Final Project Submission

Please fill out:

- Student name: Floyed Muchiri
- Student pace: full time - remote
- Scheduled project review date/time: 9th JUNE 2024
- Instructor name: Faith Rotich
- Blog post URL:
- Tableau public url: https://public.tableau.com/views/Aviation_data_17178266797050/Dashboard2
- dataset

Project overview

Business problem:

> Expanding into the aviation industry

Objective

> Determine the safest aircraft models for commercial and private use

A company is expanding into aviation industry 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.
> - As a data analyst, I am charged with determining which aircrafts are the lowest risk for the company to start the new business endeavor.
> - The findings will then help the company make decision on which aircraft to start with.

We are provided with Aviation dataset from NTSA for the research.

In the data folder is a copy of Aviation_data.csv, dataset from the National Transportation Safety Board that includes aviation accident data from 1962

to 2023 about civil aviation accidents and selected incidents in the United States and international waters.

Data Analysis

importing the required libraries

In [1]:
```
# import libraries

import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import csv
```

loading the dataset

In [2]:
```
file_path = 'data/Aviation_Data.csv'

data = pd.read_csv(file_path)
```

```
C:\Users\Floyed\AppData\Local\Temp\ipykernel_3796\562718360.py:3: DtypeWarning: Colu
mns (6,7,28) have mixed types. Specify dtype option on import or set low_memory=Fals
e.
  data = pd.read_csv(file_path)
```

Step 1: Data Exploration and Understanding

In [3]:
```
data.head()
```

Out[3]:

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

5 rows × 31 columns

In [4]:
```
data.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
```

In [5]:
```
data.describe()
```

Out[5]:

	Number.of.Engines	Total.Fatal.Injuries	Total.Serious.Injuries	Total.Minor.Injuries	Tot
count	82805.000000	77488.000000	76379.000000	76956.000000	8
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	

In this cell, we invetsgate the data under column investgation type to identify any other investgation to help us filter any irrelevant data

In [6]:
```
data['Investigation.Type'].value_counts()
```

Out[6]:
```
Investigation.Type
Accident      85015
Incident       3874
25-09-2020      702
26-09-2020       60
02-02-2021       39
              ...
13-09-2021        1
04-08-2021        1
05-08-2022        1
03-11-2020        1
06-01-2021        1
Name: count, Length: 71, dtype: int64
```

In [7]:
```
# investgating the incidents

pd.set_option('display.max_columns', None) #to display all columns


incidents = data[data['Investigation.Type']== 'Incident']
incidents.head(10)
```

Out[7]:

	Event.Id	Investigation.Type	Accident.Number	Event.Date	Location	Cou
23	20020917X02333	Incident	LAX82IA034	1982-01-03	VAN NUYS, CA	U S
40	20020917X01764	Incident	ATL82IA029	1982-01-05	PENSACOLA, FL	U S
79	20020917X01897	Incident	CHI82IA026	1982-01-12	CHICAGO, IL	U S
80	20020917X01765	Incident	ATL82IA034	1982-01-12	CLARKSBURG, WV	U S
119	20020917X01766	Incident	ATL82IA038	1982-01-19	WASHINGTON, DC	U S
131	20020917X02334	Incident	LAX82IA044	1982-01-20	SAN JOSE, CA	U S
149	20020917X01767	Incident	ATL82IA041	1982-01-22	LOUISVILLE, KY	U S
191	20020917X01768	Incident	ATL82IA042	1982-01-29	ATLANTA, GA	U S
194	20020917X02335	Incident	LAX82IA062	1982-01-30	TRUCKEE, CA	U S
279	20020917X01769	Incident	ATL82IA051	1982-02-09	BIRMINGHAM, AL	U S

we observe that the incidents investgation type does does not contain data neccessary for our safety investgation, so we need to drop them

since we are only interested with accidents, we will have to drop or filter the rows with other details other than accidents

In [8]:
```
# filtering the relevant rows

data = data[data['Investigation.Type']== 'Accident']

data.shape[0]
```

Out[8]: 85015

Great, we dropped a few rows

Also, from the preliminary investgation we observe that the dataset has many missing values and columns that are less

important to our investgation.

