a7

October 13, 2024

1 DS 3000 - Assignment 7

Student Name: Shreyas Shukla

Date: 13 Oct 2024

1.0.1 Submission Instructions

Submit this `ipynb` file to canvas.

The `ipynb` format stores outputs from the last time you ran the notebook. (When you open a notebook it has the figures and outputs of the last time you ran it too). To ensure that your submitted `ipynb` file represents your latest code, make sure to give a fresh run `Kernel > Restart & Run All` just before uploading the `ipynb` file to Canvas.

1.0.2 Academic Integrity

Writing your homework is an individual effort. You may discuss general python problems with other students but under no circumstances should you observe another student's code which was written for this assignment, from this year or past years. Pop into office hours or DM us in MS Teams if you have a specific question about your work or if you would like another pair of eyes or talk through your code.

Don't forget to cite websites which helped you solve a problem in a unique way. You can do this in markdown near the code or with a simple one-line comment. You do not need to cite the official python documentation.

Documentation / style counts for credit Please refer to the Pep-8 style, to improve the readability and consistency of your Python code. For more information, read the following article How to Write Beautiful Python Code With PEP 8 or ask your TA's for tips.

NOTE: Write python expressions to answer ALL questions below and ensure that you use the `print()` function to display the output. Each question should be answered in a new code cell. For example, your solution for question 1.1 should be in a different code cell from your solution for question 1.2.

2 Question 1

(3 pts) Load the data into a dataframe. Inspect the data and perform at least one type of data preparation.

[4]:

```
import pandas as pd

# Load the dataset into a dataframe
df_air_traffic = pd.read_csv('/Users/shreyas/Desktop/DS/
 ↪Air_Traffic_Passenger_Statistics.csv')

# Inspect the first few rows of the dataframe
print(df_air_traffic.head())
```

```
Hello, World!
   Activity Period Activity Period Start Date  \
0           199907                 1999/07/01
1           199907                 1999/07/01
2           199907                 1999/07/01
3           199907                 1999/07/01
4           199907                 1999/07/01

                         Operating Airline Operating Airline IATA Code  \
0                             ATA Airlines                          TZ
1                             ATA Airlines                          TZ
2                             ATA Airlines                          TZ
3  Aeroflot Russian International Airlines                         NaN
4  Aeroflot Russian International Airlines                         NaN

                         Published Airline Published Airline IATA Code  \
0                             ATA Airlines                          TZ
1                             ATA Airlines                          TZ
2                             ATA Airlines                          TZ
3  Aeroflot Russian International Airlines                         NaN
4  Aeroflot Russian International Airlines                         NaN

     GEO Summary GEO Region Activity Type Code Price Category Code  \
0       Domestic         US           Deplaned            Low Fare
1       Domestic         US           Enplaned            Low Fare
2       Domestic         US     Thru / Transit            Low Fare
3  International     Europe           Deplaned               Other
4  International     Europe           Enplaned               Other

     Terminal Boarding Area  Passenger Count              data_as_of  \
0  Terminal 1             B            31432  2023/12/21 12:05:27 AM
1  Terminal 1             B            31353  2023/12/21 12:05:27 AM
2  Terminal 1             B             2518  2023/12/21 12:05:27 AM
3  Terminal 2             D             1324  2023/12/21 12:05:27 AM
4  Terminal 2             D             1198  2023/12/21 12:05:27 AM

           data_loaded_at
0  2024/01/20 07:02:35 AM
1  2024/01/20 07:02:35 AM
```

```
2  2024/01/20 07:02:35 AM
3  2024/01/20 07:02:35 AM
4  2024/01/20 07:02:35 AM
```

[6]:

```
# Check the structure and types of data
print(df_air_traffic.info())

# Check for missing values
print(df_air_traffic.isnull().sum())

# Get basic statistics
print(df_air_traffic.describe())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 35172 entries, 0 to 35171
Data columns (total 15 columns):
 #   Column                       Non-Null Count  Dtype
---  ------                       --------------  -----
 0   Activity Period              35172 non-null  int64
 1   Activity Period Start Date   35172 non-null  object
 2   Operating Airline            35172 non-null  object
 3   Operating Airline IATA Code  34856 non-null  object
 4   Published Airline            35172 non-null  object
 5   Published Airline IATA Code  34856 non-null  object
 6   GEO Summary                  35172 non-null  object
 7   GEO Region                   35172 non-null  object
 8   Activity Type Code           35172 non-null  object
 9   Price Category Code          35172 non-null  object
 10  Terminal                     35172 non-null  object
 11  Boarding Area                35172 non-null  object
 12  Passenger Count              35172 non-null  int64
 13  data_as_of                   35172 non-null  object
 14  data_loaded_at               35172 non-null  object
dtypes: int64(2), object(13)
memory usage: 4.0+ MB
None
Activity Period                  0
Activity Period Start Date       0
Operating Airline                0
Operating Airline IATA Code    316
Published Airline                0
Published Airline IATA Code    316
GEO Summary                      0
GEO Region                       0
Activity Type Code               0
Price Category Code              0
Terminal                         0
Boarding Area                    0
```

