Data Analysis For Aircraft Analysis. Done by Sheilla Muli

PROJECT OVERVIEW

- The company is expanding into the aviation but lacks knowledge of potential risks across different aiarcrafts.
- This notebook anaalyze avaiation accident data identify low-risk aircraft and provide business recommendations.

Objectives

- Clean and prepare aviation accident data.
- Explore accident trends by year, manufacturer, weather, and flight purpose.
- Identify high-risk and low-risk aircraft manufacturers.
- Provide insights for safer investment decisions in aviation.

```
#import libraries
import pandas as pd
import numpy as np
```

```
#loading the csv file
#csv file is stored in myDrive together with the python code
from google.colab import drive
drive.mount ('/content/drive')
file_path = "/content/drive/My Drive/Dsc-phase1-project/Aviation_Data.csv"
df = pd.read_csv(file_path, low_memory=False)
#to check if it has been loaded
df.head()
df.shape
```

```
Mounted at /content/drive
(90348, 31)
```

```
#Data inspection- to better understand the data set and what I am dealing w
df.info

df.isnull().mean().sort_values(ascending=False).head(20)
```

	0
Schedule	0.860738
Air.carrier	0.815735
FAR.Description	0.645559
Aircraft.Category	0.642637
Longitude	0.619549
Latitude	0.619449
Airport.Code	0.445123
Airport.Name	0.416656
Broad.phase.of.flight	0.316819
Publication.Date	0.184719
Total.Serious.Injuries	0.154613
Total.Minor.Injuries	0.148227
Total.Fatal.Injuries	0.142339
Engine.Type	0.094689
Report.Status	0.086809
Purpose.of.flight	0.084684
Number.of.Engines	0.083488
Total.Uninjured	0.081585
Weather.Condition	0.065868
Aircraft.damage	0.051501

