submission-aviation-project

September 27, 2024

[6]:
```
#load necessary files
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

[7]:
```
'''The input data is in the form of csv files.
  the code converts the csv file into a dataframe
   the output is a df containing the data from the csv file
'''
#load data
aviation_df=pd.read_csv('/content/AviationData.csv',encoding='ISO-8859-1')
UsStatesCodes_df=pd.read_csv('/content/USState_Codes.csv')
```

```
<ipython-input-7-d9a32f69423f>:2: DtypeWarning: Columns (6,7,28) have mixed
types. Specify dtype option on import or set low_memory=False.
  aviation_df=pd.read_csv('/content/AviationData.csv',encoding='ISO-8859-1')
```

[8]:
```
#view data frame
aviation_df
```

[8]:

	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]
```

[9]:
```
UsStatesCodes_df
```

[9]:
```
          US_State Abbreviation
0          Alabama           AL
1           Alaska           AK
2          Arizona           AZ
3         Arkansas           AR
4       California           CA
..             …            …
57  Virgin Islands           VI
58   Washington_DC           DC
59  Gulf of mexico           GM
60  Atlantic ocean           AO
61   Pacific ocean           PO

[62 rows x 2 columns]
```

[10]:
```
'''the code accept the df as input
   The code displays the data type and number of column together with datatype
    The output is a summary of the dataframe
'''
#display info about the dataframes
aviation_df.info()
UsStatesCodes_df.info()
```

```
<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
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 62 entries, 0 to 61
Data columns (total 2 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   US_State      62 non-null     object
 1   Abbreviation  62 non-null     object
dtypes: object(2)
memory usage: 1.1+ KB
```

[11]:
```
#shows the number of columns and rows
aviation_df.shape
```

[11]: (88889, 31)

[12]:
```
UsStatesCodes_df.shape
```

[12]:
```
(62, 2)
```

[105]:
```
'''This code take the aviation_df as the input
   It calculate the mean,values of distribution and changes the columns to rows
   The output is a summary of aggregate values of the dataframe
'''
#aggregation
aviation_df.describe().transpose()
```

[105]:

	count	mean	std	min	25%	50%	75%	max
Number.of.Engines	82805.0	1.146585	0.446510	0.0	1.0	1.0	1.0	8.0
Total.Fatal.Injuries	77488.0	0.647855	5.485960	0.0	0.0	0.0	0.0	349.0
Total.Serious.Injuries	76379.0	0.279881	1.544084	0.0	0.0	0.0	0.0	161.0
Total.Minor.Injuries	76956.0	0.357061	2.235625	0.0	0.0	0.0	0.0	380.0
Total.Uninjured	82977.0	5.325440	27.913634	0.0	0.0	1.0	2.0	699.0

[14]:
```
###DATA PREPARATION AND CLEANING
```

[107]:
```
'''The code has the dataframe as its input
   it checks for missing values(NAN values) and counts them
   The output is a sum of missing values arrange in ascending order
'''
# Check for missing values
missing_data = aviation_df.isnull().sum().sort_values(ascending=False)
print(missing_data)
```

```
Airport.Code              37176
Airport.Name              35352
Broad.phase.of.flight     27165
Publication.Date          13771
Total.Serious.Injuries    12510
Total.Minor.Injuries      11933
Total.Fatal.Injuries      11401
Engine.Type                7096
Report.Status              6384
Purpose.of.flight          6192
Number.of.Engines          6084
Total.Uninjured            5912
```

```
Aircraft.Category          4601
Weather.Condition          4492
Aircraft.damage            3194
Registration.Number        1382
Injury.Severity            1000
Abbreviation                622
Country                     226
Amateur.Built               102
Model                        92
Make                         63
City                         52
Investigation.Type            0
Event.Date                    0
Accident.Number               0
Event.Id                      0
dtype: int64
```

[16]:
```
'''The code has the dataframe as its input
   it checks for duplicate values and counts them
   The output is a sum of duplicate values
   '''
#checking for duplicate values
aviation_df.duplicated().sum()
```