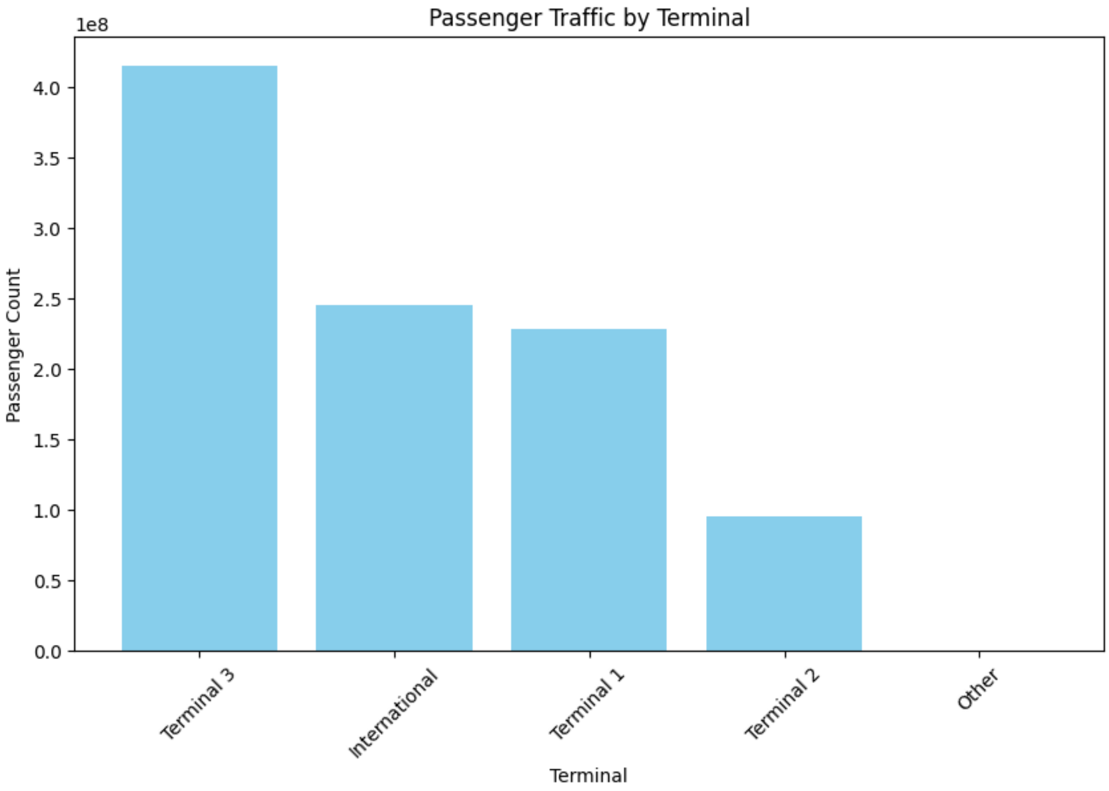
```
Passenger Count                 0
data_as_of                      0
data_loaded_at                  0
dtype: int64
       Activity Period  Passenger Count
count     35172.000000     35172.000000
mean     201173.280792     28000.508075
std         706.637527     62772.762446
min      199907.000000         0.000000
25%      200603.000000      4452.000000
50%      201206.000000      8634.000000
75%      201804.000000     19893.750000
max      202311.000000    856501.000000
```

[8]:
```
df_cleaned = df_air_traffic.dropna()
```

3 Question 2

(5 pts) Determine which terminal has the most passenger traffic and visualize the results. Explain the graph.

