Phase One Project

1. PROJECT OVERVIEW

"""

The objective of this project is to investigate the dataset for peculiarities,perform data cleaning,visualize and summarize data to uncover patterns, trends and relationships and also to ensure that project workflow is well documented and reproducable for others to follow.This notebook contain dataset from Aviation Accident Database & Synopses, up to 2023 Dataset. This dataset from Kaggleaviation accident database contains Aviation data and USState code that provides information from 1962 and later about civil aviation accidents and selected incidents within the United States, its territories and possessions, and in international waters.The primary goal is to establish a structured approach to organizing and analyzing the data effectively.

"""

2. DATA LOADING

```python

# Import pandas as pd

# import numpy as np

```python
# loading data
# This method imports pandas library tool inorder to read the attached csv file
import pandas as pd
data= pd.read_csv('./AviationData.csv', encoding='latin1')
print(data)
```

```
             Event.Id Investigation.Type Accident.Number
Event.Date  \
0      20001218X45444           Accident      SEA87LA080  1948-10-24

1      20001218X45447           Accident      LAX94LA336  1962-07-19

2      20061025X01555           Accident      NYC07LA005  1974-08-30

3      20001218X45448           Accident      LAX96LA321  1977-06-19

4      20041105X01764           Accident      CHI79FA064  1979-08-02

...               ...                ...             ...         ...
```
```

```
88884  20221227106491           Accident      ERA23LA093  2022-12-26

88885  20221227106494           Accident      ERA23LA095  2022-12-26

88886  20221227106497           Accident      WPR23LA075  2022-12-26

88887  20221227106498           Accident      WPR23LA076  2022-12-26

88888  20221230106513           Accident      ERA23LA097  2022-12-29


               Location        Country   Latitude  Longitude
Airport.Code  \
0      MOOSE CREEK, ID  United States        NaN        NaN
NaN
1       BRIDGEPORT, CA  United States        NaN        NaN
NaN
2        Saltville, VA  United States  36.922223 -81.878056
NaN
3           EUREKA, CA  United States        NaN        NaN
NaN
4           Canton, OH  United States        NaN        NaN
NaN
...                ...            ...        ...        ...           .
..
88884    Annapolis, MD  United States        NaN        NaN
NaN
88885      Hampton, NH  United States        NaN        NaN
NaN
88886       Payson, AZ  United States    341525N   1112021W
PAN
88887       Morgan, UT  United States        NaN        NaN
NaN
88888       Athens, GA  United States        NaN        NaN
NaN

      Airport.Name  ... Purpose.of.flight         Air.carrier  \
0              NaN  ...          Personal                 NaN
1              NaN  ...          Personal                 NaN
2              NaN  ...          Personal                 NaN
3              NaN  ...          Personal                 NaN
4              NaN  ...          Personal                 NaN
...            ...  ...               ...                 ...
88884          NaN  ...          Personal                 NaN
88885          NaN  ...               NaN                 NaN
88886       PAYSON  ...          Personal                 NaN
88887          NaN  ...          Personal  MC CESSNA 210N LLC
88888          NaN  ...          Personal                 NaN
```

```
      Total.Fatal.Injuries Total.Serious.Injuries Total.Minor.Injuries
\
0                      2.0                     0.0                   0.0

1                      4.0                     0.0                   0.0

2                      3.0                     NaN                   NaN

3                      2.0                     0.0                   0.0

4                      1.0                     2.0                   NaN

...                    ...                     ...                   ...

88884                  0.0                     1.0                   0.0

88885                  0.0                     0.0                   0.0

88886                  0.0                     0.0                   0.0

88887                  0.0                     0.0                   0.0

88888                  0.0                     1.0                   0.0


      Total.Uninjured Weather.Condition  Broad.phase.of.flight  \
0                 0.0               UNK                 Cruise
1                 0.0               UNK               Unknown
2                 NaN               IMC                 Cruise
3                 0.0               IMC                 Cruise
4                 0.0               VMC              Approach
...               ...               ...                   ...
88884             0.0               NaN                   NaN
88885             0.0               NaN                   NaN
88886             1.0               VMC                   NaN
88887             0.0               NaN                   NaN
88888             1.0               NaN                   NaN

        Report.Status Publication.Date
0      Probable Cause              NaN
1      Probable Cause       19-09-1996
2      Probable Cause       26-02-2007
3      Probable Cause       12-09-2000
4      Probable Cause       16-04-1980
...               ...              ...
88884             NaN       29-12-2022
88885             NaN              NaN
88886             NaN       27-12-2022
88887             NaN              NaN
88888             NaN       30-12-2022
```

```
[88889 rows x 31 columns]

C:\Users\oguda\AppData\Local\Temp\ipykernel_48560\2887764154.py:3:
DtypeWarning: Columns (6,7,28) have mixed types. Specify dtype option
on import or set low_memory=False.
  data= pd.read_csv('./AviationData.csv', encoding='latin1')

# checking the first 5 rows of the data
# it is a function that displays only first 5 rows of the dataset
data.head()

         Event.Id Investigation.Type Accident.Number  Event.Date  \
0  20001218X45444           Accident      SEA87LA080  1948-10-24
1  20001218X45447           Accident      LAX94LA336  1962-07-19
2  20061025X01555           Accident      NYC07LA005  1974-08-30
3  20001218X45448           Accident      LAX96LA321  1977-06-19
4  20041105X01764           Accident      CHI79FA064  1979-08-02

          Location        Country   Latitude  Longitude
Airport.Code  \
0  MOOSE CREEK, ID  United States        NaN        NaN          NaN

1   BRIDGEPORT, CA  United States        NaN        NaN          NaN

2    Saltville, VA  United States  36.922223 -81.878056          NaN

3       EUREKA, CA  United States        NaN        NaN          NaN

4       Canton, OH  United States        NaN        NaN          NaN


  Airport.Name  ... Purpose.of.flight Air.carrier Total.Fatal.Injuries
\
0          NaN  ...          Personal         NaN                  2.0

1          NaN  ...          Personal         NaN                  4.0

2          NaN  ...          Personal         NaN                  3.0

3          NaN  ...          Personal         NaN                  2.0

4          NaN  ...          Personal         NaN                  1.0


  Total.Serious.Injuries Total.Minor.Injuries Total.Uninjured  \
0                    0.0                  0.0             0.0
1                    0.0                  0.0             0.0
2                    NaN                  NaN             NaN
3                    0.0                  0.0             0.0
4                    2.0                  NaN             0.0
```

```
  Weather.Condition  Broad.phase.of.flight   Report.Status
Publication.Date
0               UNK                 Cruise  Probable Cause
NaN
1               UNK                Unknown  Probable Cause       19-
09-1996
2               IMC                 Cruise  Probable Cause       26-
02-2007
3               IMC                 Cruise  Probable Cause       12-
09-2000
4               VMC               Approach  Probable Cause       16-
04-1980

[5 rows x 31 columns]
```