dtype: float64

Data cleaning To do Drop irrelevant columns

- Handle missing values.
- Standardize column names for consistency.
- Convert date fields into datetime format.

```
#data cleaning
#removing colums I do not need
drop_cols = ['Latitude' , 'Longitude', 'Registration.Number' , 'Airport.Co
df = df.drop(columns=[col for col in drop_cols if col in df.columns])

#now I handle missing values in the columns that are not droped
threshold = 0.6
df = df.loc[:, df.isnull().mean() < threshold]

for col in df.select_dtypes(include=['float64','int64']).columns:
    df[col] = df[col].fillna(df[col].median())

for col in df.select_dtypes(include=['object']).columns:
```

```
        df[col] = df[col].fillna(df[col].mode()[0])

    #then I remove duplicates
    df = df.drop_duplicates()

    #get the data types to be the same
    if 'Event.Date' in df.columns:
       df['Event.Date'] = pd.to_datetime(df['Event.Date'], errors='coerce')


    #convert data to int
    for col in ['Total.Fatal.Injuries', 'Total.Serious.Injuries', 'Total.Minor
      if col in df.columns:
         df[col] = pd.to_numeric(df[col], errors='coerce').fillna(0).astype(in

    if 'Event.Date' in df.columns:
      df['Year'] = df['Event.Date'].dt.year
      df['Month'] = df['Event.Date'].dt.month
      df['Day'] = df['Event.Date'].dt.day

    if all(col in df.columns for col in ['Total.Fatal.Injuries','Total.Serious
        df['Total.Injuries'] = (
            df['Total.Fatal.Injuries'] +
            df['Total.Serious.Injuries'] +
            df['Total.Minor.Injuries']
        )

        if 'Total.fatal.Injuries' in df.columns:
          df['Severity'] = np.where(df['Total.Fatal.Injuries'] > 0, 'FATAL', 'I


    for col in ['Total.Fatal.Injuries','Total.Serious.Injuries','Total.Minor.I
        if col in df.columns:
            upper = df[col].quantile(0.99)
            df[col] = np.where(df[col] > upper, upper, df[col])

    #check if everything is upto par
    print("✅ Data Cleaning Done")
    print(df.info())
    print(df.head())
```

```
 17  Weather.Condition       88954 non-null  object
 18  Broad.phase.of.flight   88954 non-null  object
 19  Report.Status           88954 non-null  object
 20  Publication.Date        88954 non-null  object
 21  Year                    88954 non-null  int32
 22  Month                   88954 non-null  int32
 23  Day                     88954 non-null  int32
 24  Total.Injuries          88954 non-null  float64
dtypes: datetime64[ns](1), float64(6), int32(3), object(15)
memory usage: 16.6+ MB
None
         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 Airport.Name Injury.Severity  \
0  MOOSE CREEK, ID  United States      Private        Fatal(2)
1   BRIDGEPORT, CA  United States      Private        Fatal(4)
2    Saltville, VA  United States      Private        Fatal(3)
3       EUREKA, CA  United States      Private        Fatal(2)
4       Canton, OH  United States      Private        Fatal(1)

  Aircraft.damage      Make  ... Total.Minor.Injuries  Total.Uninjured  \
0       Destroyed   Stinson  ...                  0.0              0.0
1       Destroyed     Piper  ...                  0.0              0.0
2       Destroyed    Cessna  ...                  0.0              1.0
3       Destroyed  Rockwell  ...                  0.0              0.0
4       Destroyed    Cessna  ...                  0.0              0.0

  Weather.Condition  Broad.phase.of.flight   Report.Status  Publication.Date  \
0               UNK                 Cruise  Probable Cause        25-09-2020
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

   Year Month Day Total.Injuries
0  1948    10  24            2.0
1  1962     7  19            4.0
2  1974     8  30            3.0
3  1977     6  19            2.0
4  1979     8   2            3.0

[5 rows x 25 columns]
```

```
#clean column names
def clean_column_names (df):
   df = df.copy()
   df.columns = (
   df.columns.str.strip()
      .str.replace('.', '', regex=False)
      .str.replace('/', '_', regex=False)
      .str.replace(' ', '_', regex=False)
```

```
        .str.lower()
    )
    return df

    df = clean_column_names(df)
    df.columns.tolist()
```

```
# trim and normalize case
for c in ['make','model','airport_name','location','air_carrier']:
    if c in df.columns:
        df[c] = df[c].astype(str).str.strip().str.title().replace({'Nan':'Unknown'
```

```
#check if the changes have taken effect and how they look
df.info
df.head

print (df.info)
print (df.head)
```

```
1                0.0              UNK              Unknown
2                1.0              IMC               Cruise
3                0.0              IMC               Cruise
4                0.0              VMC             Approach
...              ...              ...                  ...
90343            0.0              VMC              Landing
90344            0.0              VMC              Landing
90345            1.0              VMC              Landing
90346            0.0              VMC              Landing
90347            1.0              VMC              Landing

        Report.Status  Publication.Date  Year Month Day Total.Injuries
0      Probable Cause        25-09-2020  1948    10  24            2.0
1      Probable Cause        19-09-1996  1962     7  19            4.0
2      Probable Cause        26-02-2007  1974     8  30            3.0
3      Probable Cause        12-09-2000  1977     6  19            2.0
4      Probable Cause        16-04-1980  1979     8   2            3.0
...               ...               ...   ...   ...  ..            ...
90343  Probable Cause        29-12-2022  2022    12  26            1.0
90344  Probable Cause        25-09-2020  2022    12  26            0.0
90345  Probable Cause        27-12-2022  2022    12  26            0.0
90346  Probable Cause        25-09-2020  2022    12  26            0.0
90347  Probable Cause        30-12-2022  2022    12  29            1.0

[88954 rows x 25 columns]>
```

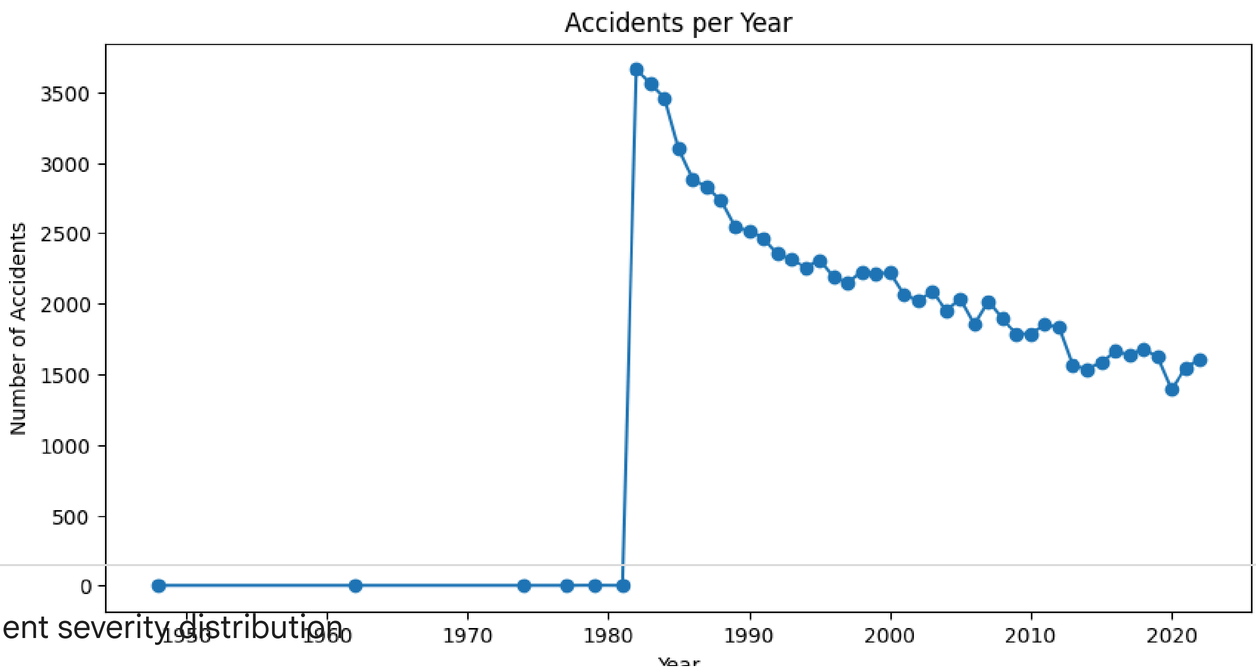
EXPLARATORY DATA ANALYSIS Explore data sets to understand historical patterns