Data Cleaning

Drop less relevant columns

In [9]:
```
# columns that do not have relevance to our question can be dropped
columns_to_drop = ['Accident.Number', 'Registration.Number', 'Latitude', 'Longitude

data = data.drop(columns=columns_to_drop)
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 85015 entries, 0 to 90347
Data columns (total 24 columns):
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   Event.Id                85015 non-null  object 
 1   Investigation.Type      85015 non-null  object 
 2   Event.Date              85015 non-null  object 
 3   Location                84975 non-null  object 
 4   Country                 84809 non-null  object 
 5   Injury.Severity         84688 non-null  object 
 6   Aircraft.damage         83555 non-null  object 
 7   Aircraft.Category       30535 non-null  object 
 8   Make                    84979 non-null  object 
 9   Model                   84955 non-null  object 
 10  Amateur.Built           84957 non-null  object 
 11  Number.of.Engines       80115 non-null  float64
 12  Engine.Type             79155 non-null  object 
 13  FAR.Description         30368 non-null  object 
 14  Schedule                10104 non-null  object 
 15  Purpose.of.flight       80688 non-null  object 
 16  Air.carrier             14672 non-null  object 
 17  Total.Fatal.Injuries    74259 non-null  float64
 18  Total.Serious.Injuries  73152 non-null  float64
 19  Total.Minor.Injuries    73706 non-null  float64
 20  Total.Uninjured         79247 non-null  float64
 21  Weather.Condition       81882 non-null  object 
 22  Broad.phase.of.flight   59806 non-null  object 
 23  Report.Status           79776 non-null  object 
dtypes: float64(5), object(19)
memory usage: 16.2+ MB
```

change event date to date type

In [10]:
```
data['Event.Date'] = pd.to_datetime(data['Event.Date'], errors='coerce')
```

In [11]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 85015 entries, 0 to 90347
Data columns (total 24 columns):
 #   Column                  Non-Null Count  Dtype         
---  ------                  --------------  -----         
 0   Event.Id                85015 non-null  object        
 1   Investigation.Type      85015 non-null  object        
 2   Event.Date              85015 non-null  datetime64[ns]
 3   Location                84975 non-null  object        
 4   Country                 84809 non-null  object        
 5   Injury.Severity         84688 non-null  object        
 6   Aircraft.damage         83555 non-null  object        
 7   Aircraft.Category       30535 non-null  object        
 8   Make                    84979 non-null  object        
 9   Model                   84955 non-null  object        
 10  Amateur.Built           84957 non-null  object        
 11  Number.of.Engines       80115 non-null  float64       
 12  Engine.Type             79155 non-null  object        
 13  FAR.Description         30368 non-null  object        
 14  Schedule                10104 non-null  object        
 15  Purpose.of.flight       80688 non-null  object        
 16  Air.carrier             14672 non-null  object        
 17  Total.Fatal.Injuries    74259 non-null  float64       
 18  Total.Serious.Injuries  73152 non-null  float64       
 19  Total.Minor.Injuries    73706 non-null  float64       
 20  Total.Uninjured         79247 non-null  float64       
 21  Weather.Condition       81882 non-null  object        
 22  Broad.phase.of.flight   59806 non-null  object        
 23  Report.Status           79776 non-null  object        
dtypes: datetime64[ns](1), float64(5), object(18)
memory usage: 16.2+ MB
```

Dealing with null values

for key data which is not continuous such as model, make, we should drop all rows with null values

1. we should first drop rows that had investagtion that was not accidental, because we are interested in accidents.

2. Critical columns for this analysis can include Make, Model, Injury.Severity, Aircraft.damage, Total.Fatal.Injuries, Total.Serious.Injuries, Total.Minor.Injuries, Total.Uninjured., rows with Null values in these columns should be dropped.

In [12]:
```
# Drop rows where Investigation.Type is not relevant (I.e., keep only 'Accident')
data_cleaned = data[data['Investigation.Type'] == 'Accident']

# Verify the number of rows remaining
print(f"Number of rows after dropping irrelevant rows: {data_cleaned.shape[0]}")
```

```
Number of rows after dropping irrelevant rows: 85015
```

In [13]:
```
# Drop Rows with Missing Values in Critical Columns

critical_columns = ['Make', 'Model', 'Injury.Severity', 'Aircraft.damage', 'Total.F
                    'Total.Minor.Injuries', 'Total.Uninjured']

# Drop rows with missing values in any of the critical columns
data_cleaned = data_cleaned.dropna(subset=critical_columns)