3. DATA EXPLORATION

. Summary statistics:

```
#checking the shape of the data
# It is a function that checks the data structure rows and column
respectively
data.shape
```

```
(88889, 31)
```

```
# checking the information of the data
# This function highlights the content of data e.g column,rows,index,
datatypes
data.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            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        75118 non-null  object
dtypes: float64(5), object(26)
memory usage: 21.0+ MB
```

```
# check summary statistics of the data
data.describe()
```

```
       Number.of.Engines  Total.Fatal.Injuries  Total.Serious.Injuries
\
count       82805.000000          77488.000000            76379.000000

mean            1.146585              0.647855                0.279881

std             0.446510              5.485960                1.544084

min             0.000000              0.000000                0.000000

25%             1.000000              0.000000                0.000000

50%             1.000000              0.000000                0.000000

75%             1.000000              0.000000                0.000000

max             8.000000            349.000000              161.000000


       Total.Minor.Injuries  Total.Uninjured
count          76956.000000     82977.000000
mean               0.357061         5.325440
std                2.235625        27.913634
min                0.000000         0.000000
25%                0.000000         0.000000
50%                0.000000         1.000000
75%                0.000000         2.000000
max              380.000000       699.000000
```

```
# checking duplicates
data.duplicated().value_counts()
```

```
False    88889
Name: count, dtype: int64
```

```
# checking the missing values in the data
data.isna().sum()
```

```
Event.Id                      0
Investigation.Type            0
Accident.Number               0
Event.Date                    0
Location                     52
Country                     226
Latitude                  54507
Longitude                 54516
Airport.Code              38757
Airport.Name              36185
Injury.Severity            1000
Aircraft.damage            3194
Aircraft.Category         56602
Registration.Number        1382
Make                         63
Model                        92
Amateur.Built               102
Number.of.Engines          6084
Engine.Type                7096
FAR.Description           56866
Schedule                  76307
Purpose.of.flight          6192
Air.carrier               72241
Total.Fatal.Injuries      11401
Total.Serious.Injuries    12510
Total.Minor.Injuries      11933
Total.Uninjured            5912
Weather.Condition          4492
Broad.phase.of.flight     27165
Report.Status              6384
Publication.Date          13771
dtype: int64
```

