Phase 1 Project

- Student Name: MUTHEE JAMES WACHIRA
- Student Pace: Part time
- Instructor Name: William Onkomba

Business Understanding

Our company is expanding into new industries, and the management has decided to explore the aviation sector as a promising opportunity for diversification and growth. Acquiring and operating aircraft for both commercial and private purposes is a strategic decision, particularly with the increasing global demand for air travel. However, entering the aviation industry presents significant challenges mostly in managing risks related to aircraft safety and reliability

It is essential to fully understand and address these risks to ensure a successful entry into the aviation market. Choosing aircraft that minimize operational and safety risks is key to safeguarding the company's assets, reputation, and long-term success.

Problem Statement

The company is seeking to identify the lowest-risk aircraft to purchase and operate as it enters the aviation sector. The primary challenge is to is to assess and compare different aircraft make and models based on several risk factors which include total fatal injuries, frequency of accidents, total serious injuries, Aircraft damage after the accident as well as the injury severity and recommend to the company the best aircraft to purchase and operate. The selected aircraft should meet the safety requirements, minimize potential liabilities and align with the company's business strategy. The metrics of success for our selected aircraft will be minimal or zero accidents over the time period under study, zero or less fatalities and injury after accidents and less damage to the aircraft after the accident. The insights and recommendations shall be presented to the new head of aviation division in the company.

Data

In order to give the company strategic insights and recommendations on which airplane to buy and operate, Aviation dataset from [https://www.kaggle.com/datasets/khsamaha/aviation-accident-database-synopses (https://www.kaggle.com/datasets/khsamaha/aviation-accident-database-synopses)](https://www.kaggle.com/datasets/khsamaha/aviation-accident-database-synopses) has been provided. Analysis will be done on this data on jupyter notebook using python programming language. After the analysis of the provided data, we should be able to recommend the following to the company the safest aircraft make and model to purchase and operate based on the metrics of success highlighted earlier.

Data Preparation and Cleaning

This will be the initial step of our data analysis process. This is a crucial step as it will help us do the following:

- Load the relevant libraries to use for the data analysis and visualization
- Load the Aviation dataset which is in csv form to Jupyter Notebook
- Understand the data by checking the number of rows and columns, statistics of numerical columns as well as the first few rows of the data
- Identify and fix the missing values
- Ensure the columns have the correct data type
- Create new features that will be important for our analysis
- Merge important Data Frames

Load the Python Libraries

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

Loading the dataset

In [2]:
```
df = pd.read_csv("AviationData.csv", encoding = "ISO-8859-1", low_memory=False, index_col=0)
State_Codes = pd.read_csv("USState_Codes.csv")
pd.set_option("Max_colwidth", 500)
```

Understanding the Aviation dataset

In [3]:
```
#check the first five rows of our dataset
df.head()
```

Out[3]:

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

5 rows × 30 columns

In [4]:
```
# check the column names
df.columns
```

Out[4]:
```
Index(['Investigation.Type', 'Accident.Number', 'Event.Date', 'Location',
       'Country', 'Latitude', 'Longitude', 'Airport.Code', 'Airport.Name',
       'Injury.Severity', 'Aircraft.damage', 'Aircraft.Category',
       'Registration.Number', 'Make', 'Model', 'Amateur.Built',
       'Number.of.Engines', 'Engine.Type', 'FAR.Description', 'Schedule',
       'Purpose.of.flight', 'Air.carrier', 'Total.Fatal.Injuries',
       'Total.Serious.Injuries', 'Total.Minor.Injuries', 'Total.Uninjured',
       'Weather.Condition', 'Broad.phase.of.flight', 'Report.Status',
       'Publication.Date'],
      dtype='object')
```

In [5]:
```
# check the number of rows and columns
df.shape

# print the number of rows and columns
print(f"This data has {df.shape[0]} rows and {df.shape[1]} columns")
```

```
This data has 88889 rows and 30 columns
```

In [6]:
```
# check information about your data
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 88889 entries, 20001218X45444 to 20221230106513
Data columns (total 30 columns):
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   Investigation.Type      88889 non-null  object 
 1   Accident.Number         88889 non-null  object 
 2   Event.Date              88889 non-null  object 
 3   Location                88837 non-null  object 
 4   Country                 88663 non-null  object 
 5   Latitude                34382 non-null  object 
 6   Longitude               34373 non-null  object 
 7   Airport.Code            50249 non-null  object 
 8   Airport.Name            52790 non-null  object 
 9   Injury.Severity         87889 non-null  object 
 10  Aircraft.damage         85695 non-null  object 
 11  Aircraft.Category       32287 non-null  object 
 12  Registration.Number     87572 non-null  object 
 13  Make                    88826 non-null  object 
 14  Model                   88797 non-null  object 
 15  Amateur.Built           88787 non-null  object 
 16  Number.of.Engines       82805 non-null  float64
 17  Engine.Type             81812 non-null  object 
 18  FAR.Description         32023 non-null  object 
 19  Schedule                12582 non-null  object 
 20  Purpose.of.flight       82697 non-null  object 
 21  Air.carrier             16648 non-null  object 
 22  Total.Fatal.Injuries    77488 non-null  float64
 23  Total.Serious.Injuries  76379 non-null  float64
 24  Total.Minor.Injuries    76956 non-null  float64
 25  Total.Uninjured         82977 non-null  float64
 26  Weather.Condition       84397 non-null  object 
 27  Broad.phase.of.flight   61724 non-null  object 
 28  Report.Status           82508 non-null  object 
 29  Publication.Date        75118 non-null  object 
dtypes: float64(5), object(25)
memory usage: 21.0+ MB
```

In [7]:
```
# check summary statistics of numerical columns
df.describe()
```

Out[7]:

	Number.of.Engines	Total.Fatal.Injuries	Total.Serious.Injuries	Total.Minor.Injuries	Total.Uninjured
count	82805.000000	77488.000000	76379.000000	76956.000000	82977.000000
mean	1.146585	0.647855	0.279881	0.357061	5.325440
std	0.446510	5.485960	1.544084	2.235625	27.913634
min	0.000000	0.000000	0.000000	0.000000	0.000000
25%	1.000000	0.000000	0.000000	0.000000	0.000000
50%	1.000000	0.000000	0.000000	0.000000	1.000000
75%	1.000000	0.000000	0.000000	0.000000	2.000000
max	8.000000	349.000000	161.000000	380.000000	699.000000

In [8]:
```
# Check descriptive statistics for categorical data
df.select_dtypes(include="O").describe().T
```

Out[8]:

	count	unique	top	freq
Investigation.Type	88889	2	Accident	85015
Accident.Number	88889	88863	CEN22LA149	2
Event.Date	88889	14782	1984-06-30	25
Location	88837	27758	ANCHORAGE, AK	434
Country	88663	219	United States	82248
Latitude	34382	25589	332739N	19
Longitude	34373	27154	0112457W	24
Airport.Code	50249	10375	NONE	1488
Airport.Name	52790	24871	Private	240
Injury.Severity	87889	109	Non-Fatal	67357
Aircraft.damage	85695	4	Substantial	64148
Aircraft.Category	32287	15	Airplane	27617
Registration.Number	87572	79105	NONE	344
Make	88826	8237	Cessna	22227
Model	88797	12318	152	2367
Amateur.Built	88787	2	No	80312
Engine.Type	81812	13	Reciprocating	69530
FAR.Description	32023	31	091	18221
Schedule	12582	3	NSCH	4474
Purpose.of.flight	82697	26	Personal	49448
Air.carrier	16648	13590	Pilot	258
Weather.Condition	84397	4	VMC	77303
Broad.phase.of.flight	61724	12	Landing	15428
Report.Status	82508	17075	Probable Cause	61754
Publication.Date	75118	2924	25-09-2020	17019

Data Preparation

This is the process of cleaning and organizing raw data into format that can be used for analysis. It is a crucial step that ensures that the data is accurate, consistent, and structured in a way that will allow for generating meaningful analysis and insights. The entire data preparation process is aimed at improving the quality and usability of the data. The process involves identifying and handling missing data, identifying and removing duplicate records, creating new features, identifying and correcting wrong column data types and lastly but not least removing columns that will not be necessary for our overall analysis.

Identifying Missing Data and Duplicate Rows

In [9]:
```
#Identifying Duplicate rows
duplicated = df.duplicated().sum()

#print the number of duplicated rows
print(f"This dataset has {duplicated} duplicate rows")
```

```
This dataset has 0 duplicate rows
```

In [10]:
```
# Identify null data
null_values = df.isna().sum().sort_values(ascending= False)

null_values
```

Out[10]:
```
Schedule                  76307
Air.carrier               72241
FAR.Description           56866
Aircraft.Category         56602
Longitude                 54516
Latitude                  54507
Airport.Code              38640
Airport.Name              36099
Broad.phase.of.flight     27165
Publication.Date          13771
Total.Serious.Injuries    12510
Total.Minor.Injuries      11933
Total.Fatal.Injuries      11401
Engine.Type                7077
Report.Status              6381
Purpose.of.flight          6192
Number.of.Engines          6084
Total.Uninjured            5912
Weather.Condition          4492
Aircraft.damage            3194
Registration.Number        1317
Injury.Severity            1000
Country                     226
Amateur.Built               102
Model                        92
Make                         63
Location                     52
Accident.Number               0
Event.Date                    0
Investigation.Type            0
dtype: int64
```

Handling Missing Data

From the above, our data has a lot of missing data. We need to clean the dataset to ensure we are working with a clean data for greater accuracy and better insights.

Data Cleaning

In [11]:
```
# Convert null_values to a DataFrame
null_values_df = pd.DataFrame(null_values).reset_index()

# Renaming the columns of the DataFrame
null_values_df.columns = ['Feature', 'NullData']

# Displaying the first few rows of the DataFrame
null_values_df.head()
```

Out[11]:

	Feature	NullData
0	Schedule	76307
1	Air.carrier	72241
2	FAR.Description	56866
3	Aircraft.Category	56602
4	Longitude	54516

In [12]:
```
#Checking the length of the data
len_aviation_dataset = len(df)


#Create a new feature "Percentage Missing Values"
null_values_df["Pct Null Values"] = null_values_df["NullData"] / len(df) * 100

print(f"The length of our aviation dataset is {len(df)}")

null_values_df.head()
```

```
The length of our aviation dataset is 88889
```

Out[12]:

	Feature	NullData	Pct Null Values
0	Schedule	76307	85.845268
1	Air.carrier	72241	81.271023
2	FAR.Description	56866	63.974170
3	Aircraft.Category	56602	63.677170
4	Longitude	54516	61.330423

From the percentage of missing values calculated, several columns will be removed from the dataset due to the high percentage of missing values.

Similarly, those columns that have less impact on our analysis and final insights and recommendations shall be removed.

In [13]:
```
#Identify columns to remove from the dataset
columns_to_discard = ["Schedule", "Air.carrier", "FAR.Description", "Airport.Code", "Airport.Name",
                  "Publication.Date","Registration.Number", "Broad.phase.of.flight",
                  "Accident.Number", "Latitude", "Longitude"]

updated_df = df.drop(columns=columns_to_discard)

updated_df.head(3)
```

Out[13]:

	Investigation.Type	Event.Date	Location	Country	Injury.Severity	Aircraft.damage	Aircraft.Category	Make	Model
Event.Id									
20001218X45444	Accident	1948-10-24	MOOSE CREEK, ID	United States	Fatal(2)	Destroyed	NaN	Stinson	108-3
20001218X45447	Accident	1962-07-19	BRIDGEPORT, CA	United States	Fatal(4)	Destroyed	NaN	Piper	PA24-180
20061025X01555	Accident	1974-08-30	Saltville, VA	United States	Fatal(3)	Destroyed	NaN	Cessna	172M

From the percentage of missing data DataFrame, the rows of missing values corresponding to the columns for Country, Amateur Built, Model, Make, Location, Injury Severity, Aircraft Damage, Purpose of Flight, Weather Condition and Report Status can be removed as their effect in the analysis of the data will not be significant. Their removal will have insignifant effect to the overall analysis.

In [14]:
```
# List of columns to check for missing data
rows_to_exclude = ["Purpose.of.flight", "Injury.Severity", "Weather.Condition", "Report.Status",
                        "Amateur.Built", "Make", "Aircraft.damage", "Model", "Country", "Location"]

# Remove rows with null data in the specified columns
df2 = updated_df.dropna(subset=rows_to_exclude)

df2.head(3)
```

Out[14]:

	Investigation.Type	Event.Date	Location	Country	Injury.Severity	Aircraft.damage	Aircraft.Category	Make	Model
Event.Id									
20001218X45444	Accident	1948-10-24	MOOSE CREEK, ID	United States	Fatal(2)	Destroyed	NaN	Stinson	108-3
20001218X45447	Accident	1962-07-19	BRIDGEPORT, CA	United States	Fatal(4)	Destroyed	NaN	Piper	PA24-180
20061025X01555	Accident	1974-08-30	Saltville, VA	United States	Fatal(3)	Destroyed	NaN	Cessna	172M

In [15]:
```
# Filtering our data to have specific values of Purpose of flight column

df2 = df2[df2["Purpose.of.flight"].isin(["Personal", "Public Aircraft", "Public Aircraft - Local",
                                           "Business", "Executive/corporate"])]
```

As per the business problem understanding, our company wants to venture into a new industry and specifically into purchasing and operating of Airplanes. And from the dataset provided, United States represents the greatest percentage of the countries in the "Country" column. We shall therefore filter our data to have Airplanes only on the 'Aircraft Category' column and United States on the 'Country' column.

In [16]:
```
#Filter the data for Country in United States and Aircraft Category is Airplanes

df3 = df2[(df2["Country"] == "United States") & (df2["Aircraft.Category"] == "Airplane")]

df3.head(3)
```

Out[16]:

	Investigation.Type	Event.Date	Location	Country	Injury.Severity	Aircraft.damage	Aircraft.Category	Make	Model	An
Event.Id										
20020909X01562	Accident	1982-01-01	PULLMAN, WA	United States	Non-Fatal	Substantial	Airplane	Cessna	140	
20020909X01561	Accident	1982-01-01	EAST HANOVER, NJ	United States	Non-Fatal	Substantial	Airplane	Cessna	401B	
20020917X02148	Accident	1982-01-02	HOMER, LA	United States	Non-Fatal	Destroyed	Airplane	Bellanca	17-30A	

For us to be able to do a proper data analysis and get accurate insights, we need to fill in the missing values in the different columns that we shall use for our analysis.

Filling in the Missing Values in the Numerical Columns

In [17]:
```
# Getting aggregates statistics for numerical Columns with null data

df3[["Number.of.Engines", "Total.Serious.Injuries","Total.Uninjured",
              "Total.Fatal.Injuries", "Total.Minor.Injuries"]].agg(["mean", "median", "max", "min"])
```

Out[17]:

	Number.of.Engines	Total.Serious.Injuries	Total.Uninjured	Total.Fatal.Injuries	Total.Minor.Injuries
mean	1.081416	0.250799	1.137754	0.342208	0.209401
median	1.000000	0.000000	1.000000	0.000000	0.000000
max	4.000000	9.000000	38.000000	14.000000	6.000000
min	0.000000	0.000000	0.000000	0.000000	0.000000

We shall fill in the null values of our numerical columns using the median value of the columns as the mean will not give an accurate picture. As can be observed the minimum and maximum values for the columns save for the Number of engines have a significant difference, hence possibility of outliers. Hence we shall use the median to fill in the null values for the columns

In [18]:
```
# Creating a copy of the filtered data
df3 = df3.copy()

# inpute the null values of the columns with the median

df3["Total.Fatal.Injuries"].fillna(0, inplace=True)
df3["Total.Serious.Injuries"].fillna(0, inplace=True)
df3["Total.Minor.Injuries"].fillna(0, inplace=True)
df3["Number.of.Engines"].fillna(1, inplace=True)
df3["Total.Uninjured"].fillna(1, inplace=True)
```

In [19]:
```
# check for updated missing values

df3.isna().sum()
```

Out[19]:
```
Investigation.Type        0
Event.Date                0
Location                  0
Country                   0
Injury.Severity           0
Aircraft.damage           0
Aircraft.Category         0
Make                      0
Model                     0
Amateur.Built             0
Number.of.Engines         0
Engine.Type              96
Purpose.of.flight         0
Total.Fatal.Injuries      0
Total.Serious.Injuries    0
Total.Minor.Injuries      0
Total.Uninjured           0
Weather.Condition         0
Report.Status             0
dtype: int64
```

Fill in the missing values for categorical data

In [20]:
```
# Check mode of 'Engine.Type' Column
mode_engine_type = df3["Engine.Type"].mode()[0]

# impute the null values in the column with the mode

df3["Engine.Type"].fillna(mode_engine_type, inplace=True)
```

In [21]:
```
#check for updated null values dataframe
df3.isna().sum()
```

Out[21]:
```
Investigation.Type        0
Event.Date                0
Location                  0
Country                   0
Injury.Severity           0
Aircraft.damage           0
Aircraft.Category         0
Make                      0
Model                     0
Amateur.Built             0
Number.of.Engines         0
Engine.Type               0
Purpose.of.flight         0
Total.Fatal.Injuries      0
Total.Serious.Injuries    0
Total.Minor.Injuries      0
Total.Uninjured           0
Weather.Condition         0
Report.Status             0
dtype: int64
```

In [22]:
```
#Checking the data types of updated data
df3.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 15125 entries, 20020909X01562 to 20221109106272
Data columns (total 19 columns):
 #   Column                 Non-Null Count  Dtype  
---  ------                 --------------  -----  
 0   Investigation.Type     15125 non-null  object 
 1   Event.Date             15125 non-null  object 
 2   Location               15125 non-null  object 
 3   Country                15125 non-null  object 
 4   Injury.Severity        15125 non-null  object 
 5   Aircraft.damage        15125 non-null  object 
 6   Aircraft.Category      15125 non-null  object 
 7   Make                   15125 non-null  object 
 8   Model                  15125 non-null  object 
 9   Amateur.Built          15125 non-null  object 
 10  Number.of.Engines      15125 non-null  float64
 11  Engine.Type            15125 non-null  object 
 12  Purpose.of.flight      15125 non-null  object 
 13  Total.Fatal.Injuries   15125 non-null  float64
 14  Total.Serious.Injuries 15125 non-null  float64
 15  Total.Minor.Injuries   15125 non-null  float64
 16  Total.Uninjured        15125 non-null  float64
 17  Weather.Condition      15125 non-null  object 
 18  Report.Status          15125 non-null  object 
dtypes: float64(5), object(14)
memory usage: 2.8+ MB
```

```
The "Event.Date" column is categorized as an object data type instead of Datetime format.
```

In [23]:
```
# Make a copy of the data
df3 = df3.copy()

# Change the 'Event.Date' column to datetime Format
df3['Event.Date'] = pd.to_datetime(df3['Event.Date'])

#Check if the data type of the column has updated
df3['Event.Date'].dtype
```

Out[23]:
```
dtype('<M8[ns]')
```

In [24]:
```
#Check the count of unique values in the Make column

df3["Make"].value_counts().head(10)
```

Out[24]:
```
CESSNA              2576
Cessna              2112
PIPER               1654
Piper               1213
BEECH                598
Beech                431
MOONEY               173
Mooney               147
CIRRUS DESIGN CORP   135
BELLANCA             116
Name: Make, dtype: int64
```

From the above, the records in the 'Make' column need to be converted to the same case.

In [25]:
```
# Convert the string values in the "Make" column to lower case then capitalize
df3['Make'] = df3['Make'].str.lower()

df3['Make'] = df3['Make'].str.capitalize()

# Counts unique values and show the top 10
unique_counts = df3['Make'].value_counts().head(10)
print(unique_counts)
```

```
Cessna                4688
Piper                 2867
Beech                 1029
Mooney                 320
Bellanca               212
Maule                  193
Aeronca                166
Cirrus design corp     137
Luscombe               134
Champion               133
Name: Make, dtype: int64
```

In [26]:
```
# Create a new feature "Year"
df3["Year"] = df3["Event.Date"].dt.year

#Create a new feature 'Abbreviation'
df3['Abbreviation'] = df3['Location'].str.split(',').str[1].str.strip()

df3.head(2)
```

Out[26]:

	Investigation.Type	Event.Date	Location	Country	Injury.Severity	Aircraft.damage	Aircraft.Category	Make	Model	Am
Event.Id										
20020909X01562	Accident	1982-01-01	PULLMAN, WA	United States	Non-Fatal	Substantial	Airplane	Cessna	140	
20020909X01561	Accident	1982-01-01	EAST HANOVER, NJ	United States	Non-Fatal	Substantial	Airplane	Cessna	401B	

2 rows × 21 columns

In [50]:
```
# Merging the Aviation DataFrame and State_codes DataFrame
df3 = df3.copy()

df3 = df3.merge(State_Codes, on = "Abbreviation")

df3["Abbreviation"].head(3)
```

Out[50]:
```
0    WA
1    WA
2    WA
Name: Abbreviation, dtype: object
```

In [28]:
```
# Get the name of the last column
last_col = df3.columns[-1]

# Remove the last column and place it in the 3rd position (index 2)
cols = list(df3.columns)
cols.remove(last_col)
cols.insert(3, last_col)

# Reorder the DataFrame columns
df3 = df3[cols]

# Display the top 2 rows of the modified DataFrame
df3.head(2)
```

Out[28]:

	Investigation.Type	Event.Date	Location	US_State	Country	Injury.Severity	Aircraft.damage	Aircraft.Category	Make	Model	...	
0	Accident	1982-01-01	PULLMAN, WA	Washington	United States	Non-Fatal	Substantial	Airplane	Cessna	140	...	F
1	Accident	1982-01-08	PULLMAN, WA	Washington	United States	Non-Fatal	Substantial	Airplane	Cessna	TU206G	...	F

2 rows × 22 columns

In [29]:
```
# Group the data by 'Investigation.Type' and aggregate the 4 Injuries columns
df3.groupby("Investigation.Type")[["Total.Serious.Injuries","Total.Minor.Injuries",
                                   "Total.Fatal.Injuries", "Total.Uninjured"]].sum()
```

Out[29]:

Investigation.Type	Total.Serious.Injuries	Total.Minor.Injuries	Total.Fatal.Injuries	Total.Uninjured
Accident	3290.0	2808.0	4467.0	16805.0
Incident	2.0	2.0	0.0	326.0

From the above, it is clear that Incidents do not have any effect on the columns of our metrics of success. So we shall filter the data to have only Accidents on the Investigation type

In [30]:
```
# Filter data to have Accidents only on the "Investigation.Type" column
df3 = df3[df3["Investigation.Type"] == "Accident"]

df3.head(2)
```

Out[30]:

	Investigation.Type	Event.Date	Location	US_State	Country	Injury.Severity	Aircraft.damage	Aircraft.Category	Make	Model	...	
0	Accident	1982-01-01	PULLMAN, WA	Washington	United States	Non-Fatal	Substantial	Airplane	Cessna	140	...	F
1	Accident	1982-01-08	PULLMAN, WA	Washington	United States	Non-Fatal	Substantial	Airplane	Cessna	TU206G	...	F

2 rows × 22 columns

In [31]:
```
df3.shape

print(f"Our clean data has {df3.shape[0]} rows and {df3.shape[1]} columns")
```

```
Our clean data has 14995 rows and 22 columns
```

In [32]:
```
df3.isna().sum()
```

Out[32]:
```
Investigation.Type        0
Event.Date                0
Location                  0
US_State                  0
Country                   0
Injury.Severity           0
Aircraft.damage           0
Aircraft.Category         0
Make                      0
Model                     0
Amateur.Built             0
Number.of.Engines         0
Engine.Type               0
Purpose.of.flight         0
Total.Fatal.Injuries      0
Total.Serious.Injuries    0
Total.Minor.Injuries      0
Total.Uninjured           0
Weather.Condition         0
Report.Status             0
Year                      0
Abbreviation              0
dtype: int64
```

The Data is now clean and ready for EDA

In this EDA section, we shall assign a Safety weightage to the Injury Columns and then calculate a Safety Score that we shall use to understand the various trends in Air Travel Safety over time as well as compare different Airplane Makes and Models based on the Safety Score Calculated

In [33]:
```
# Create a new Feature "Airplane.Make_Model"
df3['Airplane.Make_Model'] = df3['Make'].str.cat(df3['Model'], sep=', ')
df3.head(5)
```

Out[33]:

	Investigation.Type	Event.Date	Location	US_State	Country	Injury.Severity	Aircraft.damage	Aircraft.Category	Make	Model	..
0	Accident	1982-01-01	PULLMAN, WA	Washington	United States	Non-Fatal	Substantial	Airplane	Cessna	140	..
1	Accident	1982-01-08	PULLMAN, WA	Washington	United States	Non-Fatal	Substantial	Airplane	Cessna	TU206G	..
2	Accident	1982-01-18	ORCHARDS, WA	Washington	United States	Non-Fatal	Destroyed	Airplane	Beech	C23	..
3	Accident	1982-02-02	MOSES LAKE, WA	Washington	United States	Non-Fatal	Substantial	Airplane	Cessna	U206G	..
4	Accident	1982-03-07	EVERETT, WA	Washington	United States	Non-Fatal	Destroyed	Airplane	Rockwell intl	114	..

5 rows × 23 columns

In [51]:
```
# Grouping the data by 'Make' and 'Model' and summing up the injury-related columns
df5 = df3.groupby("Airplane.Make_Model")[['Total.Fatal.Injuries', "Total.Serious.Injuries",
                                          "Total.Minor.Injuries"]].sum().reset_index()

# Define Safety weightages for each category of Injury
fatal =  7      #   70% weight to Fatal Injuries
serious = 2     #   20% weight to Serious Injuries
minor =  1      #   10% weight to Minor Injuries

# Create a new feature "Safety_Score"
df5['Safety_Score'] =   (df5['Total.Fatal.Injuries'] * fatal
                           + df5['Total.Serious.Injuries'] * serious
                          + df5['Total.Minor.Injuries'] * minor)

#confirms our Safety Score feature has been created
df5["Safety_Score"].value_counts().sort_values(ascending=False).head()
```

Out[51]:
```
0.0     2151
2.0      624
7.0      472
1.0      467
14.0     243
Name: Safety_Score, dtype: int64
```

In [35]:
```
# Getting the Safest Aircraft
safest_aircraft = df5.sort_values('Safety_Score', ascending=True)

# Display the top 5 safest airplane
safest_aircraft.head()
```

Out[35]:

	Airplane.Make_Model	Total.Fatal.Injuries	Total.Serious.Injuries	Total.Minor.Injuries	Safety_Score
4925	Zwicker murray r, GLASTAR	0.0	0.0	0.0	0.0
3292	Pergerson, Highlander	0.0	0.0	0.0	0.0
1564	Dehavilland, DHC 1	0.0	0.0	0.0	0.0
1563	Dehavilland, DH82A	0.0	0.0	0.0	0.0
1562	Dehavilland, BEAVER U-6A	0.0	0.0	0.0	0.0

In [58]:
```
# Filter to get specific information for Aircraft Make and Model
aircraft_details = df3.loc[df3.set_index('Airplane.Make_Model').index.isin(safest_aircraft.set_index('Airplane.M
                          ['Airplane.Make_Model', 'Number.of.Engines', 'Engine.Type', 'Purpose.of.flight']]

# Filtering for information in 'Purpose.of.flight'
aircraft_details_personal = aircraft_details[aircraft_details['Purpose.of.flight']
                 .isin(['Personal', 'Public Aircraft - Local', "Public Aircraft"])]
# Merging the safety score back into the details DataFrame
aircraft_details_personal = aircraft_details_personal.merge(safest_aircraft[["Airplane.Make_Model", 'Safety_Scor
                                                             on=['Airplane.Make_Model'], how='left')

#Display the top 10
aircraft_details_personal.head()
```

Out[58]:

	Airplane.Make_Model	Number.of.Engines	Engine.Type	Purpose.of.flight	Safety_Score
0	Cessna, 140	1.0	Reciprocating	Personal	126.0
1	Cessna, TU206G	1.0	Reciprocating	Business	14.0
2	Beech, C23	1.0	Reciprocating	Business	86.0
3	Cessna, U206G	1.0	Reciprocating	Business	81.0
4	Rockwell intl, 114	1.0	Reciprocating	Personal	0.0

In [59]:
```
# # Get specific information Airplane makes and models
aircraft_details = df3.loc[df3.set_index("Airplane.Make_Model").index.isin(safest_aircraft.set_index("Airplane.M
                                   ["Airplane.Make_Model", 'Number.of.Engines', 'Engine.Type', 'Purpose.of.flig

# Filtering for Public Aircraft only in "Purpose of Flight"
aircraft_details_public = aircraft_details[aircraft_details['Purpose.of.flight']
                 .isin(['Public Aircraft', 'Public Aircraft - Local'])]

# Merging the safety score back into the details DataFrame
aircraft_details_public = aircraft_details_public.merge(safest_aircraft[["Airplane.Make_Model", 'Safety_Score']]

# Displaying the top 10 safest aircraft along with their safety score
aircraft_details_public = aircraft_details_public.sort_values(by='Safety_Score', ascending=True)

aircraft_details_public.head()
```

Out[59]:

	Airplane.Make_Model	Number.of.Engines	Engine.Type	Purpose.of.flight	Safety_Score
41	Cessna, 404	2.0	Reciprocating	Public Aircraft	0.0
19	Cessna, 0-1A	1.0	Reciprocating	Public Aircraft	0.0
27	Bae systems, MK-67 HAWK	1.0	Turbo Jet	Public Aircraft	0.0
11	Piper, PA 18-125	1.0	Reciprocating	Public Aircraft	0.0
29	Aviat aircraft inc, A 1	1.0	Reciprocating	Public Aircraft - Local	0.0

Creating a Safety_Score Feature in the Initial Aviation Data DataFrame

In [60]:
```
df3["Safety_Score"] = df3["Total.Fatal.Injuries"] * 7 + df3["Total.Serious.Injuries"] * 2 + df3["Total.Minor.Inj
df3.head(2)
```

Out[60]:

	Investigation.Type	Event.Date	Location	US_State_x	Country	Injury.Severity	Aircraft.damage	Aircraft.Category	Make	Model	...
0	Accident	1982-01-01	PULLMAN, WA	Washington	United States	Non-Fatal	Substantial	Airplane	Cessna	140	...
1	Accident	1982-01-08	PULLMAN, WA	Washington	United States	Non-Fatal	Substantial	Airplane	Cessna	TU206G	...

2 rows × 25 columns

In [40]:
```
# Grouping the data by 'State and Weather_Condition' and calculating the Summed Safety Score
abb_safety = df3.groupby(['US_State','Weather.Condition'])[["Safety_Score", "Total.Fatal.Injuries", "Total.Serio
                                                        "Total.Minor.Injuries"]].agg("sum").reset_index()

# Identify risky areas (those with higher Safety Scores)
risky_areas = abb_safety.sort_values('Safety_Score', ascending=False)

# Display top risky areas
risky_areas.head(5)
```

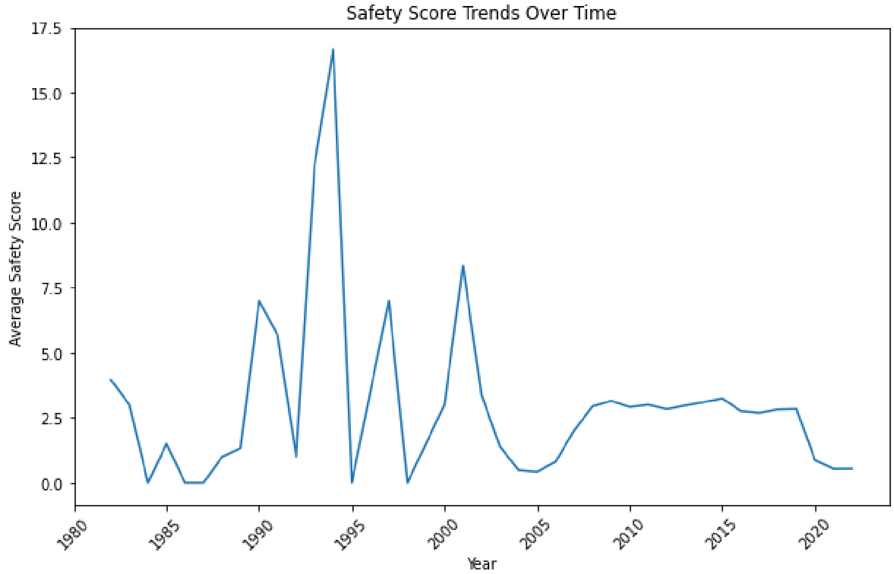
Out[40]:

	US_State	Weather.Condition	Safety_Score	Total.Fatal.Injuries	Total.Serious.Injuries	Total.Minor.Injuries
17	California	VMC	3767.0	412.0	314.0	255.0
119	Texas	VMC	2803.0	282.0	280.0	269.0
27	Florida	VMC	2376.0	251.0	211.0	197.0
9	Arizona	VMC	1314.0	154.0	76.0	84.0
5	Alaska	VMC	1227.0	124.0	131.0	97.0

Plotting a Line Plot of Safety Score Vs Year

In [61]:
```
# Grouping by year to calculate Ave. Safety_Score
yearly_safety = df3.groupby('Year')['Safety_Score'].mean().reset_index()

# Plotting line plot
plt.figure(figsize=(10,6))
sns.lineplot(x='Year', y='Safety_Score', data=yearly_safety)
plt.title('Safety Score Trends Over Time')
plt.xlabel('Year')
plt.ylabel('Average Safety Score')
plt.xticks(rotation=45)
plt.show()
```

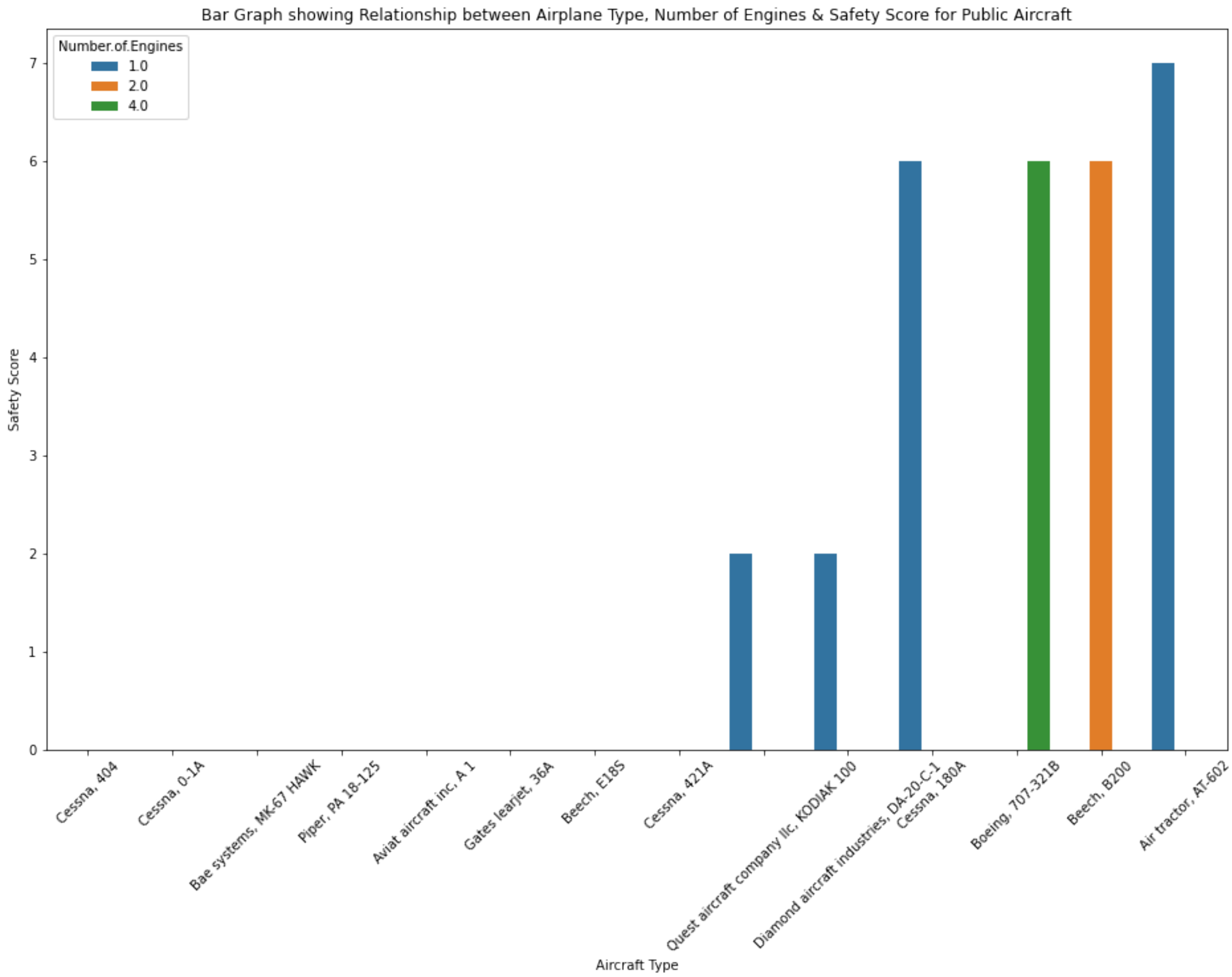



```
This line plot suggests that air travel safety has improved over time. Between 1982 and around 2002, there were
fluctuations in safety, but from 2002 to 2023, there has been a marked improvement. This improvement could be
attributed to technological advancements that have enhanced the safety of air travel
```

Type *Markdown* and LaTeX: α^2

Plotting a bar graph showing the Relationship between Safety Score for Public Aircraft and Number of Engines Associated with it

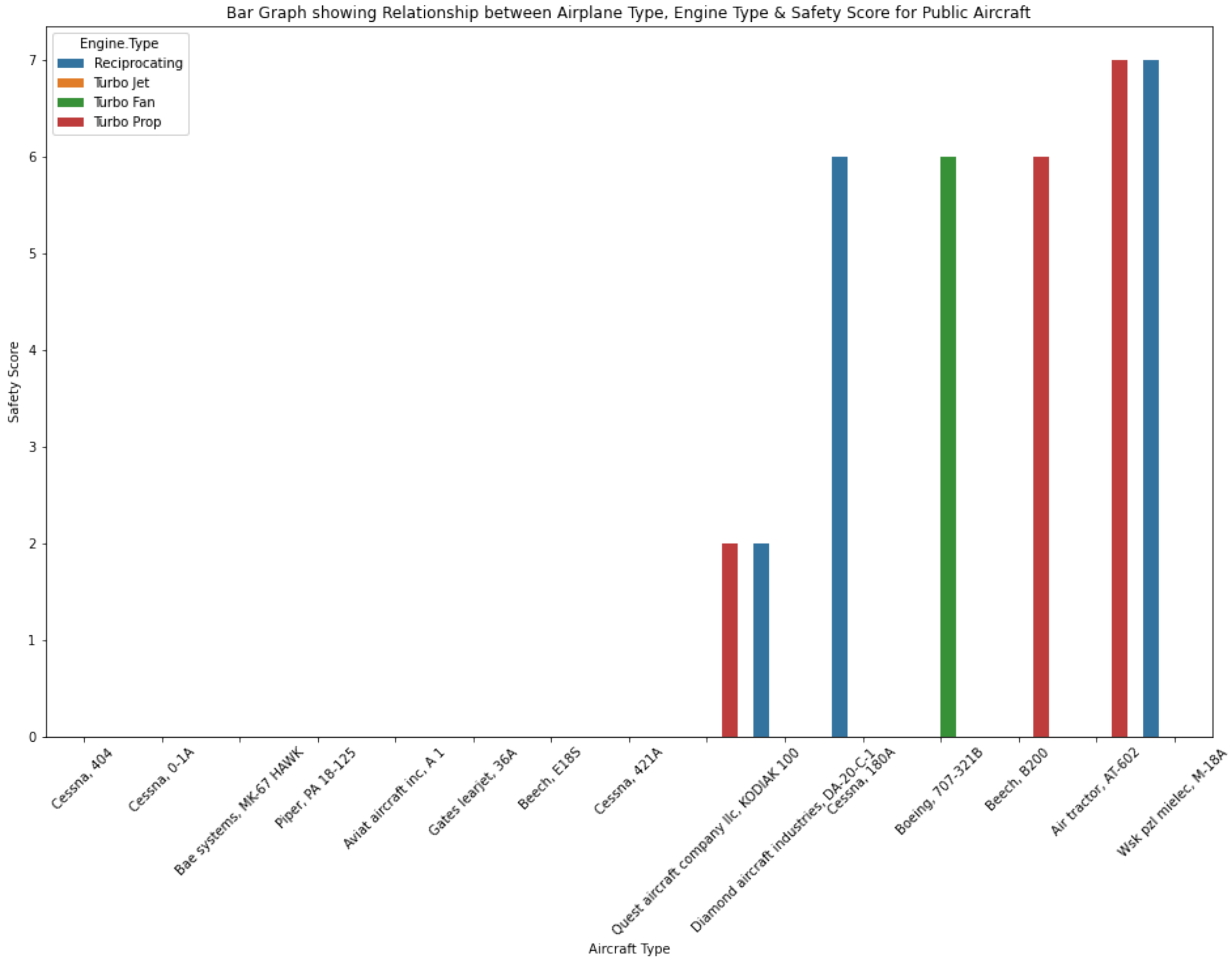
In [43]:
```
# Relationship between Aircraft Type, Number of Engines and Safety Score for Public Aircraft
plt.figure(figsize=(16,10))
sns.barplot(x='Airplane.Make_Model', y='Safety_Score', hue = "Number.of.Engines", data = aircraft_details_public
plt.title('Bar Graph showing Relationship between Airplane Type, Number of Engines & Safety Score for Public Air
plt.xlabel('Aircraft Type')
plt.ylabel('Safety Score')
plt.xticks(rotation=45)
plt.show()
```




```
The plot above shows the Seven safest Aircraft Make and Model by Safety Score which is zero. It also shows that
the most common and preferred Number of Engines for the Public Aircraft is 1.
```

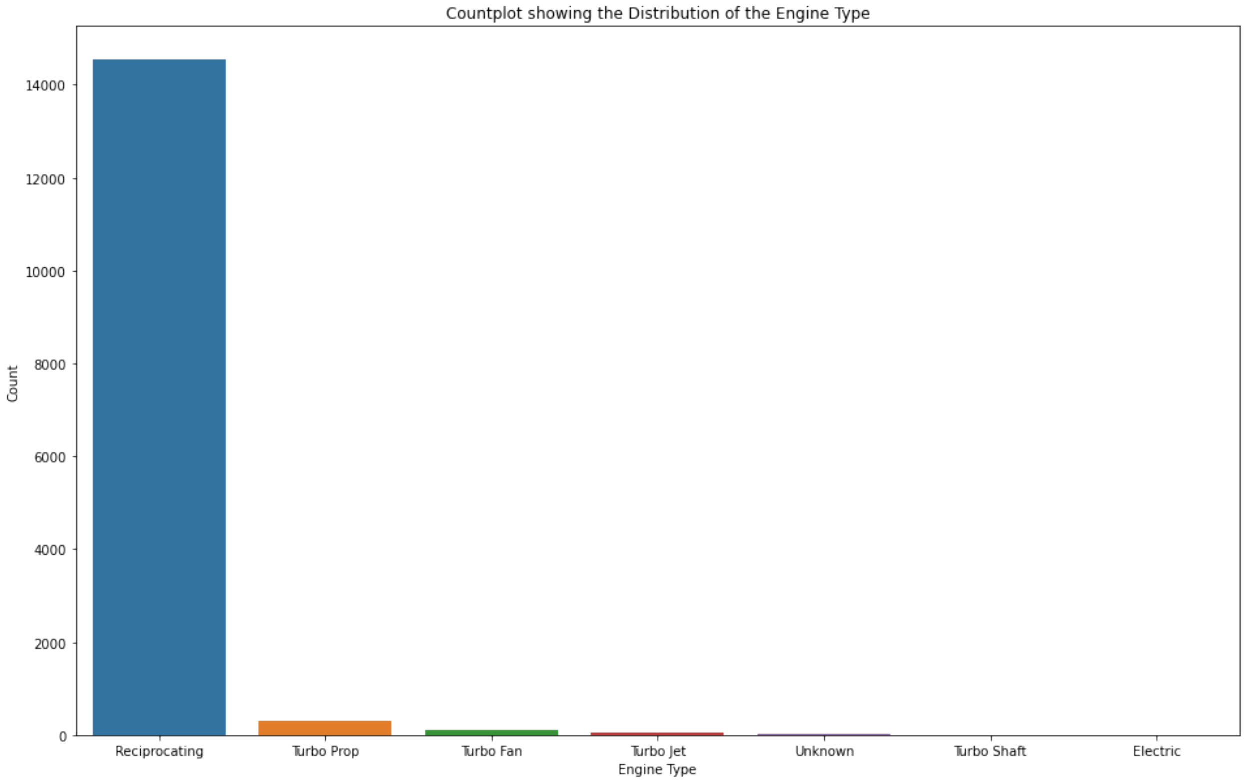
Plot showing Relationship between Aircraft Type, Engine Type and Safety Score