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

Accidents by manufacturer

- Different manufacturers have varying accident rates. This helps identify high-risk and low-risk brands.

```
#Accident Trends over time


accidents_per_year = df['Year'].value_counts().sort_index()
accidents_per_year.plot(kind='line', figsize=(10,5), marker='o')
plt.title("Accidents per Year")
plt.xlabel("Year")
plt.ylabel("Number of Accidents")
plt.show()
```



Accident severity distribution

- Look at the breakdown of fatal vs non-fatal accidents.

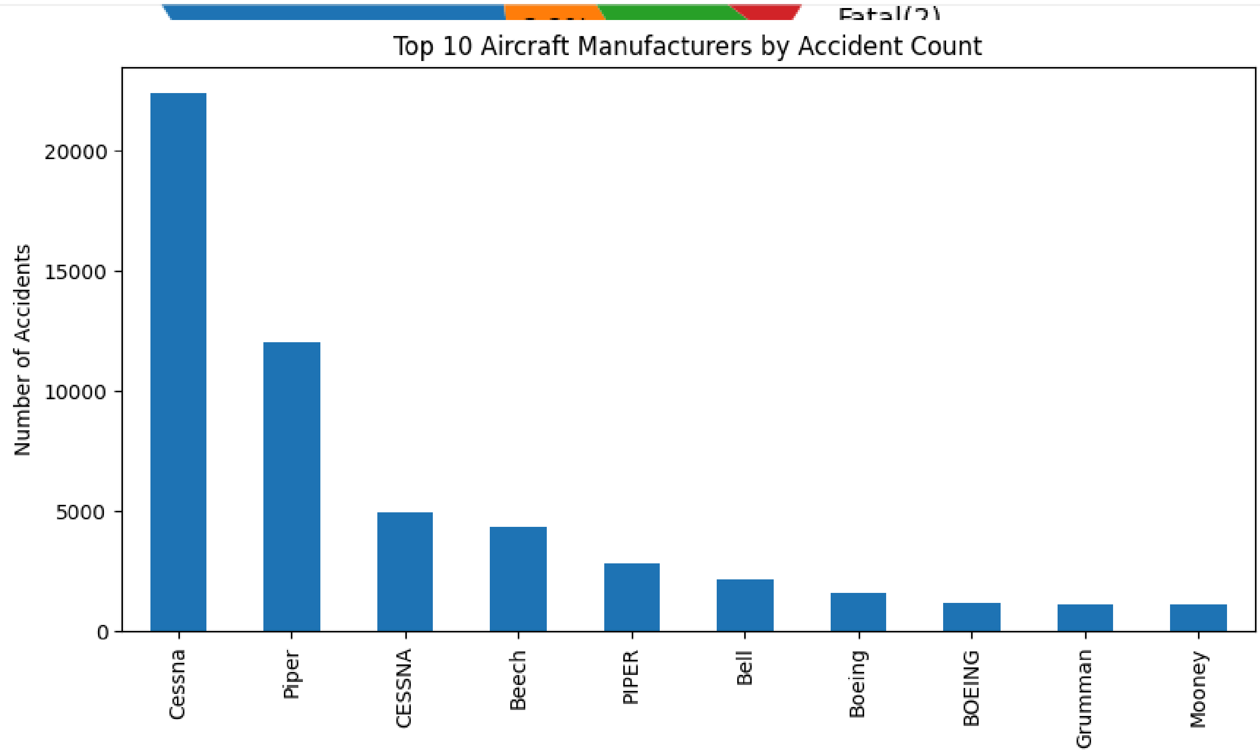
```
#Fatal vs Non-Fatal Accidents
severity_counts = df['Injury.Severity'].value_counts()
severity_counts.plot(kind='pie', autopct='%1.1f%%', figsize=(6,6))
plt.title("Fatal vs Non-Fatal Accidents")
plt.ylabel("")
plt.show()
```

