Phase 1 Project

Overview

For this project, you will use data cleaning, imputation, analysis, and visualization to generate insights for a business stakeholder. In order to complete this project, we'll have to import, clean,analysis, and visualize data to answer questions provided, as well as your own questions!

Objectives

You will be able to:

Practice opening and inspecting the contents of CSVs using pandas dataframes Practice identifying and handling missing values Practice identifying and handling invalid values Practice data cleaning Practise data imputation Practise data analysis Practice data visualization

aviationdata

Photo by Yulia Matvienko on Unsplash

Business Understanding

The business questions you have been provided are:

what are potential risks of aircraft? which aircraft are the lowest risk for the company to start this new business endeavor?

whats your recommendation on findings into actionable insights that the head of the new aviation division can use to help decide which aircraft to purchase?

Data Understanding

The Data In the data folder is a datasetLinks to an external site. from the National Transportation Safety Board that includes aviation accident data from 1962 to 2023 about civil aviation accidents and selected incidents in the United States and international waters. In this lab, we'll work with a version of the comprehensive Aviationdata Dataset, which can be found on Kaggle

The data is contained one CSV files:

Aviationdata.csv:About Dataset Content The NTSB aviation accident database contains information from 1962 and later about civil aviation accidents and selected incidents within the United States, its territories and possessions, and in international waters. Acknowledgements Generally, a preliminary report is available online within a few days of an accident. Factual information is added when available, and when the investigation is completed, the preliminary report is replaced with a final description of the accident and its probable cause.

Requirements

1. Load the Data with Pandas

Create a dataframes dfthat represent the CSV file. Use pandas methods to inspect the shape and other attributes of these dataframes.

2. Perform Data Cleaning Required to Answer First Question

The first question is: what are potential risks of aircraft?

In order to answer this question, you will need to:

Practice identifying and handling missing values Practice identifying and handling invalid values Practice data cleaning Practise data imputation

3. Perform Data Analysis and Data Visualization to Answer Second Question

The second question is: which aircraft are the lowest risk for the company to start this new business endeavor?

In order to answer this question, you will need to:

Data Analysis Data Visualization

4. Formulate and Give Recommendations on the insight Answer

This part is fairly open-ended. Think of a question that can be answered with the available data,

1. Load the Data with Pandas In the cell below, we:

Import and alias pandas as pd Import and alias numpy as np Import and alias seaborn as sns Import and alias matplotlib.pyplot as plt Set Matplotlib visualizations to display inline in the notebook

In [6]:
```
#importing of labraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
```

In [7]:
```
#convering csv to pandas
df= pd.read_csv("AviationData.csv", encoding= "latin-1")# encoding=  "latin-1" is used if you have a value error
df.head()
```

```
C:\Users\user\anaconda3\envs\learn-env\lib\site-packages\IPython\core\interactiveshell.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,
```

Out[7]:

	Event.Id	Investigation.Type	Accident.Number	Event.Date	Location	Country	Lati
0	20001218X45444	Accident	SEA87LA080	10/24/1948	MOOSE CREEK, ID	United States	
1	20001218X45447	Accident	LAX94LA336	7/19/1962	BRIDGEPORT, CA	United States	
2	20061025X01555	Accident	NYC07LA005	8/30/1974	Saltville, VA	United States	36.
3	20001218X45448	Accident	LAX96LA321	6/19/1977	EUREKA, CA	United States	
4	20041105X01764	Accident	CHI79FA064	8/2/1979	Canton, OH	United States	

5 rows × 31 columns

It looks like that CSV came with a Value error, There are two ways to do this:

1. Re-load with `read_csv` , and specify the parameter `encoding= "latin-1"`
2. Re-load with `read_csv` , and specify the parameter `encoding= "utf-8"`

Now you want to get familiar with the data. This step includes:

Understanding the dimensionality of your dataset Investigating what type of data it contains, and the data types used to store it Discovering how missing values are encoded, and how many there are Getting a feel for what information it does and doesn't contain In the cell below, inspect the overall,columns,index, shape of the dataframe:Explore the data to understand it check data type,info,decribe,missing values

columns index shape info describe missing values

Now lets look at columns

In [8]:
```
# exploring the that to understand
df.columns
```

Out[8]:
```
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')
```

Now lets look at index

In [9]:
```
#index
df.index
```

Out[9]:
```
RangeIndex(start=0, stop=88889, step=1)
```

Now lets look at shape

In [10]:
```
#shape
df.shape
```

Out[10]:
```
(88889, 31)
```

Now lets look at .info

In [11]:
```
#summary of the data
df.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
```

Now lets look at describe

In [12]:
```
df.describe()
```

Out[12]:

	Number.of.Engines	Total.Fatal.Injuries	Total.Serious.Injuries	Total.Minor.Injuries	Total.Uninj
count	82805.000000	77488.000000	76379.000000	76956.000000	82977.00
mean	1.146585	0.647855	0.279881	0.357061	5.32
std	0.446510	5.485960	1.544084	2.235625	27.91
min	0.000000	0.000000	0.000000	0.000000	0.00
25%	1.000000	0.000000	0.000000	0.000000	0.00
50%	1.000000	0.000000	0.000000	0.000000	1.00
75%	1.000000	0.000000	0.000000	0.000000	2.00
max	8.000000	349.000000	161.000000	380.000000	699.00

Now lets look at Number of missing values

In [13]:
```
df.isnull().sum()# check the count of the various missing values
```

Out[13]:
```
Event.Id                    0
Investigation.Type          0
Accident.Number             0
Event.Date                  0
Location                   52
Country                   226
Latitude                54507
Longitude               54516
Airport.Code            38640
Airport.Name            36099
Injury.Severity          1000
Aircraft.damage          3194
Aircraft.Category       56602
Registration.Number      1317
Make                       63
Model                      92
Amateur.Built             102
Number.of.Engines        6084
Engine.Type              7077
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            6381
Publication.Date        13771
dtype: int64
```

We have noted that there are a lot of missing values from the above cell.

