# 1.0 Project Title: Risk Assessment for the Operations Of Private and Commercial Aircrafts

### *Author:* Patrice Okoiti

## 1.1 Data Understanding

The selected Dataset [https://www.kaggle.com/datasets/khsamaha/aviation-accident-database-synopses (https://www.kaggle.com/datasets/khsamaha/aviation-accident-database-synopses)](https://www.kaggle.com/datasets/khsamaha/aviation-accident-database-synopses) for this analysis is from the National Transportation Safety Board, available on Kaggle, detailing the civil aviation accidents and selected incidents in the United States and international waters between 1962 and 2023. It details aircraft accidents, including information on accident, aircraft specifications, weather conditions, and injury severity which are relevant to our analysis.

### 1.1.1 column description

| Columns | Description |
|---:|---:|
| Event Id, Accident Number, Event Date, Location, Country, Latitude, Longitude, Airport Code, Airport Name | Unique identifiers for each accident and its location. |
| Make, Model, Aircraft Category, Amateur Built, Number of Engines, Engine Type | Details about the aircraft involved in the accident. |
| Injury Severity, Aircraft Damage, Weather Condition, Broad Phase of Flight | Risk factors contributing to the accident. |
| FAR Description, Schedule, Purpose of Flight, Air Carrier | Type of operations and flight purpose. |
| Total Fatal Injuries, Total Serious Injuries, Total Minor Injuries, Total Uninjured | Casualties per accident. |

## 1.2 Business Problem

Umoja Logistics is diversifying their portfolio by venturing into the aviation industry. The aim is to purchase and operate aircraft for commercial and private enterprises. However, aviation involves significant safety risks, including accidents and operational hazards. The goal of this project is to analyze historical aircraft accident data to identify low-risk aircraft models and key risk factors that could impact operations.

### 1.2.1 Objectives

1. Identify the safest type of aircraft
2. Identify risk factors contributing to aircraft accidents
3. Evaluate flight risks based on operations

In [3]:
```
# First step is to import the important libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

## 1.3 Data Mining

This involves reading and loading our data on to our notebook based on the file format

In [7]:
```
# Next step is data loading
df = pd.read_csv('Data/AviationData.csv', encoding='latin-1', low_memory=False)

# Display the fisrt 5 rows of the dataframe
df.head()
```

Out[7]:

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

5 rows × 31 columns

## 1.4 Data Preparation

This involved inspecting our dataset to identify the shape, name of columns, datatype of each column and any columns with missing values

In [30]:

```python
# Inspect the dataframe
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 88889 entries, 0 to 88888
Data columns (total 31 columns):
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   Event.Id                88889 non-null  object 
 1   Investigation.Type      88889 non-null  object 
 2   Accident.Number         88889 non-null  object 
 3   Event.Date              88889 non-null  object 
 4   Location                88837 non-null  object 
 5   Country                 88663 non-null  object 
 6   Latitude                34382 non-null  object 
 7   Longitude               34373 non-null  object 
 8   Airport.Code            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 [9]:
```
# Get summary statistics of the data
df.describe()
```

Out[9]:

| | Number.of.Engines | Total.Fatal.Injuries | Total.Serious.Injuries | Total.Minor.Injuries | Total.Uninjured |
|---|---|---|---|---|---|
| **count** | 82805.000000 | 77488.000000 | 76379.000000 | 76956.000000 | 82977.000000 |
| **mean** | 1.146585 | 0.647855 | 0.279881 | 0.357061 | 5.325440 |
| **std** | 0.446510 | 5.485960 | 1.544084 | 2.235625 | 27.913634 |
| **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** | 8.000000 | 349.000000 | 161.000000 | 380.000000 | 699.000000 |

From the above information, we can deduce that our dataframe has a shape of 88889 rows and 31 columns. The dataframe has 5 numerical columns and 26 categorical column. All the 5 numerical columns have float datatype. Of the 31 columns only the first 4 have complete entries, meaning 27 columns have missing values

The above information is a statistical summary of the numerical column of the dataframe

## 1.5 Data Preparation

This step involved validation of the dataset. This involved identifying and handling missing values and duplicates so as to get a clean dataset

In [10]:
```
# First Step is to create a copy of the original dataset
df_copy = df.copy() # Henceforth we will use the copy to clean our dataset

# Identify duplicates from the created copy of dataset
df_copy.duplicated().value_counts()
```

