Final Project Submission

Please fill out:

- Student name: Felix Limo
- Student pace: part time
- Scheduled project review date/time: 09/09/2024
- Instructor name: William Okomba
- Blog post URL:https://github.com/Felix-87/phase_1_project.git (https://github.com/Felix-87/phase_1_project.git)

1.0 Project Overview

1.1 Introduction

The project aims at drawing insights from the NTSB dataset to determine the kind of aircraft to purchase and operate, commercial and private enterprises based on the potential risks of different aircrafts. The criteria is finding the aircraft with the lowest risk to recommend. This project will, therefore adopt Cross Industry Standard Prcocedures- Data Mining(CRISP-DM) for the aviation industry.

2.0 Business Understanding

2.1 Objective

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. You are charged with determining which aircraft are the lowest risk for the company to start this new business endeavor. You must then translate your findings into actionable insights that the head of the new aviation division can use to help decide which aircraft to purchase.

2.2 Empirical Summary

Conventionally, the choice of the aircraft to purchase and operate is guided by various factors. This includes the investor budget, plane type and engine size or configurations, interior and layouts, passengers and business requirements, destinations or routes, operational and maintenance costs, return on investment, and regulatory and safety requirements(https://aircraftmaintenancestands.com/blog (https://aircraftmaintenancestands.com/blog), https://skyaviationholdings.com/

3.0 The Data

The data provided for this analysis is from the National Transportation Safety Board(NTSB) database that includes aviation accident data from 1962 to 2023 about civil aviation accidents and selected incidents in the United States and international waters.

Dataset:https://www.kaggle.com/datasets/khsamaha/aviation-accident-database-synopses (https://www.kaggle.com/datasets/khsamaha/aviation-accident-database-synopses)

Importing python libraries

In [89]:
```
#importing relevant python libraries
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

In [90]:
```
# Loading AviationData.csv dataset as data1 dataframe
data1 = pd.read_csv('AviationData.csv', encoding= 'ISO 8859-1')
```

```
c:\Users\Admin\anaconda3\envs\learn-env\lib\site-packages\IPython\core\inter
activeshell.py:3145: DtypeWarning: Columns (6,7,28) have mixed types.Specify
dtype option on import or set low_memory=False.
  has_raised = await self.run_ast_nodes(code_ast.body, cell_name,
```

In [91]:
```
# Loading USState_Codes.csv dataset as data2 dataframe
data2 = pd.read_csv('USState_Codes.csv')
```

3.1 Data Understanding

Preview of data1 dataframe.

Data preview before preparation, serves as familiarization with its features and be able to map out the essential features relevant to the scope of the problem statement. This invokes pertinent questions to draw insights from the data which gives confidence in data-driven decision making that guides business strategic direction.

In [92]:
```
# # Display of the first 5 rows of the dataframe
data1.head()
```

Out[92]:

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

5 rows × 31 columns

In [93]:
```
# Checking dataset information
data1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 88889 entries, 0 to 88888
Data columns (total 31 columns):
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   Event.Id                88889 non-null  object 
 1   Investigation.Type      88889 non-null  object 
 2   Accident.Number         88889 non-null  object 
 3   Event.Date              88889 non-null  object 
 4   Location                88837 non-null  object 
 5   Country                 88663 non-null  object 
 6   Latitude                34382 non-null  object 
 7   Longitude               34373 non-null  object 
 8   Airport.Code            50249 non-null  object 
 9   Airport.Name            52790 non-null  object 
 10  Injury.Severity         87889 non-null  object 
 11  Aircraft.damage         85695 non-null  object 
 12  Aircraft.Category       32287 non-null  object 
 13  Registration.Number     87572 non-null  object 
 14  Make                    88826 non-null  object 
 15  Model                   88797 non-null  object 
 16  Amateur.Built           88787 non-null  object 
 17  Number.of.Engines       82805 non-null  float64
 18  Engine.Type             81812 non-null  object 
 19  FAR.Description         32023 non-null  object 
 20  Schedule                12582 non-null  object 
 21  Purpose.of.flight       82697 non-null  object 
 22  Air.carrier             16648 non-null  object 
 23  Total.Fatal.Injuries    77488 non-null  float64
 24  Total.Serious.Injuries  76379 non-null  float64
 25  Total.Minor.Injuries    76956 non-null  float64
 26  Total.Uninjured         82977 non-null  float64
 27  Weather.Condition       84397 non-null  object 
 28  Broad.phase.of.flight   61724 non-null  object 
 29  Report.Status           82508 non-null  object 
 30  Publication.Date        75118 non-null  object 
dtypes: float64(5), object(26)
memory usage: 21.0+ MB
```

In [94]:
```
# Checking features to note the essential ones to answer research question
data1.columns
```

Out[94]:
```
Index(['Event.Id', 'Investigation.Type', 'Accident.Number', 'Event.Date',
       'Location', 'Country', 'Latitude', 'Longitude', 'Airport.Code',
       'Airport.Name', 'Injury.Severity', 'Aircraft.damage',
       'Aircraft.Category', 'Registration.Number', 'Make', 'Model',
       'Amateur.Built', 'Number.of.Engines', 'Engine.Type', 'FAR.Descriptio
n',
       'Schedule', 'Purpose.of.flight', 'Air.carrier', 'Total.Fatal.Injurie
s',
       'Total.Serious.Injuries', 'Total.Minor.Injuries', 'Total.Uninjured',
       'Weather.Condition', 'Broad.phase.of.flight', 'Report.Status',
       'Publication.Date'],
      dtype='object')
```

In [95]:
```
# Concise summary (numerical features)
data1.describe().T
```

Out[95]:

	count	mean	std	min	25%	50%	75%	max
Number.of.Engines	82805.0	1.146585	0.446510	0.0	1.0	1.0	1.0	8.0
Total.Fatal.Injuries	77488.0	0.647855	5.485960	0.0	0.0	0.0	0.0	349.0
Total.Serious.Injuries	76379.0	0.279881	1.544084	0.0	0.0	0.0	0.0	161.0
Total.Minor.Injuries	76956.0	0.357061	2.235625	0.0	0.0	0.0	0.0	380.0
Total.Uninjured	82977.0	5.325440	27.913634	0.0	0.0	1.0	2.0	699.0

In [96]:
```
# Summary of categoricals features
data1.describe(include='object').T
```

Out[96]:

	count	unique	top	freq
Event.Id	88889	87951	20001212X19172	3
Investigation.Type	88889	2	Accident	85015
Accident.Number	88889	88863	WPR22FA309	2
Event.Date	88889	14782	2000-07-08	25
Location	88837	27758	ANCHORAGE, AK	434
Country	88663	219	United States	82248
Latitude	34382	25592	332739N	19
Longitude	34373	27156	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

In [97]:
```
# shape of the dataframe (rows, cols)
data1.shape
```

Out[97]: (88889, 31)

In [98]:
```
for column in data1:
  unique_values = data1[column].unique()
  print(f"Unique values in column '{column}','\n': {unique_values}",'\n')
```

```
Unique values in column 'Event.Id','
': ['20001218X45444' '20001218X45447' '20061025X01555' ... '2022122710649
7'
 '20221227106498' '20221230106513']

Unique values in column 'Investigation.Type','
': ['Accident' 'Incident']

Unique values in column 'Accident.Number','
': ['SEA87LA080' 'LAX94LA336' 'NYC07LA005' ... 'WPR23LA075' 'WPR23LA076'
 'ERA23LA097']

Unique values in column 'Event.Date','
': ['1948-10-24' '1962-07-19' '1974-08-30' ... '2022-12-22' '2022-12-26'
 '2022-12-29']

Unique values in column 'Location','
': ['MOOSE CREEK, ID' 'BRIDGEPORT, CA' 'Saltville, VA' ... 'San Manual, A
Z'
```

Preview of data2 dataframe

In [99]:
```
# preview data in data2 dataframe
data2.head()
```

Out[99]:

	US_State	Abbreviation
0	Alabama	AL
1	Alaska	AK
2	Arizona	AZ
3	Arkansas	AR
4	California	CA

In [100]:
```
# Checking data information
data2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 62 entries, 0 to 61
Data columns (total 2 columns):
 #   Column        Non-Null Count  Dtype 
---  ------        --------------  ----- 
 0   US_State      62 non-null     object
 1   Abbreviation  62 non-null     object
dtypes: object(2)
memory usage: 1.1+ KB
```

Observation: The dataframe has all feature as object dtype

3.2 Refining Problem Statement.

My company contemplates a dive into aviation industry but limited in the knowledge and experience in the sector. Conventionally a number of factors are considered in settling on the aircrafts to purchase and operate as seen in the empirical review summary above. The objective therefore, is to able to recommend on the kind of aircraft to invest in based on the scope set out.

Research objective: To identify the aircraft with the lowest risk to purchase and operate.

Specific Objectives:

1. To extract data on the aicrafts operated for purposes of business or private enterprises
2. To extract data on the aircrafts that sustained the lowest degree of damage in the event of accident/incident
3. To extract data on the aircrafts that did not inflict injury to users during accident/incident
4. To select the aircraft Make and Purpose with the lowest combine risk.

Scope The scope of this research is limited to the dataset provided.

Assumptions That USState_Codes.csv is not critical in this analysis as observed at data preview,hence set aside.

3.2.1 Metrics of Success

My project will be successful if, using the provided data and scope, be able to find the aircraft of commercial and private enterprises with the lowest risks and make recommend to aid the investment decision making. This will be guided by the formulated research questions on the criteria of selection as defined by the scope of business case and the provided datasets.

*1. Criteria 1; Which aircrafts are operated for business or private enterprises?

*2. Criteria 2; Asset Risk: Which aircrafts lowest asset risk? This is the potential loss of aircraft in the event of an accident. Selection will be based on the degree of damage sustained.

*3. Criteria 3; User related risk: Which aircraft posses lowest risk to users(crew and passengers) in the event of ana accident. Selection will be based on the levels of injuries inflicted on users.

*4. Criteria 4; Ranking based on a combination of lowest risk category on asset risk and user risk, and be able to pick the aircraft Make and Purpose with the lowest risk.Achieved by use of visualization.

4.0 Data Preparation

4.1 Data Cleaning

This phase involves checking on data validity(relevance), accuracy(removal of outliers), completeness...

and treatment of missing values and duplicates. Duplicates are removed while missing values are either dropped/deleted if by so doing do not significantly impact on the clean dataset, or values imputed.

In [101]:
```
# Making a copy of the dataset
df = data1.copy(deep= True)
```

In [102]:
```
# Checking columns
df.columns
```

Out[102]:
```
Index(['Event.Id', 'Investigation.Type', 'Accident.Number', 'Event.Date',
       'Location', 'Country', 'Latitude', 'Longitude', 'Airport.Code',
       'Airport.Name', 'Injury.Severity', 'Aircraft.damage',
       'Aircraft.Category', 'Registration.Number', 'Make', 'Model',
       'Amateur.Built', 'Number.of.Engines', 'Engine.Type', 'FAR.Descriptio
n',
       'Schedule', 'Purpose.of.flight', 'Air.carrier', 'Total.Fatal.Injurie
s',
       'Total.Serious.Injuries', 'Total.Minor.Injuries', 'Total.Uninjured',
       'Weather.Condition', 'Broad.phase.of.flight', 'Report.Status',
       'Publication.Date'],
      dtype='object')
```

4.1.1 Validity check

This achieved by checking irrelevant features and removing them or selecting the revelant features

In [103]:
```
df.columns
```

Out[103]:
```
Index(['Event.Id', 'Investigation.Type', 'Accident.Number', 'Event.Date',
       'Location', 'Country', 'Latitude', 'Longitude', 'Airport.Code',
       'Airport.Name', 'Injury.Severity', 'Aircraft.damage',
       'Aircraft.Category', 'Registration.Number', 'Make', 'Model',
       'Amateur.Built', 'Number.of.Engines', 'Engine.Type', 'FAR.Descriptio
n',
       'Schedule', 'Purpose.of.flight', 'Air.carrier', 'Total.Fatal.Injurie
s',
       'Total.Serious.Injuries', 'Total.Minor.Injuries', 'Total.Uninjured',
       'Weather.Condition', 'Broad.phase.of.flight', 'Report.Status',
       'Publication.Date'],
      dtype='object')
```

In [104]:
```
#Selecting the relevant features for analysis
df1 = df[['Event.Date','Investigation.Type','Location','Injury.Severity','Air
df1.head(2)
```

Out[104]:

	Event.Date	Investigation.Type	Location	Injury.Severity	Aircraft.damage	Aircraft.Catego
0	1948-10-24	Accident	MOOSE CREEK, ID	Fatal(2)	Destroyed	Na
1	1962-07-19	Accident	BRIDGEPORT, CA	Fatal(4)	Destroyed	Na

In [105]:
```
#Changing columns to lower case and removing white spaces for uniformity
df1.columns = df1.columns.str.lower().str.replace('.', '_')
df1.columns
```

Out[105]:
```
Index(['event_date', 'investigation_type', 'location', 'injury_severity',
       'aircraft_damage', 'aircraft_category', 'make', 'model',
       'purpose_of_flight', 'total_fatal_injuries', 'total_serious_injurie
s',
       'total_minor_injuries', 'total_uninjured', 'broad_phase_of_flight'],
      dtype='object')
```

In [106]:
```
#Rename 'broad.phase.of.flight' column as 'phase.of.flight'
df1.rename(columns = {'broad.phase.of.flight': 'phase.of.flight'}, inplace = 
```

