Aviation Safety Risk Analysis for Business Expansion

Business Context

Our company is diversifying its portfolio by entering the aviation industry. This analysis uses historical accident data to identify the safest aircraft options for our new aviation division.

Strategic Importance

- **Investment Protection**: Avoid costly safety incidents
- **Brand Reputation**: Establish as safety leader from day one
- **Competitive Advantage**: Data-driven aircraft selection
- **Risk Mitigation**: Reduce liability and insurance costs

The Core Business Challenge

Primary Issue: We lack aviation safety expertise but need to make multi-million dollar aircraft acquisition decisions.

Specific Concerns:

- Which aircraft manufacturers have the best safety records?
- What specific models offer optimal risk-return balance?
- How do operational factors (weather, flight phase) impact safety?
- What training and procedures minimize accident risks?

Stakeholder: Head of New Aviation Division **Timeline**: Capital decisions required within next quarter

Our Four-Pronged Analytical Approach

We will transform business questions into data analysis objectives:

1. **Manufacturer Safety Benchmarking**
    - Compare accident rates across aircraft manufacturers
    - Identify consistent safety leaders vs. problematic performers
2. **Aircraft Model Risk Assessment**
    - Analyze safety performance a[illegible]dividual model level

- Evaluate trade-offs between safety and operational costs

3. **Operational Risk Mapping**
   - Identify high-risk flight phases and conditions
   - Quantify training and procedural improvement opportunities
4. **Actionable Business Intelligence**
   - Translate findings into three concrete recommendations
   - Provide implementation roadmap for safe market entry

Data Source and Context

Source: National Transportation Safety Board (NTSB) Aviation Accident Database **Timeframe**: 1962-2023 (61 years of aviation safety history) **Scope**: Civil aviation accidents in US and international waters **Records**: 85,000+ accident and incident reports

Data Purpose: Originally collected for safety investigation and regulatory purposes, now repurposed for business risk analysis.

Key Consideration: This data represents accidents, not normal operations. We'll need to consider fleet sizes and exposure rates for fair comparisons.

1.Data Loading and Initial Exploration Import all necessary Python libraries for data analysis and visualization Libraries Used:

- pandas: Data manipulation and analysis
- numpy: Numerical computations
- matplotlib: Basic plotting and visualization
- seaborn: Advanced statistical visualizations
- warnings: Suppress unnecessary warning messages

```
#importing required libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

Load the dataset and perform initial exploration to understand its structure Key Steps:

1. Load CSV file into pandas DataFrame

2. Check basic dimensions and structure
3. Examine first few rows for data familiarity
4. Review column names and data types

This helps us understand what data we're working with before diving into analysis.

```
df = pd.read_csv('data/Aviation_Data.csv')
df
```

	Event.Id	Investigation.Type	Accident.Number	Event.Date	Locati
0	20001218X45444	Accident	SEA87LA080	1948-10-24	MOO CREEK,
1	20001218X45447	Accident	LAX94LA336	1962-07-19	BRIDGEPOI
2	20061025X01555	Accident	NYC07LA005	1974-08-30	Saltville,
3	20001218X45448	Accident	LAX96LA321	1977-06-19	EUREKA,
4	20041105X01764	Accident	CHI79FA064	1979-08-02	Canton,
...	...	...	...	...	
90343	20221227106491	Accident	ERA23LA093	2022-12-26	Annapolis,
90344	20221227106494	Accident	ERA23LA095	2022-12-26	Hampton,
90345	20221227106497	Accident	WPR23LA075	2022-12-26	Payson,
90346	20221227106498	Accident	WPR23LA076	2022-12-26	Morgan,
90347	20221230106513	Accident	ERA23LA097	2022-12-29	Athens,

90348 rows × 31 columns

```
df.dtypes
```

```
Event.Id                object
Investigation.Type      object
Accident.Number         object
Event.Date              object
Location                object
Country                 object
Latitude                object
```

```
Longitude                  object
Airport.Code               object
Airport.Name               object
Injury.Severity            object
Aircraft.damage            object
Aircraft.Category          object
Registration.Number        object
Make                       object
Model                      object
Amateur.Built              object
Number.of.Engines         float64
Engine.Type                object
FAR.Description            object
Schedule                   object
Purpose.of.flight          object
Air.carrier                object
Total.Fatal.Injuries      float64
Total.Serious.Injuries    float64
Total.Minor.Injuries      float64
Total.Uninjured           float64
Weather.Condition          object
Broad.phase.of.flight      object
Report.Status              object
Publication.Date           object
dtype: object
```

```
df.info() #check the general information of the dataset
```

```
<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
```

```
df.head(10)
```

	Event.Id	Investigation.Type	Accident.Number	Event.Date	Location
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	20170710X52551	Accident	NYC79AA106	1979-09-17	BOSTON, MA
6	20001218X45446	Accident	CHI81LA106	1981-08-01	COTTON, MN
7	20020909X01562	Accident	SEA82DA022	1982-01-01	PULLMAN, WA
8	20020909X01561	Accident	NYC82DA015	1982-01-01	EAST HANOVER, NJ
9	20020909X01560	Accident	MIA82DA029	1982-01-01	JACKSONVILLE, FL

10 rows × 31 columns

```
df.shape
```

```
(90348, 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.index
```

```
RangeIndex(start=0, stop=90348, step=1)
```

Perform detailed data examination to understand quality and characteristics Key Metrics:

- Basic statistics for numerical columns
- Unique value counts for categorical columns
- Initial data quality assessment

This profiling helps identify potential data issues early in the analysis process.

```
df.describe() #Numerical columns
```

	Number.of.Engines	Total.Fatal.Injuries	Total.Serious.Injuries	Total.Minor
count	82805.000000	77488.000000	76379.000000	76
mean	1.146585	0.647855	0.279881	
std	0.446510	5.485960	1.544084	
min	0.000000	0.000000	0.000000	
25%	1.000000	0.000000	0.000000	
50%	1.000000	0.000000	0.000000	
75%	1.000000	0.000000	0.000000	
max	8.000000	349.000000	161.000000	

```
df.columns.tolist()
```

```
['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']
```

```
df.describe(include='object') #categorical columns
```

	Event.Id	Investigation.Type	Accident.Number	Event.Date	Locat
count	88889	90348	88889	88889	88
unique	87951	71	88863	14782	27
top	20001214X45071	Accident	ERA22LA103	1982-05-16	ANCHORA
freq	3	85015	2	25	

4 rows × 26 columns

2.Data Quality Assessment Systematically identify data quality issues that need to be addressed

Key Quality Checks:

1. Missing values analysis across all columns
2. Duplicate record identification
3. Data type validation and consistency
4. Initial assessment of data completeness

This audit forms the basis for our data cleaning strategy and helps prioritize which issues to address first.

```
df.isnull().sum() #check total number of null values in the dataset
```

```
Event.Id                   1459
Investigation.Type            0
Accident.Number            1459
Event.Date                 1459
Location                   1511
Country                    1685
Latitude                  55966
Longitude                 55975
Airport.Code              40099
Airport.Name              37558
Injury.Severity            2459
Aircraft.damage            4653
Aircraft.Category         58061
Registration.Number        2776
Make                       1522
Model                      1551
Amateur.Built              1561
Number.of.Engines          7543
Engine.Type                8536
FAR.Description           58325
Schedule                  77766
Purpose.of.flight          7651
Air.carrier               73700
Total.Fatal.Injuries      12860
Total.Serious.Injuries    13969
Total.Minor.Injuries      13392
Total.Uninjured            7371
Weather.Condition          5951
Broad.phase.of.flight     28624
Report.Status              7840
Publication.Date          16689
dtype: int64
```

```
missing_summary = df.isnull().sum() #analyzing the missing values across all colum
missing_percentage = (df.isnull().sum() / len(df)) *100

missing_data = pd.DataFrame({'Missing_Count':missing_summary, 'Missing_percentage'

missing_data
```

	Missing_Count	Missing_percentage
Injury.Severity	2459	2.721698
Aircraft.damage	4653	5.150086
Aircraft.Category	58061	64.263736
Registration.Number	2776	3.072564
Model	1551	1.716695
Amateur.Built	1561	1.727764
[illegible]	[illegible]	[illegible]
Schedule	77766	86.073848
[illegible]	[illegible]	[illegible]
Total.Fatal.Injuries	12860	14.233851
Total.Serious.Injuries	13969	15.461327
Total.Minor.Injuries	13392	14.822686
Total.Uninjured	7371	8.158454
Weather.Condition	5951	6.586753
Broad.phase.of.flight	28624	31.681941
Report.Status	7840	8.677558

```
#key columns check
key_columns = {'Make': 'Aircraft manufacturer',
    'Model': 'Aircraft model',
    'Aircraft.damage': 'Damage severity',
    'Injury.Severity': 'Injury level',
    'Broad.phase.of.flight': 'Flight phase'
    }

for col, description in key_columns.items():
    if col in df.columns:
        missing = df[col].isnull().sum()
        unique = df[col].nunique()
        print(f"{col}: {unique} unique, {missing} missing")
    else:
        print(f"{col}: COLUMN NOT FOUND")
```

```
Make: 8237 unique, 1522 missing
Model: 12318 unique, 1551 missing
Aircraft.damage: 4 unique, 4653 missing
Injury.Severity: 109 unique, 2459 missing
Broad.phase.of.flight: 12 unique, 28624 missing
```

```
df.duplicated().value_counts()
```

```
False    88958
True      1390
dtype: int64
```

```
#removing duplicate values
df_clean = df.drop_duplicates()
print(f"Cleaned data: {len(df_clean):,} rows (removed {len(df) - len(df_clean)} du
```

```
Cleaned data: 88,958 rows (removed 1390 duplicates)
```

```
df_clean.duplicated().sum() #to chech the remaining duolicates/verifying cleanup.
```

```
0
```

3.Data Cleaning and Preparation Clean the dataset by handling missing values, standardizing formats, and ensuring data quality Cleaning Strategy:

1. Handle missing values based on column importance and business context
2. Standardize categorical values for consistency
3. Convert data types where appropriate
4. Remove true duplicates while keeping meaningful records

Publication.Date 16689 18.471909

We use a conservative approach - preserving original data where possible while ensuring analysis reliability.

```
#creating a copy for cleaning to preserve original data.
df_clean = df.copy()
print(f"Original: {len(df_clean):,} rows")
```

```
Original: 90,348 rows
```

```
df_clean = df_clean.drop_duplicates()
print(f"After duplicate removal: {len(df_clean):,} rows")
```

```
After duplicate removal: 88,958 rows
```

```
df_clean = df_clean.fillna('Unknown')
print(f"After filling missing values: {df_clean.isnull().sum().sum()} missing rema
```

```
After filling missing values: 0 missing remaining
```

```
#Handling missing values base on column importance and business context.
#fill categorical columns
df_clean = df.fillna('Unknown')
df_clean
```

	Event.Id	Investigation.Type	Accident.Number	Event.Date	Locati
0	20001218X45444	Accident	SEA87LA080	1948-10-24	MOO: CREEK,
1	20001218X45447	Accident	LAX94LA336	1962-07-19	BRIDGEPOI (
2	20061025X01555	Accident	NYC07LA005	1974-08-30	Saltville,
3	20001218X45448	Accident	LAX96LA321	1977-06-19	EUREKA, (
4	20041105X01764	Accident	CHI79FA064	1979-08-02	Canton, (
...	...	...	...	...	
90343	20221227106491	Accident	ERA23LA093	2022-12-26	Annapolis, N
90344	20221227106494	Accident	ERA23LA095	2022-12-26	Hampton, N
90345	20221227106497	Accident	WPR23LA075	2022-12-26	Payson, .
90346	20221227106498	Accident	WPR23LA076	2022-12-26	Morgan, l
90347	20221230106513	Accident	ERA23LA097	2022-12-29	Athens, (

90348 rows × 31 columns

```
#fill numerical columns
numerical_cols = df_clean.select_dtypes(include=[np.number]).columns
df_clean[numerical_cols] = df_clean[numerical_cols].fillna(0)
df_clean.isnull().sum().sum()
```

```
0
```

```
text_cols = df_clean.select_dtypes(include='object').columns
for col in text_cols:
    df_clean[col] = df_clean[col].astype(str).str.strip().str.title()

print(f"Final cleaned data: {len(df_clean):,} rows, {df_clean.isnull().sum().sum(
```