[16]:
```
0
```

[17]:
```
#fill in missing values
'''This code take the dataframe as its input
   The code  compares the value in each relating row(Airport.Name,Airport.code)␣
 ↪comparing them to each other and filling in the missing values for each␣
 ↪column based on matching values
   The output is a dataframe with filled missing values
'''
#filling missing values
#Group by 'Airport.Name' to find the most common 'Airport.Code' for each airport
airport_code_map =aviation_df.groupby('Airport.Name')['Airport.Code'].
 ↪agg(lambda x: x.mode().iloc[0] if not x.mode().empty else None)

#  Define a function to map and fill missing values in 'Airport.Code'
def fill_airport_code(row):
    if pd.isnull(row['Airport.Code']) and not pd.isnull(row['Airport.Name']):
        return airport_code_map.get(row['Airport.Name'], None)
    else:
        return row['Airport.Code']

# Apply the function to fill missing 'Airport.Code'
aviation_df['Airport.Code'] = aviation_df.apply(fill_airport_code, axis=1)
```

```
# Handle missing values in 'Airport.Name' based on 'Airport.Code'
# Similar logic to fill missing 'Airport.Name' using 'Airport.Code' if available
airport_name_map = aviation_df.groupby('Airport.Code')['Airport.Name'].
 ↪agg(lambda x: x.mode().iloc[0] if not x.mode().empty else None)

def fill_airport_name(row):
    if pd.isnull(row['Airport.Name']) and not pd.isnull(row['Airport.Code']):
        return airport_name_map.get(row['Airport.Code'], None)
    else:
        return row['Airport.Name']

#  Apply the function to fill missing 'Airport.Name'
aviation_df['Airport.Name'] = aviation_df.apply(fill_airport_name, axis=1)
aviation_df.head()
```

[17]:

	Event.Id	Investigation.Type	Accident.Number	Event.Date	Location	Country	Latitude	Longitude	Airport.Code	Airport.Name	…	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
0	20001218X45444	Accident	SEA87LA080	1948-10-24	MOOSE CREEK, ID	United States	NaN	NaN	NaN	NaN	…	Personal	NaN	2.0	0.0	0.0	0.0	UNK	Cruise	Probable Cause	NaN
1	20001218X45447	Accident	LAX94LA336	1962-07-19	BRIDGEPORT, CA	United States	NaN	NaN	NaN	NaN	…	Personal	NaN	4.0	0.0	0.0	0.0	UNK	Unknown	Probable Cause	19-09-1996
2	20061025X01555	Accident	NYC07LA005	1974-08-30	Saltville, VA	United States	36.922223	-81.878056	NaN	NaN	…	Personal	NaN	3.0	NaN	NaN	NaN				
3	20001218X45448	Accident	LAX96LA321	1977-06-19	EUREKA, CA	United States	NaN	NaN	NaN	NaN	…	Personal	NaN	2.0	0.0	0.0	0.0				
4	20041105X01764	Accident	CHI79FA064	1979-08-02	Canton, OH	United States	NaN	NaN	NaN	NaN	…	Personal	NaN	1.0	2.0	NaN	0.0				

```
2              IMC               Cruise  Probable Cause       26-02-2007
3              IMC               Cruise  Probable Cause       12-09-2000
4              VMC             Approach  Probable Cause       16-04-1980

[5 rows x 31 columns]
```

[18]:
```
'''This code take the dataframe as its input
   The code  compares the value in each relating column and row(Make,Airport␣
 ↪category)comparing them to each other and filling in the missing values for␣
 ↪each column based on matching values
   The output is a dataframe with filled missing values
'''
#filling missing values
# Group by 'Make' to find the most common (mode) 'Aircraft.Category' for each␣
 ↪aircraft make
make_category_map = aviation_df.groupby('Make')['Aircraft.Category'].agg(lambda␣
 ↪x: x.mode().iloc[0] if not x.mode().empty else None)

#  Define a function to map and fill missing values in 'Aircraft.Category', but␣
 ↪only if 'Make' is not NaN
def fill_category(row):
    if pd.isnull(row['Aircraft.Category']) and not pd.isnull(row['Make']):
        return make_category_map.get(row['Make'], None)
    else:
        return row['Aircraft.Category']

# Apply the function to fill missing Aircraft.Category
aviation_df['Aircraft.Category'] = aviation_df.apply(fill_category, axis=1)
aviation_df.head()
```

[18]:
```
         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

          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

  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
3          NaN  …          Personal         NaN                    2.0
4          NaN  …          Personal         NaN                    1.0

  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
3                    0.0                  0.0             0.0
4                    2.0                  NaN             0.0

  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               IMC                 Cruise  Probable Cause       12-09-2000
4               VMC               Approach  Probable Cause       16-04-1980

[5 rows x 31 columns]
```