```
c:\Users\Admin\anaconda3\envs\learn-env\lib\site-packages\pandas\core\frame.
py:4296: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/
stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pand
as.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-v
ersus-a-copy)
  return super().rename(
```

In [107]:
```
df1.columns
```

Out[107]:
```
Index(['event_date', 'investigation_type', 'location', 'injury_severity',
       'aircraft_damage', 'aircraft_category', 'make', 'model',
       'purpose_of_flight', 'total_fatal_injuries', 'total_serious_injurie
s',
       'total_minor_injuries', 'total_uninjured', 'broad_phase_of_flight'],
      dtype='object')
```

In [108]:
```
df1.dtypes
```

Out[108]:
```
event_date                 object
investigation_type         object
location                   object
injury_severity            object
aircraft_damage            object
aircraft_category          object
make                       object
model                      object
purpose_of_flight          object
total_fatal_injuries      float64
total_serious_injuries    float64
total_minor_injuries      float64
total_uninjured           float64
broad_phase_of_flight      object
dtype: object
```

4.1.2 Data completeness

Checking for missing values and treating them. Missing values are either dropped/deleted if by so doing do not significantly impact on the clean dataset, or values imputed.

In [109]:
```
#Checking for missing values
df1.isna().sum()
```

Out[109]:
```
event_date                    0
investigation_type            0
location                     52
injury_severity            1000
aircraft_damage            3194
aircraft_category         56602
make                         63
model                        92
purpose_of_flight          6192
total_fatal_injuries      11401
total_serious_injuries    12510
total_minor_injuries      11933
total_uninjured            5912
broad_phase_of_flight     27165
dtype: int64
```

In [110]:
```
df1['aircraft_category'].value_counts()
```

Out[110]:
```
Airplane             27617
Helicopter            3440
Glider                 508
Balloon                231
Gyrocraft              173
Weight-Shift           161
Powered Parachute       91
Ultralight              30
Unknown                 14
WSFT                     9
Powered-Lift             5
Blimp                    4
UNK                      2
ULTR                     1
Rocket                   1
Name: aircraft_category, dtype: int64
```

In [111]:
```
df1['broad_phase_of_flight'].value_counts()
```

Out[111]:
```
Landing        15428
Takeoff        12493
Cruise         10269
Maneuvering     8144
Approach        6546
Climb           2034
Taxi            1958
Descent         1887
Go-around       1353
Standing         945
Unknown          548
Other            119
Name: broad_phase_of_flight, dtype: int64
```

In [112]:
```
# Fill missing values in aircraft_category and broad_phase_of_flight features
df1[['aircraft_category', 'broad_phase_of_flight', 'purpose_of_flight']] = df
```

```
c:\Users\Admin\anaconda3\envs\learn-env\lib\site-packages\pandas\core\frame.
py:3065: SettingWithCopyWarning:
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/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pand
as.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-v
ersus-a-copy)
  self[k1] = value[k2]
```

In [113]:
```
df1.isna().sum()
```

Out[113]:
```
event_date                  0
investigation_type          0
location                   52
injury_severity          1000
aircraft_damage          3194
aircraft_category           0
make                       63
model                      92
purpose_of_flight           0
total_fatal_injuries    11401
total_serious_injuries  12510
total_minor_injuries    11933
total_uninjured          5912
broad_phase_of_flight       0
dtype: int64
```

In [114]:
```
df1.describe().T
```

Out[114]:

	count	mean	std	min	25%	50%	75%	max
total_fatal_injuries	77488.0	0.647855	5.485960	0.0	0.0	0.0	0.0	349.0
total_serious_injuries	76379.0	0.279881	1.544084	0.0	0.0	0.0	0.0	161.0
total_minor_injuries	76956.0	0.357061	2.235625	0.0	0.0	0.0	0.0	380.0
total_uninjured	82977.0	5.325440	27.913634	0.0	0.0	1.0	2.0	699.0

In [115]:
```
# Imputing null values in numerical features in dataframe df1 using their med
numerical_features = ['total_fatal_injuries','total_serious_injuries','total_

#Calculating their medians
medians = df1[numerical_features].median()

# Filling null value with their feature median
df1[numerical_features] = df1[numerical_features].fillna(medians)
```

```
c:\Users\Admin\anaconda3\envs\learn-env\lib\site-packages\pandas\core\frame.
py:3065: SettingWithCopyWarning:
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/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pand
as.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-v
ersus-a-copy)
  self[k1] = value[k2]
```

In [116]:
```
df1.describe().T
```

Out[116]:

	count	mean	std	min	25%	50%	75%	max
total_fatal_injuries	88889.0	0.564761	5.126649	0.0	0.0	0.0	0.0	349.0
total_serious_injuries	88889.0	0.240491	1.434614	0.0	0.0	0.0	0.0	161.0
total_minor_injuries	88889.0	0.309127	2.083715	0.0	0.0	0.0	0.0	380.0
total_uninjured	88889.0	5.037755	26.990914	0.0	0.0	1.0	2.0	699.0

In [117]:
```
df1.shape
```

Out[117]:
```
(88889, 14)
```

In [118]:
```
df1.isna().sum()
```

Out[118]:
```
event_date                   0
investigation_type           0
location                    52
injury_severity           1000
aircraft_damage           3194
aircraft_category            0
make                        63
model                       92
purpose_of_flight            0
total_fatal_injuries         0
total_serious_injuries       0
total_minor_injuries         0
total_uninjured              0
broad_phase_of_flight        0
dtype: int64
```

In [119]:
```
# Missing values in other features are significantly low and can be dropped
df1.dropna(inplace=True)
```