[9]:
```
import matplotlib.pyplot as plt

# Group by terminal and sum the passenger count
terminal_traffic = df_cleaned.groupby('Terminal')['Passenger Count'].sum().
 ↪reset_index()

# Sort the values for better visualization
terminal_traffic = terminal_traffic.sort_values(by='Passenger Count',␣
 ↪ascending=False)

# Plot the results
plt.figure(figsize=(10, 6))
plt.bar(terminal_traffic['Terminal'], terminal_traffic['Passenger Count'],␣
 ↪color='skyblue')
plt.xlabel('Terminal')
plt.ylabel('Passenger Count')
plt.title('Passenger Traffic by Terminal')
plt.xticks(rotation=45)
plt.show()
```



Explain Above

4 Question 3

(10 pts) Plot the annual domestic and international enplanement and deplanement using a line chart. Explain the patterns and discuss which years have the most and least passenger traffic. Tip: It is recommended that you create two charts; chart #1 for the international travelers and Chart #2 for the domestic travelers. Then in each chart, visualize the number of passengers who enplaned and deplaned each year. The annual passenger enplanement and deplanement should be visualized using separate lines.

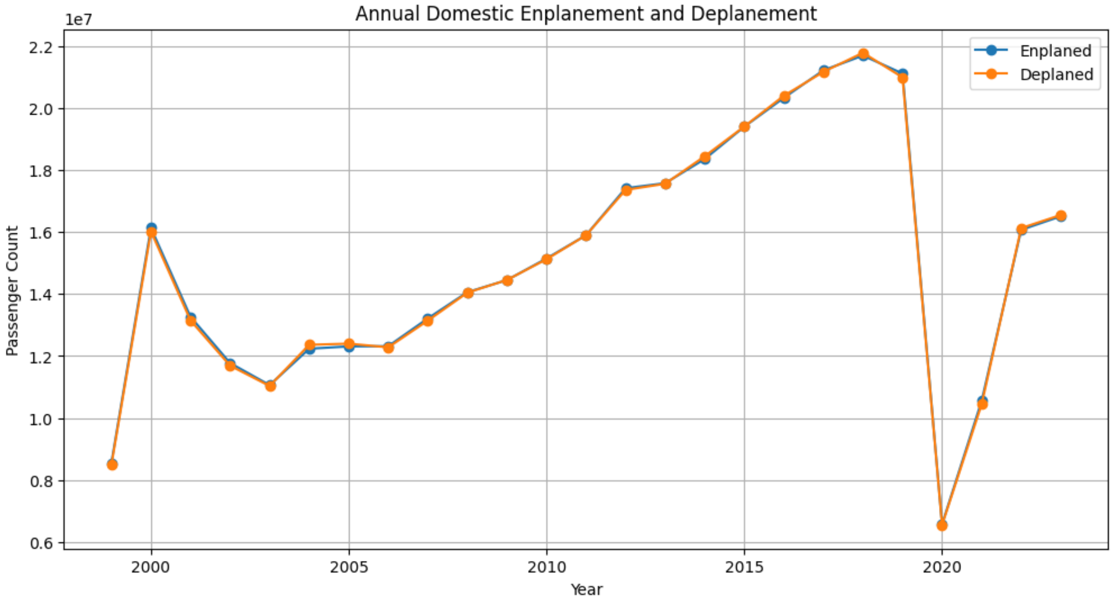
[10]:

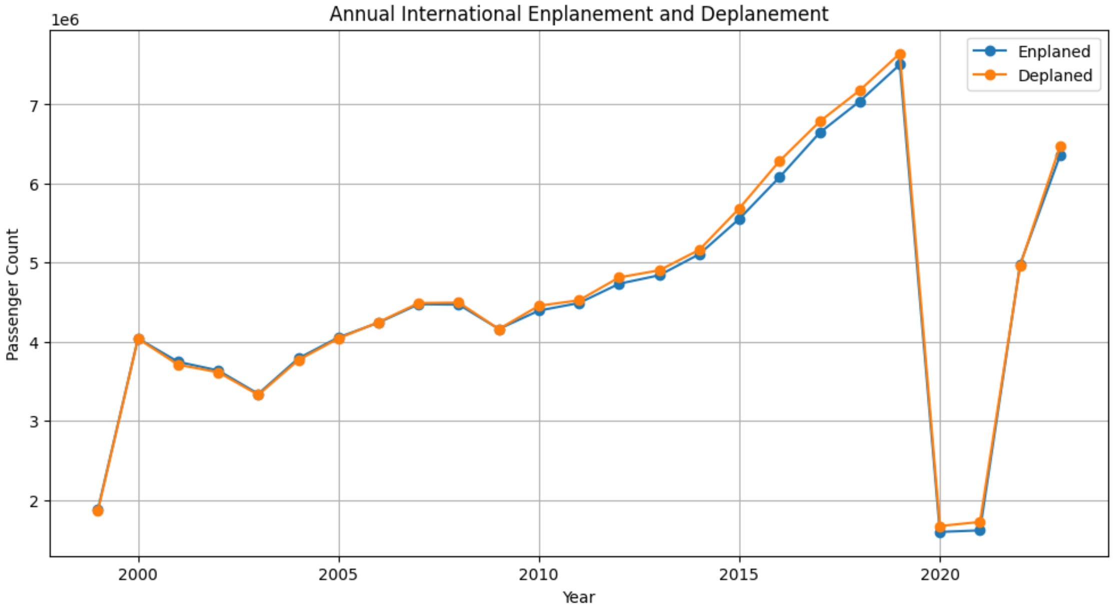
```
# Convert 'Activity Period Start Date' to datetime
df_cleaned['Activity Period Start Date'] = pd.to_datetime(df_cleaned['Activity␣
  ↪Period Start Date'])

# Extract year from 'Activity Period Start Date'
df_cleaned['Year'] = df_cleaned['Activity Period Start Date'].dt.year

# Filter data for domestic and international passengers
df_domestic = df_cleaned[df_cleaned['GEO Summary'] == 'Domestic']
df_international = df_cleaned[df_cleaned['GEO Summary'] == 'International']
```

```
# Group by year and activity type, then sum the passenger count
domestic_annual = df_domestic.groupby(['Year', 'Activity Type␣
↪Code'])['Passenger Count'].sum().unstack()
international_annual = df_international.groupby(['Year', 'Activity Type␣
↪Code'])['Passenger Count'].sum().unstack()

# Plot domestic passengers
plt.figure(figsize=(12, 6))
plt.plot(domestic_annual.index, domestic_annual['Enplaned'], label='Enplaned',␣
↪marker='o')
plt.plot(domestic_annual.index, domestic_annual['Deplaned'], label='Deplaned',␣
↪marker='o')
plt.xlabel('Year')
plt.ylabel('Passenger Count')
plt.title('Annual Domestic Enplanement and Deplanement')
plt.legend()
plt.grid(True)
plt.show()

# Plot international passengers
plt.figure(figsize=(12, 6))
plt.plot(international_annual.index, international_annual['Enplaned'],␣
↪label='Enplaned', marker='o')
plt.plot(international_annual.index, international_annual['Deplaned'],␣
↪label='Deplaned', marker='o')
plt.xlabel('Year')
plt.ylabel('Passenger Count')
plt.title('Annual International Enplanement and Deplanement')
plt.legend()
plt.grid(True)
plt.show()
```

```
/var/folders/_w/v3q2hr0x7qzb2pk_1v1k0slc0000gn/T/ipykernel_80520/2227113137.py:2
: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_cleaned['Activity Period Start Date'] = pd.to_datetime(df_cleaned['Activity
Period Start Date'])
/var/folders/_w/v3q2hr0x7qzb2pk_1v1k0slc0000gn/T/ipykernel_80520/2227113137.py:5
: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
```

```
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_cleaned['Year'] = df_cleaned['Activity Period Start Date'].dt.year
```