[19]:
```
#checking for improvements
missing_data = aviation_df.isnull().sum().sort_values(ascending=False)
print(missing_data)
```

```
Schedule                 76307
Air.carrier              72241
FAR.Description          56866
Longitude                54516
Latitude                 54507
Airport.Code             37176
Airport.Name             35352
Broad.phase.of.flight    27165
Publication.Date         13771
Total.Serious.Injuries   12510
Total.Minor.Injuries     11933
Total.Fatal.Injuries     11401
Engine.Type               7096
Report.Status             6384
Purpose.of.flight         6192
Number.of.Engines         6084
Total.Uninjured           5912
Aircraft.Category         4601
Weather.Condition         4492
Aircraft.damage           3194
Registration.Number       1382
Injury.Severity           1000
Country                    226
```

```
Amateur.Built               102
Model                        92
Make                         63
Location                     52
Investigation.Type            0
Event.Date                    0
Accident.Number               0
Event.Id                      0
dtype: int64
```

[20]:
```
'''This code take the dataframe as its input
   The code  deletes the columns that are not needed and seen to have most Nan␣
 ↪valurs
    The output is a dataframe with dropped columns
'''
#Dropping columns that have the most null values
columns=['Schedule','Air.carrier','FAR.Description','Longitude','Latitude']
aviation_df=aviation_df.drop(columns,axis=1)
aviation_df.head()
```

[20]:

	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	

	Location	Country	Airport.Code	Airport.Name	Injury.Severity	\
0	MOOSE CREEK, ID	United States	NaN	NaN	Fatal(2)	
1	BRIDGEPORT, CA	United States	NaN	NaN	Fatal(4)	
2	Saltville, VA	United States	NaN	NaN	Fatal(3)	
3	EUREKA, CA	United States	NaN	NaN	Fatal(2)	
4	Canton, OH	United States	NaN	NaN	Fatal(1)	

	Aircraft.damage	…	Engine.Type	Purpose.of.flight	Total.Fatal.Injuries	\
0	Destroyed	…	Reciprocating	Personal	2.0	
1	Destroyed	…	Reciprocating	Personal	4.0	
2	Destroyed	…	Reciprocating	Personal	3.0	
3	Destroyed	…	Reciprocating	Personal	2.0	
4	Destroyed	…	NaN	Personal	1.0	

	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	
3	0.0	0.0	0.0	
4	2.0	NaN	0.0	

```
  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               IMC                Cruise  Probable Cause        12-09-2000
4               VMC              Approach  Probable Cause        16-04-1980

[5 rows x 26 columns]
```

[21]:
```
'''The code accepts  location column as its dataset
  The code separate the location column into two making a new abbreviation␣
 ↪column
   the output is a new column abbreviation and a new column location
'''

# Split the 'Location' column into two new columns: 'City' and 'State.
 ↪Abbreviation'
# Using `n=1` to limit split to at most two parts and `expand=True` to get␣
 ↪separate columns
aviation_df[['City', 'Abbreviation']] = aviation_df['Location'].str.split(',␣
 ↪',n=1,expand=True)
#drop old location column
aviation_df.drop('Location', axis=1, inplace=True)
#view new dataset
aviation_df.head()
```

[21]:
```
         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

         Country Airport.Code Airport.Name Injury.Severity Aircraft.damage  \
0  United States          NaN          NaN        Fatal(2)       Destroyed
1  United States          NaN          NaN        Fatal(4)       Destroyed
2  United States          NaN          NaN        Fatal(3)       Destroyed
3  United States          NaN          NaN        Fatal(2)       Destroyed
4  United States          NaN          NaN        Fatal(1)       Destroyed

  Aircraft.Category  … Total.Fatal.Injuries Total.Serious.Injuries  \
0          Airplane  …                  2.0                    0.0
1          Airplane  …                  4.0                    0.0
2          Airplane  …                  3.0                    NaN
3          Airplane  …                  2.0                    0.0
4          Airplane  …                  1.0                    2.0
```

```
  Total.Minor.Injuries Total.Uninjured  Weather.Condition  \
0                  0.0             0.0                UNK
1                  0.0             0.0                UNK
2                  NaN             NaN                IMC
3                  0.0             0.0                IMC
4                  NaN             0.0                VMC

  Broad.phase.of.flight   Report.Status  Publication.Date         City  \
0               Cruise  Probable Cause               NaN  MOOSE CREEK
1              Unknown  Probable Cause        19-09-1996   BRIDGEPORT
2               Cruise  Probable Cause        26-02-2007    Saltville
3               Cruise  Probable Cause        12-09-2000       EUREKA
4             Approach  Probable Cause        16-04-1980       Canton

   Abbreviation
0            ID
1            CA
2            VA
3            CA
4            OH

[5 rows x 27 columns]
```

