Aircraft Accident Data Analysis

Project Overview

This project analyzes aircraft accident data from the National Transportation Safety Board (NTSB) dataset. The goal is to clean, process, and visualize accident trends to derive insights that can help in risk assessment and safety improvements.

Key Objectives

1. **Data Cleaning & Preprocessing** – Handling missing values, duplicates, and inconsistencies.
2. **Exploratory Data Analysis (EDA)** – Identifying trends, distributions, and relationships in the dataset.
3. **Data Visualization** – Creating informative charts to communicate insights.
4. **Business Recommendations** – Providing actionable insights to improve aviation safety.

This notebook will take you through the entire data analysis process, from raw data exploration to final insights.

Loading and Exploring the Dataset

This function **`load_data_information`** performs the following tasks:

- Reads the dataset from a CSV file.
- Displays the **first 3 rows** to preview the data.
- Prints the **shape** (number of rows and columns).
- Provides an **overview of the dataset's structure** using `info()`.
  The dataset is loaded using `low_memory=False` to handle large datasets efficiently.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

def load_data_information (file_path):

    df= pd.read_csv(file_path, low_memory=False)

    print('\n---Fist 3 rows---')
    print(df.head(3))
    print('\n---df shape---')
    print(df.shape)
    print('\n---df information--')
    print(df.info())
```

```
    print('/n --Coluns of Dataset')
    print(list(df.columns))

    return df
file_path="./AviationData_utf8.csv"
df=load_data_information(file_path)


---Fist 3 rows---
         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

          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


  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


  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

  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 rows x 31 columns]

---df shape---
(88889, 31)
```

```
---df information--
<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
None
/n --Coluns of Dataset
['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']
```

Checking for Duplicates and Missing Values

This function **`check_duplicates_missing`**:

- Identifies duplicate `Event.Id` values.
- Counts and displays missing values for each column.
- Helps in assessing data quality before cleaning.

```
#checks for duplicates and missing values
def check_duplicates_missing(df):
    print('Duplicated')
    print(df['Event.Id'].duplicated().value_counts())
    print('/n missing values')
    print(df.isna().sum())

    return(df)
check_duplicates_missing(df)
```

```
Duplicated
Event.Id
False    87951
True       938
Name: count, dtype: int64
/n missing values
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

             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]
```

Converting Data Types

- **Converted `Event.Date` & `Publication.Date`** → `datetime` format.

- **Converted `Latitude` & `Longitude`** → `float`, handling errors gracefully.

```
#convert date to date time
df['Event.Date'] = pd.to_datetime(df['Event.Date'], errors='coerce')
df['Publication.Date'] = pd.to_datetime(df['Publication.Date'],
errors='coerce')

# Convert 'Latitude' and 'Longitude' to float, handling errors
df['Latitude'] = pd.to_numeric(df['Latitude'], errors='coerce')
df['Longitude'] = pd.to_numeric(df['Longitude'], errors='coerce')
print(df.dtypes[['Event.Date', 'Publication.Date', 'Latitude',
'Longitude']])
print(df[['Event.Date', 'Publication.Date', 'Latitude',
'Longitude']].head(3))

Event.Date          datetime64[ns]
Publication.Date    datetime64[ns]
Latitude                   float64
Longitude                  float64
dtype: object
```

```
  Event.Date Publication.Date   Latitude  Longitude
0 1948-10-24              NaT        NaN        NaN
1 1962-07-19       1996-09-19        NaN        NaN
2 1974-08-30       2007-02-26  36.922223 -81.878056

C:\Users\hezronkatila\AppData\Local\Temp\
ipykernel_13492\1380211228.py:3: UserWarning: Parsing dates in %d-%m-
%Y format when dayfirst=False (the default) was specified. Pass
`dayfirst=True` or specify a format to silence this warning.
  df['Publication.Date'] = pd.to_datetime(df['Publication.Date'],
errors='coerce')
```

Checking Unique Values in Categorical Columns

understanding the distribution of categorical variables.
This step helps in identifying:

- **Possible inconsistencies** (e.g., different spellings of the same category).
- **Low-frequency categories** that might need to be grouped.
- **Unexpected placeholders** that might exist.

```
# check unique values of categorical columns
cat_cols = df.select_dtypes(exclude=['number', 'datetime']).columns

for col in cat_cols:
    print(f"\nUnique values in {col}:")
    print(df[col].value_counts())


Unique values in Event.Id:
Event.Id
20001214X45071    3
20001212X19172    3
20001214X45064    2
20001212X17570    2
20001214X37556    2
                 ..
20221213106451    1
20221213106455    1
20221215106461    1
20221220106480    1
20020909X01560    1
Name: count, Length: 87951, dtype: int64

Unique values in Investigation.Type:
Investigation.Type
Accident    85015
Incident     3874
Name: count, dtype: int64
```

```
Unique values in Accident.Number:
Accident.Number
ERA22LA103    2
DCA22WA089    2
DCA22WA167    2
ERA22LA119    2
CEN22LA149    2
              ..
ERA23FA087    1
WPR23LA065    1
ERA23LA089    1
WPR23LA072    1
MIA82DA029    1
Name: count, Length: 88863, dtype: int64

Unique values in Location:
Location
ANCHORAGE, AK        434
MIAMI, FL            200
ALBUQUERQUE, NM      196
HOUSTON, TX          193
CHICAGO, IL          184
                    ...
Abu Dhabi,             1
Medellin,              1
Saint Joseph, MO       1
Ronselarestraat,       1
Oxfordshire,           1
Name: count, Length: 27758, dtype: int64

Unique values in Country:
Country
United States               82248
Brazil                        374
Canada                        359
Mexico                        358
United Kingdom                344
                             ...
Mauritania                      1
Pacific Ocean                   1
Obyan                           1
Guernsey                        1
Turks and Caicos Islands        1
Name: count, Length: 219, dtype: int64

Unique values in Airport.Code:
Airport.Code
NONE    1488
PVT      485
APA      160
```

```
ORD      149
MRI      137
        ...
RGA        1
RCV        1
D51        1
THU        1
SKBS       1
Name: count, Length: 10374, dtype: int64

Unique values in Airport.Name:
Airport.Name
Private                          240
PRIVATE                          224
Private Airstrip                 153
NONE                             146
PRIVATE STRIP                    111
                                ...
Moron Airport                      1
SEAMANS FLD                        1
EVANSTON-UINTA COUNTY BURNS FL     1
WILLIAMSBURG RGNL                  1
ESTANCIA MUNICIPAL                 1
Name: count, Length: 24870, dtype: int64

Unique values in Injury.Severity:
Injury.Severity
Non-Fatal     67357
Fatal(1)       6167
Fatal          5262
Fatal(2)       3711
Incident       2219
              ...
Fatal(96)         1
Fatal(89)         1
Fatal(199)        1
Fatal(114)        1
Fatal(57)         1
Name: count, Length: 109, dtype: int64

Unique values in Aircraft.damage:
Aircraft.damage
Substantial    64148
Destroyed      18623
Minor           2805
Unknown          119
Name: count, dtype: int64

Unique values in Aircraft.Category:
Aircraft.Category
```

```
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
Rocket                    1
ULTR                      1
Name: count, dtype: int64

Unique values in Registration.Number:
Registration.Number
NONE      344
UNREG     126
UNK        13
USAF        9
N20752      8
          ...
N231GZ      1
N415RX      1
N74586      1
C-GZPU      1
N498DS      1
Name: count, Length: 79104, dtype: int64

Unique values in Make:
Make
Cessna               22227
Piper                12029
CESSNA                4922
Beech                 4330
PIPER                 2841
                     ...
SCOTT TERRY G            1
JAMES R DERNOVSEK        1
ORLICAN S R O            1
ROYSE RALPH L            1
RHINEHART                1
Name: count, Length: 8237, dtype: int64

Unique values in Model:
Model
152                  2367
```

```
172                   1756
172N                  1164
PA-28-140              932
150                    829
                      ...
A22LS                    1
SPORTSMAN GS2            1
ROARING EAGLE            1
ZENITH CH-750            1
ZENAIR STOLCH 750        1
Name: count, Length: 12318, dtype: int64

Unique values in Amateur.Built:
Amateur.Built
No     80312
Yes     8475
Name: count, dtype: int64

Unique values in Engine.Type:
Engine.Type
Reciprocating      69530
Turbo Shaft         3609
Turbo Prop          3391
Turbo Fan           2481
Unknown             2051
Turbo Jet            703
Geared Turbofan       12
Electric              10
NONE                   2
LR                     2
Hybrid Rocket          1
UNK                    1
Name: count, dtype: int64

Unique values in FAR.Description:
FAR.Description
091                                18221
Part 91: General Aviation           6486
NUSN                                1584
NUSC                                1013
137                                 1010
135                                  746
121                                  679
Part 137: Agricultural               437
UNK                                  371
Part 135: Air Taxi & Commuter        298
PUBU                                 253
129                                  246
Part 121: Air Carrier                165
133                                  107
```

```
Part 129: Foreign                    100
Non-U.S., Non-Commercial              97
Non-U.S., Commercial                  93
Part 133: Rotorcraft Ext. Load        32
Unknown                               22
Public Use                            19
091K                                  14
ARMF                                   8
Part 125: 20+ Pax,6000+ lbs            5
125                                    5
107                                    4
103                                    2
Public Aircraft                        2
Part 91 Subpart K: Fractional          1
Armed Forces                           1
Part 91F: Special Flt Ops.             1
437                                    1
Name: count, dtype: int64

Unique values in Schedule:
Schedule
NSCH    4474
UNK     4099
SCHD    4009
Name: count, dtype: int64

Unique values in Purpose.of.flight:
Purpose.of.flight
Personal                     49448
Instructional                10601
Unknown                       6802
Aerial Application            4712
Business                      4018
Positioning                   1646
Other Work Use                1264
Ferry                          812
Aerial Observation             794
Public Aircraft                720
Executive/corporate            553
Flight Test                    405
Skydiving                      182
External Load                  123
Public Aircraft - Federal      105
Banner Tow                     101
Air Race show                   99
Public Aircraft - Local         74
Public Aircraft - State         64
Air Race/show                   59
Glider Tow                      53
```

```
Firefighting                    40
Air Drop                        11
ASHO                             6
PUBS                             4
PUBL                             1
Name: count, dtype: int64

Unique values in Air.carrier:
Air.carrier
Pilot                          258
American Airlines               90
United Airlines                 89
Delta Air Lines                 53
SOUTHWEST AIRLINES CO           42
                              ...
AERY AVIATION                    1
Creamer Pilot Services LLC       1
James Tapp                       1
MFC CORP                         1
Island Airlines Hawaii Inc.      1
Name: count, Length: 13590, dtype: int64

Unique values in Weather.Condition:
Weather.Condition
VMC    77303
IMC     5976
UNK      856
Unk      262
Name: count, dtype: int64

Unique values in Broad.phase.of.flight:
Broad.phase.of.flight
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: count, dtype: int64

Unique values in Report.Status:
Report.Status
Probable Cause
61754
```

```
Foreign
1999
<br /><br />
167
Factual
145
The pilot's failure to maintain directional control during the landing
roll.
56

...
The left wing skin "fabric patch" separated from the wing ribs and
spars for an undetermined reason.
1
A deer darting into the path of the airplane during the landing roll.
1
A loss of engine power due to inadequate maintenance inspection
resulting in a worn throttle housing going undetected and failing.
1
The pilot's inadequate compensation for wind conditions during the
landing roll.  A factor was the gusting crosswind.
1
The pilot's failure to maintain directional control during landing.
Contributing to the accident was the pilot's selection of an
unsuitable landing area.        1
Name: count, Length: 17074, dtype: int64
```

Replacing Placeholder Values with NaN

Some columns contain placeholder values like `'Unknown'`, `'UNK'`, `'Unk'`, `'NONE'`, and `'Unavailable'` instead of missing values.

- These placeholders can interfere with analysis.
- So i replaced them with `NaN` to properly handle missing values.

```
# replca place holders with nall
placeholders = ['Unknown', 'UNK', 'Unk', 'NONE', 'Unavailable']
df.replace(placeholders, np.nan, inplace=True)
```

Standardizing Categorical Column Values

To ensure consistency in categorical data, all values were converted to lowercase. This prevents duplicates caused by case differences (e.g., "Piper" and "PIPER").

Steps Taken:

1. Identified all categorical columns (excluding numerical and datetime types).
2. Converted all string values in these columns to lowercase.

3. Rechecked unique values to confirm standardization.

This helps maintain data uniformity and improves accuracy in analysis.

```
# Convert all categorical columns to lowercase to standardize values
uper_cols = df.select_dtypes(exclude=['number', 'datetime']).columns
df[uper_cols] = df[uper_cols].apply(lambda x: x.str.lower())

# Check unique values again after standardization
for col in uper_cols:
    print(f"\nUnique values in {col}:")
    print(df[col].value_counts())
```

```
Unique values in Event.Id:
Event.Id
20001214x45071    3
20001212x19172    3
20001214x45064    2
20001212x17570    2
20001214x37556    2
                 ..
20221213106451    1
20221213106455    1
20221215106461    1
20221220106480    1
20020909x01560    1
Name: count, Length: 87951, dtype: int64

Unique values in Investigation.Type:
Investigation.Type
accident    85015
incident     3874
Name: count, dtype: int64

Unique values in Accident.Number:
Accident.Number
era22la103    2
dca22wa089    2
dca22wa167    2
era22la119    2
cen22la149    2
              ..
era23fa087    1
wpr23la065    1
era23la089    1
wpr23la072    1
mia82da029    1
Name: count, Length: 88863, dtype: int64

Unique values in Location:
```

```
Location
anchorage, ak      548
miami, fl          275
houston, tx        271
albuquerque, nm    265
chicago, il        256
                  ...
star valley, az      1
bignell, ne          1
winnipeg,            1
amman,               1
aguilar, co          1
Name: count, Length: 21978, dtype: int64

Unique values in Country:
Country
united states     82248
brazil              374
canada              359
mexico              358
united kingdom      344
                  ...
mauritania            1
obyan                 1
wolseley              1
albania               1
guernsey              1
Name: count, Length: 214, dtype: int64

Unique values in Airport.Code:
Airport.Code
pvt     497
apa     160
ord     149
mri     137
den     115
       ...
xwa       1
rga       1
rcv       1
d51       1
edma      1
Name: count, Length: 10345, dtype: int64

Unique values in Airport.Name:
Airport.Name
private                          471
private airstrip                 266
private strip                    161
merrill field                    109
```

```
centennial                          102
                                   ...
williamsburg rgnl                     1
addington fld                         1
dead cow lakebed airstrip us-0        1
dallas executive airport              1
jose celestino mutis                  1
Name: count, Length: 21564, dtype: int64

Unique values in Injury.Severity:
Injury.Severity
non-fatal     67357
fatal(1)       6167
fatal          5262
fatal(2)       3711
incident       2219
               ...
fatal(96)         1
fatal(89)         1
fatal(199)        1
fatal(114)        1
fatal(57)         1
Name: count, Length: 108, dtype: int64

Unique values in Aircraft.damage:
Aircraft.damage
substantial    64148
destroyed      18623
minor           2805
Name: count, dtype: int64

Unique values in Aircraft.Category:
Aircraft.Category
airplane             27617
helicopter            3440
glider                 508
balloon                231
gyrocraft              173
weight-shift           161
powered parachute       91
ultralight              30
wsft                     9
powered-lift             5
blimp                    4
rocket                   1
ultr                     1
Name: count, dtype: int64

Unique values in Registration.Number:
Registration.Number
```

```
unreg      131
usaf         9
n20752       8
unknown      7
n11vh        6
           ...
n231gz       1
n415rx       1
n74586       1
c-gzpu       1
nc6404       1
Name: count, Length: 79093, dtype: int64

Unique values in Make:
Make
cessna               27149
piper                14870
beech                 5372
boeing                2745
bell                  2722
                     ...
phantom                  1
greg hobbs               1
james r dernovsek        1
orlican s r o            1
means rober c            1
Name: count, Length: 7587, dtype: int64

