Analyzing, Filtering, and Cleaning Aviation Database for the Safest Plane

Jupyter Notebook coded by Allison Ward, Rick Lataille, and Anthony Mansion

Project Goal:

The main thing we want coming out of the analysis is idenifying planes with the lowest risk in the U.S.

Source of Data:

Our initial data set was pulled from the National Transportation Safety Board's (NTSB) Aviation Accident data set from 1962 to 2023. The data contains information regarding civil aviation accidents and selected incidents in the United States and international waters. The data set we will be using is filled with about 90,000 rows of accidents involving all types of aircrafts.

Limitations:

The dataset used in this analysis was provided by The Flatiron School. It only shows planes in accidents; we do not know how many flights took place overall, so we cannot normalize our data. Additionally, it does not differentiate between hardware failures and pilot error. Therefore, the scope of our analysis is limited.

And of course, to start off the project in Python fashion, imported pandas to manipulate our new dataframe buddy.

In [1]:
```
#Import the modules we need
import pandas as pd
import numpy as np

# And this is the big data that we will be using
df = pd.read_csv('Aviation_Data.csv', low_memory=False)

print({len(df)})
```

```
{90348}
```

Here, we will be dropping columns we see no need for, the information won't help us with our goal. These columns are dropped because the information it provides has no use in finding the results our stakeholder is looking for, or even WE can't use it to find other helpful data beyond that.

In [2]:
```
# Drop the columns we know that we don't need
dropped_columns = ['Schedule', 'Report.Status', 'Publication.Date']
df.drop(columns = dropped_columns, inplace=True)
print(f"{len(df)} items.")
```

```
90348 items.
```

Now looking at the rest of the columns, we filter some columns for the information we need. We leave the original "df" alone and instead make another variable to hold our filtered information. The ways we filtered were:

- Filtering for rows with data from the last 10 years + turned into date-time, and also created days of the week using those, * * Filter data for aircrafts to airplanes only since that's the data we will be using
- Exclude the rows for planes that are amateur built since they... kind of screw the results we need over
- Filtered even more plane uses to continue providing relevant data adjusted to our stakeholder
- Filtering for the United States only. Filtering for the U.S. only is mainly because the rows that aren't in here aren't filled and can't tell of anything, and it even the same case for the U.S. territories that aren't between the Altantic and Pacific Oceans. Dropped from about 90,000 items to 7,000.

In [3]:
```
# Convert date column to datetime, then filter event dates to include 2013 and later
df['Event.Date'] = pd.to_datetime(df['Event.Date'])
df_filtered = df.loc[df['Event.Date'] >= '2013-01-01']
print(f"{len(df_filtered)} items.")

# Creating a new column with Day of Week
df_filtered['Day_Of_Week'] = df['Event.Date'].dt.day_name()

# Filter aircraft categories for Airplanes only
df_filtered = df_filtered.loc[df_filtered['Aircraft.Category'] == 'Airplane']
print(f"{len(df_filtered)} items.")

# Exclude Amateur-built planes
df_filtered = df_filtered.loc[df_filtered['Amateur.Built'] != 'Yes']
print(f"{len(df_filtered)} items.")

# Exclude certain identified purposes as irrelevant to our stakeholder
allowed_purposes = ['Personal', np.nan, 'Business', 'Executive/corporate', \
                    'Positioning', 'Other Work Use', 'Ferry', 'Unknown', 'Public Aircraft - Federal', \
                   'Public Aircraft - State', 'Public Aircraft - Local', 'Public Aircraft', 'PUBS']
df_filtered = df_filtered.loc[df_filtered['Purpose.of.flight'].isin(allowed_purposes)]
print(f"{len(df_filtered)} items.")

# Include only events that happened in the United States or US Territories
allowed_countries = ['United States']
df_filtered = df_filtered.loc[df_filtered['Country'].isin(allowed_countries)]
print(f"{len(df_filtered)} items.")

# Drop even more columns that are no longer useful
obsolete_columns = ['Event.Id', 'Country', 'Aircraft.Category', 'Registration.Number', 'Broad.phase.of.flight'
df_filtered.drop(columns = obsolete_columns, inplace=True)
```

```
15829 items.
13262 items.
11726 items.
9497 items.
7320 items.
```

```
<ipython-input-3-aa051abb3ac1>:7: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  df_filtered['Day_Of_Week'] = df['Event.Date'].dt.day_name()
```

Alriight, now that we've done as much filtering we believe we need, we move on to cleaning the remaining columns. And if you threw the CSV into Tableau, you'd notice that-- even though we filtered for the United States specifically-- some of the points in the map are NOT in the United States, or even U.S. territories. So the cell block below was to try and limit the lies the data was telling us, or just lazy humans being human or something.

In [4]:
```
# Filter for foreign locations not noted as foreign using the 'OF' state code in Location
df_filtered['State_Code'] = df_filtered['Location'].str.slice(-2)
df_filtered = df_filtered.loc[df_filtered['State_Code'] != 'OF']
print(f"{len(df_filtered)} items.")

# Drop rows that are missing latitude coordinates (also captures missing Longitude)
df_filtered.dropna(subset=['Latitude'], inplace=True)
print(f"{len(df_filtered)} items.")

#Converting latitude and longitude from Degrees, Minutes, and Seconds to Decimal Degrees

df_filtered.dropna(subset=['Latitude', 'Longitude'], inplace=True)

def convert_latitude(x):
    degrees = float(x[:2])
    minutes = float(x[2:4])
    seconds = float(x[4:6])
    return degrees + minutes/60 + seconds/3600

df_filtered["new_lats"] = df_filtered['Latitude'].map(convert_latitude)

def convert_longitude(x):
    degrees = float(x[:3])
    minutes = float(x[3:5])
    seconds = float(x[5:7])
    return -(degrees + minutes/60 + seconds/3600)

df_filtered["new_longs"] = df_filtered['Longitude'].map(convert_longitude)
```

```
7307 items.
7302 items.
```

As you may know, Python is case sensitive, and the data does NOT reflect that. And because of that, things that should be grouped together will be grouped seperatively based on one just starting with "A" instead of "a." So it's time to fix that. :) The original_makes variable wasn't useful in the long run, it was just as a way to see how many duplicates were left and things like that. Lamba function was used to try and clean the names. I know its a LOT of the same line repeating pretty much, but knowing the natural language processing or any other methods is currently beyond our levels, so unfortunately we have to do what we had to do. Lol

In [5]:
```
# Record original makes for later comparison
Original_makes = len(df_filtered['Make'].unique())
```