[22]:
```
'''The code take the Aviation.df and Usstactecodes.df as its input dataset
    The code  merges the 2 dataframes on the Abbreviations columns,preseving␣
 ↪the 1st df and matching columns on the 2nd df
    The output is a merged dataframe
'''
# Merging the DataFrames on 'Country_Code'
merged_df = pd.merge(aviation_df,UsStatesCodes_df, on='Abbreviation',␣
 ↪how='left')

# Show the merged DataFrame
merged_df.head(10)
```

[22]:
```
         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
5  20170710X52551           Accident      NYC79AA106  1979-09-17
6  20001218X45446           Accident      CHI81LA106  1981-08-01
7  20020909X01562           Accident      SEA82DA022  1982-01-01
8  20020909X01561           Accident      NYC82DA015  1982-01-01
9  20020909X01560           Accident      MIA82DA029  1982-01-01
```

```
         Country Airport.Code        Airport.Name Injury.Severity  \
0  United States          NaN                 NaN        Fatal(2)
1  United States          NaN                 NaN        Fatal(4)
2  United States          NaN                 NaN        Fatal(3)
3  United States          NaN                 NaN        Fatal(2)
4  United States          NaN                 NaN        Fatal(1)
5  United States          NaN                 NaN       Non-Fatal
6  United States          NaN                 NaN        Fatal(4)
7  United States         None  BLACKBURN AG STRIP       Non-Fatal
8  United States          N58             HANOVER       Non-Fatal
9  United States          JAX   JACKSONVILLE INTL       Non-Fatal

  Aircraft.damage Aircraft.Category  … Total.Serious.Injuries  \
0       Destroyed          Airplane  …                   0.0
1       Destroyed          Airplane  …                   0.0
2       Destroyed          Airplane  …                   NaN
3       Destroyed          Airplane  …                   0.0
4       Destroyed          Airplane  …                   2.0
5     Substantial          Airplane  …                   NaN
6       Destroyed          Airplane  …                   0.0
7     Substantial          Airplane  …                   0.0
8     Substantial          Airplane  …                   0.0
9     Substantial          Airplane  …                   0.0

  Total.Minor.Injuries Total.Uninjured Weather.Condition  \
0                  0.0             0.0               UNK
1                  0.0             0.0               UNK
2                  NaN             NaN               IMC
3                  0.0             0.0               IMC
4                  NaN             0.0               VMC
5                  1.0            44.0               VMC
6                  0.0             0.0               IMC
7                  0.0             2.0               VMC
8                  0.0             2.0               IMC
9                  3.0             0.0               IMC

   Broad.phase.of.flight   Report.Status Publication.Date          City  \
0                 Cruise  Probable Cause              NaN   MOOSE CREEK
1                Unknown  Probable Cause       19-09-1996    BRIDGEPORT
2                 Cruise  Probable Cause       26-02-2007     Saltville
3                 Cruise  Probable Cause       12-09-2000        EUREKA
4               Approach  Probable Cause       16-04-1980        Canton
5                  Climb  Probable Cause       19-09-2017        BOSTON
6                Unknown  Probable Cause       06-11-2001        COTTON
7                Takeoff  Probable Cause       01-01-1982       PULLMAN
8                Landing  Probable Cause       01-01-1982  EAST HANOVER
```

```
9                 Cruise  Probable Cause       01-01-1982  JACKSONVILLE

   Abbreviation       US_State
0            ID          Idaho
1            CA     California
2            VA       Virginia
3            CA     California
4            OH           Ohio
5            MA  Massachusetts
6            MN      Minnesota
7            WA     Washington
8            NJ     New Jersey
9            FL        Florida

[10 rows x 28 columns]
```

[23]:
```
'''This code take the dataframe as its input
   The code  compares the value in each relating column and row(Make,Airport␣
 ↪category)comparing them to each other and filling in the missing values for␣
 ↪each column based on matching values
   The output is a dataframe with filled missing values
'''
#replace all Nan values with unknown for the non-numeric columns
non_numeric_cols = merged_df.select_dtypes(include=['object']).columns
merged_df[non_numeric_cols] =merged_df[non_numeric_cols].fillna('Unknown')
merged_df.fillna('Unknown', inplace=True)
merged_df.head()
```

```
<ipython-input-23-5bae4783aa93>:4: FutureWarning: Setting an item of
incompatible dtype is deprecated and will raise in a future error of pandas.
Value 'Unknown' has dtype incompatible with float64, please explicitly cast to a
compatible dtype first.
  merged_df.fillna('Unknown', inplace=True)
```

[23]:
```
         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

         Country Airport.Code Airport.Name Injury.Severity Aircraft.damage  \
0  United States      Unknown      Unknown        Fatal(2)       Destroyed
1  United States      Unknown      Unknown        Fatal(4)       Destroyed
2  United States      Unknown      Unknown        Fatal(3)       Destroyed
3  United States      Unknown      Unknown        Fatal(2)       Destroyed
4  United States      Unknown      Unknown        Fatal(1)       Destroyed
```

```
  Aircraft.Category  … Total.Serious.Injuries Total.Minor.Injuries  \
0          Airplane  …                   0.0                   0.0
1          Airplane  …                   0.0                   0.0
2          Airplane  …                Unknown               Unknown
3          Airplane  …                   0.0                   0.0
4          Airplane  …                   2.0               Unknown

  Total.Uninjured Weather.Condition Broad.phase.of.flight   Report.Status  \
0             0.0               UNK                Cruise  Probable Cause
1             0.0               UNK               Unknown  Probable Cause
2         Unknown               IMC                Cruise  Probable Cause
3             0.0               IMC                Cruise  Probable Cause
4             0.0               VMC              Approach  Probable Cause

  Publication.Date         City Abbreviation    US_State
0          Unknown  MOOSE CREEK           ID       Idaho
1       19-09-1996   BRIDGEPORT           CA  California
2       26-02-2007    Saltville           VA    Virginia
3       12-09-2000       EUREKA           CA  California
4       16-04-1980       Canton           OH        Ohio

[5 rows x 28 columns]
```

[24]:
```
'''This code takes the column Event.date and Publication.date as its input
   The code  converts the data to datetime format datatype and displays a␣
 ↪summary of the dataframe
   The output is a dataframe with the columns in datetime(64)as dtype format
'''
#CONVERT THE DATE TIME FORMATS
merged_df['Event.Date'] = pd.to_datetime(merged_df['Event.Date'],␣
 ↪errors='coerce')
merged_df['Publication.Date'] = pd.to_datetime(merged_df['Publication.Date'],␣
 ↪errors='coerce')

merged_df.head()
merged_df.info()
merged_df.shape
```

```
<ipython-input-24-e66a3ccb850c>:3: UserWarning: Could not infer format, so each
element will be parsed individually, falling back to `dateutil`. To ensure
parsing is consistent and as-expected, please specify a format.
  merged_df['Publication.Date'] = pd.to_datetime(merged_df['Publication.Date'],
errors='coerce')

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 88889 entries, 0 to 88888
Data columns (total 28 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  datetime64[ns]
 4   Country                 88889 non-null  object
 5   Airport.Code            88889 non-null  object
 6   Airport.Name            88889 non-null  object
 7   Injury.Severity         88889 non-null  object
 8   Aircraft.damage         88889 non-null  object
 9   Aircraft.Category       88889 non-null  object
 10  Registration.Number     88889 non-null  object
 11  Make                    88889 non-null  object
 12  Model                   88889 non-null  object
 13  Amateur.Built           88889 non-null  object
 14  Number.of.Engines       88889 non-null  object
 15  Engine.Type             88889 non-null  object
 16  Purpose.of.flight       88889 non-null  object
 17  Total.Fatal.Injuries    88889 non-null  object
 18  Total.Serious.Injuries  88889 non-null  object
 19  Total.Minor.Injuries    88889 non-null  object
 20  Total.Uninjured         88889 non-null  object
 21  Weather.Condition       88889 non-null  object
 22  Broad.phase.of.flight   88889 non-null  object
 23  Report.Status           88889 non-null  object
 24  Publication.Date        75118 non-null  datetime64[ns]
 25  City                    88889 non-null  object
 26  Abbreviation            88889 non-null  object
 27  US_State                88889 non-null  object
dtypes: datetime64[ns](2), object(26)
memory usage: 19.0+ MB
```