```
<ipython-input-119-516f9ae57843>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  df1.dropna(inplace=True)
```

In [120]:
```
df1['aircraft_category'].value_counts()
```

Out[120]:
```
Unknown              54759
Airplane             25901
Helicopter            3307
Glider                 505
Gyrocraft              173
Weight-Shift           160
Balloon                135
Powered Parachute       88
Ultralight              29
WSFT                     9
Powered-Lift             4
Blimp                    4
ULTR                     1
Rocket                   1
Name: aircraft_category, dtype: int64
```

In [121]:
```
# Unknown values has highest frequency hence will distort data outcome
# Remove Unknown values in aircraft_category feature
df1=df1[df1['aircraft_category'] != 'Unknown']
```

In [122]:
```
df1['aircraft_category'].value_counts()
```

Out[122]:
```
Airplane             25901
Helicopter            3307
Glider                 505
Gyrocraft              173
Weight-Shift           160
Balloon                135
Powered Parachute       88
Ultralight              29
WSFT                     9
Powered-Lift             4
Blimp                    4
ULTR                     1
Rocket                   1
Name: aircraft_category, dtype: int64
```

In [123]:
```
df1.isna().sum()
```

Out[123]:
```
event_date               0
investigation_type       0
location                 0
injury_severity          0
aircraft_damage          0
aircraft_category        0
make                     0
model                    0
purpose_of_flight        0
total_fatal_injuries     0
total_serious_injuries   0
total_minor_injuries     0
total_uninjured          0
broad_phase_of_flight    0
dtype: int64
```

In [124]:
```
df1.head()
```

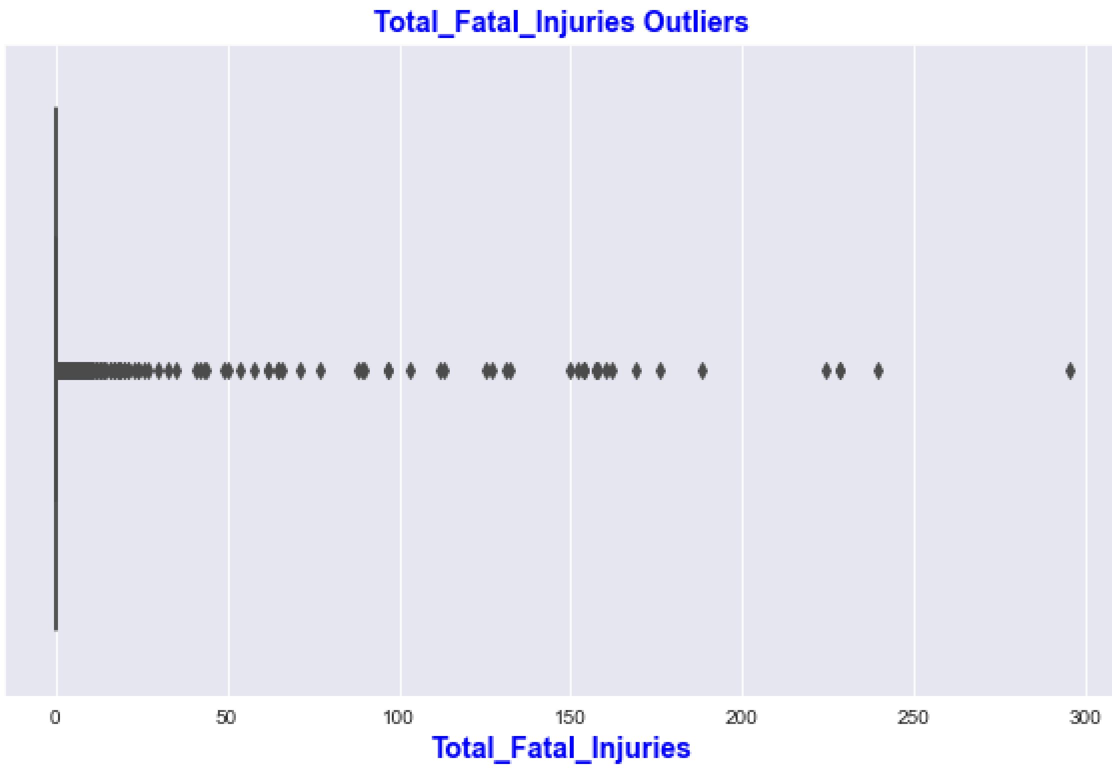
Out[124]:

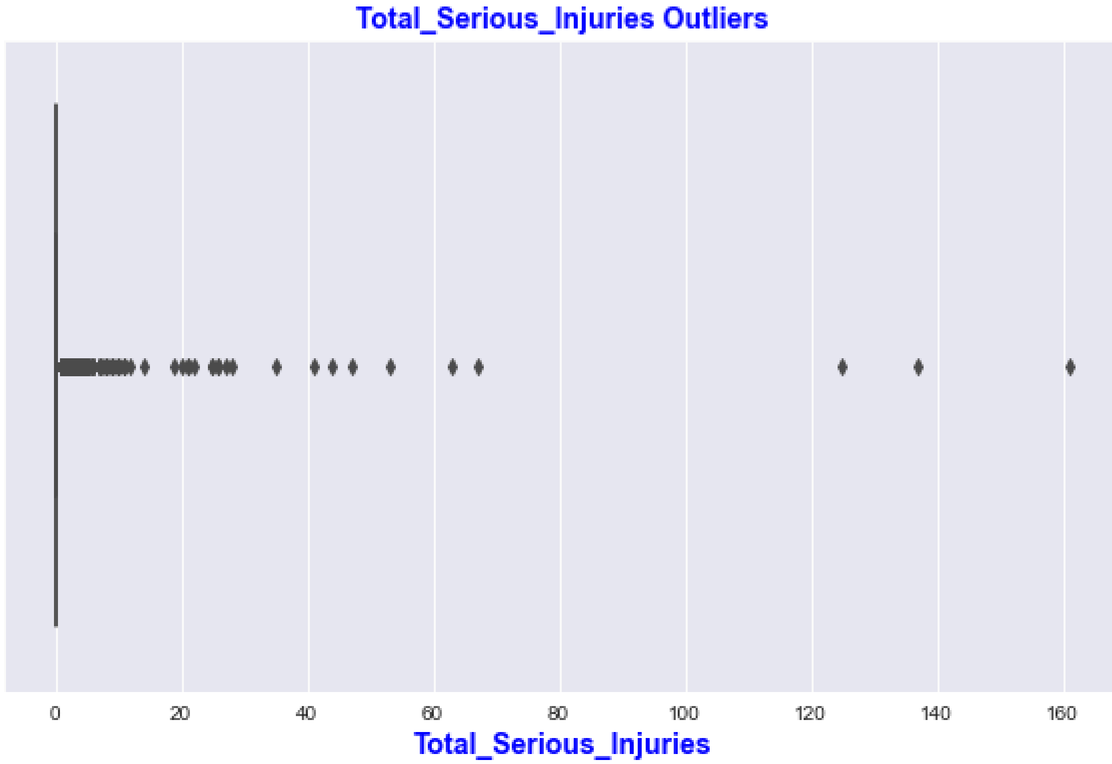
	event_date	investigation_type	location	injury_severity	aircraft_damage	aircraft_category
5	1979-09-17	Accident	BOSTON, MA	Non-Fatal	Substantial	Airplane
7	1982-01-01	Accident	PULLMAN, WA	Non-Fatal	Substantial	Airplane
8	1982-01-01	Accident	EAST HANOVER, NJ	Non-Fatal	Substantial	Airplane
12	1982-01-02	Accident	HOMER, LA	Non-Fatal	Destroyed	Airplane
13	1982-01-02	Accident	HEARNE, TX	Fatal(1)	Destroyed	Airplane

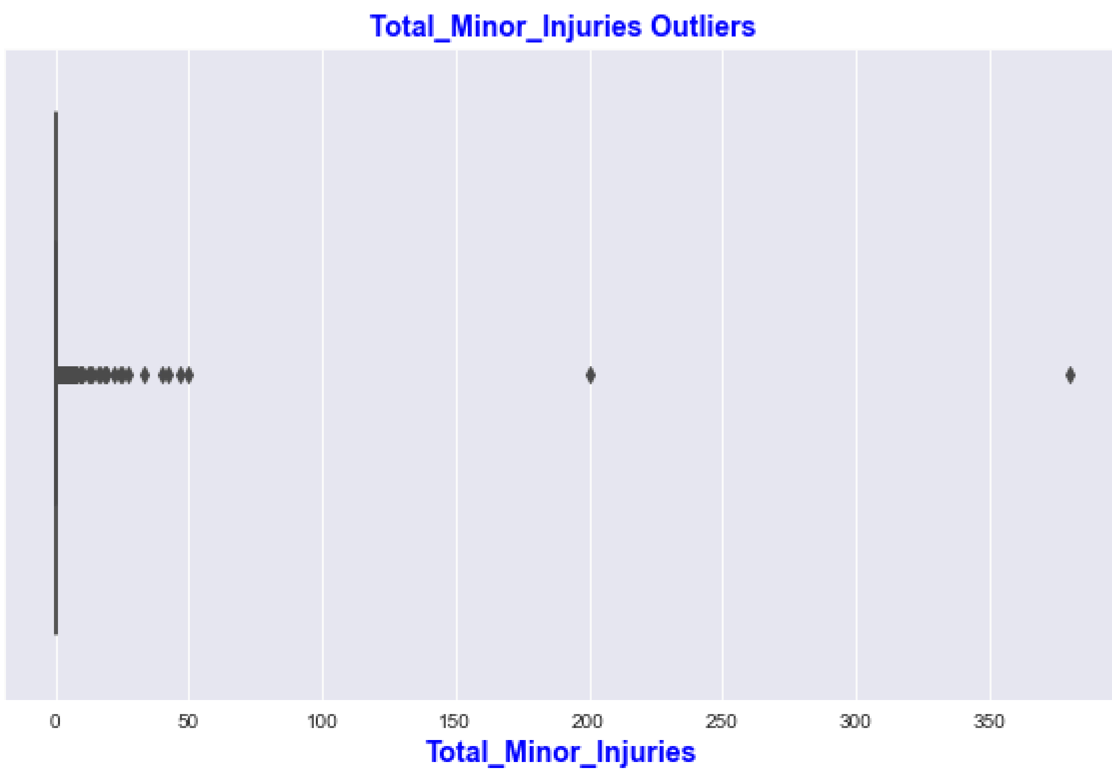
4.1.3 Data accuracy

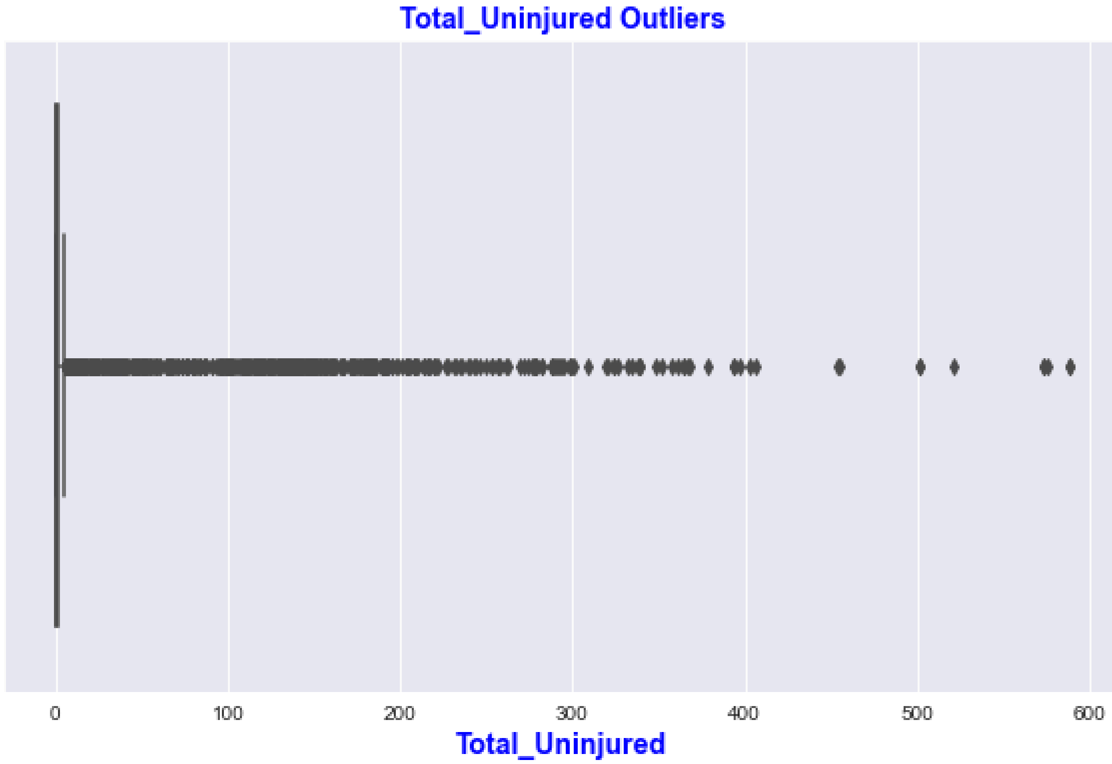
Checking for outlier values in the data that distorts its accuracy. This is mitigated by drop/removing outliers

In [125]:
```
#Checking for outliers visually using plots for numerical features
float_features = df1.select_dtypes(include='float').columns
for feature in float_features:
    plt.figure(figsize= (10,6))
    sns.boxplot(x=df1[feature])

