index', 'injury_index', '

# Optional: Venn Diagram for visualization
from matplotlib_venn import venn3

plt.figure(figsize=(8,6))
venn3([set(top_fatality), set(top_injury), set(top_damage)],
      set_labels=('Top Fatality', 'Top Severe Injury', 'Top Damage'))
plt.title("Overlap of Top 10 Models Across All Metrics")
plt.show()
```

```
✅ Models appearing in top 30 for all 3 metrics:
{'cessna 180a', 'cessna 195a', 'boeing 747'}

✅ Top 10 Models by Combined Safety Rank:
                            make_model  fatality_index  injury_index  \
1428                        boeing 757        0.000000      0.004972
4759                    maule m-5-210c        0.000000      0.038462
3010                 embraer emb-145lr        0.000000      0.039906
1930                       cessna 180a        0.033333      0.066667
3202  evektor-aerotechnik as sportstar        0.000000      0.111111
1387                    boeing 737 7h4        0.000604      0.089426
1938                       cessna 180j        0.000000      0.119048
1981                       cessna 195a        0.038462      0.076923
5632                   piper pa 28-161        0.032258      0.096774
1460                        boeing 777        0.000000      0.008671

      damage_severity_index  risk_score  combined_rank
1428               0.000000    0.000994           10.0
4759               0.000000    0.007692           19.0
3010               0.000000    0.007981           23.0
1930               0.000000    0.030000           78.0
3202               0.000000    0.022222           80.0
1387               0.000000    0.018187           90.0
1938               0.000000    0.023810           91.0
1981               0.000000    0.034615           95.0
5632               0.000000    0.035484          115.0
1460               0.058824    0.019381          117.0
```



In [24]:

```
# Ensure make_model column exists and is normalized
if 'make_model' not in model_summary_df.columns:
    model_summary_df['make_model'] = (model_summary_df['Make'] + ' ' +
else:
    model_summary_df['make_model'] = model_summary_df['make_model'].s

# Define columns for export
export_cols = [
    'make_model',
    'total_accidents', 'total_people',
    'fatality_index', 'injury_index', 'damage_severity_index', 'risk_
]

# Check if all columns exist
missing_cols = [col for col in export_cols if col not in model_summar
if missing_cols:
    print(f"⚠️ Missing columns: {missing_cols}")
else:
    # Export CSV and Excel
    model_summary_df[export_cols].to_csv('Aviation_Safety_Tableau.csv
    model_summary_df[export_cols].to_excel('Aviation_Safety_Tableau.x
    print("✅ Export completed successfully!")

# Show a preview of exported data
model_summary_df[export_cols].head(10)
```

✅ Export completed successfully!

Out[24]:

	make_model	total_accidents	total_people	fatality_index	injury_index	damage_s
32	aero commander 100	13	21.0	0.095238	0.285714	
71	aero commander s2r	18	18.0	0.222222	0.222222	
98	aeronca 11ac	29	50.0	0.140000	0.340000	
101	aeronca 15ac	10	12.0	0.083333	0.083333	
110	aeronca 7ac	85	129.0	0.162791	0.286822	
113	aeronca 7bcm	14	17.0	0.058824	0.470588	
115	aeronca 7ccm	10	16.0	0.000000	0.312500	
163	aerospatiale as350	15	32.0	0.281250	0.093750	
233	agusta a109	11	30.0	0.633333	0.166667	
283	air tractor at 602	12	13.0	0.000000	0.153846	

Aircraft Safety Analysis – Recommended Models

Based on the computed safety indices (**Fatality Index**, **Injury Index**, **Damage Severity Index**) and overall **Risk Score**, here are the insights deduced:

Insights

1. **Models with lowest risk scores** tend to have fewer accidents and lower fatality ratios.
2. **Purpose of flight patterns** show that some of these safer models are commonly used for **personal purposes**.
3. **Engine configurations** (type and number) may indicate suitability for specific operations.

✅ Recommendations for Client:

- _*Personal Use:_ For private operations, prioritize single-engine piston types with historically low injury rates.
- **Top 10 models as illustrated in the bar graph with the boeing 757 being the safest evaluated model too invest in.