[24]:
```
(88889, 28)
```

[25]:
```
#check data quality in the merged data frame
missing_data =merged_df.isnull().sum().sort_values(ascending=False)
print(missing_data)
```

```
Publication.Date          13771
Event.Id                      0
Investigation.Type            0
Abbreviation                  0
City                          0
Report.Status                 0
Broad.phase.of.flight         0
Weather.Condition             0
Total.Uninjured               0
```

```
Total.Minor.Injuries          0
Total.Serious.Injuries        0
Total.Fatal.Injuries          0
Purpose.of.flight             0
Engine.Type                   0
Number.of.Engines             0
Amateur.Built                 0
Model                         0
Make                          0
Registration.Number           0
Aircraft.Category             0
Aircraft.damage               0
Injury.Severity               0
Airport.Name                  0
Airport.Code                  0
Country                       0
Event.Date                    0
Accident.Number               0
US_State                      0
dtype: int64
```

[26]:
```
'''This code takes  the  datetime columns as its input
  The code  converts the columns to object datatype and checks for nan values
  replacing them with unknown
   The output is a df with the nan values of date time columns converted to␣
 ↪Unknown
'''
# Step 1: Identify datetime columns
datetime_cols = merged_df.select_dtypes(include=['datetime64[ns]']).columns

# Step 2: Convert datetime columns to object type
merged_df[datetime_cols] = merged_df[datetime_cols].astype(object)

# Step 3: Replace NaN values in datetime columns with 'Unknown'
for col in datetime_cols:
    merged_df[col] = merged_df[col].fillna('Unknown')

merged_df.head()
```

[26]:
```
         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

         Country Airport.Code Airport.Name Injury.Severity Aircraft.damage  \
```

```
0  United States      Unknown      Unknown        Fatal(2)       Destroyed
1  United States      Unknown      Unknown        Fatal(4)       Destroyed
2  United States      Unknown      Unknown        Fatal(3)       Destroyed
3  United States      Unknown      Unknown        Fatal(2)       Destroyed
4  United States      Unknown      Unknown        Fatal(1)       Destroyed

  Aircraft.Category  … Total.Serious.Injuries Total.Minor.Injuries  \
0          Airplane  …                    0.0                  0.0
1          Airplane  …                    0.0                  0.0
2          Airplane  …                Unknown              Unknown
3          Airplane  …                    0.0                  0.0
4          Airplane  …                    2.0              Unknown

  Total.Uninjured Weather.Condition Broad.phase.of.flight   Report.Status  \
0             0.0               UNK                Cruise  Probable Cause
1             0.0               UNK               Unknown  Probable Cause
2         Unknown               IMC                Cruise  Probable Cause
3             0.0               IMC                Cruise  Probable Cause
4             0.0               VMC              Approach  Probable Cause

      Publication.Date         City Abbreviation    US_State
0              Unknown  MOOSE CREEK            ID       Idaho
1  1996-09-19 00:00:00   BRIDGEPORT            CA  California
2  2007-02-26 00:00:00    Saltville            VA    Virginia
3  2000-12-09 00:00:00       EUREKA            CA  California
4  1980-04-16 00:00:00       Canton            OH        Ohio

[5 rows x 28 columns]
```

[27]:
```
''''This code takes the merged dataframe as its input
    The code checks and removes trailing whitespaces and newlines
    The output is a cleaned merged dataframe with no extra number of rows or␣
 ↪columns
  '''



#Check on merged_df info
merged_df.info()
merged_df.shape
#removes whitespaces or newlines
merged_df = merged_df.apply(lambda x: x.str.strip() if x.dtype == "object" else␣
 ↪x)
#removes any empty row or null values
merged_df.dropna(how='all', inplace=True)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 88889 entries, 0 to 88888
```

```
Data columns (total 28 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  datetime64[ns]
 4   Country                 88889 non-null  object
 5   Airport.Code            88889 non-null  object
 6   Airport.Name            88889 non-null  object
 7   Injury.Severity         88889 non-null  object
 8   Aircraft.damage         88889 non-null  object
 9   Aircraft.Category       88889 non-null  object
 10  Registration.Number     88889 non-null  object
 11  Make                    88889 non-null  object
 12  Model                   88889 non-null  object
 13  Amateur.Built           88889 non-null  object
 14  Number.of.Engines       88889 non-null  object
 15  Engine.Type             88889 non-null  object
 16  Purpose.of.flight       88889 non-null  object
 17  Total.Fatal.Injuries    88889 non-null  object
 18  Total.Serious.Injuries  88889 non-null  object
 19  Total.Minor.Injuries    88889 non-null  object
 20  Total.Uninjured         88889 non-null  object
 21  Weather.Condition       88889 non-null  object
 22  Broad.phase.of.flight   88889 non-null  object
 23  Report.Status           88889 non-null  object
 24  Publication.Date        88889 non-null  object
 25  City                    88889 non-null  object
 26  Abbreviation            88889 non-null  object
 27  US_State                88889 non-null  object
dtypes: datetime64[ns](1), object(27)
memory usage: 19.0+ MB
```