    plt.title(f'{feature} Outliers'.title(), size=14, color='blue', weight='bo
    plt.xlabel(feature.title(), size=14, color='blue', weight='bold')
    plt.show()
```



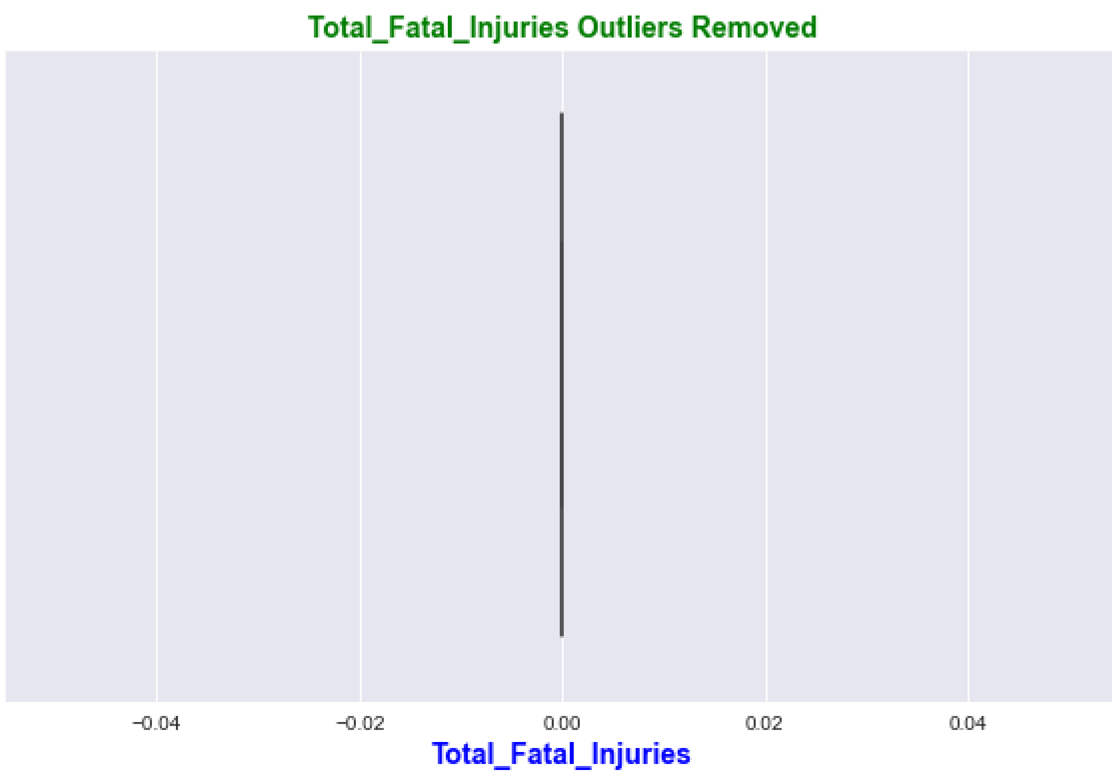

Total_Serious_Injuries Outliers
0
20
40
60
80
100
120
140
160
Total_Serious_Injuries

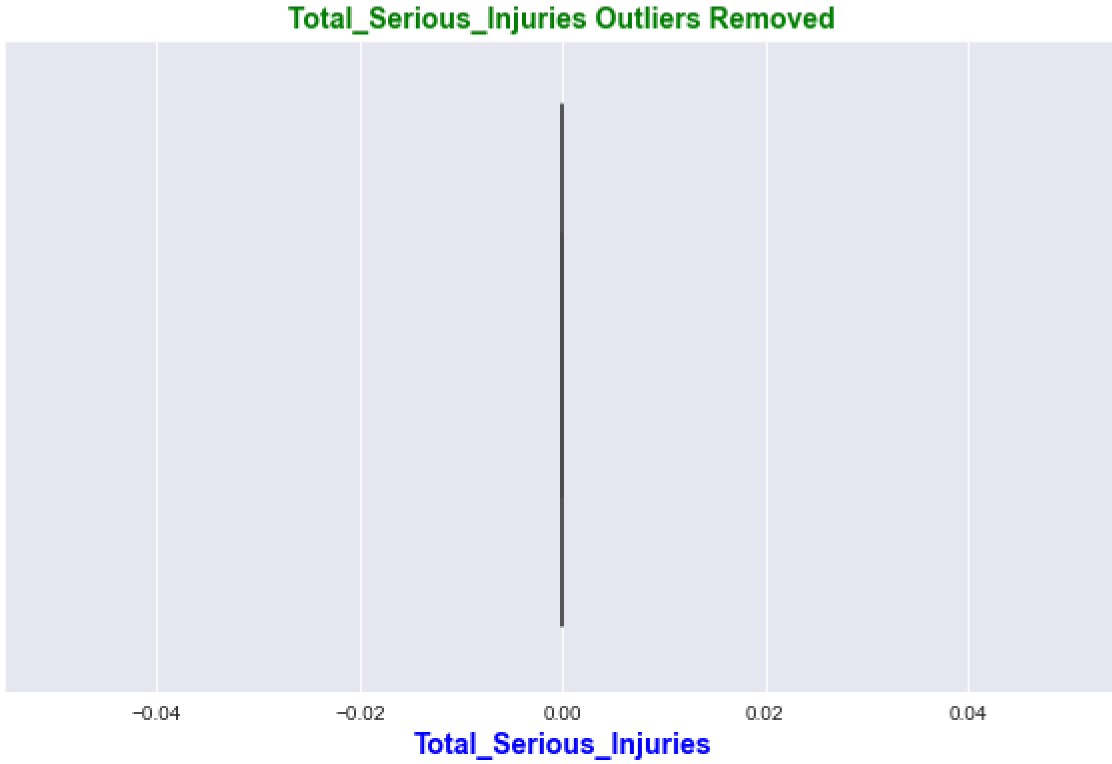

Total_Minor_Injuries Outliers
0
50
100
150
200
250
300
350
Total_Minor_Injuries

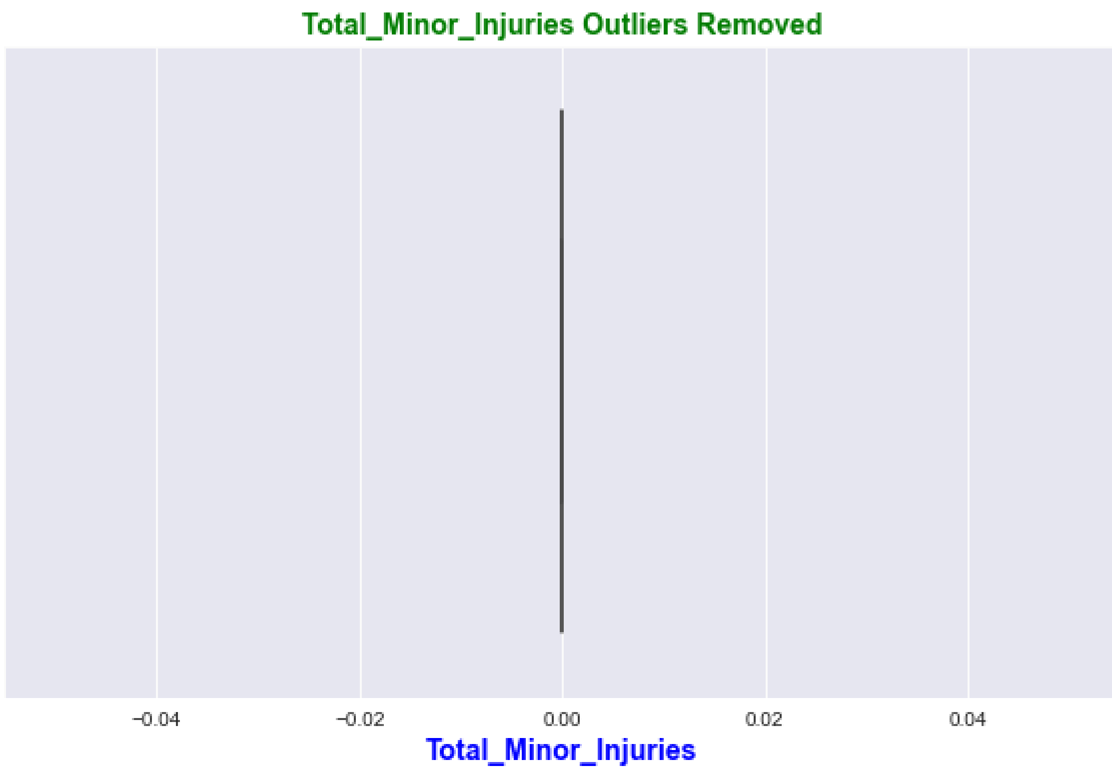


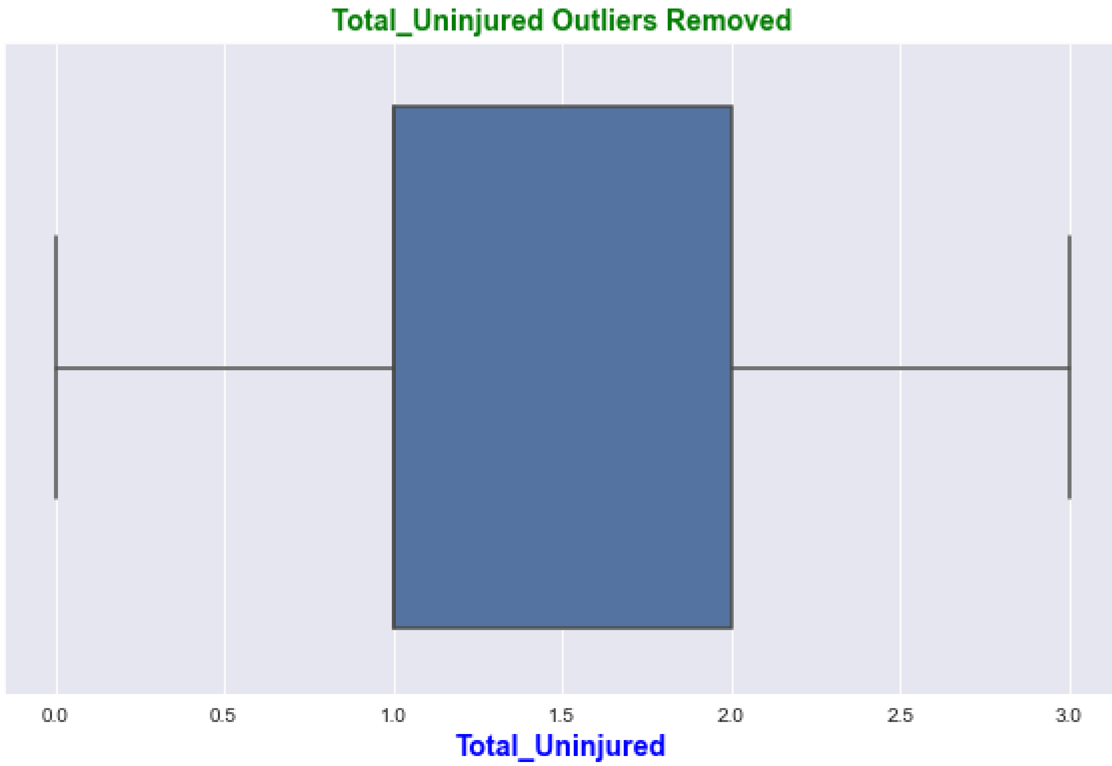
Total_Uninjured Outliers
0
100
200
300
400
500
600
Total_Uninjured

In [126]:
```
#Using interquartile range to remove the outliers
# Loop over each feature in df1
for feature in float_features:
# Calculate the interquartile range (IQR)
    Q1 = df1[feature].quantile(0.25)
    Q3 = df1[feature].quantile(0.75)
    IQR = Q3 - Q1

# Define the lower and upper bounds for outliers
    lower_limit = Q1 - 1.5 * IQR
    upper_limit = Q3 + 1.5 * IQR

# Filter the data to remove outliers
    df1 = df1[(df1[feature] >= lower_limit) & (df1[feature] <= upper_limit)]

# Check the boxplot again
    plt.figure(figsize= (10,6))
    sns.boxplot(x=df1[feature])

    plt.title(f'{feature} Outliers Removed'.title(), size=14, color='green', 
    plt.xlabel(feature.title(), size=14, color='blue', weight='bold')
    plt.show()
```




Total_Serious_Injuries Outliers Removed
−0.04
−0.02
0.00
0.02
0.04
Total_Serious_Injuries


Total_Minor_Injuries Outliers Removed
−0.04
−0.02
0.00
0.02
0.04
Total_Minor_Injuries



In [127]:
```
df1.describe().T
```

Out[127]:

	count	mean	std	min	25%	50%	75%	max
total_fatal_injuries	15207.0	0.000000	0.000000	0.0	0.0	0.0	0.0	0.0
total_serious_injuries	15207.0	0.000000	0.000000	0.0	0.0	0.0	0.0	0.0
total_minor_injuries	15207.0	0.000000	0.000000	0.0	0.0	0.0	0.0	0.0
total_uninjured	15207.0	1.522786	0.643355	0.0	1.0	1.0	2.0	3.0

In [128]:
```
df1.shape
```

Out[128]: (15207, 14)

4.1.4 Data consistency

Consistency is achieved through removal of duplicates in the dataframe

In [129]:
```
#Checking for duplicates
df1.duplicated().sum()
```

Out[129]: 4

In [130]:
```
#Removing duplicates
clean_df1=df1.drop_duplicates()
```

In [131]:
```
#preview clean_df1
clean_df1
```

Out[131]:

	event_date	investigation_type	location	injury_severity	aircraft_damage	aircraft_categ
7	1982-01-01	Accident	PULLMAN, WA	Non-Fatal	Substantial	Airp
8	1982-01-01	Accident	EAST HANOVER, NJ	Non-Fatal	Substantial	Airp
16	1982-01-02	Accident	MIDWAY, UT	Non-Fatal	Destroyed	Helicc
18	1982-01-02	Accident	GALETON, PA	Non-Fatal	Substantial	Airp
19	1982-01-02	Accident	MIAMI, FL	Non-Fatal	Substantial	Helicc
...	...	...	...	...	...	
88865	2022-12-12	Accident	Knoxville, TN	Non-Fatal	Substantial	Airp
88869	2022-12-13	Accident	Lewistown, MT	Non-Fatal	Substantial	Airp
88873	2022-12-14	Accident	San Juan, PR	Non-Fatal	Substantial	Airp
88876	2022-12-15	Accident	Wichita, KS	Non-Fatal	Substantial	Airp
88886	2022-12-26	Accident	Payson, AZ	Non-Fatal	Substantial	Airp

15203 rows × 14 columns

4.1.5 Data Uniformity

Involves feature engineering

In [132]:
```
#Required is to filter data within the set time frame on 'Event.Date' attribu
clean_df1['event_date'] = pd.to_datetime(clean_df1['event_date'])


#filtering dataframe within date(1962-2023 range
start_date = '1962-01-01'
end_date = '2023-01-01'

clean_df1=clean_df1.loc[(clean_df1['event_date'] >= start_date) & (clean_df1[
```

