Airplane Crashes since 1908

By: Matthew Cooper

This project uses a dataset that contains information about airplane crashes starting from 1908. The data was scrapped from a website (not by me) which can be found here: http://www.planecrashinfo.com/.

So much time was spent identifying and fixing the massive number of errors in this dataset, that I regularly contemplated if I should end the project and move on to another. Ultimately, I decided to make this project strictly a learning experience instead of a full analysis project.

Even after all of the corrections had been made, I believe there were still errors in the 'ac_type' column. Since the variations in the entries were so numerous, I felt that spending any more time on it would be a waste. At that point, I decided to move on to visualization in Tableau.

This project was a challenge. I still learned a lot and I am happy that I chose to work on it. Apart from getting more comfortable with the Python environment, this project also allowed me to become reintroduced to Tableau.

The code below was written in Python. The final "dashboard" visualization was performed in Tableau.

Dataset Details:

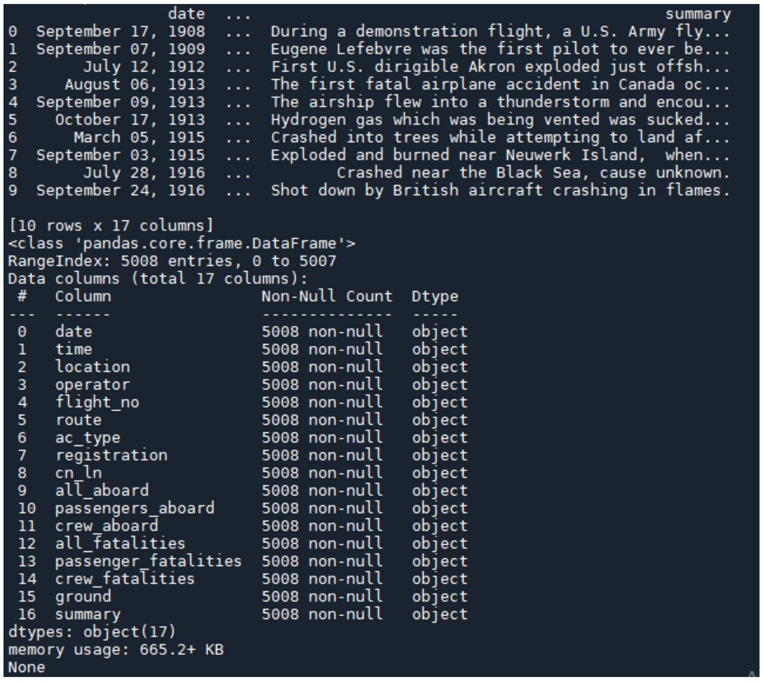
Dataset source: https://www.kaggle.com/datasets/landfallmotto/airplane-crashes-dataset-since-1908
Column Names:

- Date: Date of accident, in the format - January 01, 2001
- Time: Local time, in 24 hr. format unless otherwise specified
- Operator: Airline or operator of the aircraft
- Flight #: Flight number assigned by the aircraft operator
- Route: Complete or partial route flown prior to the accident
- AC Type: Aircraft type
- Reg: ICAO registration of the aircraft
- cn / ln: Construction or serial number / Line or fuselage number
- Aboard: Total aboard (passengers / crew)
- Passengers aboard : Passengers abroad
- Crew aboard : Crew abroad
- All fatalities : Total fatalities aboard (passengers / crew)
- Passenger fatalities: Total Passenger fatalities
- Crew fatalities: Total Crew fatalities
- Ground: Total killed on the ground
- Summary: Brief description of the accident and cause if known

Number of data rows: 5008
Missing or incomplete or erroneous data: A lot. Way more than I anticipated.

The Project:

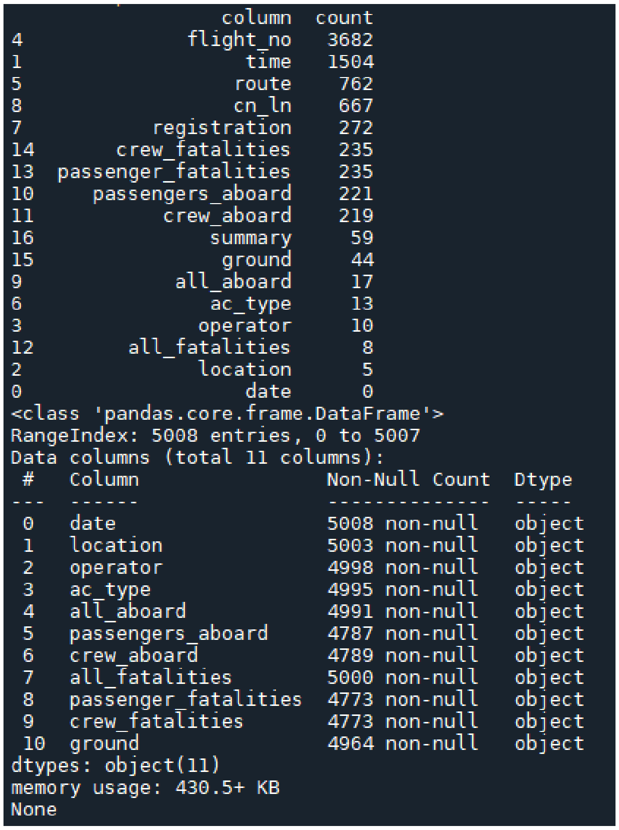
```
# -*- coding: utf-8 -*-
"""
@author: Matt
"""

import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import scipy as sp
import collections as cl
import seaborn as sns
import re
from collections import OrderedDict

np.random.seed(1029384756)

#Loading the dataset
#data = pd.read_csv('C:/Datasets/Plane_Crash_Data/Airplane_crashes_1908.csv')
data =
pd.read_csv('C:/Datasets/Plane_Crash_Data/Airplane_crashes_dataset_since_1908.
csv', encoding="ISO-8859-1")

#Printing a short version of the data for easy viewing
print(data.head(10))

#Prints the details of the data
print(data.info())

#Since data.head negates the ability to see a good data printout
#I decided to look directly at the dataframe to see the details
```



```
                 date  ...                                            summary
0  September 17, 1908  ...  During a demonstration flight, a U.S. Army fly...
1  September 07, 1909  ...  Eugene Lefebvre was the first pilot to ever be...
2       July 12, 1912  ...  First U.S. dirigible Akron exploded just offsh...
3     August 06, 1913  ...  The first fatal airplane accident in Canada oc...
4  September 09, 1913  ...  The airship flew into a thunderstorm and encou...
5    October 17, 1913  ...  Hydrogen gas which was being vented was sucked...
6      March 05, 1915  ...  Crashed into trees while attempting to land af...
7  September 03, 1915  ...  Exploded and burned near Neuwerk Island,  when...
8       July 28, 1916  ...         Crashed near the Black Sea, cause unknown.
9  September 24, 1916  ...  Shot down by British aircraft crashing in flames.

[10 rows x 17 columns]
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5008 entries, 0 to 5007
Data columns (total 17 columns):
 #   Column               Non-Null Count  Dtype 
---  ------               --------------  ----- 
 0   date                 5008 non-null   object
 1   time                 5008 non-null   object
 2   location             5008 non-null   object
 3   operator             5008 non-null   object
 4   flight_no            5008 non-null   object
 5   route                5008 non-null   object
 6   ac_type              5008 non-null   object
 7   registration         5008 non-null   object
 8   cn_ln                5008 non-null   object
 9   all_aboard           5008 non-null   object
 10  passengers_aboard    5008 non-null   object
 11  crew_aboard          5008 non-null   object
 12  all_fatalities       5008 non-null   object
 13  passenger_fatalities 5008 non-null   object
 14  crew_fatalities      5008 non-null   object
 15  ground               5008 non-null   object
 16  summary              5008 non-null   object
dtypes: object(17)
memory usage: 665.2+ KB
None
```

So, the printouts were not very helpful. I had to open the dataframe and look at the data directly. The first thing I noticed was the large amount of missing data in many of the columns. First thing to do was to impute actual NaN where data was missing. Then I decided to drop the columns where the missing data was really high, or where I decided the columns were not helpful.

```
#Replace Question marks with Null
data = data.replace('?', np.NaN)
NaNcount = data.isna().sum()

#Counts the number of null values by column
NaNcount = NaNcount.to_frame()
NaNcount = NaNcount.reset_index()
NaNcount = NaNcount.rename(columns={0 : 'count'})
NaNcount = NaNcount.rename(columns={'index' : 'column'})
NaNcount = NaNcount.sort_values(by=['count'], ascending= False)
print(NaNcount)

#Drop columns
data = data.drop(columns=['flight_no', 'time', 'summary', 'route', 
'registration', 'cn_ln'])

#Prints the details of the data
print(data.info())
```



```
                 column  count
4             flight_no   3682
1                  time   1504
5                 route    762
8                 cn_ln    667
7          registration    272
14      crew_fatalities    235
13  passenger_fatalities   235
10     passengers_aboard   221
11          crew_aboard    219
16              summary     59
15               ground     44
9             all_aboard    17
6               ac_type     13
3              operator     10
12        all_fatalities     8
2              location      5
0                  date      0
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5008 entries, 0 to 5007
Data columns (total 11 columns):
 #   Column               Non-Null Count  Dtype 
---  ------               --------------  ----- 
 0   date                 5008 non-null   object
 1   location             5003 non-null   object
 2   operator             4998 non-null   object
 3   ac_type              4995 non-null   object
 4   all_aboard           4991 non-null   object
 5   passengers_aboard    4787 non-null   object
 6   crew_aboard          4789 non-null   object
 7   all_fatalities       5000 non-null   object
 8   passenger_fatalities 4773 non-null   object
 9   crew_fatalities      4773 non-null   object
 10  ground               4964 non-null   object
dtypes: object(11)
memory usage: 430.5+ KB
None
```

OK, that cleans it up a little better. Unfortunately, I noticed a significant problem with the 'ac_type' column. The data entries contained numerous errors. Spelling and spacing issues were rampant in this column. The errors were not consistent at all, so it required me to go through manually and identify the errors.

```
#----The following massive block of code fixes spelling and other data entry errors
#----within the 'ac_type' column
#----The errors are due to inconsistent data entry on the website where the data was pulled from

#Fixes special errors that cannot be globally modified
data['ac_type'] = data['ac_type'].replace('FD Type Zeppelin', 'Zeppelin FD Type')
data['ac_type'] = data['ac_type'].replace('Super Zeppelin (airship)', 'Zeppelin Super (airship')
data['ac_type'] = data['ac_type'].replace('Royal Zeppelin Works ZR-2 (airship)', 'Zeppelin Royal Works ZR-2 (airship)')

data['ac_type'] = data['ac_type'].replace('AirbusA310-304', 'Airbus A310-304')
data['ac_type'] = data['ac_type'].replace('Armstrong Whitworth', 'Armstrong-Whitworth')
data['ac_type'] = data['ac_type'].replace('B17G Flying Fortress', 'B-17G Flying Fortress')
```

```
data['ac_type'] = data['ac_type'].replace('ConvairCV-440', 'Convair CV-440')
data['ac_type'] = data['ac_type'].replace('DHC-5 Buffalo', 'De-Havilland Canada DHC-5
Buffalo')
data['ac_type'] = data['ac_type'].replace('DHC-6 Twin Otter 300 / NAMC YS-11', 'De-
Havilland Canada DHC-6 Twin Otter 300 / NAMC YS-11')
data['ac_type'] = data['ac_type'].replace('DC-2-243', 'Douglas DC-2-243')
data['ac_type'] = data['ac_type'].replace('DC-3-65TP', 'Douglas DC-3-65TP')
data['ac_type'] = data['ac_type'].replace('Domier Delphin III (flying boat)', 'Dormier
Delphin III (flying boat)')

data['ac_type'] = data['ac_type'].replace('EMB 721C Sertanejo', 'Embraer EMB 721C
Sertanejo')
data['ac_type'] = data['ac_type'].replace('Embraer-110 C-95B Bandeirante', 'Embraer
110 C-95B Bandeirante')

data['ac_type'] = data['ac_type'].replace('Helicopter, Hughes 369HS', 'Hughes 369HS
Helicopter')
data['ac_type'] = data['ac_type'].replace('Ilyushin76TD', 'Ilyushin 76TD')
data['ac_type'] = data['ac_type'].replace('Lear Jet 24A', 'Learjet 24A')
data['ac_type'] = data['ac_type'].replace('Let-410UVP-E', 'Let 410UVP-E')
data['ac_type'] = data['ac_type'].replace('Lisnov Li-2', 'Lisunov Li-2')
data['ac_type'] = data['ac_type'].replace('Lockhed 10 Electra', 'Lockheed 10 Electra')
data['ac_type'] = data['ac_type'].replace('McDonnel F-4E Phantom II', 'McDonnell F-4E
Phantom II')
data['ac_type'] = data['ac_type'].replace('Mil- Mi-17B-5', 'Mil Mi-17B-5')
data['ac_type'] = data['ac_type'].replace('PBY4-2 Privateer / PB4Y-2', 'PBY 4-2
Privateer / PB4Y-2')

data['ac_type'] = data['ac_type'].replace('DC3(C47)', 'Douglas DC3(C47)')
data['ac_type'] = data['ac_type'].replace('Pitcairns PA-6', 'Pitcairn PA-6 Mailwing')
data['ac_type'] = data['ac_type'].replace('Saab340B', 'Saab 340B')
data['ac_type'] = data['ac_type'].replace('Savbia-Marchetti S-73P', 'Savoia-Marchetti
S-73P')
data['ac_type'] = data['ac_type'].replace('Shaanxi Yunshuji Y-8/Yunshuji Y-8',
'Shaanxi Y-8/Yunshuji Y-8')
data['ac_type'] = data['ac_type'].replace('Transportes A�reos Orientales',
'Transportes Aereos Orientales')
data['ac_type'] = data['ac_type'].replace('Tuolev 134AK', 'Tupolev 134AK')
data['ac_type'] = data['ac_type'].replace('de Hav Can. DHC-6 Tw Otter 100/ Cessna',
'De-Havilland Canada DHC-6 Twin Otter 100/ Cessna')
data['ac_type'] = data['ac_type'].replace('B-17C Flying Fortress', 'Boeing B-17C
Flying Fortress')
data['ac_type'] = data['ac_type'].replace('B-17G Flying Fortress', 'Boeing B-17G
Flying Fortress')
data['ac_type'] = data['ac_type'].replace('NAMC-YS-11-111', 'NAMC YS-11-111')
data['ac_type'] = data['ac_type'].replace('ATR42-320', 'ATR 42-320')


#Using Regex to make corrections to data
data['ac_type'] = data['ac_type'].replace('(Pilatus Britten Norman)', 'Pilatus-
Britten-Norman',regex=True).astype(str)
data['ac_type'] = data['ac_type'].replace('(Royal Zeppelin Works)', 'Royal-Zeppelin-
Works',regex=True).astype(str)
```

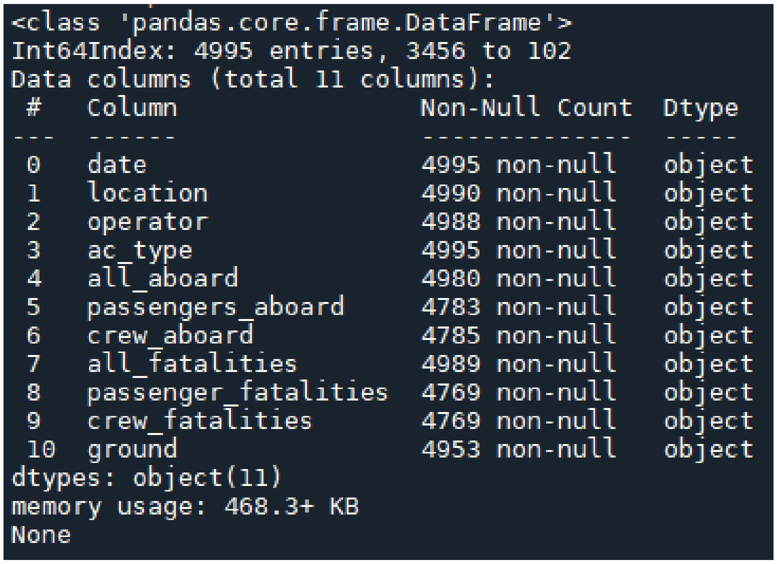
```
data['ac_type'] = data['ac_type'].replace('(Savoia Marchetti)', 'Savoia-Marchetti',regex=True).astype(str)
data['ac_type'] = data['ac_type'].replace('(Shorts)', 'Short',regex=True).astype(str)
data['ac_type'] = data['ac_type'].replace('(Travel Air)', 'Travel-Air',regex=True).astype(str)
data['ac_type'] = data['ac_type'].replace('(ATR-)', 'ATR ',regex=True).astype(str)
data['ac_type'] = data['ac_type'].replace('(North American)', 'North-American',regex=True).astype(str)
data['ac_type'] = data['ac_type'].replace('(PBY4-2)', 'PBY 4-2',regex=True).astype(str)
data['ac_type'] = data['ac_type'].replace('(Sikorksky)', 'Sikorsky',regex=True).astype(str)
data['ac_type'] = data['ac_type'].replace('(Swear,)', 'Swearingen',regex=True).astype(str)
data['ac_type'] = data['ac_type'].replace('(Zepplin)', 'Zeppelin',regex=True).astype(str)

data['ac_type'] = data['ac_type'].replace('(de Havilland)', 'De-Havilland',regex=True).astype(str)
data['ac_type'] = data['ac_type'].replace('(deHavilland)', 'De-Havilland',regex=True).astype(str)
data['ac_type'] = data['ac_type'].replace('(de havilland)', 'De-Havilland',regex=True).astype(str)
data['ac_type'] = data['ac_type'].replace('(De Havilland)', 'De-Havilland',regex=True).astype(str)
data['ac_type'] = data['ac_type'].replace('(de Hvilland)', 'De-Havilland',regex=True).astype(str)


data['ac_type'] = data['ac_type'].replace('(Sud Aviation)', 'Sud-Aviation',regex=True).astype(str)
data['ac_type'] = data['ac_type'].replace('(Lior�-et-Olivier)', 'Liore-et-Olivier',regex=True).astype(str)
data['ac_type'] = data['ac_type'].replace('(Liore et Olivier)', 'Liore-et-Olivier',regex=True).astype(str)
data['ac_type'] = data['ac_type'].replace('(Bl�riot)', 'Bleriot',regex=True).astype(str)
data['ac_type'] = data['ac_type'].replace('(Lat�co�re)', 'Latecoere',regex=True).astype(str)
data['ac_type'] = data['ac_type'].replace('(Sirkorsky)', 'Sikorsky',regex=True).astype(str)

data['ac_type'] = data['ac_type'].replace('(Dirigible)', 'Zeppelin',regex=True).astype(str)
data['ac_type'] = data['ac_type'].replace('(Airship)', 'Zeppelin',regex=True).astype(str)
data['ac_type'] = data['ac_type'].replace('(Goodyear-Zeppelin)', 'Goodyear Zeppelin',regex=True).astype(str)

data['ac_type'] = data['ac_type'].replace('(BAe)', 'BAE',regex=True).astype(str)
data['ac_type'] = data['ac_type'].replace('(Bae)', 'BAE',regex=True).astype(str)
data['ac_type'] = data['ac_type'].replace('(Cams)', 'CAMS',regex=True).astype(str)
```

```
data['ac_type'] = data['ac_type'].replace('(Beech )', 'Beechcraft
',regex=True).astype(str)
data['ac_type'] = data['ac_type'].replace('(Boeing Vertol)', 'Boeing-
Vertol',regex=True).astype(str)
data['ac_type'] = data['ac_type'].replace('(British Aerospace)',
'BAE',regex=True).astype(str)
data['ac_type'] = data['ac_type'].replace('(Britten Norman)', 'Britten-
Norman',regex=True).astype(str)
data['ac_type'] = data['ac_type'].replace('(Curtis)',
'Curtiss',regex=True).astype(str)
data['ac_type'] = data['ac_type'].replace('(Curtisss)',
'Curtiss',regex=True).astype(str)
data['ac_type'] = data['ac_type'].replace('(Curtiss )', 'Curtiss-Wright
',regex=True).astype(str)
data['ac_type'] = data['ac_type'].replace('(Curtiss Wright)', 'Curtiss-
Wright',regex=True).astype(str)

data['ac_type'] = data['ac_type'].replace('(Dassault )', 'Dassault-Breguet
',regex=True).astype(str)
data['ac_type'] = data['ac_type'].replace('(Dassault Breguet)', 'Dassault-
Breguet',regex=True).astype(str)
data['ac_type'] = data['ac_type'].replace('(Fairchild )', 'Fairchild-Hiller
',regex=True).astype(str)
data['ac_type'] = data['ac_type'].replace('(Fairchild Hiller)', 'Fairchild-
Hiller',regex=True).astype(str)

data['ac_type'] = data['ac_type'].replace('(Grummand)',
'Grumman',regex=True).astype(str)
data['ac_type'] = data['ac_type'].replace('(Hawker Siddeley)', 'Hawker-
Siddeley',regex=True).astype(str)

data['ac_type'] = data['ac_type'].replace('(Ilushin )', 'Ilyushin
',regex=True).astype(str)
data['ac_type'] = data['ac_type'].replace('(Illysushin )', 'Ilyushin
',regex=True).astype(str)
data['ac_type'] = data['ac_type'].replace('(Illyushin )', 'Ilyushin
',regex=True).astype(str)
data['ac_type'] = data['ac_type'].replace('(Ilysushin )', 'Ilyushin
',regex=True).astype(str)
data['ac_type'] = data['ac_type'].replace('(Mc Donnell)',
'McDonnell',regex=True).astype(str)
data['ac_type'] = data['ac_type'].replace('(McDonnell Douglas)', 'McDonnell-
Douglas',regex=True).astype(str)

data['ac_type'] = data['ac_type'].replace('(A�rospatiale)',
'Aerospatiale',regex=True).astype(str)
data['ac_type'] = data['ac_type'].replace('(Doublas)',
'Douglas',regex=True).astype(str)
data['ac_type'] = data['ac_type'].replace('(Aerospatiale/Aeritalia)',
'Aerospatiale',regex=True).astype(str)
data['ac_type'] = data['ac_type'].replace('(Armstrong Whitworth)', 'Armstrong-
Whitworth',regex=True).astype(str)
```

```
data['ac_type'] = data['ac_type'].replace('(Aviation Traders)', 'Aviation-
Traders',regex=True).astype(str)
```

Well, that was a nightmare. So much time was spent on that error fixing that I actually thought about stopping the project. At this point I decided to use this project strictly as a learning experience instead of a full analysis project.

```
#Fixes an error where data was entered twice into one cell
data['ac_type'] = (data['ac_type'].str.split()
.apply(lambda x: OrderedDict.fromkeys(x).keys())
.str.join(' '))


#Sorts dataset by aircraft type
data = data.sort_values(by=['ac_type'], ascending= True)


#Reintroduce NaN values back into ac_type (regex corrections turn Nan
into string)
data['ac_type'] = data['ac_type'].replace('nan', np.NaN)


#Remove rows where ac_type is Nan
data = data.dropna(subset=['ac_type'])

#Prints the details of the data
print(data.info())
```



```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 4995 entries, 3456 to 102
Data columns (total 11 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   date                  4995 non-null   object
 1   location              4990 non-null   object
 2   operator              4988 non-null   object
 3   ac_type               4995 non-null   object
 4   all_aboard            4980 non-null   object
 5   passengers_aboard     4783 non-null   object
 6   crew_aboard           4785 non-null   object
 7   all_fatalities        4989 non-null   object
 8   passenger_fatalities  4769 non-null   object
 9   crew_fatalities       4769 non-null   object
 10  ground                4953 non-null   object
dtypes: object(11)
memory usage: 468.3+ KB
None
```