# number of rows remaining
print(f"Number of rows after dropping rows with missing values in critical columns:
```

```
Number of rows after dropping rows with missing values in critical columns: 69598
```

great, now we have all relevant data accurate

In [14]:
```
data_cleaned.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 69598 entries, 0 to 90345
Data columns (total 24 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Event.Id                69598 non-null  object
 1   Investigation.Type      69598 non-null  object
 2   Event.Date              69598 non-null  datetime64[ns]
 3   Location                69570 non-null  object
 4   Country                 69408 non-null  object
 5   Injury.Severity         69598 non-null  object
 6   Aircraft.damage         69598 non-null  object
 7   Aircraft.Category       25780 non-null  object
 8   Make                    69598 non-null  object
 9   Model                   69598 non-null  object
 10  Amateur.Built           69597 non-null  object
 11  Number.of.Engines       66432 non-null  float64
 12  Engine.Type             65621 non-null  object
 13  FAR.Description         25571 non-null  object
 14  Schedule                8162 non-null   object
 15  Purpose.of.flight       67165 non-null  object
 16  Air.carrier             13611 non-null  object
 17  Total.Fatal.Injuries    69598 non-null  float64
 18  Total.Serious.Injuries  69598 non-null  float64
 19  Total.Minor.Injuries    69598 non-null  float64
 20  Total.Uninjured         69598 non-null  float64
 21  Weather.Condition       67359 non-null  object
 22  Broad.phase.of.flight   46430 non-null  object
 23  Report.Status           64973 non-null  object
dtypes: datetime64[ns](1), float64(5), object(18)
memory usage: 13.3+ MB
```

Standardize data in the 'Make' and 'Model' columns to be stringsV and in the right format

In [15]:
```
# Standardize Text
data_cleaned['Make'] = data_cleaned['Make'].str.strip().str.upper()