```
<ipython-input-132-3f9761410ae4>:2: SettingWithCopyWarning:
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/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pand
as.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-v
ersus-a-copy)
  clean_df1['event_date'] = pd.to_datetime(clean_df1['event_date'])
```

In [133]:
```
clean_df1.head()
```

Out[133]:

	event_date	investigation_type	location	injury_severity	aircraft_damage	aircraft_category
7	1982-01-01	Accident	PULLMAN, WA	Non-Fatal	Substantial	Airplane
8	1982-01-01	Accident	EAST HANOVER, NJ	Non-Fatal	Substantial	Airplane
16	1982-01-02	Accident	MIDWAY, UT	Non-Fatal	Destroyed	Helicopter
18	1982-01-02	Accident	GALETON, PA	Non-Fatal	Substantial	Airplane
19	1982-01-02	Accident	MIAMI, FL	Non-Fatal	Substantial	Helicopter

In [134]:
```
clean_df1['make'] = clean_df1['make'].str.title()
```

In [135]:
```
clean_df1['make'].value_counts()
```

Out[135]:
```
Cessna              4771
Piper               2546
Beech                738
Bell                 349
Robinson             218
                    ... 
Glenn L Smith          1
Golden Circle Air      1
Leonardo Spa           1
Cotton Galen M         1
Dassault/Sud           1
Name: make, Length: 2080, dtype: int64
```

In [136]:
```
#Renaming columns
clean_df1.rename(columns= lambda x: x.replace('.', '_').title(), inplace=True
```

In [137]:
```
clean_df1.head()
```

Out[137]:

	Event_Date	Investigation_Type	Location	Injury_Severity	Aircraft_Damage	Aircraft_Categ
7	1982-01-01	Accident	PULLMAN, WA	Non-Fatal	Substantial	Airpla
8	1982-01-01	Accident	EAST HANOVER, NJ	Non-Fatal	Substantial	Airpla
16	1982-01-02	Accident	MIDWAY, UT	Non-Fatal	Destroyed	Helicop
18	1982-01-02	Accident	GALETON, PA	Non-Fatal	Substantial	Airpla
19	1982-01-02	Accident	MIAMI, FL	Non-Fatal	Substantial	Helicop

In [138]:
```
#Removing trailing parantheses in Injury_Severity feature
clean_df1['Injury_Severity'] = clean_df1['Injury_Severity'].str.replace(r"\((
```

In [139]:
```
clean_df1['Injury_Severity'].value_counts()
```

Out[139]:
```
Non-Fatal      15061
Incident         101
Unavailable       22
Fatal             17
Minor              1
Serious            1
Name: Injury_Severity, dtype: int64
```

In [140]:
```
#Feature engineering
#extracting  year and month
clean_df1['Year'] = clean_df1['Event_Date'].dt.year
clean_df1['Month'] = clean_df1['Event_Date'].dt.month
```

In [141]:
```
clean_df1.head()
```

Out[141]:

	Event_Date	Investigation_Type	Location	Injury_Severity	Aircraft_Damage	Aircraft_Catego
7	1982-01-01	Accident	PULLMAN, WA	Non-Fatal	Substantial	Airpla
8	1982-01-01	Accident	EAST HANOVER, NJ	Non-Fatal	Substantial	Airpla
16	1982-01-02	Accident	MIDWAY, UT	Non-Fatal	Destroyed	Helicop
18	1982-01-02	Accident	GALETON, PA	Non-Fatal	Substantial	Airpla
19	1982-01-02	Accident	MIAMI, FL	Non-Fatal	Substantial	Helicop

4.1.6 Saving Cleaned Data

In [142]:
```
#save the new dataframe in svs format
clean_df1.to_csv('clean_aviation_data.csv', index=False)
```

5.0 Exploratory Data Analysis(EDA)

This is the process of analyzing data to reveal trends and patterns, detect anomalies, test hypotheses and check assumptions using visuals and summary statistics.Turkey,J.W(1977)

Key goals of EDA include:

Understanding the data: Getting a sense of the data's distribution, range, and central tendencies. Identifying patterns: Discovering trends, correlations, or anomalies within the data. Checking assumptions: Verifying assumptions made about the data before further analysis or modeling. Generating hypotheses: Developing potential explanations or questions based on the findings.

In [143]:
```
#load the clean dataset for analysis
data = pd.read_csv('clean_aviation_data.csv')
data.head()
```

Out[143]:

	Event_Date	Investigation_Type	Location	Injury_Severity	Aircraft_Damage	Aircraft_Categoı
0	1982-01-01	Accident	PULLMAN, WA	Non-Fatal	Substantial	Airplar
1	1982-01-01	Accident	EAST HANOVER, NJ	Non-Fatal	Substantial	Airplar
2	1982-01-02	Accident	MIDWAY, UT	Non-Fatal	Destroyed	Helicopt
3	1982-01-02	Accident	GALETON, PA	Non-Fatal	Substantial	Airplar
4	1982-01-02	Accident	MIAMI, FL	Non-Fatal	Substantial	Helicopt

In [144]:
```
data.isna().sum().sum()
```

Out[144]: 0

5.1 Univariate Analysis

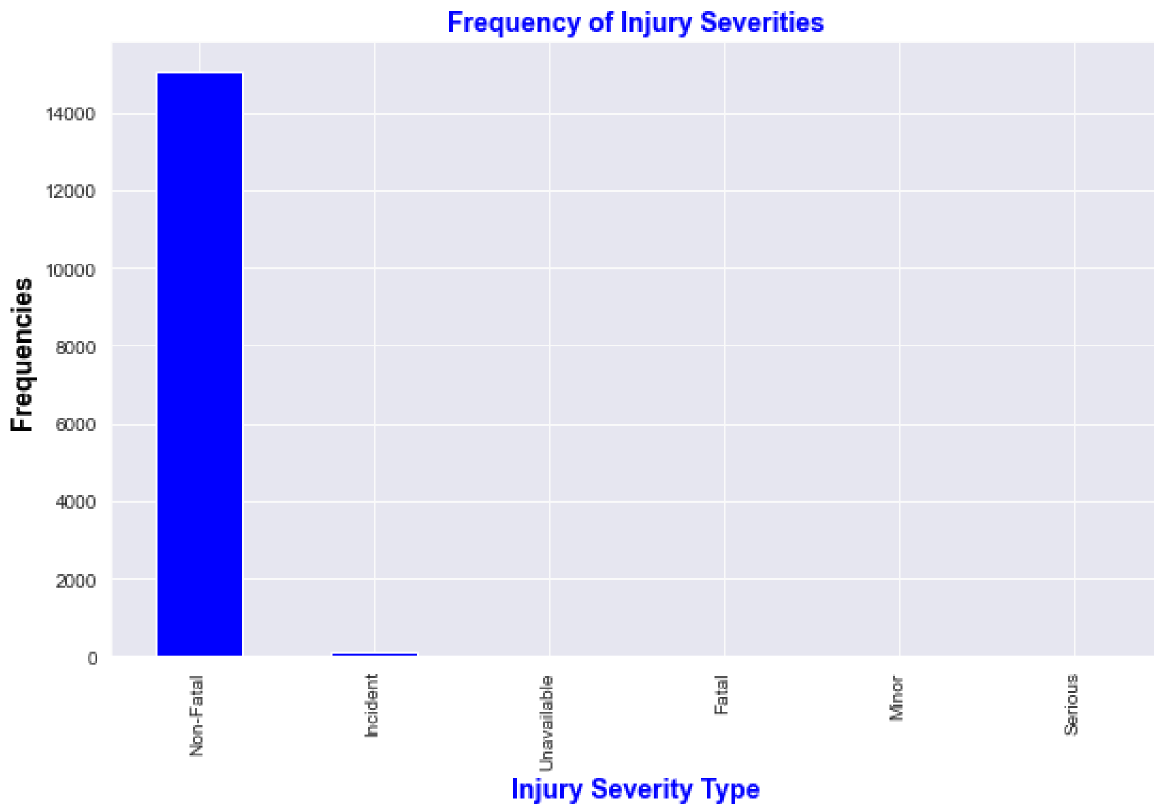
Univariate analysis examination of single variable distribution and measures of central tendency. Objective of this analysis is to identify patterns, trends, and outliers.

Count plots,bar charts, and pie charts are used to visually represent categorical data, while histogram and boxplots are used to visualize numerical data.

a. Count Plot

A count plot is a type of bar chart that shows the number of times each unique value occurs in a variable. It is often used to visualize the distribution of categorical variables.

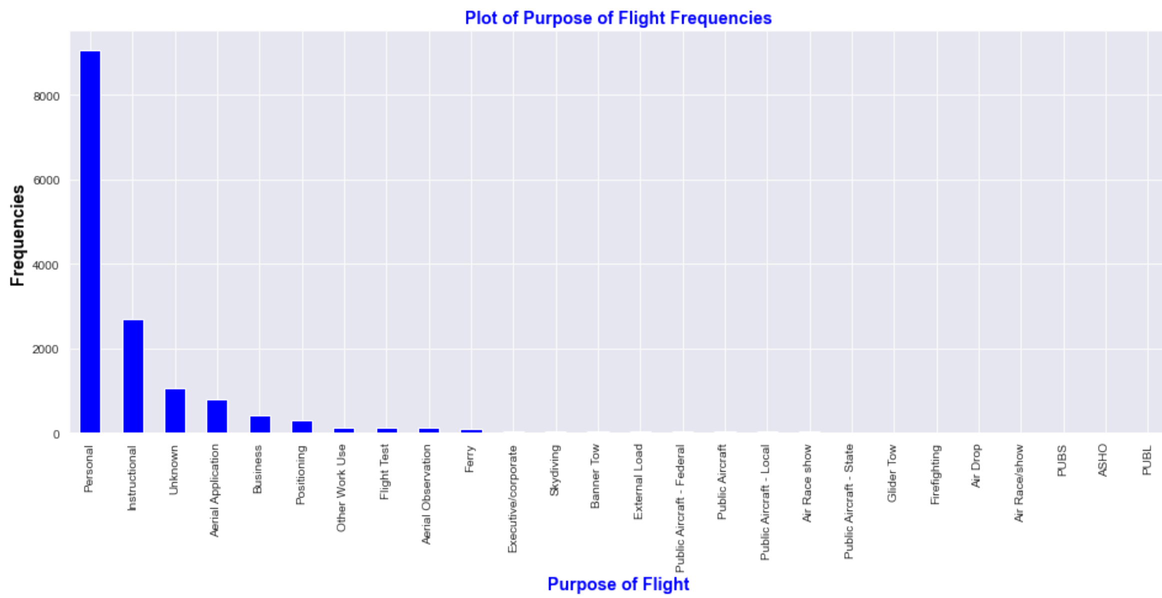
In [145]:
```
#  Exploring the  Injury severity with the highest records
Injury_Severity_count = data['Injury_Severity'].value_counts()
Injury_Severity_count
#visuals
plt.figure(figsize= (10,6))
Injury_Severity_count.plot(kind='bar', color='blue')
xlabel='Injury_Severity'
plt.title('Frequency of Injury Severities', size=14,color= 'blue', weight='bo
plt.ylabel('Frequencies', size=14,color= 'black', weight='bold')
plt.xlabel('Injury Severity Type', size=14,color= 'blue', weight='bold');
```



Observation #1

Highest number of injury severity is Non Fatal type of injuries.

In [146]:
```
# Criteria 1: a) Check distribution of the purpose of flight.

purpose_of_flight_count = data['Purpose_Of_Flight'].value_counts()
purpose_of_flight_count

# Visual in a barplot
plt.figure(figsize= (16,6))
purpose_of_flight_count.plot(kind='bar', color='blue')
xlabel='Purpose_Of_Flight'
plt.title('Plot of Purpose of Flight Frequencies', size=14,color= 'blue', wei
plt.ylabel('Frequencies', size=14,color= 'black', weight='bold')
plt.xlabel('Purpose of Flight', size=14,color= 'blue', weight='bold');
```



Observation #2

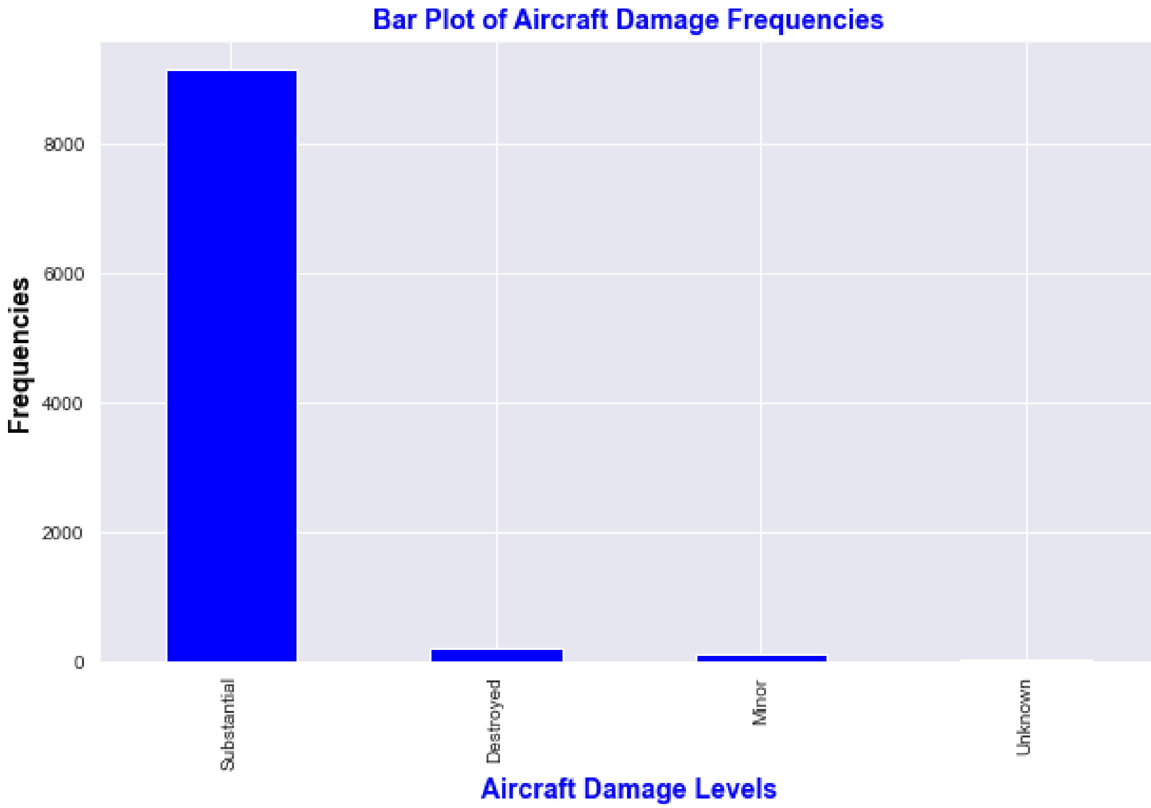
Flights for personal or private purpose has the highest frequency. Making reference to the business case above that requires choice of aircrafts for business and private enterprise; I will deduct data in accordance with the two conditions (business or private(personal))

In [147]:
```
# Criteria 1: b) Selecting data based on business or private(personal) enterpr
df_purpose = data[data['Purpose_Of_Flight'].str.contains('Personal', case=Fal
df_purpose.head()
```

Out[147]:

	Event_Date	Investigation_Type	Location	Injury_Severity	Aircraft_Damage	Aircraft_Categor
0	1982-01-01	Accident	PULLMAN, WA	Non-Fatal	Substantial	Airplar
1	1982-01-01	Accident	EAST HANOVER, NJ	Non-Fatal	Substantial	Airplar
2	1982-01-02	Accident	MIDWAY, UT	Non-Fatal	Destroyed	Helicopt
3	1982-01-02	Accident	GALETON, PA	Non-Fatal	Substantial	Airplar
4	1982-01-02	Accident	MIAMI, FL	Non-Fatal	Substantial	Helicopt

In [148]:
```
#Criteria 2: a) Asset Risk; Check degree of damage to aircraft and select the
#Checking frequency distributions for each damage degree
plt.figure(figsize= (10,6))
df_purpose['Aircraft_Damage'].value_counts().plot(kind='bar', color='blue')
xlabel='Aircraft_Damage'
plt.title('Bar Plot of Aircraft Damage Frequencies', size=14,color= 'blue', w
plt.ylabel('Frequencies', size=14,color= 'black', weight='bold')
plt.xlabel('Aircraft Damage Levels', size=14,color= 'blue', weight='bold');
```



Observation #3

The aircrafts the are substantially damaged has the highest frequency. However, criteria of selection prefers aircrafts that in the event of an accident it sustains minor damages. Therefore, the select criteria follows 'Aircraft_Damage' feature with 'Minor' data values

In [149]:
```
#Criteria 2: b) Asset Risk; Check degree of damage to aircraft and select the
#Selecting data with Minor from df_purpose dataframe
df_minor = df_purpose.query('Aircraft_Damage =="Minor"')
df_minor.head()
```

Out[149]:

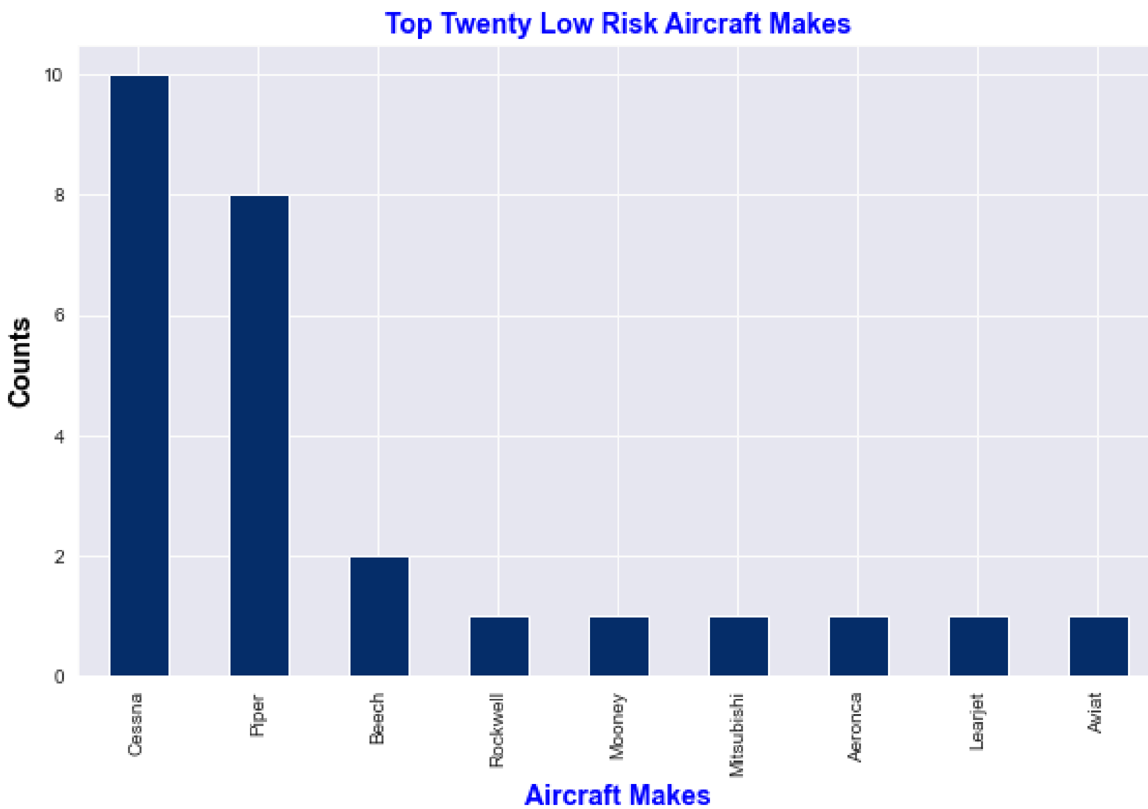
	Event_Date	Investigation_Type	Location	Injury_Severity	Aircraft_Damage	Aircraft_Ca
8	1982-01-03	Incident	VAN NUYS, CA	Incident	Minor	A
11	1982-01-05	Incident	PENSACOLA, FL	Incident	Minor	A
86	1982-01-30	Incident	TRUCKEE, CA	Incident	Minor	A
114	1982-02-06	Accident	SAN JOSE, CA	Non-Fatal	Minor	A
313	1982-03-20	Incident	MOBILE, AL	Incident	Minor	A

In [150]:
```
# Criteria 3: User Related Risk; Severity of injuries inflicted to users
# Check and select data with lower risk to user(Minor or Incident) from df_mi

df_low_risk = df_minor.query('Injury_Severity == ["Incident", "Minor"]')
df_low_risk.head()
```

Out[150]:

	Event_Date	Investigation_Type	Location	Injury_Severity	Aircraft_Damage	Aircraft_(
8	1982-01-03	Incident	VAN NUYS, CA	Incident	Minor	
11	1982-01-05	Incident	PENSACOLA, FL	Incident	Minor	
86	1982-01-30	Incident	TRUCKEE, CA	Incident	Minor	
313	1982-03-20	Incident	MOBILE, AL	Incident	Minor	
465	1982-04-20	Incident	COTTONWOOD FALL, KS	Incident	Minor	

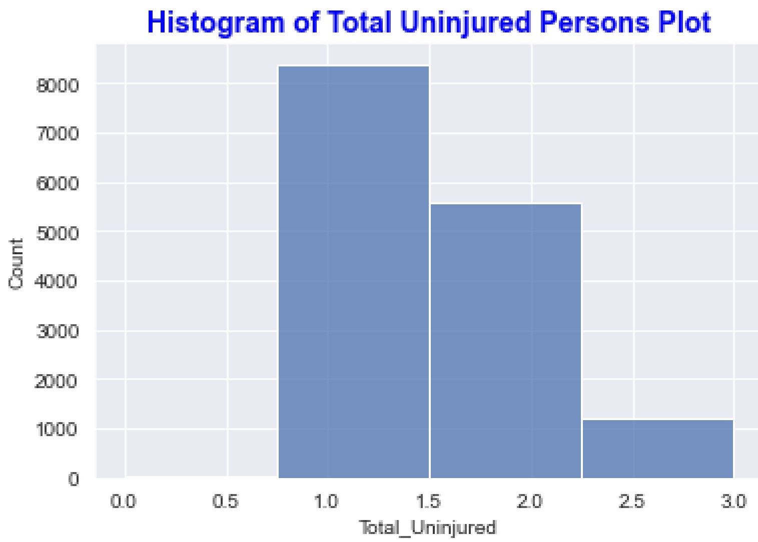
In [151]:
```
# Extract and visualize the top ten aircraft makes with low risk
#top twenty makes dataframe
top_twenty_makes = df_low_risk['Make'].value_counts().head(20)