In [6]:

```
# These map functions will clean the 'Make' column, A-C
df_filtered['Make'] = df_filtered['Make'].map(lambda x: "Aerofab" if x.lower().strip()[:7]=="aerofab" else x)
df_filtered['Make'] = df_filtered['Make'].map(lambda x: "Aeroprakt" if x.lower().strip()[:9]=="aeroprakt" else
df_filtered['Make'] = df_filtered['Make'].map(lambda x: "Aeropro" if x.lower().strip()[:7]=="aeropro" else x)
df_filtered['Make'] = df_filtered['Make'].map(lambda x: "Aerostar" if x.lower().strip()[:8]=="aerostar" else x
df_filtered['Make'] = df_filtered['Make'].map(lambda x: "Aerostar" if x.lower().strip()[:3]=="s c" else x)
df_filtered['Make'] = df_filtered['Make'].map(lambda x: "Aerotek" if x.lower().strip()[:7]=="aerotek" else x)
df_filtered['Make'] = df_filtered['Make'].map(lambda x: "Air Tractor" if x.lower().strip()[:11]=="air tractor"
df_filtered['Make'] = df_filtered['Make'].map(lambda x: "Airbus" if x.lower().strip()[:6]=="airbus" else x)
df_filtered['Make'] = df_filtered['Make'].map(lambda x: "Airbus" if x.lower().strip()[:5]=="fouga" else x)
df_filtered['Make'] = df_filtered['Make'].map(lambda x: "Aircraft Mfg" if x.lower().strip()[:12]=="aircraft mf
df_filtered['Make'] = df_filtered['Make'].map(lambda x: "American Champion" if x.lower().strip()[:17]=="americ
df_filtered['Make'] = df_filtered['Make'].map(lambda x: "American Legend" if x.lower().strip()[:15]=="americar
df_filtered['Make'] = df_filtered['Make'].map(lambda x: "Arion" if x.lower().strip()[:5]=="arion" else x)
df_filtered['Make'] = df_filtered['Make'].map(lambda x: "Aviat" if x.lower().strip()[:5]=="aviat" else x)
df_filtered['Make'] = df_filtered['Make'].map(lambda x: "Avions" if x.lower().strip()[:6]=="avions" else x)
df_filtered['Make'] = df_filtered['Make'].map(lambda x: "BAE" if x.lower().strip()[:3]=="bae" else x)
df_filtered['Make'] = df_filtered['Make'].map(lambda x: "Boeing" if x.lower().strip()[:6]=="boeing" else x)
df_filtered['Make'] = df_filtered['Make'].map(lambda x: "Boeing" if x.lower().strip()[:9]=="mcdonnell" else x)
df_filtered['Make'] = df_filtered['Make'].map(lambda x: "Boeing" if x.lower().strip()[:7]=="douglas" else x)
df_filtered['Make'] = df_filtered['Make'].map(lambda x: "Boeing" if x.lower().strip()[:8]=="rockwell" else x)
df_filtered['Make'] = df_filtered['Make'].map(lambda x: "Bombardier" if x.lower().strip()[:10]=="bombardier" e
df_filtered['Make'] = df_filtered['Make'].map(lambda x: "Bombardier" if x.lower().strip()[:5]=="gates" else x)
df_filtered['Make'] = df_filtered['Make'].map(lambda x: "Bombardier" if x.lower().strip()[:7]=="learjet" else
df_filtered['Make'] = df_filtered['Make'].map(lambda x: "Bombardier" if x.lower().strip()[:8]=="canadair" else
df_filtered['Make'] = df_filtered['Make'].map(lambda x: "BAE" if x.lower().strip()[:12]=="british aero" else x
df_filtered['Make'] = df_filtered['Make'].map(lambda x: "Britten-Norman" if x.lower().strip()[:7]=="britten" e
df_filtered['Make'] = df_filtered['Make'].map(lambda x: "Bucker" if x.lower().strip()[:6]=="bucker" else x)
df_filtered['Make'] = df_filtered['Make'].map(lambda x: "Cirrus" if x.lower().strip()[:6]=="cirrus" else x)
df_filtered['Make'] = df_filtered['Make'].map(lambda x: "Convair" if x.lower().strip()[:12]=="consolidated" el
df_filtered['Make'] = df_filtered['Make'].map(lambda x: "CubCrafters" if x.lower().strip()[:3]=="cub" else x)
df_filtered['Make'] = df_filtered['Make'].map(lambda x: "Czech" if x.lower().strip()[:5]=="czech" else x)
```

In [7]:
```
# These map functions will clean the 'Make' column, D-N
df_filtered['Make'] = df_filtered['Make'].map(lambda x: "Daher" if x.lower().strip()[:3]=="s.o" else x)
df_filtered['Make'] = df_filtered['Make'].map(lambda x: "Daher" if x.lower().strip()[:3]=="soc" else x)
df_filtered['Make'] = df_filtered['Make'].map(lambda x: "Dassault" if x.lower().strip()[:8]=="dassault" else x
df_filtered['Make'] = df_filtered['Make'].map(lambda x: "De Havilland" if x.lower().strip()[:6]=="de hav" else
df_filtered['Make'] = df_filtered['Make'].map(lambda x: "De Havilland" if x.lower().strip()[:5]=="dehav" else
df_filtered['Make'] = df_filtered['Make'].map(lambda x: "Diamond" if x.lower().strip()[:7]=="diamond" else x)
df_filtered['Make'] = df_filtered['Make'].map(lambda x: "Eclipse" if x.lower().strip()[:7]=="eclipse" else x)
df_filtered['Make'] = df_filtered['Make'].map(lambda x: "Embraer" if x.lower().strip()[:7]=="embraer" else x)
df_filtered['Make'] = df_filtered['Make'].map(lambda x: "Evektor" if x.lower().strip()[:7]=="evektor" else x)
df_filtered['Make'] = df_filtered['Make'].map(lambda x: "Evolution" if x.lower().strip()[:9]=="evolution" else
df_filtered['Make'] = df_filtered['Make'].map(lambda x: "Extra" if x.lower().strip()[:5]=="extra" else x)
df_filtered['Make'] = df_filtered['Make'].map(lambda x: "Fairchild" if x.lower().strip()[:9]=="fairchild" else
df_filtered['Make'] = df_filtered['Make'].map(lambda x: "Fantasy Air" if x.lower().strip()[:7]=="fantasy" else
df_filtered['Make'] = df_filtered['Make'].map(lambda x: "Flight Design" if x.lower().strip()[:8]=="flight d" e
df_filtered['Make'] = df_filtered['Make'].map(lambda x: "Flightstar" if x.lower().strip()[:7]=="flights" else
df_filtered['Make'] = df_filtered['Make'].map(lambda x: "FPNA" if x.lower().strip()[:4]=="fpna" else x)
df_filtered['Make'] = df_filtered['Make'].map(lambda x: "Glasair" if x.lower().strip()[:7]=="glasair" else x)
df_filtered['Make'] = df_filtered['Make'].map(lambda x: "Golden Circle" if x.lower().strip()[:8]=="golden c" e
df_filtered['Make'] = df_filtered['Make'].map(lambda x: "Gulfstream" if x.lower().strip()[:10]=="gulfstream" e
df_filtered['Make'] = df_filtered['Make'].map(lambda x: "Gulfstream" if x.lower().strip()[:3]=="iai" else x)
df_filtered['Make'] = df_filtered['Make'].map(lambda x: "Honda" if x.lower().strip()[:5]=="honda" else x)
df_filtered['Make'] = df_filtered['Make'].map(lambda x: "Jabiru" if x.lower().strip()[:6]=="jabiru" else x)
df_filtered['Make'] = df_filtered['Make'].map(lambda x: "Lancair" if x.lower().strip()[:7]=="lancair" else x)
df_filtered['Make'] = df_filtered['Make'].map(lambda x: "Maxair" if x.lower().strip()[:6]=="maxair" else x)
df_filtered['Make'] = df_filtered['Make'].map(lambda x: "Meyers" if x.lower().strip()[:6]=="meyers" else x)
df_filtered['Make'] = df_filtered['Make'].map(lambda x: "Mooney" if x.lower().strip()[:6]=="mooney" else x)
df_filtered['Make'] = df_filtered['Make'].map(lambda x: "M-Squared" if x.lower().strip()[:9]=="m-squared" else
df_filtered['Make'] = df_filtered['Make'].map(lambda x: "Nanchang" if x.lower().strip()[:8]=="nanchang" else x
df_filtered['Make'] = df_filtered['Make'].map(lambda x: "Northrop Grumman" if x.lower().strip()[:7]=="grumman"
df_filtered['Make'] = df_filtered['Make'].map(lambda x: "Northrop Grumman" if x.lower().strip()[:8]=="northrop
df_filtered['Make'] = df_filtered['Make'].map(lambda x: "North American" if x.lower().strip()[:8]=="north am"
df_filtered['Make'] = df_filtered['Make'].map(lambda x: "North Wing" if x.lower().strip()[:7]=="north w" else
df_filtered['Make'] = df_filtered['Make'].map(lambda x: "North Wing" if x.lower().strip()[:6]=="northw" else x
```

In [8]:
```
# These map functions will clean the 'Make' column, N-Z
df_filtered['Make'] = df_filtered['Make'].map(lambda x: "Orlican" if x.lower().strip()[:7]=="orlican" else x)
df_filtered['Make'] = df_filtered['Make'].map(lambda x: "Phantom" if x.lower().strip()[:7]=="phantom" else x)
df_filtered['Make'] = df_filtered['Make'].map(lambda x: "Pilatus" if x.lower().strip()[:7]=="pilatus" else x)
df_filtered['Make'] = df_filtered['Make'].map(lambda x: "Piper" if x.lower().strip()[:9]=="new piper" else x)
df_filtered['Make'] = df_filtered['Make'].map(lambda x: "Piper" if x.lower().strip()[:5]=="piper" else x)
df_filtered['Make'] = df_filtered['Make'].map(lambda x: "Pipistrel" if x.lower().strip()[:4]=="pipi" else x)
df_filtered['Make'] = df_filtered['Make'].map(lambda x: "Pitts" if x.lower().strip()[:5]=="pitts" else x)
df_filtered['Make'] = df_filtered['Make'].map(lambda x: "Pzl Okecie" if x.lower().strip()[:3]=="pzl" else x)
df_filtered['Make'] = df_filtered['Make'].map(lambda x: "Quad City" if x.lower().strip()[:4]=="quad" else x)
df_filtered['Make'] = df_filtered['Make'].map(lambda x: "Quest" if x.lower().strip()[:5]=="quest" else x)
df_filtered['Make'] = df_filtered['Make'].map(lambda x: "Quicksilver" if x.lower().strip()[:5]=="quick" else x
df_filtered['Make'] = df_filtered['Make'].map(lambda x: "Rans" if x.lower().strip()[:4]=="rans" else x)
df_filtered['Make'] = df_filtered['Make'].map(lambda x: "Remos" if x.lower().strip()[:5]=="remos" else x)
df_filtered['Make'] = df_filtered['Make'].map(lambda x: "Rockwell" if x.lower().strip()[:8]=="rockwell" else x
df_filtered['Make'] = df_filtered['Make'].map(lambda x: "Ryan" if x.lower().strip()[:4]=="ryan" else x)
df_filtered['Make'] = df_filtered['Make'].map(lambda x: "Scoda" if x.lower().strip()[:5]=="scoda" else x)
df_filtered['Make'] = df_filtered['Make'].map(lambda x: "Short" if x.lower().strip()[:5]=="short" else x)
df_filtered['Make'] = df_filtered['Make'].map(lambda x: "Stearman" if x.lower().strip()[:8]=="stearman" else x
df_filtered['Make'] = df_filtered['Make'].map(lambda x: "Taylorcraft" if x.lower().strip()[:7]=="taylorc" else
df_filtered['Make'] = df_filtered['Make'].map(lambda x: "Textron" if x.lower().strip()[:6]=="cessna" else x)
df_filtered['Make'] = df_filtered['Make'].map(lambda x: "Textron" if x.lower().strip()[:4]=="rath" else x)
df_filtered['Make'] = df_filtered['Make'].map(lambda x: "Textron" if x.lower().strip()[:4]=="rayt" else x)
df_filtered['Make'] = df_filtered['Make'].map(lambda x: "Textron" if x.lower().strip()[:7]=="textron" else x)
df_filtered['Make'] = df_filtered['Make'].map(lambda x: "Textron" if x.lower().strip()[:5]=="beech" else x)
df_filtered['Make'] = df_filtered['Make'].map(lambda x: "Textron" if x.lower().strip()[:6]=="hawker" else x)
df_filtered['Make'] = df_filtered['Make'].map(lambda x: "TL Ultralight" if x.lower().strip()[:2]=="tl" else x)
df_filtered['Make'] = df_filtered['Make'].map(lambda x: "Vans" if x.lower().strip()[:4]=="vans" else x)
df_filtered['Make'] = df_filtered['Make'].map(lambda x: "Waco" if x.lower().strip()[:4]=="waco" else x)
df_filtered['Make'] = df_filtered['Make'].map(lambda x: "Zlin" if x.lower().strip()[:4]=="zlin" else x)
```

In [9]:
```
# Show the amount of consolidation in makes
print(f"The original {Original_makes} makes have been reduced to {len(df_filtered['Make'].unique())} makes.")
```

```
The original 670 makes have been reduced to 433 makes.
```

Getting rid of whack "NaN", "None", or "Uknown" by appropriately filling in values for data usage

In [10]:
```
# Change "NaN" to 'None' or 'Unknown', as appropriate
df_filtered['Injury.Severity'].fillna('None', inplace=True)
df_filtered['Aircraft.damage'].fillna('Unknown', inplace=True)
df_filtered['Purpose.of.flight'].fillna('Unknown', inplace=True)
df_filtered['Engine.Type'].fillna('Unknown', inplace=True)
df_filtered['FAR.Description'].fillna('Unknown', inplace=True)
df_filtered['Number.of.Engines'].fillna('Unknown', inplace=True)

# This will convert all 'unknown' type entries to 'Unknown' in the Air.carrier field
df_filtered['Air.carrier'].fillna('Unknown', inplace=True)
df_filtered['Air.carrier'] = df_filtered['Air.carrier'].astype(str).map(
    lambda x: "Unknown" if x.lower().strip()[:3]=="unk" else x)

# This will convert all 'unknown' type entries to 'Unknown' in the Weather.Condition field
df_filtered['Weather.Condition'].fillna('Unknown', inplace=True)
df_filtered['Weather.Condition'] = df_filtered['Weather.Condition'].astype(str).map(
    lambda x: "Unknown" if x.lower().strip()[:3]=="unk" else x)
```

All of these lines have completely different functions, and some is more filtering than cleaning, or neither, but had to make a little more moves to get even MORE of the data comparisons we wanted to include.

In [11]:
```
# Put all Makes into Title case, for readability
df_filtered['Make'] = df_filtered['Make'].map(lambda x: x.title())

# Use dt functions to extract year and month and create new columns
df_filtered['Year'] = df['Event.Date'].dt.year
df_filtered['Month'] = df['Event.Date'].dt.month

# Create a new column to simplify the large jet analysis
separate_large_jets = ["Airbus", "Boeing", "Embraer"]
df_filtered['Large_Jets'] = df_filtered['Make'].map(lambda x: "Other" if x not in separate_large_jets else x)

# Create a new column to simplify the large jet analysis
separate_small_jets = ["Bombardier", "Dassault", "Gulfstream", "Honda", "Textron"]
df_filtered['Small_Jets'] = df_filtered['Make'].map(lambda x: "Other" if x not in separate_small_jets else x)

# Create a new column summing fatal and serious injuries
df_filtered['Major_Injuries'] = df_filtered['Total.Fatal.Injuries'] + df_filtered['Total.Serious.Injuries']
```

With all that filtered data, saved it to a new csv file for an even easier time making visualizations of the data since all the info we need won't be surrounded by the other random information. We also did this instead of overwriting the original "just in case", y'know? Just in case we wanted to undo something, we could simply overwrite the csv and throw it back into Tableau real fast.

In [12]:
```
# Write to a new CSV file/overwrites CSV file
df_filtered.to_csv('Filtered_Aviation_Data.csv', index=False)
```