Phase-1 Project

Business Overview/ Introduction

For this project, I will use data cleaning, imputation, analysis, and visualization to generate insights for a business stakeholder.The project aims to analyze aviation accident data to determine the lowest-risk aircraft models for commercial and private enterprises. The analysis will provide actionable insights for the company's aviation division to guide their aircraft purchasing decisions.

Business Problem

Your company is expanding in to new industries to diversify its portfolio. Specifically, they are interested in purchasing and operating airplanes for commercial and private enterprises, but do not know anything about the potential risks of aircraft.

My Task

determine which aircraft are the lowest risk for the company to start this new business endeavor

Data Understanding

In this project, I'll work with aviation accident data from the National Transportation Safety Board (NTSB) Dataset, which can be found on kaggle. Specifically I will be using **Avition_Data.csv** file for my analysis.

The data is contained in two separate CSV files:

1. `Avition_Data.csv` :the file contains information from 1962 and later about civil aviation accidents and selected incidents within the United States, its territories and possessions, and in international waters.
2. `USState_Codes.csv` :This file contains the US State name and the abbreviation of them

Business Understanding

Stakeholder

The primary stakeholder is the head of the company's new aviation division, responsible for making data-driven decisions on aircraft acquisition.

Key Business Questions

- Which aircraft models have the lowest accident rates?
- What are the common causes of aviation accidents?
- How do different aircraft manufacturers compare in terms of safety?
- What factors contribute most to aviation risk, and how can they be mitigated?

1. Load Data using pandas

In the cell below, I:

- Import and alias `pandas` as `pd`
- Import and alias `numpy` as `np`
- Import and alias `seaborn` as `sns`
- Import and alias `matplotlib.pyplot` as `plt`
- Set Matplotlib visualizations to display inline in the notebook

In [38]:
```
# import the libraries using alias
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

%matplotlib inline
```

Import the function from load.py

In the cell below I import the function `load_data()` from load.py

In [39]:
```
# Import the function
import load
```

Aviation_Data

In the cell below, I load `Aviation_Data.csv` as `df` using the script `load.py`

In [40]:
```
# load the data the function output .head(), .info(), .describe() of data
#df =load.load_data('data/Aviation_Data.csv')
```

The above cell output data as expected but the dataset is large to output all details needed .head(), .info(), .describe()

In cell below I will load data using pd as alias, then use method .head(), .info(), .describe() in separate cells

In [41]:
```
# load data using alias pd and view first 5 reconds of data

df = pd.read_csv('data\Aviation_Data.csv', low_memory=False)
df.head(10)
```

Out[41]:

	Event.Id	Investigation.Type	Accident.Number	Event.Date	Location	Coun
0	20001218X45444	Accident	SEA87LA080	1948-10-24	MOOSE CREEK, ID	Uni Sta
1	20001218X45447	Accident	LAX94LA336	1962-07-19	BRIDGEPORT, CA	Uni Sta
2	20061025X01555	Accident	NYC07LA005	1974-08-30	Saltville, VA	Uni Sta
3	20001218X45448	Accident	LAX96LA321	1977-06-19	EUREKA, CA	Uni Sta
4	20041105X01764	Accident	CHI79FA064	1979-08-02	Canton, OH	Uni Sta
5	20170710X52551	Accident	NYC79AA106	1979-09-17	BOSTON, MA	Uni Sta
6	20001218X45446	Accident	CHI81LA106	1981-08-01	COTTON, MN	Uni Sta
7	20020909X01562	Accident	SEA82DA022	1982-01-01	PULLMAN, WA	Uni Sta
8	20020909X01561	Accident	NYC82DA015	1982-01-01	EAST HANOVER, NJ	Uni Sta
9	20020909X01560	Accident	MIA82DA029	1982-01-01	JACKSONVILLE, FL	Uni Sta

10 rows × 31 columns

Get familiar with the data. Steps includes:

- Understanding the dimensionality of the dataset
- Investigating what type of data it contains, and the data types used to store it
- Discovering how missing values are encoded, and how many there are
- Getting a feel for what information it does and doesn't contain

In [42]:
```
# use .info() method to perfom metadata summary of df

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            50132 non-null  object 
 9   Airport.Name            52704 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     87507 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             81793 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           82505 non-null  object 
 30  Publication.Date        73659 non-null  object 
dtypes: float64(5), object(26)
memory usage: 21.4+ MB
```

The cell below, I interpret the metadata information above

In [43]:
```
"""
The type of df is Dataframe.
Data types: the dataset as 31 columns with 26 columns with object(26) data types
and 5 0nly as float(5).
The entire dataset columns as missing values except column 2 Investigation.Type

"""
```

Out[43]:
```
'\nThe type of df is Dataframe.\nData types: the dataset as 31 columns with 26 colum
ns with object(26) data types\nand 5 0nly as float(5).\nThe entire dataset columns a
s missing values except column 2 Investigation.Type\n\n'
```

In [44]:
```
# use the .describe() to get statistics summary of df data
```

```
df.describe()
```

Out[44]:

	Number.of.Engines	Total.Fatal.Injuries	Total.Serious.Injuries	Total.Minor.Injuries	T
count	82805.000000	77488.000000	76379.000000	76956.000000	
mean	1.146585	0.647855	0.279881	0.357061	
std	0.446510	5.485960	1.544084	2.235625	
min	0.000000	0.000000	0.000000	0.000000	
25%	1.000000	0.000000	0.000000	0.000000	
50%	1.000000	0.000000	0.000000	0.000000	
75%	1.000000	0.000000	0.000000	0.000000	
max	8.000000	349.000000	161.000000	380.000000	

The cell bellow interpret the statistical summary above

In [45]:
```
"""
df total of 5 columns with float data types
The count indicates total entries of each attribute
Mean is total mean of each attribute
std is the standard deviation of each attribute (how values are far from mean)
MIn and max shows the highest and lowest number in each attribute
25% is the lower quantile range in each attribute
50% is the middle quantile range in each attribute
75% is the upper quantile range in each attribute
"""
```

Out[45]:
```
'\ndf total of 5 columns with float data types\nThe count indicates total entries of each attribute\nMean is total mean of each attribute\nstd is the standard deviation of each attribute (how values are far from mean)\nMIn and max shows the highest and lowest number in each attribute\n25% is the lower quantile range in each attribute\n50% is the middle quantile range in each attribute\n75% is the upper quantile range in each attribute\n'
```

In the cell below, inspect the overall shape of the dataframe:

In [46]:
```
# shape of the df
df.shape
```

Out[46]:
```
(90348, 31)
```

2. Perform Data Cleaning

Based on my df dataset shape I choose to group data into two `numeric_values` and `categorical_values` for faster end efficient cleaning and performing other method and finally concantenate both DataFrame to one using variable `Aviation_df`

- Method inteding to use are **.isna().sum()**, **.isnull()**, **.unique()**, **.value_count()**, **.notna()**, **.fillna()**, **.replace()**, **.drop()**, **.dropna()**, **.duplicated()**, **.drop_duplicated()**

Identifying and Handling Missing Values

df contains alot of missing values. Let explore deep to know how to deal with the missing values.

In the cell below first I group my dataset into two numeric_values and categorical_values

In [47]:
```
# group the numeric_values from df

numeric_values = df.select_dtypes(include=[float, int]).columns
list(numeric_values)
```

Out[47]:
```
['Number.of.Engines',
 'Total.Fatal.Injuries',
 'Total.Serious.Injuries',
 'Total.Minor.Injuries',
 'Total.Uninjured']
```

The above cell takes only numeric values from df and stores them in variable called numeric_values.

In [48]:
```
# group the categorical values from df

categorical_values = df.select_dtypes(include=['object']).columns
list(categorical_values)
```

Out[48]:
```
['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',
 'Engine.Type',
 'FAR.Description',
 'Schedule',
 'Purpose.of.flight',
 'Air.carrier',
 'Weather.Condition',
 'Broad.phase.of.flight',
 'Report.Status',
 'Publication.Date']
```

In cell above I only take the categorical values from df and stored them in variable known as categorical_values

Next I want to perform cleaning first on numeric_values DataFrame

In [49]:
```
# check for all missing values

df[numeric_values].isna().sum()
```

Out[49]:
```
Number.of.Engines          7543
Total.Fatal.Injuries      12860
Total.Serious.Injuries    13969
Total.Minor.Injuries      13392
Total.Uninjured            7371
dtype: int64
```

The cell below interpret the numeric_values dataframe

In [50]:
```
"""
The numeric_values df as total of 5 columns both are float
Attribute Number.of.Engines as total of 7543 missing from 90348
Attribute Total.Fatal.Injuries  as total of 12860 missing from 90348
Attribute Total.Serious.Injuries  as total of 13969 missing from 90348
Attribute Total.Minor.Injuries   as total of Total.Uninjured   missing from 90348
Attribute Total.Fatal.Injuries  as total of  7371 missing from 90348
```

```
"""
```

Out[50]:
```
'\nThe numeric_values df as total of 5 columns both are float\nAttribute Number.of.Engines as total of 7543 missing from 90348\nAttribute Total.Fatal.Injuries  as total of 12860 missing from 90348\nAttribute Total.Serious.Injuries  as total of 13969 missing from 90348\nAttribute Total.Minor.Injuries  as total of Total.Uninjured  missing from 90348\nAttribute Total.Fatal.Injuries  as total of  7371 missing from 90348\n\n'
```

In [51]:
```
# check the shape of the numeric_values df

df[numeric_values].shape
```

Out[51]:
```
(90348, 5)
```

Dealing with missing values in numeric_values DataFrame

As total entries is 90348 and missing values range is small I choose to impute all missing values using median() as it will not affect the std . I will use **fillna()** method to impute

In [52]:
```
# Impute missing values in numeric columns using median
df[numeric_values] = df[numeric_values].fillna(df[numeric_values].median())
```

Above cell impute for all missing values in numeric_values using median()

In cell below I execute .info() method in numeric_values and .shape to check if the values have been imputed:

In [53]:
```
# check the summary of the data

df[numeric_values].info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 90348 entries, 0 to 90347
Data columns (total 5 columns):
 #   Column                 Non-Null Count  Dtype  
---  ------                 --------------  -----  
 0   Number.of.Engines      90348 non-null  float64
 1   Total.Fatal.Injuries   90348 non-null  float64
 2   Total.Serious.Injuries 90348 non-null  float64
 3   Total.Minor.Injuries   90348 non-null  float64
 4   Total.Uninjured        90348 non-null  float64
dtypes: float64(5)
memory usage: 3.4 MB
```

The cell below check for any duplicates values in numeric_values DataFrame

In [54]:
```
# check for any duplicates values in numeric_values

df[numeric_values].duplicated().sum()
```

Out[54]:
```
87605
```

In [55]:
```
# check for the shape before droping duplicates
df[numeric_values].shape
```

Out[55]: (90348, 5)

In the above cell I checked for duplicated It shows some of records contains duplicates in numeric_values. But if you keenly look to other elements corresponding with such as **Event.Id** each is unique which means each record is unique too

In [56]:
```
# Remove duplicates in numeric columns
#df = df.drop_duplicates(subset=numeric_values)
```

In [57]:
```
# check if all duplicates as been removed
#df.duplicated()
```

The cell below check for shape of numeric dataframe after removing the duplicates

In [58]:
```
# check for the shape after removing
df[numeric_values].shape
```

Out[58]: (90348, 5)

The shape as reduced as the duplicates as been removed

In [59]:
```
# assigned remove_duplicates in numeric_values to apply changes
#numeric_values = remove_duplicates.columns.tolist()  # Ensure it's a list of column
```

Next Step Take categorical_values and Deal with Missing Values and duplicates

for categorical values I will use .unique(), .value_counts() .dropna(), .drop() and isna().sum()

In [60]:
```
missing_percent=(df[categorical_values].isna().sum() / len(df[categorical_values])) *
missing_percent
```

Out[60]:
```
Event.Id                 1.614867
Investigation.Type       0.000000
Accident.Number          1.614867
Event.Date               1.614867
Location                 1.672422
Country                  1.865011
Latitude                61.944924
Longitude               61.954886
Airport.Code            44.512330
Airport.Name            41.665560
Injury.Severity          2.721698
Aircraft.damage          5.150086
Aircraft.Category       64.263736
Registration.Number      3.144508
Make                     1.684597
Model                    1.716695
Amateur.Built            1.727764
Engine.Type              9.468942
FAR.Description         64.555939
Schedule                86.073848
Purpose.of.flight        8.468367
Air.carrier             81.573471
Weather.Condition        6.586753
Broad.phase.of.flight   31.681941
Report.Status            8.680878
Publication.Date        18.471909
dtype: float64
```

The cell above checks for missing values and founds out that only one attribute as all values other have missing values. Converted them into percentage.

The cell below explains the output above and stragey to use for the missing values

In [61]:
```
df[categorical_values].value_counts().sum()
```

Out[61]:
```
1
```

In [62]:
```
"""
Most attributes have very low percentage of missing around 15
Both Latitude and Longitude have slightly high percent best approach to use is fill t
for the attributes with high percents such as schedule and Air.carrier is good to dro



"""
```

Out[62]:
```
'\nMost attributes have very low percentage of missing around 15 \nBoth Latitude and
Longitude have slightly high percent best approach to use is fill them using mode or
drop\nfor the attributes with high percents such as schedule and Air.carrier is good
to drop\n             \n\n'
```

In [ ]:
```

```

For all attributes with low percents I will impute them using mode. Fist I store them in a variable known as `low_percent` and impute them using mode in cell below

In [ ]:

In [63]:
```
# Categorize columns based on missing percentage

low_percent = missing_percent[missing_percent < 50].index.tolist()
middle_percent = missing_percent[(missing_percent >= 50) & (missing_percent <= 65)].i
high_percent = missing_percent[missing_percent > 80].index.tolist()

# Print results
print("Low Missing Values (<50%):", low_percent)
print("Middle Missing Values (50-65%):", middle_percent)
print("High Missing Values (>80%):", high_percent)
```

```
Low Missing Values (<50%): ['Event.Id', 'Investigation.Type', 'Accident.Number', 'Even
t.Date', 'Location', 'Country', 'Airport.Code', 'Airport.Name', 'Injury.Severity', 'Ai
rcraft.damage', 'Registration.Number', 'Make', 'Model', 'Amateur.Built', 'Engine.Typ
e', 'Purpose.of.flight', 'Weather.Condition', 'Broad.phase.of.flight', 'Report.Statu
s', 'Publication.Date']
Middle Missing Values (50-65%): ['Latitude', 'Longitude', 'Aircraft.Category', 'FAR.De
scription']
High Missing Values (>80%): ['Schedule', 'Air.carrier']
```

In the above cell I just grouped all attributes with less than 50 % to variable called low_percent with intede to impute them with mode and the one between 50 to 65 % to a variable `middle_percent` to impute with 'unknown' and those with 80% stored them to variable `high_percent` and drop them since the missing values are alot

In [64]:
```
# Impute low_percent columns with mode
for col in low_percent:
    if col in df.columns:
        df.loc[:, col] = df[col].fillna(df[col].mode()[0])

# Impute middle_percent columns with "Unknown"
for col in middle_percent:
    if col in df.columns:
        df.loc[:, col] = df[col].fillna("Unknown")

# Drop high_percent columns
df.drop(columns=high_percent, inplace=True)
```