Out[10]:
```
False    88889
dtype: int64
```

From the above output we can deduce that our dataset does not contain any duplicates

In [11]:
```
# Next we display a breakdown of missing values in our dataset
df_copy.isna().sum()
```

Out[11]:
```
Event.Id                      0
Investigation.Type            0
Accident.Number               0
Event.Date                    0
Location                     52
Country                     226
Latitude                  54507
Longitude                 54516
Airport.Code              38640
Airport.Name              36099
Injury.Severity            1000
Aircraft.damage            3194
Aircraft.Category         56602
Registration.Number        1317
Make                         63
Model                        92
Amateur.Built               102
Number.of.Engines          6084
Engine.Type                7077
FAR.Description           56866
Schedule                  76307
Purpose.of.flight          6192
Air.carrier               72241
Total.Fatal.Injuries      11401
Total.Serious.Injuries    12510
Total.Minor.Injuries      11933
Total.Uninjured            5912
Weather.Condition          4492
Broad.phase.of.flight     27165
Report.Status              6381
Publication.Date          13771
dtype: int64
```

In [12]:
```
# Next we replace our missing values
for col in df_copy.columns:
    if str(df_copy[col].dtype) == 'object':
        df_copy[col].fillna('Unknown', inplace=True)
    else:
        df_copy[col].fillna(0, inplace=True)
df_copy.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 88889 entries, 0 to 88888
Data columns (total 31 columns):
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   Event.Id                88889 non-null  object 
 1   Investigation.Type      88889 non-null  object 
 2   Accident.Number         88889 non-null  object 
 3   Event.Date              88889 non-null  object 
 4   Location                88889 non-null  object 
 5   Country                 88889 non-null  object 
 6   Latitude                88889 non-null  object 
 7   Longitude               88889 non-null  object 
 8   Airport.Code            88889 non-null  object 
 9   Airport.Name            88889 non-null  object 
 10  Injury.Severity         88889 non-null  object 
 11  Aircraft.damage         88889 non-null  object 
 12  Aircraft.Category       88889 non-null  object 
 13  Registration.Number     88889 non-null  object 
 14  Make                    88889 non-null  object 
 15  Model                   88889 non-null  object 
 16  Amateur.Built           88889 non-null  object 
 17  Number.of.Engines       88889 non-null  float64
 18  Engine.Type             88889 non-null  object 
 19  FAR.Description         88889 non-null  object 
 20  Schedule                88889 non-null  object 
 21  Purpose.of.flight       88889 non-null  object 
 22  Air.carrier             88889 non-null  object 
 23  Total.Fatal.Injuries    88889 non-null  float64
 24  Total.Serious.Injuries  88889 non-null  float64
 25  Total.Minor.Injuries    88889 non-null  float64
 26  Total.Uninjured         88889 non-null  float64
 27  Weather.Condition       88889 non-null  object 
 28  Broad.phase.of.flight   88889 non-null  object 
 29  Report.Status           88889 non-null  object 
 30  Publication.Date        88889 non-null  object 
dtypes: float64(5), object(26)
memory usage: 21.0+ MB
```

The above output involved creating a 'for' loop that iterates through our dataset columns and fill the missing values with the set values based on the datatype. Replacing the missing values helps avoid unexpected arrors and completeness of the dataset for analysis.Filling of the missing values in categorical column with placeholder 'Unknown' and numerical column with zero allows me to maintain consistency of dataset without dropping valuable records, hence avoiding bias.

In [15]:
```
# Convert our date column to datetime
df_copy['Event.Date'] = pd.to_datetime(df_copy['Event.Date'])

# Create a year column on our dataset
df_copy['Year'] = df_copy['Event.Date'].dt.year.astype(int)

# Filter for the 21st century only
df_copy = df_copy[df_copy['Year'] >= 2000]
```

The above code filters out the discontinued and outdated aircrafts and leaves us with only aircrafts active in the 21st century

In [16]:
```
# Identify the unique operational purpose of each flight
df_copy['Purpose.of.flight'].unique()
```