```

```
data_cleaned['Model'] = data_cleaned['Model'].str.strip().str.upper()
```

ANALYSIS

This is the section where we start analysing data to answer the required question

In [16]:
```
# lets check what is aircraft category

# Get the frequency of each item in aircraft category
aircraft_category = data_cleaned['Aircraft.Category'].value_counts()

# Print the frequencies
print(f"Airplane category frequencies:")
print(aircraft_category)
```

```
Airplane category frequencies:
Aircraft.Category
Airplane             21977
Helicopter            2826
Glider                 417
Weight-Shift           160
Gyrocraft              153
Balloon                113
Powered Parachute       88
Ultralight              23
Unknown                 10
WSFT                     9
Rocket                   1
Powered-Lift             1
Blimp                    1
ULTR                     1
Name: count, dtype: int64
```

here we have interesting data:

-- whats in our dataset and whats of importance to the business:

Given the company's goal of entering the aviation industry by purchasing and operating airplanes for commercial and private enterprises, the main concern should focus on categories that are most relevant to these activities. Specifically, we should focus on the "Airplane", "Helicopter" categories because it is the most common and widely used for both commercial and private aviation purposes.

Filter data for Airplanes and Helicopter categories

In [17]:
```
# Filter the cleaned dataset for relevant categories (Airplane and Helicopter)
relevant_categories = ['Airplane', 'Helicopter']
data_filtered = data_cleaned[data_cleaned['Aircraft.Category'].isin(relevant_catego
```

```
data_filtered.shape[0]
```

Out[17]: 24803

A graph of number of accidents per year

- we first extract year from the dates
- counts accidentsm per year for the graph

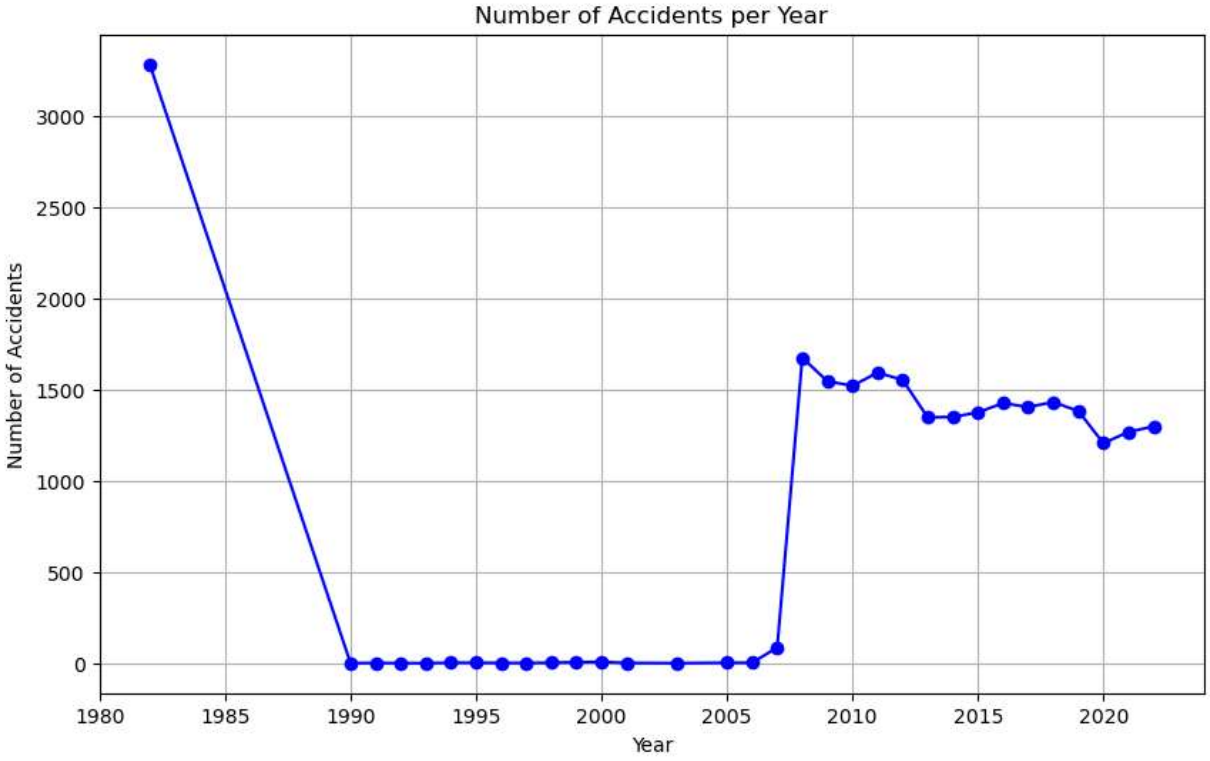
In [22]:
```
#  Extract the year from the 'Event.Date' column