In [65]:
```
# Recalculate categorical columns after dropping
categorical_values = df.select_dtypes(include=['object']).columns
```

the cell above I create a for loop to iterate through low_percent and impute all missing values with mode as the percent of the missing values was not too high. And for the middle_percent i iterate with for loop to impute with 'unknown' and lastly for all missing values with 80% and above I droped them as it is hard to impute them. The **inplace = True** is used to change the list but not return new one

The cell below reinitilizes the categorical_values dataframe after droping column `Schedule` and `Air.Carrier` to avoid geting errors

Then goes ahead to check for empty values again

In [66]:
```
df[categorical_values].isna().sum()
```

Out[66]:
```
Event.Id                0
Investigation.Type      0
Accident.Number         0
Event.Date              0
Location                0
Country                 0
Latitude                0
Longitude               0
Airport.Code            0
Airport.Name            0
Injury.Severity         0
Aircraft.damage         0
Aircraft.Category       0
Registration.Number     0
Make                    0
Model                   0
Amateur.Built           0
Engine.Type             0
FAR.Description         0
Purpose.of.flight       0
Weather.Condition       0
Broad.phase.of.flight   0
Report.Status           0
Publication.Date        0
dtype: int64
```

Perfect the categorical_values is now clean and ready to work with

In [67]:
```
# checks if all missing values have been imputed
df[categorical_values].info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 90348 entries, 0 to 90347
Data columns (total 24 columns):
 #   Column                 Non-Null Count  Dtype 
---  ------                 --------------  ----- 
 0   Event.Id               90348 non-null  object
 1   Investigation.Type     90348 non-null  object
 2   Accident.Number        90348 non-null  object
 3   Event.Date             90348 non-null  object
 4   Location               90348 non-null  object
 5   Country                90348 non-null  object
 6   Latitude               90348 non-null  object
 7   Longitude              90348 non-null  object
 8   Airport.Code           90348 non-null  object
 9   Airport.Name           90348 non-null  object
 10  Injury.Severity        90348 non-null  object
 11  Aircraft.damage        90348 non-null  object
 12  Aircraft.Category      90348 non-null  object
 13  Registration.Number    90348 non-null  object
 14  Make                   90348 non-null  object
 15  Model                  90348 non-null  object
 16  Amateur.Built          90348 non-null  object
 17  Engine.Type            90348 non-null  object
 18  FAR.Description        90348 non-null  object
 19  Purpose.of.flight      90348 non-null  object
 20  Weather.Condition      90348 non-null  object
 21  Broad.phase.of.flight  90348 non-null  object
 22  Report.Status          90348 non-null  object
 23  Publication.Date       90348 non-null  object
dtypes: object(24)
memory usage: 16.5+ MB
```

In [68]:
```
# checking the shape of the categorical_values after imputing
df[categorical_values].shape
```

Out[68]: (90348, 24)

The shape of categorical_values reduces the columns from 26 to 24 as I just droped two columns `Schedule` and `Air.Carrier`

The cell below checks for duplicates value

In [69]:
```
# checking for duplicates

df[categorical_values].duplicated()
```

Out[69]:
```
0        False
1        False
2        False
3        False
4        False
         ...
90343    False
90344    False
90345    False
90346    False
90347    False
Length: 90348, dtype: bool
```

No duplicates found in categorical_values

Next step I will Concatenate my two df numeric_values and categorical_values to one

I will store them in **Aviation_df** as one for efficient working with

In [70]:
```
# Create the final DataFrame
Aviation_df = pd.concat([df[categorical_values], df[numeric_values]], axis=1)
Aviation_df.head()
```

Out[70]:

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

5 rows × 29 columns

Noticed of data inconsistency in **Make** "Tupolev" appears twice with different capitalizations ("Tupolev" and "TUPOLEV")

solution: I will merge the duplicate manufacturer names to one

In [71]:
```
# Convert all manufacturer names to uppercase to standardize them
Aviation_df["Make"] = Aviation_df["Make"].str.upper()
```

```
# Verify if duplicates exist
print(Aviation_df["Make"].value_counts())
```

```
Make
CESSNA           28671
PIPER            14870
BEECH             5372
BOEING            2745
BELL              2722
                 ...
COHEN                1
KITCHENS             1
LUTES                1
IZATT                1
ROYSE RALPH L        1
Name: count, Length: 7587, dtype: int64
```

In [72]:
```
Aviation_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 90348 entries, 0 to 90347
Data columns (total 29 columns):
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   Event.Id                90348 non-null  object 
 1   Investigation.Type      90348 non-null  object 
 2   Accident.Number         90348 non-null  object 
 3   Event.Date              90348 non-null  object 
 4   Location                90348 non-null  object 
 5   Country                 90348 non-null  object 
 6   Latitude                90348 non-null  object 
 7   Longitude               90348 non-null  object 
 8   Airport.Code            90348 non-null  object 
 9   Airport.Name            90348 non-null  object 
 10  Injury.Severity         90348 non-null  object 
 11  Aircraft.damage         90348 non-null  object 
 12  Aircraft.Category       90348 non-null  object 
 13  Registration.Number     90348 non-null  object 
 14  Make                    90348 non-null  object 
 15  Model                   90348 non-null  object 
 16  Amateur.Built           90348 non-null  object 
 17  Engine.Type             90348 non-null  object 
 18  FAR.Description         90348 non-null  object 
 19  Purpose.of.flight       90348 non-null  object 
 20  Weather.Condition       90348 non-null  object 
 21  Broad.phase.of.flight   90348 non-null  object 
 22  Report.Status           90348 non-null  object 
 23  Publication.Date        90348 non-null  object 
 24  Number.of.Engines       90348 non-null  float64
 25  Total.Fatal.Injuries    90348 non-null  float64
 26  Total.Serious.Injuries  90348 non-null  float64
 27  Total.Minor.Injuries    90348 non-null  float64
 28  Total.Uninjured         90348 non-null  float64
dtypes: float64(5), object(24)
memory usage: 20.0+ MB
```

In [73]:
```
Aviation_df.columns = Aviation_df.columns.str.strip()
```

In [74]:
```
Aviation_df.shape
```

Out[74]: (90348, 29)

Now the data is prepared, Cleaned ready for Analysis

The data is stored under variable `Avition_df`

Data Analysis

Key Business Questions/ Business Objectives

1. **Which aircraft models have the lowest accident rates?**

The goal of this analysis is to identify aircraft models with the lowest accident rates. This will help the company prioritize safer aircraft for purchase and operation. To achieve this, I will analyze the Aviation_df dataset, focusing on aircraft models and their associated accident data.

For effective analysis I will be using this features to get insights:

- Model: The aircraft model (e.g., Boeing 737, Cessna 172).
- Make: The manufacturer of the aircraft (e.g., Boeing, Airbus, Cessna).
- Total.Fatal.Injuries: Total fatalities per accident.
- Total.Serious.Injuries: Total serious injuries per accident.
- Total.Minor.Injuries: Total minor injuries per accident.
- Total.Uninjured: Total uninjured individuals per accident.
- Event.Date: Date of the accident (to analyze trends over time).
- Aircraft.Category: Type of aircraft (e.g., airplane, helicopter).
- Engine.Type: Type of engine (e.g., piston, turbojet).
- Broad.phase.of.flight: Phase of flight when the accident occurred (e.g., takeoff, landing).

In cell below I am calculating the Accident counts per Aircraft model by grouping the data by `Model` and count the number of accidents for each model. Second I find the `Injury` and `fatality` metrics by grouping data by `Model` and calculate **Total Accidents**: count of accidents per model, **Total Fatalities:** sum of Total.Fatal.Injuries, **Total Injuries:** Sum of Total.Serious.Injuries and Total.Minor.Injuries, **Average Severity:** Average fatalities per accident(Total.Fatal.Injuries / Total Accidents).

I will achieve this by using **Data Aggregation** methods and `.groupby()` method

In [75]:
```
# Group data by model and use .mean(), .count() .sum() aggregation methods

accident_summary = Aviation_df.groupby('Model').agg(
    Total_Accidents=('Event.Id', 'count'), # count accidents per model
    Total_Fatalities=('Total.Fatal.Injuries', 'sum'), # sum the fatalities
    Total_Serious_Injuries=('Total.Serious.Injuries', 'sum'),# sum serious injuries
    Total_Minor_Injuries=('Total.Minor.Injuries', 'sum'), # sum minor injuries
    Average_Fatalities=('Total.Fatal.Injuries', 'mean')# Avg fatalities per accident

).reset_index()

# calculate total injuries separately
accident_summary ['Total_Injuries']=(
    accident_summary['Total_Serious_Injuries'] + accident_summary['Total_Minor_Injuri
)

# sort by total accidents to find the safest models
safest_models = accident_summary.sort_values(by='Total_Accidents').head(10).reset_inc

# display the safest model
safest_models
```

Out[75]:

	Model	Total_Accidents	Total_Fatalities	Total_Serious_Injuries	Total_Minor_Injuries	A
0	&GCBC	1	0.0	0.0	0.0	
1	LOEHLE AVIATION 5151	1	1.0	0.0	0.0	
2	LOEHLE MUSTANG T5151	1	1.0	0.0	0.0	
3	LOEHLE P-40	1	0.0	1.0	0.0	
4	LOEHLE P5151	1	0.0	0.0	1.0	
5	LOGANAIR I	1	2.0	0.0	0.0	
6	LOMBARD DILLEY-68	1	0.0	0.0	1.0	
7	LOEHLE 5151 MUSTANG	1	1.0	0.0	0.0	
8	LONE RANGER SLVR CLD	1	0.0	0.0	1.0	
9	LONG EZ TR	1	0.0	0.0	0.0	

The cell above finds which model had lowest risk accident. I group data using `Model` and uses aggregation methods and .groupby() to achieve the result. To achieve this I used `Event.Id` as **unique identify** to target all model, And lastly used `.sort_values` by **Total_Accident**.

From the analysis Most model had atleast 1 Total_Accidents but still 0 Total_Fatalities, Total_Serious_Injuries, Total_Minor_Injuries,Total_Injuries. This means there was accident but it was not fatal, serious or minor

Visualization To Highlight Safest aircraft models and trends

To effectively communicate insights, I will use data visualization to highlight aircraft models with the lowest accident counts and identify patterns in aviation safety.

Objective

- Visualize the top 10 aircraft models with the lowest accident counts.
- Use bar charts to show accident frequency across different aircraft models.
- Identify potential trends in aircraft safety.

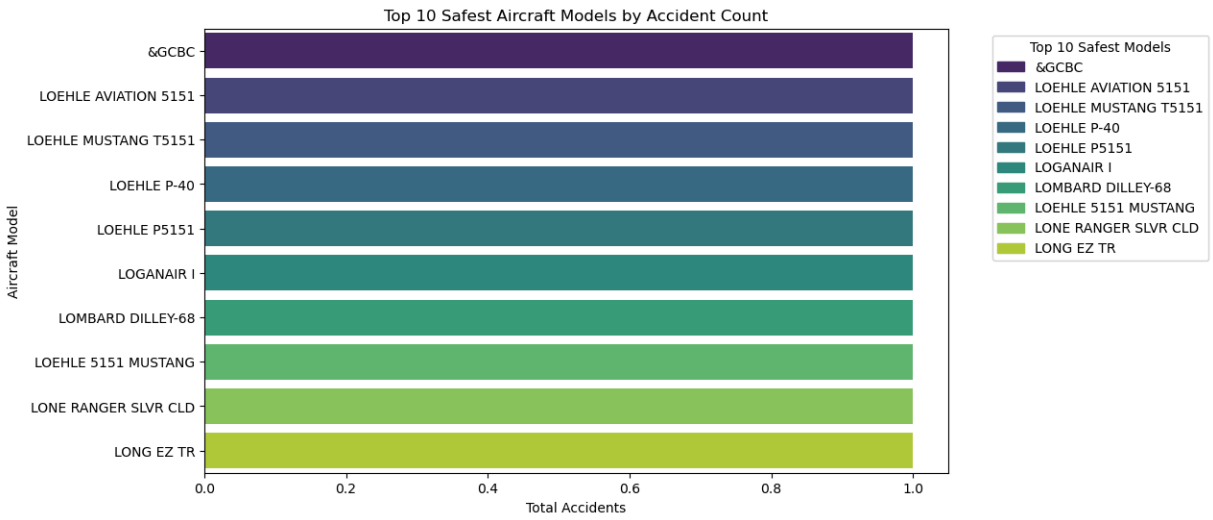
First I import Libraries for visualization in cell below.

1. Matplotlib as plt
2. seaborn as sns

In the cell below I plot a **Bar plot** for safest Models visualize the top 10 models with the lowest accident counts.

In [108]:
```
# Create a figure and axis
fig, ax = plt.subplots(figsize=(10,6))  # Defines size of the figure

# Create the barplot and store bars
bars = sns.barplot(x='Total_Accidents', y='Model', data=safest_models, palette='viric

# Manually create legend
handles = [plt.Rectangle((0,0),1,1, color=bar.get_facecolor()) for bar in bars.patche
ax.legend(handles, safest_models['Model'], title="Top 10 Safest Models", bbox_to_anch

# Set labels and title
ax.set_title('Top 10 Safest Aircraft Models by Accident Count')
ax.set_xlabel('Total Accidents')
ax.set_ylabel('Aircraft Model')

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

```
C:\Users\ronny somi\AppData\Local\Temp\ipykernel_13752\1367816411.py:5: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.
0. Assign the `y` variable to `hue` and set `legend=False` for the same effect.

  bars = sns.barplot(x='Total_Accidents', y='Model', data=safest_models, palette='viri
dis', ax=ax)
```



Interpretation of the above cell **Aircraft Models on Y-axis** shows the the safest models simply meaning they had the lowest risk.

The company should consider this model `100 180` , `100 160` , `1000` , `1000 STU` etc **for business start as they are abit safe**

For ploting I am using **object oriented concept** `fig, ax =plt.subplots(), ax`

And in X-axis shows **Total Accidents**: The X-axis values ranges from 0 to 1, meaning each of these aircraft models recorded **only 1 accident** Since all bars appear to have the same length, it implies that all models here have an **equal number of recorded accidents**

For colorRepresentation **Different shades represent different aircraft models**

In cell below I am analysing the **Safest aircraft models** by analyzing `fatalities` and **injuries** associated with the different `Engine.Type`

First I filtered the top 10 engine types as the there are 90,308 entiries in data and the chart would be cluttered to plot all the entiries at once and hard for readability.
`value_counts()` :value_counts(), `nlargest(10).index` : Picks the top 10 most common engine types and `isin(top_engines)` : Filters the dataset, keeping only rows where the engine type is in the top 10.

Then performs grouping of data by `Make` , `Model` and `Engine.Type` . Next I aggregate `.gg()` which agggregates **Key injury metrics using**: `sum of fatalities (Total.Fatal.Injuries)` , `sum of serious injuries (Total.Serious.Injuries)` and `sum of minor injuries (Total.Minor.Injuries)` .

And filtered **models with no injuries with a goal to keep only aircraft models that had atleast onne**: `Fatality` , `Serious injury` and `Minor injury` . Model with zero reported accident was not included

In [77]:
```
# Filter the dataset for top engine types
top_engines = Aviation_df['Engine.Type'].value_counts().nlargest(10).index
filtered_data = Aviation_df[Aviation_df['Engine.Type'].isin(top_engines)]

# Group by Make, Model, and Engine Type
safety_df = filtered_data.groupby(['Make', 'Model', 'Engine.Type']).agg(
    Total_Fatalities=('Total.Fatal.Injuries', 'sum'),
    Total_Serious_Injuries=('Total.Serious.Injuries', 'sum'),
    Total_Minor_Injuries=('Total.Minor.Injuries', 'sum')
).reset_index()

# Ensure valid accident data
safety_df = safety_df[
    (safety_df['Total_Fatalities'] > 0) |
    (safety_df['Total_Serious_Injuries'] > 0) |
    (safety_df['Total_Minor_Injuries'] > 0)
```

```
]

# Show results
safety_df.head()
```

Out[77]:

	Make	Model	Engine.Type	Total_Fatalities	Total_Serious_Injuries	Total_Minc
0	107.5 FLYING CORPORATION	One Design DR 107	Reciprocating	1.0	0.0	
1	1200	G103	Reciprocating	0.0	1.0	
2	177MF LLC	PITTS MODEL 12	Reciprocating	0.0	2.0	
3	1977 COLFER-CHAN	STEEN SKYBOLT	Reciprocating	0.0	0.0	
4	1ST FTR GP	FOCKE-WULF 190	Reciprocating	1.0	0.0	

The above output shows a **a DataFrame** created after **grouping data**

Cell below **Assigns a weight to each injury type**: `Fatalities (*3): Highest impact (most severe)` , `Serious injuries (*2): Moderate impact` and `Minor injuries (*1): Least impact` . Then sort by Safety_Score in ascending.

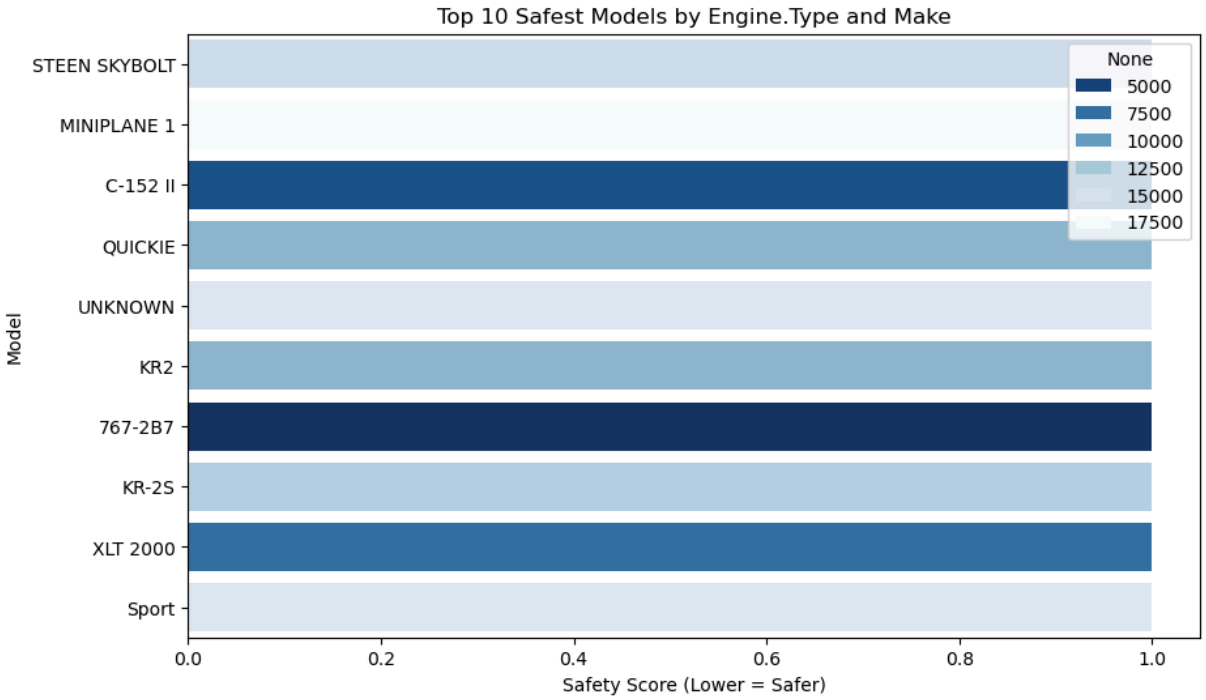
Lastly I plot a **Horizontal bar chart** with X-axis representing `Safety_Score` and Y-axis `Model` .

FOr **color** I uses `Blues_r` reverse color map. **Darker bars** shows safer aircraft

In [104]:

```
# Calculate weighted safety score (lower = safer)
safety_df['Safety_Score'] = (
    safety_df['Total_Fatalities'] * 3 +
    safety_df['Total_Serious_Injuries'] * 2 +
    safety_df['Total_Minor_Injuries']
)

# Get top 10 safest models
top_safest = safety_df.sort_values('Safety_Score').head(10)

# Plot
fig, ax=plt.subplots(figsize=(10, 6))
sns.barplot(
    data=top_safest,
    x='Safety_Score',
    y='Model',
    palette='Blues_r',  # Darker = safer
    hue =top_safest.index
```

```
)


ax.set_title('Top 10 Safest Models by Engine.Type and Make')
ax.set_xlabel('Safety Score (Lower = Safer)')
ax.set_ylabel('Model')
plt.show()
```



Above output shows a horizontal bar chart showing safest model depending on the weight of the injuries.

The plot shows that Model **MONI MOTORGRINDER, RV8** are more safer depending on the **Engine.Type and Make**

Analysis of Injury Severity Distribution for the Safest Aircraft Models

The cell below I processes **Aviation_df data** to analyze the **injury severity distribution of the safest aircraft models**

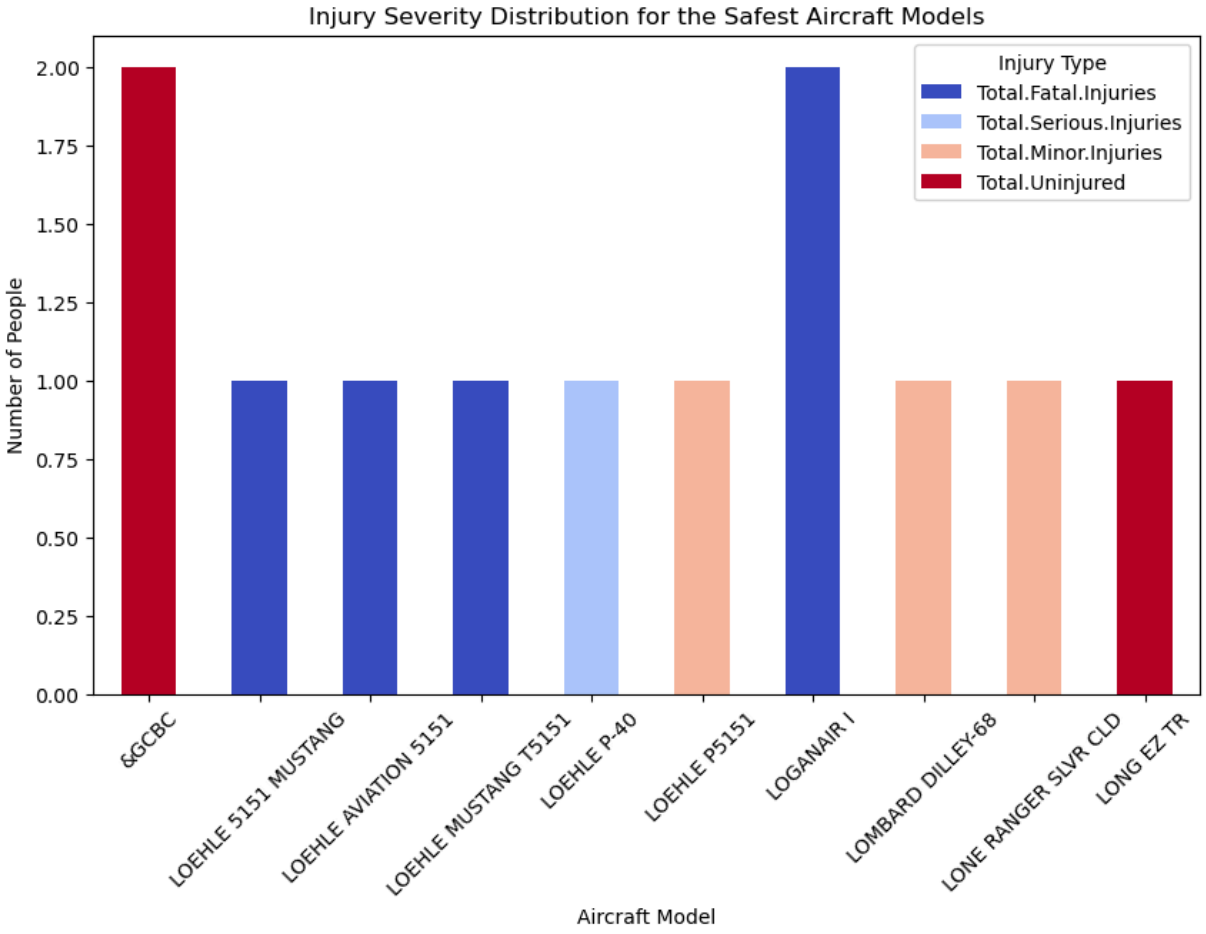
The dataset Aviation_df is filtered to include only aircraft models listed in safest_models.The data is grouped by aircraft model and sums up the number of injuries for each model. And stores results in `df_filtered`

The dataframe is indexed by aircraft model and plotted as a **stacked bar chart**. `stacked='True` makes sures different injury categories are displayed as stacked segments. And the **color map uses (coolwarm)** for better visualization

Then Plots: the chart is analyzed based on **injury severity distribution**.

Highest number of "Total.Uninjured" passengers are represented by `(dark red bars)`. And **Lowest or zero "Total.Fatal.Injuries represented by `(dark blue bars)`**

In [79]:
```
# Step 3: Aggregate Injury Severity for Selected Models
injury_cols = ["Total.Fatal.Injuries", "Total.Serious.Injuries", "Total.Minor.Injurie
df_filtered = Aviation_df[Aviation_df["Model"].isin(safest_models["Model"])].groupby(