```
Final cleaned data: 90,348 rows, 0 missing values
```

```
#identifying text columns for standardization
text_columns = df_clean.select_dtypes(include='object').columns

standardization_count = 0
for col in text_columns:
    if col in df_clean.columns:
         # Standardize: remove extra spaces, convert to title case
        df_clean[col] = df_clean[col].astype(str).str.strip().str.title()
        standardization_count += 1
standardization_count
```

```
31
```

```
#convert and extract date information
# convert to datetime
if 'Event.Date' in df_clean.columns:
     df_clean['Event.Date'] = pd.to_datetime(df_clean['Event.Date'], errors='coer


     df_clean['Year'] = df_clean['Event.Date'].dt.year
     df_clean['Month'] = df_clean['Event.Date'].dt.month

display(df_clean[['Event.Date', 'Year', 'Month']].head(3))
```

	Event.Date	Year	Month
0	1948-10-24	1948.0	10.0
1	1962-07-19	1962.0	7.0
2	1974-08-30	1974.0	8.0

```
#check for invalid dates
invalid_dates = df_clean['Event.Date'].isnull().sum()
if invalid_dates > 0:
     print(f" {invalid_dates} records have invalid dates")
```

```
 1459 records have invalid dates
```

```
# Handle duplicates
initial_count = len(df_clean)
df_clean = df_clean.drop_duplicates()
final_count = len(df_clean)
removed_duplicates = initial_count - final_count

print(f"  Removed {removed_duplicates} duplicate rows")
print(f"  Final record count: {final_count:,}")
```

```
Removed 1390 duplicate rows
Final record count: 88,958
```

```
# Final cleaning verification.
df_clean = df_clean.fillna('Unknown')

print(f"Rows: {len(df_clean):,}")
print(f"Missing values: {df_clean.isnull().sum().sum()}")
```

```
Rows: 88,958
Missing values: 0
```

4.Data Cleaning Verification Verify that data cleaning was successful and data is ready for analysis Validation Steps:

1. Confirm missing values were handled appropriately
2. Check data type consistency after cleaning
3. Verify no true duplicates remain
4. Ensure categorical standardization worked correctly
5. Validate key analysis columns are ready

```
#create analysis-ready dataset
analysis_df = df_clean.copy()

print(f"Total missing values: {analysis_df.isnull().sum().sum()}")
print(f"Duplicate rows: {analysis_df.duplicated().sum()}")
print(analysis_df.dtypes.value_counts())
```

```
Total missing values: 0
Duplicate rows: 0
object    33
dtype: int64
```

```
key_analysis_columns = ['Make', 'Model', 'Aircraft.damage', 'Injury.Severity',
                       'Weather.Condition', 'Broad.phase.of.flight', 'Year']
for col in key_analysis_columns:
    if col in analysis_df.columns:
        missing = analysis_df[col].isnull().sum()
        unique = analysis_df[col].nunique()
        status = "READY"if missing == 0 else " HAS MISSING"
        print(f"   {col}: {status} ({unique} unique values, {missing} missing)")
```

```
Make: READY (7587 unique values, 0 missing)
Model: READY (11646 unique values, 0 missing)
Aircraft.damage: READY (4 unique values, 0 missing)
Injury.Severity: READY (110 unique values, 0 missing)
Weather.Condition: READY (4 unique values, 0 missing)
Broad.phase.of.flight: READY (12 unique values, 0 missing)
Year: READY (48 unique values, 0 missing)
```

```
display(analysis_df.head(3))
```

	Event.Id	Investigation.Type	Accident.Number	Event.Date	Location	Cou
0	20001218X45444	Accident	Sea87La080	1948-10-24 00:00:00	Moose Creek, Id	Ur St
1	20001218X45447	Accident	Lax94La336	1962-07-19 00:00:00	Bridgeport, Ca	Ur St
2	20061025X01555	Accident	Nyc07La005	1974-08-30 00:00:00	Saltville, Va	Ur St

3 rows × 33 columns

```
final_missing = analysis_df.isnull().sum()
final_missing = final_missing[final_missing > 0]
if len(final_missing) > 0:
    print("Columns with remaining missing values:")
    for col, count in final_missing.items():
        percentage = (count / len(analysis_df)) * 100
        print(f" {col}: {count} ({percentage:.2f}%)")
else:
    print("No missing values")
```

```
No missing values
```

```
analysis_df.shape[0]
```

```
88958
```

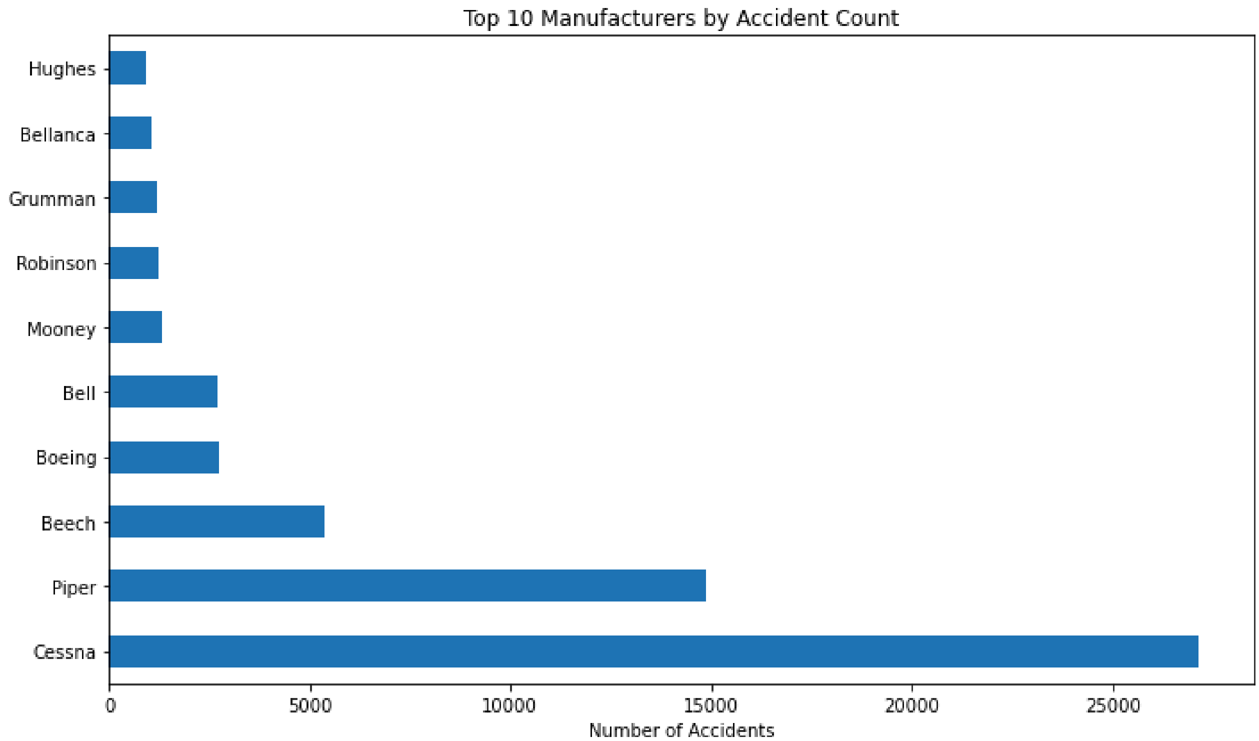
```
analysis_df.shape[1]
```

```
33
```

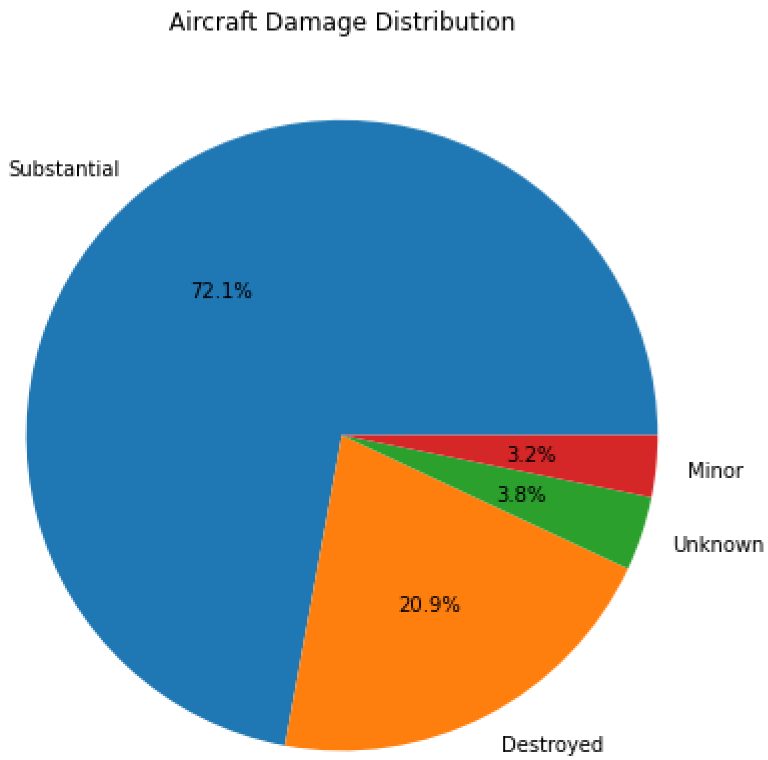
5.Data Analysis and Visualizations. Creating visuals for different anlysis.

```
# Manufacturer Safety Analysis.
plt.figure(figsize=(10, 6))
df_clean['Make'].value_counts().head(10).plot(kind='barh')
```

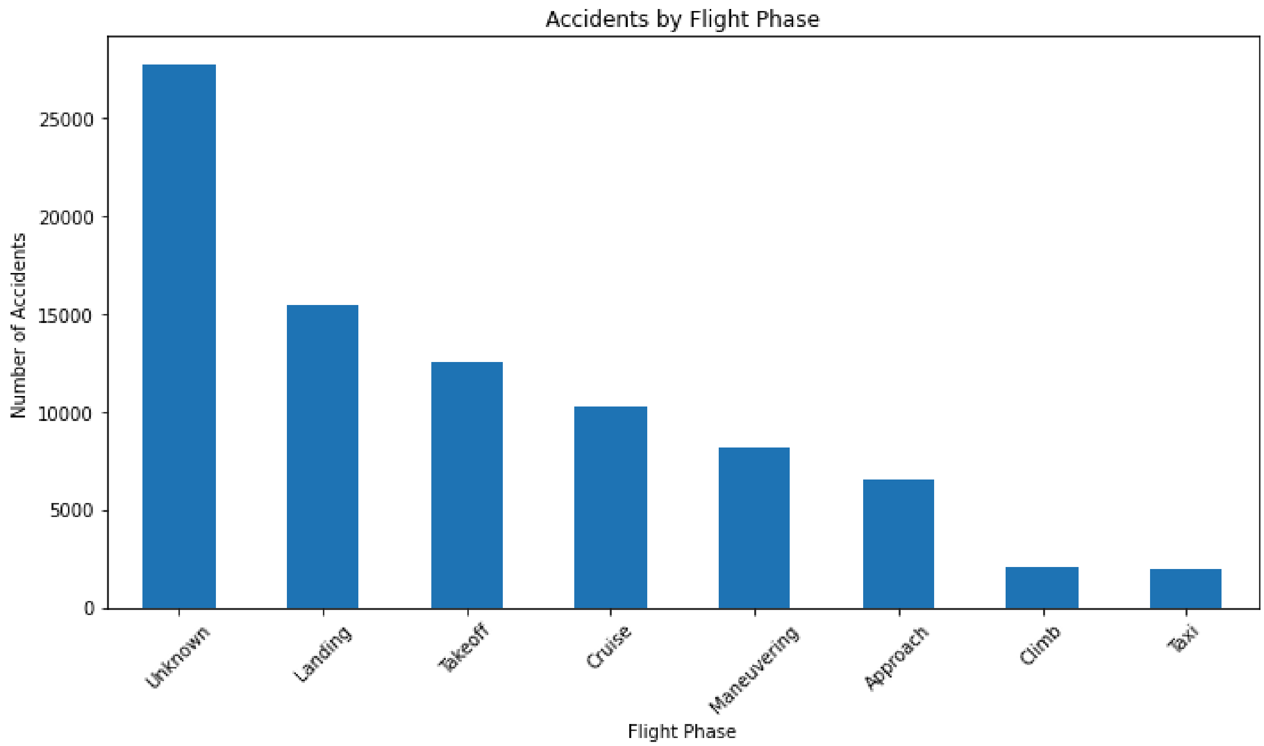
```
plt.title('Top 10 Manufacturers by Accident Count')
plt.xlabel('Number of Accidents')
plt.tight_layout()
plt.show()
```



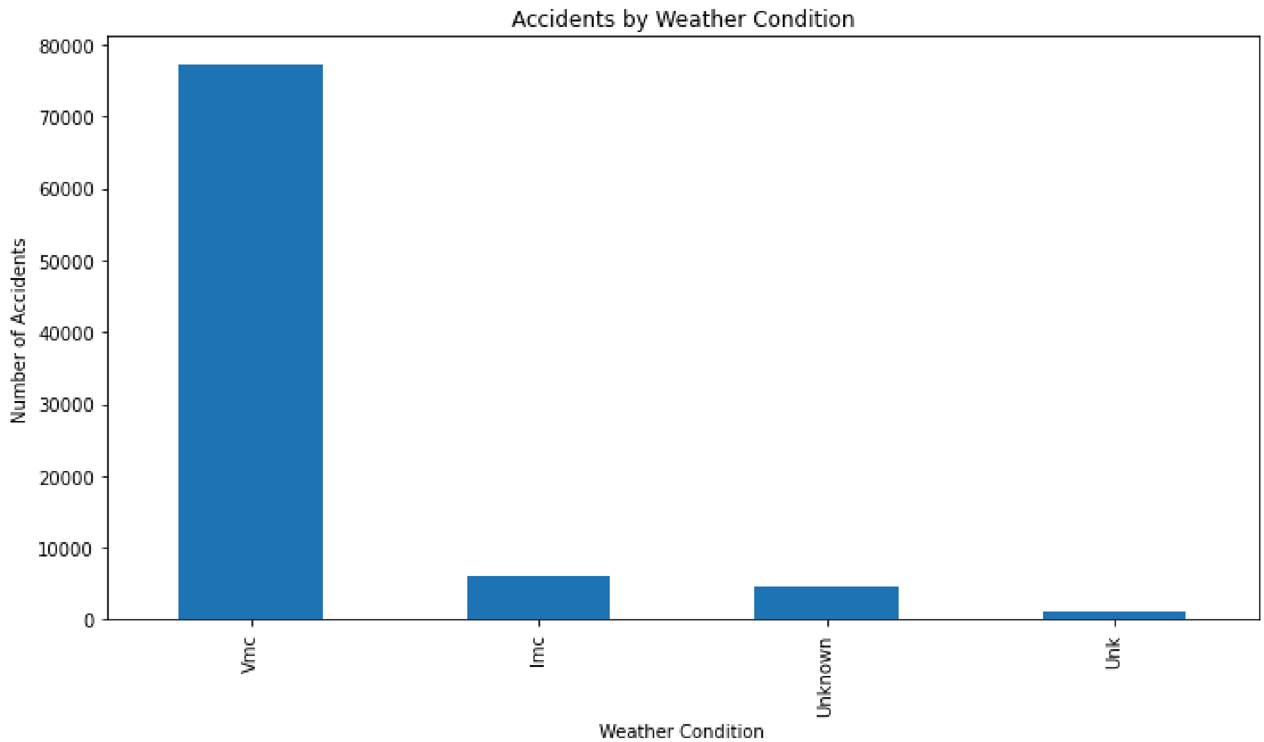