Explain Above

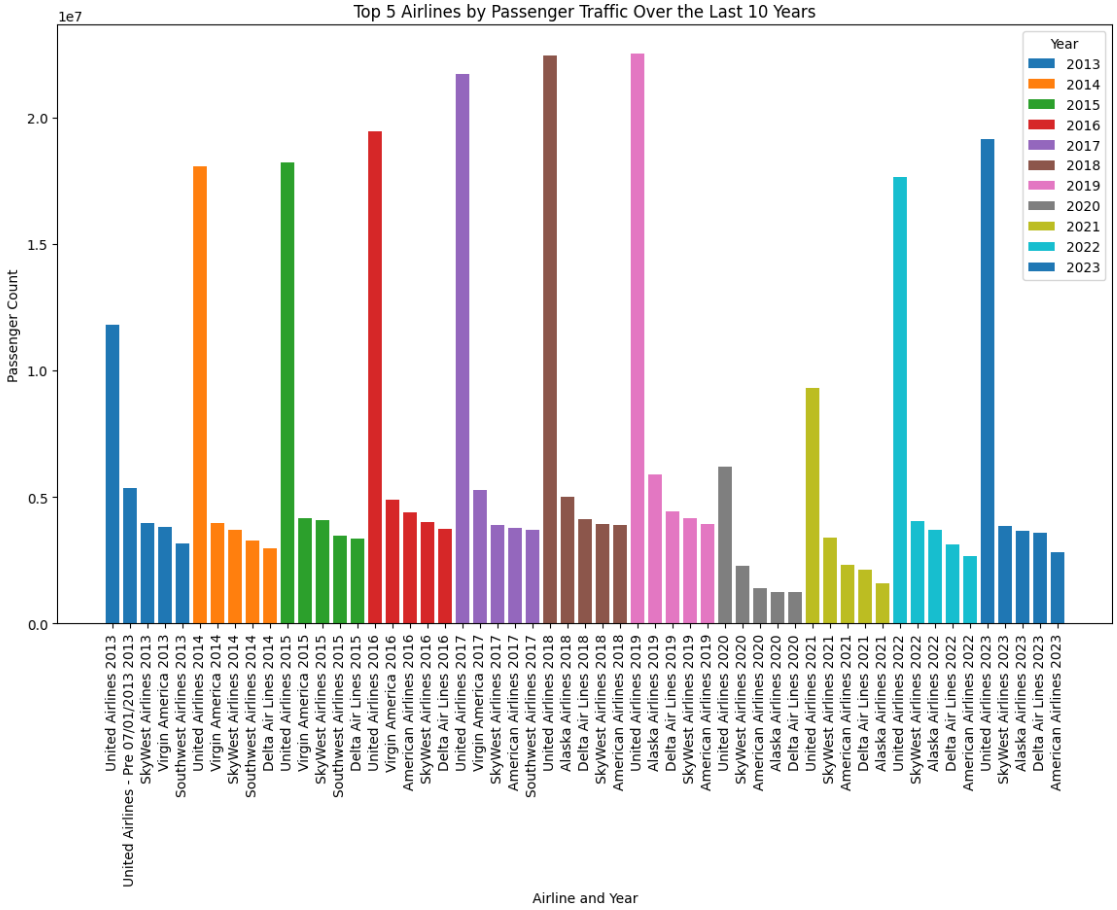
5 Question 4

Questions 4 requires that you analyze the data for each year separately i.e. you are analyzing the data year over year. (10 pts) Demonstrate which airlines contributed to the most passenger traffic over the last 10 years and visualize the results**? Are there changes in the top airlines each year? Ensure that you visualize the top 5 airlines and explain the patterns.

[11]:

```
# Filter data for the last 10 years
recent_years = df_cleaned[df_cleaned['Year'] >= df_cleaned['Year'].max() - 10]

# Group by year and airline, then sum the passenger count
airline_traffic = recent_years.groupby(['Year', 'Operating␣
 ↪Airline'])['Passenger Count'].sum().reset_index()

# Identify the top 5 airlines for each year
top_airlines_per_year = airline_traffic.groupby('Year').apply(lambda x: x.
 ↪nlargest(5, 'Passenger Count')).reset_index(drop=True)

# Plot the results
plt.figure(figsize=(14, 8))
for year in top_airlines_per_year['Year'].unique():
    subset = top_airlines_per_year[top_airlines_per_year['Year'] == year]
    plt.bar(subset['Operating Airline'] + ' ' + subset['Year'].astype(str),␣
 ↪subset['Passenger Count'], label=str(year))

plt.xlabel('Airline and Year')
plt.ylabel('Passenger Count')
plt.title('Top 5 Airlines by Passenger Traffic Over the Last 10 Years')
plt.xticks(rotation=90)
plt.legend(title='Year')
plt.show()
```

```
/var/folders/_w/v3q2hr0x7qzb2pk_1v1k0slc0000gn/T/ipykernel_80520/3521855140.py:8
: DeprecationWarning: DataFrameGroupBy.apply operated on the grouping columns.
This behavior is deprecated, and in a future version of pandas the grouping
columns will be excluded from the operation. Either pass `include_groups=False`
to exclude the groupings or explicitly select the grouping columns after groupby
to silence this warning.
  top_airlines_per_year = airline_traffic.groupby('Year').apply(lambda x:
x.nlargest(5, 'Passenger Count')).reset_index(drop=True)
```



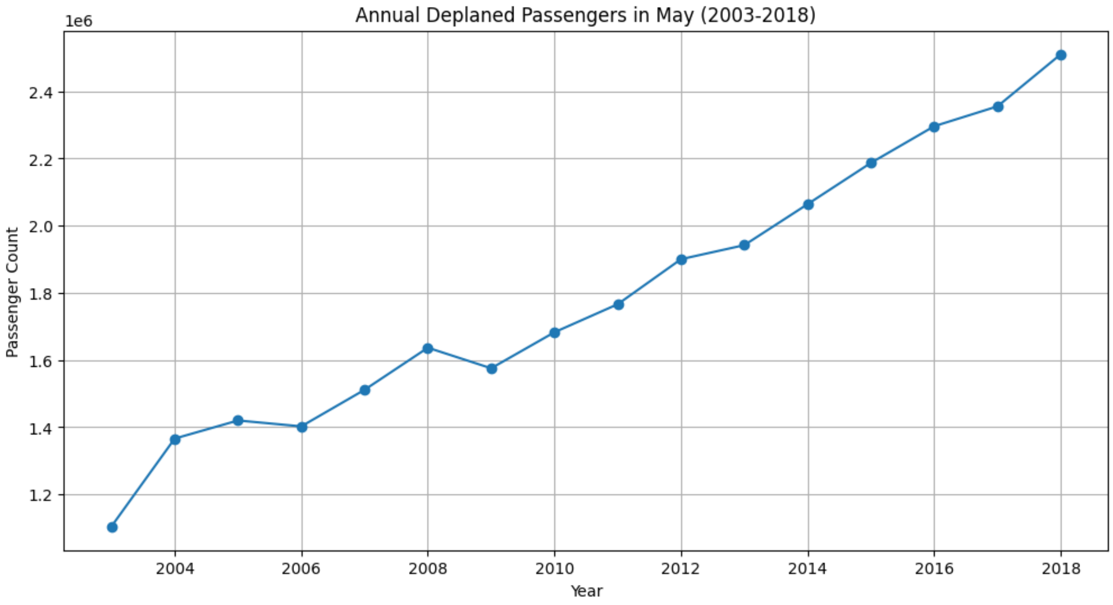
Explain above

6 Question 5

(7 pts) Extract data from 2003 to 2018 (inclusive), for all passengers who deplaned the aircraft in May e.g. May 2003, May 2004, May 2005 etc. Visualize the extracted data, using a line chart, and comment on the trend.

[12]:
```
# Filter data for the years 2003 to 2018 and for passengers who deplaned in May
may_deplaned = df_cleaned[(df_cleaned['Year'] >= 2003) & (df_cleaned['Year'] <=␣
 ↪2018) &
                          (df_cleaned['Activity Type Code'] == 'Deplaned') &
                          (df_cleaned['Activity Period Start Date'].dt.month ==␣
 ↪5)]

# Group by year and sum the passenger count
may_deplaned_annual = may_deplaned.groupby('Year')['Passenger Count'].sum().
 ↪reset_index()
```

```
# Plot the results
plt.figure(figsize=(12, 6))
plt.plot(may_deplaned_annual['Year'], may_deplaned_annual['Passenger Count'],␣
  ↪marker='o')
plt.xlabel('Year')
plt.ylabel('Passenger Count')
plt.title('Annual Deplaned Passengers in May (2003-2018)')
plt.grid(True)
plt.show()
```