> From this dataset, It is evident that there are a number of missing values therefore, need to be cleaned

4. DATA CLEANING

```
# Making a copy of the data
clean_copy = data.copy()

print(clean_copy.columns)
```

```
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.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')
```

```
# droping the columns with more than 50% missing values
clean_copy.dropna(subset=['Latitude','Longitude','Aircraft.Category','
FAR.Description','Schedule','Air.carrier'], inplace=True)
```

```
# filling the missing values in the data
clean_copy['Registration.Number'] =
clean_copy['Registration.Number'].fillna('Unknown')
clean_copy['Airport.Code'] =
clean_copy['Airport.Code'].fillna('Unknown')
clean_copy['Airport.Name'] =
clean_copy['Airport.Name'].fillna(clean_copy['Airport.Name'].value_cou
nts().idxmax())
clean_copy['Model'] =
clean_copy['Model'].fillna(clean_copy['Model'].value_counts().idxmax()
)
clean_copy['Number.of.Engines'] =
clean_copy['Number.of.Engines'].fillna(clean_copy['Number.of.Engines']
.median())
clean_copy['Engine.Type'] =
clean_copy['Engine.Type'].fillna(clean_copy['Engine.Type'].value_count
s().idxmax())
clean_copy['Purpose.of.flight'] =
clean_copy['Purpose.of.flight'].fillna(clean_copy['Purpose.of.flight']
.value_counts().idxmax())
clean_copy['Total.Fatal.Injuries'] =
clean_copy['Total.Fatal.Injuries'].fillna(clean_copy['Total.Fatal.Inju
ries'].median())
clean_copy['Total.Serious.Injuries'] =
clean_copy['Total.Serious.Injuries'].fillna(clean_copy['Total.Serious.
Injuries'].median())
clean_copy['Total.Minor.Injuries'] =
clean_copy['Total.Minor.Injuries'].fillna(clean_copy['Total.Minor.Inju
ries'].median())
```

```
clean_copy['Total.Uninjured'] =
clean_copy['Total.Uninjured'].fillna(clean_copy['Total.Uninjured'].med
ian())
clean_copy['Weather.Condition'] =
clean_copy['Weather.Condition'].fillna(clean_copy['Weather.Condition']
.value_counts().idxmax())
clean_copy['Broad.phase.of.flight'] =
clean_copy['Broad.phase.of.flight'].fillna(clean_copy['Broad.phase.of.
flight'].value_counts().idxmax())
clean_copy['Report.Status'] =
clean_copy['Report.Status'].fillna(clean_copy['Report.Status'].value_c
ounts().idxmax())
clean_copy['Publication.Date'] =
clean_copy['Publication.Date'].fillna(clean_copy['Publication.Date'].v
alue_counts().idxmax())
clean_copy['Injury.Severity'] =
clean_copy['Injury.Severity'].fillna(clean_copy['Injury.Severity'].mod
e()[0])
clean_copy['Aircraft.damage'] =
clean_copy['Aircraft.damage'].fillna(clean_copy['Aircraft.damage'].val
ue_counts().idxmax())
clean_copy.isna().sum()
```

```
Event.Id                    0
Investigation.Type          0
Accident.Number             0
Event.Date                  0
Location                    0
Country                     0
Latitude                    0
Longitude                   0
Airport.Code                0
Airport.Name                0
Injury.Severity             0
Aircraft.damage             0
Aircraft.Category           0
Registration.Number         0
Make                        0
Model                       0
Amateur.Built               0
Number.of.Engines           0
Engine.Type                 0
FAR.Description             0
Schedule                    0
Purpose.of.flight           0
Air.carrier                 0
Total.Fatal.Injuries        0
Total.Serious.Injuries      0
Total.Minor.Injuries        0
Total.Uninjured             0
```

```
Weather.Condition          0
Broad.phase.of.flight     0
Report.Status             0
Publication.Date          0
dtype: int64
```

"""

Since we are dealing with big dataset,It is hard to to fill a row of a dataset with less than 50% content since it may create more bias and so we remove them to avoid misleading data during analysis, also to ensure that subsequent operations like statistical analysis are perfomed on a clean data.