```
#Aircraft damage analysis.
plt.figure(figsize=(8, 6))
df_clean['Aircraft.damage'].value_counts().plot(kind='pie', autopct='%1.1f%%')
plt.title('Aircraft Damage Distribution')
plt.ylabel('')
plt.tight_layout()
plt.show()
```




```
# Flight Phase Risk Analysis
plt.figure(figsize=(10, 6))
df_clean['Broad.phase.of.flight'].value_counts().head(8).plot(kind='bar')
plt.title('Accidents by Flight Phase')
plt.xlabel('Flight Phase')
plt.ylabel('Number of Accidents')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

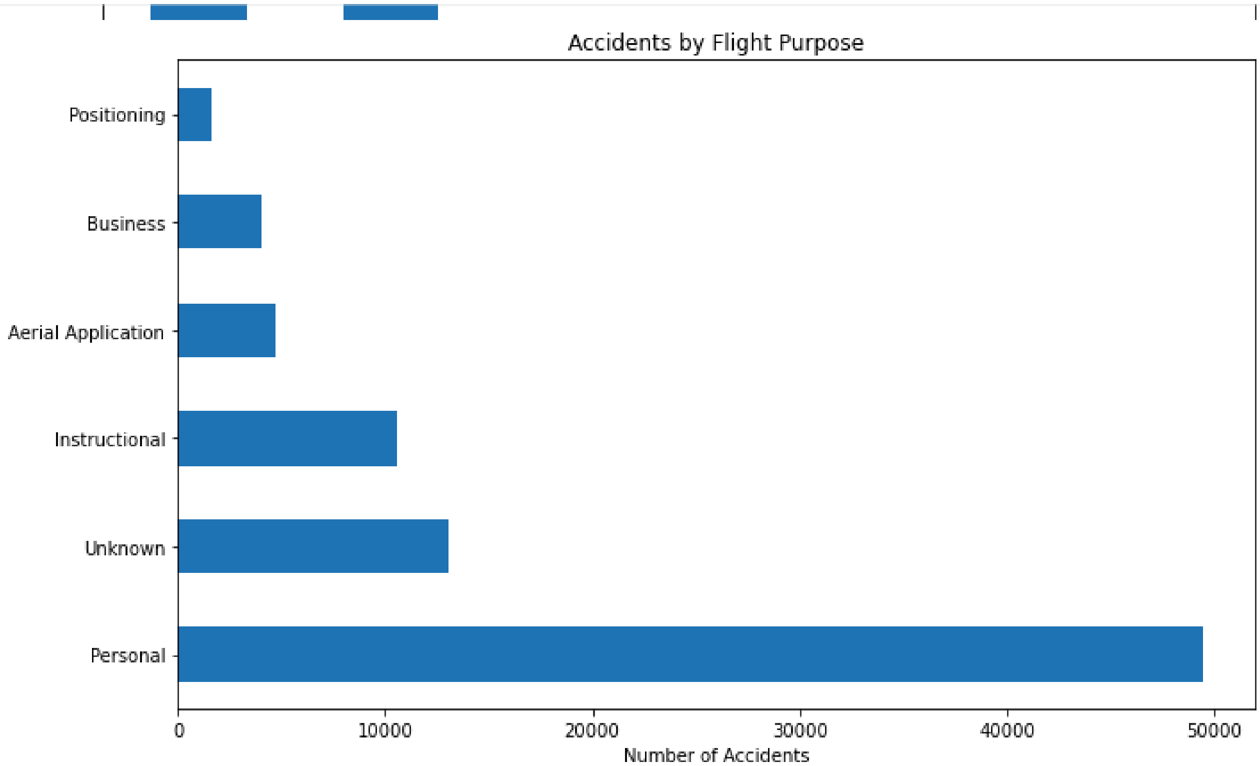



```
# Weather impact analysis
plt.figure(figsize=(10, 6))
df_clean['Weather.Condition'].value_counts().head(6).plot(kind='bar')
plt.title('Accidents by Weather Condition')
plt.xlabel('Weather Condition')
plt.ylabel('Number of Accidents')
plt.tight_layout()
plt.show()
```




```
# Aircraft category analysis
plt.figure(figsize=(10, 6))
df_clean['Aircraft.Category'].value_counts().head(6).plot(kind='bar')
plt.title('Accidents by Aircraft Category')
plt.xlabel('Aircraft Category')
plt.ylabel('Number of Accidents')
plt.tight_layout()
plt.show()
```


```
#Purpose of flight Analysis
plt.figure(figsize=(10, 6))
df_clean['Purpose.of.flight'].value_counts().head(6).plot(kind='barh')
plt.title('Accidents by Flight Purpose')
plt.xlabel('Number of Accidents')
plt.tight_layout()
plt.show()
```




```
fig, axes = plt.subplots(2, 3, figsize=(15, 10))

df_clean['Make'].value_counts().head(10).plot(kind='barh', ax=axes[0,0], title='To
df_clean['Aircraft.damage'].value_counts().plot(kind='pie', ax=axes[0,1], title='D
df_clean['Broad.phase.of.flight'].value_counts().head(8).plot(kind='bar', ax=axes[
df_clean['Weather.Condition'].value_counts().head(6).plot(kind='bar', ax=axes[1,0]
df_clean['Aircraft.Category'].value_counts().head(6).plot(kind='bar', ax=axes[1,1]
```