[28]:
```
'''This code takes the merged dataframe as its input
    The code converts the merged df to csv format and  downloads the data
    The output is a csv file with the merged dataframe data
  '''
#convert to csv for visualization
merged_df.to_csv('Cleaned_aviation_data', index=False)
#download merged dataset as updated_aviation_data.csv
from google.colab import files
files.download('/content/Cleaned_aviation_data')
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.Javascript object>
```

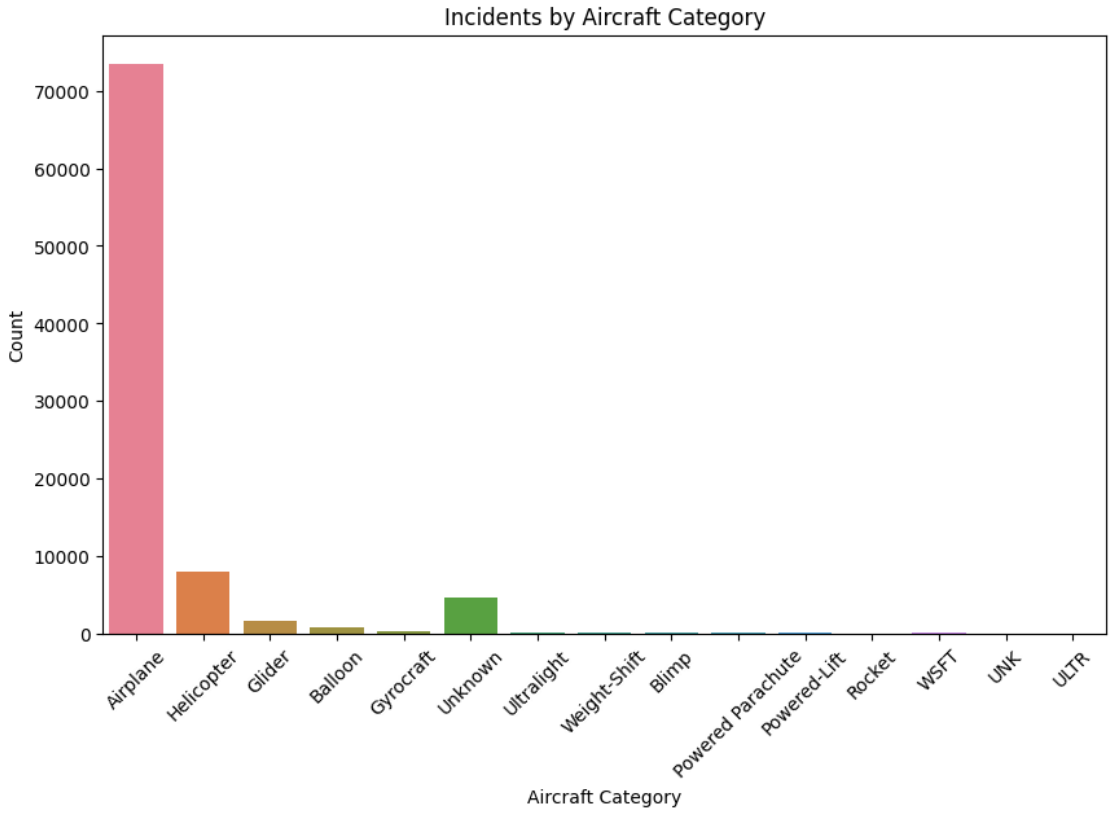
[29]:
```
df=pd.read_csv('/content/Cleaned_aviation_data')
df.head()
df.shape
```

```
<ipython-input-29-52b1635a79b9>:1: DtypeWarning: Columns (17,18,19,20) have
mixed types. Specify dtype option on import or set low_memory=False.
  df=pd.read_csv('/content/Cleaned_aviation_data')
```