# Visualization
plt.figure(figsize=(10, 6))
top_twenty_makes.plot(kind='bar', colormap='Blues_r')
#Title
plt.title('Top Twenty Low Risk Aircraft Makes',size=14,color= 'blue', weight=
# Name axes
plt.xlabel('Aircraft Makes',size=14,color= 'blue', weight='bold')
plt.ylabel('Counts',size=14,color= 'black', weight='bold');
```



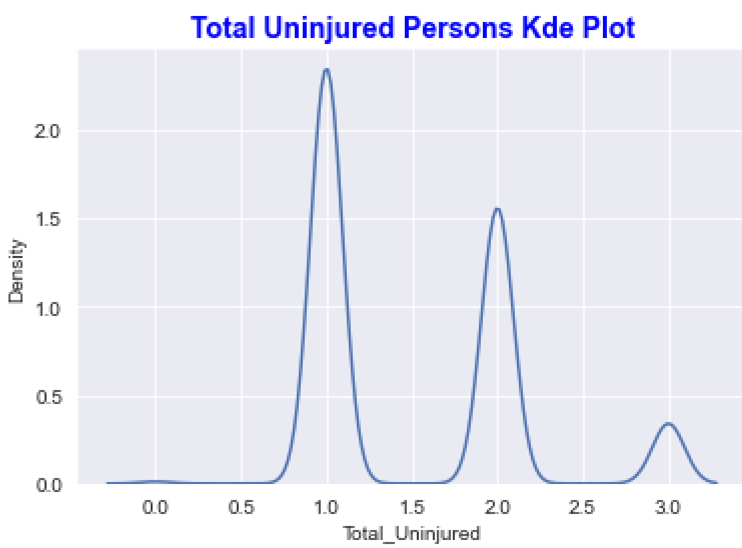
Observation #4

This visual shows top twenty low risk aircraft makes. It is observed that Cessna make has the highest frequency. Based on this criteria, it is safe to adduce that Cessna aircraft make in general is a low risk aircraft.

In [152]:
```
#hist plot for Total Uninjured  feature.
sns.histplot(x=data['Total_Uninjured'], bins=4)
plt.title('Histogram of Total Uninjured Persons Plot', fontsize=14, color='bl
plt.show()
```



In [153]:
```
sns.kdeplot(x=data['Total_Uninjured'])
plt.title('Total Uninjured Persons Kde Plot', fontsize=14, color='blue', weig
plt.show()
```



In [154]:
```
df_low_risk.describe().T
```

Out[154]:

	count	mean	std	min	25%	50%	75%	max
Total_Fatal_Injuries	26.0	0.000000	0.000000	0.0	0.00	0.0	0.00	0.0
Total_Serious_Injuries	26.0	0.000000	0.000000	0.0	0.00	0.0	0.00	0.0
Total_Minor_Injuries	26.0	0.000000	0.000000	0.0	0.00	0.0	0.00	0.0
Total_Uninjured	26.0	1.692308	0.735893	1.0	1.00	2.0	2.00	3.0
Year	26.0	1988.346154	10.691837	1982.0	1982.00	1982.0	1997.75	2007.0
Month	26.0	6.692308	3.259070	1.0	4.25	7.0	9.00	12.0

5.2 Bivariate Analysis

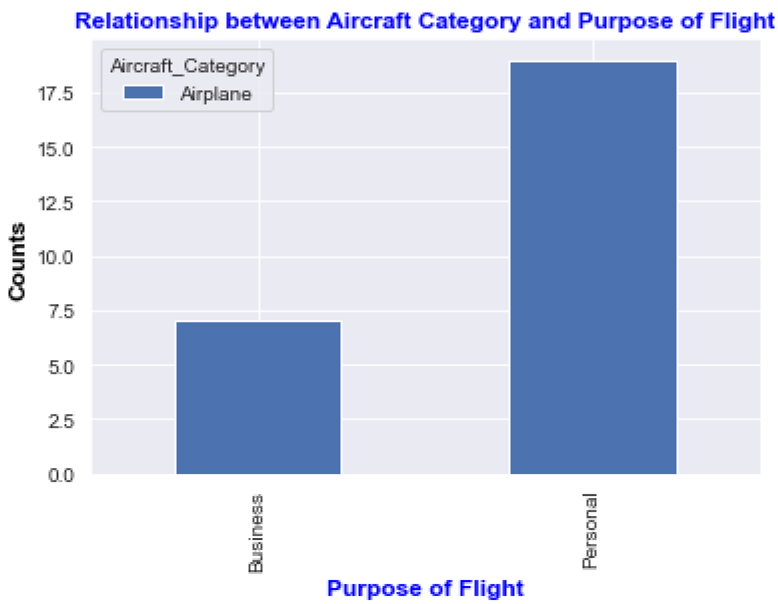
This is the analysis of data to identify patterns, trends, and correlations of two variables in a given dataset. This can be achieved by use of bar plots, scatter plots, correlation coefficient and regression analysis

In [155]:
```
df_low_risk.head()
```

Out[155]:

	Event_Date	Investigation_Type	Location	Injury_Severity	Aircraft_Damage	Aircraft_(
8	1982-01-03	Incident	VAN NUYS, CA	Incident	Minor	
11	1982-01-05	Incident	PENSACOLA, FL	Incident	Minor	
86	1982-01-30	Incident	TRUCKEE, CA	Incident	Minor	
313	1982-03-20	Incident	MOBILE, AL	Incident	Minor	
465	1982-04-20	Incident	COTTONWOOD FALL, KS	Incident	Minor	

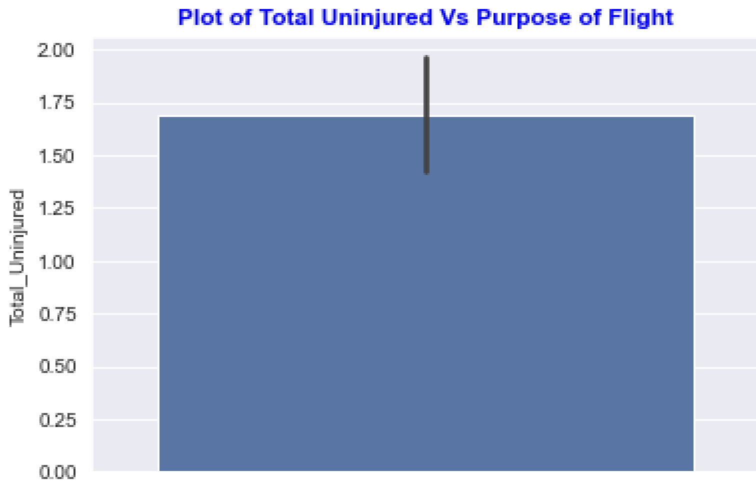
In [156]:
```
#Criteria 4; Combination of conditions
#Create a crosstab
Aircraft_purpose = pd.crosstab(df_low_risk['Purpose_Of_Flight'], df_low_risk[
Aircraft_purpose

#visualize barchart
Aircraft_purpose.plot(kind='bar')
plt.title('Relationship between Aircraft Category and Purpose of Flight', siz
plt.xlabel('Purpose of Flight', size=12, color='blue', weight='bold')
plt.ylabel('Counts', size=12, color='black',weight='bold');
```



Observation #1

Aircrafts for personal purpose has the highest frequency. This implies that low risk aircrafts are majorly operated for private enterprises than for business enterprises.

In [157]:
```
sns.barplot(y=df_low_risk['Total_Uninjured'], hue = df_low_risk['Purpose_Of_F
plt.title('Plot of Total Uninjured Vs Purpose of Flight', size=12, color='blu
```



In [158]:
```
#selecting numerical variables only
df_nums = df_low_risk[['Total_Fatal_Injuries','Total_Serious_Injuries','Total_
```

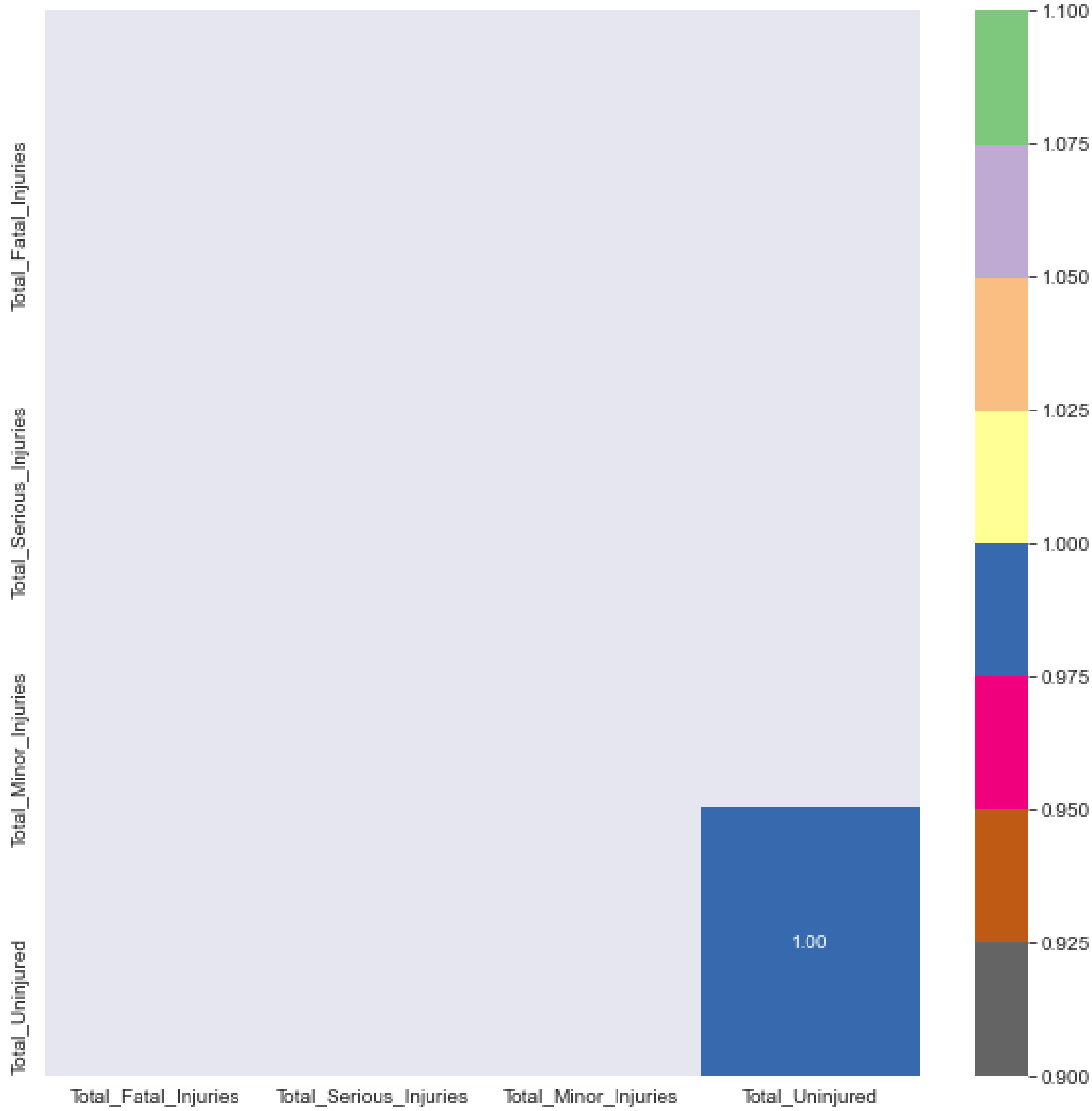
In [159]:
```
corr = df_nums.corr()
corr
```

Out[159]:

	Total_Fatal_Injuries	Total_Serious_Injuries	Total_Minor_Injuries	Total_Unii
Total_Fatal_Injuries	NaN	NaN	NaN	
Total_Serious_Injuries	NaN	NaN	NaN	
Total_Minor_Injuries	NaN	NaN	NaN	
Total_Uninjured	NaN	NaN	NaN	

In [160]:
```
plt.figure(figsize=(10,10))
sns.heatmap(corr, annot=True, fmt='.2f',cmap='Accent_r')
```

Out[160]: <AxesSubplot:>



Observation #2

There are no correlations between numerical variables in low risk dataframe (df_low_risk).

5.3 Multivariate Analayis

Multivariate analysis is a statistical technique used to describe and summarize patterns, trends, and correlations between three or more variables. It is achieved by deployement of various analysis techniques such as ;

** Multiple regression analysis

** Factor analysis

** Cluster analysis

** Discriminant analysis

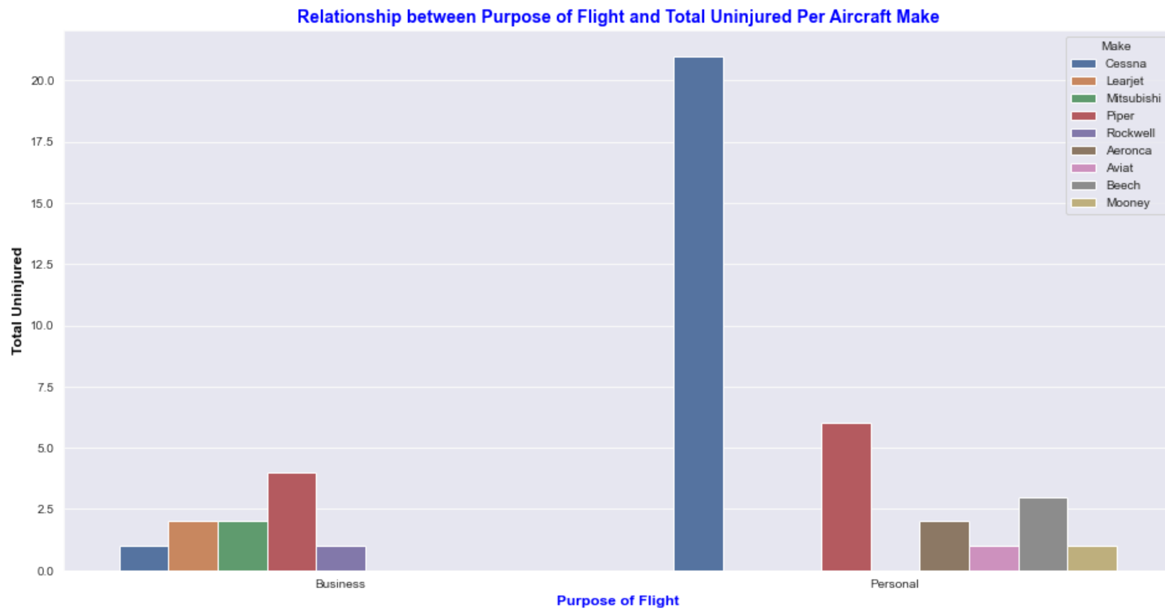
In [161]:
```
#Criteria 4; Combination of conditions
#Select the top ten makes as per their frequencies
top_makes = df_low_risk['Make'].value_counts().nlargest(10).index
selected_df = df_low_risk[df_low_risk['Make'].isin(top_makes)]
selected_df.head()
```

Out[161]:

	Event_Date	Investigation_Type	Location	Injury_Severity	Aircraft_Damage	Aircraft_(
8	1982-01-03	Incident	VAN NUYS, CA	Incident	Minor	
11	1982-01-05	Incident	PENSACOLA, FL	Incident	Minor	
86	1982-01-30	Incident	TRUCKEE, CA	Incident	Minor	
313	1982-03-20	Incident	MOBILE, AL	Incident	Minor	
465	1982-04-20	Incident	COTTONWOOD FALL, KS	Incident	Minor	

In [162]:
```
# Sort data using .groupby.
# This enables comparison of more than two variables
group_df = selected_df.groupby(["Purpose_Of_Flight", "Make", ])['Total_Uninju
```

In [163]:
```
plt.figure(figsize=(16,8))
sns.set_style('darkgrid')
sns.set_palette('deep')

sns.barplot(x='Purpose_Of_Flight', y= 'Total_Uninjured', hue='Make', data=gro
plt.title('Relationship between Purpose of Flight and Total Uninjured Per Airc
plt.xlabel('Purpose of Flight', size=12,color='blue', weight='bold')
plt.ylabel('Total Uninjured', size=12, color='black', weight='bold');
```



Observation #1

The interpretation of the above visual is that; those aircrafts with low risk are majorly operated for personal(private) enterprises. Cessna aircrafts has the highest frequency for personal purpose whereas Piper aircrafts have the highest frequency in business purpose.

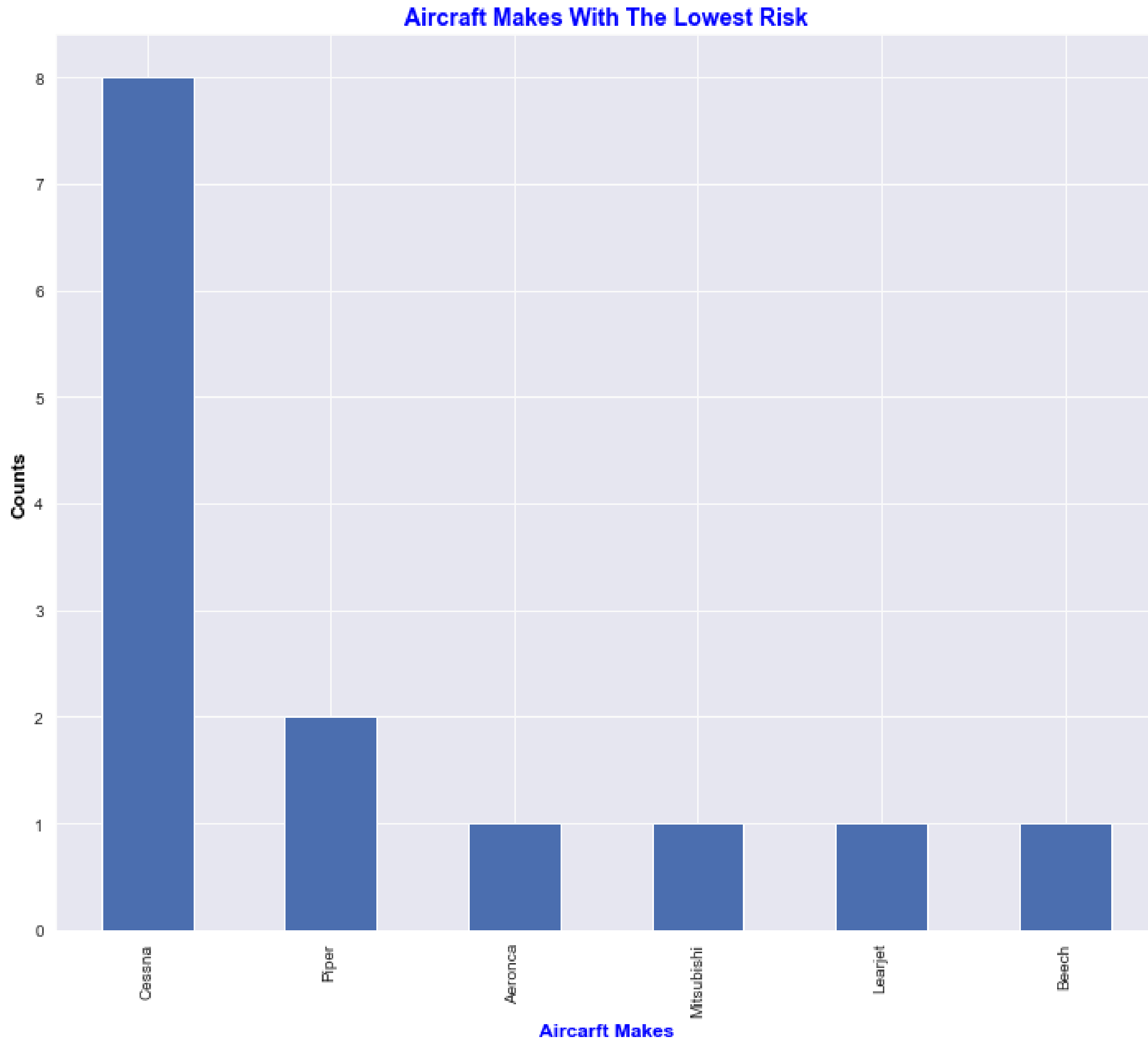
In [164]:
```
# Sorting data in df_low_risk dataframe using crosstab()
# Considering multiple features 'Injury_Severity','Aircraft_Damage','Purpose_(

combined_df = pd.crosstab(index= [df_low_risk['Injury_Severity'], df_low_risk
                          df_low_risk['Purpose_Of_Flight']], columns= 'count'
                          aggfunc= 'max')
```

In [165]:
```
combined_df.head()
```

Out[165]:

		col_0	count
Injury_Severity	Aircraft_Damage	Purpose_Of_Flight	
Incident	Minor	Business	2.0
		Personal	3.0

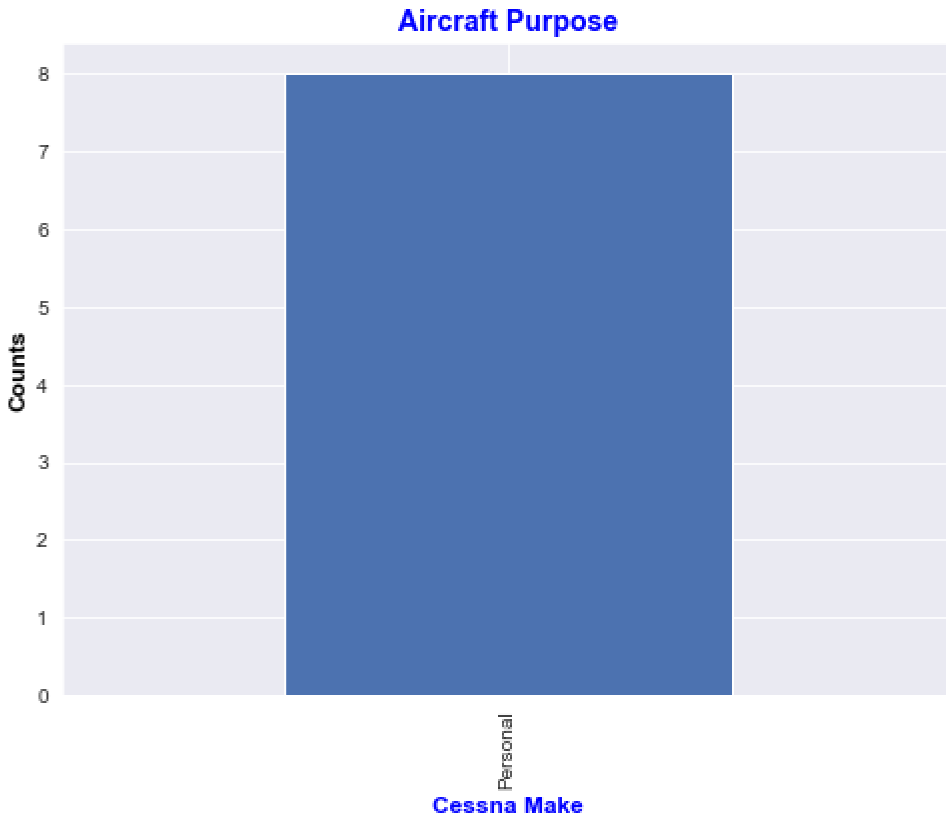
In [166]:
```
#Extracting aircraft makes that meet the low risk criteria
low_risk_aircraft = df_low_risk[df_low_risk['Total_Uninjured'].isin(combined_
low_risk_makes = low_risk_aircraft['Make'].value_counts()

#Visualization of the makes
plt.figure(figsize=(12,10))
low_risk_makes.plot(kind='bar')
plt.title('Aircraft Makes With The Lowest Risk', size=14, color='blue', weigh
plt.xlabel('Aircarft Makes', size=12, color='blue', weight='bold')
plt.ylabel('Counts', size=12, color='black', weight='bold');
```



Observation #2

Cessna make appears to be of a lowest risk aircraft. This can be extracted to show which purpose (business or personal) has the lowest risk.

In [167]:
```
# Visualizing the purpose of aircraft make with lowest risk
#From observation above, Cessna make has highest counts of instances with high
cessna_make = low_risk_aircraft[low_risk_aircraft['Make']=='Cessna']
plt.figure(figsize=(8,6))
cessna_make['Purpose_Of_Flight'].value_counts().plot(kind='bar')
plt.title('Aircraft Purpose', size=14, color='blue',weight='bold')
plt.xlabel( 'Cessna Make', size=12, color='blue', weight='bold')
plt.ylabel('Counts', size=12, color='black', weight='bold');
```



Observation #3

It can be deduced that among the low risk aircraft makes Cessna,Pipper and Mitsubishi show in the event of and accident or an incident;

** The aircraft will, to a large extent, sustain 'minor' damages, and

** The users have higher probability of remaining uninjured.

Subsequently, it is observed from the number of persons uninjured that there are more in aircrafts operated for personal/private enterprise than in those operated for business enterprise.

Finally, data shows Cessna make proves to be safer to operate for business and personal(private) enterprises especially for personal enterprises.

Summary

This analysis relied on AviationData.csv dataset provided and following the criteria set to achieve the research objective.

Critical variables include; Injury_Severity, Aircraft_Damage, Make, Purpose_Of_Flight and Total_Uninjured to select the aircraft with the lowest risk.

Cessna Make of aicrafts showed to posses lowest risks.

Cessna aircrafts operated for private(personal) enterprises appeared to have lower risks than those operated for business enterprises.