In [45]:
```
# Relationship between Aircraft Type, Engine type and Safety Score for Public Aircraft
plt.figure(figsize=(16,10))
sns.barplot(x='Airplane.Make_Model', y='Safety_Score', hue = "Engine.Type", data=aircraft_details_public.head(15
plt.title('Bar Graph showing Relationship between Airplane Type, Engine Type & Safety Score for Public Aircraft'
plt.xlabel('Aircraft Type')
plt.ylabel('Safety Score')
plt.xticks(rotation=45)
plt.show()
```



Countplot showing the distribution of Engine Type

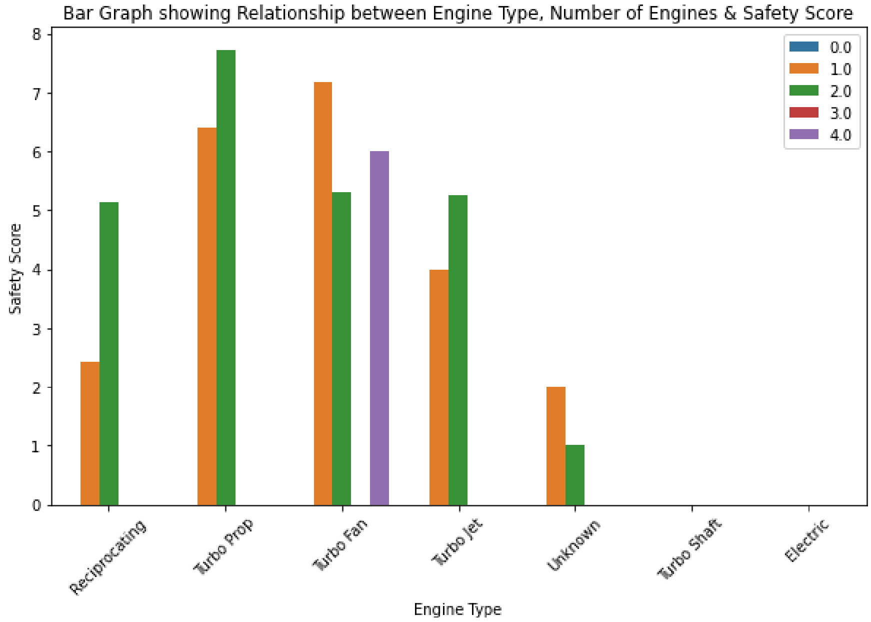
In [46]:
```
# Countplot showing the Distribution of the Engine Type for
plt.figure(figsize=(16,10))
sns.countplot(x='Engine.Type', data=df3)
plt.title('Countplot showing the Distribution of the Engine Type')
plt.xlabel('Engine Type')
plt.ylabel('Count')
plt.show()
```



This plot shows that the most common preferred Engine type for Aircraft is 1

Plot showing relationship between Engine Type, Number of Engines and Safety Score

In [49]:
```
# Relationship between engine type, Number of Engines and Safety Score
plt.figure(figsize=(10,6))
sns.barplot(x='Engine.Type', y='Safety_Score', hue= "Number.of.Engines", data=df3, ci = None)
plt.legend(loc="upper right")
plt.title('Bar Graph showing Relationship between Engine Type, Number of Engines & Safety Score')
plt.xlabel('Engine Type')
plt.ylabel('Safety Score')
plt.xticks(rotation=45)
plt.show()
```



The plot above shows that the most preferred Engine type is reciprocating with either 1 or 2 Engines because of the low safety scores associated with it. This means that an Airplane with an Reciprocating Engine type and 1 engine is very safe

The final insights from the Analysis done are as below:

- The Safest Airplane to purchase for Public-Local purposes is Beech E18S. Based on the Safety Metrics, this Airplane has had only one accident which was not Fatal. No Fatal, Serious or Minor Injuries were recorded. Additionally, the last accident happened in 2009 highlighting the safety measures the Aircraft manufacturers have deployed over the years to ensure the aircraft is safe. This Aircraft has 2 Engines which are of Reciprocating type, which is the most common type of Engine for Aircraft.
- The Safest Airplane for Personal/Private purposes for the company to purchase is Diamond Aircraft ind inc, DA 20 C1. The aircraft has had only one accident in the last 20 years which did not have any Fatal or Serious Injury. Furthermore, after the accident, the aircraft had a minor damage which highlights the manufacturer's high regard for Safety of the aircraft in the Manufacturing Process. Since this aircraft is for private purposes, it has one engine of Reciprocating type.
- The Safest Airplane to purchase for Public international purposes is a Cessna 421A. This Aircraft has had only one accident in the last 30 years. This being a Public Aircraft, it has two engines of Reciprocating type. Despite the Aircraft having substantial damage after the accident, it had a Safety Score of zero, meaning no cases of injuries were reported.
- It is important to note that all the Aircrafts recommended above are not Amateur built, hence the company should consider aircraft that are not Amateur built due to their high standards of safety
- Since air safety is mostly influenced by the type of weather in the routes the aircraft is flying, the type of local weather that experienced few accidents and which we can say was favourable for air safety is IMC type of weather. So the company could consider Local flights in the regions that have this type of weather.

In [ ]: