Flight Data Analysis Pipeline Documentation

Overview:

This Document provides all the guidelines to set up the project with executing and creating a basic understanding of ‘Flight Data Analysis Pipeline’. The Pipeline includes data loading, cleaning, analysis, visualization, Insights,recommendations based on analysis and storage in an SQLite database.

NOTE: I will be sharing guidelines for running the project in Vs Code as well as Google Colab.

Requirements:

Following libraries will be needed:

* pandas
* numpy
* matplotlib
* seaborn
* sqlalchemy

**NOTE**: In the below code,there are **three main DataFrames**:

1. **df\_flight**: This is the original DataFrame loaded from the CSV file containing the aviation data. It is the starting point for data cleaning and transformation.
2. **df\_unique\_flights**: This DataFrame is created by dropping duplicate entries from df\_flight, specifically keeping unique combinations of FlightNumber and DepartureDate.
3. **new\_df\_flights**: This DataFrame is created by further processing df\_unique\_flights to combine date and time information, handle arrival times, and create new columns like FlightDuration.
4. **average\_delay**: This DataFrame is created by grouping new\_df\_flights by Airline to calculate the average delay for each airline.
5. **delay\_analysis**: This DataFrame is created to analyze delay times grouped by DepartureHour.

So, in total, there are **five DataFrames**: df\_flight, df\_unique\_flights, new\_df\_flights, average\_delay, and delay\_analysis.

**Pipeline Steps**

* **FIRSTLY LETS START WITH VS CODE**

**Step 1.** Open Vs Code and go to views->terminal,after the terminal gets opened you can install the libraries with a simple code:

”pip install pandas numpy matplotlib seaborn sqlalchemy”

**Step 2.** Load the flight data from a CSV file using Pandas:

“df\_flight = pd.read\_csv('C:/Users/palak/OneDrive/Desktop/aviation\_data.csv')”

**Step 3**. **Data Cleaning** :

1. Checking the Original Date Format:

“print(df\_flight[['DepartureDate', 'ArrivalDate']].head())”

1. Converting the Dates to Proper Format:

“ df\_flight['DepartureDate'] = pd.to\_datetime(df\_flight['DepartureDate'], format='%m-%d-%Y', errors='coerce')

df\_flight['ArrivalDate'] = pd.to\_datetime(df\_flight['ArrivalDate'], format='%m-%d-%Y', errors='coerce') “

1. Converting Time to 24-hour Format:

“ def parse\_time(time\_str):

try:

return pd.to\_datetime(time\_str, format='%I:%M %p').time()

except ValueError:

return pd.to\_datetime(time\_str, format='%H:%M:%S').time

df\_flight['DepartureTime'] = df\_flight['DepartureTime'].apply(parse\_time)

df\_flight['ArrivalTime'] = df\_flight['ArrivalTime'].apply(parse\_time)”

1. Handling NaN Values in Delay Minutes:

“ df\_flight['DelayMinutes'] = df\_flight['DelayMinutes'].fillna(df\_flight['DelayMinutes'].median())”

e) # Checking unique flights:

unique\_flight\_dates = df\_flight[['FlightNumber', 'DepartureDate']].drop\_duplicates()

print(f'Total number of unique: {unique\_flight\_dates}')

1. Dropping the Duplicate Entries:

df\_unique\_flights = df\_flight.drop\_duplicates(subset=['FlightNumber', 'DepartureDate'])

print(f'Total number of unique entries: {len(df\_unique\_flights)}')

print(df\_unique\_flights)

**Step 4:** **Combine Date and Time:**

1. Combining Departure and Arrival Date with Time:

df\_unique\_flights['DepartureDateTime'] = pd.to\_datetime(

df\_unique\_flights['DepartureDate'].astype(str) + ' ' + df\_unique\_flights['DepartureTime'].astype(str)

)

df\_unique\_flights['ArrivalDateTime'] = pd.to\_datetime(

df\_unique\_flights['ArrivalDate'].astype(str) + ' ' + df\_unique\_flights['ArrivalTime'].astype(str)

2) Displaying the updated DataFrame

print(df\_unique\_flights[['FlightNumber', 'DepartureDateTime', 'ArrivalDateTime']])

1. Dropping unnecessary columns

new\_df\_flights = df\_unique\_flights.drop(['DepartureDate', 'DepartureTime', 'ArrivalDate', 'ArrivalTime'], axis=1)

1. Adjust Arrival Time as asked:

“ bool\_mask = new\_df\_flights['ArrivalDateTime'] < new\_df\_flights['DepartureDateTime']

new\_df\_flights.loc[bool\_mask, 'ArrivalDateTime'] += pd.Timedelta(days=1)

print(new\_df\_flights[['FlightNumber', 'DepartureDateTime', 'ArrivalDateTime']]) “

4) Creating a new column for FlightDuration

new\_df\_flights['FlightDuration'] = new\_df\_flights['ArrivalDateTime'] - new\_df\_flights['DepartureDateTime']

**Step 5: Data Analysis and Visualization:**

1. Analyzing the Distribution of delays:

“ print(new\_df\_flights['DelayMinutes'].describe()) “

1. Visualizing Delays by Airline:

“ plt.figure(figsize=(10, 6))

sns.boxplot(x='Airline', y='DelayMinutes', data= new\_df\_flights)

plt.title('Flight Delays by Airline')

plt.xlabel('Airline')

plt.ylabel('Delay (minutes)')

plt.show() “

1. Calculate Average Delay by Airline:

“average\_delay = df\_new\_flights.groupby('Airline')['DelayMinutes'].mean().reset\_index() “

#Also creating a new Departure Hour Collumn:

“ new\_df\_flights['DepartureHour'] = new\_df\_flights['DepartureDateTime'].dt.hour “

1. Grouping by departure hour

delay\_analysis = new\_df\_flights.groupby('DepartureHour')['DelayMinutes'].mean().reset\_index()

print(delay\_analysis)

1. Bar Chart for Average Delay:

“ plt.figure(figsize=(10, 6))

sns.barplot(x='Airline', y='DelayMinutes', data=average\_delay,palette='viridis')

plt.title('Average Delay by Airline')

plt.xlabel('Airline')

plt.ylabel('Average Delay (Minutes)')

plt.xticks(rotation=45)

plt.grid(axis='y')

plt.show() “

**Step 6: Save Cleaned Data**

1. Saving the Cleaned Data to a CSV File:

“ new\_df\_flights.to\_csv('C:/Users/palak/OneDrive/Desktop/cleaned\_aviation\_dataaa.csv', index=False)

print("Cleaned data saved as 'cleaned\_aviation\_dataaa.csv'.") “

**Step 7: Storing the data in SQLite Database**

1. Creating a Database Connection and Storing the Data:

engine=create\_engine('sqlite:///C:/Users/palak/OneDrive/Desktop/aviation\_data.db')

# Storing the cleaned data into the database

new\_df\_flights.to\_sql('flights', con=engine, if\_exists='replace', index=False)

print("Cleaned data saved to the database 'aviation\_data.db' in the 'flights' table.")

**Pipeline 2 Steps**

* **NOW LETS START WITH GOOGLE COLAB**

1.Open Google Colab:

* + Go to Google Colab.
  + It may be needed to sign in with your Google account.

2.Create a New Notebook:

* + Click on "File" -> "New Notebook".

3. Upload Your Data:

a) You can upload your aviation\_data.csv file directly to Colab:

“ from google.colab import files

uploaded = files.upload() “

b)Firstly we’ll import all the libraries :

“ import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from sqlalchemy import create\_engine “

c) Load the data

“ df\_flight = pd.read\_csv('aviation\_data.csv') “

d) For cleaning the data

“print(df\_flight[['DepartureDate', 'ArrivalDate']].head()) “

e) Converting departure date and arrival date into proper date format

“ df\_flight['DepartureDate'] = pd.to\_datetime(df\_flight['DepartureDate'], format='%m-%d-%Y', errors='coerce') df\_flight['ArrivalDate'] = pd.to\_datetime(df\_flight['ArrivalDate'], format='%m-%d-%Y', errors='coerce')”

1. Converting Departure time and Arrival time into 24-hour format

def parse\_time(time\_str):

try:

return pd.to\_datetime(time\_str, format='%I:%M %p').time()

except ValueError:

return pd.to\_datetime(time\_str, format='%H:%M:%S').time()

df\_flight['DepartureTime'] = df\_flight['DepartureTime'].apply(parse\_time) df\_flight['ArrivalTime'] = df\_flight['ArrivalTime'].apply(parse\_time)

1. Handling NaN values in the DelayMinutes column

df\_flight['DelayMinutes']df\_flight['DelayMinutes'].fillna(df\_flight['DelayMinutes'].median())

1. Dropping duplicates based on FlightNumber and DepartureDate

df\_unique\_flights = df\_flight.drop\_duplicates(subset=['FlightNumber', 'DepartureDate'])

print(f'Total number of unique entries: {len(df\_unique\_flights)}')

df\_unique\_flights

1. Combining Departure/Arrival date and time

df\_unique\_flights ['DepartureDateTime'] = pd.to\_datetime(df\_unique\_flights ['DepartureDate'].astype(str) + ' ' + df\_unique\_flights ['DepartureTime'].astype(str))

df\_unique\_flights ['ArrivalDateTime'] = pd.to\_datetime(df\_unique\_flights ['ArrivalDate'].astype(str) + ' ' + df\_unique\_flights ['ArrivalTime'].astype(str))

j) It contains all the collumns except Departure/Arrivale Date/Time.

new\_df\_flights=df\_unique\_flights.drop(['DepartureDate', 'DepartureTime', 'ArrivalDate', 'ArrivalTime'], axis=1)

1. Adjusting ArrivalDateTime if it is less than DepartureDateTime

bool\_mask = new\_df\_flights ['ArrivalDateTime'] < new\_df\_flights ['DepartureDateTime'] df\_flight.loc[bool\_mask, 'ArrivalDateTime'] += pd.Timedelta(days=1)

1. Creating a new column for FlightDuration by calculating the difference between DepartureTime and ArrivalTime on the same day

“ new\_df\_flights ['FlightDuration']= new\_df\_flights ['ArrivalDateTime'] - new\_df\_flights ['DepartureDateTime']df\_flight “

1. Analyze the distribution of delays and identify any trends or patterns

“ print(new\_df\_flights ['DelayMinutes'].describe()) “

1. Creating visualization using matplotlib

plt.subplot(1, 1, 1)

“ sns.boxplot(x='Airline', y='DelayMinutes', data= new\_df\_flights)

plt.title('Flight Delays by Airline')

plt.xlabel('Airline')

plt.ylabel('Delay (minutes)') “

n) Calculate the average delay for each airline.

“new\_df\_flights.groupby('Airline')['DelayMinutes'].mean() “

o) Creating Delay Analysis

import matplotlib.pyplot as plt

“ new\_df\_flights ['DepartureHour'] =pd.to\_datetime(df\_flight['DepartureDateTime'])

# Grouping by departure hour to calculate average delay

delay\_analysis= new\_df\_flights.groupby('DepartureHour')['DelayMinutes'].mean().re set\_index()

print(delay\_analysis)

p) Creating data visualization using a Box chart

plt.subplot(1, 1, 1)

sns.boxplot(x='Airline', y='DelayMinutes', data=new\_df\_flights)

plt.title('Flight Delays by Airline')

plt.xlabel('Airline')

plt.ylabel('Delay (minutes)')

# Calculate average delay by airline

average\_delay=new\_df\_flights.groupby('Airline')['DelayMinutes'].mean().reset\_index()

# Bar chart for average delay by airline

plt.figure(figsize=(10, 6))

sns.barplot(x='Airline', y='DelayMinutes', data=average\_delay, palette='viridis')

plt.title('Average Delay by Airline')

plt.xlabel('Airline')

plt.ylabel('Average Delay (Minutes)')

plt.xticks(rotation=45)

plt.grid(axis='y')

plt.show()

q) Creating a new upated csv file and downloading it as well

new\_df\_flights.to\_csv('cleaned\_aviation\_dataaa.csv', index=False)

files.download('cleaned\_aviation\_dataaa.csv')

**INSIGHTS:**

1)Provide a summary of the key findings from the data:

**Flight Delays:**

* Average flight delays varied from flight to flight, with some flights being significantly delayed (up to 60 minutes).
* Many airlines, such as American Airlines, was seen to have more frequent delays, indicating viability.

**Flight Duration:**

* Most flights had a continuous flight time of about 2 to 2.5 hours. May be due to this, extremes such as flights being longer due to delays or schedule differences were also observed.

**Airline Performance:**

* American Airlines had the highest number of recorded flights in this given dataset, with significant delays.
* Delta and United Airlines also had more bookings, but had fewer average delays compared to American Airlines.

**Time Trends:**

* Flights scheduled in the afternoon showed greater variability in flight delays compared to morning flights.For instance two flights which departed after 6 PM had got comparatively greater delay as those flights which departed before 6PM

**DELIVERABLES**

1)Insights derived from the data analysis:

**Efficiency Improvements:**

* Airlines will benefit from research on route sharing to implement strategies aimed at reducing delays, especially during peak travel times.

**Customer Communication:**

* As we have seen a varying nature of delays, improving customer communication about potential delays and their causes can enhance the travel experience and also customer satisfaction as well.

**Further Analysis:**

* Identifying relationships between delays, deadlines.Weather can provide insights that can further reduce delays.

**Monitor Trends:**

* Continuous monitoring of these metrics over time can help identify trends, monitor the effectiveness of operational changes, and enable predictive modeling for future flights.

Overall, the dataset highlights important examples and potential areas for operational improvement in the airline industry, as well as the need for robust data governance practices.