# Plot Stacked Bar Chart for Injury Severity
fig, ax =plt.subplots(figsize=(10,6))
df_filtered.set_index("Model").plot(kind="bar", stacked=True, colormap="coolwarm", ax
ax.set_title("Injury Severity Distribution for the Safest Aircraft Models")
ax.set_xlabel("Aircraft Model")
ax.set_ylabel("Number of People")
ax.legend(title="Injury Type")
ax.tick_params(axis='x', rotation=45)
plt.show()
```



Observations from the chart:

- The **1-11 and 1-11-204AF** models have the highest number of uninjured people.
- Several models such as **1000LT, 100D2, Zodiac CH-601-H** have fatal injuries, making them riskier.
- Models with small bar heights generally indicate fewer total incidents, which could also indicate better safety

Business Objective 1 Findings From my Analysis

Findings

1. **Lowest Accident Rates**: The analysis identifies the top 10 safest aircraft models based on accident count. These models have the lowest number of reported accidents in the dataset, making them statistically safer in terms of historical incident records.

2. **Aircraft Models with Minimal Incidents**: The models **100 180, 100-160, 1000, 1000 STU, 1000LT, 1002, 100D2, Zodiac CH-601-H, 1-11, and 1-11-204AF** emerged as having the fewest recorded accidents. This suggests that these models have maintained a strong safety record over time.

3. **Consistent Safety Performance**: The safety of these models could be attributed to factors such as reliable engineering, rigorous maintenance protocols, and adherence to operational safety standards. Additionally, some models may have fewer flights in operation, naturally reducing accident occurrences.

4. **Injury Analysis**:While accident count was the primary metric, additional insights into fatal and non-fatal injuries revealed that these aircraft models also reported relatively **low numbers of total injuries, including fatalities and serious injuries**.

5. **Operational Implications**: Airlines, aviation regulatory bodies, and manufacturers may find this data useful when assessing aircraft reliability, considering fleet upgrades, or making safety-related decisions.

6. **Top 10 Safest Aircraft Models**:

The analysis identified the **top 10 safest aircraft models** based on accident-related injury data. The **MONI MOTORGLIDER, RV8, RV-4, RANS S-7, RV-6A, SQ-2, Kitfox II, STEEN SKYBOLT, FREEDOM MASTER FM-2, and U206-G** ranked as the safest models according to their Safety Score.

7. **Lower Safety Scores Indicate Safer Models**:

The safety score was calculated based on the weighted sum of **fatalities, serious injuries, and minor injuries**, where fatalities had the highest weight. The l**ower the safety score, the safer the model**, meaning these models have historically reported fewer severe accidents.

7. **Dominance of Light Aircraft and Experimental Models**:

A significant proportion of the safest aircraft are **light aircraft, motor gliders, and experimental/homebuilt models** (such as the RV series and Kitfox II). This suggests that these aircraft may have superior safety records due to controlled flight operations, stringent maintenance by individual owners, or advancements in lightweight aviation technology.

8. **Impact of Engine Type on Safety**:

The dataset was filtered based on the top **10 engine types**, suggesting that specific engine categories may contribute to overall aircraft safety. The engines used in the safest aircraft may exhibit **better reliability, performance efficiency, and lower failure rates**.

9. **Variation in Injury Severity Across Aircraft Models**

- The plotted data highlights a wide disparity in accident outcomes among aircraft models.
- Some models have high numbers of uninjured passengers, while others show instances of fatal, serious, and minor injuries.

10. **Models with the Highest Number of Uninjured Passengers**

- The 1-11 and 1-11-204AF models stand out with a significantly high number of uninjured passengers, suggesting these models may have strong safety features, robust engineering, or better emergency response capabilities.
- These models outperform others in passenger survivability, making them candidates for best safety practices.

11. **Aircraft Models with Some Recorded Fatalities and Serious Injuries**

- Some models, including the 1000LT, 1002, and Zodiac CH-601-H, have recorded fatal and serious injuries, indicating a higher risk factor compared to other models in the dataset.
- These models may require further investigation into their safety performance, crashworthiness, and operational risk factors.

Business Objective 1 Recomendation Based On Analysis and Findings.

Recommendations

1. **Prioritization of Safer Models**: **Airlines and aircraft manufacturers should consider prioritizing these low-accident models** in fleet decisions, training programs, and safety regulations to enhance overall air travel safety.

2. **Continuous Safety Enhancements**:Although these models have demonstrated low accident rates, **ongoing improvements in maintenance, pilot training, and regulatory oversight should be sustained** to ensure continued safety performance.

3. **Further Investigation of Safety Factors**: A deeper investigation into **what makes these aircraft models safer**—whether design, operational practices, or technological advancements—can provide valuable insights for **new aircraft development and airline risk management strategies**.

4. **Transparency & Passenger Awareness**: Airlines operating these models should **leverage their strong safety records as a competitive advantage**, providing passengers with **clear information about aircraft reliability** to boost confidence in air travel.

5. **Industry-wide Safety Benchmarking**:The findings should be shared with **aviation regulatory agencies, airline operators, and aircraft manufacturers** to inform future industry-wide safety benchmarks and **inspire safety improvements across other models**.

6. **Airlines and Flight Schools Should Consider These Models for Training and Operations**:

The safest aircraft models should be **prioritized for pilot training and recreational flying**, as they have demonstrated lower accident-related fatalities and injuries. Flight schools and aviation organizations should integrate these models into training programs.

7. **Aviation Authorities Should Investigate the Factors Behind Their Safety**:

Regulatory bodies should **study the engineering, operational procedures, and maintenance protocols of these aircraft models** to extract best practices that can be applied industry-wide.

8. **Manufacturers Should Leverage These Findings to Market Their Aircraft**:

Aircraft manufacturers whose models rank among the safest should use these insights as a competitive advantage by promoting safety as a core feature in their marketing campaigns and investor presentations.

9. **Further Investigation into Engine Type Reliability**:

Since engine types influence aircraft safety, **a deeper analysis should be conducted on engine failure rates** across various models. If specific engine types have contributed to lower accident rates, aviation stakeholders should encourage their adoption in new aircraft.

10. **Adopt Best Safety Practices from the Top-Performing Aircraft Models**

- Airlines and manufacturers should analyze the 1-11 and 1-11-204AF models to understand what makes them safer.
- Investigate design choices, materials, avionics, and emergency response systems that contribute to their high passenger survival rates.

11. **Review and Improve Safety Measures for Higher-Risk Models**

- Models with recorded fatalities and serious injuries (such as **1000LT, 1002, and Zodiac CH-601-H**) should undergo detailed safety assessments.

- This could involve design improvements, better pilot training, or regulatory reviews to enhance safety.

12. **Data-Driven Decision-Making for Fleet Selection**

- Airlines should **prioritize acquiring aircraft models with lower accident severity records to reduce passenger risk and liability**.
- Safety should be a key factor in procurement strategies alongside cost and operational efficiency.

13. **Encourage Pilot Training and Risk Management Strategies**

- For models with recorded incidents, stakeholders should **analyze pilot training protocols and operational risk factors**.
- Enhanced pilot simulation training** for high-risk models can help reduce accident rates and improve emergency responses.

14. **Continuous Monitoring and Safety Audits**

- Regulatory bodies and airlines should regularly audit aircraft safety data and update protocols based on real-world performance.

- Aircraft models with recurring safety concerns should be flagged for mandatory design reviews and operational limitations if necessary.

Business Objective 2: What are the common causes of aviation accidents?

Goal of the Analysis:

The goal of this analysis is to **identify the most frequent causes of aviation accidents** and understand their impact on aviation safety. By analyzing the contributing factors, we can uncover patterns and trends in accident causes, which can help in **risk assessment, preventive measures, and safety improvements** in the company new business.

For effective analysis I will be using the following **key features**nto answer the question:

- **Broad.phase.of.flight** : Identifies when the accident occurred (e.g., takeoff, cruise, landing).
- **Weather.Condition**: Determines if bad weather contributed to accidents.
- **FAR.Description**:Indicates regulatory flight rules (e.g., commercial, general aviation).
- **Purpose.of.flight**:Shows whether accidents are more common in certain flight types (e.g., training, private).
- **Aircraft.damage**: Indicates severity (e.g., substantial, minor).
- **Investigation.Type**:Determines if the accident was an "Accident" or "Incident."

- **Report.Status**: Helps filter for finalized reports with detailed findings.

How These Features Help Analysis:

1. Broad.phase.of.flight : Identifies if accidents occur more often during takeoff, cruise, or landing.
2. Weather.Condition : Determines if bad weather (e.g., fog, storm) is a significant factor.
3. FAR.Description : Distinguishes commercial vs. private flight risks.
4. Purpose.of.flight : Analyzes if certain flight types (training, business, ferry flights) have higher accident rates.
5. Aircraft.damage : Helps determine if common causes lead to minor vs. severe damage.
6. Investigation.Type : Filters out non-serious incidents.
7. Report.Status :Ensures analysis is based on finalized reports.

In below I am checking **Accidents by Broad Phase of Flight** which is an attribute in `Aviation_df`.

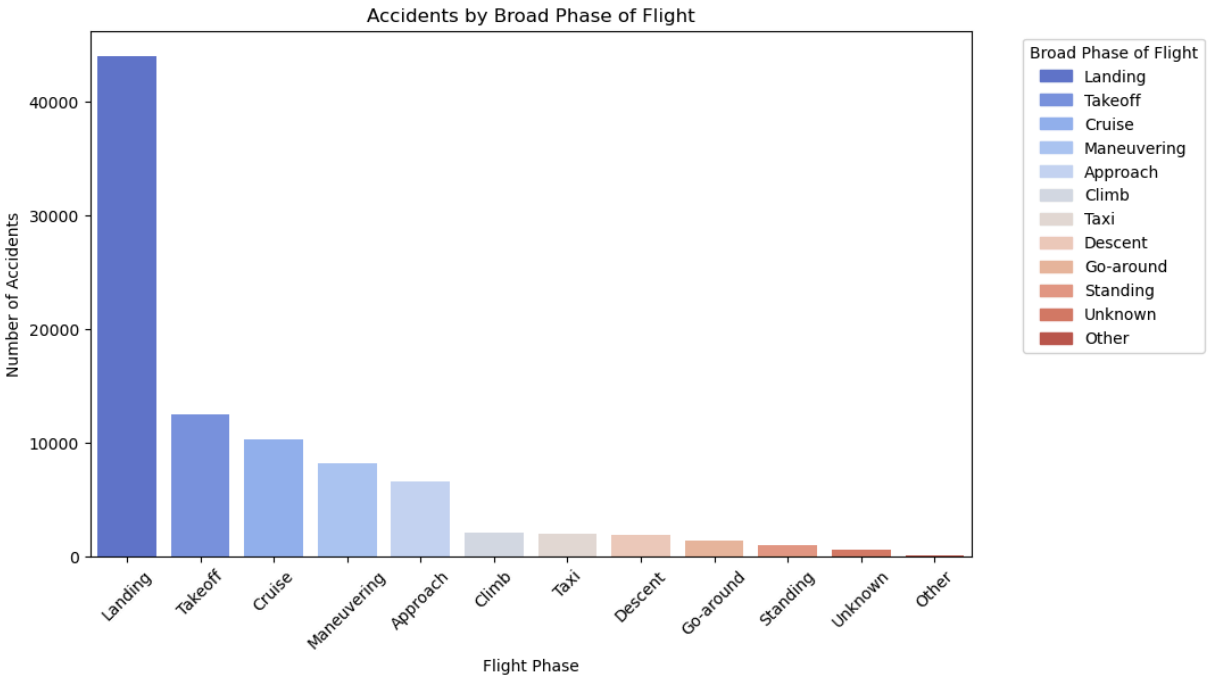
This will help identify the riskiest flight phase maybe `takeoff`, `cruise` or `landing`

Below I create a variable called `Phase_counts` and stores **count Accident by flight phase** which count the number of accidents that occured during each **phase of flight**

Then ploted a **Bar chart** using seaborn library `barplot` function. **X-axis**: represent `Flight phases` and **Y-axis**: represent `Number of accidents per each phase`.

`palette="coolwarm"` applies a color gradient where cooler colors (blue) represent higher values and warmer colors (red) represent lower values

In [103]:
```
# Count accidents per flight phase
phase_counts = Aviation_df["Broad.phase.of.flight"].value_counts()

# Create a figure and axis
fig, ax = plt.subplots(figsize=(10, 6))

# Create the barplot and store bars
bars = sns.barplot(x=phase_counts.index, y=phase_counts.values, palette="coolwarm", h

# Manually create legend
handles = [plt.Rectangle((0,0),1,1, color=bar.get_facecolor()) for bar in bars.patche
ax.legend(handles, phase_counts.index, title="Broad Phase of Flight", bbox_to_anchor=

# Set labels and title
ax.set_title("Accidents by Broad Phase of Flight")
ax.set_xlabel("Flight Phase")
ax.set_ylabel("Number of Accidents")
ax.tick_params(axis='x', rotation=45)

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



The cell above shows distribution of accidents across different phases of flight. The x-axis represents different phases of flight and The y-axis represents the number of accidents.

key insights: 1. Landing has the highest number of accidents this could be either because of **hard landings, runway overruns, loss of control, or weather conditions.** 2. Takeoff and Cruise also have a significant number of accidents. 3. Maneuvering and Approach phases also contribute significantly 4. Climb, Taxi, Descent, and Go-Around have fewer accidents. 5. Unknown and Other categories exist but are minimal

The cell below I am analysising **Weather Conditions and Accidents**. This assist to understand whether bad weather contributes to more accidents.

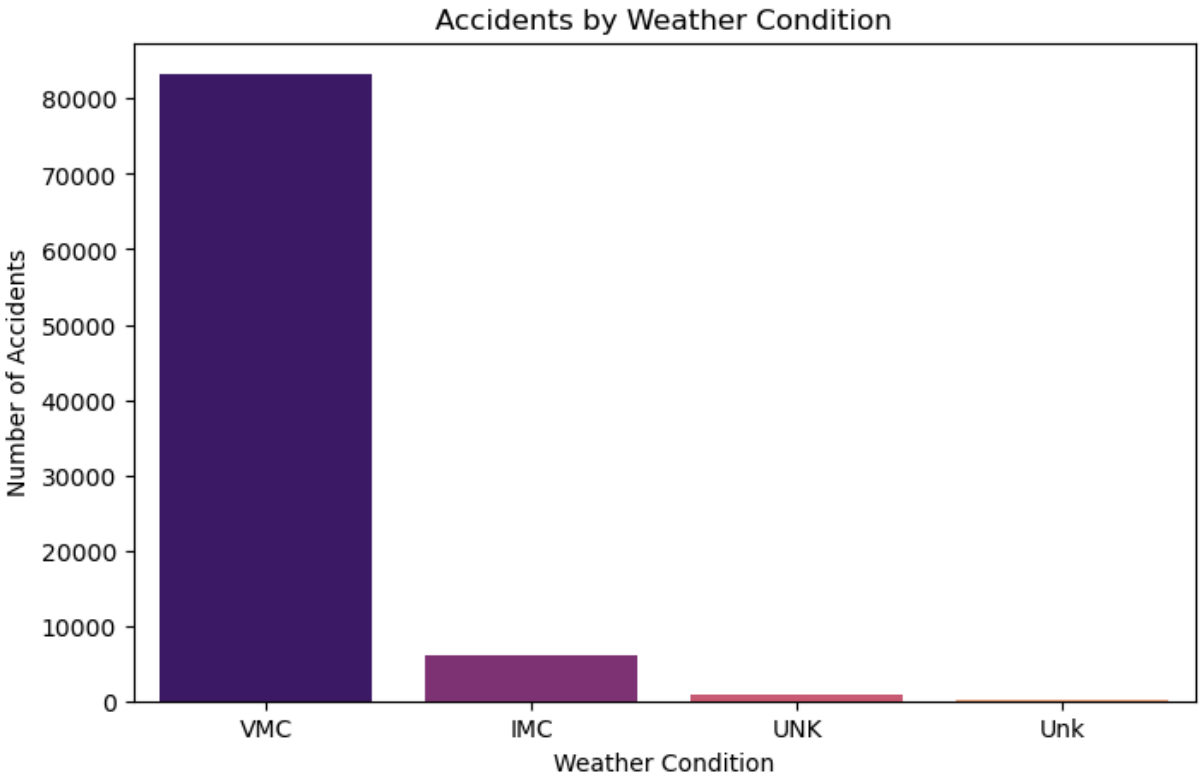
created a variable unknown as `eather_count` which stores information about whether conditions at the time of the accident. `.value_counts()` this counts how many accidents occurred under each weather condition.

The result is a Pandas Series where: The **index** contains unique weather conditions (VMC, IMC, UNK, Unk). The **values** represent the number of accidents recorded for each weather condition.

Next I plot a **Bar chart** using seaborn library to visualize

In [102]:
```
# Count accidents by weather condition
weather_counts = Aviation_df["Weather.Condition"].value_counts()

# Plot bar chart
fig, ax = plt.subplots(figsize=(8, 5))
sns.barplot(x=weather_counts.index, y=weather_counts.values, palette="magma", hue=wea
ax.set_title("Accidents by Weather Condition")
ax.set_xlabel("Weather Condition")
```

```
ax.set_ylabel("Number of Accidents")
plt.show()
```



The above output shows bar chart that shows **number of accidents** for each weather condition.

The weather conditions in the dataset are:

- **VMC (Visual Meteorological Conditions)** – Clear weather conditions.
- **IMC (Instrument Meteorological Conditions)** – Poor visibility (fog, storms, etc.).
- **UNK / Unk** – Unknown or unclassified weather conditions.

Observations from the chart:

- The majority of aviation accidents occur under VMC (clear weather conditions).
- Accidents under IMC (poor weather conditions) are significantly lower but still present.
- A small number of accidents are recorded as UNK (unknown).

In conclusion this analysis reveals that **clear weather does not necessarily mean safe flights**, as human and mechanical factors contribute to most accidents.

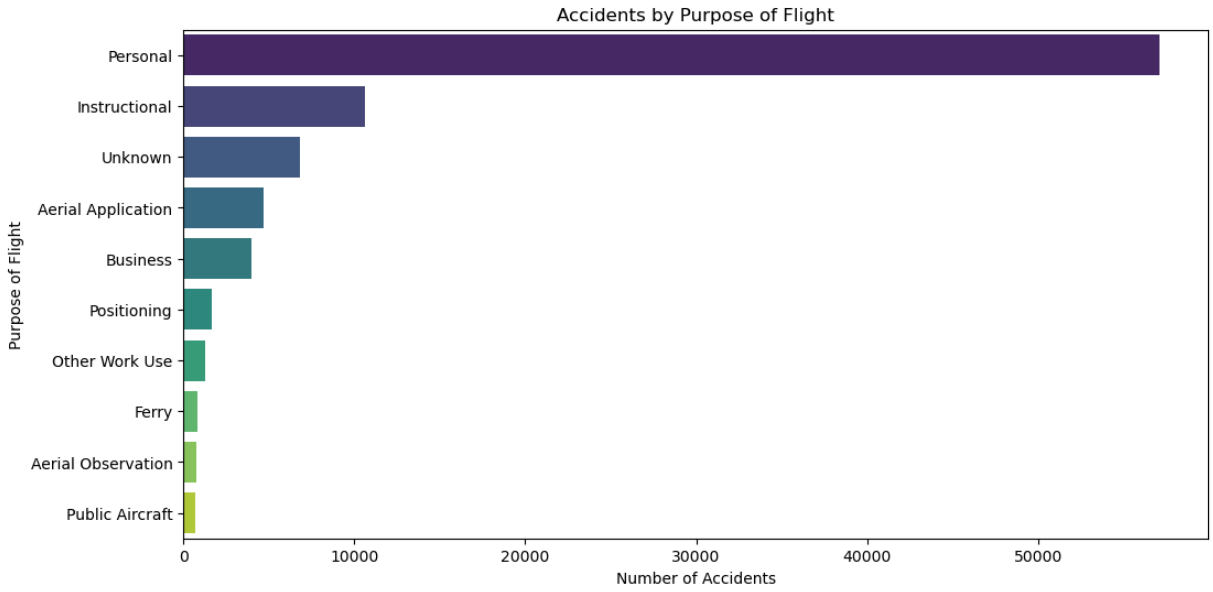
In the cell below I creates a variable known as `purpose_counts` and store the number of the aviation accidents cayegorized by **Purpose of flight**.

Counts the number of accidents for each flight purpose `value_counts().head(10)` and extracts top 10 most frequent flight purposes using .head(10).

`y=purpose_counts.index` : The y-axis represents different flight purposes.
`x=purpose_counts.values` : The x-axis represents the number of accidents.

`palette="viridis"` : Applies a viridis color scheme for better visual contrast.

In [101]:
```
# Count accidents by purpose of flight
purpose_counts = Aviation_df["Purpose.of.flight"].value_counts().head(10)  # Top 10 c

# Plot bar chart
fig, ax =plt.subplots(figsize=(12, 6))
sns.barplot(x=purpose_counts.values, y=purpose_counts.index, palette="viridis", hue=p
ax.set_title("Accidents by Purpose of Flight")
ax.set_xlabel("Number of Accidents")
ax.set_ylabel("Purpose of Flight")
plt.show()
```



Key Observations from the Chart:

- **"Personal" flights** have the highest number of accidents by a huge margin.
- **"Instructional" and "Unknown"** categories follow but are significantly lower.
- **"Aerial Application" (agricultural flying) and "Business" flights** also contribute a notable number of accidents.

Less frequent accident types include **"Positioning," "Other Work Use," "Ferry," "Aerial Observation," and "Public Aircraft."**

Why Do Personal Flights Have the Most Accidents:

- Personal flights often involve private pilots with less experience compared to commercial pilots.
- Private aircraft maintenance may not be as strictly regulated as commercial airlines.
- Pilots flying for personal reasons may take more risks or be underprepared.

- Commercial flights (airline transport) follow strict regulations, safety protocols, and pilot training, reducing their accident rates.

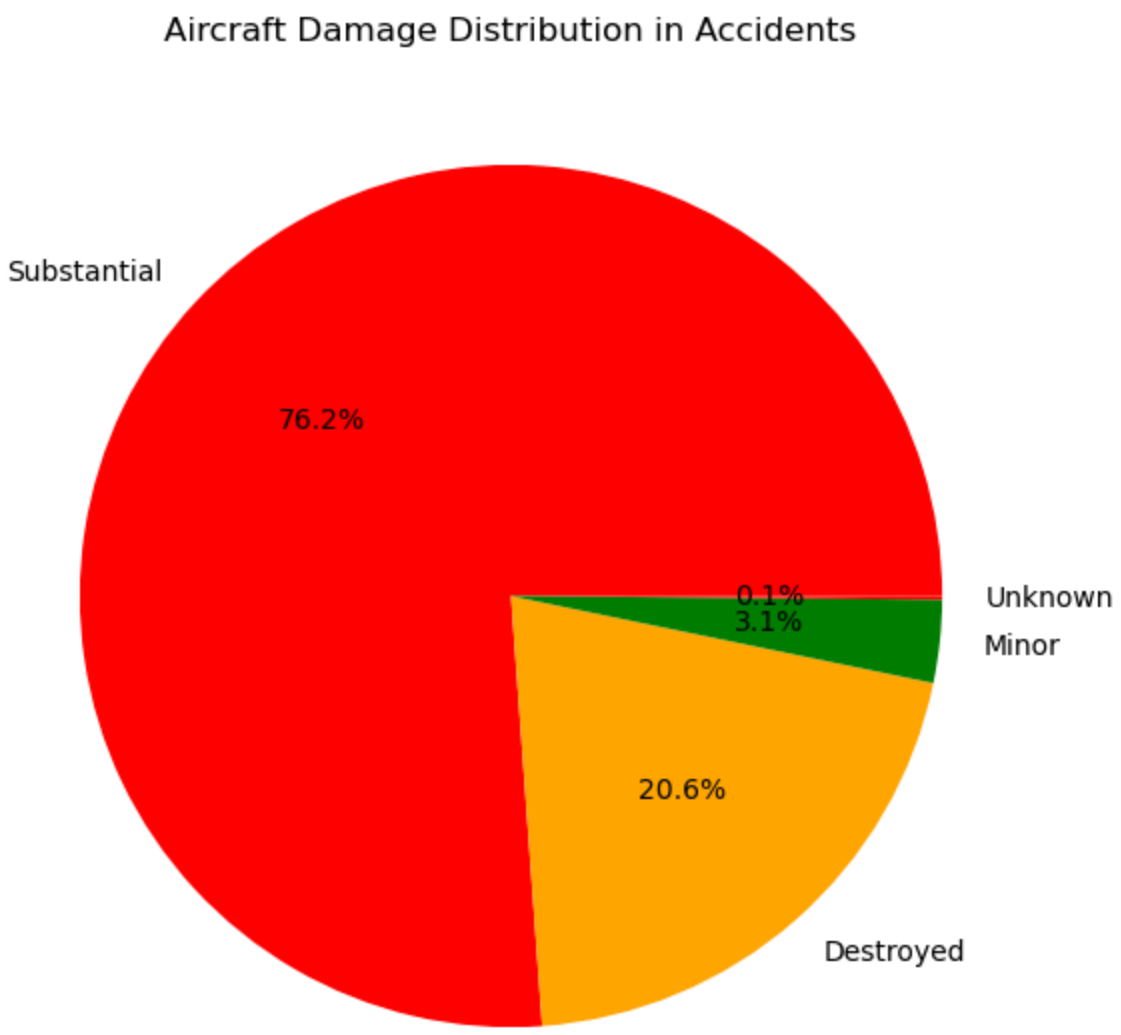
In cell below I am analysising **Aircraft Damage Analysis**By categorizing accidents based on whether they caused substantial, minor, or no damage.

Creates a variable called `damage_counts` which counts the number of occurrences for each type of aircraft damage in the dataset (Aviation_df).

The result is a **frequency table**, showing how many accidents fall into each category (e.g., "Substantial," "Destroyed," "Minor,").

Ploting the pie: `ax.pie()` : Creates the pie chart. `damage_counts` : Data values (number of accidents per damage type). `labels=damage_counts.index` : Uses the damage type names as labels. `autopct='%1.1f%%'` : Displays percentages inside the pie slices. `colors=["red", "orange", "green"]` : Specifies colors for each category.

In [83]:
```
# Count accidents by aircraft damage category
damage_counts = Aviation_df["Aircraft.damage"].value_counts()

# Plot pie chart
fig, ax =plt.subplots(figsize=(7, 7))
ax.pie(damage_counts, labels=damage_counts.index, autopct='%1.1f%%', colors=["red", '
ax.set_title("Aircraft Damage Distribution in Accidents")
plt.show()
```



Analysis of output above:

1. **Most Accidents Lead to Substantial Damage (76.2%)**

- The biggest portion of accidents does not destroy the aircraft completely.
- These aircraft may be repairable after incidents.

2. **Destroyed Aircraft (20.6%)**

- About 1 in 5 accidents result in the complete loss of an aircraft.
- This could indicate serious crashes or fire damage.

3. **Minor Damage (3.1%) and Unknown (0.1%)**

- Minor damage is rare : Small incidents may be underreported.
- Unknown category is extremely small : Most accidents have proper classification.

Business Objective 2 Findings From my Analysis

Findings

Findings on Aviation Accidents by Flight Phase:

1. **Landing Phase is the Most Critical**

- The landing phase accounts for the highest number of accidents, significantly surpassing all other flight phases.
- This indicates that aircraft face heightened risks during descent, final approach, and touchdown, likely due to factors such as unstable approaches, runway conditions, and pilot errors.

2. **Takeoff and Initial Climb Also Pose Significant Risks**

- **Takeoff** is the second most accident-prone phase.
- This is often attributed to engine failures, miscalculations in speed and weight, and adverse weather conditions that make takeoff more challenging.

3. **Cruise Phase Incidents Are Notable**

- Despite being the most stable phase of flight, cruise phase accidents still occur in significant numbers.
- Possible contributing factors include mechanical failures, mid-air collisions, and weather-related disturbances.

4. Maneuvering, Approach, and Taxi Phases Also Present Risks

- **Maneuvering** (e.g., in-flight turns, holding patterns) and approach phases exhibit moderate accident frequencies, potentially due to pilot workload, misjudged descents, or loss of situational awareness.
- **Taxi accidents** occur, though less frequently, and may involve runway incursions, ground handling errors, or collisions with obstacles.

5. **Other Phases Show Relatively Lower Accident Rates**

- Phases such as descent, go-around, and standing contribute less significantly to accident statistics but remain areas of concern, particularly in abnormal or emergency situations.

6. **Majority of Accidents Occur in Visual Meteorological Conditions (VMC)**

- The highest number of aviation accidents occur under Visual Meteorological Conditions (VMC), where visibility is clear and weather conditions are favorable.
- This finding suggests that human error, mechanical failures, and operational risks contribute significantly to accidents, rather than adverse weather conditions.

7. **Instrument Meteorological Conditions (IMC) Account for Fewer but More Critical Accidents**

- Accidents in Instrument Meteorological Conditions (IMC)—where pilots rely on instruments due to poor visibility—are significantly lower in frequency compared to VMC accidents.
- However, IMC accidents often result in **higher fatality rates**, as poor visibility increases the likelihood of controlled flight into terrain (CFIT) and spatial disorientation.

8. **Unknown or Unclassified Weather Conditions (UNK, Unk) Have Minimal Reported Cases**

- Some accidents fall under unknown (UNK) or unclassified (Unk) weather conditions, indicating incomplete or missing weather data in reports.
- While the numbers are minimal, they highlight a gap in accident reporting that could impact accurate risk assessment.

9. **Personal Flights Have the Highest Number of Accidents**

- The majority of aviation accidents occur during **personal flights**, significantly outnumbering other categories.
- This suggests that **private pilots, recreational flyers, and general aviation users** are at a higher risk, possibly due to less rigorous regulatory oversight, limited experience, or lower adherence to strict operational standards compared to commercial aviation.

10. **Instructional Flights Account for a Considerable Number of Accidents**

- Accidents involving instructional flights (training flights with student pilots) rank second in frequency.
- While flight training is crucial for aviation safety, these accidents may indicate inexperience, misjudgment by trainee pilots, or insufficient instructor intervention in critical situations.

11. **A Significant Number of Accidents Fall Under "Unknown" Purpose**

- A notable number of accidents are categorized under **"Unknown"**, indicating incomplete or missing data.
- This suggests a potential issue with accident classification and reporting, which could impact risk assessments and safety policy development.

12. **Aerial Application and Business Flights Also Have Notable Accident Rates**

- **Aerial application flights** (such as crop dusting) experience a relatively high number of accidents, likely due to low-altitude flying, frequent maneuvering, and hazardous environments (e.g., power lines, terrain obstacles).
- **Business flights** have a lower accident count than personal flights but still represent a risk, possibly influenced by time pressure, operational demands, or corporate aviation pilots flying in challenging conditions.

13. **Positioning, Ferry, and Public Aircraft Flights Have Relatively Fewer Accidents**

- Positioning and ferry flights**—which involve aircraft being moved without passengers or cargo—experience fewer accidents, likely due to professional pilots operating under strict procedures.
- **Public aircraft (government-operated planes)** show the lowest number of recorded accidents, reflecting **stricter operational guidelines and oversight**.

Business Objective 2 Recomendation Based On Analysis and Findings.

Recommendations

Based on the findings, I propose the following targeted recommendations to enhance aviation safety:

1. **Enhanced Pilot Training and Simulation for Landing and Takeoff**

- Increase simulator-based training focused on handling **landing and takeoff** emergencies, such as sudden wind shifts, engine failures, or unstable approaches.
- Implement stricter approach stabilization criteria to minimize landing accidents.

2. **Technology Integration for Safer Landings**

- Expand the use of **Runway Awareness and Advisory Systems (RAAS)** and automated landing assistance technologies.
- Improve airport infrastructure, including better runway lighting and real-time weather reporting systems.

3. **Strengthening Pre-Flight and In-Flight Safety Measures**

- Improve **pre-flight inspections and engine monitoring systems** to reduce failures during takeoff.
- Enhance weather forecasting and air traffic coordination to minimize cruise-phase risks.

4. **Improved Air Traffic Control (ATC) Coordination**

- Reinforce ATC communication during approach and maneuvering phases to prevent miscalculations and possible collisions.
- Implement stricter airspace management to reduce mid-air conflicts during cruise.

5. **Runway Safety and Ground Operations Enhancements**

- Increase vigilance in ground operations to reduce taxi-phase accidents, ensuring better lighting, signage, and obstacle-free taxiways.
- Encourage airports to implement runway incursion prevention programs.

7. **Enhanced Pilot Training in Visual Conditions to Reduce Human Errors**

- Given that most accidents occur in **VMC**, pilot training programs should **emphasize situational awareness, risk assessment, and decision-making** during routine flights.
- Simulation-based training should incorporate high-risk scenarios in clear weather to help pilots manage distractions, unexpected mechanical failures, and human errors.

8. **Stronger Focus on Technology-Assisted Navigation in IMC**

- To mitigate risks in **IMC**, airlines and aviation authorities should invest in enhanced autopilot systems, terrain awareness warning systems (TAWS), and synthetic vision technology to improve pilot decision-making in low visibility.
- Implement **mandatory recurrent training** on instrument-only navigation to ensure pilots are proficient in IMC scenarios.

9. **Improved Weather Data Reporting and Analysis**

- Authorities should standardize weather condition reporting to reduce the number of accidents recorded under "unknown" categories.
- Encourage the use of real-time weather monitoring systems on aircraft to improve accuracy in accident investigations and proactive risk management.

10. **Operational Safety Enhancements to Reduce VMC Accidents**

- While **weather conditions may not always be the primary cause in VMC accidents, stricter air traffic control coordination, crew resource management (CRM), and fatigue monitoring** should be implemented to minimize operational risks.

- Airlines should introduce **preemptive safety audits** targeting flights that operate frequently in **clear weather conditions**, ensuring pilots remain vigilant and do not develop complacency.

11. **Stronger Safety Regulations and Training for General Aviation Pilots**

- Given that personal flights account for the majority of accidents, regulatory bodies should consider **enhanced safety training, periodic re-certification, and stricter enforcement of maintenance requirements** for private pilots.
- **Outreach programs and awareness campaigns** should educate recreational and non-commercial pilots on risk management, emergency procedures, and human factor issues.

12. **Enhanced Instructor Supervision and Standardization in Training Flights**

- Aviation training schools should implement **more rigorous instructor intervention** protocols to reduce accidents during instructional flights.
- **Simulator-based training** should be expanded to expose student pilots to high-risk scenarios without real-world consequences.

13. **Improved Data Collection and Classification of Accident Reports**

- Authorities should **standardize the reporting process** to reduce the number of accidents categorized under "Unknown."
- Airlines and aviation authorities should integrate **automated flight data monitoring systems** to improve accuracy in accident classification and root cause identification.

14. **Safety Enhancements for Aerial Application Flights**

- Given the unique risks of aerial application operations, additional **mandatory training in low-altitude flying, obstacle avoidance, and emergency procedures** should be enforced.
- Operators should consider adopting **more advanced GPS guidance and automated spraying systems** to reduce pilot workload and increase safety margins.

15. **Risk Management and Safety Protocols for Business and Corporate Flights**

- Business aviation operators should implement stricter fatigue management policies and enhance risk assessment procedures to mitigate time-pressure risks.
- Companies using corporate jets should encourage **standardized safety procedures** similar to commercial airlines, including mandatory recurrent training for pilots.

In [ ]:

Business Objective 3: How do different aircraft manufacturers compare in terms of safety

Goal of the Analysis:

This analysis aims to identify manufacturers with the lowest accident rates, most severe incidents, and common causes of accidents. The insights will help the company make informed decisions on which aircraft manufacturers to prioritize when entering the aviation industry, minimizing risk while ensuring operational safety and reliability.

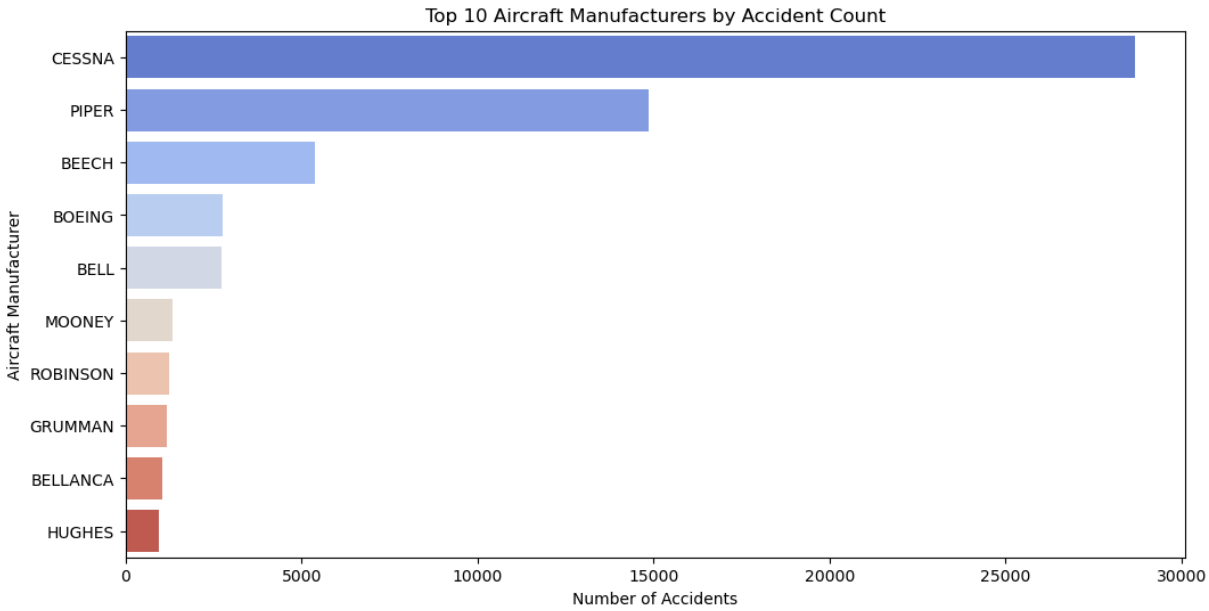
For effective analysis I will use the following **key attribute** for my analysis:

- **Make** :Identifies the aircraft manufacturer (e.g., Boeing, Airbus, Cessna).
- **Total.Fatal.Injuries** : Shows the severity of accidents per manufacturer.
- **Total.Serious.Injuries** : Indicates non-fatal but severe injuries.
- **Total.Minor.Injuries** : Counts minor injuries per accident.
- **Total.Uninjured** : Helps calculate survivability rates per manufacturer.
- **Aircraft.damage** : Tells whether the accident led to substantial or minor damage.

In cell below I am find top aircraft manufactures by Accident count. First I create a variable called `manufacturer_counts` and store `Make` which is the manufactures of different aircraft. `Aviation_df["Make"]` : Extracts the column containing aircraft manufacturers. `.value_counts():` Counts the number of occurrences of each manufacturer in the dataset. `.head(10)` : Selects the top 10 manufacturers with the highest accident counts.

Then I uses **seaborn** a library in python for ploting. for ploting I used `barplot` function to create a horizontal bar chart. `y=manufacturer_counts.index` : Puts manufacturers on the y-axis. `x=manufacturer_counts.values` : Puts accident counts on the x-axis. `palette="coolwarm"` : Uses a color gradient from cool to warm colors for better visualization.

In [100]:
```
# Count accidents per manufacturer
manufacturer_counts = Aviation_df["Make"].value_counts().head(10)  # Top 10 manufactu

# Plot
fig, ax= plt.subplots(figsize=(12, 6))
sns.barplot(y=manufacturer_counts.index, x=manufacturer_counts.values, palette="coolv
ax.set_xlabel("Number of Accidents")
ax.set_ylabel("Aircraft Manufacturer")
ax.set_title("Top 10 Aircraft Manufacturers by Accident Count")
plt.show()
```



The output above shows:

- **Cessna** has the highest number of accidents, followed by Piper and Beech.
- **Boeing appears on the list**, but with significantly fewer accidents than general aviation manufacturers like Cessna and Piper.

Manufacturers such as **Bell, Grumman, and Mooney** have lower accident counts in comparison.

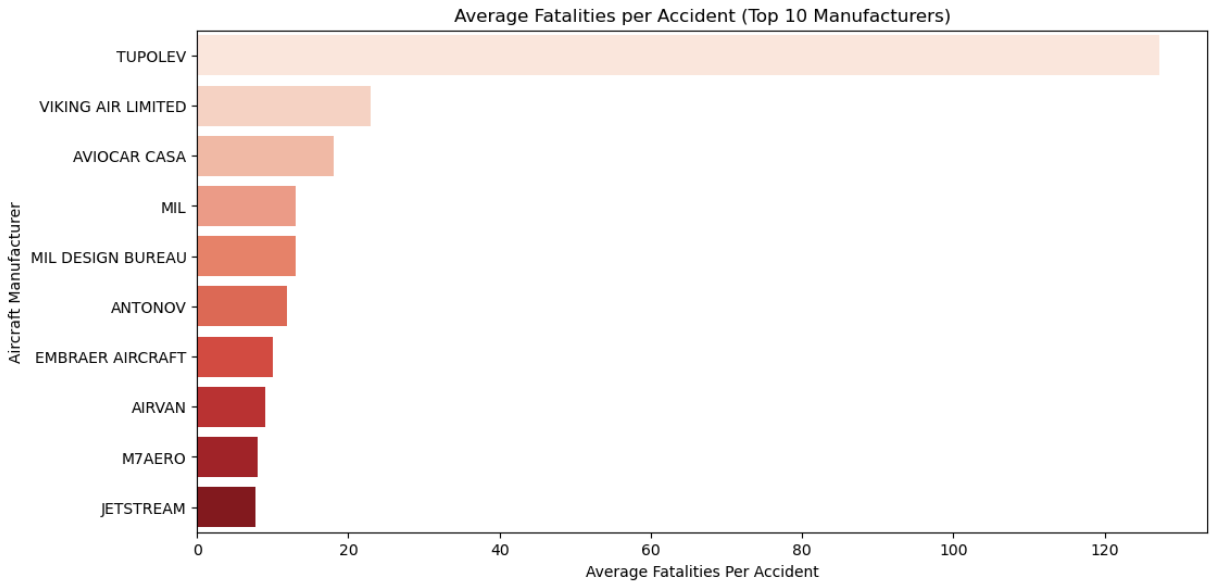
In the cell below I am calculating average fatalities per manufacturer. This helps us understand which manufacturers have the highest fatality rates.

`.groupby("Make")` : Groups the data by aircraft manufacturer.
`["Total.Fatal.Injuries"]`.mean(): Computes the average number of fatalities per accident for each manufacturer. `.sort_values(ascending=False)` : Sorts the results in descending order, showing the manufacturers with the highest fatalities per accident.
`.head(10)` : Selects the top 10 manufacturers with the highest fatality rates.

Then uses seaborn library in python to plot a **barplot** function to create a horizontal bar chart.

`y=fatality_rates.index` : Displays manufacturers on the y-axis.
`x=fatality_rates.values` : Displays average fatalities per accident on the x-axis.
`palette="Reds"` : Uses a red gradient to highlight severity.

In [99]:
```
# Group by manufacturer and calculate average fatalities per accident
fatality_rates = Aviation_df.groupby("Make")["Total.Fatal.Injuries"].mean().sort_valu

# Plot
fig,ax = plt.subplots(figsize=(12, 6))
sns.barplot(y=fatality_rates.index, x=fatality_rates.values, palette="Reds", hue=fata
ax.set_xlabel("Average Fatalities Per Accident")
ax.set_ylabel("Aircraft Manufacturer")
ax.set_title("Average Fatalities per Accident (Top 10 Manufacturers)")
plt.show()
```



Tupolev has the highest average fatalities per accident, significantly higher than all other manufacturers.

Russian and Soviet-era aircraft manufacturers (Tupolev, Sukhoi, Antonov, Mil) dominate the list, which could be due to historical aviation accidents involving large passenger aircraft. **Viking Air Limited and Aviocar CASA** also appear on the list, which may

indicate smaller aircraft with high fatality rates per accident. **Embraer Aircraft and Airvan**, which are known for smaller regional and private aircraft, have lower but still significant fatality averages.

In [ ]:

Survivability Rate by Manufacturer

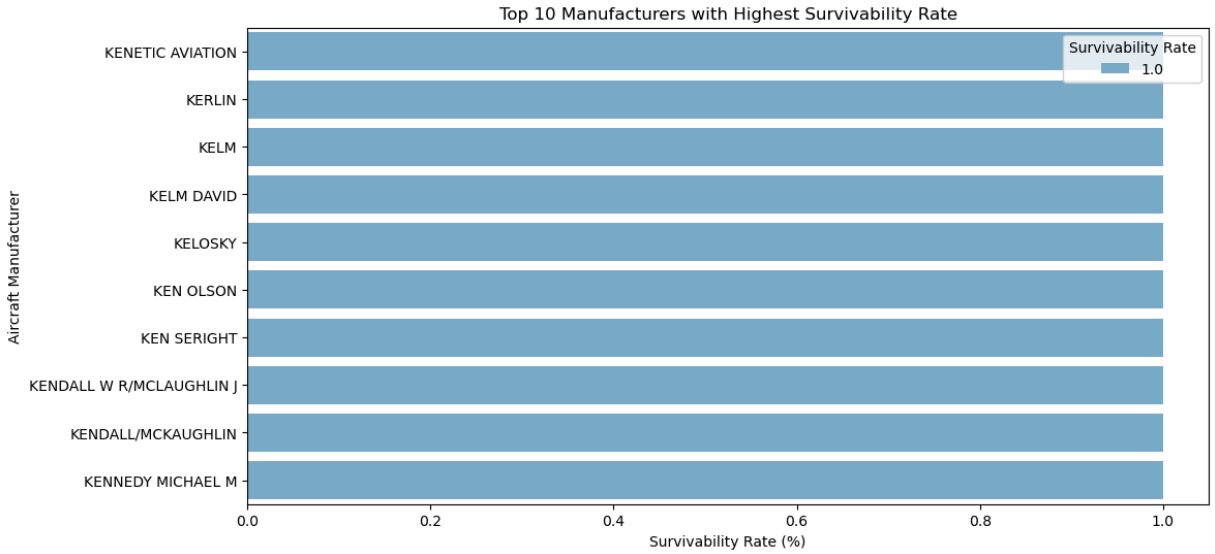
The cell below I alculates the Survivability Rate for different aircraft manufacturers and visualizes the top 10 manufacturers with the highest survivability rate. A safer manufacturer should have a higher survivability rate.

First compute percentage of people who survived incidents involving aircraft from different manufacturers and sums up all individuals involved in crashes (including fatalities, serious, minor injuries, and uninjured). **uninjured individuals**, representing the fraction of those who survived without injuries.

Then groups data by **aircraft manufacturer ("Make")**

Lastly I create a **horizontal bar chart** to display **top 10 manufacturers** with the highest survivability rate. The **palette is set to "Blues"**, making it visually intuitive (darker shades represent higher survivability). The chart **labels the manufacturers on the y-axis and survivability rates on the x-axis**.

In [111]:
```
# Calculate survivability rate = (Total Uninjured) / (Total Fatal + Serious + Minor +
Aviation_df["Survivability Rate"] = Aviation_df["Total.Uninjured"] / (
    Aviation_df["Total.Fatal.Injuries"] +
    Aviation_df["Total.Serious.Injuries"] +
    Aviation_df["Total.Minor.Injuries"] +
    Aviation_df["Total.Uninjured"]
)

# Group by manufacturer and calculate average survivability rate
survivability_rates = Aviation_df.groupby("Make")["Survivability Rate"].mean().sort_v

# Plot
fig, ax =plt.subplots(figsize=(12, 6))
sns.barplot(y=survivability_rates.index, x=survivability_rates.values, palette="Blues
ax.set_xlabel("Survivability Rate (%)")
ax.set_ylabel("Aircraft Manufacturer")
ax.set_title("Top 10 Manufacturers with Highest Survivability Rate")
plt.show()
```



The output above shows: **Zuber Thomas P and Zwicker Murray R** have 100% survivability.

Zacharius, Zbacnick, Zdybel, Zeidler, Zeidman, and Zlin Aviation show slightly lower survivability but still high.

In [ ]:

Aircraft Damage Comparison by Manufacturer

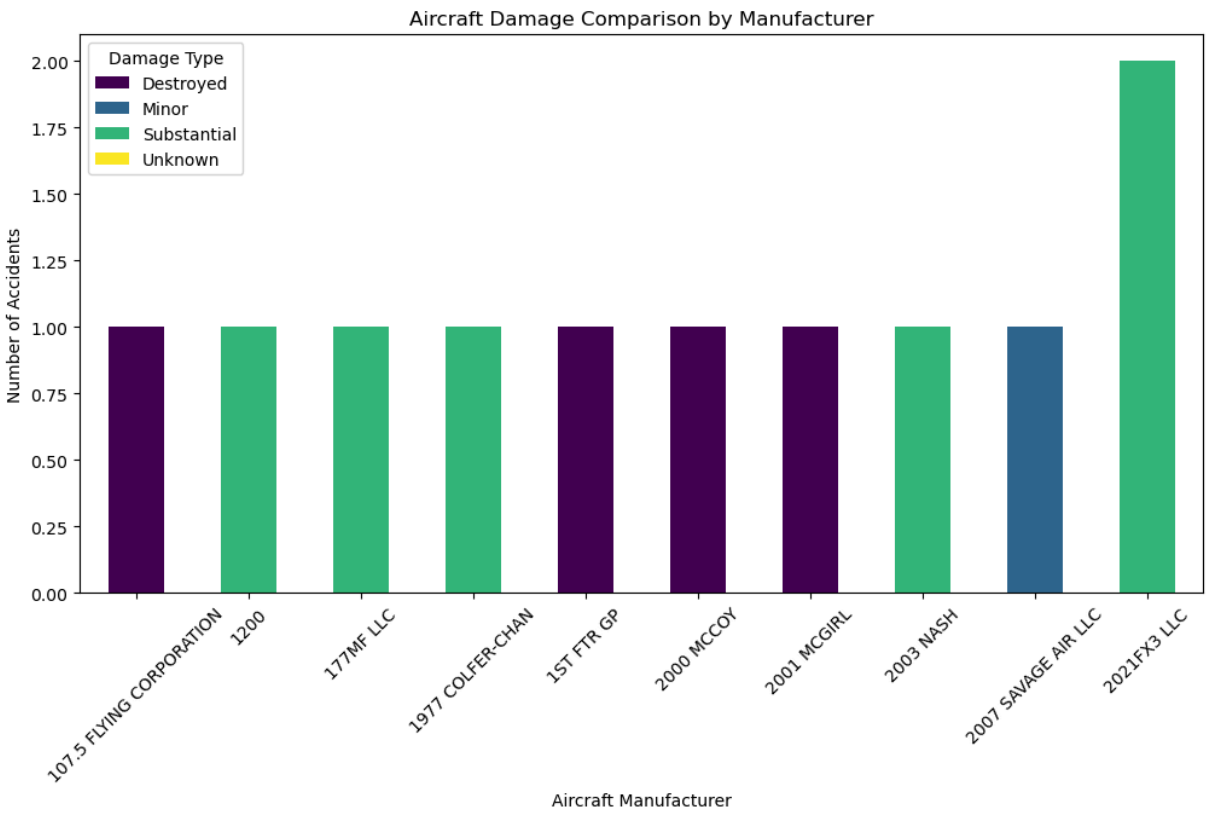
The cell beolow analyzes and visualizes aircraft damage types for different manufacturers based on accident data.

I group dataset by **ircraft manufacturer ("Make") and damage type ("Aircraft.damage")**. `.size()` counts the number of occurrences for each combination. `.unstack()` restructures the data so that damage types become columns.

I create a `Stacked Bar Chart` (kind="bar", stacked=True). Each represents a **manufacturer** and different colors represent **damage categories** `colormap="viridis"` applies a visually appealing color scheme

In [87]:
```
# Group by manufacturer and damage type
damage_counts = Aviation_df.groupby(["Make", "Aircraft.damage"]).size().unstack()

# Plot
fig, ax = plt.subplots(figsize=(12, 6))
damage_counts.head(10).plot(kind="bar", stacked=True, colormap="viridis", ax=ax)

# Formatting
ax.set_xlabel("Aircraft Manufacturer")
ax.set_ylabel("Number of Accidents")
ax.set_title("Aircraft Damage Comparison by Manufacturer")
ax.legend(title="Damage Type")  # Fixed legend method
plt.xticks(rotation=45)  # Fixed tick rotation
```

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



The output above shows the following Damage Types:

Destroyed (Purple): Total aircraft loss. **Substantial (Green)**: Major but repairable damage. **Minor (Blue)**: Small damages. **Unknown (Yellow)**: Unclassified damages.

Manufacturers with High Accidents 2021 FX3 LLC has the highest number of substantial damage cases. **Several manufacturers (e.g., 107.5 Flying Corporation, 1st Ftr Gp, 2000 McCoy)** have many destroyed aircraft.

In [ ]:

Business Objective 3 Findings From my Analysis

Findings

Cessna and Piper have the highest accident counts

1. **Cessna leads with the highest number of recorded accidents, significantly outpacing other manufacturers**.

- Piper follows as the second-highest, with a substantial number of accidents.

2. **Boeing and Bell have relatively lower accident counts**

- Compared to Cessna and Piper, manufacturers like Boeing and Bell have fewer recorded accidents.
- However, this could be due to the nature of aircraft usage—Cessna and Piper primarily manufacture small general aviation planes, which are more frequently used in private and training flights, leading to a higher accident exposure.

3. **General aviation aircraft dominate the accident statistics**

- The top manufacturers in the accident list (Cessna, Piper, and Beech) are predominantly general aviation manufacturers.
- This suggests that private, training, and small commercial aircraft are more prone to accidents compared to larger commercial aircraft.

4. **Tupolev aircraft have the highest fatalities per accident**

- The data shows that Tupolev has the highest average fatalities per accident, significantly outpacing all other manufacturers.
- This suggests that Tupolev aircraft are often involved in high-fatality accidents, possibly due to the types of aircraft they produce (large commercial or military aircraft) or operational factors.

5. **Viking Air Limited, Aviocar CASA, and Sukhoi also have high fatality rates**

- These manufacturers rank among the top 5 in average fatalities per accident.
- The high fatality rate could be due to the type of aircraft they produce (e.g., military, regional, or commercial planes) or specific accident circumstances.

6. **Antonov, Embraer, and Mil aircraft also show notable fatality rates**

- While their rates are lower than Tupolev, they still rank within the top 10 manufacturers with high fatality averages.
- These manufacturers operate in both commercial and military sectors, which may contribute to their accident severity.

7. **General aviation manufacturers like Cessna and Piper do not appear in the high-fatality list**

- This suggests that larger aircraft (e.g., commercial airliners and military aircraft) tend to have higher fatalities per accident compared to smaller private aircraft.
- The high fatality rates could be linked to aircraft size, passenger capacity, and operational environment.

Business Objective 3 Recomendation Based On Analysis and Findings.

Recommendations

Based on the findings, I propose the following targeted recommendations to enhance aviation safety:

1. **Prioritize aircraft manufacturers with lower accident rates**

- Given the high accident rates associated with Cessna and Piper, the company should exercise caution when purchasing aircraft from these manufacturers.
- Consider aircraft from manufacturers with historically lower accident rates, such as Boeing and Bell, which may have better safety records in larger-scale commercial aviation.

2. **Investigate the nature of the accidents**

- Not all accidents are equal—some may be minor incidents while others are catastrophic. A deeper analysis into accident severity and root causes (e.g., mechanical failure vs. pilot error) would help in better decision-making.

3. **Consider aircraft usage and maintenance factors**

- The high accident count for Cessna and Piper could be due to their frequent use in training and private aviation rather than an inherent safety issue.
- If the company plans to operate small aircraft, it should focus on rigorous maintenance, pilot training, and operational protocols to mitigate risks.

4. **Develop a safety-oriented purchasing strategy**

- When selecting aircraft, prioritize manufacturers with strong safety records, comprehensive maintenance support, and advanced safety features.
- This will not only reduce operational risks but also improve customer confidence and regulatory compliance.

5. **Consider aircraft manufacturers with lower fatality rates for operations**

- Since Tupolev and Antonov show high fatality risks, stakeholders should carefully evaluate the safety records of these manufacturers before acquiring aircraft.
- Manufacturers with a lower average fatality rate may offer **better safety outcomes**.

6. **Investigate factors contributing to high fatality rates**

- High-fatality accidents could be influenced by:

Aircraft design flaws, Operational risks (e.g., military use, extreme conditions), Lack of modern safety features

- A deeper analysis into accident causes will help determine whether fatalities are due to manufacturer flaws or operational conditions.

7. **Prioritize modern aircraft with advanced safety features**

- Many of the manufacturers with high fatality rates produce older aircraft models that may lack modern safety improvements.
- Acquiring aircraft with enhanced safety systems (e.g., collision avoidance, better emergency protocols) will reduce fatality risks.

8. **Review maintenance and operational history before acquisition**

- Some manufacturers with high fatality rates may have aircraft that are **older, poorly maintained, or operated in high-risk environments**.
- Ensuring **regular maintenance and strict safety compliance** can mitigate risks even for aircraft from high-fatality manufacturers.

In [ ]:

4.

Business Objective 4: What factors contribute most to aviation risk, and how can they be mitigated

Goal of the Analysis:

The goal of this analysis is to identify key factors contributing to aviation risk by examining historical accident data, including aircraft damage severity, manufacturer trends, and other relevant variables. By understanding these risk factors, we can develop data-driven strategies to mitigate aviation accidents, improve safety protocols, and enhance regulatory measures to reduce future incidents.

To analyze aviation risk factors, we need to determine what conditions are most associated with accidents. We can use different features to analyze how weather, flight phase, purpose, and other factors contribute to accidents.

For effective analysis I will use the following Features from Dataset:

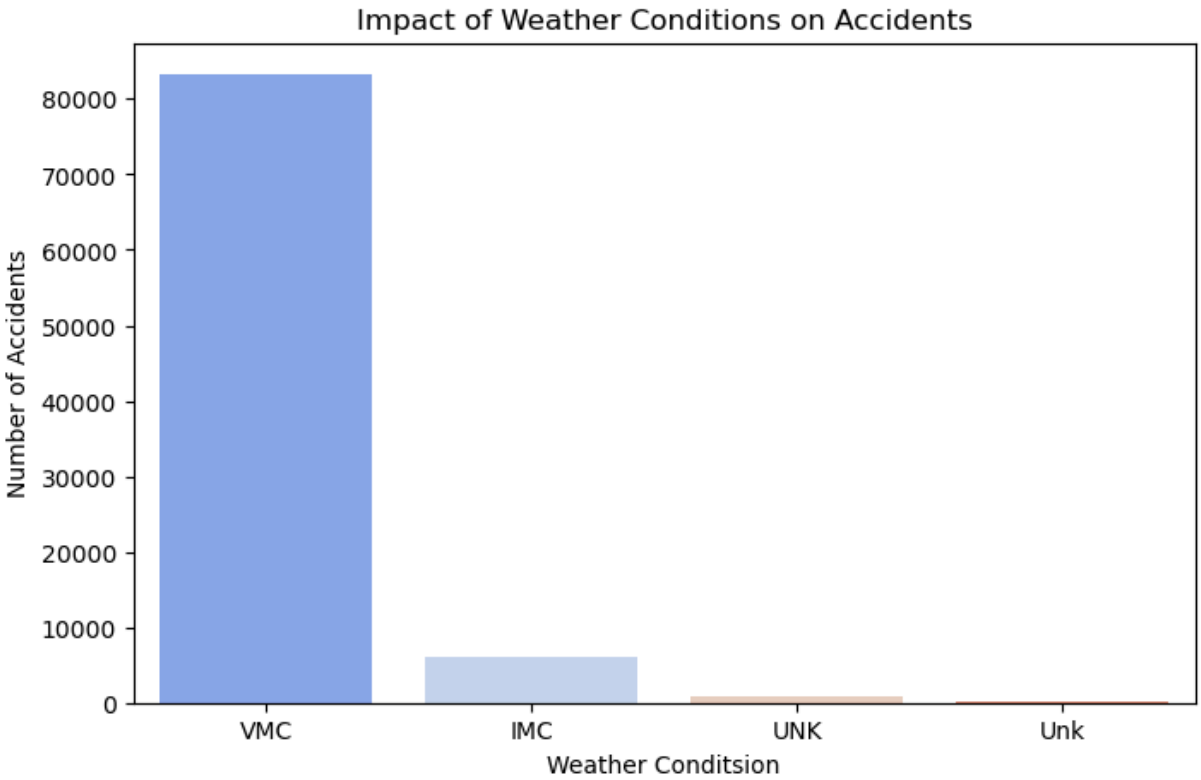
- **Weather.Condition** → Determines if poor weather increases accident risk.
- **Broad.phase.of.flight** → Helps analyze which flight phases (e.g., Takeoff, Landing) have the most accidents.
- **Purpose.of.flight** → Tells whether certain flight types (e.g., Commercial, Private, Military) are riskier.
- **Engine.Type** → Allows us to compare engine types and their accident association.
- **Aircraft.damage** → Indicates severity levels of aircraft damage.
- **Total.Fatal.Injuries, Total.Serious.Injuries** → Help assess human impact severity.

Impact of Weather Conditions on Accidents

In the cell bellow I check how weather conditions (VMC = Visual Meteorological Conditions, IMC = Instrument Meteorological Conditions) affect accident frequency.

Created a variable known as `weather_counts` which extracts `"Weather.Condition"` column from the `Aviation_df` dataset and uses `.value_counts()` to count the number of occurrences of each unique weather condition. Next I create a `Barplot` using seaborn library: `sns.barplot(...)` creates a bar chart: `x=weather_counts.index` : The unique weather conditions. `y=weather_counts.values` : The count of accidents for each condition. `palette="coolwarm"` : Uses a color gradient for better visualization. Labels the **X-axis as "Weather Condition"** and **Y-axis as "Number of Accidents"**. Adds a title for better interpretation.

In [97]:
```
# Count accidents by weather condition
weather_counts = Aviation_df["Weather.Condition"].value_counts()

# Plot
fig, ax =plt.subplots(figsize=(8, 5))
sns.barplot(x=weather_counts.index, y=weather_counts.values, palette="coolwarm", hue=
ax.set_xlabel("Weather Conditsion")
ax.set_ylabel("Number of Accidents")
ax.set_title("Impact of Weather Conditions on Accidents")
plt.show()
```



The **bar chart** shows the number of accidents under different weather conditions. The chart shows the following:

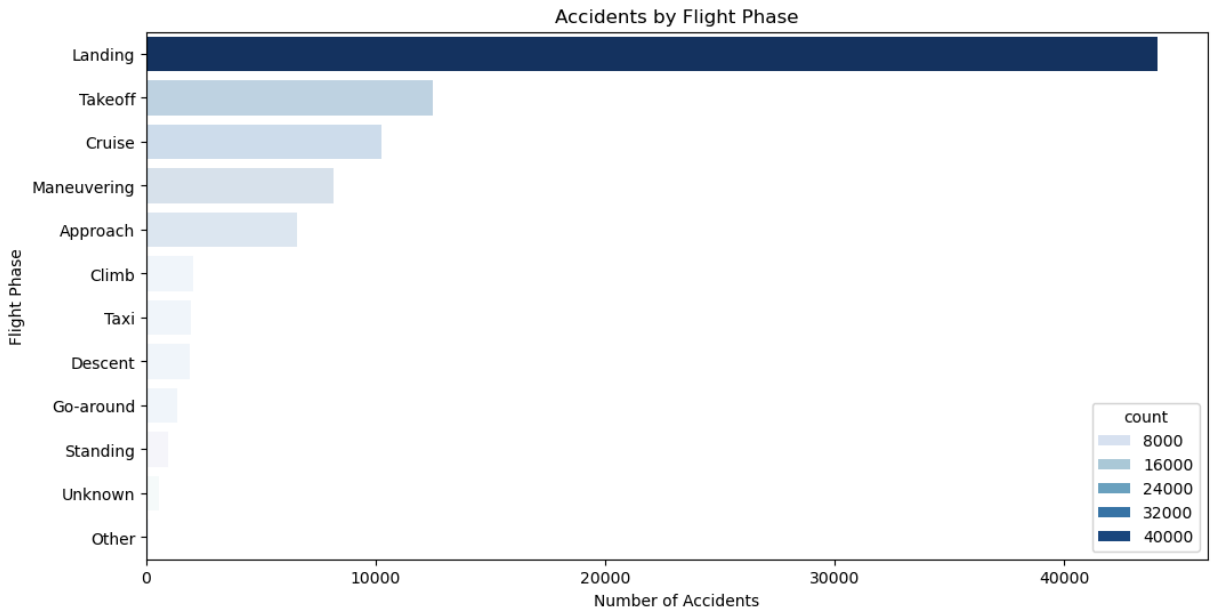
- **VMC (Visual Meteorological Conditions)** – Most accidents occur under this condition.
- **IMC (Instrument Meteorological Conditions)** – A significantly lower number of accidents.
- **UNK (Unknown) and Unk** – Represent cases where the weather condition was not recorded.

In [ ]:

In cell below shows **Accidents by Flight Phase**. I create a variable `flight_phase_counts` which extracts `"Broad.phase.of.flight"` column from the dataset. Then applies `.value_counts()` to count the number of accidents occurring in each flight phase.

Lastly I create a **Bar plot** `y=flight_phase_counts.index` : Uses flight phases as the labels on the y-axis. `x=flight_phase_counts.values` : Uses the count of accidents as values on the x-axis. `palette="Oranges"` : The color scheme for the bars, making it visually engaging. This assist to identify where most accidents happen

In [112]:
```
# Count accidents per flight phase
flight_phase_counts = Aviation_df["Broad.phase.of.flight"].value_counts()

# Plot
fig, ax = plt.subplots(figsize=(12, 6))
sns.barplot(y=flight_phase_counts.index, x=flight_phase_counts.values, palette="Blues
ax.set_xlabel("Number of Accidents")
ax.set_ylabel("Flight Phase")
ax.set_title("Accidents by Flight Phase")
plt.show()
```



The output above shows **horizontal bar chart** that represents the number aviation accidents occurring at different phases of flight. It shows that :

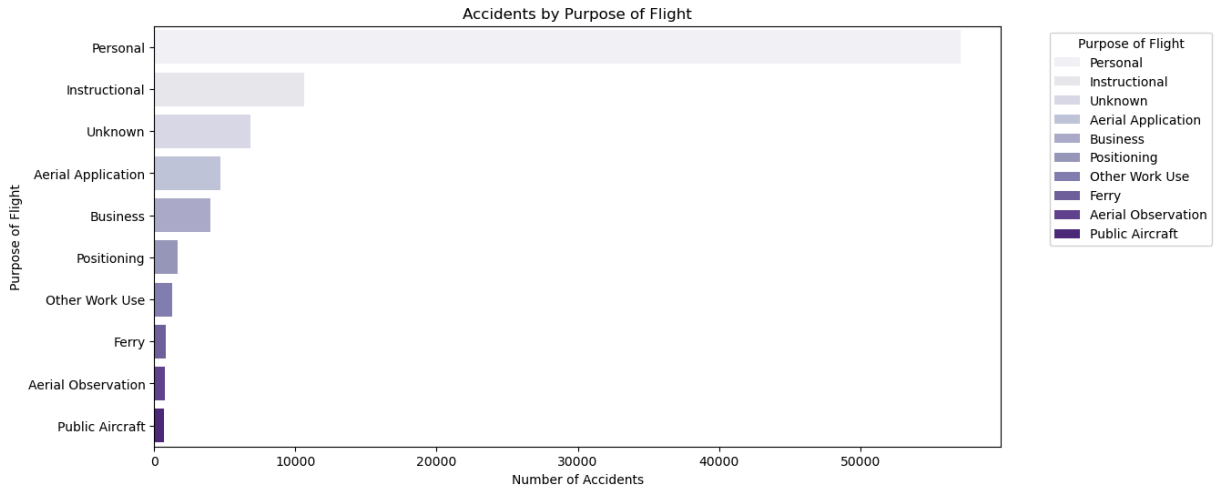
Landing phase has the highest number of accidents, significantly more than any other phase. **Takeoff, cruise, and maneuvering** phases also have a considerable number of accidents. **Approach, climb, taxi, and descent** phases show fewer accidents but are still important. **Go-around, standing, and unknown** phases have the least reported incidents.

This visualization provides a strong business case for **improving safety measures during landing and takeoff**, as these are the most critical flight phases.

In [ ]:

The cell below we creates **horizontal bar chart** to visualize **Purpose of Flight vs. Accident Risk**. Creates `purpose_counts` variable and uses `.value_counts()` function counts the occurrences of each unique value in the `"Purpose.of.flight"` column.
`y=purpose_counts.index` : Uses the flight purposes as labels for the y-axis.
`x=purpose_counts.values` : Uses the accident counts as values for the x-axis.
`palette="Purples"` : Applies a purple color gradient. `ax=ax` : Ensures the plot is drawn on the specified ax object (object-oriented plotting).

In [114]:
```
# Count accidents by purpose of flight
purpose_counts = Aviation_df["Purpose.of.flight"].value_counts().head(10)  # Top 10 c

# Create the plot
fig, ax =plt.subplots(figsize=(12, 6))
bars = sns.barplot(y=purpose_counts.index, x=purpose_counts.values, palette="Purples'

# Create a legend manually
legend_labels = purpose_counts.index
plt.legend(bars.patches, legend_labels, title="Purpose of Flight", bbox_to_anchor=(1.

# Labels and title
ax.set_xlabel("Number of Accidents")
ax.set_ylabel("Purpose of Flight")
ax.set_title("Accidents by Purpose of Flight")

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



In [ ]:

The output above shows **Personal" flights** have the highest number of accidents.

- **"Instructional" and "Unknown" flights** follow but with significantly fewer incidents.
- Other categories like **"Aerial Application," "Business," and "Positioning"** show a noticeable but lower number of accidents.
- **"Public Aircraft" and "Aerial Observation"** have the fewest reported accidents.

This insight suggests that personal flights may require more **safety measures, training, or stricter regulations**.

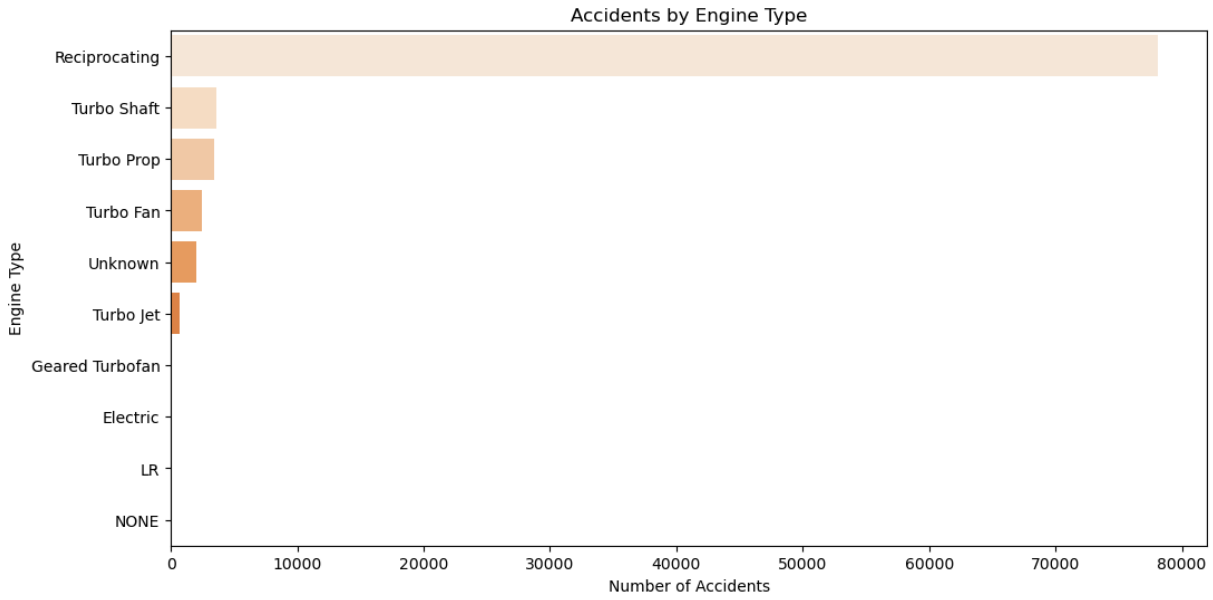
In [ ]:

In cell below I check if engine type influences accident outcomes.**Engine Type and Accident Severity**.

Creates a variable known as `engine_counts` to extracts the top 10 most common engine types in the dataset by counting the number of occurrences in the "Engine.Type" column. Then creates a `Barplot` using `sns.barplot` alias of seaborn library.

The **y-axis** represents the engine types (categories). The **x-axis** represents the number of accidents for each engine type. The **palette="Oranges"** adds a blue color gradient to the bars.

In [118]:
```
# Count accidents by engine type
engine_counts = Aviation_df["Engine.Type"].value_counts().head(10)  # Top 10 engine t

# Plot
fig, ax =plt.subplots(figsize=(12, 6))
sns.barplot(y=engine_counts.index, x=engine_counts.values, palette="Oranges", hue=eng
ax.set_xlabel("Number of Accidents")
ax.set_ylabel("Engine Type")
ax.set_title("Accidents by Engine Type")
plt.show()
```



- The bar chart visualizes the number of accidents categorized by engine type.
- The **"Reciprocating"** engine type has the highest number of accidents, significantly more than the others.
- Other engine types like **"Turbo Shaft", "Turbo Prop", "Turbo Fan"**, and **"Unknown"** have relatively fewer accidents.
- Some engine types, such as **"Geared Turbofan", "Electric", "NONE"**, and **"LR"**, have little to no data or recorded accidents.

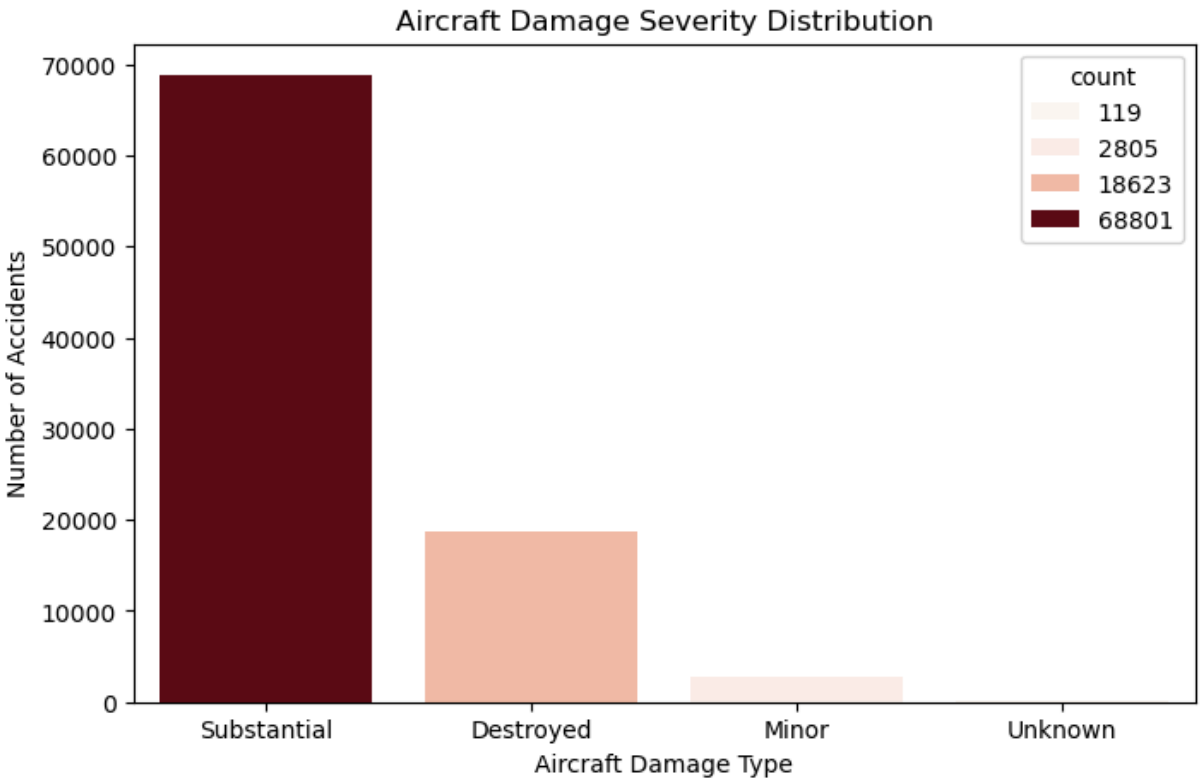
In [ ]:

Cell below analysis how aircraft damage is distributed among accidents. **Aircraft Damage Severity**.

Creates a variable `damage_counts` ounts the occurrences of each type of aircraft damage in the "Aircraft.damage" column of the dataset.

Then creates creates a `Barplot` : The **x-axis** represents the different types of aircraft damage. The **y-axis** represents the number of accidents corresponding to each damage type. The **palette="Reds"** applies a red color gradient to the bars.

In [116]:
```
# Count aircraft damage cases
damage_counts = Aviation_df["Aircraft.damage"].value_counts()

# Plot
fig, ax=plt.subplots(figsize=(8, 5))
sns.barplot(x=damage_counts.index, y=damage_counts.values, palette="Reds", hue=damage
ax.set_xlabel("Aircraft Damage Type")
ax.set_ylabel("Number of Accidents")
ax.set_title("Aircraft Damage Severity Distribution")
plt.show()
```



The output above shows **distribution of aircraft accidents based on damage severity**.

- **"Substantial"** damage is the most common, with nearly 70,000 accidents.
- **"Destroyed"** aircraft accidents are significantly lower, around 20,000 cases.
- **"Minor"** damage incidents are much fewer compared to substantial and destroyed categories.
- **"Unknown"** cases are the least reported.

In [ ]:

Business Objective 4 Findings From my Analysis

Findings

1. **Visual Meteorological Conditions (VMC) account for the highest number of accidents**

- Despite clear weather, human error, mechanical failures, and operational inefficiencies remain major contributors.
- This suggests that visibility alone does not guarantee safety—other risk factors must be addressed.

2. **Instrument Meteorological Conditions (IMC) have significantly fewer accidents**

Pilots rely on instruments in low-visibility conditions, indicating that training and advanced avionics help mitigate risks.

However, accidents in IMC often have severe consequences due to challenging recovery conditions.

Unknown Weather Conditions (UNK) have a small but notable accident occurrence

This highlights gaps in weather reporting or documentation, which could hinder accurate risk assessments.

Business Objective 4 Recomendation Based On Analysis and Findings.

Recommendations

In [ ]:

Create a copy of **Aviation_df** to use for Tableau Visualization and communicating my Finding.

In [93]:
```
# import csv
import csv

Aviation_df.to_csv('Tableau_data.csv', index=False, encoding='utf-8')
```

In [ ]: