# Aviation Accident Data Analysis

## Business Understanding

The company is expanding into the aviation industry and needs to identify the lowest-risk aircraft for purchase and operation. This analysis will help the company make data-driven decisions to minimize risks and ensure safety.

## Data Understanding

The dataset used is the **NTSB Aviation Accident Database**, which contains information about civil aviation accidents and incidents in the United States and international waters from 1962 to 2023. Key columns include:

- `Event.Date` : Date of the accident.
- `Injury.Severity` : Severity of injuries (e.g., Fatal, Serious, Minor).
- `Aircraft.damage` : Extent of damage to the aircraft.
- `Make` : Manufacturer and model of the aircraft.
- `Engine.Type` : Engine details.
- `Total.Fatal.Injuries` , `Total.Serious.Injuries` , `Total.Minor.Injuries` , `Total.Uninjured` : Injury statistics.
- `Weather.Condition and Broad.phase.of.flight` : Contextual details about the accident.

## Data Preparation

We start by downloading and cleaning the dataset to ensure data integrity

In [15]:
```
import os
import pandas as pd
import kagglehub

# Download latest version
path = kagglehub.dataset_download("khsamaha/aviation-accident-datab
```

In [16]:
```
# Find the CSV file inside the downloaded directory
csv_files = [f for f in os.listdir(path) if f.endswith('.csv')]
csv_path = os.path.join(path, csv_files[csv_files.index('AviationDa
df = pd.read_csv(csv_path, encoding='latin1')
df.head(5)
```

```
/var/folders/hk/9q5rgqm970gcnk6z0xvy1p7m0000gn/T/ipykernel_93062/3
256873801.py:4: DtypeWarning: Columns (6,7,28) have mixed types. S
pecify dtype option on import or set low_memory=False.
  df = pd.read_csv(csv_path, encoding='latin1')
```

Out[16]:

| | Event.Id | Investigation.Type | Accident.Number | Event.Date | Location | Country |
|---|---|---|---|---|---|---|
| **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** | 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 [17]:
```
df.shape
```

Out[17]:
```
(88889, 31)
```

### Data Cleaning

- Drop irrelevant columns.
- Impute missing values using mode or median.
- Create new features: `Total.Injuries` and `Risk.Score` to quantify accident severity.

In [18]:
```
# List of data's columns
df.columns
```

Out[18]:
```
Index(['Event.Id', 'Investigation.Type', 'Accident.Number', 'Even
t.Date',
       'Location', 'Country', 'Latitude', 'Longitude', 'Airport.Co
de',
       'Airport.Name', 'Injury.Severity', 'Aircraft.damage',
       'Aircraft.Category', 'Registration.Number', 'Make', 'Mode
l',
       'Amateur.Built', 'Number.of.Engines', 'Engine.Type', 'FAR.D
escription',
       'Schedule', 'Purpose.of.flight', 'Air.carrier', 'Total.Fata
l.Injuries',
       'Total.Serious.Injuries', 'Total.Minor.Injuries', 'Total.Un
injured',
       'Weather.Condition', 'Broad.phase.of.flight', 'Report.Statu
s',
       'Publication.Date'],
      dtype='object')
```

In [19]:
```
# Summary statistics of numeric columns
df.describe()
```

Out[19]:

| | Number.of.Engines | Total.Fatal.Injuries | Total.Serious.Injuries | Total.Minor.Injuries | Total |
|---|---|---|---|---|---|
| **count** | 82805.000000 | 77488.000000 | 76379.000000 | 76956.000000 | 829 |
| **mean** | 1.146585 | 0.647855 | 0.279881 | 0.357061 | |
| **std** | 0.446510 | 5.485960 | 1.544084 | 2.235625 | |
| **min** | 0.000000 | 0.000000 | 0.000000 | 0.000000 | |
| **25%** | 1.000000 | 0.000000 | 0.000000 | 0.000000 | |
| **50%** | 1.000000 | 0.000000 | 0.000000 | 0.000000 | |
| **75%** | 1.000000 | 0.000000 | 0.000000 | 0.000000 | |
| **max** | 8.000000 | 349.000000 | 161.000000 | 380.000000 | 6 |