Out[16]:
```
array(['Positioning', 'Personal', 'Instructional', 'Unknown',
       'Aerial Observation', 'Ferry', 'Public Aircraft', 'Business',
       'Aerial Application', 'Executive/corporate', 'Other Work Use',
       'Flight Test', 'Skydiving', 'Air Race/show', 'Air Drop',
       'Public Aircraft - Federal', 'Glider Tow',
       'Public Aircraft - Local', 'External Load',
       'Public Aircraft - State', 'Banner Tow', 'Firefighting',
       'Air Race show', 'PUBS', 'ASHO', 'PUBL'], dtype=object)
```

In [17]:
```
# Create a list of private and commercial purposes
private = ["Personal", "Executive/corporate", "Business", "Ferry"]
commercial = ["Aerial Application", "Aerial Observation", "Air Carrier", "Public Aircraft"]

# Create a column that describes operations category
flight_purpose = []

for purpose in df_copy['Purpose.of.flight']:
    if purpose in private:
        flight_purpose.append('Private')
    elif purpose in commercial:
        flight_purpose.append('Commercial')
    else:
        flight_purpose.append('Other')

df_copy['Category of Operation'] = flight_purpose

# Filter our dataset to include only private and commercial used planes
df_clean = df_copy.loc[(df_copy['Category of Operation'] == 'Private') |
                       (df_copy['Category of Operation'] == 'Commercial')]
df_clean = df_clean.copy() # Handles error of working with sliced dataframe rather than a dataframe
df_clean.head()
```

Out[17]:

| | Event.Id | Investigation.Type | Accident.Number | Event.Date | Location | Country | Latitude | Longitude | Airpo |
|---|---|---|---|---|---|---|---|---|---|
| **47676** | 20001212X20327 | Accident | ATL00FA019 | 2000-01-01 | MONTEAGLE, TN | United States | Unknown | Unknown | U |
| **47677** | 20001212X20383 | Accident | LAX00LA063 | 2000-01-02 | VICTORVILLE, CA | United States | Unknown | Unknown | U |
| **47679** | 20001212X20364 | Accident | FTW00LA067 | 2000-01-02 | CORNING, AR | United States | Unknown | Unknown | U |
| **47680** | 20001212X20358 | Accident | FTW00LA057 | 2000-01-02 | ODESSA, TX | United States | Unknown | Unknown | U |
| **47681** | 20001212X20344 | Accident | DEN00FA037 | 2000-01-02 | TELLURIDE, CO | United States | Unknown | Unknown | U |

5 rows × 33 columns

The above code filters out all aircrafts that are were not utilized for private and commercial flight use

In [18]:
```
# Format the text of categorical columns for consistency
df_clean['Weather.Condition'] = df_clean['Weather.Condition'].str.title()

# Edit the initials of weather to the full names
df_clean['Weather.Condition'] = df_clean['Weather.Condition'].apply(
    lambda x: 'Visual Meteorological Conditions'
    if x == 'Vmc'
    else 'Instrument Meteorological Conditions'
    if x == 'Imc'
    else "Unknown"
    if x == 'Unk'
    else x)

# Format the text of categorical columns for consistency
df_clean['Make'] = df_clean['Make'].str.title()
df_clean['Model'] = df_clean['Model'].str.upper()

# Create a new column that combines make and model
df_clean['Make_and_Model'] = df_clean['Make'] + ' ' + df_clean['Model']
```