data_filtered['Event.Year'] = data_filtered['Event.Date'].dt.year
```

```
C:\Users\Floyed\AppData\Local\Temp\ipykernel_3796\2780474457.py:3: SettingWithCopyWa
rning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  data_filtered['Event.Year'] = data_filtered['Event.Date'].dt.year
```

In [23]:
```
# Count the number of accidents per year
accidents_per_year = data_filtered['Event.Year'].value_counts().sort_index()
accidents_per_year
```

Out[23]:
```
Event.Year
1982.0    3285
1990.0       1
1991.0       2
1992.0       1
1993.0       1
1994.0       3
1995.0       3
1996.0       2
1997.0       2
1998.0       3
1999.0       6
2000.0       8
2001.0       2
2003.0       1
2005.0       3
2006.0       3
2007.0      86
2008.0    1676
2009.0    1547
2010.0    1522
2011.0    1594
2012.0    1554
2013.0    1348
2014.0    1351
2015.0    1376
2016.0    1427
2017.0    1405
2018.0    1433
2019.0    1383
2020.0    1207
2021.0    1269
2022.0    1299
Name: count, dtype: int64
```

In [24]:
```
#  Plotting the data as a line graph

plt.figure(figsize=(10, 6))
plt.plot(accidents_per_year.index, accidents_per_year.values, marker='o', linestyle
plt.xlabel('Year')
plt.ylabel('Number of Accidents')
plt.title('Number of Accidents per Year')
plt.grid(True)
plt.show()
```



- From the graph, we have a sharp decline from 1982 - 1990 - this could be a cumulative sum for all accidents before, secondly, companies from these times could have undergone significant changes to help determine the risk of aircraft
- > < 100 accidents from 1990, to 2006 - this could be due to underrecording of accidents and may not give accurate or sufficient summarry
- 86 accident in 2007
- A sharp increase to above 1500 in 2008 > this could be the period NTSA started recording all aviation data
- a consistent graph from 2008 -- the accurate data to give us consistent results

we have to work with consistent data, a more recent data that reflects the technological changes in companies.

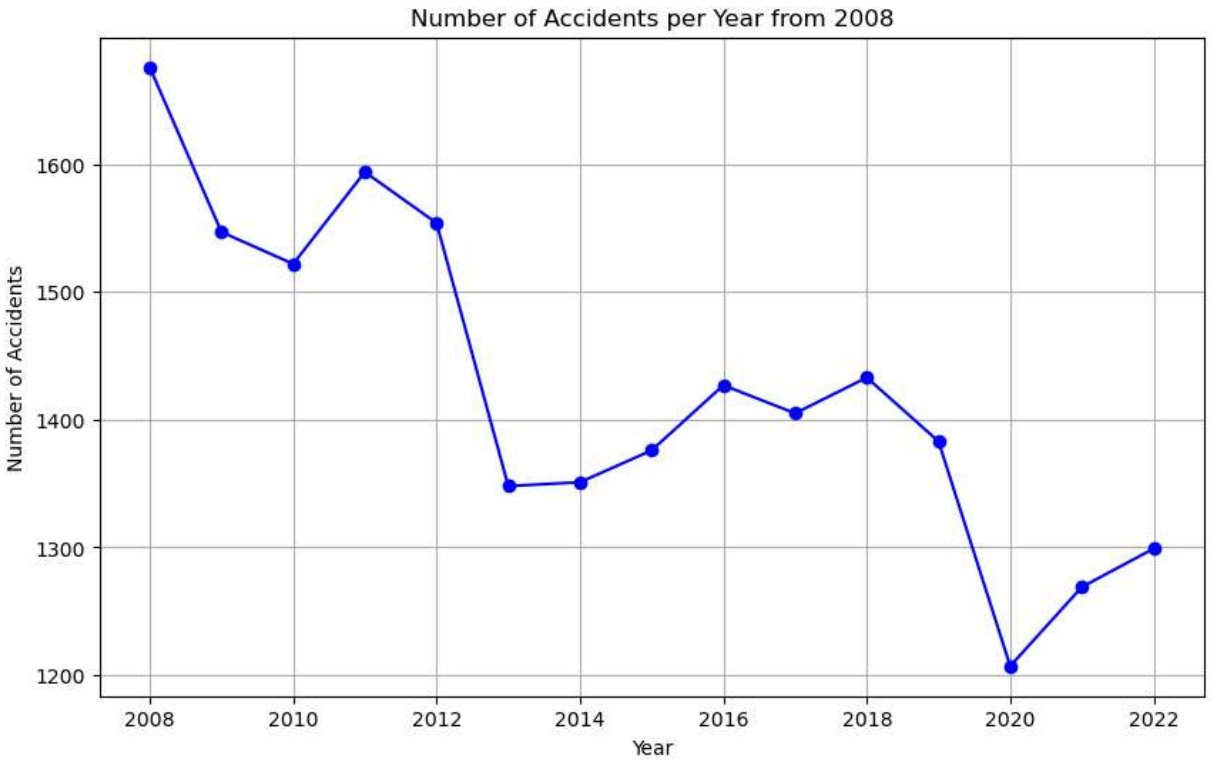
- > so we will filter to have a consistent data from 2008 to work with and perform our analysis.

In [25]:
```
working_data = data_filtered[data_filtered['Event.Year'] > 2007]
working_data.shape[0]
```

Out[25]: 21391

we plot the curve again

In [26]:
```
accidents_per_year = working_data['Event.Year'].value_counts().sort_index()

#  Plotting the line graph

plt.figure(figsize=(10, 6))
plt.plot(accidents_per_year.index, accidents_per_year.values, marker='o', linestyle
plt.xlabel('Year')
plt.ylabel('Number of Accidents')
plt.title('Number of Accidents per Year from 2008')
plt.grid(True)
plt.show()
```



In [ ]:
```

```

1. we can determine the models and number of accidents per model in our working data

In [27]:
```
# Get the frequency of each model
model_counts = working_data['Model'].value_counts()