The remaining data set containing more than 50% value in a column are filled with column specific strategy in relation to their data type e.g for numerical values, median or mode can be used , idxmax can be used for categorical columns and unknown used when missing values indicates that the information is unavailable used also for categorical values.

This approach is preferred so as to preserve data and ensure retention of data as much as posible instead of dropping all rows with missing value,It also improves data quality ensuring that the dataset is complete and ready for further analysis without errror casses by NaN values

"""

```
# checking  information on sorted data
clean_copy.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 1732 entries, 5 to 88867
Data columns (total 31 columns):
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   Event.Id                1732 non-null   object 
 1   Investigation.Type      1732 non-null   object 
 2   Accident.Number         1732 non-null   object 
 3   Event.Date              1732 non-null   object 
 4   Location                1732 non-null   object 
 5   Country                 1732 non-null   object 
 6   Latitude                1732 non-null   object 
 7   Longitude               1732 non-null   object 
 8   Airport.Code            1732 non-null   object 
 9   Airport.Name            1732 non-null   object 
 10  Injury.Severity         1732 non-null   object 
 11  Aircraft.damage         1732 non-null   object 
 12  Aircraft.Category       1732 non-null   object 
 13  Registration.Number     1732 non-null   object 
 14  Make                    1732 non-null   object 
 15  Model                   1732 non-null   object 
 16  Amateur.Built           1732 non-null   object 
 17  Number.of.Engines       1732 non-null   float64
```

```
 18  Engine.Type              1732 non-null   object
 19  FAR.Description          1732 non-null   object
 20  Schedule                 1732 non-null   object
 21  Purpose.of.flight        1732 non-null   object
 22  Air.carrier              1732 non-null   object
 23  Total.Fatal.Injuries     1732 non-null   float64
 24  Total.Serious.Injuries   1732 non-null   float64
 25  Total.Minor.Injuries     1732 non-null   float64
 26  Total.Uninjured          1732 non-null   float64
 27  Weather.Condition        1732 non-null   object
 28  Broad.phase.of.flight    1732 non-null   object
 29  Report.Status            1732 non-null   object
 30  Publication.Date         1732 non-null   object
dtypes: float64(5), object(26)
memory usage: 433.0+ KB
```

```
# Saving the cleaned data as csv file
clean_copy.to_csv('cleaned_data.csv', index=False)

#Saving the cleaned data as json file
clean_copy.to_json('cleaned_data.json', orient='records')
```

5. DATA VISUALIZATION

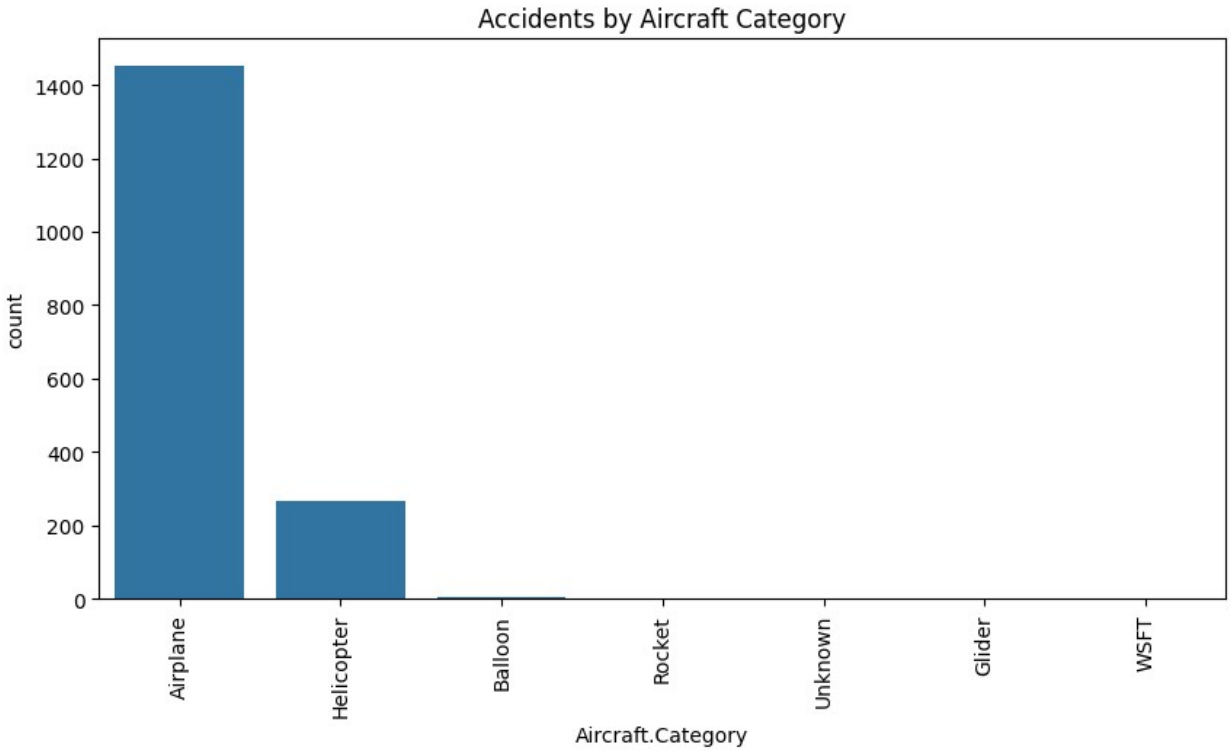
based on the objective of the study,we have cleaned the data and now want to provide more meaningful insights of the data through visualization.

```
import matplotlib.pyplot as plt
import seaborn as sns
```

