Final Project Submission

Please fill out:

- Student name: Jeffrey Gathigi
- Student pace: part time
- Scheduled project review date/time:
- Instructor name: Brian Chacha
- Blog post URL:

```
# Your code here - remember to use markdown cells for comments as well!
```

Aviation Safety Analysis: Strategic Aircraft Acquisition Recommendations

Business Understanding

Our company is expanding into aviation by purchasing and operating aircraft for commercial and private enterprises. As data scientists, we need to identify the lowest-risk aircraft options and operational strategies to guide this new business venture.

Business Problem:

Which aircraft present the lowest risk for our new aviation division, and what operational strategies should we implement to maximize safety?

Key Questions:

Which aircraft models have the best safety records?

What operational factors (weather, flight phase, purpose) most impact safety?

How should we structure our operations to minimize risk?

Data Understanding

Let's first explore our dataset to understand its structure and quality.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from datetime import datetime
import warnings
warnings.filterwarnings('ignore')
```

```
# Load the dataset
df = pd.read_csv('Aviation_Data.csv')

print("Dataset Shape:", df.shape)
print("\nColumn Names:")
print(df.columns.tolist())
print("\nBasic Info:")
df.info()
df
```

```
Dataset Shape: (90348, 31)

Column Names:
['Event.Id', 'Investigation.Type', 'Accident.Number', 'Event.Date', 'Location', 'Country', 'Latitude', 'Longitude', 'Airport

Basic 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
 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        73659 non-null  object
dtypes: float64(5), object(26)
memory usage: 21.4+ MB
```

	Event.Id	Investigation.Type	Accident.Number	Event.Date	Location	Country	Latitude	Longitude	Airport
0	20001218X45444	Accident	SEA87LA080	1948-10-24	MOOSE CREEK, ID	United States	NaN	NaN	
1	20001218X45447	Accident	LAX94LA336	1962-07-19	BRIDGEPORT, CA	United States	NaN	NaN	
2	20061025X01555	Accident	NYC07LA005	1974-08-30	Saltville, VA	United States	36.922223	-81.878056	
3	20001218X45448	Accident	LAX96LA321	1977-06-19	EUREKA, CA	United States	NaN	NaN	
4	20041105X01764	Accident	CHI79FA064	1979-08-02	Canton, OH	United States	NaN	NaN	
...	...	...	...	...	...	...	...	...	
90343	20221227106491	Accident	ERA23LA093	2022-12-26	Annapolis, MD	United States	NaN	NaN	
90344	20221227106494	Accident	ERA23LA095	2022-12-26	Hampton, NH	United States	NaN	NaN	
90345	20221227106497	Accident	WPR23LA075	2022-12-26	Payson, AZ	United States	341525N	1112021W	
90346	20221227106498	Accident	WPR23LA076	2022-12-26	Morgan, UT	United States	NaN	NaN	
90347	20221230106513	Accident	ERA23LA097	2022-12-29	Athens, GA	United States	NaN	NaN	

90348 rows × 31 columns

```
df.describe
```

```
1              NaN  ...          Personal                 NaN
2              NaN  ...          Personal                 NaN
3              NaN  ...          Personal                 NaN
4              NaN  ...          Personal                 NaN
...            ...  ...               ...                 ...
90343          NaN  ...          Personal                 NaN
90344          NaN  ...               NaN                 NaN
90345       PAYSON  ...          Personal                 NaN
90346          NaN  ...          Personal  MC CESSNA 210N LLC
90347          NaN  ...          Personal                 NaN

      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
...                    ...                    ...                  ...
90343                  0.0                    1.0                  0.0
90344                  0.0                    0.0                  0.0
90345                  0.0                    0.0                  0.0
90346                  0.0                    0.0                  0.0
90347                  0.0                    1.0                  0.0

      Total.Uninjured Weather.Condition  Broad.phase.of.flight  \
0                 0.0               UNK                 Cruise
1                 0.0               UNK                Unknown
2                 NaN               IMC                 Cruise
3                 0.0               IMC                 Cruise
4                 0.0               VMC               Approach
...               ...               ...                    ...
90343             0.0               NaN                    NaN
90344             0.0               NaN                    NaN
90345             1.0               VMC                    NaN
90346             0.0               NaN                    NaN
90347             1.0               NaN                    NaN

        Report.Status Publication.Date
0      Probable Cause              NaN
1      Probable Cause       19-09-1996
2      Probable Cause       26-02-2007
3      Probable Cause       12-09-2000
4      Probable Cause       16-04-1980
...               ...              ...
90343             NaN       29-12-2022
90344             NaN              NaN
90345             NaN       27-12-2022
90346             NaN              NaN
90347             NaN       30-12-2022

[90348 rows x 31 columns]>
```

```python
# Initial data exploration
print("First few rows:")
display(df.head())

print("\nMissing values by column:")
missing_data = df.isnull().sum()
missing_percent = (missing_data / len(df)) * 100
missing_info = pd.DataFrame({
    'Missing Count': missing_data,
    'Missing Percentage': missing_percent
})
display(missing_info[missing_info['Missing Count'] > 0].sort_values('Missing Count', ascending=False))
```

First few rows:

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

5 rows × 31 columns

Missing values by column:

	Missing Count	Missing Percentage
Schedule	77766	86.073848
Air.carrier	73700	81.573471
FAR.Description	58325	64.555939
Aircraft.Category	58061	64.263736
Longitude	55975	61.954886
Latitude	55966	61.944924
Airport.Code	40099	44.382831
Airport.Name	37558	41.570372
Broad.phase.of.flight	28624	31.681941
Publication.Date	16689	18.471909
Total.Serious.Injuries	13969	15.461327
Total.Minor.Injuries	13392	14.822686
Total.Fatal.Injuries	12860	14.233851
Engine.Type	8536	9.447913
Report.Status	7840	8.677558
Purpose.of.flight	7651	8.468367
Number.of.Engines	7543	8.348829
Total.Uninjured	7371	8.158454
Weather.Condition	5951	6.586753
Aircraft.damage	4653	5.150086
Registration.Number	2776	3.072564
Injury.Severity	2459	2.721698
Country	1685	1.865011
Amateur.Built	1561	1.727764
Model	1551	1.716695
Make	1522	1.684597
Location	1511	1.672422
Accident.Number	1459	1.614867
Event.Date	1459	1.614867
Event.Id	1459	1.614867

```
# Key columns for our analysis
key_columns = [
    'Event.Id', 'Investigation.Type', 'Event.Date', 'Injury.Severity',
    'Aircraft.damage', 'Aircraft.Category', 'Make', 'Model',
    'Number.of.Engines', 'Engine.Type', 'FAR.Description', 'Purpose.of.flight',
    'Total.Fatal.Injuries', 'Total.Serious.Injuries', 'Total.Minor.Injuries',
    'Total.Uninjured', 'Weather.Condition', 'Broad.phase.of.flight'
]
```

```
print("Summary statistics for key numerical columns:")
display(df[['Total.Fatal.Injuries', 'Total.Serious.Injuries', 'Total.Minor.Injuries', 'Total.Uninjured']].describe())
```

Summary statistics for key numerical columns:

	Total.Fatal.Injuries	Total.Serious.Injuries	Total.Minor.Injuries	Total.Uninjured
count	77488.000000	76379.000000	76956.000000	82977.000000
mean	0.647855	0.279881	0.357061	5.325440
std	5.485960	1.544084	2.235625	27.913634
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

Data Preparation

Now let's clean and prepare our data for analysis.