# Print the model frequencies
print(f"Airplane model frequencies:")
print(model_counts)
```

```
Airplane model frequencies:
Model
172                     694
R44                     271
182                     266
PA28                    261
SR22                    233
                       ...
DC9                       1
BEAVER U-6                1
U206D                     1
FOKKER DR-1 TRIPLANE      1
PA-44                     1
Name: count, Length: 4302, dtype: int64
```

lets separate the two categories

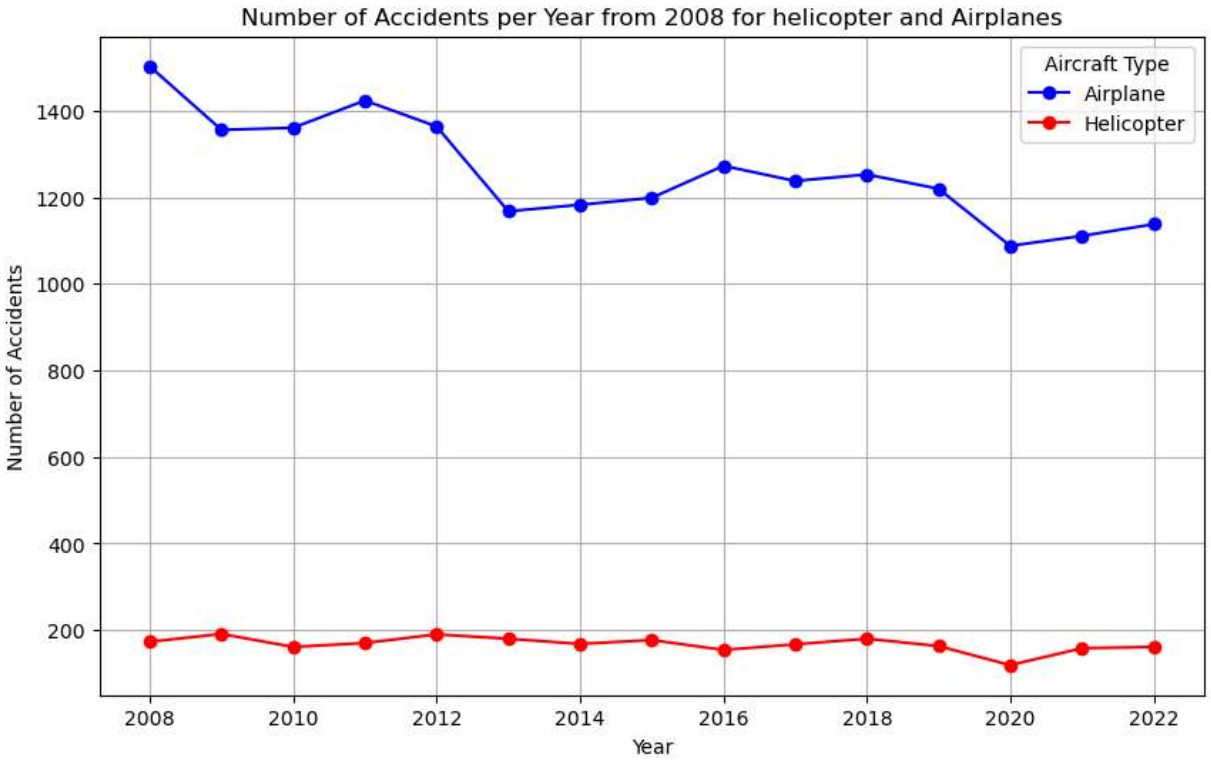
1. Airplane

2. Helicopter

In [28]:
```
# Airplane category

Airplane_data = working_data[working_data['Aircraft.Category'] == 'Airplane']

# Helicopter category

Helicopter_data = working_data[working_data['Aircraft.Category'] == 'Helicopter']
```

A graph for number of accidents for each category from 2008

In [29]:
```
Airplane_accidents_per_year = Airplane_data['Event.Year'].value_counts().sort_index

Helicopter_accidents_per_year = Helicopter_data['Event.Year'].value_counts().sort_i


#  Plotting the line graph

plt.figure(figsize=(10, 6))
plt.plot(Airplane_accidents_per_year.index, Airplane_accidents_per_year.values, mar
plt.plot(Helicopter_accidents_per_year.index, Helicopter_accidents_per_year.values,
plt.xlabel('Year')
plt.ylabel('Number of Accidents')
plt.title('Number of Accidents per Year from 2008 for helicopter and Airplanes')
plt.grid(True)
plt.legend(title='Aircraft Type')
plt.show()
```



Export the two data sets for further investgation using other tools

In [51]:
```
# Export the datasets to CSV files
Helicopter_accidents_per_year.to_csv('data/Helicopter_accidents.csv', index=False)
Airplane_accidents_per_year.to_csv('data/Airplane_accidents.csv', index=False)

print("Datasets have been successfully exported to 'Helicopter_accidents.csv' and '
```

```
Datasets have been successfully exported to 'Helicopter_accidents.csv' and 'Airplane
_accidents.csv'
```

In [55]:
```
# Export the datasets to CSV files
Helicopter_data.to_csv('data/Helicopter_data.csv', index=False)
Airplane_data.to_csv('data/Airplane_data.csv', index=False)

print("Datasets have been successfully exported to 'Helicopter_data.csv' and 'Airpl
```

```
Datasets have been successfully exported to 'Helicopter_data.csv' and 'Airplane_dat
a.csv'
```

In [30]:
```
# Get the frequency of each helicopter make
Helicopter_make_counts = Helicopter_data['Make'].value_counts()

# Print the make frequencies
print(f"Helicopter model frequencies:")
print(Helicopter_make_counts)
```

```
Helicopter model frequencies:
Make
BELL                            620
ROBINSON                        363
ROBINSON HELICOPTER             221
ROBINSON HELICOPTER COMPANY     181
HUGHES                          152
                               ...
BERTRAM WILLIAM J                 1
SCOTTS-BELL 47 INC                1
JOHNSTON DOUGLAS S                1
CROMAN CORPORATION                1
CHILDS MICHAEL  A                 1
Name: count, Length: 217, dtype: int64
```

In [31]:
```
# Get the frequency of each Airplane make make
Airplane_make_counts = Airplane_data['Make'].value_counts()

# Print the make frequencies
print(f"Airplane model frequencies:")
print(Airplane_make_counts)
```

```
Airplane model frequencies:
Make
CESSNA                5657
PIPER                 3321
BEECH                 1193
MOONEY                 290
BOEING                 259
                      ...
SHOOK                    1
CROSLEY                  1
SHAW DOUGLAS WAYNE       1
KOTH LARRY               1
ORLICAN S R O            1
Name: count, Length: 2995, dtype: int64
```

In [32]:
```
Helicopter_data.head()
```

Out[32]:

	Event.Id	Investigation.Type	Event.Date	Location	Country	Injury.Seve
63929	20080122X00087	Accident	2008-01-07	DeLand, FL	United States	Non-F
63931	20080220X00219	Accident	2008-01-07	Zuzenhausen, Germany	Germany	F
63933	20080204X00131	Accident	2008-01-08	Riverside, CA	United States	Non-F
63934	20080210X00162	Accident	2008-01-08	North Palm Bch, FL	United States	Non-F
63939	20080211X00175	Accident	2008-01-10	Pitt Meadow, BC, Canada	Canada	Non-F

In [33]:
```
Airplane_data.head()
```

Out[33]:

	Event.Id	Investigation.Type	Event.Date	Location	Country	Injury.Severit
63913	20080107X00026	Accident	2008-01-01	Sonoma, CA	United States	Non-Fat
63914	20080109X00036	Accident	2008-01-01	Arcola, TX	United States	Non-Fat
63915	20080115X00051	Accident	2008-01-02	Loveland, CO	United States	Non-Fat
63916	20080210X00166	Accident	2008-01-02	Brunswick, GA	United States	Non-Fat
63919	20080115X00046	Accident	2008-01-03	Stevensville, MD	United States	Non-Fat

Checking the Injury Severity of each make

In [34]:
```
Airplane_data['Aircraft.damage'].value_counts()
```

Out[34]:
```
Aircraft.damage
Substantial    16475
Destroyed       2208
Minor            129
Unknown           67
Name: count, dtype: int64
```

In [35]:
```
Helicopter_data['Purpose.of.flight'].value_counts()
```

Out[35]:
```
Purpose.of.flight
Personal                  524
Instructional             372
Aerial Application        263
Other Work Use            143
Positioning               128
Aerial Observation        121
External Load              96
Business                   84
Unknown                    80
Public Aircraft - Local    47
Flight Test                31
Public Aircraft - Federal  26
Public Aircraft - State    26
Firefighting               16
Public Aircraft            15
Executive/corporate        12
Ferry                      11
Air Drop                    4
Air Race show               3
PUBS                        1
PUBL                        1
Name: count, dtype: int64
```

In [36]:
```
Airplane_data['Injury.Severity'].value_counts()
```

Out[36]:
```
Injury.Severity
Non-Fatal    14414
Fatal         4187
Minor          162
Serious        116
Name: count, dtype: int64
```

SAFETY ANALYSIS

A safety analysis will analyze the safety of each model of either airplane or helicopter to determine the top safe models

Steps

- We calculate the total number of accidents for each aircraft model.
- We calculate the total number of fatalities, serious injuries, and minor injuries for each aircraft model.
- We define weights for each factor (e.g., fatalities, serious injuries, minor injuries) to create a safety score.
- We calculate the safety score for each aircraft model by combining the factors using the defined weights.
- We rank the aircraft models based on their safety scores, with lower scores indicating higher safety.

we create a function, get_safe_models to return the safety summarry

In [37]:
```
def calculate_safety_score(data):
    # Define weights for each factor (e.g., fatalities, serious injuries, minor inj
    weight_fatalities = 0.4
    weight_serious_injuries = 0.3
    weight_minor_injuries = 0.2
    weight_aircraft_damage = 0.1

    weight_injury_severity_fatal = 0.4
    weight_injury_severity_serious = 0.3
    weight_injury_severity_minor = 0.2
    weight_injury_severity_non_fatal = 0.1

    # Calculate the total number of accidents for each aircraft model
    accidents_per_model = data.groupby('Model')['Event.Id'].count()