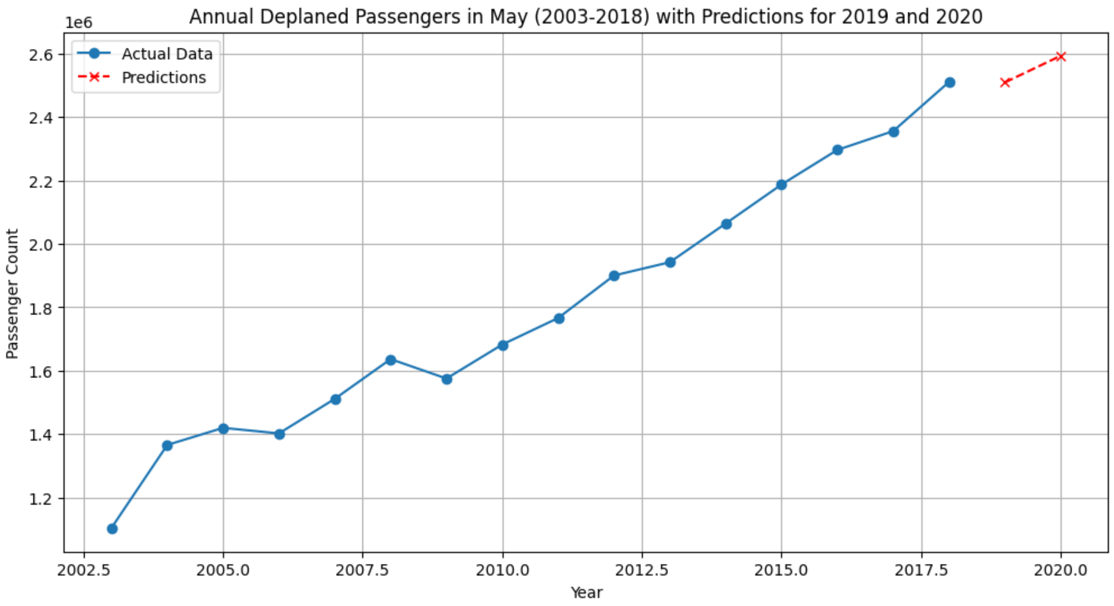
Explain Above

7 Question 6

Answer questions 6 & 7 using the extracted data from question 5 above (10 pts) Predict the passenger count for May 2019 and May 2020, using simple linear regression. Ensure that you visualize the original data and also include the predictions from the model. Compare the predicted passenger count with the actual passenger count. In your explanation, discuss how close/far your predictions are from the actual values.

[13]:
```
from sklearn.linear_model import LinearRegression
import numpy as np

# Prepare the data
X = may_deplaned_annual['Year'].values.reshape(-1, 1)
y = may_deplaned_annual['Passenger Count'].values
```

```python
# Create and train the model
model = LinearRegression()
model.fit(X, y)

# Predict for May 2019 and May 2020
years_to_predict = np.array([[2019], [2020]])
predictions = model.predict(years_to_predict)

# Visualize the original data and predictions
plt.figure(figsize=(12, 6))
plt.plot(may_deplaned_annual['Year'], may_deplaned_annual['Passenger Count'], marker='o', label='Actual Data')
plt.plot(years_to_predict, predictions, marker='x', linestyle='--', color='red', label='Predictions')
plt.xlabel('Year')
plt.ylabel('Passenger Count')
plt.title('Annual Deplaned Passengers in May (2003-2018) with Predictions for 2019 and 2020')
plt.legend()
plt.grid(True)
plt.show()

# Print the predictions
print(f"Predicted passenger count for May 2019: {predictions[0]:.0f}")
print(f"Predicted passenger count for May 2020: {predictions[1]:.0f}")
```



```
Predicted passenger count for May 2019: 2508690
Predicted passenger count for May 2020: 2592643
```

Explain above

8 Question 7

(5 pts) Calculate the R-squared and explain what this tells you about the model.

[15]:

```
# Calculate the R-squared value
r_squared = model.score(X, y)

# Print the R-squared value
print(f"R-squared value: {r_squared:.4f}")
```

```
R-squared value: 0.9752
```

Explain Above

9 Question 8

(5 pts)[optional/bonus] This is a challenge question that may require some research. Choose ONE of the following:

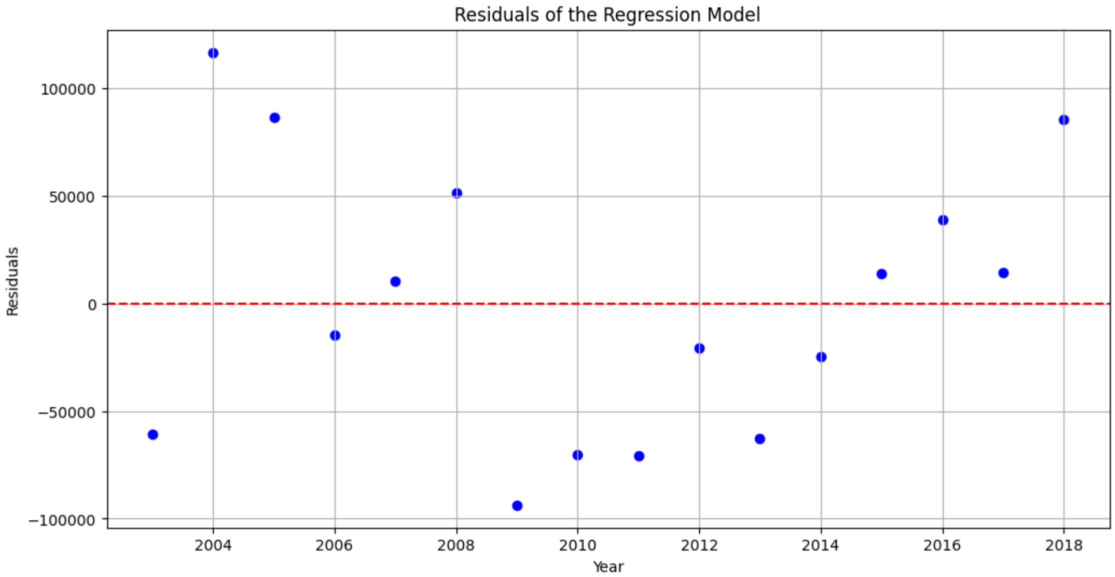
8.1 optional/challenge - residuals: Visualize the residuals of the regression model in question 6 and explain the chart. Note: A residual is the difference between the actual and predicted values.