```
# Create a copy for cleaning
df_clean = df.copy()

print("Initial dataset shape:", df_clean.shape)
```

```
Initial dataset shape: (90348, 31)
```

```
# Handle missing values in key columns
print("Handling missing values...")

# For injury columns, assume missing means 0
injury_columns = ['Total.Fatal.Injuries', 'Total.Serious.Injuries', 'Total.Minor.Injuries', 'Total.Uninjured']
df_clean[injury_columns] = df_clean[injury_columns].fillna(0)

# Create a fatal accident flag
df_clean['Is_Fatal'] = df_clean['Total.Fatal.Injuries'] > 0
df_clean['Is_Fatal'] = df_clean['Is_Fatal'].fillna(False)

# For categorical columns, create 'Unknown' category
categorical_columns = ['Weather.Condition', 'Broad.phase.of.flight', 'FAR.Description', 'Purpose.of.flight']
for col in categorical_columns:
    df_clean[col] = df_clean[col].fillna('UNK')

# Filter for accidents only (exclude incidents)
df_accidents = df_clean[df_clean['Investigation.Type'] == 'Accident'].copy()
print(f"After filtering for accidents only: {df_accidents.shape}")

# Extract year from event date for temporal analysis
df_accidents['Event.Date'] = pd.to_datetime(df_accidents['Event.Date'], errors='coerce')
df_accidents['Year'] = df_accidents['Event.Date'].dt.year

print(f"Final cleaned dataset shape: {df_accidents.shape}")
```

```
Handling missing values...
After filtering for accidents only: (85015, 32)
Final cleaned dataset shape: (85015, 33)
```

```
df_accidents
```

	Event.Id	Investigation.Type	Accident.Number	Event.Date	Location	Country	Latitude	Longitude	Airport
0	20001218X45444	Accident	SEA87LA080	1948-10-24	MOOSE CREEK, ID	United States	NaN	NaN	
1	20001218X45447	Accident	LAX94LA336	1962-07-19	BRIDGEPORT, CA	United States	NaN	NaN	
2	20061025X01555	Accident	NYC07LA005	1974-08-30	Saltville, VA	United States	36.922223	-81.878056	
3	20001218X45448	Accident	LAX96LA321	1977-06-19	EUREKA, CA	United States	NaN	NaN	
4	20041105X01764	Accident	CHI79FA064	1979-08-02	Canton, OH	United States	NaN	NaN	
...	...	...	...	...	...	...	...	...	
90343	20221227106491	Accident	ERA23LA093	2022-12-26	Annapolis, MD	United States	NaN	NaN	
90344	20221227106494	Accident	ERA23LA095	2022-12-26	Hampton, NH	United States	NaN	NaN	
90345	20221227106497	Accident	WPR23LA075	2022-12-26	Payson, AZ	United States	341525N	1112021W	
90346	20221227106498	Accident	WPR23LA076	2022-12-26	Morgan, UT	United States	NaN	NaN	
90347	20221230106513	Accident	ERA23LA097	2022-12-29	Athens, GA	United States	NaN	NaN	

85015 rows × 33 columns

```
# Create additional derived features
# Combine Make and Model for full aircraft identification
df_accidents['Aircraft_Full_Name'] = df_accidents['Make'] + ' ' + df_accidents['Model']

# Create total occupants feature
df_accidents['Total_Occupants'] = (
    df_accidents['Total.Fatal.Injuries'] +
    df_accidents['Total.Serious.Injuries'] +
    df_accidents['Total.Minor.Injuries'] +
    df_accidents['Total.Uninjured']
)

# Create survival rate
df_accidents['Survival_Rate'] = (
    (df_accidents['Total.Uninjured'] + df_accidents['Total.Minor.Injuries'] + df_accidents['Total.Serious.Injuries']) /
    df_accidents['Total_Occupants']
)
df_accidents['Survival_Rate'] = df_accidents['Survival_Rate'].fillna(0)

print("Data preparation complete.")
print(f"Total accidents in analysis: {len(df_accidents)}")
print(f"Fatal accidents: {df_accidents['Is_Fatal'].sum()} ({df_accidents['Is_Fatal'].mean()*100:.1f}%)")
```

```
Data preparation complete.
Total accidents in analysis: 85015
Fatal accidents: 17794 (20.9%)
```

Data Analysis

Now let's analyze the data to answer our key business questions.

1. Aircraft Model Safety Analysis

```
# Analyze aircraft models by accident frequency and severity
model_analysis = df_accidents.groupby(['Make', 'Model']).agg({
    'Event.Id': 'count',
    'Is_Fatal': 'sum',
    'Total.Fatal.Injuries': 'sum',
    'Total_Occupants': 'sum',
    'Survival_Rate': 'mean'
}).rename(columns={
    'Event.Id': 'Total_Accidents',
    'Is_Fatal': 'Fatal_Accidents'
})
```

```
model_analysis['Fatal_Rate'] = model_analysis['Fatal_Accidents'] / model_analysis['Total_Accidents']
model_analysis['Avg_Fatalities_Per_Accident'] = model_analysis['Total.Fatal.Injuries'] / model_analysis['Total_Accidents']