Unique values in Model:
Model
152             2367
172             1756
172n            1164
pa-28-140        932
150              829
                ...
c-a185f            1
b-4 pc 11a         1
adventura ii       1
j3f-60             1
a22ls              1
Name: count, Length: 11646, dtype: int64

Unique values in Amateur.Built:
Amateur.Built
no     80312
yes     8475
Name: count, dtype: int64

Unique values in Engine.Type:
```

```
Engine.Type
reciprocating      69530
turbo shaft         3609
turbo prop          3391
turbo fan           2481
turbo jet            703
geared turbofan       12
electric              10
lr                     2
hybrid rocket          1
Name: count, dtype: int64

Unique values in FAR.Description:
FAR.Description
091                                18221
part 91: general aviation           6486
nusn                                1584
nusc                                1013
137                                 1010
135                                  746
121                                  679
part 137: agricultural               437
part 135: air taxi & commuter        298
pubu                                 253
129                                  246
part 121: air carrier                165
133                                  107
part 129: foreign                    100
non-u.s., non-commercial              97
non-u.s., commercial                  93
part 133: rotorcraft ext. load        32
public use                            19
091k                                  14
armf                                   8
part 125: 20+ pax,6000+ lbs            5
125                                    5
107                                    4
public aircraft                        2
103                                    2
armed forces                           1
part 91f: special flt ops.             1
part 91 subpart k: fractional          1
437                                    1
Name: count, dtype: int64

Unique values in Schedule:
Schedule
nsch    4474
schd    4009
```

```
Name: count, dtype: int64

Unique values in Purpose.of.flight:
Purpose.of.flight
personal                      49448
instructional                 10601
aerial application             4712
business                       4018
positioning                    1646
other work use                 1264
ferry                           812
aerial observation              794
public aircraft                 720
executive/corporate             553
flight test                     405
skydiving                       182
external load                   123
public aircraft - federal       105
banner tow                      101
air race show                    99
public aircraft - local          74
public aircraft - state          64
air race/show                    59
glider tow                       53
firefighting                     40
air drop                         11
asho                              6
pubs                              4
publ                              1
Name: count, dtype: int64

Unique values in Air.carrier:
Air.carrier
pilot                          258
american airlines               90
united airlines                 89
delta air lines                 53
delta air lines inc             44
                               ...
james tapp                        1
creamer pilot services llc        1
aery aviation                     1
reva, inc.                        1
mc cessna 210n llc                1
Name: count, Length: 13207, dtype: int64

Unique values in Weather.Condition:
Weather.Condition
vmc    77303
imc     5976
```

```
Name: count, dtype: int64

Unique values in Broad.phase.of.flight:
Broad.phase.of.flight
landing        15428
takeoff        12493
cruise         10269
maneuvering     8144
approach        6546
climb           2034
taxi            1958
descent         1887
go-around       1353
standing         945
other            119
Name: count, dtype: int64

Unique values in Report.Status:
Report.Status
probable cause
61754
foreign
1999
<br /><br />
167
factual
145
the pilot's failure to maintain directional control during the landing
roll.
56

...
the pilot's inadequate compensation for wind conditions during the
landing roll.  a factor was the gusting crosswind.
1
the pilot's failure to maintain directional control during landing.
contributing to the accident was the pilot's selection of an
unsuitable landing area.
1
a total loss of engine power due to the fatigue failure of a third
stage turbine wheel blade.
1
an inadvertent encounter with severe turbulence during descent and the
failure of two passenger seat belt attach fittings.  contributing to
the incident was the failure of the operator to comply with the saib.
1
the departing pilot's inadequate visual lookout.
1
Name: count, Length: 17061, dtype: int64
```

Handling Missing Values

To ensure data consistency, I:

1. Removed duplicate rows based on `Event.Id`.
2. Droped columns where more than 50% of values are missing.
3. Filled missing values in numeric columns using:
   - **Mean** (for normally distributed data)
   - **Median** (for skewed data)
4. Filled missing values in categorical columns with the most frequent value (mode).

```
#working with missing values

def filling_missing(df):
    df = df.drop_duplicates(subset=['Event.Id'], keep='first')

    # Droping columns with more than 50% missing values
    df = df.dropna(thresh=0.5* len(df), axis=1)

    # Filling missing values in numeric columns
    for col in df.select_dtypes(include=['number']).columns:
        if df[col].isna().sum() > 0:
            skewness = df[col].skew()
            fill_value = df[col].mean() if abs(skewness) < 0.5 else
df[col].median()
            df[col].fillna(fill_value, inplace=True)
   #  Filling missing values in categorical columns
    for col in df.select_dtypes(exclude=['number']).columns:
        if df[col].isna().sum() > 0:
            most_frequent = df[col].mode()[0] if not
df[col].mode().empty else "Unknown"
            df[col].fillna(most_frequent, inplace=True)


    return df

df = filling_missing(df)
```

```
C:\Users\hezronkatila\AppData\Local\Temp\
ipykernel_13492\559727326.py:14: FutureWarning: A value is trying to
be set on a copy of a DataFrame or Series through chained assignment
using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never
work because the intermediate object on which we are setting values
always behaves as a copy.
```

```
For example, when doing 'df[col].method(value, inplace=True)', try
using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on the
original object.


  df[col].fillna(fill_value, inplace=True)
C:\Users\hezronkatila\AppData\Local\Temp\
ipykernel_13492\559727326.py:14: FutureWarning: A value is trying to
be set on a copy of a DataFrame or Series through chained assignment
using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never
work because the intermediate object on which we are setting values
always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try
using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on the
original object.


  df[col].fillna(fill_value, inplace=True)
C:\Users\hezronkatila\AppData\Local\Temp\
ipykernel_13492\559727326.py:14: FutureWarning: A value is trying to
be set on a copy of a DataFrame or Series through chained assignment
using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never
work because the intermediate object on which we are setting values
always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try
using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on the
original object.


  df[col].fillna(fill_value, inplace=True)
C:\Users\hezronkatila\AppData\Local\Temp\
ipykernel_13492\559727326.py:14: FutureWarning: A value is trying to
be set on a copy of a DataFrame or Series through chained assignment
using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never
work because the intermediate object on which we are setting values
always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try
using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on the
original object.
```

```
  df[col].fillna(fill_value, inplace=True)
C:\Users\hezronkatila\AppData\Local\Temp\
ipykernel_13492\559727326.py:14: FutureWarning: A value is trying to
be set on a copy of a DataFrame or Series through chained assignment
using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never
work because the intermediate object on which we are setting values
always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try
using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on the
original object.


  df[col].fillna(fill_value, inplace=True)
C:\Users\hezronkatila\AppData\Local\Temp\
ipykernel_13492\559727326.py:19: FutureWarning: A value is trying to
be set on a copy of a DataFrame or Series through chained assignment
using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never
work because the intermediate object on which we are setting values
always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try
using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on the
original object.


  df[col].fillna(most_frequent, inplace=True)
C:\Users\hezronkatila\AppData\Local\Temp\
ipykernel_13492\559727326.py:19: FutureWarning: A value is trying to
be set on a copy of a DataFrame or Series through chained assignment
using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never
work because the intermediate object on which we are setting values
always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try
using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on the
original object.


  df[col].fillna(most_frequent, inplace=True)
```

Final Check: Missing Values & Duplicates

After handling missing values and removing duplicates, I verified:

- Are there still missing values in any column?
- Are there still duplicate `Event.Id` values?

```
#check if null and  duplicate wer droped
print(df.isna().sum())
print(df['Event.Id'].duplicated().value_counts())
```

```
Event.Id                    0
Investigation.Type          0
Accident.Number             0
Event.Date                  0
Location                    0
Country                     0
Airport.Code                0
Airport.Name                0
Injury.Severity             0
Aircraft.damage             0
Registration.Number         0
Make                        0
Model                       0
Amateur.Built               0
Number.of.Engines           0
Engine.Type                 0
Purpose.of.flight           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
Event.Id
False    87951
Name: count, dtype: int64
```

Outlier Detection: Injury Data

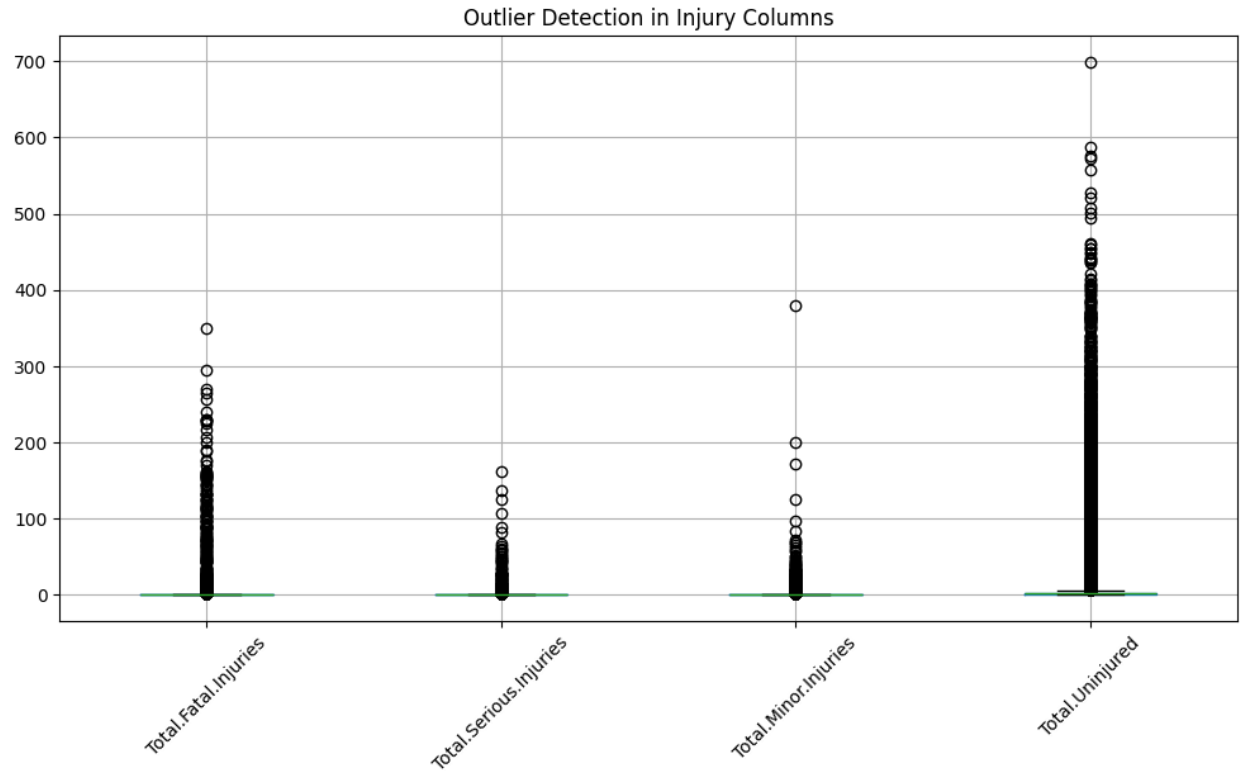
To identify potential **outliers** in injury-related columns, i used a **boxplot**.
The boxplot helps visualize:

- **Skewness** in injury distributions.
- **Extreme values** that might indicate data entry errors or significant incidents.

The columns analyzed:

- **Total.Fatal.Injuries**
- **Total.Serious.Injuries**
- **Total.Minor.Injuries**
- **Total.Uninjured**

```
# Plot boxplots for key numerical columns
num_cols = ['Total.Fatal.Injuries', 'Total.Serious.Injuries',
'Total.Minor.Injuries', 'Total.Uninjured']

plt.figure(figsize=(12, 6))
df[num_cols].boxplot()
plt.title("Outlier Detection in Injury Columns")
plt.xticks(rotation=45)
plt.show()
```



Removing Outliers Using IQR

Removing **outliers** using the **Interquartile Range (IQR) Method**.

Columns Processed:

- **Total.Fatal.Injuries**

- **Total.Serious.Injuries**
- **Total.Minor.Injuries**
- **Total.Uninjured**

```
# remove outliers using IQR
def remove_outliers(df, cols):
    for col in cols:
        Q1 = df[col].quantile(0.25)
        Q3 = df[col].quantile(0.75)
        IQR = Q3 - Q1
        lower_bound = Q1 - 1.5 * IQR
        upper_bound = Q3 + 1.5 * IQR


        df = df[(df[col] >= lower_bound) & (df[col] <= upper_bound)]

    return df


df = remove_outliers(df, num_cols)
```

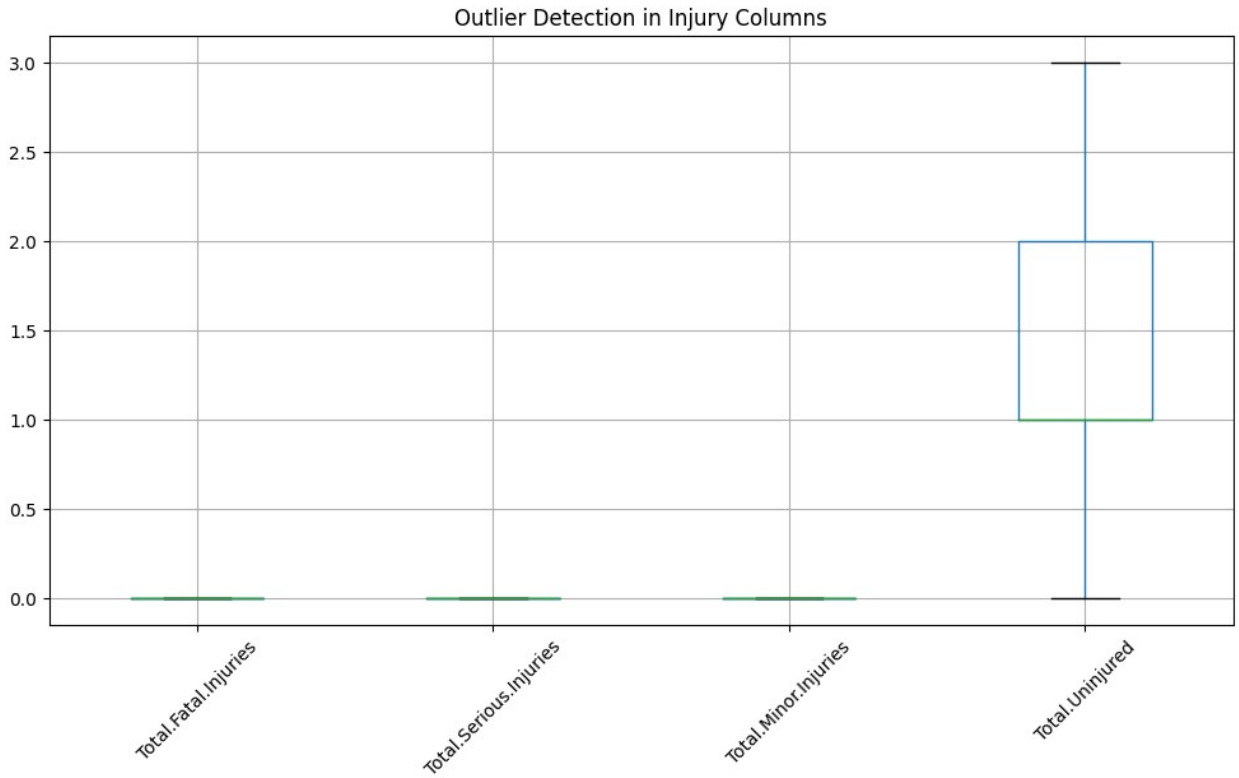
Checking if Outliers Were Successfully Removed

Verifying the changes by **plotting boxplots again**.

Expected Outcome:

- The **extreme points** (outliers) outside the whiskers should be **removed**.
- The distribution of data should now be **more compact and representative**.

```
# check ouliers have been removed
num_cols = ['Total.Fatal.Injuries', 'Total.Serious.Injuries',
'Total.Minor.Injuries', 'Total.Uninjured']

plt.figure(figsize=(12, 6))
df[num_cols].boxplot()
plt.title("Outlier Detection in Injury Columns")
plt.xticks(rotation=45)
plt.show()
```

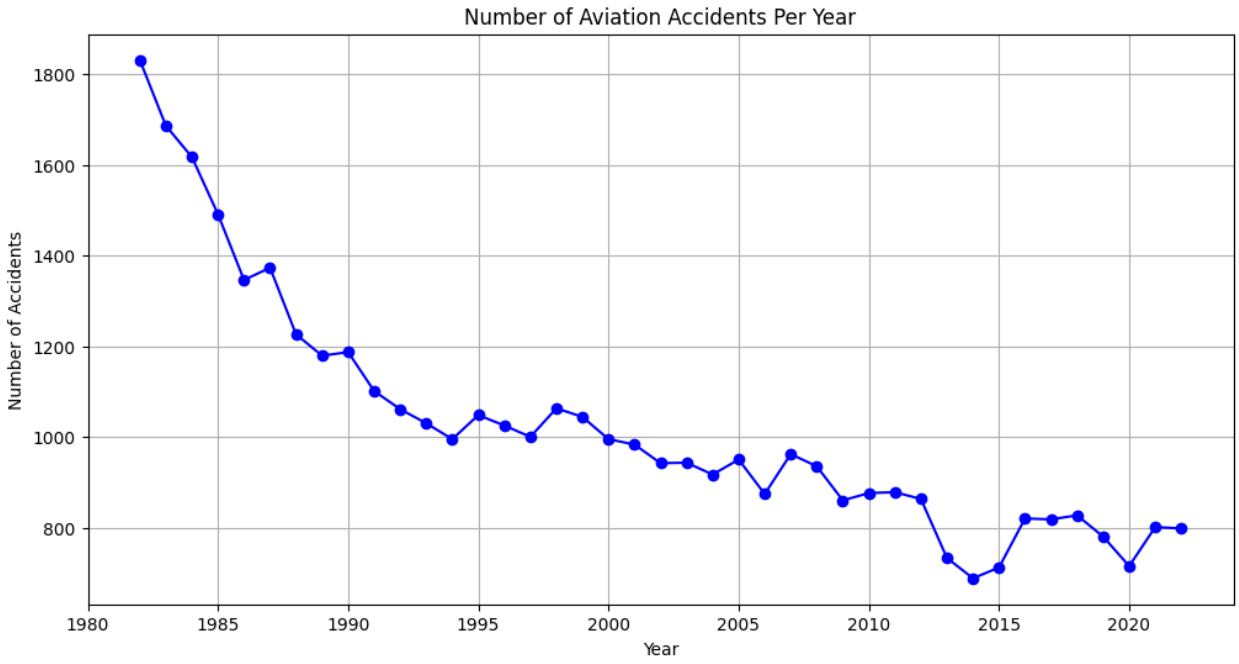


Accident Trend Over Time

The number of aviation accidents per year was analyzed to identify trends. This helped in understanding whether incidents increased or decreased over time.

Visualization:

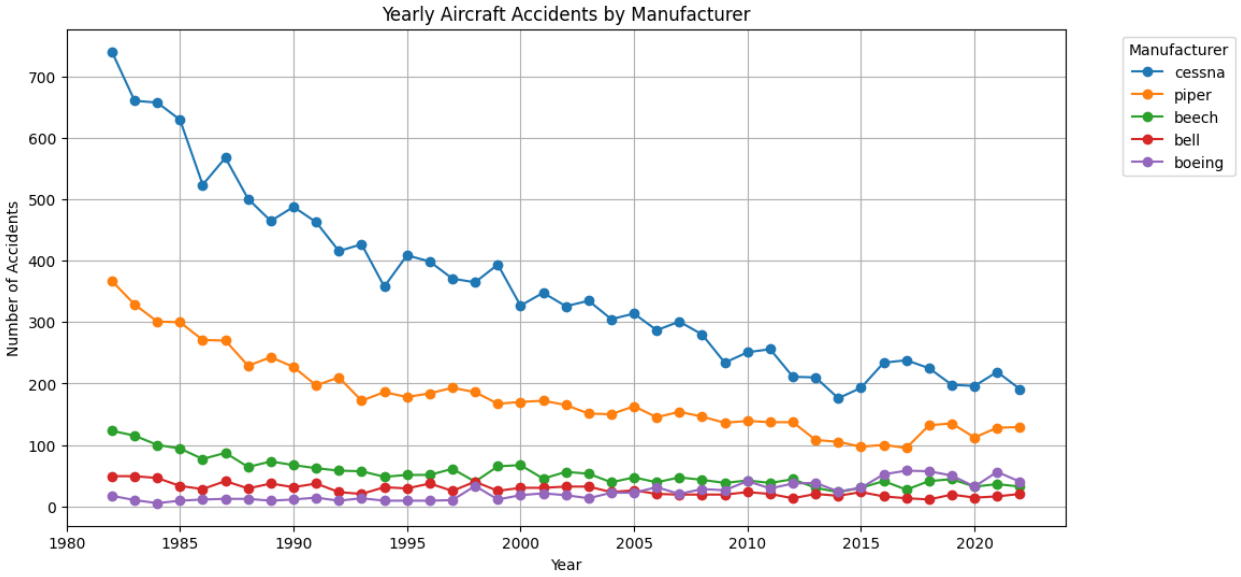
```
# Accident Trend Over Time
df['Year'] = df['Event.Date'].dt.year

plt.figure(figsize=(12, 6))
df['Year'].value_counts().sort_index().plot(kind='line', marker='o',
color='b')
plt.title("Number of Aviation Accidents Per Year")
plt.xlabel("Year")
plt.ylabel("Number of Accidents")
plt.grid(True)
plt.show()
```



Yearly Aircraft Accidents by Manufacturer

To identify trends in aviation accidents over time, I analyzed the number of incidents per year for different aircraft manufacturers. This helps in understanding which manufacturers have had the highest accident rates historically and how their trends have changed over time.

```
# Extracting the year from Event.Date
df['Event.Year'] = pd.to_datetime(df['Event.Date']).dt.year

# Counting accidents by year and manufacturer
yearly_make = df.groupby(['Event.Year',
'Make']).size().unstack().fillna(0)

# Gettig the top 5 manufacturers with the highest total accidents
top_makes_list = df['Make'].value_counts().head(5).index


yearly_make[top_makes_list].plot(figsize=(12, 6), marker='o')
plt.title("Yearly Aircraft Accidents by Manufacturer")
plt.xlabel("Year")
plt.ylabel("Number of Accidents")
plt.legend(title="Manufacturer", bbox_to_anchor=(1.05, 1), loc='upper
left')
plt.grid()
plt.show()
```

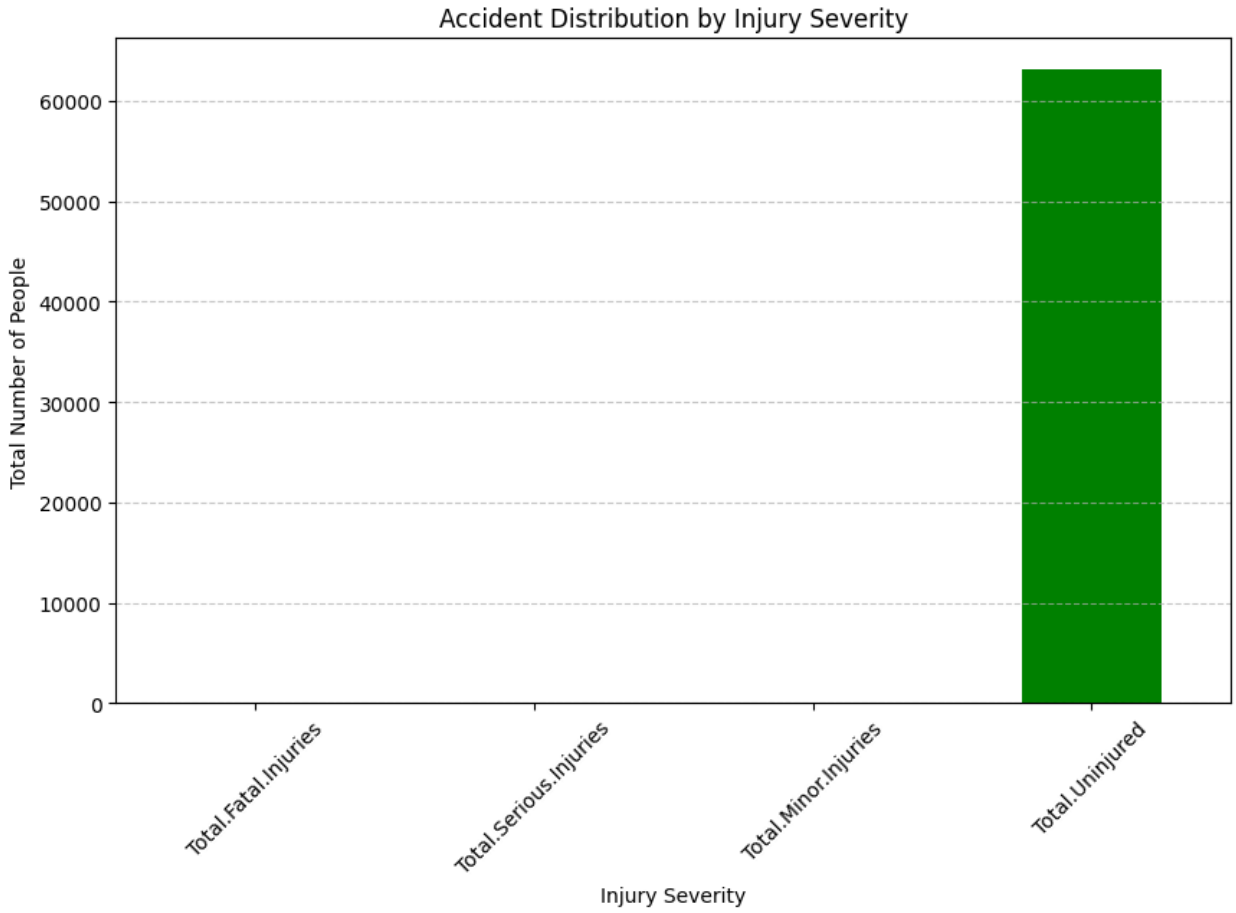


Accident Distribution by Injury Severity

The dataset was analyzed to determine the distribution of aviation accidents based on injury severity. This provided insights into how frequently accidents resulted in fatalities, serious injuries, minor injuries, or no injuries.

Visualization:

```
# Accident Distribution by Injury Severity
injury_cols = ['Total.Fatal.Injuries', 'Total.Serious.Injuries',
'Total.Minor.Injuries', 'Total.Uninjured']
injury_counts = df[injury_cols].sum()

plt.figure(figsize=(10, 6))
injury_counts.plot(kind='bar', color=['red', 'orange', 'yellow',
'green'])
plt.title("Accident Distribution by Injury Severity")
plt.xlabel("Injury Severity")
plt.ylabel("Total Number of People")
plt.xticks(rotation=45)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()
```



Accidents by Country

This visualization shows the top 10 countries with the highest number of aircraft accidents. Understanding accident distribution by country helps identify regions with the most recorded incidents, which could be influenced by factors such as air traffic density, regulatory policies, or operational conditions.

A bar chart was used to display the accident count for each country, sorted in descending order.

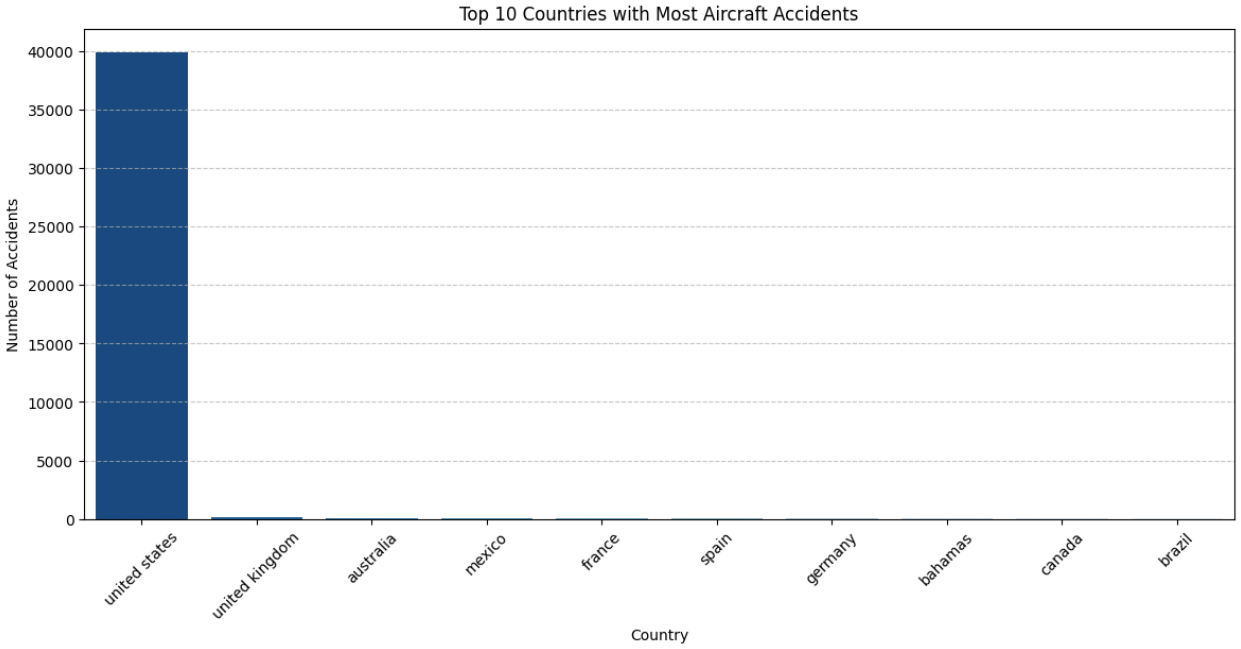
Insights:

- The countries with the highest number of aircraft accidents are prominently visible.
- This data can help aviation authorities focus on regions with higher accident occurrences for further investigation.

```
# Accidents by Country
plt.figure(figsize=(14, 6))
top_countries = df['Country'].value_counts().head(10)  # Get top 10
countries

sns.barplot(x=top_countries.index, y=top_countries.values,
palette="Blues_r")
plt.title("Top 10 Countries with Most Aircraft Accidents")
```

```
plt.xlabel("Country")
plt.ylabel("Number of Accidents")
plt.xticks(rotation=45)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()
```

```
C:\Users\hezronkatila\AppData\Local\Temp\
ipykernel_13492\326703483.py:5: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set
`legend=False` for the same effect.

  sns.barplot(x=top_countries.index, y=top_countries.values,
palette="Blues_r")
```



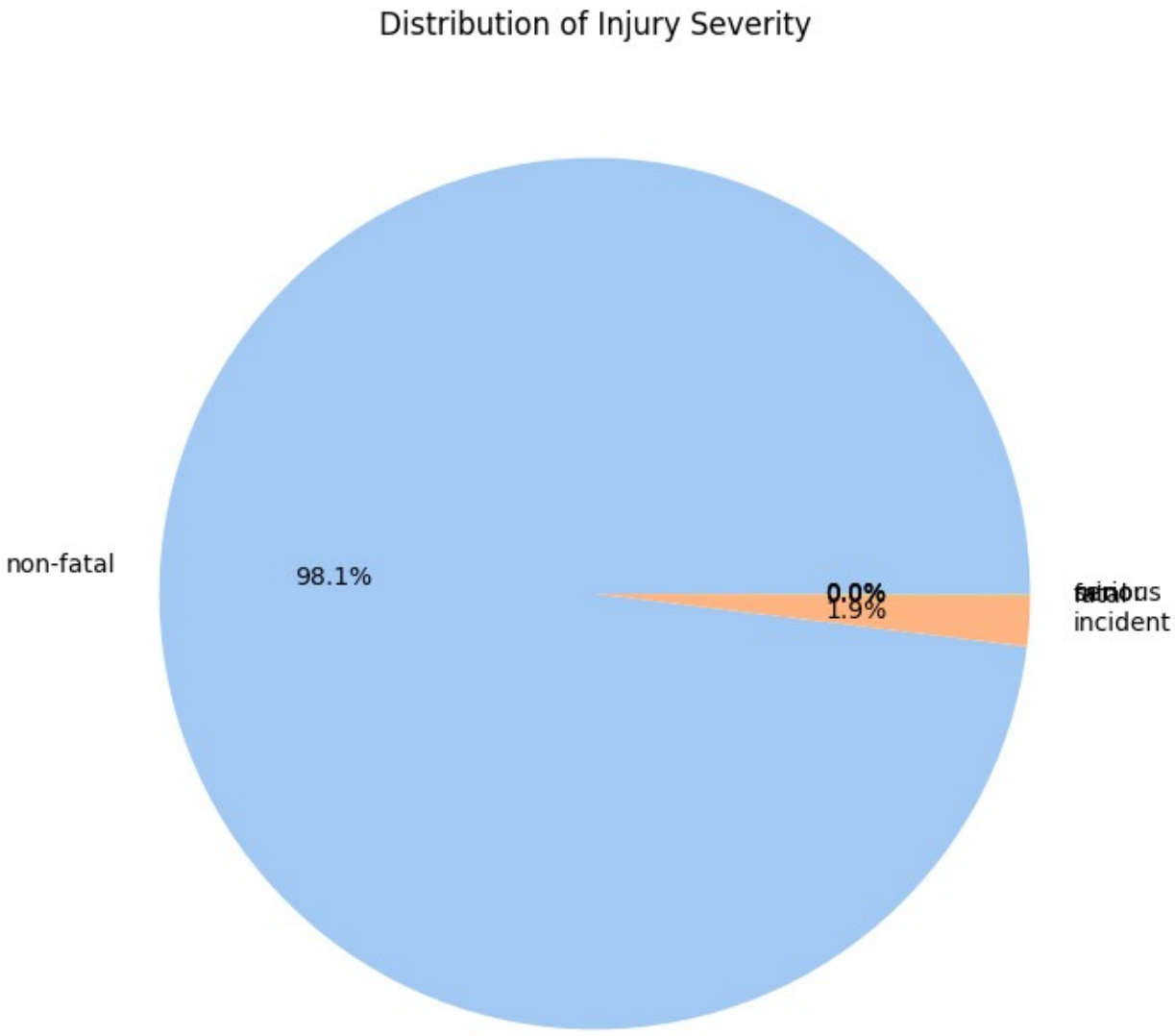
Distribution of Injury Severity

The pie chart below represents the distribution of different injury severity levels in aircraft accidents. By visualizing this data, we can understand the proportion of accidents that resulted in fatal, serious, minor, or no injuries.

- The largest section indicates the most common severity level in recorded accidents.
- This helps in identifying trends and areas that may need further investigation or safety improvements.

```
plt.figure(figsize=(8, 8))
df['Injury.Severity'].value_counts().plot(kind='pie', autopct='%1.1f%
%', colors=sns.color_palette('pastel'))
```

```
plt.title("Distribution of Injury Severity")
plt.ylabel("")  # Hide y-label
plt.show()
```



Top 10 Aircraft Manufacturers Involved in Accidents

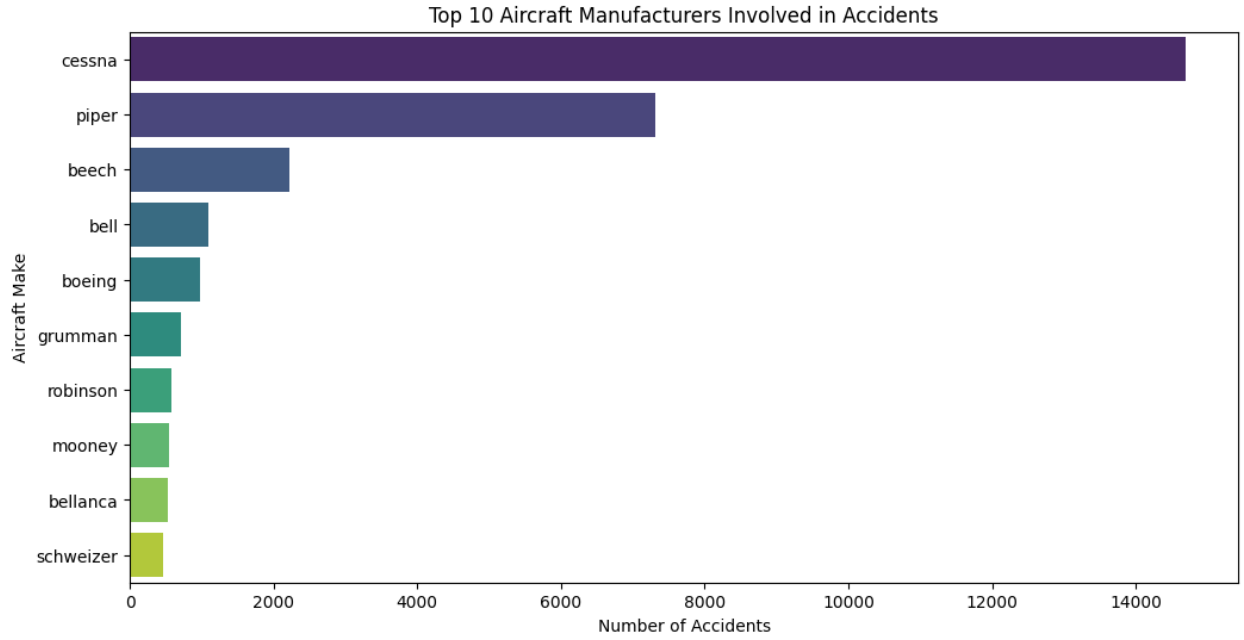
The bar chart below displays the top 10 aircraft manufacturers with the highest number of recorded accidents.

- The x-axis represents the number of accidents.
- The y-axis lists the manufacturers.

- This visualization helps identify which aircraft models have been most frequently involved in accidents, which could be due to higher production numbers, operational conditions, or other factors.

```
top_makes = df['Make'].value_counts().head(10)
plt.figure(figsize=(12, 6))
sns.barplot(x=top_makes.values, y=top_makes.index, palette='viridis')
plt.xlabel("Number of Accidents")
plt.ylabel("Aircraft Make")
plt.title("Top 10 Aircraft Manufacturers Involved in Accidents")
plt.show()
```

```
C:\Users\hezronkatila\AppData\Local\Temp\
ipykernel_13492\4081644643.py:3: FutureWarning:

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

  sns.barplot(x=top_makes.values, y=top_makes.index,
palette='viridis')
```

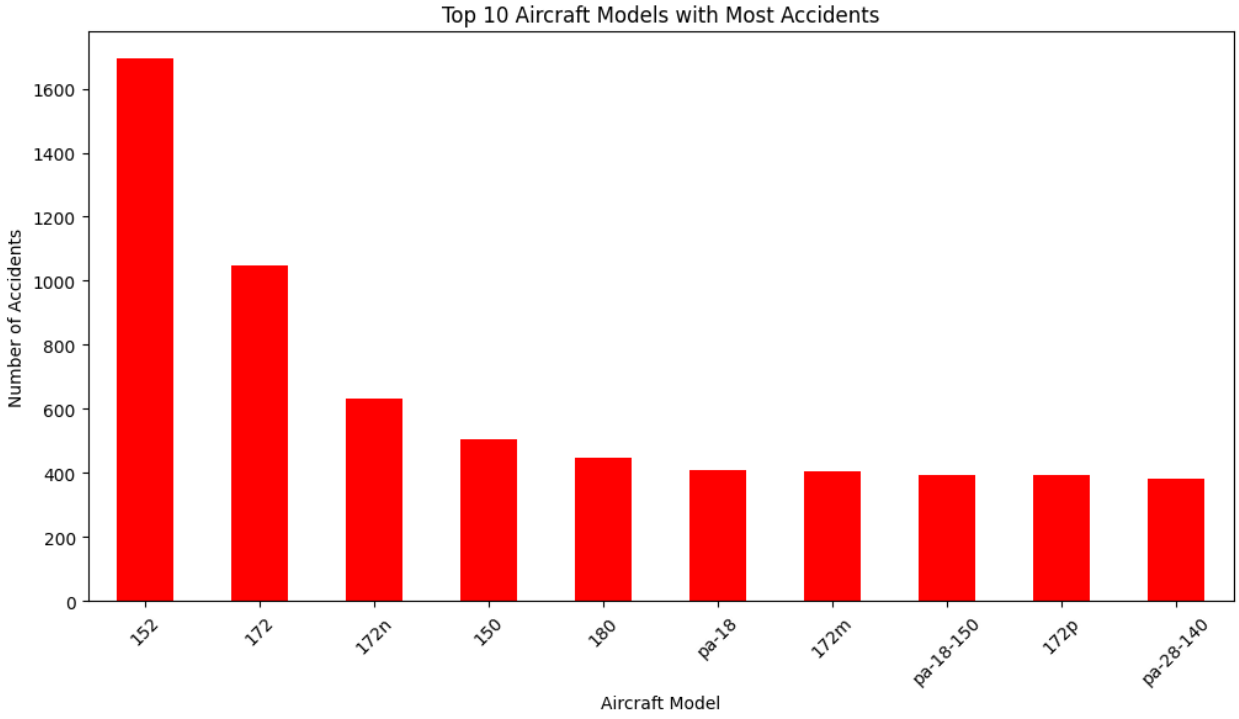


Top 10 Aircraft Models with Most Accidents

This analysis focuses on identifying the aircraft models with the highest number of recorded accidents. Understanding which models have been involved in the most incidents can provide valuable insights into potential risk factors and safety concerns.