Fatal vs Non-Fatal Accidents

Accidents by aircraft manufactuer

- It is important to understand which comapny prodeces the best aircrafts and how they perform in action in terms of accidents

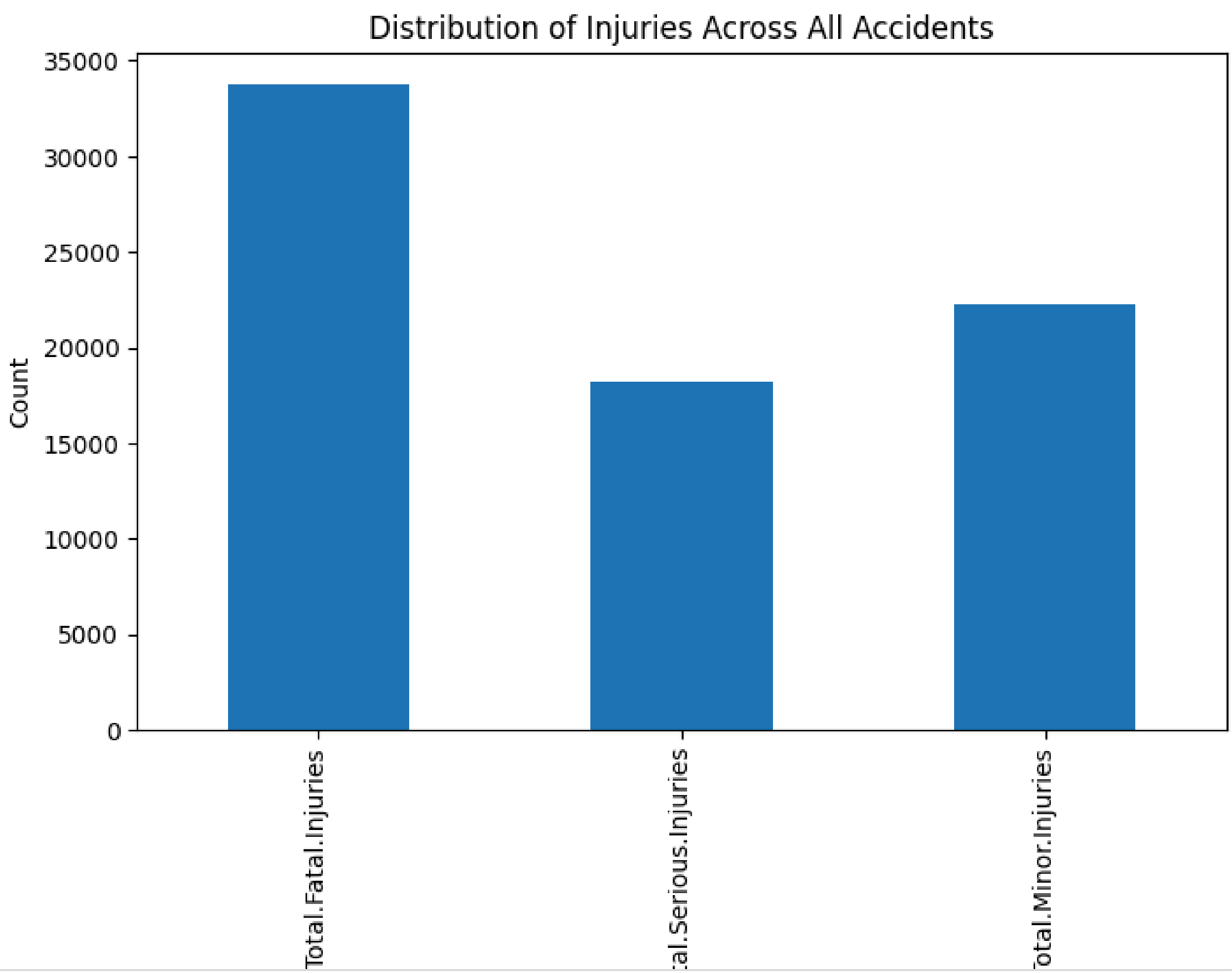
```
#Accidents by aicraft manufacturer (make)

top_makes = df['Make'].value_counts().head(10)
top_makes.plot(kind='bar', figsize=(10,5))
plt.title("Top 10 Aircraft Manufacturers by Accident Count")
plt.xlabel("Manufacturer")
plt.ylabel("Number of Accidents")
plt.show()
```




```
#column mapping to help with the eda stage- not sure if this works honestly
col_map = {
    'event_date':None,
    'make': None,
    'model': None,
    'aircraft_category': None,
    'total_fatal_injuries': None,
    'total_serious_injuries': None,
    'total_minor_injuries': None,
    'total_injuries': None,
    'total_uninjured': None,
```