# Filter for models with sufficient data (at least 5 accidents)
model_analysis_filtered = model_analysis[model_analysis['Total_Accidents'] >= 5].copy()

print("Top 20 Aircraft Models by Number of Accidents:")
top_models = model_analysis_filtered.sort_values('Total_Accidents', ascending=False).head(20)
display(top_models)
```

Top 20 Aircraft Models by Number of Accidents:

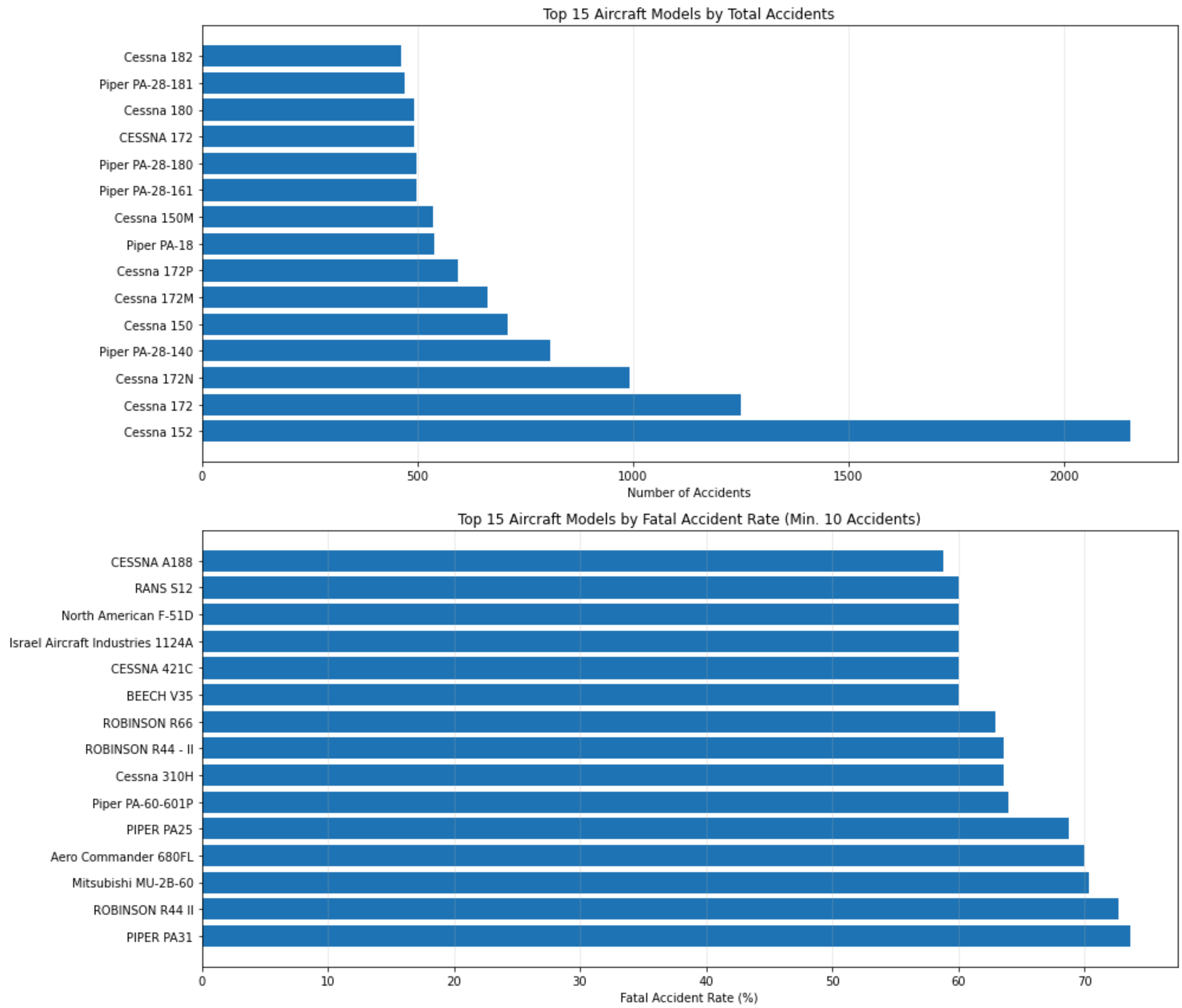
Make	Model	Total_Accidents	Fatal_Accidents	Total.Fatal.Injuries	Total_Occupants	Survival_Rate	Fatal_Rate	Avg_Fat
Cessna	152	2155	222	349.0	3067.0	0.904532	0.103016	
	172	1250	115	231.0	2476.0	0.920022	0.092000	
	172N	993	173	365.0	2009.0	0.842502	0.174220	
Piper	PA-28-140	809	156	284.0	1606.0	0.830171	0.192831	
Cessna	150	711	58	78.0	1035.0	0.927332	0.081575	
	172M	663	111	201.0	1397.0	0.863625	0.167421	
	172P	594	97	227.0	1195.0	0.856558	0.163300	
Piper	PA-18	539	53	82.0	831.0	0.914811	0.098330	
Cessna	150M	537	58	91.0	811.0	0.901237	0.108007	
Piper	PA-28-180	498	128	238.0	1064.0	0.772055	0.257028	
	PA-28-161	498	100	209.0	1049.0	0.817704	0.200803	
CESSNA	172	494	96	167.0	853.0	0.806680	0.194332	
Cessna	180	494	41	91.0	1011.0	0.930601	0.082996	
Piper	PA-28-181	472	128	359.0	1204.0	0.751000	0.271186	
Cessna	182	462	59	110.0	986.0	0.888240	0.127706	

```
# Visualize top aircraft models by accident count and fatal rate
fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(14, 12))

# Plot 1: Total accidents by model
top_accident_models = model_analysis_filtered.nlargest(15, 'Total_Accidents')
ax1.barh(range(len(top_accident_models)), top_accident_models['Total_Accidents'])
ax1.set_yticks(range(len(top_accident_models)))
ax1.set_yticklabels([f"{idx[0]} {idx[1]}" for idx in top_accident_models.index])
ax1.set_xlabel('Number of Accidents')
ax1.set_title('Top 15 Aircraft Models by Total Accidents')
ax1.grid(axis='x', alpha=0.3)

# Plot 2: Fatal rate by model (for models with at least 10 accidents)
high_volume_models = model_analysis_filtered[model_analysis_filtered['Total_Accidents'] >= 10]
high_fatal_rate = high_volume_models.nlargest(15, 'Fatal_Rate')
ax2.barh(range(len(high_fatal_rate)), high_fatal_rate['Fatal_Rate'] * 100)
ax2.set_yticks(range(len(high_fatal_rate)))
ax2.set_yticklabels([f"{idx[0]} {idx[1]}" for idx in high_fatal_rate.index])
ax2.set_xlabel('Fatal Accident Rate (%)')
ax2.set_title('Top 15 Aircraft Models by Fatal Accident Rate (Min. 10 Accidents)')
ax2.grid(axis='x', alpha=0.3)

plt.tight_layout()
plt.show()
```




```
# Identify promising aircraft with good safety records
promising_models = model_analysis_filtered[
    (model_analysis_filtered['Total_Accidents'] >= 5) &
    (model_analysis_filtered['Fatal_Rate'] < 0.2) &
    (model_analysis_filtered['Survival_Rate'] > 0.8)
].sort_values('Fatal_Rate')

print("Promising Aircraft Models with Good Safety Records:")
display(promising_models.head(10))
```