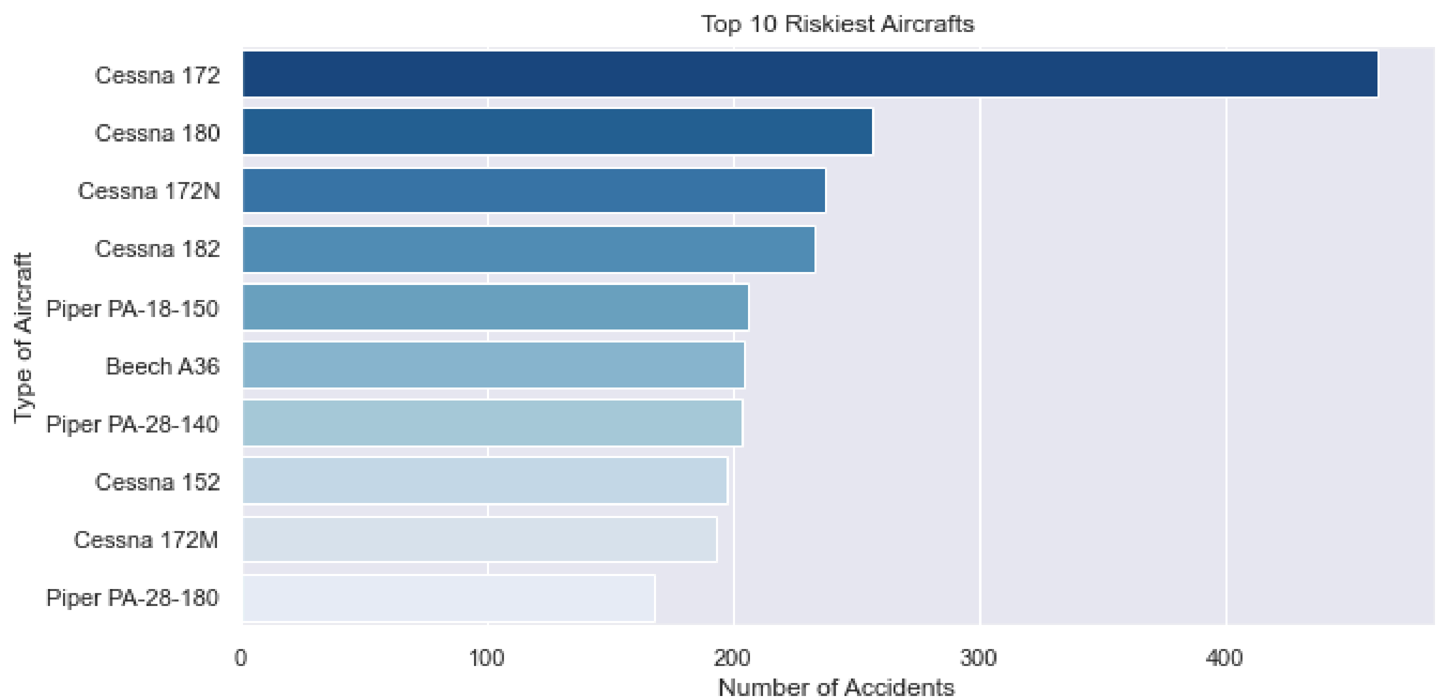
The above code edits the texts of the relevant columns to ensure consistency since python is case sensitive

## 1.6 Data Evaluation

### 1.6.1 Objective 1

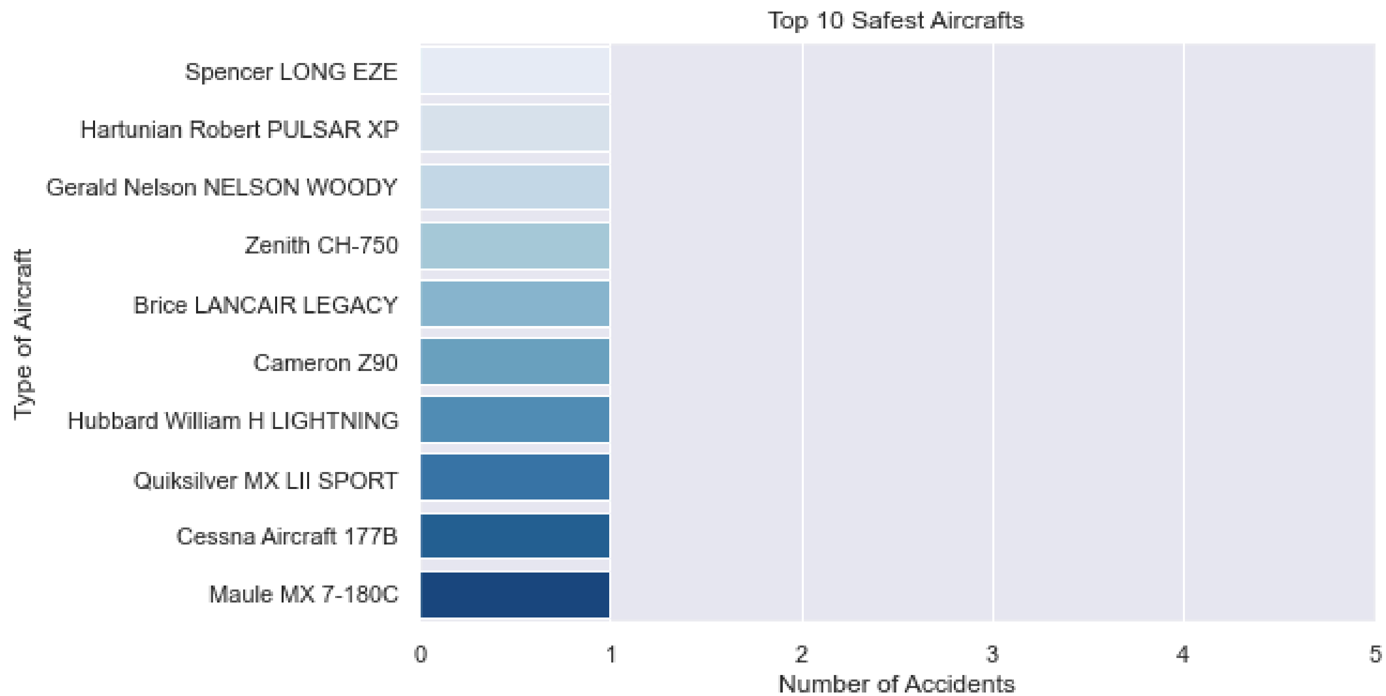
The first objective is to identify safest aircraft. This involves analyzing the number of accidents based on aircraft 'Make' and 'Model' to determine the aircraft with the lowest risk of accidents

In [22]:
```
# Identify the risky aircrafts based on highest number of accidents
riskiest_aircraft = df_clean['Make_and_Model'].value_counts().head(10)


# Create a horizontal bar graph visualisation
fig, ax = plt.subplots(figsize=(10,5))
sns.set_theme(style='darkgrid')
sns.barplot(y=riskiest_aircraft.index, x=riskiest_aircraft.values, palette='Blues_r')
ax.set(title='Top 10 Riskiest Aircrafts',
       xlabel='Number of Accidents',
       ylabel= 'Type of Aircraft')

#save the visualization and ensure full image displayed
plt.savefig('Images/Risky-aircrafts.jpg', dpi=300, bbox_inches='tight')
```




The plot shows the the top 10 riskiest aircrafts based on the most accidents between 2000 and 2023.

In [23]:
```
# Identify safest aircrafts based on the lowest number of accidents
safest_aircraft= df_clean['Make_and_Model'].value_counts().tail(10)

# Visualize the safest aicrafts
fig, ax = plt.subplots(figsize=(8,5))
sns.set_theme(style='darkgrid')
sns.barplot(y=safest_aircraft.index, x=safest_aircraft.values, palette='Blues')
ax.set(title='Top 10 Safest Aircrafts',
       xlabel='Number of Accidents',
       ylabel= 'Type of Aircraft',
       xlim=(0, 5))

#save the image and ensure full image displayed
fig.savefig('Images/Safest-aircrafts.jpg', dpi=300, bbox_inches='tight')
```



The above output shows the safest aircraft based on the number of accidents encountered between 2000 and 2023. Note that the aircrafts will keep on changing since their is a large number of aircrafts with one accident only

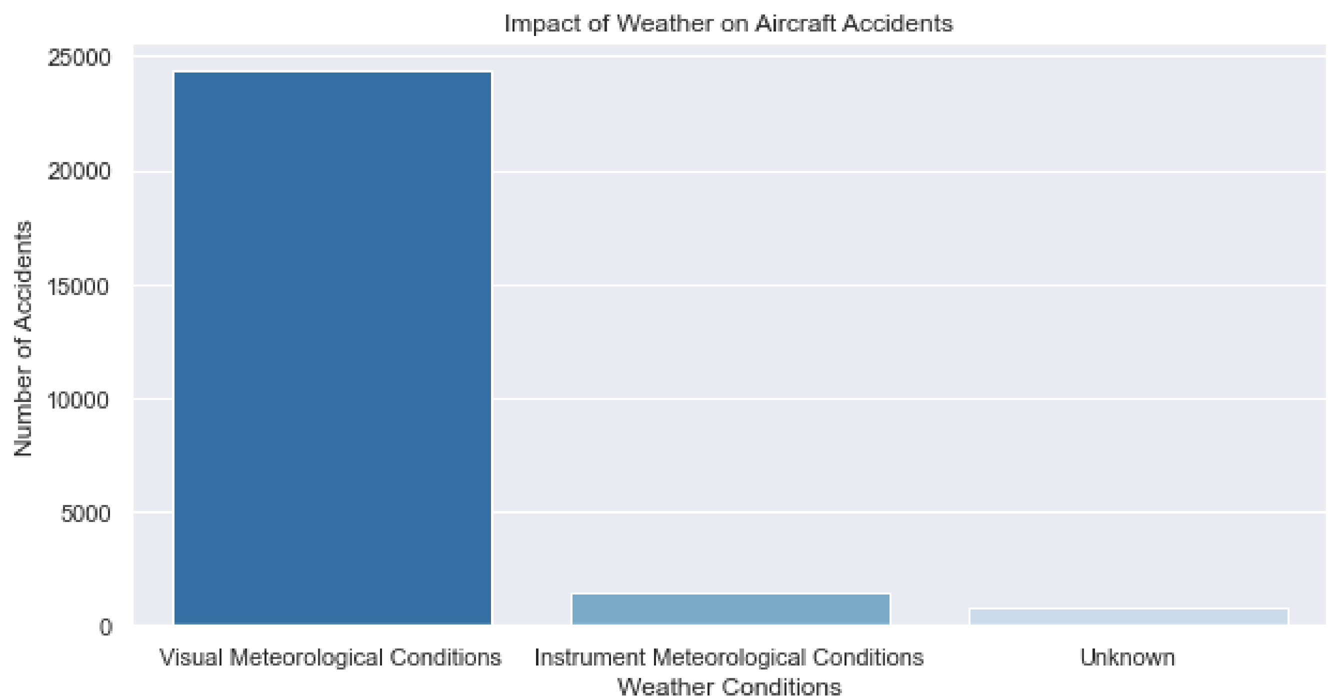
### 1.6.2 Objective 2

The second objective involves identifying the risk factors contributing to accidents by analyzing the Weather conditions and the broad phase of the aircraft, for example landing or taking off, during the accident

In [24]:
```
# Number of accidents per aircraft per year
yearly_accidents = df_clean.groupby('Year')['Make_and_Model'].count()
yearly_accidents

# visualize number of accidents over the years for the past 23years
fig, ax = plt.subplots(figsize=(10,5))
sns.set_theme(style='darkgrid')
sns.lineplot(x=yearly_accidents.index, y=yearly_accidents.values, marker='o')
ax.set(title="Aircraft Accident Trend Overtime",
       xlabel='Years',
       ylabel='Number of Accidents',
       ylim=(500,2000))

# save the image and ensure image fuly displayed
plt.savefig('Images/Accident-trend.jpg', dpi=300, bbox_inches='tight')
```

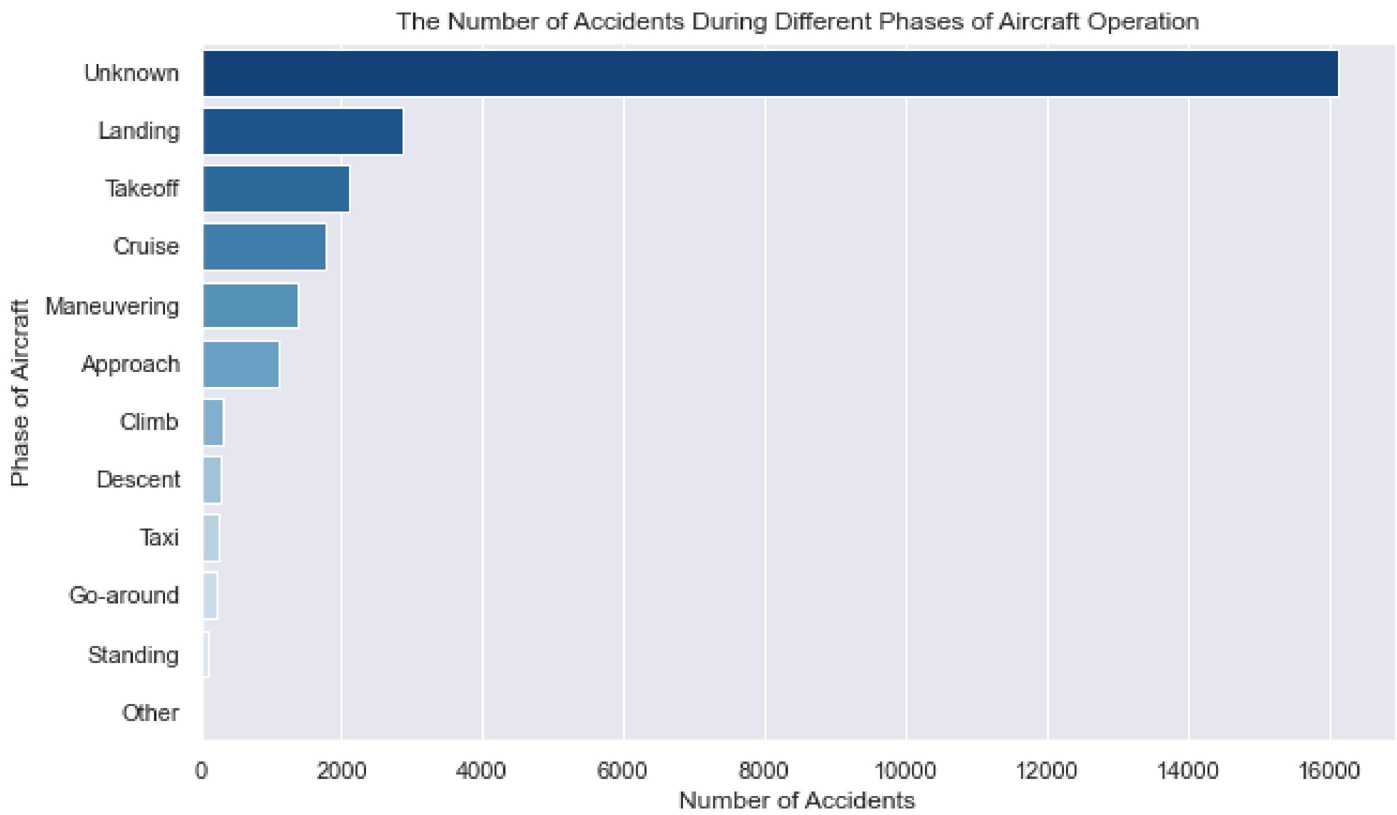
From the above output it can be seen that the overall accidents risk associated with the aircrafts is at a steady decline over the years

In [25]:
```
# Weather as risk factor in aircraft accidents
weather_risk = df_clean['Weather.Condition'].value_counts()

# Visualize weather as a risk factor
fig, ax = plt.subplots(figsize=(10,5))
sns.set_theme(style='darkgrid')
sns.barplot(y=weather_risk.values, x=weather_risk.index, palette='Blues_r')
ax.set(title='Impact of Weather on Aircraft Accidents',
       ylabel= 'Number of Accidents',
       xlabel = 'Weather Conditions')

#save the image and ensure image fully dispalyed
plt.savefig('Images/Weather-impact.jpg', dpi=300, bbox_inches='tight')
```



From above output it can be seen that most accidents happened during Visual Meteorological Conditions, meaning that they happened when the weather conditions allowed the pilots to fly with visual references to the ground and other aircrafts without solely relying on instruments. This rules out weather conditions as primary cause of aircraft accidents

In [26]:
```
# Identify the phase the aircraft when most accident occur
aircraft_phase = df_clean['Broad.phase.of.flight'].value_counts()

#Visualize every phase by accidents
fig, ax = plt.subplots(figsize=(10,6))
sns.set_theme(style='darkgrid')
sns.barplot(y=aircraft_phase.index, x=aircraft_phase.values, palette='Blues_r')
ax.set(title='The Number of Accidents During Different Phases of Aircraft Operation',
       ylabel= 'Phase of Aircraft',
       xlabel = 'Number of Accidents')

#save the image and ensure image fully displayed
plt.savefig('Images/Flight-phase.jpg', dpi=300, bbox_inches='tight')
```