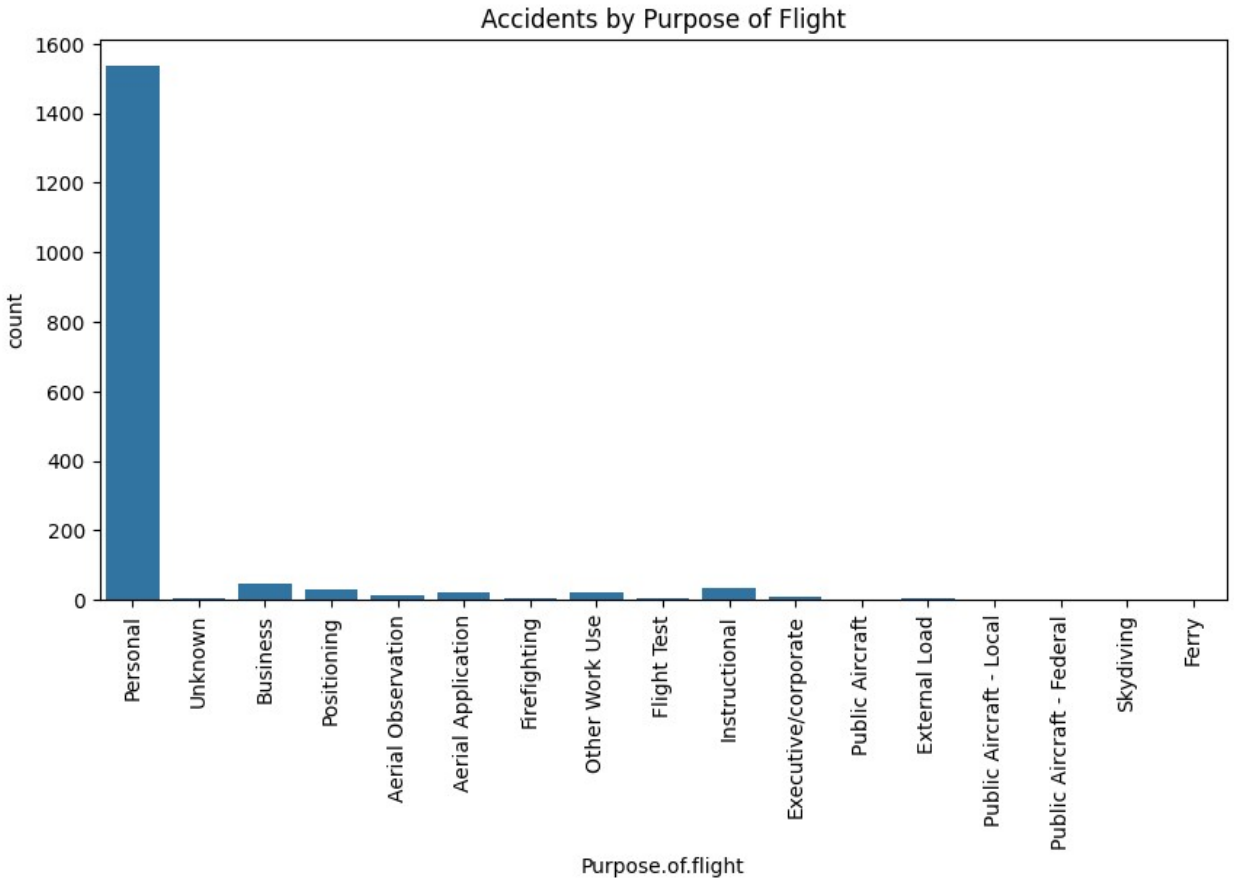
5.1 Univariate Analysis

Method used to analyse individual variable

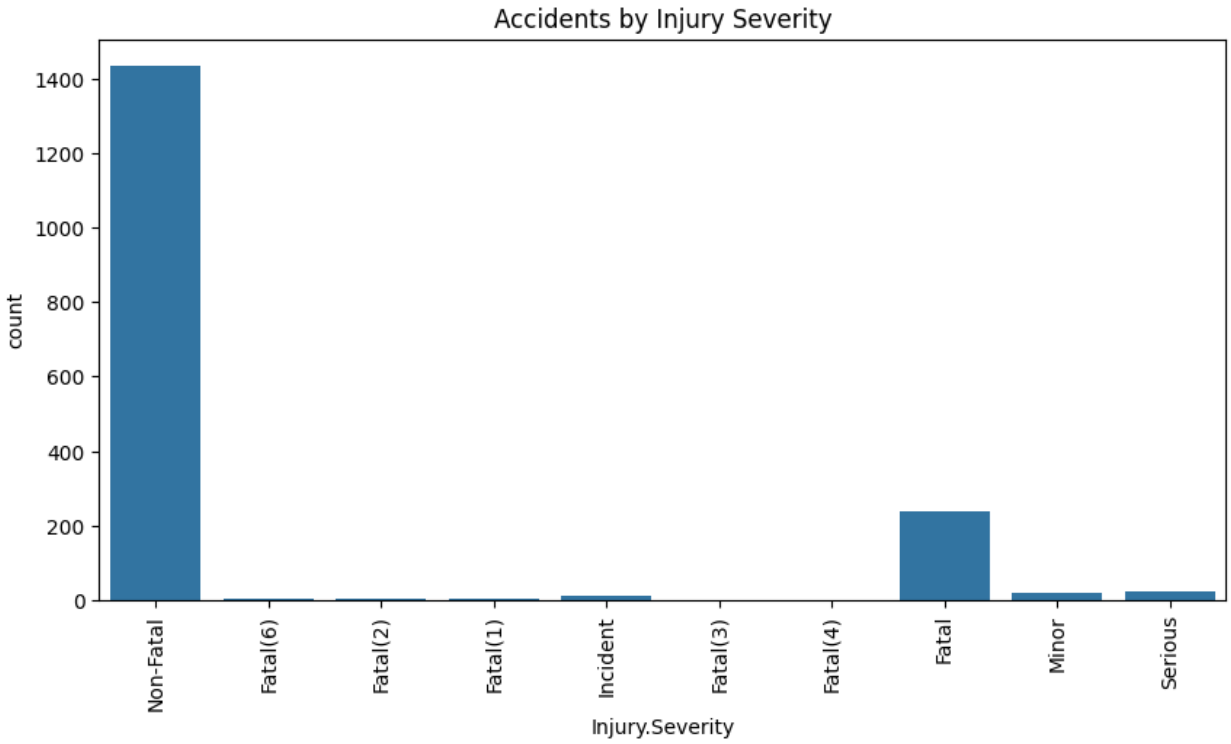
```
# analyzing accidents by aircraft category using a countplot
plt.figure(figsize=(10,5))
sns.countplot(x='Aircraft.Category', data=clean_copy)
plt.xticks(rotation=90)
plt.title('Accidents by Aircraft Category')
plt.show()
```




```
# using bar chat to analyze accidents by purpose of flight
plt.figure(figsize=(10,5))
sns.countplot(x='Purpose.of.flight', data=clean_copy)
plt.xticks(rotation=90)
plt.title('Accidents by Purpose of Flight')
plt.show()
```