[29]:
```
(88889, 28)
```

[30]:
```
###VISUALIZATION
```

[38]:
```
'''The code takes the new csv df as its dataset
   The code generates a count plot to visualize the number of incidents for␣
 ↪each aircraft category, displaying the counts on the y-axis and labeling the␣
 ↪axes and title appropriately.
   The output is a count plot showing the number of incidents for each aircraft␣
 ↪category
'''
#plot showing number of accidents   per category
plt.figure(figsize=(10,6))
sns.countplot(x='Aircraft.Category', data=df, hue='Aircraft.Category')
plt.title('Incidents by Aircraft Category')
plt.xlabel('Aircraft Category')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.show()
```

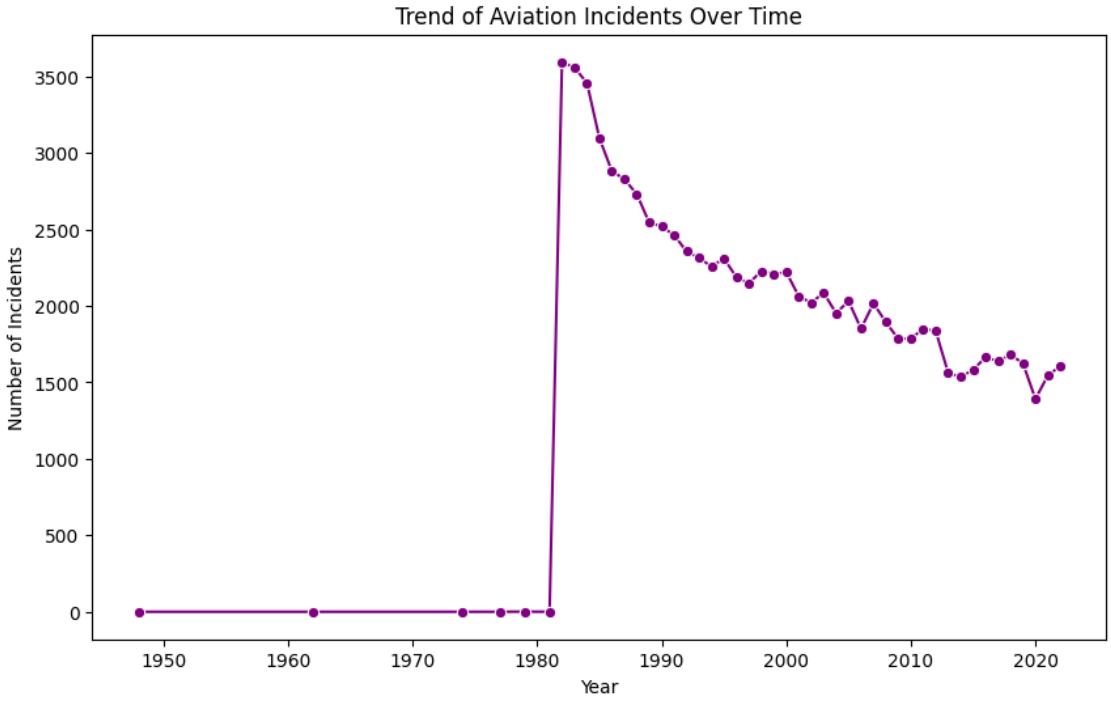


[40]:
```
'''The code takes the dataframe as its input dataset
   The code converts the `Event.Date` column to a datetime format, extracts the␣
 ↪year from the dates, groups the data by year to count incidents, and then␣
 ↪creates a line plot to visualize the trend of aviation incidents over time
   The output is a line plot showing the trend of aviation incidents over time
'''
#Plot of trend of incidents over time
# Convert 'Event.Date' to datetime
df['Event.Date'] = pd.to_datetime(df['Event.Date'])

# Group data by year
df['Year'] = df['Event.Date'].dt.year
incidents_per_year = df.groupby('Year').size()

# Line plot for incidents over the years
plt.figure(figsize=(10,6))
sns.lineplot(x=incidents_per_year.index, y=incidents_per_year.values,␣
 ↪marker='o', color='purple')
plt.title('Trend of Aviation Incidents Over Time')
plt.xlabel('Year')
plt.ylabel('Number of Incidents')
```

```
plt.show()
```


Trend of Aviation Incidents Over Time
Number of Incidents
Year
0
500
1000
1500
2000
2500
3000
3500
1950
1960
1970
1980
1990
2000
2010
2020