```
Promising Aircraft Models with Good Safety Records:
```

Make	Model	Total_Accidents	Fatal_Accidents	Total.Fatal.Injuries	Total_Occupants	Survival_Rate	Fatal_Rate	Avg_F
Zenair	CH 2000	5	0	0.0	6.0	1.0	0.0	
Maule	MXT-7-180A	5	0	0.0	7.0	1.0	0.0	
Mcdonnell Douglas	600N	7	0	0.0	18.0	1.0	0.0	
CESSNA	175A	8	0	0.0	13.0	1.0	0.0	
Mcdonnell Douglas	MD-11F	5	0	0.0	277.0	1.0	0.0	
	MD-80	7	0	0.0	841.0	1.0	0.0	
Mooney	M-							

2. Operational Factors Analysis

```
# Analyze safety by weather conditions
weather_analysis = df_accidents.groupby('Weather.Condition').agg({
    'Event.Id': 'count',
    'Is_Fatal': 'sum',
    'Total.Fatal.Injuries': 'sum'
}).rename(columns={'Event.Id': 'Total_Accidents', 'Is_Fatal': 'Fatal_Accidents'})

weather_analysis['Fatal_Rate'] = weather_analysis['Fatal_Accidents'] / weather_analysis['Total_Accidents']
weather_analysis = weather_analysis.sort_values('Total_Accidents', ascending=False)

print("Accident Analysis by Weather Condition:")
display(weather_analysis)
```

Accident Analysis by Weather Condition:

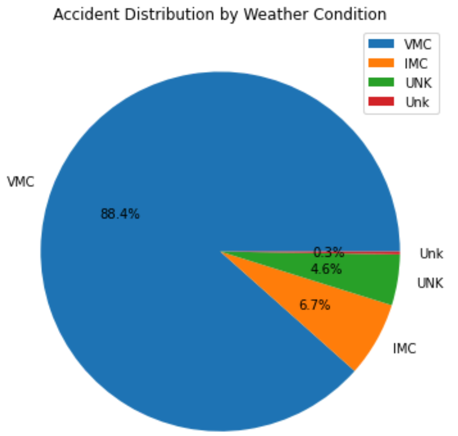
Weather.Condition	Total_Accidents	Fatal_Accidents	Total.Fatal.Injuries	Fatal_Rate
VMC	75186	12242	25540.0	0.162823
IMC	5725	3462	11823.0	0.604716
UNK	3889	1954	12490.0	0.502443
Unk	215	136	326.0	0.632558

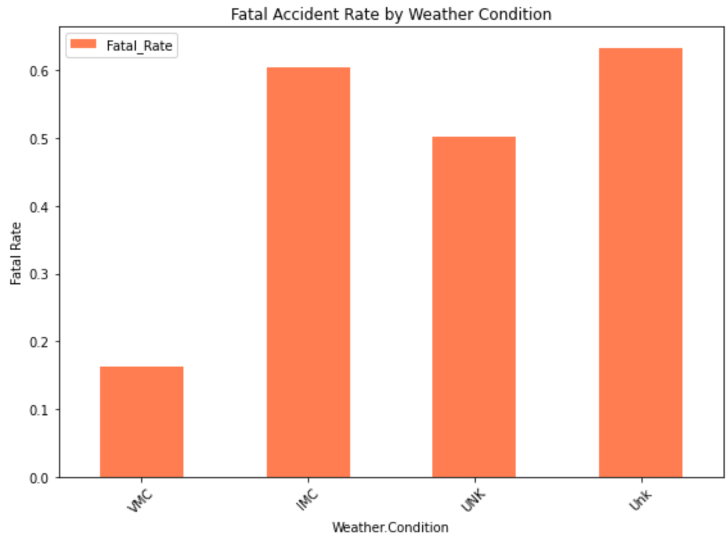
```
# Visualize weather impact
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(15, 6))

# Accident count by weather
weather_analysis.plot(y='Total_Accidents', kind='pie', ax=ax1, autopct='%1.1f%%')
ax1.set_ylabel('')
ax1.set_title('Accident Distribution by Weather Condition')

# Fatal rate by weather
weather_analysis.plot(y='Fatal_Rate', kind='bar', ax=ax2, color='coral')
ax2.set_title('Fatal Accident Rate by Weather Condition')
ax2.set_ylabel('Fatal Rate')
ax2.tick_params(axis='x', rotation=45)

plt.tight_layout()
plt.show()
```






```
# Analyze safety by flight phase
phase_analysis = df_accidents.groupby('Broad.phase.of.flight').agg({
    'Event.Id': 'count',
    'Is_Fatal': 'sum',
    'Total.Fatal.Injuries': 'sum'
}).rename(columns={'Event.Id': 'Total_Accidents', 'Is_Fatal': 'Fatal_Accidents'})

phase_analysis['Fatal_Rate'] = phase_analysis['Fatal_Accidents'] / phase_analysis['Total_Accidents']
phase_analysis = phase_analysis.sort_values('Total_Accidents', ascending=False)