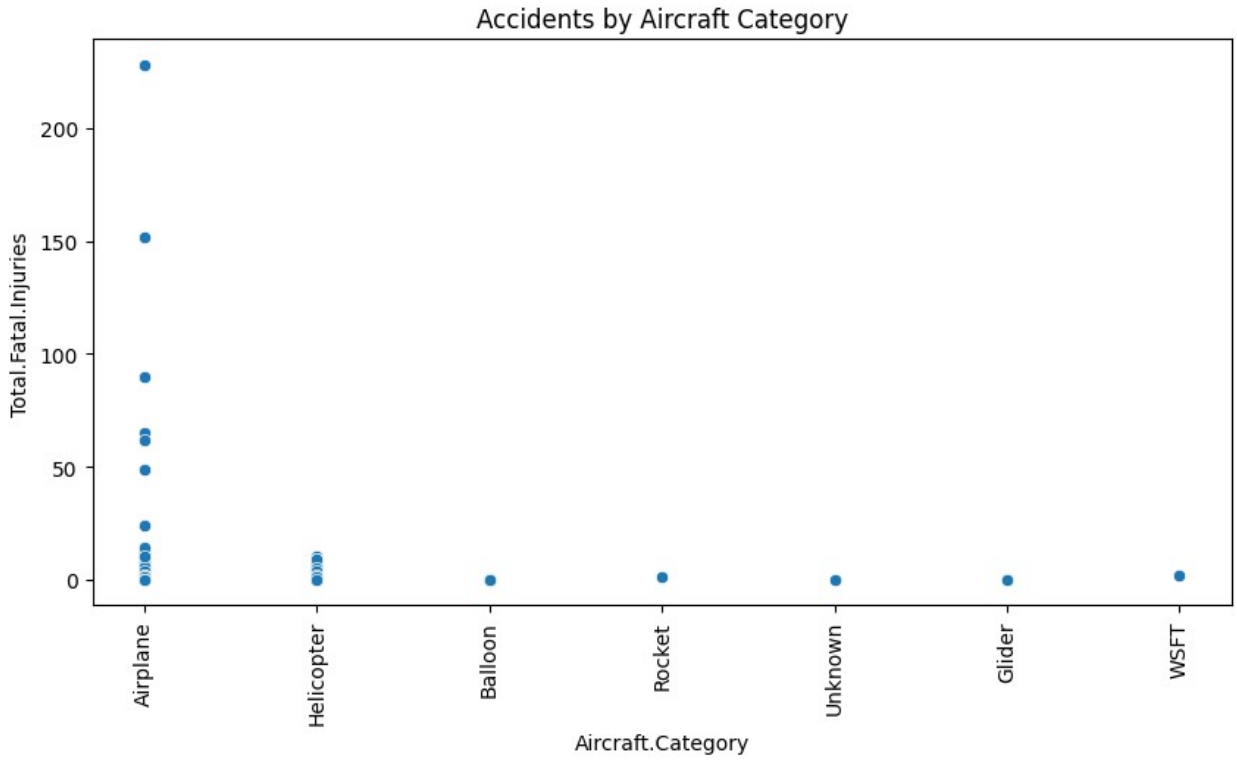

```
# using bar chat to analyze frequency of injury severity in accident
plt.figure(figsize=(10,5))
sns.countplot(x='Injury.Severity', data=clean_copy)
plt.xticks(rotation=90)
plt.title('Accidents by Injury Severity')
plt.show()
```