A few other things I found that needed to be fixed before moving on. A duplicate entry in one cell needed to be corrected. All of the regex corrections changed the 'ac_type' column into string, so the NaN entries were now just words. Then all rows with NaN/na in 'ac_type' were dropped from the dataframe.

```
#Splits the ac_type column to create new column with manufacturer
data2 = data["ac_type"].str.split(" ", n = 1, expand = True)
data2 = data2.rename(columns={ 0 : 'manufacturer'})
data2 = data2.rename(columns={ 1 : 'extra'})

#Add manufacturer column to original dataset
data['manufacturer'] = data2['manufacturer']

#reorder columns
cols = data.columns.tolist()
print(cols)
cols = ['date', 'location', 'operator', 'ac_type', 'manufacturer',
'all_aboard',
    'passengers_aboard', 'crew_aboard', 'all_fatalities',
'passenger_fatalities', 'crew_fatalities']
data = data[cols]
```

One thing I noticed was that the 'ac_type' column contained the manufacturer and plane type. I wanted to create a new column that only contained the manufacturer.

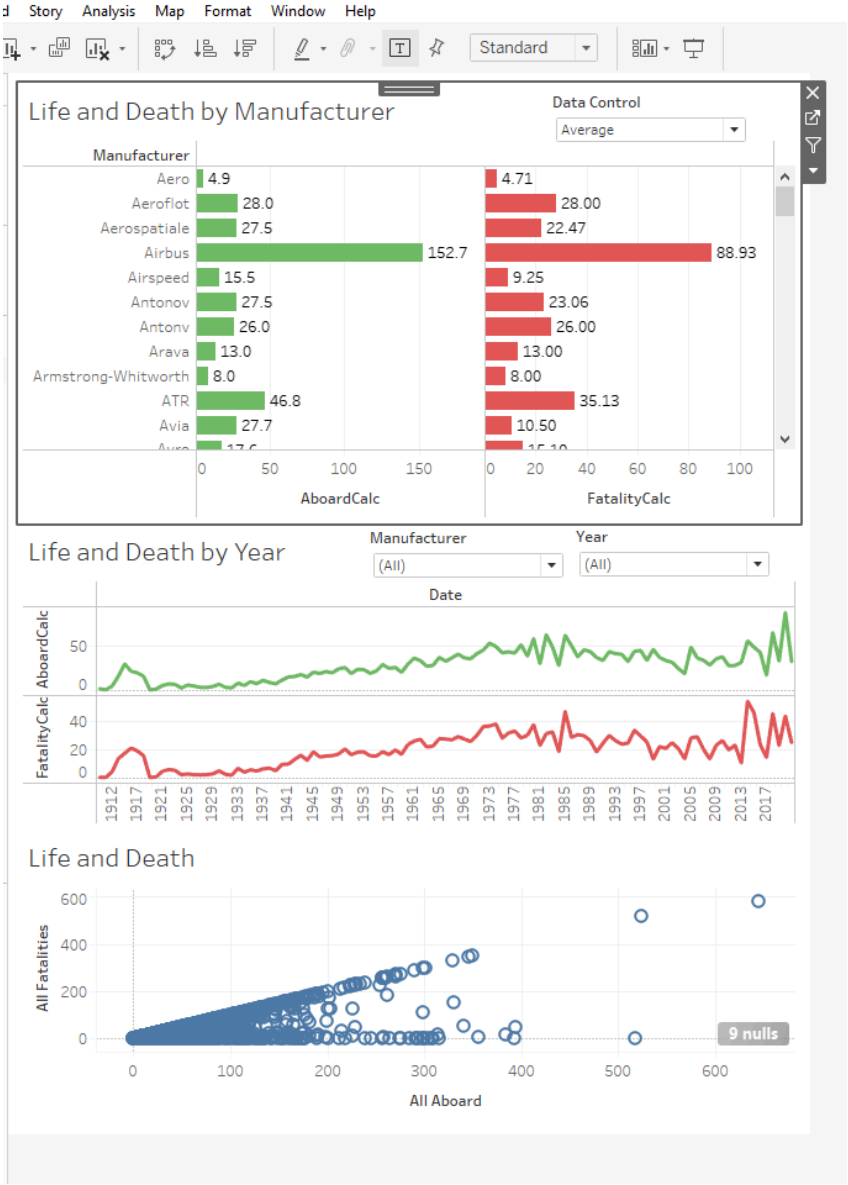
```
#/-/-/- This block gets a count off ac_type and manufacturer columns
#/-/-/- It helps to make sure the earlier corrections were performed
accurately
#/-/-/- It also allows the ability to check for more errors to fix
#Counts the instances of each column
actypenames = data['ac_type'].value_counts()
manunames = data['manufacturer'].value_counts()

#Converts to dataframe, resets index, renames columns
actypenames = actypenames.to_frame()
actypenames = actypenames.reset_index()
actypenames = actypenames.rename(columns={'ac_type' : 'count'})
actypenames = actypenames.rename(columns={'index' : 'ac_type'})
manunames = manunames.to_frame()
manunames = manunames.reset_index()
manunames = manunames.rename(columns={'manufacturer' : 'count'})
manunames = manunames.rename(columns={'index' : 'manufacturer'})
#/-/-/- End of correction checking block
```

This was a check I did to see how many errors in 'ac_type' could still exist. A couple were located and corrected but since I had already spent so much time on error-fixing, I decided that the remaining errors were not worth the time-sink.

```
#Prepares to removes all data where airplane type is only two entries
#creates a list of entries where the count is greater than 2
nameslst = actypenames[actypenames['count']<2]['ac_type']
nameslst2=manunames[manunames['count']<2]['manufacturer']

#Removes all data where airplane type is only two entries
data3 = data[~data.ac_type.isin(nameslst)]
data4 = data[~data.manufacturer.isin(nameslst2)]

#Exports cleaned dataset for visualization in Tableau
data4.to_csv(path_or_buf='C:/Users/Matt/Downloads/Aircraft_Crashes_190
8/Airplane_crashes_1988_Cleaned.csv', index=False)
```

In this last bit of code, I decided to remove any entries of manufacturer <= 2. I did this because I felt that the data of any manufacturer less that 3 would not be useful. Looking back at this, I probably could have increased that to 10. I then exported the resulting dataset to csv to import into Tableau.

In Tableau, I created some visualizations and ended with a dashboard which is shown above.



Data Control: Average

Manufacturer: (All)

Year: (All)