Data Cleaning

We have identified various missing values .Now we will create a strategy on how to clean the data to the correct formart we have drop the columns that we will not be using in our analysis by. drop()

In [14]:
```
# drop unwanted columns
df= df.drop(['Latitude', 'Longitude', 'Airport.Code','Airport.Name', 'Registra
tion.Number','Number.of.Engines', 'Engine.Type', 'FAR.Description',
       'Schedule', 'Report.Status','Publication.Date'], axis ="columns") # hav
e drop the columns
```

confirm if the columns have been droped

In [15]:
```
df.isna().sum() # check the missing values in country column
```

Out[15]:
```
Event.Id                    0
Investigation.Type          0
Accident.Number             0
Event.Date                  0
Location                   52
Country                   226
Injury.Severity          1000
Aircraft.damage          3194
Aircraft.Category       56602
Make                       63
Model                      92
Amateur.Built             102
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
dtype: int64
```

we are filling the missing values in the rows by .fillna() method

In [16]:
```
df["Country"].fillna("Overseas" , inplace= True)# filling the missing with ove
rseas
```

Replacing all the rows with missing values with "Unknown"

In [17]:
```
df["Location"].fillna("Unknown", inplace =True)# we are filling the rows of ca
tegorical columns
df["Aircraft.damage"].fillna("Unknown", inplace =True)# we are filling the row
s of categorical columns
df['Aircraft.Category'].fillna("Unknown", inplace =True)# we are filling the r
ows of categorical columns
df["Make"].fillna("Unknown", inplace =True)# we are filling the rows of catego
rical columns
df["Model"].fillna("Unknown", inplace =True)# we are filling the rows of categ
orical columns
df["Purpose.of.flight"].fillna("Unknown", inplace =True)# we are filling the r
ows of categorical columns
df["Air.carrier"].fillna("Unknown", inplace =True)# we are filling the rows of
categorical columns
df["Weather.Condition"].fillna("Unknown", inplace =True)# we are filling the r
ows of categorical columns
df["Broad.phase.of.flight"].fillna("Unknown", inplace =True)# we are filling t
he rows of categorical columns
df["Amateur.Built"].fillna("Unknown", inplace = True)# we are filling the rows
of categorical columns
df["Injury.Severity"].fillna("Unknown", inplace = True)# we are filling the ro
ws of categorical columns
```

In [18]:
```
df.isna().sum()#checking for the missing values if they have been filled
```

Out[18]:
```
Event.Id                    0
Investigation.Type          0
Accident.Number             0
Event.Date                  0
Location                    0
Country                     0
Injury.Severity             0
Aircraft.damage             0
Aircraft.Category           0
Make                        0
Model                       0
Amateur.Built               0
Purpose.of.flight           0
Air.carrier                 0
Total.Fatal.Injuries    11401
Total.Serious.Injuries  12510
Total.Minor.Injuries    11933
Total.Uninjured          5912
Weather.Condition           0
Broad.phase.of.flight       0
dtype: int64
```

From the above steps we were able to identify missing In order to answer this question, you will need to:

Practice identifying and handling missing values Practice identifying and handling invalid values Practice data cleaning Practise data imputatio Practice identifying and handling missing values Practice identifying and handling invalid values Practice data cleaning Practise data imputatio

In [19]:
```
df.columns
```

Out[19]:
```
Index(['Event.Id', 'Investigation.Type', 'Accident.Number', 'Event.Date',
       'Location', 'Country', 'Injury.Severity', 'Aircraft.damage',
       'Aircraft.Category', 'Make', 'Model', 'Amateur.Built',
       'Purpose.of.flight', 'Air.carrier', 'Total.Fatal.Injuries',
       'Total.Serious.Injuries', 'Total.Minor.Injuries', 'Total.Uninjured',
       'Weather.Condition', 'Broad.phase.of.flight'],
      dtype='object')
```

In [20]:
```
df= df[df['Aircraft.Category']!= "Unknown"]# droping the unknown in df["Aircra
ft.Category"]
```

In [21]:
```
df["Aircraft.Category"].unique() #confirming if the unknown has droped
```

Out[21]:
```
array(['Airplane', 'Helicopter', 'Glider', 'Balloon', 'Gyrocraft',
       'Ultralight', 'Blimp', 'Powered-Lift', 'Weight-Shift',
       'Powered Parachute', 'Rocket', 'WSFT', 'UNK', 'ULTR'], dtype=object)
```

In [22]:
```
df['Total.Fatal.Injuries'].median()
```

Out[22]:
```
0.0
```

Filling in the missing value with their median for numerical values

In [ ]:
```
#filling of numerical columns by their median
df = df.fillna(df.median())
```

In [ ]:
```
df.isnull().sum()# checking if all the missing values are replaced
```

Data Visualization

We will visualixe the overall graph of accidents from 1962. we

In [ ]:
```
import matplotlib.pyplot as plt
%matplotlib inline
```

In [ ]:
```
import matplotlib.pyplot as plt


# Assuming your DataFrame is named 'df'
# Convert 'Event.Date' to datetime objects and extract the year
df['Event.Date'] = pd.to_datetime(df['Event.Date'], errors='coerce')  # Handle
potential errors
df['Year'] = df['Event.Date'].dt.year

# Group by year and count the number of accidents
accidents_per_year = df.groupby('Year')['Event.Id'].count().reset_index()

# Create the line plot
plt.plot(accidents_per_year['Year'], accidents_per_year['Event.Id'], marker
='o', linestyle='-', color='b')

# Add labels and title
plt.xlabel('Year')
plt.ylabel('Number of Accidents')
plt.title('Number of Accidents Over the Years')

# Display the graph
plt.grid(True)
plt.show()
```

In [35]:
```
df= df[df['Weather.Condition']!= "Unknown"] # droping the unknows
df= df[df['Weather.Condition']!= "UNK"]
df= df[df['Weather.Condition']!= "Unk"]
```

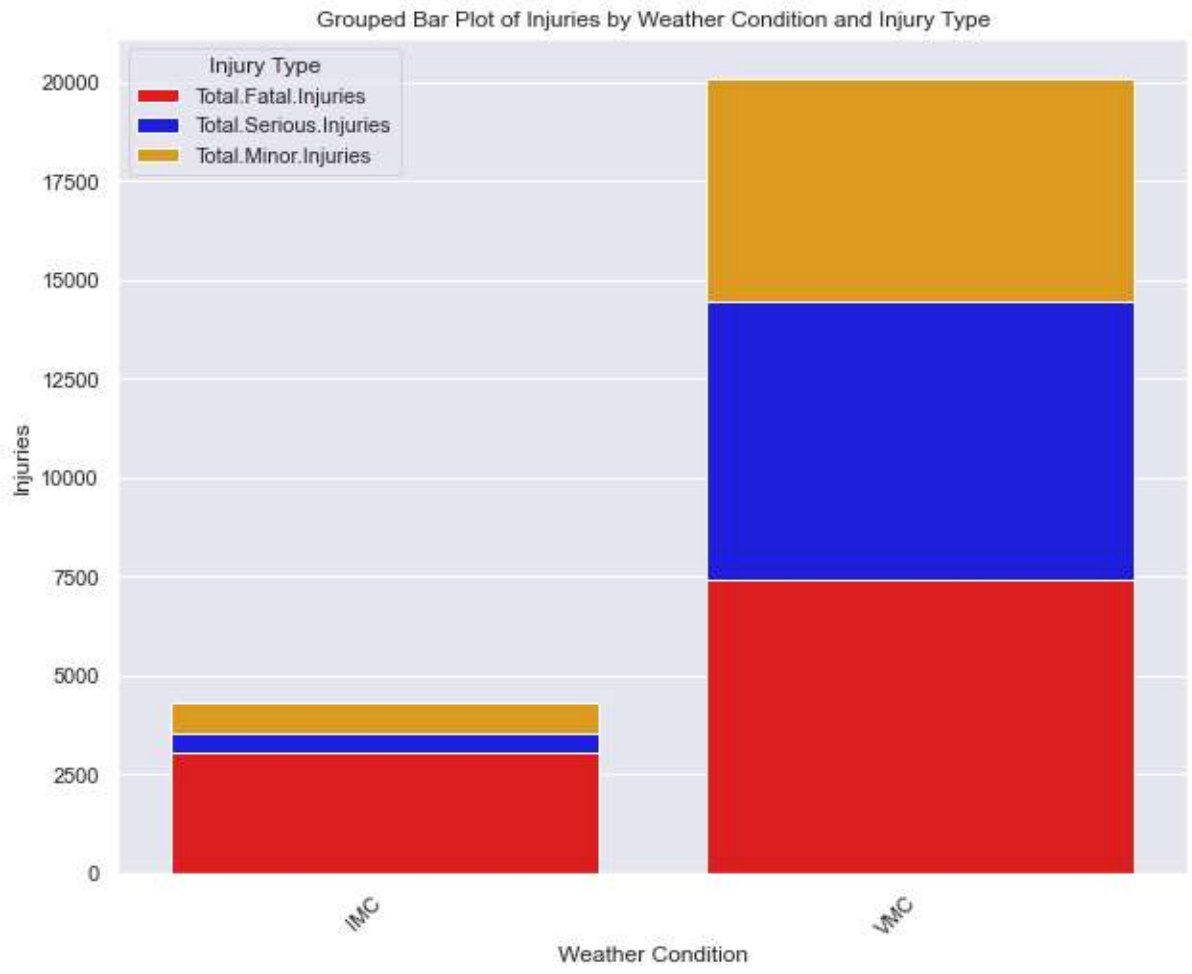
In [36]:

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

plt.figure(figsize=(10, 8))

# Calculate total injuries for each Weather.Condition
injury_counts = df.groupby('Weather.Condition')[['Total.Fatal.Injuries', 'Tota
l.Serious.Injuries', 'Total.Minor.Injuries']].sum().reset_index()

# Plot stacked bars
sns.barplot(x='Weather.Condition', y='Total.Fatal.Injuries', data=injury_count
s, color="red", label="Total.Fatal.Injuries")
sns.barplot(x='Weather.Condition', y='Total.Serious.Injuries', data=injury_cou
nts, color="blue", label="Total.Serious.Injuries", bottom=injury_counts['Tota
l.Fatal.Injuries'])
sns.barplot(x='Weather.Condition', y='Total.Minor.Injuries', data=injury_count
s, color="orange", label="Total.Minor.Injuries", bottom=injury_counts['Total.F
atal.Injuries'] + injury_counts['Total.Serious.Injuries'])
#plot enhance
plt.title("Grouped Bar Plot of Injuries by Weather Condition and Injury Type")
plt.xlabel("Weather Condition")
plt.ylabel("Injuries")
#display the graph
plt.legend(title="Injury Type")
plt.xticks(rotation=45, ha="right")
plt.show()
```



from the bar graph we can see the weather is a risk factor for accidents inthe aviation section. the graph show that there are many total minor injuries as compare to the serious and fatal injuries.

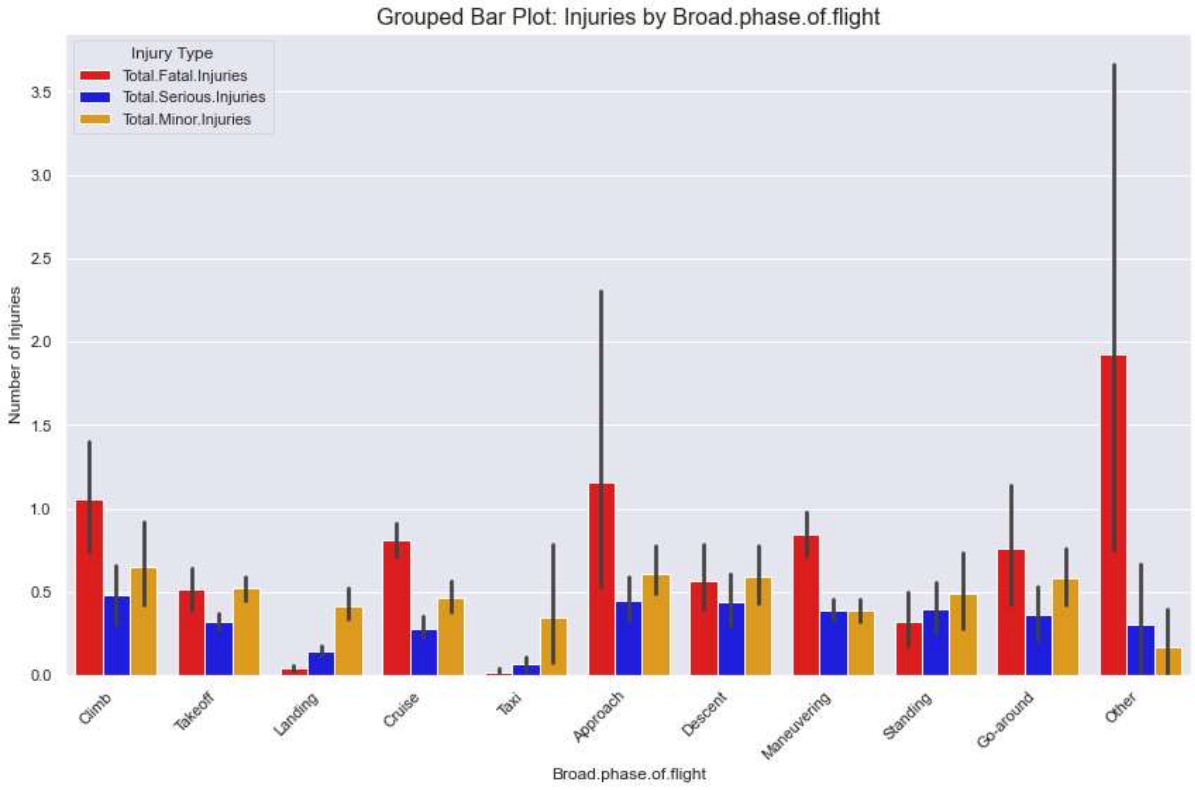
In [41]:
```
df= df[df['Broad.phase.of.flight']!= "Unknown"]#droping the unknown in the row
s
```

In [42]:

```
#ploting of a bar graph
# Melt the DataFrame to long format
df_melted = df.melt(
    id_vars="Broad.phase.of.flight",
    value_vars=["Total.Fatal.Injuries", "Total.Serious.Injuries", "Total.Mino
r.Injuries"],
    var_name="Injury.Type",
    value_name="Injuries"
)

# Create the grouped bar plot
plt.figure(figsize=(12, 8))
sns.barplot(
    x="Broad.phase.of.flight",
    y="Injuries",
    hue="Injury.Type",  # Group bars by Injury.Type
    data=df_melted,
    palette={"Total.Fatal.Injuries": "red", "Total.Serious.Injuries": "blue",
"Total.Minor.Injuries": "orange"}
)

# Enhance plot
plt.title("Grouped Bar Plot: Injuries by Broad.phase.of.flight", fontsize=16)
plt.xlabel("Broad.phase.of.flight", fontsize=12)
plt.ylabel("Number of Injuries", fontsize=12)
plt.xticks(rotation=45, ha="right")  # Rotate x-axis labels
plt.legend(title="Injury Type")  # Add legend
plt.tight_layout()
plt.show()
```



We can note that Phase of Flight is a risk factor to aviation industry. Variable like thr Approach,Climb,Cruise, Maneuvering if they aren't taken serious may lead to accidents

In [45]:
```
df= df[df['Purpose.of.flight']!= "Unknown"]# droping the unknown in the rows
```

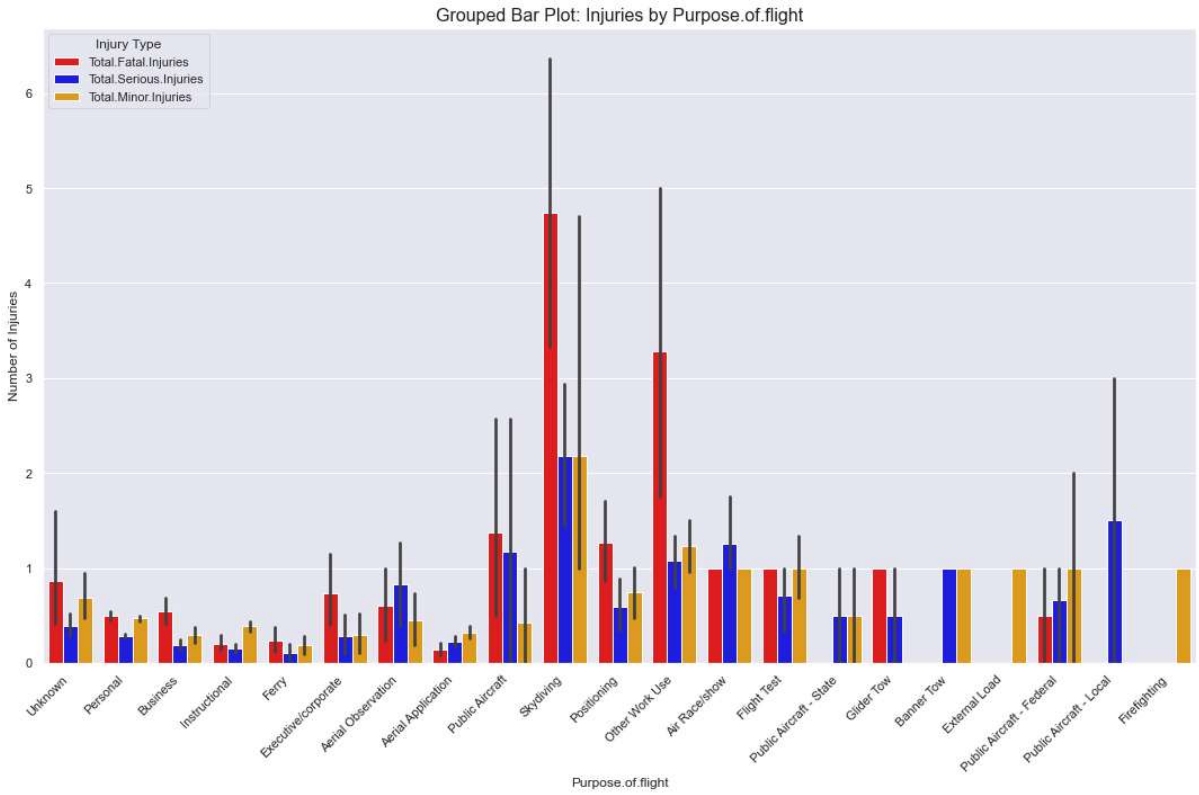
In [44]:

```
df_melted = df.melt(
    id_vars="Purpose.of.flight",
    value_vars=["Total.Fatal.Injuries", "Total.Serious.Injuries", "Total.Mino
r.Injuries"],
    var_name="Injury.Type",
    value_name="Injuries"
)

# Create the grouped bar plot
plt.figure(figsize=(15,10))
sns.barplot(
    x="Purpose.of.flight",
    y="Injuries",
    hue="Injury.Type",  # Group bars by Injury.Type
    data=df_melted,
    palette={"Total.Fatal.Injuries": "red", "Total.Serious.Injuries": "blue",
"Total.Minor.Injuries": "orange"}
)

# Enhance plot
plt.title("Grouped Bar Plot: Injuries by Purpose.of.flight", fontsize=16)
plt.xlabel("Purpose.of.flight", fontsize=12)
plt.ylabel("Number of Injuries", fontsize=12)
plt.xticks(rotation=45, ha="right")  # Rotate x-axis labels
plt.legend(title="Injury Type")  # Add legend
plt.tight_layout()
plt.show()
```

```
C:\Users\user\anaconda3\envs\learn-env\lib\site-packages\seaborn\algorithms.p
y:98: RuntimeWarning: Mean of empty slice
  boot_dist.append(f(*sample, **func_kwargs))
C:\Users\user\anaconda3\envs\learn-env\lib\site-packages\numpy\lib\nanfunctio
ns.py:1556: RuntimeWarning: All-NaN slice encountered
  # so deal them upfront
```



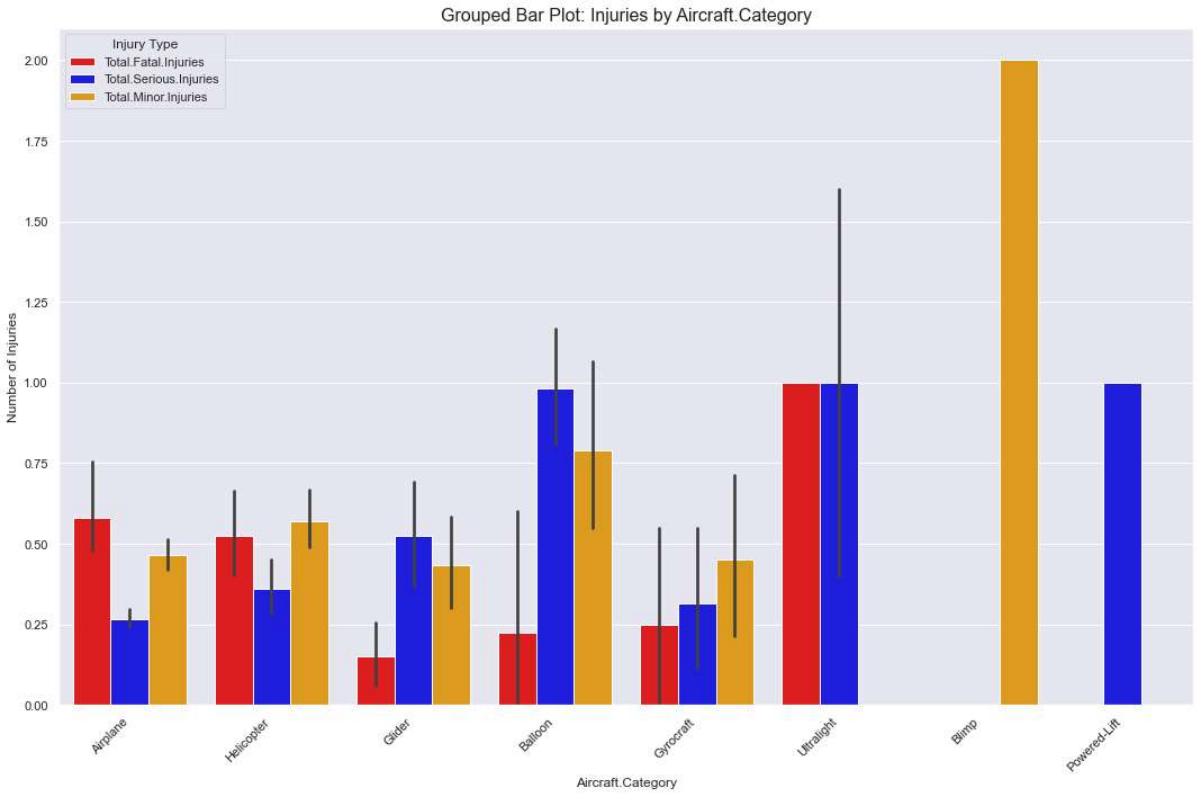
In [49]:

```
df_melted = df.melt(
    id_vars="Aircraft.Category",
    value_vars=["Total.Fatal.Injuries", "Total.Serious.Injuries", "Total.Mino
r.Injuries"],
    var_name="Injury.Type",
    value_name="Injuries"
)

# Create the grouped bar plot
plt.figure(figsize=(15,10))
sns.barplot(
    x="Aircraft.Category",
    y="Injuries",
    hue="Injury.Type",  # Group bars by Injury.Type
    data=df_melted,
    palette={"Total.Fatal.Injuries": "red", "Total.Serious.Injuries": "blue",
"Total.Minor.Injuries": "orange"}
)

# Enhance plot
plt.title("Grouped Bar Plot: Injuries by Aircraft.Category", fontsize=16)
plt.xlabel("Aircraft.Category", fontsize=12)
plt.ylabel("Number of Injuries", fontsize=12)
plt.xticks(rotation=45, ha="right")  # Rotate x-axis labels
plt.legend(title="Injury Type")  # Add legend
plt.tight_layout()
plt.show()
```

```
C:\Users\user\anaconda3\envs\learn-env\lib\site-packages\seaborn\algorithms.p
y:98: RuntimeWarning: Mean of empty slice
  boot_dist.append(f(*sample, **func_kwargs))
C:\Users\user\anaconda3\envs\learn-env\lib\site-packages\numpy\lib\nanfunctio
ns.py:1556: RuntimeWarning: All-NaN slice encountered
  # so deal them upfront
```



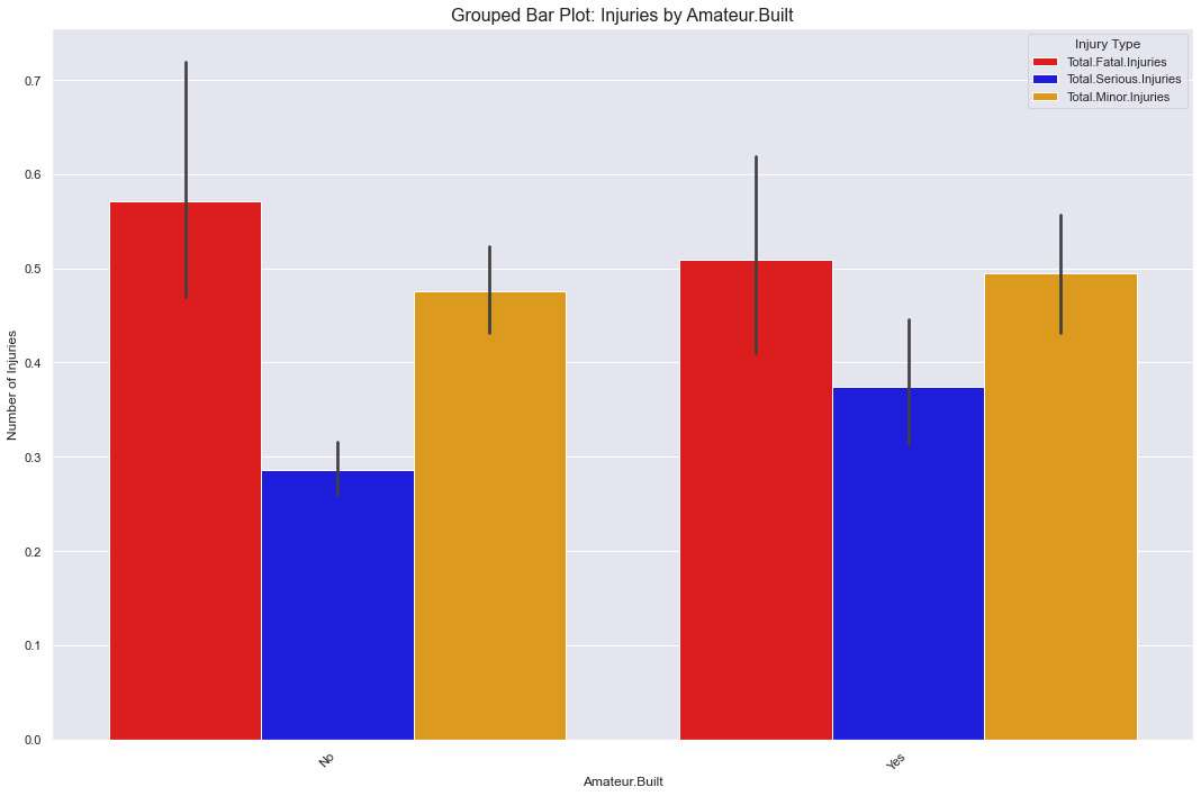
In [ ]:

In [47]:
```
df= df[df['Amateur.Built']!= "Unknown"]
```

In [48]:
```
df_melted = df.melt(
    id_vars="Amateur.Built",
    value_vars=["Total.Fatal.Injuries", "Total.Serious.Injuries", "Total.Mino
r.Injuries"],
    var_name="Injury.Type",
    value_name="Injuries"
)

# Create the grouped bar plot
plt.figure(figsize=(15,10))
sns.barplot(
    x="Amateur.Built",
    y="Injuries",
    hue="Injury.Type",  # Group bars by Injury.Type
    data=df_melted,
    palette={"Total.Fatal.Injuries": "red", "Total.Serious.Injuries": "blue",
"Total.Minor.Injuries": "orange"}
)

# Enhance plot
plt.title("Grouped Bar Plot: Injuries by Amateur.Built", fontsize=16)
plt.xlabel("Amateur.Built", fontsize=12)
plt.ylabel("Number of Injuries", fontsize=12)
plt.xticks(rotation=45, ha="right")  # Rotate x-axis labels
plt.legend(title="Injury Type")  # Add legend
plt.tight_layout()
plt.show()
```



In [ ]:
```
we can observe that there is a higher risk of having air craft make from ameat
ure as compared to establish to companies
```

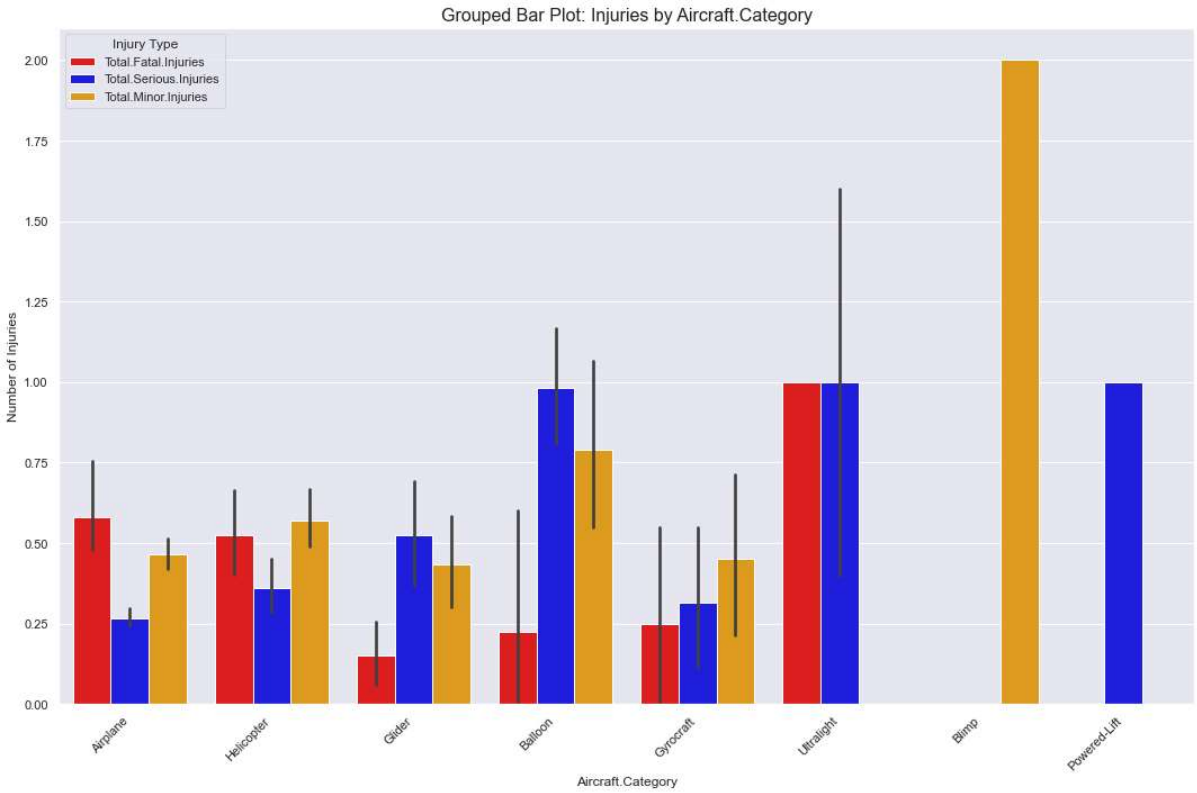
Perform Data Analysis and Data Visualization to Answer Second Question

In [49]:
```
df_melted = df.melt(
    id_vars="Aircraft.Category",
    value_vars=["Total.Fatal.Injuries", "Total.Serious.Injuries", "Total.Mino
r.Injuries"],
    var_name="Injury.Type",
    value_name="Injuries"
)

# Create the grouped bar plot
plt.figure(figsize=(15,10))
sns.barplot(
    x="Aircraft.Category",
    y="Injuries",
    hue="Injury.Type",  # Group bars by Injury.Type
    data=df_melted,
    palette={"Total.Fatal.Injuries": "red", "Total.Serious.Injuries": "blue",
"Total.Minor.Injuries": "orange"}
)

# Enhance plot
plt.title("Grouped Bar Plot: Injuries by Aircraft.Category", fontsize=16)
plt.xlabel("Aircraft.Category", fontsize=12)
plt.ylabel("Number of Injuries", fontsize=12)
plt.xticks(rotation=45, ha="right")  # Rotate x-axis labels
plt.legend(title="Injury Type")  # Add legend
plt.tight_layout()
plt.show()
```

```
C:\Users\user\anaconda3\envs\learn-env\lib\site-packages\seaborn\algorithms.p
y:98: RuntimeWarning: Mean of empty slice
  boot_dist.append(f(*sample, **func_kwargs))
C:\Users\user\anaconda3\envs\learn-env\lib\site-packages\numpy\lib\nanfunctio
ns.py:1556: RuntimeWarning: All-NaN slice encountered
  # so deal them upfront
```



Formulate and Give Recommendations on the insight Answer

Key Observation

1 Accident Trends Over Time Accidents have fluctuated over the decades, with spikes observed in earlier years. The number of a decrease in recent years reflecting improvements in aviation safety, technology, and regulations

2 Weather Condition IMC (Instrument Meteorological Conditions): Accidents during adverse weather result in more fatal injuiries and 2,732 minor Injuries. VMC (Visual Meteorological Conditions): While VMC conditions account for more total injuries higher volume of fatal injuries under these conditions. 3 compared to takeoff or landing, cruise-phase accidents tend to be more catastrophic. Takeoff has significant injury counts fatalties, hìghlighting the risks of this c‧ritical phase. Maneuvering also records a high number of fatalities (5,323), indicating higher across standard fight maneuvers.

1. Top 10 Aircraft Manufacturers with Miost Accidents Cessna and Piper dominate accident records, with 22,227 and 12,029 accidents nanufacturers are prevalent in personal and recreational aviation. Commercial Manufacturers: Boeing and Airbus eport fewer accid advanced safety features and operational standards. Observation: Selecting modern commercial aircraft with proven safety records 6 Top 10 Countries with Most Accidents The United States leads in accident counts which is expected due to its large aviation sect

Recommendation

ircraft Selection Recommendation:

1.Focus on acquiring modern commercial aircraft from manufacturers with a strong safety trackrecoed Airbus) while avoiding older or recreational aircraft models from high-risk manufacturers (e.g., Cessna, Piper) jecreational aviation. Commercial aircraft from Boeing and Airbus have advanced safety features, robust engineering and adherer standards

1. Prioritize Critical Flight Phases Recommendation: Implement targeted training and automated safety systems for high-risk flight phases i.e cruise, takeoff, and maneuvering. Rationale: Crulse phase accldents are less frequent but highly catastrophic. Takeoff and maneuver have significant injury counts, requiring precision and real-time decision-making.
2. Enhance Weather-Related Safety Recommendation: Develop stringent protocols and pilot training programs for operating in adversity condition (IMC). supported y advanced avionics systems such as predictlve weather radar and real-time navigation aids. Rationale: IMC con increase the severity of accidents compared to VMC, highlighting the mportance of preparing for adverse weather scenarios
3. Target Personal Aviation Risks Recommendation invest in initiatives to enhance safety in personal aviation, including: Improved training pilots. Stricter maintenance standards for personal and recreational aircraft. Educationa programs on weather and operational safety Rationale: Personal flights account for the majonity of accidents, often due to variability in pilot experience and regulatory oversight.
4. Regional Safety Initiatives Recommendation: Focus safety improvement efforts in regions with the highest accident counts, particularl States and other countries with notable risks. Rationale: Geographical concentration of accidents indicates areas where targeted safety yield significant impact
5. Invest in Technology-Driven Safety Recommendation: Equip aircraft with modern safety technologies, including: Automated emergency Enhanced collision avoidance systems. Real-time engine and system monitoring tools, Rationale: Advanced technologies mitigate hym improve outcomes in critical situations