5.2 Bivariate Analysis:

Method used to explore relationships between two variables

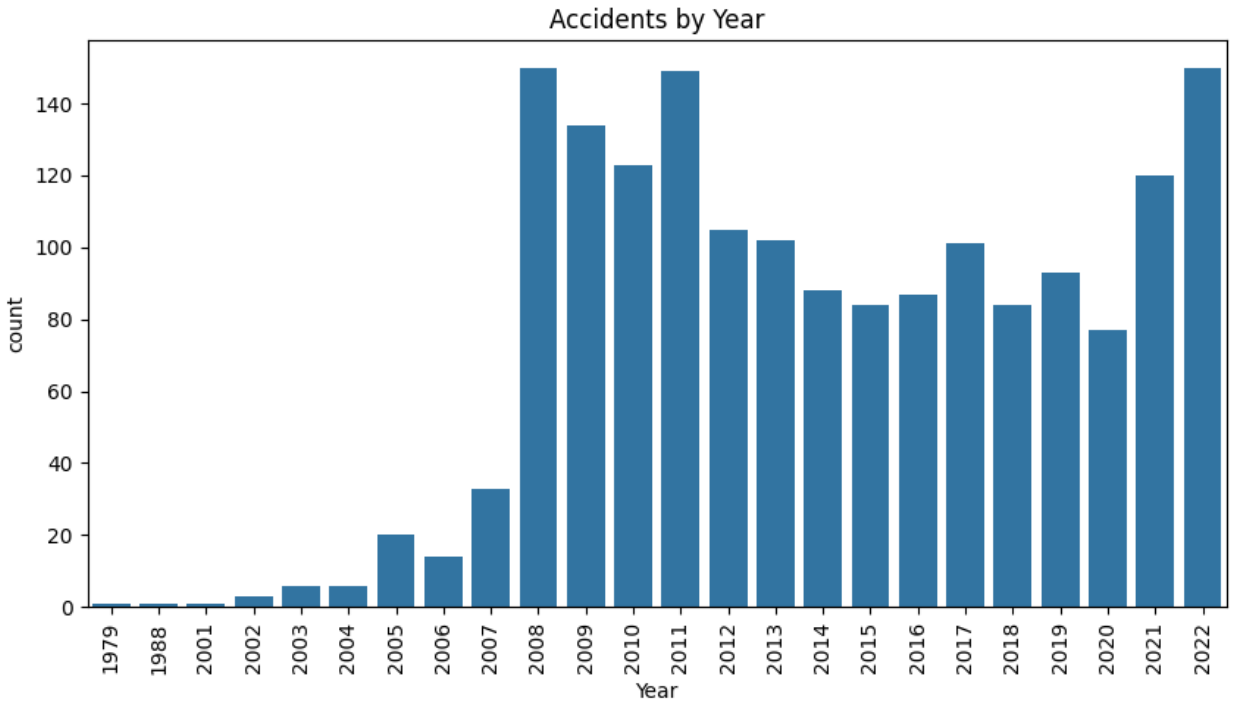
```
# Using scatter plot to analyse accident number by aircraft category
plt.figure(figsize=(10,5))
sns.scatterplot(x='Aircraft.Category', y='Total.Fatal.Injuries',
data=clean_copy)
plt.xticks(rotation=90)
plt.title('Accidents by Aircraft Category')
plt.show()
```



5.3 Trend Analysis:

Method used to identify any patterns over time

```
# change in accident frequency over the years
clean_copy['Event.Date'] = pd.to_datetime(clean_copy['Event.Date'])
clean_copy['Year'] = clean_copy['Event.Date'].dt.year
plt.figure(figsize=(10,5))
sns.countplot(x='Year', data=clean_copy)
plt.xticks(rotation=90)
plt.title('Accidents by Year')
plt.show()
```



6 DATA SUMMARY

Key Findings:

"""

From our findings in univariate analysis, it is evident that there has been more number of airplane accidents recorded over the years followed by helicopters. Aircrafts such as rockets, glider, WSFT had few or no record of accidents. Besides,personal flights has recorded highest number of accidents followed by business flight recording few cases of accident. Other purpose of flight such as public aircraft,external load,skydiver recorded no cases of accidents overtime. most of accidents recorded are non fatal.

From our findings in Bivariate analysis, Airplane has recorded Fatal injuries followed by helicopters.other aircraft such as balloon, gider, rocket has recorded few number of Fatal injuries over time.

From our trend analysis, it is evident that aircraft accidents has been gradually increasing with some significant drop over the years.peak accident years were 2008,2011 and 2022.

"""

7. CONCLUSION

"""

7.1 Recap:

The analysis revealed how the number of accidents has changed over the years. For example, there may be a noticeable increase or decrease in accidents during specific

periods, potentially linked to changes in regulations, technology, or other external factors.

Strong correlations were observed between variables such Aicraft category and total fatal injuries This indicates that number of accident fatalities often corresponds to type of aicraft e.g there was high number of fatal injuries recorded in aeroplane potentially because it carries high number of passangers.

The distribution of Injury.Severity showed which categories e.g. minor, serious, or fatal are most common in the dataset, providing insights into the overall safety trends.

7.2 Limitations:

Several columns had missing values that were filled using statistical methods e.g. median, idxmax and unknown. This imputation may introduce bias or reduce the accuracy of the analysis.

The dataset did not include detailed geographical information e.g. latitude and longitude for all accidents which limited the ability to perform map visualizations.

7.3 Future work:

Develop machine learning models to predict accident severity such as Injury.Severity based on variables such as Aircraft.Category, Weather.Condition, and Purpose.of.flight.

Incorporate detailed geographical data e.g. latitude and longitude to identify accident hotspots and analyze spatial pattern. Study the impact of changes in aviation regulations or safety protocols on accident trends over time.

"""