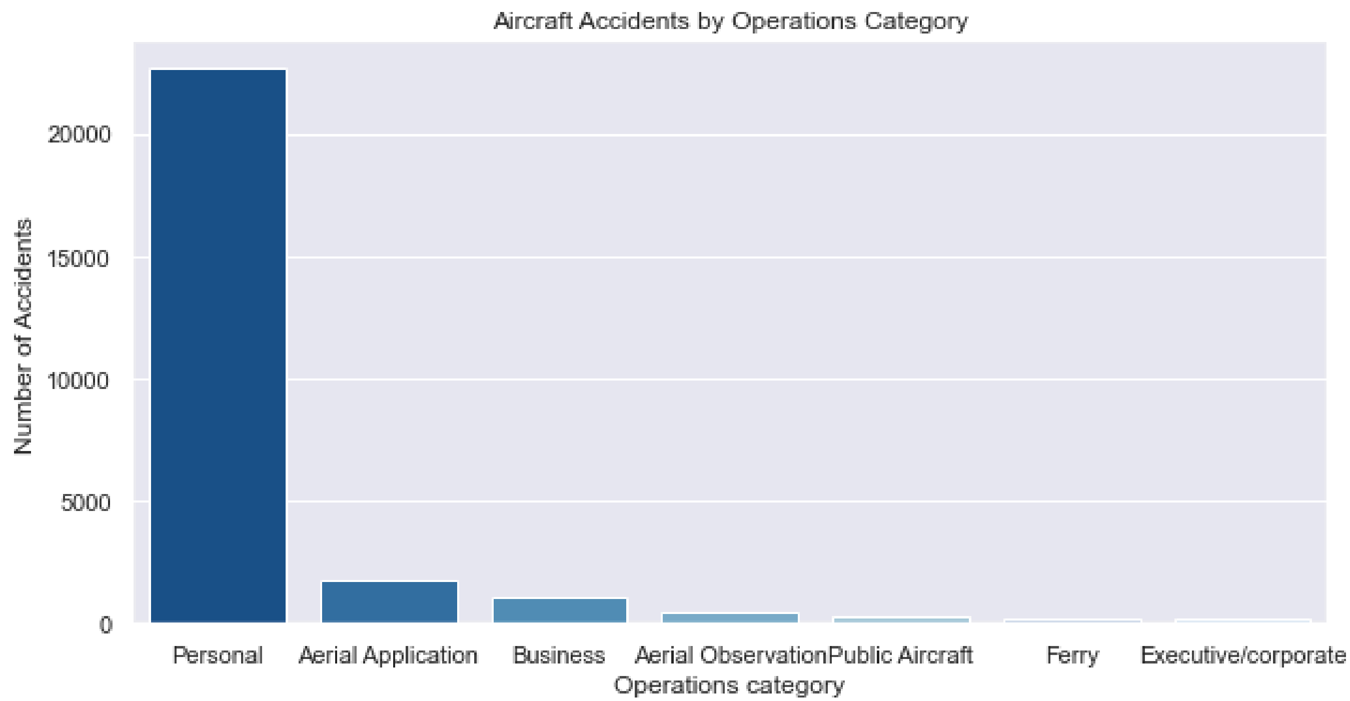


From the above output it can be clearly seen that phase of accidents for most of accidents remain clearly unknown. But We can also clearly deduce that a significant number of accidents happened during landing, taking off and during cruising, hence we should view these 3 phases as major risk factors associated with aircraft accidents.

### 1.6.3 Objective 3

The final objective is to evaluate operational risk factors of the aircraft. This involves analysing the number of accidents as per operations category of aircraft, that is, private and commercial.

In [27]:
```
# Evaluate operational risk
operational_category = df_clean['Purpose.of.flight'].value_counts()

# Visualize opeartional risk
fig, ax = plt.subplots(figsize=(10,5))
sns.set_theme(style='darkgrid')
sns.barplot(x=operational_category.index, y=operational_category.values, palette='Blues_r')
ax.set(title='Aircraft Accidents by Operations Category',
       xlabel='Operations category',
       ylabel='Number of Accidents')

#save the image and ensure image fully displayed
plt.savefig('Images/Operations-risk.jpg', dpi=300, bbox_inches='tight')
```




The above output clearly shows that venturing into private flights is riskier compared to commercial fight due to the number of accidents by private flights

In [28]:
```
# Next we save our cleaned data which will be useful during creation Dashboard
df_clean.to_csv('Data/CleanAviationData.csv')
```

## 1.7 Conclusion

From the analysis of the Aviation Data it be can concluded that:

- The aircrafts with the high number of accidents may be due to high levels of usage
- Adverse weather conditions is a significant risk factor in aircraft accidents but it has not been the primary risk factor in the 21st Century.
- The phase the aircraft is in when accidents occur remains majorly unknown, but a significant number of accidents often occur during landing, taking off and cruising making them significant risk factors.
- Prioritizing operations of commercial flights is more viable as compared to private flights due to the high number of accidents encountered by private flights