binadata

November 25, 2023

0.1 Airplane_Crashes_and_Fatalities_Since_1908

0.1.1 Groups:

Team 5

- Aya Mchaouri
- Oussama Majid
- Ayoub Sami
- Khawla Elgoumiri
- Imane Ait Azekri
- Abdelaziz El Adarissi

0.1.2 Imports

[866]:
```
# Importing necessary libraries for data analysis and visualization.
import pandas as pd
import numpy as np
import plotly.graph_objects as go
from plotly.subplots import make_subplots
from matplotlib import pyplot as plt
import seaborn as sns
import plotly.express as px
```

0.1.3 Setup

[867]:
```
## plots configuration
sns.set(style="whitegrid")
plt.figure(figsize=(10, 6))
```

[867]:
```
<Figure size 1000x600 with 0 Axes>
```

```
<Figure size 1000x600 with 0 Axes>
```

0.1.4 Data description

This dataset contains information about **airplane crashes around the world**. The data spans September 1908 to August 2009. A variety of entities broadcast data about the air crashes, including **Location** , **operator**, **Fatality** , **Aircraft type** and **Reason for the accident** . This dataset currently contains **5268 records of air crashes**.

Name of columns	Description
Time	The time of the incident
Location	The location of the incident
Operator	The operator of the aircraft
Flight #	The flight number of the aircraft
Route	The route of the aircraft
Type	The type of aircraft
Registration	The registration of the aircraft
cn/In	The construction number/serial number of the aircraft
Aboard	The number of people on board the aircraft
Fatalities	The number of fatalities in the incident
Ground	The number of people on the ground killed in the incident
Summary	A summary of the incident

0.1.5 Data Loading

[893]:
```
pd_frame=pd.read_csv("C:/Users/Paname/Desktop/bigdataproject/
↪Airplane_Crashes_and_Fatalities_Since_1908.csv")
```

0.1.6 Data Exploration

[894]:
```
# Display the first few rows of the DataFrame
pd_frame.head()
```

[894]:
```
         Date   Time                            Location  \
0  09/17/1908  17:18                 Fort Myer, Virginia
1  07/12/1912  06:30             AtlantiCity, New Jersey
2  08/06/1913    NaN  Victoria, British Columbia, Canada
3  09/09/1913  18:30                  Over the North Sea
4  10/17/1913  10:30          Near Johannisthal, Germany

                Operator Flight #          Route                    Type  \
0    Military - U.S. Army      NaN  Demonstration        Wright Flyer III
1    Military - U.S. Navy      NaN    Test flight               Dirigible
2                Private        -            NaN        Curtiss seaplane
3  Military - German Navy      NaN            NaN  Zeppelin L-1 (airship)
4  Military - German Navy      NaN            NaN  Zeppelin L-2 (airship)

  Registration cn/In  Aboard  Fatalities  Ground  \
0          NaN     1     2.0         1.0     0.0
1          NaN   NaN     5.0         5.0     0.0
2          NaN   NaN     1.0         1.0     0.0
3          NaN   NaN    20.0        14.0     0.0
4          NaN   NaN    30.0        30.0     0.0

                                             Summary
```

```
0   During a demonstration flight, a U.S. Army fly…
1   First U.S. dirigible Akron exploded just offsh…
2   The first fatal airplane accident in Canada oc…
3   The airship flew into a thunderstorm and encou…
4   Hydrogen gas which was being vented was sucked…
```

[895]:
```
# Display the shape of the DataFrame (number of rows, number of columns)
pd_frame.shape
```

[895]:
```
(5268, 13)
```

[896]:
```
# Display basic information about the DataFrame
pd_frame.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5268 entries, 0 to 5267
Data columns (total 13 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Date          5268 non-null   object
 1   Time          3049 non-null   object
 2   Location      5248 non-null   object
 3   Operator      5250 non-null   object
 4   Flight #      1069 non-null   object
 5   Route         3561 non-null   object
 6   Type          5241 non-null   object
 7   Registration  4933 non-null   object
 8   cn/In         4040 non-null   object
 9   Aboard        5246 non-null   float64
 10  Fatalities    5256 non-null   float64
 11  Ground        5246 non-null   float64
 12  Summary       4878 non-null   object
dtypes: float64(3), object(10)
memory usage: 535.2+ KB
```

[897]:
```
#Display summary statistics for numerical variables.
pd_frame.describe()
```

[897]:
```
            Aboard   Fatalities       Ground
count  5246.000000  5256.000000  5246.000000
mean     27.554518    20.068303     1.608845
std      43.076711    33.199952    53.987827
min       0.000000     0.000000     0.000000
25%       5.000000     3.000000     0.000000
50%      13.000000     9.000000     0.000000
75%      30.000000    23.000000     0.000000
max     644.000000   583.000000  2750.000000
```

0.1.7 Data Preparation

Data checks

[898]:
```
#cheking missing values :
col=pd_frame.columns
for i in col:
    print("The number of missing values in the {0}  :{1}".format(i,pd_frame[i].
 ↪isnull().sum()))
pd_frame.shape
```

```
The number of missing values in the Date  :0
The number of missing values in the Time  :2219
The number of missing values in the Location  :20
The number of missing values in the Operator  :18
The number of missing values in the Flight #  :4199
The number of missing values in the Route  :1707
The number of missing values in the Type  :27
The number of missing values in the Registration  :335
The number of missing values in the cn/In  :1228
The number of missing values in the Aboard  :22
The number of missing values in the Fatalities  :12
The number of missing values in the Ground  :22
The number of missing values in the Summary  :390
```

[898]:
```
(5268, 13)
```

[899]:
```
col=pd_frame.columns
for i in col:
    print("The Type of {0} is {1}".format(i,pd_frame[i].dtype))
```

```
The Type of Date is object
The Type of Time is object
The Type of Location is object
The Type of Operator is object
The Type of Flight # is object
The Type of Route is object
The Type of Type is object
The Type of Registration is object
The Type of cn/In is object
The Type of Aboard is float64
The Type of Fatalities is float64
The Type of Ground is float64
The Type of Summary is object
```

[900]:
```
# Get the top 5 largest values from the 'Fatalities' column
pd_frame['Fatalities'].nlargest(5)
```

[900]:
```
2963    583.0
3568    520.0
4455    349.0
2726    346.0
3562    329.0
Name: Fatalities, dtype: float64
```

[875]:
```
# Get the top 5 largest values from the 'Aboard' column
pd_frame["Aboard"].nlargest(5)
```

[875]:
```
2963    644.0
3568    524.0
4645    517.0
3378    394.0
4536    393.0
Name: Aboard, dtype: float64
```

[876]:
```
(pd_frame["Fatalities"]==0).sum()
```

[876]:
```
58
```

Useless columns

Delete unwanted columns:

- **cn/In** : The construction number/serial number of the aircraft.
- **Registration** : The registration of the aircraft.

- **Flight #** : The flight number of the aircraft .

[901]:
```
columns_to_drop = ['cn/In','Flight #','Registration']

columns_to_drop_existing = [col for col in columns_to_drop if col in pd_frame.
 ↪columns]

if columns_to_drop_existing:
    pd_frame = pd_frame.drop(columns=columns_to_drop_existing, errors='ignore')
```

[903]:
```
pd_frame.head()
```

[903]:
```
         Date   Time                             Location  \
0  09/17/1908  17:18                  Fort Myer, Virginia
1  07/12/1912  06:30              AtlantiCity, New Jersey
2  08/06/1913    NaN  Victoria, British Columbia, Canada
3  09/09/1913  18:30                   Over the North Sea
4  10/17/1913  10:30           Near Johannisthal, Germany

                 Operator        Route                    Type  Aboard  \
0    Military - U.S. Army  Demonstration        Wright Flyer III     2.0
1    Military - U.S. Navy    Test flight               Dirigible     5.0
2                 Private            NaN        Curtiss seaplane     1.0
3  Military - German Navy            NaN  Zeppelin L-1 (airship)    20.0
4  Military - German Navy            NaN  Zeppelin L-2 (airship)    30.0

   Fatalities  Ground                                            Summary
0         1.0     0.0  During a demonstration flight, a U.S. Army fly…
1         5.0     0.0  First U.S. dirigible Akron exploded just offsh…
2         1.0     0.0  The first fatal airplane accident in Canada oc…
3        14.0     0.0  The airship flew into a thunderstorm and encou…
4        30.0     0.0  Hydrogen gas which was being vented was sucked…
```

Transformations de Donnees

Add new column:

- **Survivors**
- **Year**
- **Day**
- **Heure**

[904]:
```python
# ADD Survivors column:
pd_frame["Survivors"] = pd_frame["Aboard"] - pd_frame["Fatalities"]
```