print("Accident Analysis by Flight Phase:")
display(phase_analysis)
```

Accident Analysis by Flight Phase:

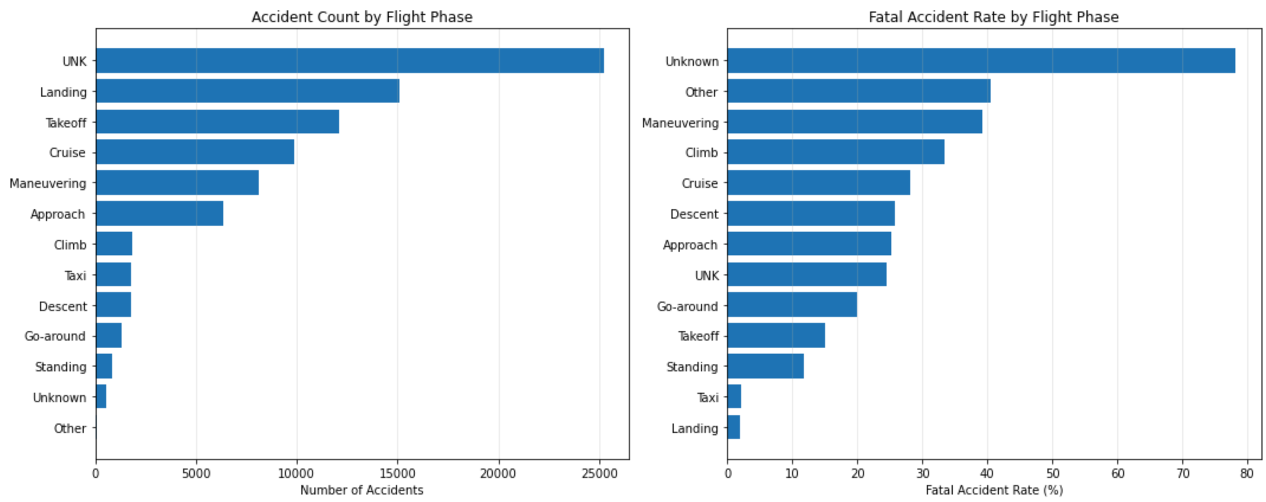
Broad.phase.of.flight	Total_Accidents	Fatal_Accidents	Total.Fatal.Injuries	Fatal_Rate
UNK	25209	6172	25677.0	0.244833
Landing	15074	291	517.0	0.019305
Takeoff	12133	1824	4304.0	0.150334
Cruise	9904	2781	6168.0	0.280796
Maneuvering	8107	3183	5323.0	0.392624
Approach	6338	1597	3840.0	0.251972
Climb	1850	618	1762.0	0.334054
Taxi	1783	39	99.0	0.021873
Descent	1778	457	913.0	0.257030
Go-around	1338	267	587.0	0.199552
Standing	854	101	155.0	0.118267
Unknown	536	419	749.0	0.781716
Other	111	45	85.0	0.405405

```
# Visualize flight phase impact - Simplified bar charts
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(15, 6))

# Accident count by phase - Bar chart
phase_analysis_sorted = phase_analysis.sort_values('Total_Accidents', ascending=True)
ax1.barh(range(len(phase_analysis_sorted)), phase_analysis_sorted['Total_Accidents'])
ax1.set_yticks(range(len(phase_analysis_sorted)))
ax1.set_yticklabels(phase_analysis_sorted.index)
ax1.set_xlabel('Number of Accidents')
ax1.set_title('Accident Count by Flight Phase')
ax1.grid(axis='x', alpha=0.3)

# Fatal rate by phase - Bar chart
phase_analysis_fatal_sorted = phase_analysis.sort_values('Fatal_Rate', ascending=True)
ax2.barh(range(len(phase_analysis_fatal_sorted)), phase_analysis_fatal_sorted['Fatal_Rate'] * 100)
ax2.set_yticks(range(len(phase_analysis_fatal_sorted)))
ax2.set_yticklabels(phase_analysis_fatal_sorted.index)
ax2.set_xlabel('Fatal Accident Rate (%)')
ax2.set_title('Fatal Accident Rate by Flight Phase')
ax2.grid(axis='x', alpha=0.3)

plt.tight_layout()
plt.show()
```



3. Regulatory Framework Analysis

```
# Analyze safety by FAR Description (regulatory framework)
far_analysis = df_accidents.groupby('FAR.Description').agg({
```

```
    'Event.Id': 'count',
    'Is_Fatal': 'sum',
    'Total.Fatal.Injuries': 'sum',
    'Total_Occupants': 'sum'
}).rename(columns={'Event.Id': 'Total_Accidents', 'Is_Fatal': 'Fatal_Accidents'})

far_analysis['Fatal_Rate'] = far_analysis['Fatal_Accidents'] / far_analysis['Total_Accidents']
far_analysis['Fatality_Rate_Per_Occupant'] = far_analysis['Total.Fatal.Injuries'] / far_analysis['Total_Occupants']
far_analysis = far_analysis.sort_values('Total_Accidents', ascending=False)

print("Safety Analysis by Regulatory Framework (FAR Description):")
display(far_analysis)
```

Safety Analysis by Regulatory Framework (FAR Description):

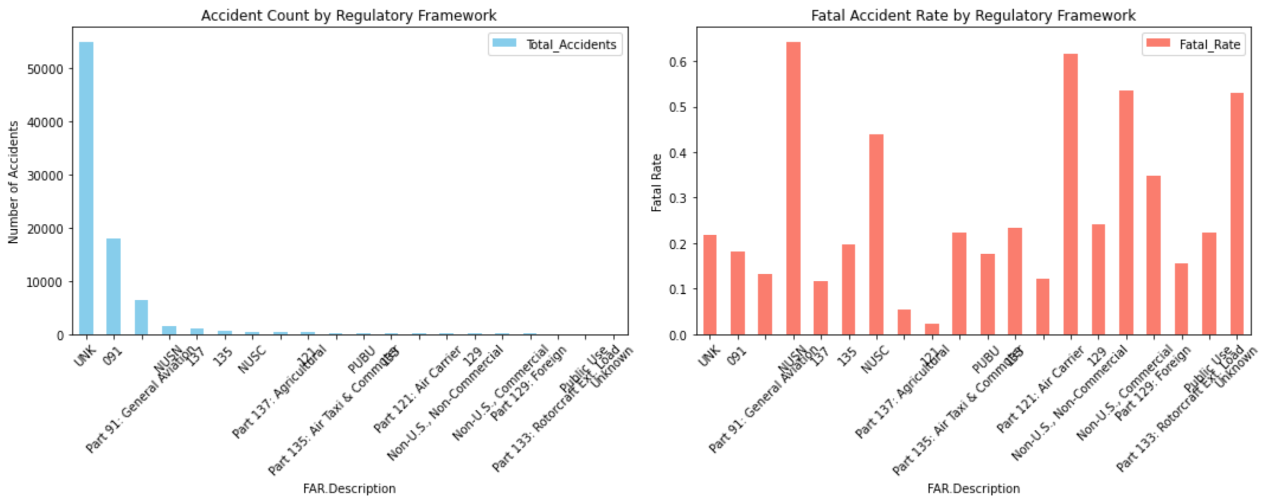
FAR.Description	Total_Accidents	Fatal_Accidents	Total.Fatal.Injuries	Total_Occupants	Fatal_Rate	Fatality_Rate_Per_Occu
UNK	54938	11930	33477.0	213419.0	0.217154	0.15
091	18043	3265	5449.0	32395.0	0.180957	0.16
Part 91: General Aviation	6403	845	2010.0	12530.0	0.131969	0.16
NUSN	1481	952	2197.0	3638.0	0.642809	0.60
137	1006	118	120.0	1021.0	0.117296	0.11
135	681	135	395.0	2662.0	0.198238	0.14
NUSC	488	214	4171.0	16543.0	0.438525	0.25
Part 137: Agricultural	437	24	28.0	461.0	0.054920	0.06
121	407	9	68.0	43745.0	0.022113	0.00
Part 135: Air Taxi & Commuter	264	59	164.0	1047.0	0.223485	0.15
PUBU	251	44	83.0	543.0	0.175299	0.15
133	107	25	34.0	151.0	0.233645	0.22
Part 121: Air Carrier	98	12	282.0	13339.0	0.122449	0.02
Non-U.S., Non-Commercial	94	58	202.0	324.0	0.617021	0.62
129	91	22	55.0	5535.0	0.241758	0.00
Non-U.S., Commercial	69	37	509.0	1351.0	0.536232	0.37
Part 129: Foreign	63	22	867.0	3523.0	0.349206	0.24
Part 133: Rotorcraft Ext. Load	32	5	6.0	42.0	0.156250	0.14
Public Use	18	4	4.0	39.0	0.222222	0.10
Unknown	17	9	49.0	99.0	0.529412	0.49
091K	8	0	0.0	25.0	0.000000	0.00
ARMF	4	2	4.0	10.0	0.500000	0.40
125	4	2	4.0	58.0	0.500000	0.06
Part 125: 20+ Pax,6000+ lbs	3	0	0.0	7.0	0.000000	0.00