OR

8.2 optional/challenge - EDA: Perform a year over year analysis of the passengers who traveled through the international terminal over the last 5 years to identify the peak travel times. Visualize the results and you are welcome to be very creative with this analysis.

[16]:

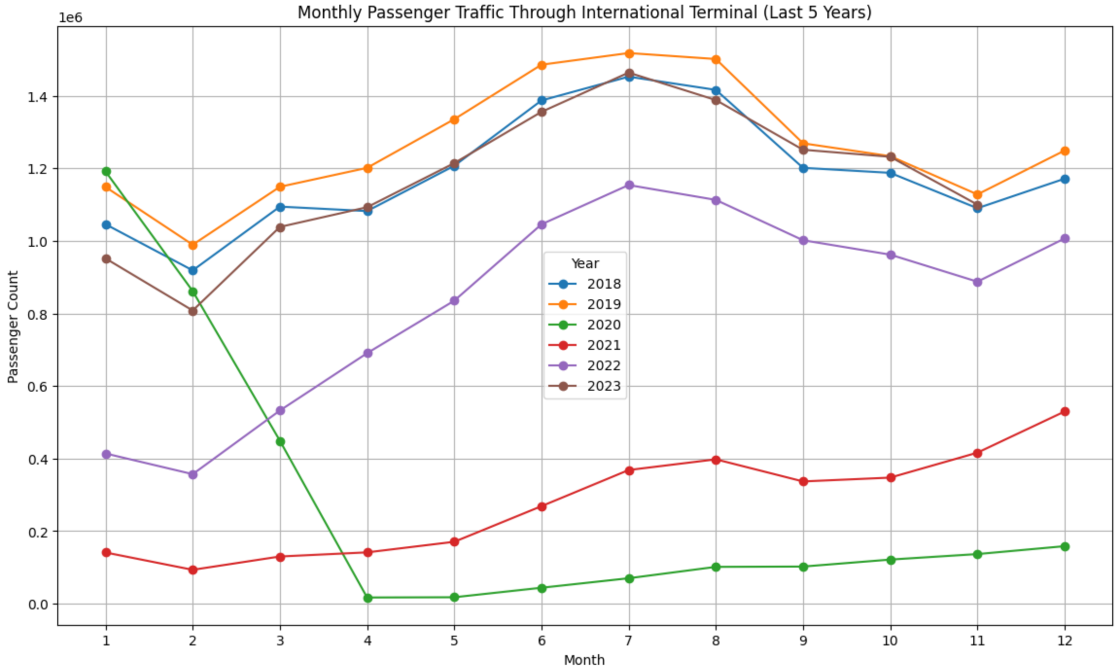
```
# Calculate the residuals
residuals = y - model.predict(X)

# Plot the residuals
plt.figure(figsize=(12, 6))
plt.scatter(X, residuals, marker='o', color='blue')
plt.axhline(y=0, color='red', linestyle='--')
plt.xlabel('Year')
plt.ylabel('Residuals')
plt.title('Residuals of the Regression Model')
plt.grid(True)
plt.show()
```



Explain Above

[17]:
```
# Filter data for the last 5 years and international terminal
last_5_years = df_international[df_international['Year'] >=␣
 ↪df_international['Year'].max() - 5]

# Extract month from 'Activity Period Start Date'
last_5_years['Month'] = last_5_years['Activity Period Start Date'].dt.month

# Group by year and month, then sum the passenger count
monthly_traffic = last_5_years.groupby(['Year', 'Month'])['Passenger Count'].
 ↪sum().unstack()

# Plot the results
plt.figure(figsize=(14, 8))
for year in monthly_traffic.index:
    plt.plot(monthly_traffic.columns, monthly_traffic.loc[year], marker='o',␣
 ↪label=str(year))

plt.xlabel('Month')
plt.ylabel('Passenger Count')
plt.title('Monthly Passenger Traffic Through International Terminal (Last 5␣
 ↪Years)')
plt.xticks(range(1, 13))
plt.legend(title='Year')
plt.grid(True)
plt.show()
```

```
/var/folders/_w/v3q2hr0x7qzb2pk_1v1k0slc0000gn/T/ipykernel_80520/1895115520.py:5
: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  last_5_years['Month'] = last_5_years['Activity Period Start Date'].dt.month
```



Explain Above