[905]:
```python
#Extracting year, month, and day as features
pd_frame['Date'] = pd.to_datetime(pd_frame['Date'], format='%m/%d/%Y')
pd_frame['Year'] = pd_frame['Date'].dt.year
pd_frame['Month'] = pd_frame['Date'].dt.month
pd_frame['Day'] = pd_frame['Date'].dt.day
```

[906]:
```python
#Extracting Heure feature
pd_frame['Heure'] = pd_frame['Time'].str.split(":", expand = True)[0]
pd_frame['Heure'] = pd_frame['Heure'].str.replace("'", ":")
pd_frame['Heure'] = pd_frame['Heure'].str.replace(".", ":")
pd_frame['Heure'].unique()
```

[906]:
```
array(['17', '06', nan, '18', '10', '01', '15', '23', '05', '08', '07',
       '21', '02', '13', '09', 'c', '22', '20', '04', '14', '12', '00',
       '03', '19', '11', '16', '1', 'c16', '12:20', '18:40', '114', 'c14',
       '0943', '2', '22:08', '8', '9'], dtype=object)
```

[907]:
```python
l=['0943','c14','0','c16','114','c','nan']
l1=['1','2','8','9']
l2=['12:20','18:40','22:08']
for i in l:
    pd_frame= pd_frame[pd_frame['Heure'] != i]
```

```
for i in l2:
        pd_frame['Heure'] = pd_frame['Heure'].str.split(":", expand = True)[0]

for index, row in pd_frame.iterrows():
    if str(row["Heure"]) in l1:
        pd_frame.at[index, "Heure"] = '0' + str(row["Heure"])

pd_frame.sort_values(by='Heure', inplace=True)
pd_frame['Heure'].unique()
```

```
[907]: array(['00', '01', '02', '03', '04', '05', '06', '07', '08', '09', '10',
              '11', '12', '13', '14', '15', '16', '17', '18', '19', '20', '21',
              '22', '23', nan], dtype=object)
```

```
[908]: # ADD country column
       pd_frame['Country'] = pd_frame['Location'].str.split(',').str[-1]
```

```
[909]: #cheking
       print("All Columns : ",pd_frame.columns)
       pd_frame.head()
```

```
All Columns :  Index(['Date', 'Time', 'Location', 'Operator', 'Route', 'Type',
'Aboard',
       'Fatalities', 'Ground', 'Summary', 'Survivors', 'Year', 'Month', 'Day',
       'Heure', 'Country'],
      dtype='object')
```

```
[909]:           Date   Time                             Location  \
       2100 1967-02-07  00:02              Albuquerque, New Mexico
       4729 2000-07-19  00:30                       Linneus, Maine
       1141 1951-07-21  00:00                   Near Sitka, Alaska
       1076 1950-08-31  00:03              Near Wadi Natrun, Egypt
       2069 1966-09-01  00:47  Near Ljubljana, Slovenia, Yugoslavia

                           Operator               Route  \
       2100  Avanti Aviation -Air Taxi                 NaN
       4729          Airwave Transport  Moncton - Montreal
       1141   Canadian PacifiAir Lines   Vancouver - Tokyo
       1076       Trans World Airlines        Cairo - Rome
       2069          Britannia Airways   Luton - Ljubljana

                                  Type  Aboard  Fatalities  Ground  \
       2100                Cessna 210-5A     2.0         2.0     0.0
       4729   Grumman G-159 Gulfstream I     2.0         2.0     0.0
       1141                Douglas C-54A    37.0        37.0     0.0
       1076  Lockheed 749A Constellation    55.0        55.0     0.0
       2069          Bristol Britannia 102   117.0        98.0     0.0
```

```
                                                Summary  Survivors  Year  \
2100  Pilot misjudged altitude and distance and cras…        0.0  1967
4729  After declaring an emergency the cargo plane c…        0.0  2000
1141  Disappeared with no trace over the PacifiOcean…        0.0  1951
1076  While en route from Cairo to Rome, witnesses o…        0.0  1950
2069  The plane crashed into forest during a landing…       19.0  1966

      Month  Day Heure     Country
2100      2    7    00  New Mexico
4729      7   19    00       Maine
1141      7   21    00      Alaska
1076      8   31    00       Egypt
2069      9    1    00  Yugoslavia
```

Cleaning data

Missing values

[910]:
```
#Before Droping or filling missing values
pd_frame.isnull().sum()
```

[910]:
```
Date             0
Time          2219
Location        20
Operator        18
Route         1706
Type            27
Aboard          22
Fatalities      12
Ground          21
Summary        390
Survivors       22
Year             0
Month            0
Day              0
Heure         2219
Country         20
dtype: int64
```

[886]:
```
#Drop missing values in"Location","Type","Operator","Fatalities","Ground" and␣
 ↪"Aboard"
Subset=["Location","Type","Operator","Fatalities","Ground","Aboard"]
pd_frame=pd_frame.dropna(subset=Subset)
```

[887]:
```
#fill missing values in Time and Heure features
pd_frame['Time'].fillna(pd_frame['Time'].mode().iloc[0], inplace=True)
```

```
pd_frame['Heure'].fillna(pd_frame['Heure'].mode().iloc[0], inplace=True)
```

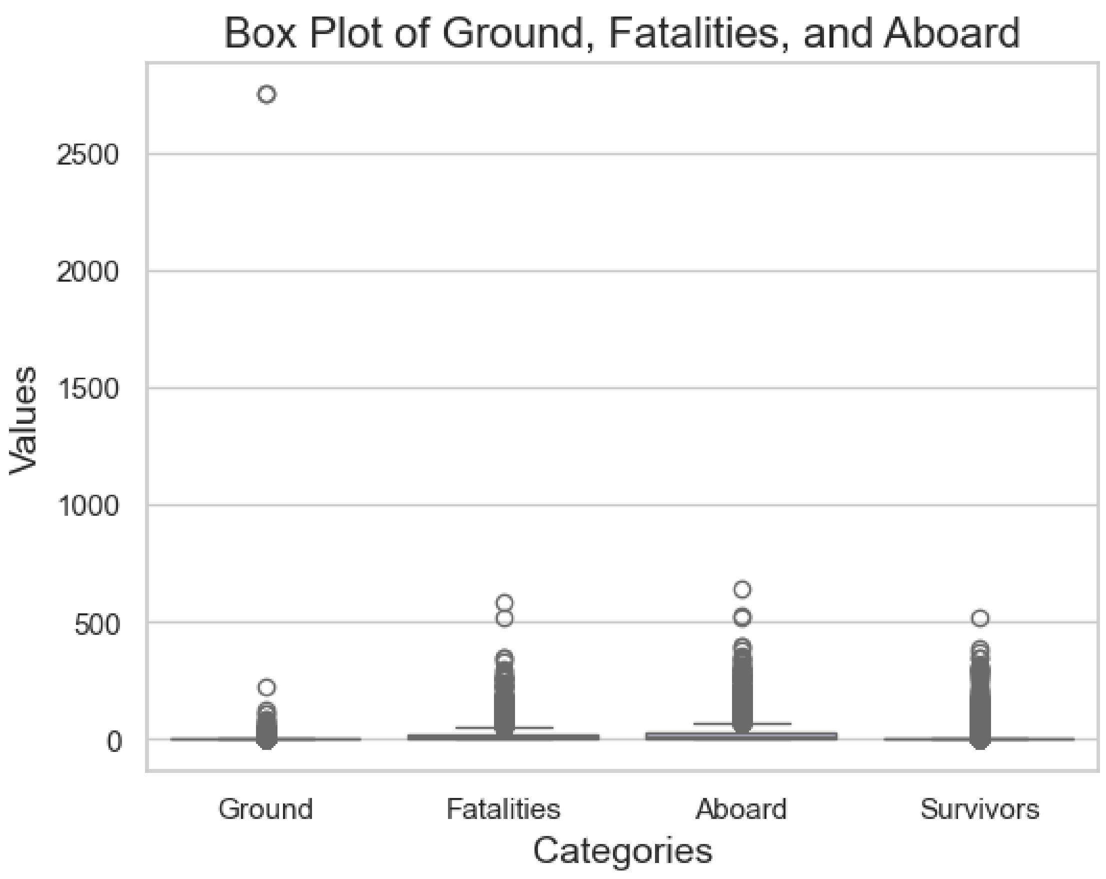
[888]:
```
#fill missing values in "Summary" and "Route "
pd_frame['Summary'] = pd_frame['Summary'].replace({pd.NA: 'No Remarque', None:␣
↪'No Remarque'})
pd_frame['Route']=pd_frame['Route'].fillna('unavailable')
```

[889]:
```
# After Droping or filling missing values
pd_frame.isnull().sum()
```

[889]:
```
Date           0
Time           0
Location       0
Operator       0
Route          0
Type           0
Aboard         0
Fatalities     0
Ground         0
Summary        0
Survivors      0
Year           0
Month          0
Day            0
Heure          0
Country        0
dtype: int64
```

Outliers

[773]:
```
# Create a box plot for the specified columns
sns.boxplot(data=pd_frame[['Ground', 'Fatalities', 'Aboard','Survivors']],␣
↪palette="Set3")
# Set the title and labels
plt.title('Box Plot of Ground, Fatalities, and Aboard', fontsize=16)
plt.xlabel('Categories', fontsize=14)
plt.ylabel('Values', fontsize=14)
plt.show()
```



[774]:
```
#cheking Outliers in Ground columns
Q1 = pd_frame['Ground'].quantile(0.25)
Q3 = pd_frame['Ground'].quantile(0.75)
IQR = Q3 - Q1
# Indices of outliers based on the IQR method
outliers_indices_Ground = pd_frame.index[(pd_frame['Ground'] < Q1 - 1.5 * IQR)␣
 ↪| (pd_frame['Ground'] > Q3 + 1.5 * IQR)].tolist()
print('the number of outlies is ',len(outliers_indices_Ground))
# Replace outliers with the mean
mean = pd_frame['Ground'].mean()
pd_frame.loc[outliers_indices_Ground, 'Ground'] = mean
```

```
the number of outlies is  216
```

[789]:
```
#cheking Outliers in Fatalities columns
Q1 = pd_frame['Fatalities'].quantile(0.25)
Q3 = pd_frame['Fatalities'].quantile(0.75)
IQR = Q3 - Q1
# Indices of outliers based on the IQR method
```

```
outliers_indices_Fatalities = pd_frame.index[(pd_frame['Fatalities'] < Q1 - 1.5␣
↪* IQR) | (pd_frame['Fatalities'] > Q3 + 1.5 * IQR)].tolist()
print('the number of outlies is ',len(outliers_indices_Fatalities))
# Replace outliers with the mean
mean = pd_frame['Fatalities'].mean()
pd_frame.loc[outliers_indices_Fatalities, 'Fatalities'] = mean

outliers_indices_Aboard_next = pd_frame.index[(pd_frame['Fatalities'] < Q1 - 1.
↪5 * IQR) | (pd_frame['Fatalities'] > Q3 + 1.5 * IQR)].tolist()
print('the number of outlies  apres la supression is␣
↪',len(outliers_indices_Aboard_next))
```

```
the number of outlies is  0
the number of outlies  apres la supression is  0
```

[787]:
```
#cheking Outliers in Aboard columns
Q1 = pd_frame['Aboard'].quantile(0.25)
Q3 = pd_frame['Aboard'].quantile(0.75)
IQR = Q3 - Q1
# Indices of outliers based on the IQR method
outliers_indices_Aboard = pd_frame.index[(pd_frame['Aboard'] < Q1 - 1.5 * IQR)␣
↪| (pd_frame['Aboard'] > Q3 + 1.5 * IQR)].tolist()
print('the number of outlies is ',len(outliers_indices_Aboard))
# Replace outliers with the mean
mean = pd_frame['Aboard'].mean()
pd_frame.loc[outliers_indices_Aboard, 'Aboard'] = mean

outliers_indices_Aboard_next = pd_frame.index[(pd_frame['Aboard'] < Q1 - 1.5 *␣
↪IQR) | (pd_frame['Aboard'] > Q3 + 1.5 * IQR)].tolist()
print('the number of outlies  apres la supression is␣
↪',len(outliers_indices_Aboard_next))
```

```
the number of outlies is  0
the number of outlies  apres la supression is  0
```

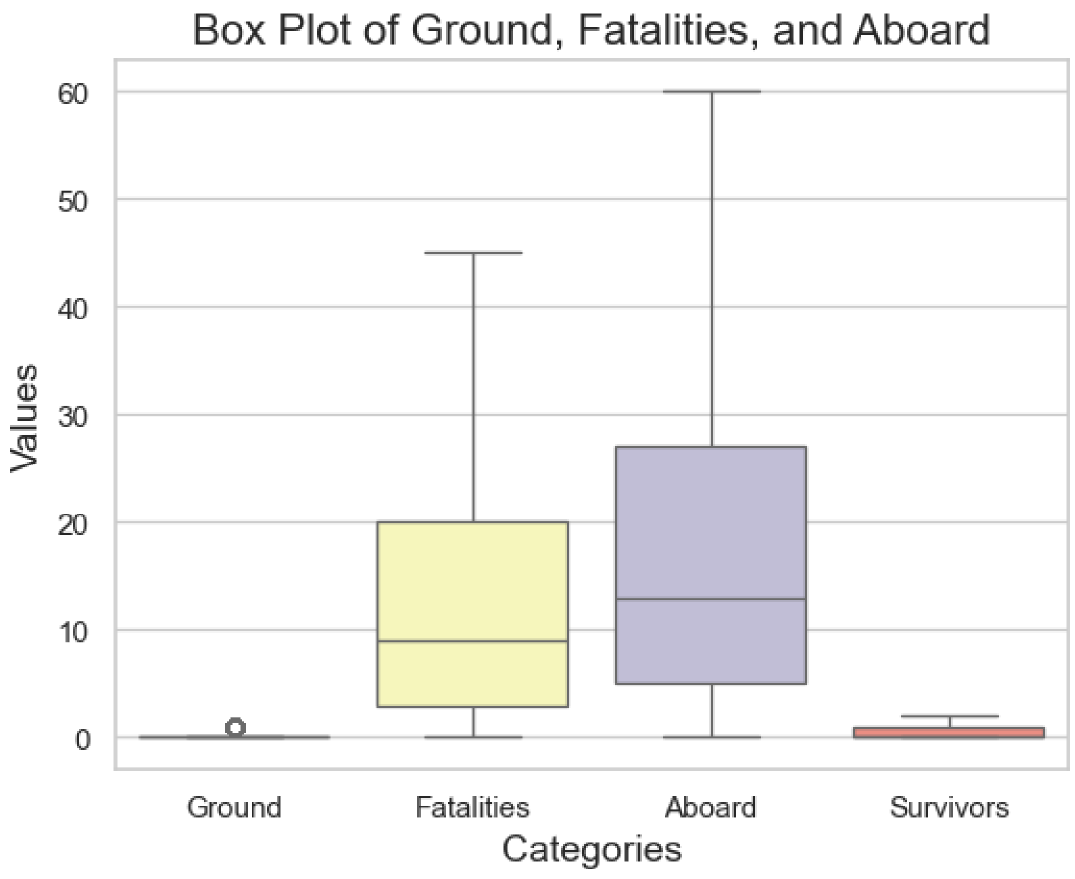
[784]:
```
#cheking Outliers in Aboard columns
Q1 = pd_frame['Survivors'].quantile(0.25)
Q3 = pd_frame['Survivors'].quantile(0.75)
IQR = Q3 - Q1
# Indices of outliers based on the IQR method
outliers_indices_Survivors = pd_frame.index[(pd_frame['Survivors'] < Q1 - 1.5 *␣
↪IQR) | (pd_frame['Survivors'] > Q3 + 1.5 * IQR)].tolist()
print('the number of outlies is ',len(outliers_indices_Survivors))
# Replace outliers with the mean
mean = pd_frame['Survivors'].mean()
pd_frame.loc[outliers_indices_Survivors, 'Survivors'] = mean
```

```
outliers_indices_Aboard_next = pd_frame.index[(pd_frame['Survivors'] < Q1 - 1.5␣
↪* IQR) | (pd_frame['Survivors'] > Q3 + 1.5 * IQR)].tolist()
print('the number of outlies  apres la supression␣
↪is',len(outliers_indices_Aboard_next))
```

```
the number of outlies is  0
the number of outlies  apres la supression is 0
```

[790]:

```
# Create a box plot for the specified columns
sns.boxplot(data=pd_frame[['Ground', 'Fatalities', 'Aboard','Survivors']],␣
↪palette="Set3")
# Set the title and labels
plt.title('Box Plot of Ground, Fatalities, and Aboard', fontsize=16)
plt.xlabel('Categories', fontsize=14)
plt.ylabel('Values', fontsize=14)
plt.show()
```



Change Types

[791]:
```
pd_frame['Fatalities']=pd_frame['Fatalities'].astype(int)
pd_frame['Ground']=pd_frame['Ground'].astype(int)
pd_frame['Aboard']=pd_frame['Aboard'].astype(int)
pd_frame['Survivors']=pd_frame['Survivors'].astype(int)

pd_frame.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 5172 entries, 2100 to 5267
Data columns (total 16 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Date        5172 non-null   datetime64[ns]
 1   Time        5172 non-null   object
 2   Location    5172 non-null   object
 3   Operator    5172 non-null   object
 4   Route       5172 non-null   object
 5   Type        5172 non-null   object
 6   Aboard      5172 non-null   int32
 7   Fatalities  5172 non-null   int32
 8   Ground      5172 non-null   int32
 9   Summary     5172 non-null   object
 10  Survivors   5172 non-null   int32
 11  Year        5172 non-null   int32
 12  Month       5172 non-null   int32
 13  Day         5172 non-null   int32
 14  Heure       5172 non-null   object
 15  Country     5172 non-null   object
dtypes: datetime64[ns](1), int32(7), object(8)
memory usage: 674.5+ KB
```

0.1.8 Data analysis

Analyses by Country:

- which country has reported the most fatilities due to air crashes?

[792]:
```
pd0= pd_frame.groupby('Country')[['Fatalities']].count()

pd0.sort_values(by='Fatalities',ascending=False)
pd0 = pd_frame.groupby('Country')[['Fatalities']].count().
 ↪sort_values(by='Fatalities',ascending=False).head(20)
pd0_sorted = pd0.sort_values(by='Fatalities', ascending=False)

fig = px.bar(x=pd0_sorted.index, y=pd0_sorted['Fatalities'],␣
 ↪color=pd0_sorted['Fatalities'],
             labels={'x': 'Country', 'y': 'Number of Fatalities','color':
 ↪'nbrs'},
             title='Number of Fatalities by Country',
             color_continuous_scale=px.colors.sequential.Viridis)

fig.update_layout(xaxis_tickangle=-45, xaxis=dict(tickfont=dict(size=12)),␣
 ↪yaxis=dict(title=dict(text='Number of Fatalities', font=dict(size=15))))
fig.show()
```

Analysis by Date :

- How many air crashes occur in the world on average each year?
- Which year had the most air crashes in the world?
- Which Month had the most air crashes in the world?

[793]:
```
crashes_year = pd_frame["Year"].value_counts().sort_index()
crashes_month = pd_frame["Month"].value_counts().sort_index()
months = ["January", "February", "March", "April", "May", "June", "July",␣
 ↪"August", "September", "October", "November", "December"]

fig = make_subplots(rows=2, cols=1, subplot_titles=["Crashes per year",␣
 ↪"Crashes in a month"])
# Add traces
fig.add_trace(go.Bar(x=crashes_year.index, y=crashes_year.values,␣
 ↪marker_color='blue'), row=1, col=1)
fig.add_trace(go.Bar(x=crashes_month.index, y=crashes_month.values), row=2,␣
 ↪col=1)
# Update layout
fig.update_layout(
    height=800,
    width=1200,
    showlegend=False,
    title_text="Crashes Statistics",
)

# Update x-axis labels
fig.update_xaxes(title_text="Years", row=1, col=1)
fig.update_xaxes(title_text="Month", ticktext=months, tickvals=crashes_month.
 ↪index, row=2, col=1)

# Update y-axis labels
fig.update_yaxes(title_text="Crashes", row=1, col=1)
fig.update_yaxes(title_text="Crashes", row=2, col=1)

# Show the plot
fig.show()
```

[794]:
```
pd_fata.sort_values(by='Fatalities',ascending=False)
```

```
pd_fata = pd_frame.groupby('Heure')[['Fatalities']].count().
 ↪sort_values(by='Fatalities')
pd0_sorted = pd_fata.sort_values(by='Heure')

fig = px.bar(x=pd0_sorted.index, y=pd0_sorted['Fatalities'],␣
 ↪color=pd0_sorted['Fatalities'],
             labels={'x': 'Heure', 'y': 'Number of Fatalities','color':'nbrs'},
             title='Number of Fatalities by Country',
             color_continuous_scale=px.colors.sequential.Viridis)

fig.update_layout(height=700,xaxis_tickangle=-35,␣
 ↪xaxis=dict(tickfont=dict(size=20)), yaxis=dict(title=dict(text='Number of␣
 ↪Fatalities', font=dict(size=15))))
fig.show()
```

Insights :

- Le nombre d'accidents a augmenté progressivement à partir de 1908.
- Après le progrès du pilote automatique et quelques autres innovations majeures de l'avion, le nombre a commencé à diminuer après 1972.
- le mois de décembre ont plus de crashs avec plus de 500+ crashs.
- Les mois de January et August ont plus aussi de crashs.

[911]:
```
aboard_fatalities_new = pd_frame.pivot_table(values=["Survivors",␣
 ↪"Fatalities"], index="Year", aggfunc=np.sum)

fig = go.Figure()

fig.add_trace(go.Bar(
    x=aboard_fatalities_new.index,
    y=aboard_fatalities_new['Survivors'],
    name='Survivors',
    marker_color='green'
))

fig.add_trace(go.Bar(
    x=aboard_fatalities_new.index,
    y=aboard_fatalities_new['Fatalities'],
    name='Fatalities',
    marker_color='red'
))

fig.update_layout(
    barmode='stack',
    title="Fatalities and Survivors over the years",
    xaxis_title="Years",
    yaxis_title="Count",
    width=1200,
    height=800
)

fig.show()
```

```
C:\Users\Paname\AppData\Local\Temp\ipykernel_15580\17761351.py:1: FutureWarning:

The provided callable <function sum at 0x0000021C34F12D30> is currently using
DataFrameGroupBy.sum. In a future version of pandas, the provided callable will
be used directly. To keep current behavior pass the string "sum" instead.
```

Analyses by Operator :

[796]:
```
# Groupez les données par 'Operator' et agrégez les heures de vol
grouped_data = pd_frame.groupby('Operator')['Route'].agg(list)

# Affichez les résultats
print(grouped_data)
```

```
Operator
A B Aerotransport                                  [unavailable, Malmo -
Amsterdam]
AB Aerotransport                 [Istanbul-Athens-Rome-Geneve-Copenhagen-
Stockh…
ACES Colombia                    [Bogota - Saravena, Medellin - Bahia Solano,
u…
ADC Airlines                             [Lagos - Abuja - Sokoto, Lagos -
Calabar]
ADES Colombia                    [Mitu - Villavicencio, Villavicencio -
Miraflo…
                                                       …
Zantop Air Transport             [Cleveland - Detroit - Denver, Detroit, MI -
K…
Zantop Airways                                        [Detroit, MI - Lexington,
KY]
Zantop International Airlines                                 [Baltimore -
Detroit]
Zen Nippon                                                         [Osaka -
Tokyo]
de Havilland Aircraft
[unavailable]
Name: Route, Length: 2464, dtype: object
```

[797]:
```
operator_name = 'de Havilland Aircraft'
operator_values = grouped_data.loc[operator_name]
print(f"les routes de vol pour l'opérateur {operator_name} sont :")
print(operator_values)
```

```
les routes de vol pour l'opérateur de Havilland Aircraft sont :
['unavailable']
```

[798]:
```
# Groupez les données par 'Operator' et vérifiez si tous les éléments de␣
↪'heure' sont nuls
operators_with_null_time = pd_frame.groupby('Operator')['Heure'].apply(lambda x:
↪ x.isnull().all())
operators_all_null_time = operators_with_null_time[operators_with_null_time].
↪index
pd_frame['Heure'].fillna(0,inplace=True)
pd_frame['Heure'].dropna()
```

[798]:
```
2100    00
4729    00
1141    00
1076    00
2069    00
        ..
5153    09
5166    09
5172    09
5211    09
5267    09
Name: Heure, Length: 5172, dtype: object
```

[799]:
```
top_operators = pd_frame["Operator"].value_counts().head(30)

fig = px.bar(x=top_operators.index, y=top_operators.values, color=top_operators.
↪values,
             labels={'x': 'Operators', 'y': 'Fatalities','color':'nbrs'},
             title='Operators with highest number of Fatalities',
             color_continuous_scale=px.colors.sequential.Viridis)

fig.update_layout(height=650,xaxis_tickangle=-45,␣
↪xaxis=dict(tickfont=dict(size=12)), yaxis=dict(title=dict(text='fatalities',␣
↪font=dict(size=20))))
fig.show()
```

[800]:
```
top_aircraft_types = pd_frame["Type"].value_counts().head(30)

fig = px.bar(x=top_aircraft_types.index, y=top_aircraft_types.values,␣
↪color=top_aircraft_types.values,
             labels={'x': 'Type', 'y': 'Crashes','color':'nbrs'},
             title='Air Craft Type with highest number of Fatalities',
             color_continuous_scale=px.colors.sequential.Viridis,
            )

fig.update_layout(height=650,xaxis_tickangle=-45,␣
↪xaxis=dict(tickfont=dict(size=12)), yaxis=dict(title=dict(text='fatalities',␣
↪font=dict(size=20))))
fig.show()
```

Insights :

0.1.9 Conclusion :

- How many air crashes occur in the world on average each year?
- Which year had the most air crashes in the world?
- Which Type had the most air crashes in the world?
- Which Operator had the most air crashes in the world?
- Which Time had the most air crashes in the world?
- Which Month had the most air crashes in the world?
- which country has reported the most fatilities due to air crashes?