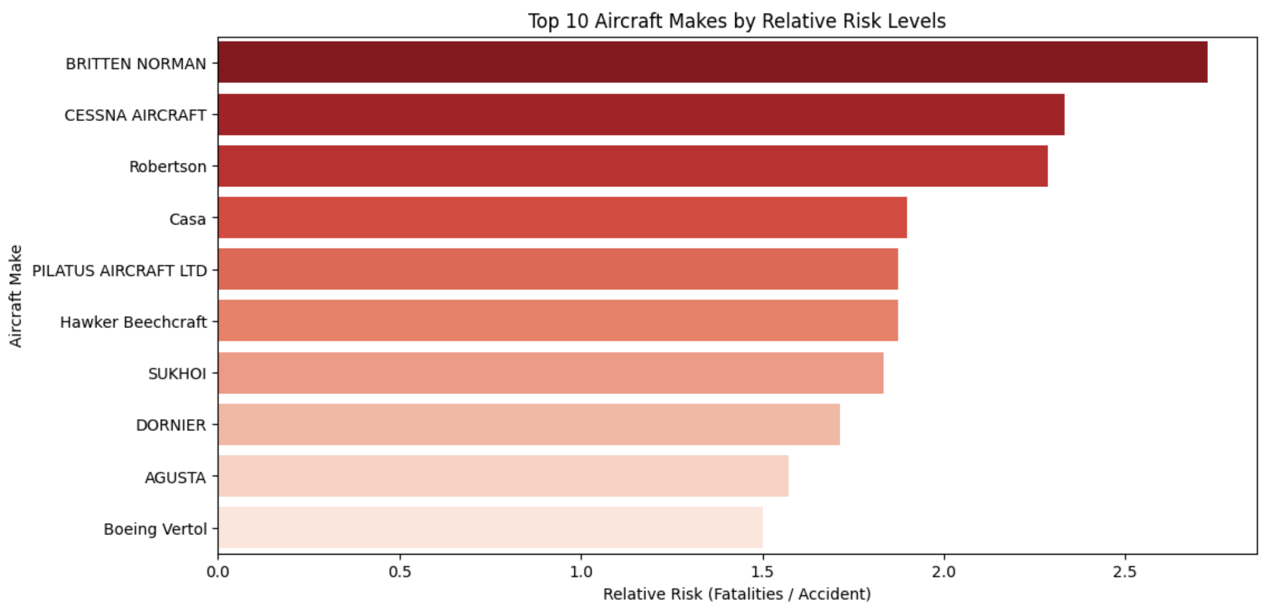
```
    'weather_condition': None,
    'airport_name': None,
    'location': None,
    'air_carrier': None,
    'year': None,
    'month': None,
    'day': None,
    'year_of_manufacture': None,
    'registration_number': None,
    'engine_type': None,
    'engine_manufacturer': None,


}
```

```
#Injury Analysis
injuries = df[['Total.Fatal.Injuries','Total.Serious.Injuries','Total.Minoı
injuries.plot(kind='bar', figsize=(8,5))
plt.title("Distribution of Injuries Across All Accidents")
plt.ylabel("Count")
plt.show()
```



```
#Aircraft models by Relative risk levels
import matplotlib.pyplot as plt
import seaborn as sns

# Create aircraft model column using only 'Make'
df['Aircraft_Model'] = df['Make'].astype(str)

# Group by model and calculate risk
model_stats = df.groupby('Aircraft_Model').agg(
    total_accidents = ('Event.Id', 'count'),
    total_fatalities = ('Total.Fatal.Injuries', 'sum')
).reset_index()

# Calculating relative risk = fatalities per accident
model_stats['Relative_Risk'] = model_stats['total_fatalities'] / model_stats['tota

# Geting top 10 risky models
top_risky = model_stats[model_stats['total_accidents'] > 5].sort_values('Relative_

# Plot
plt.figure(figsize=(12,6))
sns.barplot(x='Relative_Risk', y='Aircraft_Model', data=top_risky, palette="Reds_r
plt.title("Top 10 Aircraft Makes by Relative Risk Levels")
plt.xlabel("Relative Risk (Fatalities / Accident)")
plt.ylabel("Aircraft Make")
plt.show()
```