In [20]:
```
# The sum of missing values in each column
df.isna().sum()
```

Out[20]:
```
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
```

In [21]:
```
# Drop irrelevant columns
df = df.drop(columns=['Event.Id', 'Accident.Number', 'Total.Uninjur
```

In [22]:
```
# Impute missing values

df['Injury.Severity'] = df['Injury.Severity'].fillna(df['Injury.Sev
df['Aircraft.damage'] = df['Aircraft.damage'].fillna(df['Aircraft.d
df['Number.of.Engines'] = df['Number.of.Engines'].fillna(df['Number
df['Engine.Type'] = df['Engine.Type'].fillna(df['Engine.Type'].mode
df['Total.Serious.Injuries'] = df['Total.Serious.Injuries'].fillna(
df['Total.Minor.Injuries'] = df['Total.Minor.Injuries'].fillna(df['
df['Total.Fatal.Injuries'] = df['Total.Fatal.Injuries'].fillna(df['
df['Weather.Condition'] = df['Weather.Condition'].fillna('Unknown')
df['Broad.phase.of.flight'] = df['Broad.phase.of.flight'].fillna('U
df['Make'] = df['Make'].fillna('Unknown')
```

In [23]:
```
# Create new columns
df['Total.Injuries'] = df['Total.Fatal.Injuries'] + df['Total.Serio
df['Risk.Score'] = (df['Total.Fatal.Injuries'] * 3) + (df['Total.Se
```

In [24]:
```
# Filter data for commercial aviation focus
df_clean = df[df['Aircraft.Category'] == 'Airplane']
```

In [25]:
```
# Drop rows with missing critical data
df_clean = df_clean.dropna(subset=['Injury.Severity', 'Aircraft.dam

# Convert 'Event.Date' to datetime
df_clean['Event.Date'] = pd.to_datetime(df_clean['Event.Date'])

# Check cleaned dataset
print("Cleaned Dataset Shape:", df_clean.shape)
```

```
Cleaned Dataset Shape: (27586, 16)
```

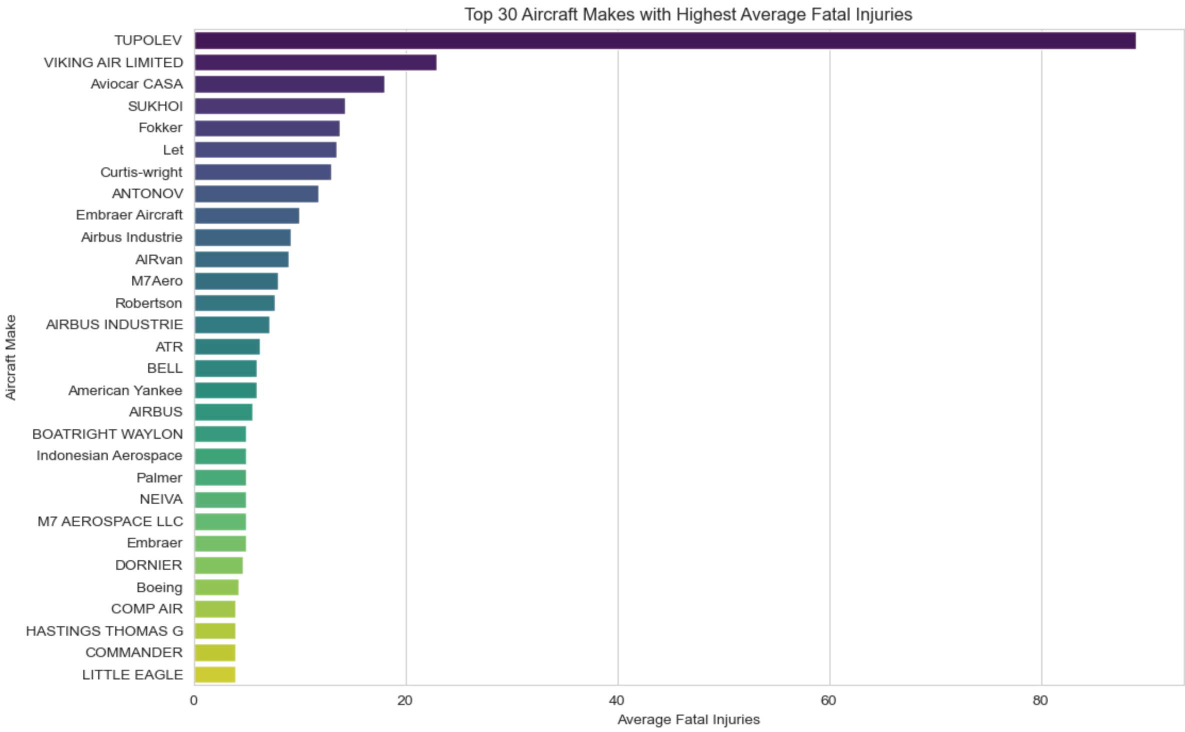
## Data Analysis and Visualization

#### Injury Severity by Aircraft Make

We analyze which aircraft manufacturers have the highest average fatal injuries.

In [26]:
```
import matplotlib.pyplot as plt
import seaborn as sns
# Group by 'Make' and calculate average injuries
injury_by_make = df_clean.groupby('Make').agg({
    'Total.Fatal.Injuries': 'mean',
    'Total.Serious.Injuries': 'mean',
    'Total.Minor.Injuries': 'mean'
}).reset_index()

# Sort by 'Total.Fatal.Injuries'
injury_by_make = injury_by_make.sort_values(by='Total.Fatal.Injurie

# Plot
plt.figure(figsize=(12, 8))
sns.barplot(data=injury_by_make.head(30), x='Total.Fatal.Injuries',
plt.title('Top 30 Aircraft Makes with Highest Average Fatal Injurie
plt.xlabel('Average Fatal Injuries')
plt.ylabel('Aircraft Make')
plt.show()
```

```
/var/folders/hk/9q5rgqm970gcnk6z0xvy1p7m0000gn/T/ipykernel_93062/3
026956304.py:15: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will b
e removed in v0.14.0. Assign the `y` variable to `hue` and set `le
gend=False` for the same effect.

  sns.barplot(data=injury_by_make.head(30), x='Total.Fatal.Injurie
s', y='Make', palette='viridis')
```

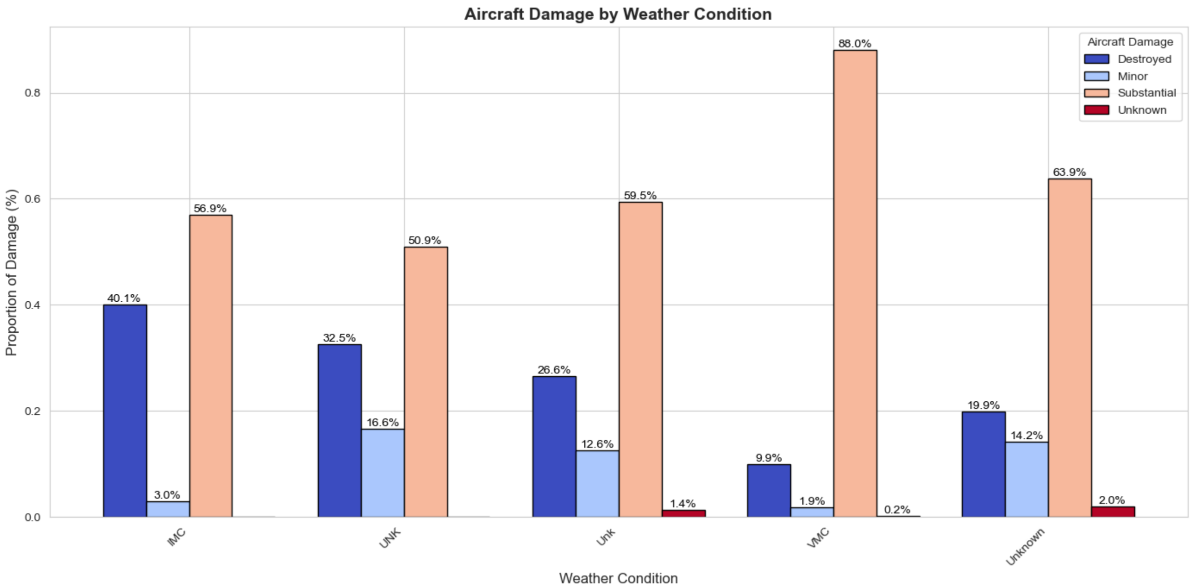


**Insight:** Manufacturers like **Tupolev, Viking Air Limited, and Aviocar CASA** have the highest fatal injury averages.

#### Aircraft Damage by Weather Condition

We examine how different weather conditions impact aircraft damage.

In [27]:
```
# Group by 'Weather.Condition' and calculate damage frequency
damage_by_weather = df_clean.groupby('Weather.Condition')['Aircraft


# Sort by total occurrences
damage_by_weather = damage_by_weather.reindex(damage_by_weather.sum

# Set Seaborn style
sns.set_style("whitegrid")

# Create bar plot (grouped bars)
ax = damage_by_weather.plot(kind='bar', figsize=(14, 7), colormap='

# Add labels on top of bars
for p in ax.patches:
    if p.get_height() > 0:
        ax.annotate(f'{p.get_height()*100:.1f}%', (p.get_x() + p.ge
                    ha='center', va='bottom', fontsize=10, color='b

# Titles and labels
plt.title('Aircraft Damage by Weather Condition', fontsize=14, font
plt.xlabel('Weather Condition', fontsize=12)
plt.ylabel('Proportion of Damage (%)', fontsize=12)
plt.xticks(rotation=45, ha='right', fontsize=10)
plt.legend(title='Aircraft Damage', fontsize=10)
plt.tight_layout()

# Show plot
plt.show()
```



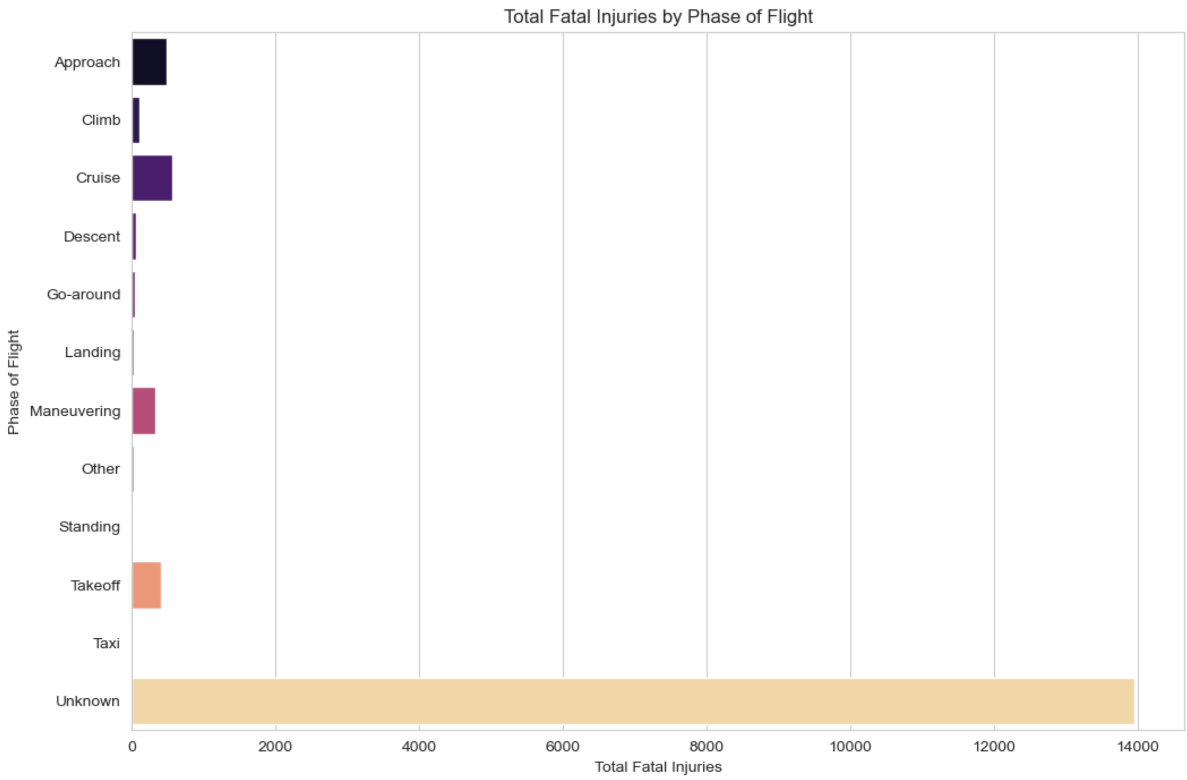

**Insight:** Accidents in **Instrument Meteorological Conditions (IMC)** result in higher aircraft damage.

### Fatal Injuries by Phase of Flight

We identify the most dangerous flight phases.

In [28]:
```
# Group by 'Broad.phase.of.flight' and calculate total fatal injuri
fatal_by_phase = df_clean.groupby('Broad.phase.of.flight')['Total.F

# Plot
plt.figure(figsize=(12, 8))
sns.barplot(data=fatal_by_phase, x='Total.Fatal.Injuries', y='Broad
plt.title('Total Fatal Injuries by Phase of Flight')
plt.xlabel('Total Fatal Injuries')
plt.ylabel('Phase of Flight')
plt.show()
```

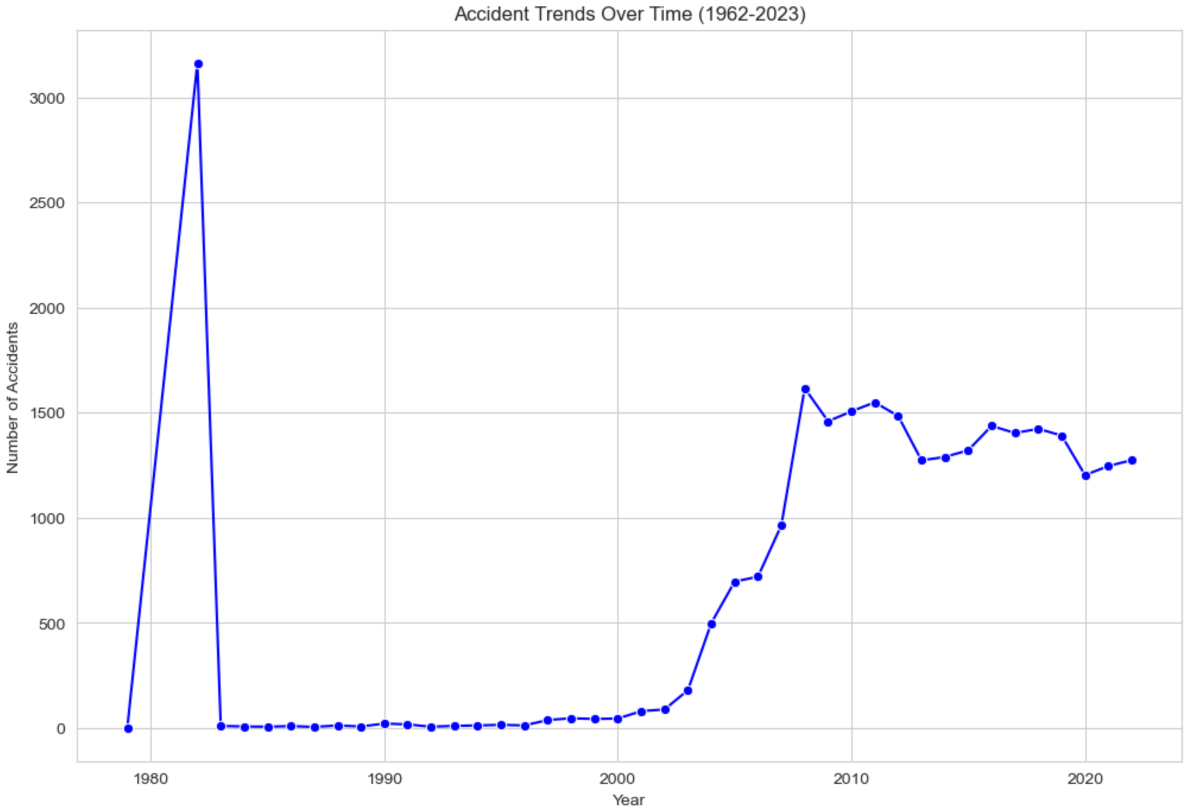
```
/var/folders/hk/9q5rgqm970gcnk6z0xvy1p7m0000gn/T/ipykernel_93062/2
248461279.py:6: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will b
e removed in v0.14.0. Assign the `y` variable to `hue` and set `le
gend=False` for the same effect.

  sns.barplot(data=fatal_by_phase, x='Total.Fatal.Injuries', y='Br
oad.phase.of.flight', palette='magma')
```




**Insight:** Takeoff and landing phases have the highest fatal injuries

# Accident Trends Over Time

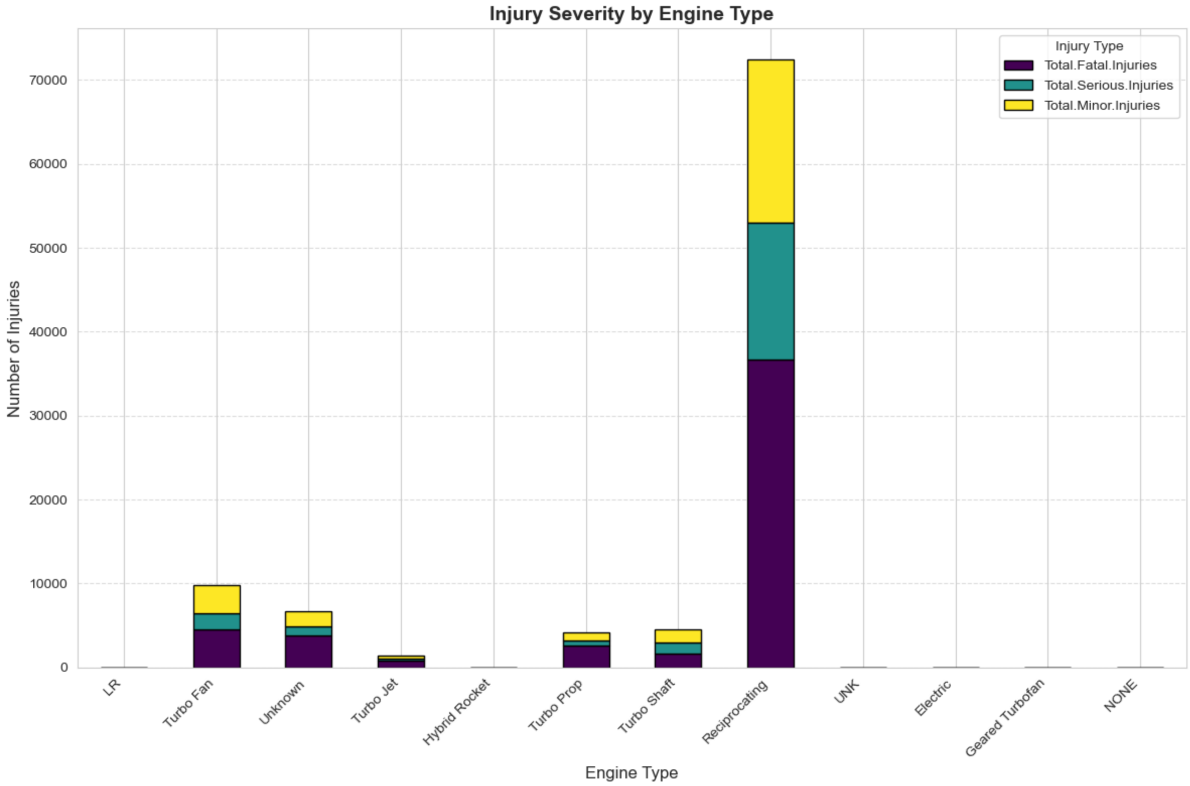
In [29]:
```
# Extract year from 'Event.Date'
df_clean['Year'] = df_clean['Event.Date'].dt.year

# Group by year and count accidents
accidents_by_year = df_clean.groupby('Year').size().reset_index(nam

# Plot
plt.figure(figsize=(12, 8))
sns.lineplot(data=accidents_by_year, x='Year', y='Accident.Count', 
plt.title('Accident Trends Over Time (1962-2023)')
plt.xlabel('Year')
plt.ylabel('Number of Accidents')
plt.show()
```




### Engine Type vs. Fatal Injuries

We compare the engine type with the injury severity

In [30]:
```
# Analyze Engine Type and Accident Severity
engine_analysis = df.groupby('Engine.Type').agg({
    'Total.Fatal.Injuries': 'sum',
    'Total.Serious.Injuries': 'sum',
    'Total.Minor.Injuries': 'sum',
    'Risk.Score': 'mean'
}).reset_index()

# Sort by Risk Score
engine_analysis = engine_analysis.sort_values(by='Risk.Score', ascen

# Set index for stacking
engine_analysis.set_index('Engine.Type', inplace=True)

# Plot stacked bar chart
engine_analysis[['Total.Fatal.Injuries', 'Total.Serious.Injuries',
    kind='bar', stacked=True, figsize=(12, 8), colormap='viridis', 
)

# Titles and labels
plt.title('Injury Severity by Engine Type', fontsize=14, fontweight
plt.xlabel('Engine Type', fontsize=12)
plt.ylabel('Number of Injuries', fontsize=12)
plt.xticks(rotation=45, ha='right', fontsize=10)
plt.legend(title='Injury Type', fontsize=10)
plt.grid(axis='y', linestyle='--', alpha=0.7)

plt.tight_layout()
plt.show()
```



## Risk Score by Engine Type

We generate the risk score for each engine type which informs our recommendations

In [31]:
```
import plotly.express as px

fig = px.bar(engine_analysis, x=engine_analysis.index, y='Risk.Scor
             labels={'Risk.Score': 'Risk Score', 'Engine.Type': 'En
             color='Risk.Score', height=500)
fig.show()
```

In [32]:
```
# %%
# Save cleaned data
df.to_csv('cleaned_data.csv', index=False)
```

## Recommendations

# Based on the analysis, here are three actionable recommendations:

1. **Avoid Purchasing aircraft models with the highest Fatal injuries** (e.g., Tupolev, Viking Air Limited and Aviocar CASA). Also focus on Engines with lower risk score as they have the fewest accidents and lowest injury rates. Such as Geared Tubofan engines and electric engine types.
2. **Prioritize aircraft with Geared turbofan engines** over reciprocating engines. Geared Turbofan engines have lower accident severity compared to reciprocating engines.
3. **Avoid operating aircraft in adverse weather conditions** (e.g., IMC). Accidents in poor weather are more likely to result in fatalities.
4. **Avoid High-Risk Phases of Flight**: Accidents during Takeoff and Landing phases result in higher fatal injuries. Implement additional safety measures during these phases.

## Conclusion

This analysis provides valuable insights into aviation safety, helping the company make informed decisions as it enters the aviation industry. The recommendations focus on minimizing risks and ensuring the safety of operations.