```
# Visualize regulatory framework impact
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(15, 6))

# Filter for main categories and plot accident count
main_far = far_analysis[far_analysis['Total_Accidents'] > 10]
main_far.plot(y='Total_Accidents', kind='bar', ax=ax1, color='skyblue')
ax1.set_title('Accident Count by Regulatory Framework')
ax1.set_ylabel('Number of Accidents')
ax1.tick_params(axis='x', rotation=45)

# Fatal rate by regulatory framework
main_far.plot(y='Fatal_Rate', kind='bar', ax=ax2, color='salmon')
ax2.set_title('Fatal Accident Rate by Regulatory Framework')
ax2.set_ylabel('Fatal Rate')
ax2.tick_params(axis='x', rotation=45)
```

```
plt.tight_layout()
plt.show()
```



4. Engine Type and Configuration Analysis

```
# Analyze safety by engine type and number of engines
engine_analysis = df_accidents.groupby(['Engine.Type', 'Number.of.Engines']).agg({
    'Event.Id': 'count',
    'Is_Fatal': 'sum',
    'Total.Fatal.Injuries': 'sum'
}).rename(columns={'Event.Id': 'Total_Accidents', 'Is_Fatal': 'Fatal_Accidents'})

engine_analysis['Fatal_Rate'] = engine_analysis['Fatal_Accidents'] / engine_analysis['Total_Accidents']
engine_analysis = engine_analysis.sort_values('Total_Accidents', ascending=False)

print("Safety Analysis by Engine Type and Configuration:")
display(engine_analysis.head(10))
```

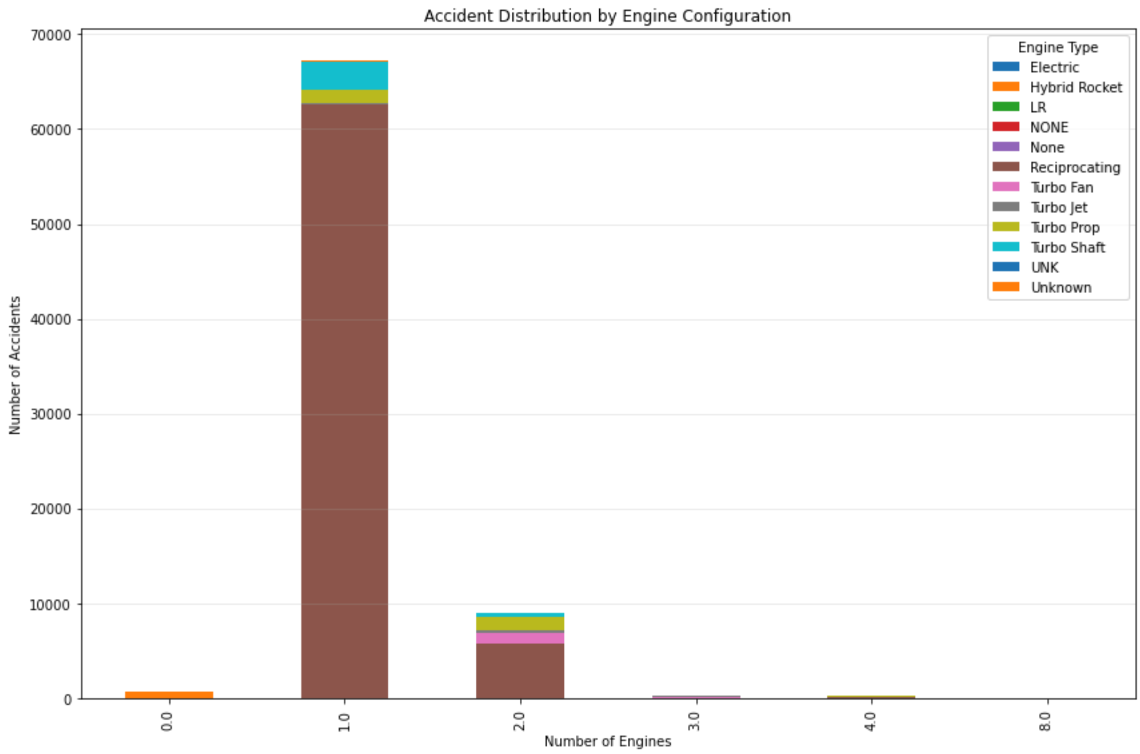
Safety Analysis by Engine Type and Configuration:

Engine.Type	Number.of.Engines	Total_Accidents	Fatal_Accidents	Total.Fatal.Injuries	Fatal_Rate
Reciprocating	**1.0**	62602	11139	19140.0	0.177934
	2.0	5748	1816	4112.0	0.315936
Turbo Shaft	**1.0**	2956	618	1252.0	0.209066
Turbo Prop	**1.0**	1453	325	609.0	0.223675
	2.0	1420	449	1765.0	0.316197
Turbo Fan	**2.0**	1064	136	2203.0	0.127820
Unknown	**0.0**	660	95	327.0	0.143939
Turbo Shaft	**2.0**	461	153	387.0	0.331887
Turbo Jet	**2.0**	304	80	398.0	0.263158
Turbo Fan	**3.0**	160	15	688.0	0.093750

```
# Visualize engine configuration safety
fig, ax = plt.subplots(figsize=(12, 8))

# Prepare data for visualization
engine_summary = df_accidents.groupby(['Number.of.Engines', 'Engine.Type']).size().unstack(fill_value=0)
engine_summary.plot(kind='bar', stacked=True, ax=ax)
ax.set_title('Accident Distribution by Engine Configuration')
ax.set_xlabel('Number of Engines')
ax.set_ylabel('Number of Accidents')
ax.legend(title='Engine Type')
ax.grid(axis='y', alpha=0.3)

plt.tight_layout()
plt.show()
```



5. Temporal Trends Analysis

```
# Analyze safety trends over time
temporal_analysis = df_accidents.groupby('Year').agg({
    'Event.Id': 'count',
    'Is_Fatal': 'sum',
    'Total.Fatal.Injuries': 'sum'
}).rename(columns={'Event.Id': 'Total_Accidents', 'Is_Fatal': 'Fatal_Accidents'})

temporal_analysis['Fatal_Rate'] = temporal_analysis['Fatal_Accidents'] / temporal_analysis['Total_Accidents']

# Calculate 3-year moving averages for smoothing
temporal_analysis['MA_Total_Accidents'] = temporal_analysis['Total_Accidents'].rolling(window=3).mean()
temporal_analysis['MA_Fatal_Rate'] = temporal_analysis['Fatal_Rate'].rolling(window=3).mean()

print("Temporal Safety Trends:")
display(temporal_analysis.tail(10))
```

Temporal Safety Trends:

	Total_Accidents	Fatal_Accidents	Total.Fatal.Injuries	Fatal_Rate	MA_Total_Accidents	MA_Fatal_Rate
Year						
2013	1462	342	822.0	0.233926	1637.000000	0.230092
2014	1451	357	1428.0	0.246037	1543.333333	0.235507
2015	1488	363	1101.0	0.243952	1467.000000	0.241305
2016	1542	335	820.0	0.217250	1493.666667	0.235746
2017	1514	336	640.0	0.221929	1514.666667	0.227710
2018	1562	356	1044.0	0.227913	1539.333333	0.222364
2019	1503	375	960.0	0.249501	1526.333333	0.233114
2020	1307	292	770.0	0.223412	1457.333333	0.233609
2021	1446	290	589.0	0.200553	1418.666667	0.224489
2022	1479	301	668.0	0.203516	1410.666667	0.209161

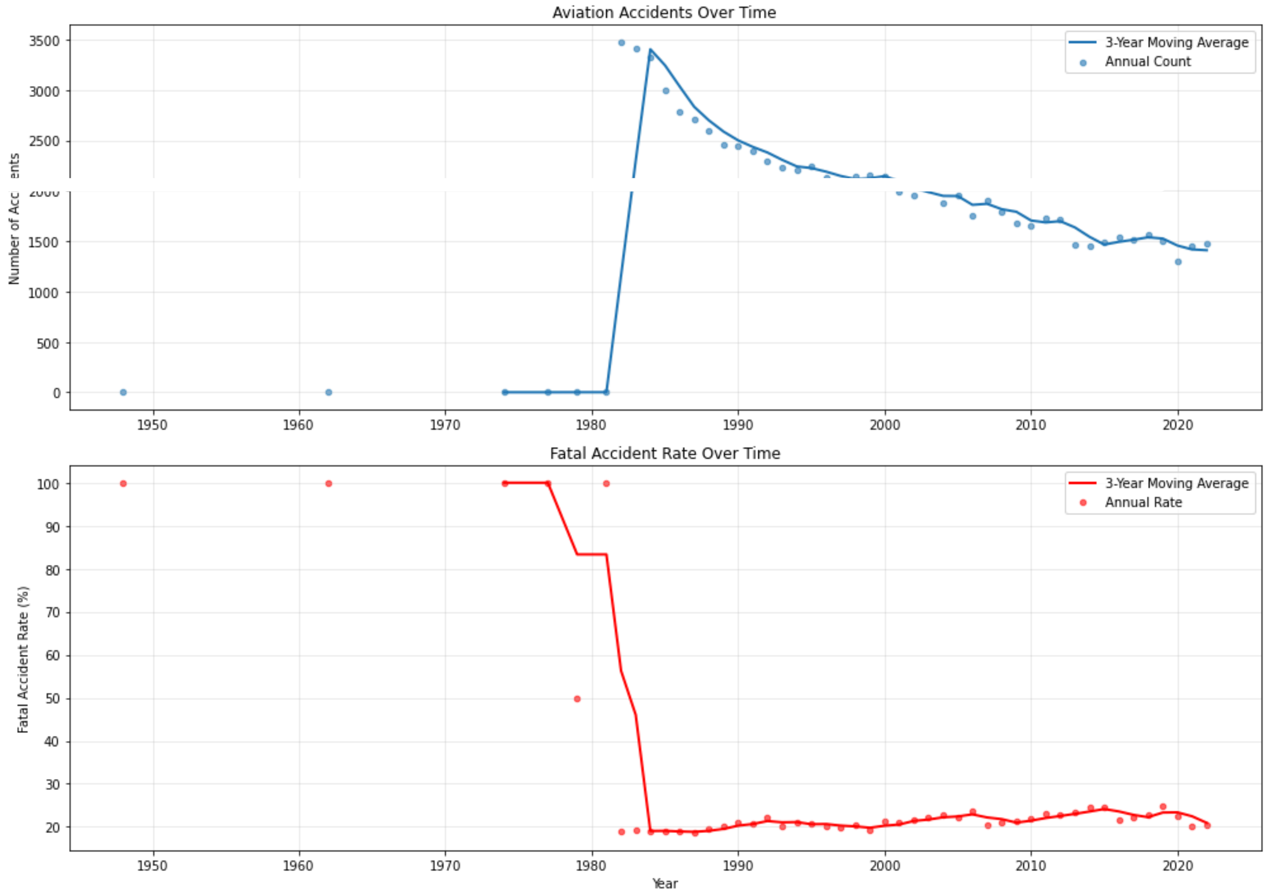
```
# Visualize temporal trends
fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(14, 10))

# Plot 1: Total accidents over time
ax1.plot(temporal_analysis.index, temporal_analysis['MA_Total_Accidents'], linewidth=2, label='3-Year Moving Average')
ax1.scatter(temporal_analysis.index, temporal_analysis['Total_Accidents'], alpha=0.6, s=20, label='Annual Count')
ax1.set_title('Aviation Accidents Over Time')
ax1.set_ylabel('Number of Accidents')
```

```
ax1.legend()
ax1.grid(True, alpha=0.3)

# Plot 2: Fatal rate over time
ax2.plot(temporal_analysis.index, temporal_analysis['MA_Fatal_Rate'] * 100, linewidth=2, color='red', label='3-Year Moving
ax2.scatter(temporal_analysis.index, temporal_analysis['Fatal_Rate'] * 100, alpha=0.6, s=20, color='red', label='Annual Ra
ax2.set_title('Fatal Accident Rate Over Time')
ax2.set_ylabel('Fatal Accident Rate (%)')
ax2.set_xlabel('Year')
ax2.legend()
ax2.grid(True, alpha=0.3)

plt.tight_layout()
plt.show()
```



Results and Business Recommendations

Based on our comprehensive analysis, here are our key findings and recommendations

```
# Summary statistics for key findings
print("KEY FINDINGS SUMMARY")
print("=" * 50)

total_accidents = len(df_accidents)
fatal_accidents = df_accidents['Is_Fatal'].sum()
overall_fatal_rate = (fatal_accidents / total_accidents) * 100

print(f"Total Accidents Analyzed: {total_accidents:,}")
print(f"Fatal Accidents: {fatal_accidents:,} ({overall_fatal_rate:.1f}%)")
print(f"Total Fatalities: {df_accidents['Total.Fatal.Injuries'].sum():,}")

# Key risk factors
print("\nTOP RISK FACTORS:")
print("- Weather: IMC conditions have {:.1f}x higher fatal rate than VMC".format(
    weather_analysis.loc['IMC', 'Fatal_Rate'] / weather_analysis.loc['VMC', 'Fatal_Rate']
))
print("- Flight Phase: Approach and landing phases account for {:.1f}% of all accidents".format(
    (phase_analysis.loc['Approach', 'Total_Accidents'] + phase_analysis.loc['Landing', 'Total_Accidents']) / total_acciden
))
print("- Regulatory Framework: Part 91 operations have {:.1f}x higher fatal rate than Part 121".format(
```

```
    far_analysis.loc['Part 91: General Aviation', 'Fatal_Rate'] / far_analysis.loc['Part 121: Air Carrier', 'Fatal_Rate']
))
```

```
KEY FINDINGS SUMMARY
==================================================
Total Accidents Analyzed: 85,015
Fatal Accidents: 17,794 (20.9%)
Total Fatalities: 50,179.0

TOP RISK FACTORS:
- Weather: IMC conditions have 3.7x higher fatal rate than VMC
- Flight Phase: Approach and landing phases account for 25.2% of all accidents
- Regulatory Framework: Part 91 operations have 1.1x higher fatal rate than Part 121
```

Business Recommendation 1: Aircraft Selection Strategy

```
# Identify recommended aircraft models
recommended_models = model_analysis_filtered[
    (model_analysis_filtered['Total_Accidents'] >= 10) &
    (model_analysis_filtered['Fatal_Rate'] < 0.1) &
    (model_analysis_filtered['Survival_Rate'] > 0.85)
].sort_values('Fatal_Rate')

print("RECOMMENDED AIRCRAFT MODELS:")
print("Models with strong safety records (low fatal rate, high survival rate):")
display(recommended_models.head(10))
```

```
RECOMMENDED AIRCRAFT MODELS:
Models with strong safety records (low fatal rate, high survival rate):
```

Make	Model	Total_Accidents	Fatal_Accidents	Total.Fatal.Injuries	Total_Occupants	Survival_Rate	Fatal_Rate	A
CUBCRAFTERS INC	**CC11-160**	16	0	0.0	24.0	1.0	0.0	
GRUMMAN ACFT ENG COR-SCHWEIZER	**G-164B**	24	0	0.0	24.0	1.0	0.0	
ROBINSON HELICOPTER	**R22**	21	0	0.0	36.0	1.0	0.0	
CESSNA	**305A**	16	0	0.0	25.0	1.0	0.0	
	195	24	0	0.0	46.0	1.0	0.0	
North American	**AT-6G**	11	0	0.0	18.0	1.0	0.0	
Maule	**MX-7-235**	13	0	0.0	19.0	1.0	0.0	

Recommendation: Focus on aircraft with proven safety records including turbine-powered aircraft and those with modern safety systems. Specifically consider models from manufacturers with strong safety cultures and comprehensive training programs.

Business Recommendation 2: Operational Risk Mitigation

```
# Calculate risk reduction from operational controls
vmc_fatal_rate = weather_analysis.loc['VMC', 'Fatal_Rate']
imc_fatal_rate = weather_analysis.loc['IMC', 'Fatal_Rate']
risk_reduction = ((imc_fatal_rate - vmc_fatal_rate) / imc_fatal_rate) * 100

print("OPERATIONAL RISK MITIGATION:")
print(f"Operating in VMC vs IMC reduces fatal accident risk by {risk_reduction:.1f}%")
print(f"VMC Fatal Rate: {vmc_fatal_rate*100:.1f}%")
print(f"IMC Fatal Rate: {imc_fatal_rate*100:.1f}%")
```

```
OPERATIONAL RISK MITIGATION:
Operating in VMC vs IMC reduces fatal accident risk by 73.1%
VMC Fatal Rate: 16.3%
IMC Fatal Rate: 60.5%
```

Recommendation: Implement strict operational controls including:

VMC-only operations during initial phase

Enhanced training for high-risk flight phases (approach and landing)

Comprehensive weather minimums and go-around policies

Business Recommendation 3: Regulatory Framework Strategy

```
# Compare regulatory frameworks
commercial_ops = ['Part 121: Air Carrier', 'Part 135: Air Taxi & Commuter']
general_aviation = 'Part 91: General Aviation'

commercial_fatal_rate = far_analysis.loc[commercial_ops, 'Fatal_Rate'].mean()
ga_fatal_rate = far_analysis.loc[general_aviation, 'Fatal_Rate']

safety_improvement = ((ga_fatal_rate - commercial_fatal_rate) / ga_fatal_rate) * 100

print("REGULATORY FRAMEWORK ANALYSIS:")
print(f"Commercial Operations (Part 121/135) Fatal Rate: {commercial_fatal_rate*100:.2f}%")
print(f"General Aviation (Part 91) Fatal Rate: {ga_fatal_rate*100:.2f}%")
```