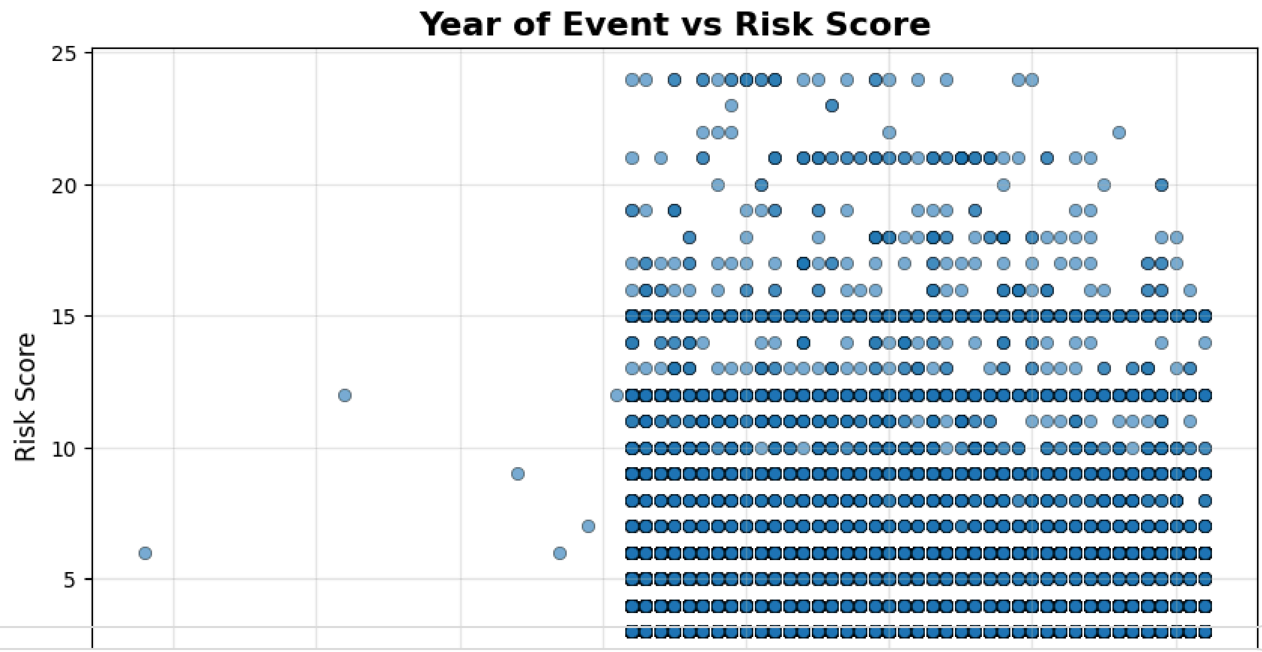
```
/tmp/ipython-input-205383992.py:22: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.1

  sns.barplot(x='Relative_Risk', y='Aircraft_Model', data=top_risky, palette="Reds_
```



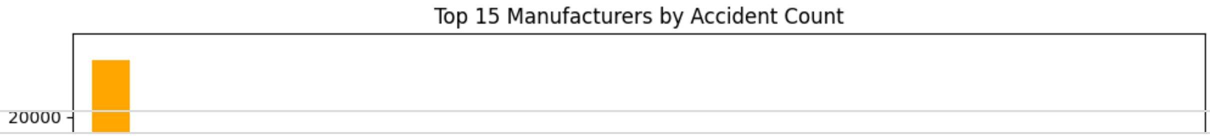

```
#crate a risk score- sice we do not have aircraft age
df['Total.Fatal.Injuries'] = df['Total.Fatal.Injuries'].fillna(0)
```

```
df['Total.Serious.Injuries'] = df['Total.Serious.Injuries'].fillna(0)
df['Total.Minor.Injuries'] = df['Total.Minor.Injuries'].fillna(0)
```