```
# Count accidents by aircraft model
top_models = df['Model'].value_counts().head(10)
```

```
plt.figure(figsize=(12, 6))
top_models.plot(kind='bar', color='red')
plt.title("Top 10 Aircraft Models with Most Accidents")
plt.xlabel("Aircraft Model")
plt.ylabel("Number of Accidents")
plt.xticks(rotation=45)
plt.show()
```



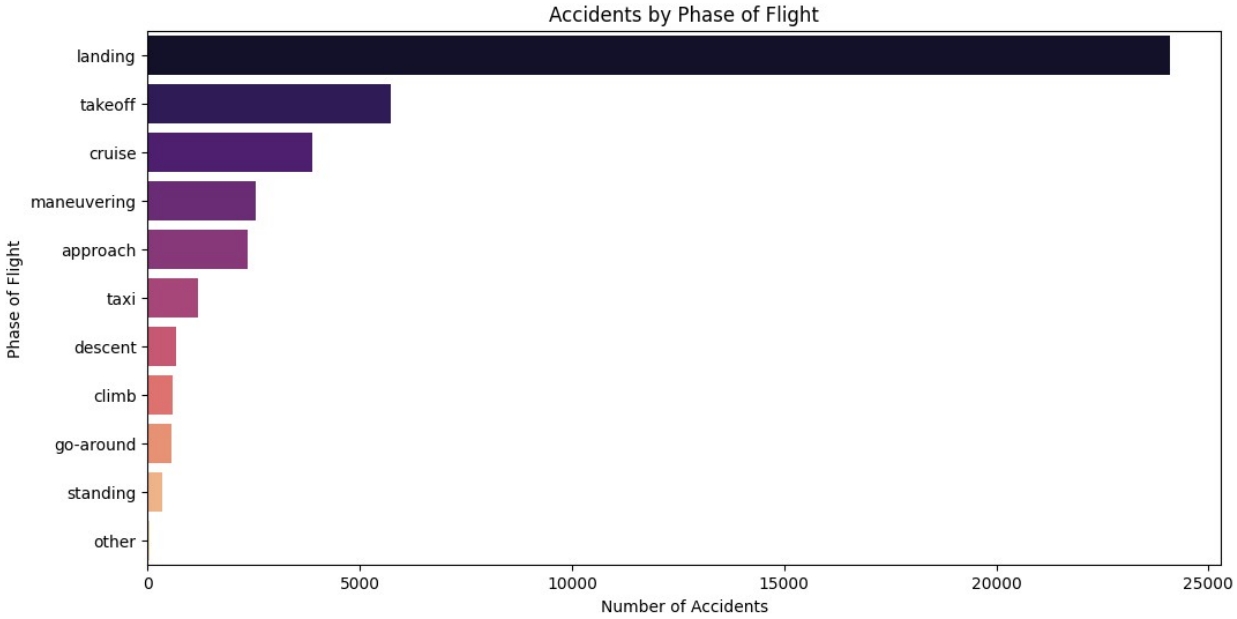
Accidents by Phase of Flight

The bar chart below shows the number of aircraft accidents categorized by the phase of flight in which they occurred.

- The x-axis represents the number of accidents.
- The y-axis lists different phases of flight (e.g., takeoff, cruise, landing).
- This analysis helps determine which flight phases are most prone to accidents, providing insights into critical risk periods during flight operations.

```
plt.figure(figsize=(12, 6))
sns.countplot(y=df['Broad.phase.of.flight'],
order=df['Broad.phase.of.flight'].value_counts().index,
palette='magma')
plt.xlabel("Number of Accidents")
plt.ylabel("Phase of Flight")
```

```
plt.title("Accidents by Phase of Flight")
plt.show()

C:\Users\hezronkatila\AppData\Local\Temp\
ipykernel_13492\2462185090.py:2: FutureWarning:

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

  sns.countplot(y=df['Broad.phase.of.flight'],
order=df['Broad.phase.of.flight'].value_counts().index,
palette='magma')
```



Insights from the Visualizations

1. Accident Trend Over Time

- The number of aircraft accidents fluctuates over the years, with noticeable spikes in certain periods.

- Potential reasons for these peaks include changes in aviation regulations, increased air traffic, or improvements in accident reporting.

2. Accidents by Country

- The **United States** has the highest number of accidents, likely due to its large aviation industry and high flight volume.

- Other countries with significant accident rates may have high air traffic or less stringent safety regulations.

3. Severity Distribution

- A substantial portion of incidents resulted in **minor injuries or no injuries**.
- However, a notable percentage of accidents involved **fatalities**, underscoring the need for continuous safety improvements in aviation.

4. Accidents by Aircraft Make & Model

- Certain aircraft manufacturers, such as **Cessna, Piper, and Boeing**, appear more frequently in accident records.
- However, higher accident counts may correlate with the **overall number of aircraft in operation** rather than inherent safety issues.
- The top **10 aircraft models** with the most accidents were identified, providing insights into potential risk factors.

5. Phase of Flight Analysis

- The majority of accidents occur during the **takeoff and landing phases**, which are the most critical moments of flight.
- **Cruise-phase accidents** are rare but tend to be severe when they do occur.

Conclusion and Final Thoughts

Summary of Findings

- Aircraft accidents have shown fluctuations over time, with specific years experiencing noticeable spikes.
- The **United States** has the highest number of recorded accidents, followed by other aviation-heavy regions.
- Injury severity analysis reveals that while many accidents result in minor or no injuries, a significant number involve **fatalities or serious injuries**.
- The **most frequently involved aircraft manufacturers** include major names like **Cessna, Piper, and Boeing**.
- Most accidents occur during critical flight phases, specifically **takeoff and landing**.
- Data inconsistencies, such as variations in categorical values, were cleaned to ensure analysis accuracy.

Business Recommendations

1. **Prioritize Low-Risk Aircraft** – Select aircraft models with **low accident rates and strong safety records** for commercial and private operations.

2. **Avoid High-Risk Models** – Exclude aircraft with a history of **high accident and fatality rates** from the purchase list.

3. **Improve Aircraft Maintenance & Inspections** – Implement **stricter and more frequent maintenance checks** to prevent mechanical failures.

4. **Enhance Pilot Training** – Focus on advanced training for **takeoff, landing, and emergency handling** to improve safety.

5. **Consider Manufacturer Reputation** – Opt for **manufacturers with proven reliability** and lower accident frequency.

6. **Strengthen Safety Measures for Critical Flight Phases** – Since most accidents happen during **takeoff and landing**, emphasize **pilot preparedness and weather monitoring**.

By leveraging **data-driven insights**, the company can make informed decisions when selecting aircraft, ultimately enhancing **safety, reliability, and risk management** in its aviation venture.

```
#df.to_csv("AviationDt_clean.csv" ,index=False)
#df.to_excel("AviationData_clean.xlsx", index=False,
engine="openpyxl")
```