    # Calculate the total number of fatalities, serious injuries, minor injuries fo
    fatalities_per_model = data.groupby('Model')['Total.Fatal.Injuries'].sum()
    serious_injuries_per_model = data.groupby('Model')['Total.Serious.Injuries'].su
    minor_injuries_per_model = data.groupby('Model')['Total.Minor.Injuries'].sum()
    non_fatal_injuries_per_model = data.groupby('Model')['Total.Uninjured'].sum()

    # Define weights for each level of aircraft damage
    damage_weights = {'Destroyed': 0.5, 'Substantial': 0.3, 'Minor': 0.2}

    # Calculate the total weight of aircraft damage for each model
    total_damage_weight_per_model = data.groupby('Model')['Aircraft.damage'].apply(

    # Calculate the total weight of injury severity for each model
    total_injury_severity_weight_per_model = (
        (fatalities_per_model * weight_fatalities * weight_injury_severity_fatal) +
        (serious_injuries_per_model * weight_serious_injuries * weight_injury_sever
        (minor_injuries_per_model * weight_minor_injuries * weight_injury_severity_
        (non_fatal_injuries_per_model * weight_injury_severity_non_fatal)
    ) / accidents_per_model

    # Calculate safety score for each model
    safety_score_per_model = (
        total_injury_severity_weight_per_model +
        (total_damage_weight_per_model * weight_aircraft_damage)
    )

    # Ranking aircraft models based on safety score
    safest_models = safety_score_per_model.sort_values(ascending=True)

    # Create a DataFrame with model, make, purpose of flight and safety score
    models = safest_models.index
    model_info = data[data['Model'].isin(models)].drop_duplicates(subset=['Model',
    safety_scores_df = pd.DataFrame({
        'Model': safest_models.index,
        'Safety Score': safest_models.values
    }).merge(model_info, on='Model')
```

```
    return safety_scores_df
```

1. AIRPLANE

In [38]:
```
Airplanes_safety_score = calculate_safety_score(Airplane_data)
Airplanes_safety_score # can be sorted as [Airplanes_safety_score['Purpose.of.fligh
```

Out[38]:

	Model	Safety Score	Make	Purpose.of.flight
0	787	0.03	BOEING	NaN
1	STALKER	0.03	LOCKHEED	NaN
2	HEAVISIDE2	0.03	KITTY HAWK	Flight Test
3	JAS4-2	0.03	JOBY AERO INC	Flight Test
4	J3F-50	0.03	PIPER	Personal
...	...	...	...	...
5781	747-400	36.53	BOEING	NaN
5782	A340 - 300	39.72	AIRBUS	NaN
5783	777 - 206	42.82	BOEING	NaN
5784	A330-323	45.42	AIRBUS INDUSTRIE	NaN
5785	A380	54.90	AIRBUS	NaN

5786 rows × 4 columns

Airplane for comercial use

In [40]:
```
Airplanes_safety_score[Airplanes_safety_score['Purpose.of.flight']=='Business'].hea
```

Out[40]:

	Model	Safety Score	Make	Purpose.of.flight
23	E-LSA	0.07	BRISTELL	Business

Airplane for private use

In [41]:
```
Airplanes_safety_score[Airplanes_safety_score['Purpose.of.flight']=='Personal'].hea
```

Out[41]:

	Model	Safety Score	Make	Purpose.of.flight
4	J3F-50	0.03	PIPER	Personal

2. Helicopter

In [39]:
```
Helicopter_safety_score = calculate_safety_score(Helicopter_data)
Helicopter_safety_score.head(5)
```

Out[39]:

	Model	Safety Score	Make	Purpose.of.flight
0	APT70	0.03	BELL	NaN
1	MATRICE	0.06	DJI	Other Work Use
2	400	0.07	SAFARI	Personal
3	S-58ET	0.07	SIKORSKY	External Load
4	S-58HT	0.07	SIKORSKY	External Load

Helicopter for private use

In [42]:
```
Helicopter_safety_score[Helicopter_safety_score['Purpose.of.flight']=='Personal'].h
```

Out[42]:

	Model	Safety Score	Make	Purpose.of.flight
2	400	0.07	SAFARI	Personal

Helicopter for Comercial use

In [43]:
```
Helicopter_safety_score[Helicopter_safety_score['Purpose.of.flight']=='Business'].h
```

Out[43]:

	Model	Safety Score	Make	Purpose.of.flight
45	F28F	0.13	ENSTROM	Business

Conclusion and Recommendation

In [ ]:
```
The analysis shows that
1. For Arplanes, A boeing 787 is the safest plane to purchase
2. For Helicopters, a BELL APT70, which is a drone scored the best
```