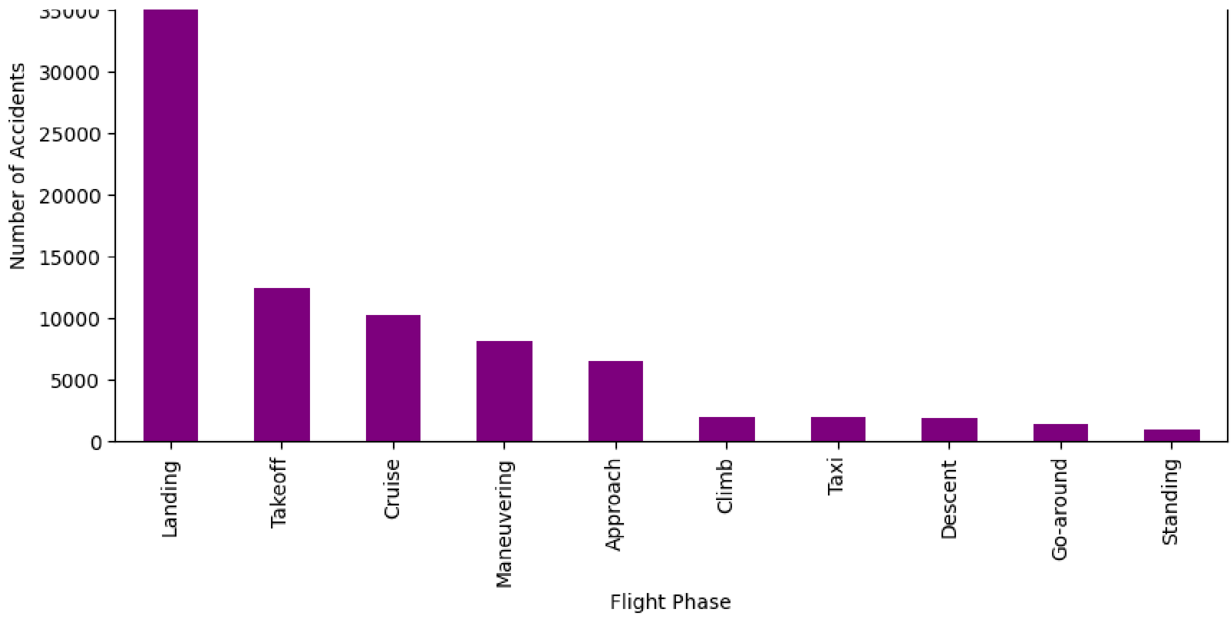
```
#age vs risk


#crating a risk score- sice we do not have aircraft age
df['Risk_Score'] = (df['Total.Fatal.Injuries'] * 3 +
                    df['Total.Serious.Injuries'] * 2 +
                    df['Total.Minor.Injuries'] * 1)
#preparing data for plot
import pandas as pd

# Converting Event.Date to datetime
df['Event.Date'] = pd.to_datetime(df['Event.Date'], errors='coerce')

# Extracting the year of the event
df['Event_Year'] = df['Event.Date'].dt.year

#now we plot
import matplotlib.pyplot as plt
import seaborn as sns
plt.figure(figsize=(10,6))
sns.scatterplot(
    data=df,
    x='Event_Year',
    y='Risk_Score',
    alpha=0.6,
    edgecolor='k'
)

# Adding a trend line
sns.regplot(
    data=df,
    x='Event_Year',
    y='Risk_Score',
    scatter=False,
    color='red',
    line_kws={"linewidth":2}
)

plt.title("Year of Event vs Risk Score", fontsize=16, fontweight='bold')
plt.xlabel("Year of Event", fontsize=12)
plt.ylabel("Risk Score", fontsize=12)
plt.grid(alpha=0.3)
plt.show()
```




```
#Accidents by phase of flight
# By Manufacturer
plt.figure(figsize=(12,6))
df['Make'].value_counts().head(15).plot(kind='bar', color='orange')
plt.title("Top 15 Manufacturers by Accident Count")
plt.xlabel("Manufacturer")
plt.ylabel("Number of Accidents")
plt.show()

# By Broad Phase of Flight
plt.figure(figsize=(10,5))
df['Broad.phase.of.flight'].value_counts().head(10).plot(kind='bar', color=
plt.title("Accidents by Phase of Flight")
plt.xlabel("Flight Phase")
plt.ylabel("Number of Accidents")
plt.show()

# By Weather Condition
plt.figure(figsize=(6,5))
df['Weather.Condition'].value_counts().plot(kind='bar', color='green')
plt.title("Accidents by Weather Condition")
plt.xlabel("Weather")
plt.ylabel("Number of Accidents")
plt.show()
```

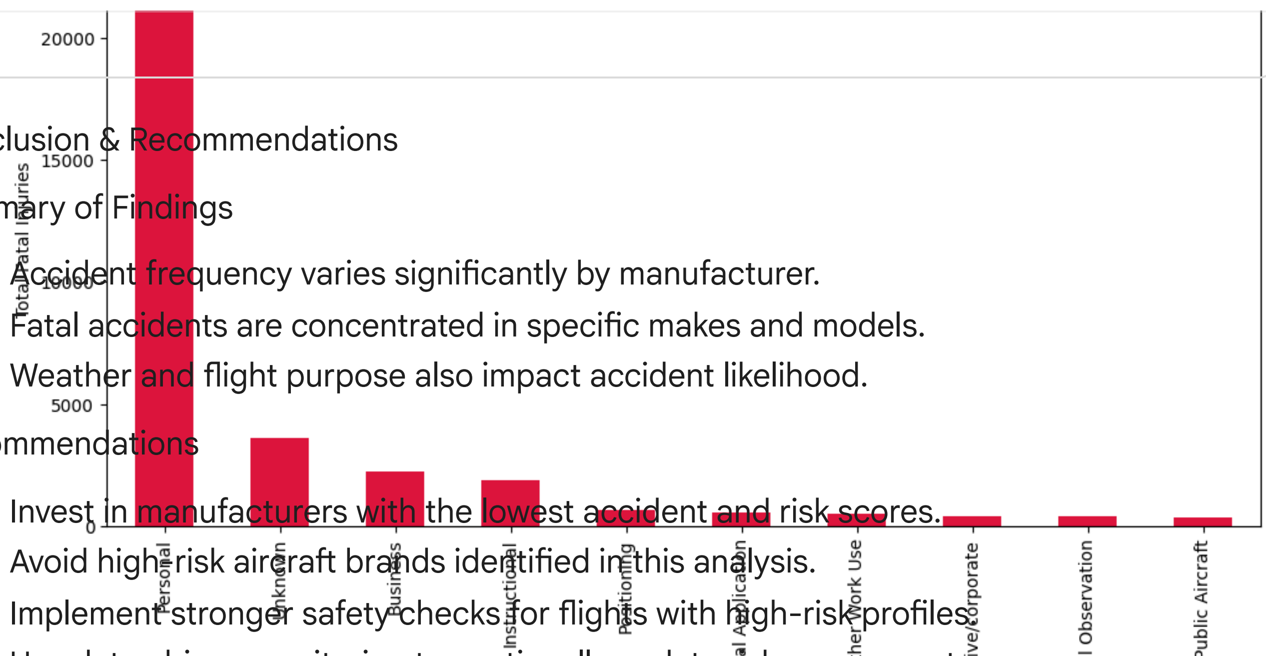



```
#Flight purpose
# Accidents by Purpose of Flight
plt.figure(figsize=(12,6))
df['Purpose.of.flight'].value_counts().head(10).plot(kind='bar', color='ora
plt.title("Accidents by Purpose of Flight")
plt.xlabel("Purpose of Flight")
plt.ylabel("Number of Accidents")
plt.show()

# Fatalities by Purpose of Flight
purpose_fatalities = df.groupby('Purpose.of.flight')['Total.Fatal.Injuries

plt.figure(figsize=(12,6))
purpose_fatalities.plot(kind='bar', color='crimson')
plt.title("Fatalities by Purpose of Flight")
plt.xlabel("Purpose of Flight")
plt.ylabel("Total Fatal Injuries")
plt.show()
```




```
#Saving all the changes
df.to_csv("Aviation_Data_Cleaned.csv", index=False)
```

```
#Downloading files
from google.colab import files

files.download("Aviation_Data_Cleaned.csv")
```



Conclusion & Recommendations

Summary of Findings

- Accident frequency varies significantly by manufacturer.
- Fatal accidents are concentrated in specific makes and models.
- Weather and flight purpose also impact accident likelihood.

Recommendations

1. Invest in manufacturers with the lowest accident and risk scores.
2. Avoid high-risk aircraft brands identified in this analysis.
3. Implement stronger safety checks for flights with high-risk profiles.
4. Use data-driven monitoring to continually update risk assessments.

Next Steps

- Build predictive models for accident likelihood.
- Integrate external datasets (weather, maintenance logs).
- Develop interactive dashboards for real-time risk tracking.