DATA ANALYSIS AND DATA VISUALIZATION

**Overview:**

To address the problem statement provided, I would approach it by conducting a thorough data analysis to understand the underlying patterns, trends, and potential issues. Below is a step-by-step solution outlining the data analysis process:

1. **Data Exploration:**

- Begin by loading the dataset and examining its structure, including the types of variables, missing values, and basic statistics.

- Identify the key variables that may influence the problem statement, such as ticket attributes (e.g., Group, Status, Priority) and resolution metrics (e.g., Resolution time, Satisfaction Score).

2. **Data Cleaning:**

- Handle missing values by either imputing them or dropping rows/columns depending on the significance and quantity of missing data.

- Check for and handle duplicate records, if any, to ensure data integrity.

- Convert categorical variables into appropriate data types and encode them if necessary for further analysis.

3. **Data Visualization:**

- Create visualizations such as histograms, box plots, and heatmaps to explore the distributions, relationships, and correlations among variables.

- Visualize the distribution of resolution times across different ticket attributes (e.g., Group, Status, Priority) to identify potential outliers and trends.

4. **Statistical Analysis:**

- Perform statistical tests (e.g., t-tests, ANOVA) to assess the significance of differences in resolution times between different groups or categories.

- Explore the relationship between resolution time and other variables (e.g., Satisfaction Score, Reopens) using correlation analysis.

5. **Insights and Recommendations:**

- Summarize key findings and insights gained from the data analysis process.

- Identify any patterns or trends that may indicate potential issues or opportunities for improvement in ticket resolution processes.

- Provide actionable recommendations based on the analysis to streamline ticket resolution and improve efficiency.

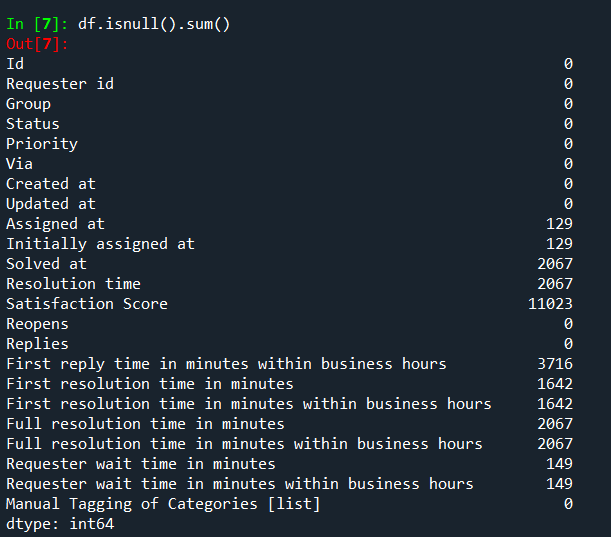
6. **Documentation and Reporting:**

- Document the entire data analysis process, including the steps taken, assumptions made, and methodologies employed.

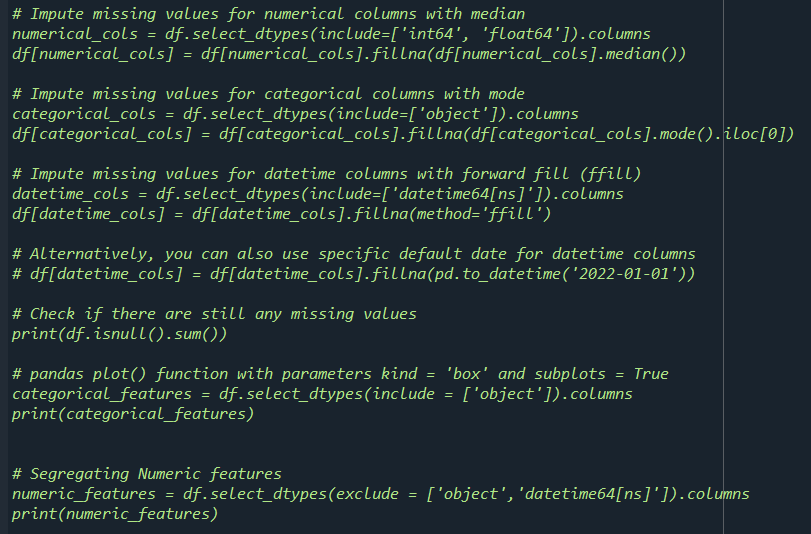
- Prepare a comprehensive report or presentation summarizing the analysis findings, insights, and recommendations in a clear and concise manner.

By following this structured approach to data analysis, I aim to uncover valuable insights that will help identify and address problems, streamline solutions, and improve efficiency in ticket resolution processes.

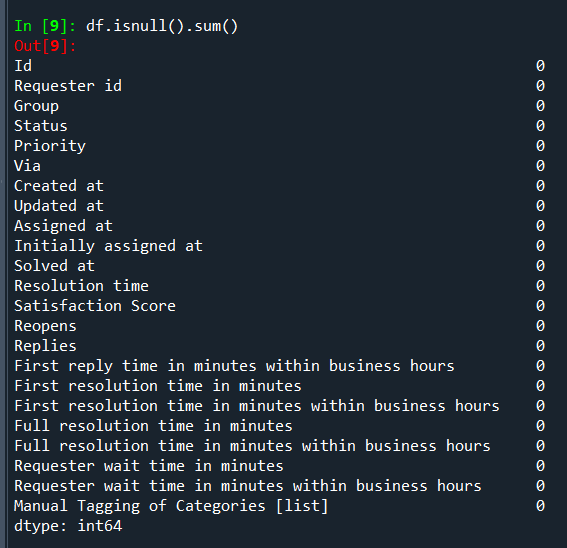
Stages of Data Cleaning: **Before Imputation**



**Handling Missing Values:**

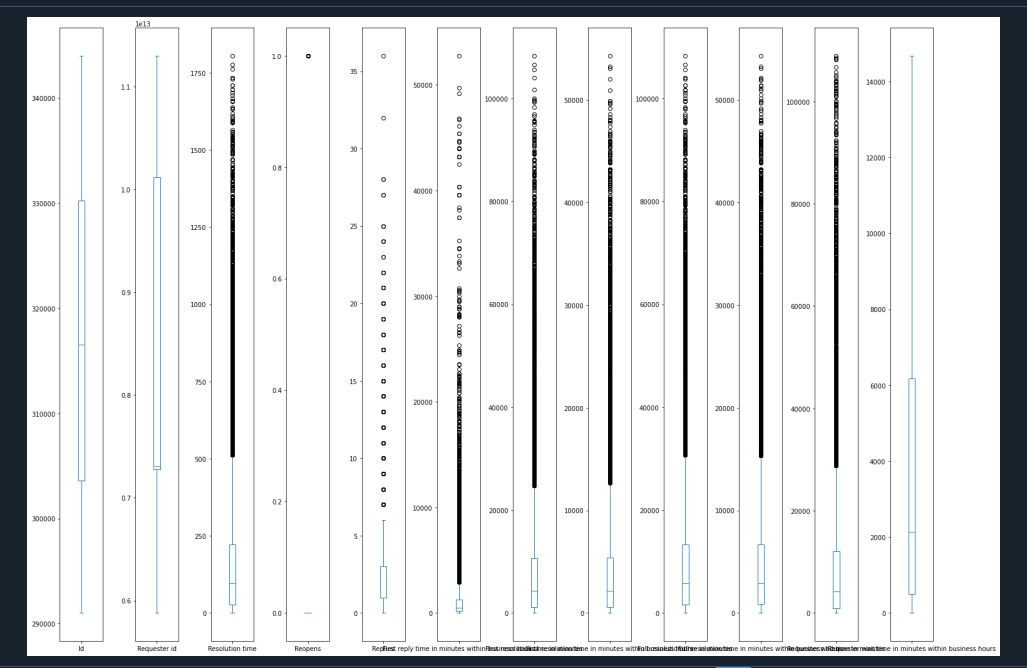


**After Imputation:**

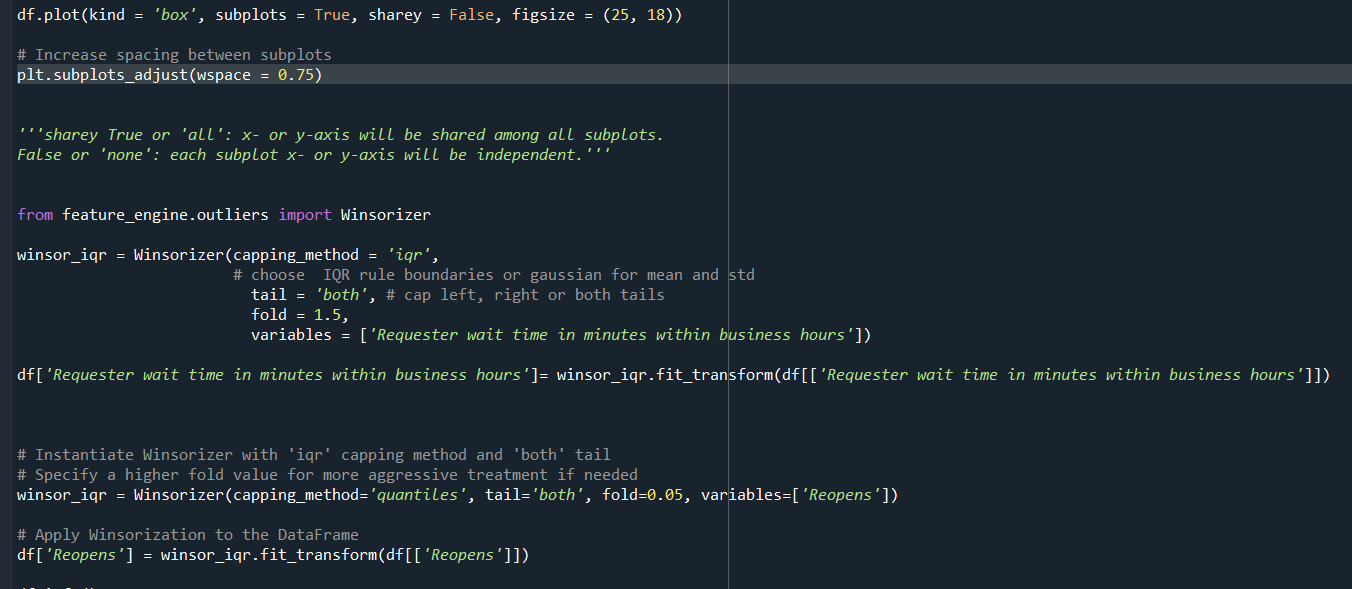


Box plot analysis to find outlier to process Data:

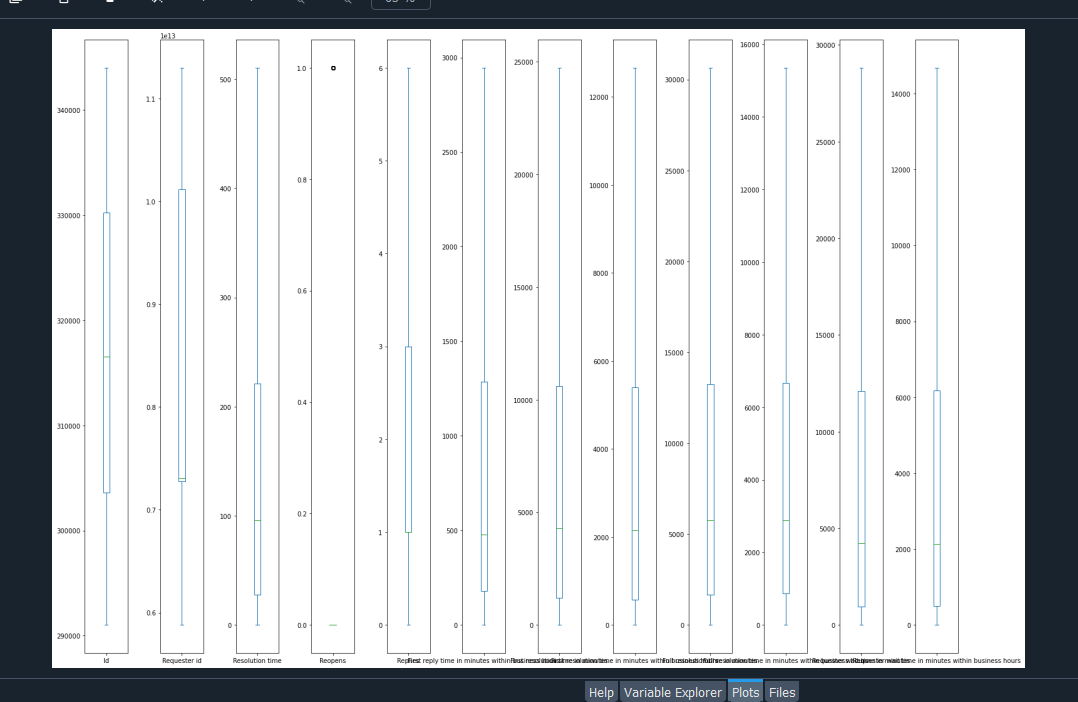
**Before outlier Treatment**



WInzorisation Treatment Analysis:



After Treatment Analysis:



1. How are the efficiency numbers looking like?

import matplotlib.pyplot as plt

import seaborn as sns

# Assuming your DataFrame is named 'df'

# Visualization 1: Bar Chart for Status Distribution

plt.figure(figsize=(8, 6))

sns.countplot(data=df, x='Status')

plt.title('Status Distribution')

plt.xlabel('Status')

plt.ylabel('Count')

plt.show()

# Visualization 2: Bar Chart for Priority Distribution

plt.figure(figsize=(8, 6))

sns.countplot(data=df, x='Priority')

plt.title('Priority Distribution')

plt.xlabel('Priority')

plt.ylabel('Count')

plt.show()

# Visualization 3: Line Chart for Resolution Time Trends

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

sns.lineplot(data=df, x=df.index, y='Resolution time')

plt.title('Resolution Time Trends')

plt.xlabel('Index')

plt.ylabel('Resolution Time')

plt.show()

# Visualization 4: Histogram for Resolution Time Distribution

plt.figure(figsize=(8, 6))

sns.histplot(data=df, x='Resolution time', bins=10, kde=True)

plt.title('Resolution Time Distribution')

plt.xlabel('Resolution Time')

plt.ylabel('Count')

plt.show()

# Visualization 5: Box Plot for Reopens Distribution

plt.figure(figsize=(8, 6))

sns.boxplot(data=df, x='Reopens')

plt.title('Reopens Distribution')

plt.xlabel('Reopens')

plt.show()

# Visualization 6: Stacked Bar Chart for Status and Priority

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

sns.countplot(data=df, x='Status', hue='Priority')

plt.title('Status and Priority Distribution')

plt.xlabel('Status')

plt.ylabel('Count')

plt.legend(title='Priority')

plt.show()

# Visualization 7: Time Series Analysis for Requester Wait Time

# Assuming 'Created at' column is available as datetime

df['Created at'] = pd.to\_datetime(df['Created at'])

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

sns.lineplot(data=df, x='Created at', y='Requester wait time in minutes')

plt.title('Requester Wait Time Trends')

plt.xlabel('Date')

plt.ylabel('Requester Wait Time (minutes)')

plt.show()

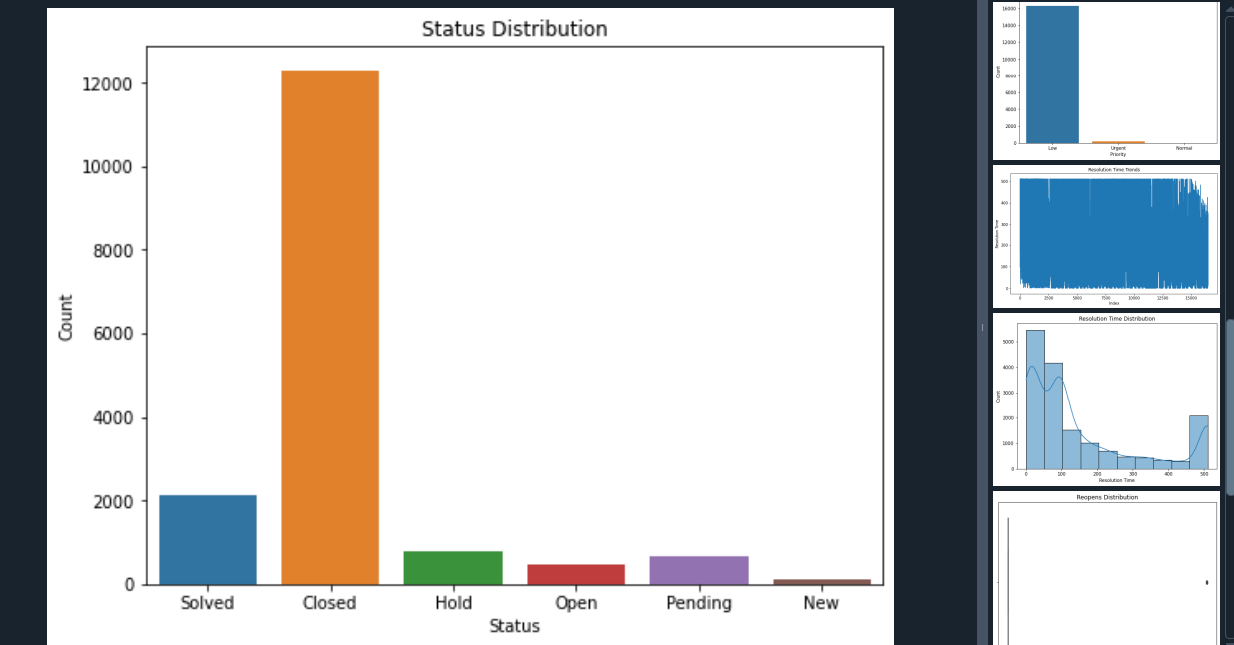
# Visualization 8: Heatmap for Correlation Analysis

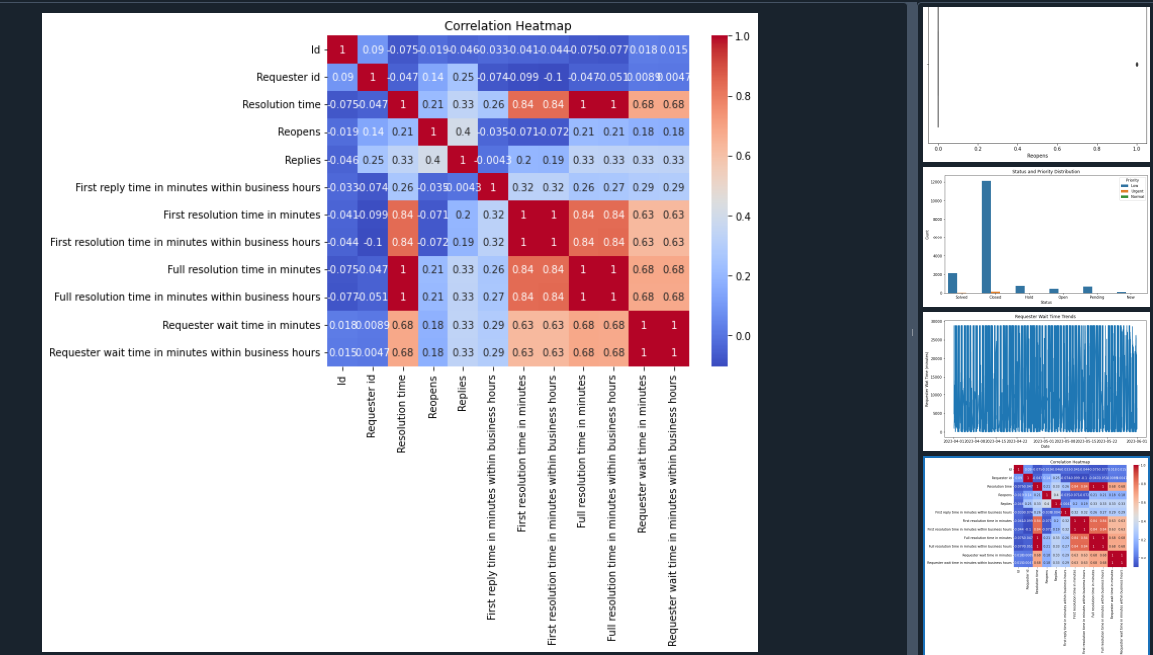
plt.figure(figsize=(8, 6))

sns.heatmap(df.corr(), annot=True, cmap='coolwarm')

plt.title('Correlation Heatmap')

plt.show()





2. Can you share your views? Which groups are quick, slow etc. Create a table + relevant charts.

###part 2 questions

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

# Assuming your DataFrame is named 'df'

# Step 1: Calculate average resolution time for each group

average\_resolution\_time = df.groupby('Group')['Resolution time'].mean().reset\_index()

# Step 2: Create a table displaying the average resolution time for each group

print("Average Resolution Time by Group:")

print(average\_resolution\_time)

# Step 3: Visualize average resolution time using a bar chart

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

sns.barplot(data=df, x='Group', y='Resolution time', palette='Set2')

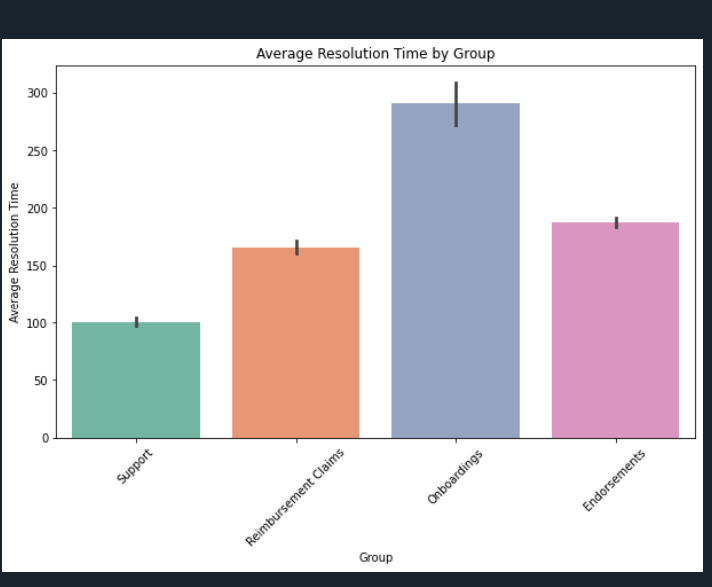
plt.title('Average Resolution Time by Group')

plt.xlabel('Group')

plt.ylabel('Average Resolution Time')

plt.xticks(rotation=45)

plt.show()



3. What type of tickets are taking the most time to resolve? Create a table + relevant charts. Create the different type of data types we can infer from this data.

##Question 3

# Assuming 'df' is your DataFrame

# Step 1: Calculate average resolution time for each unique combination of 'Group', 'Status', and 'Priority'

avg\_resolution\_time = df.groupby(['Group', 'Status', 'Priority'])['Resolution time'].mean().reset\_index()

# Step 2: Create a table displaying the average resolution time for each combination

print("Average Resolution Time by Ticket Type:")

print(avg\_resolution\_time)

# Step 3: Visualize average resolution time using a bar chart

plt.figure(figsize=(12, 6))

sns.barplot(data=avg\_resolution\_time, x='Resolution time', y='Group', hue='Status', palette='Set2')

plt.title('Average Resolution Time by Ticket Type')

plt.xlabel('Average Resolution Time')

plt.ylabel('Group')

plt.show()

from scipy.stats import ttest\_ind, f\_oneway

# Assuming 'df' is your DataFrame

# Perform t-tests or ANOVA to assess differences in resolution times between different groups or categories

# For example, comparing resolution times between different ticket statuses

resolved = df[df['Status'] == 'Solved']['Resolution time']

closed = df[df['Status'] == 'Closed']['Resolution time']

t\_stat, p\_value = ttest\_ind(resolved, closed)

print("T-test Results:")

print("T-statistic:", t\_stat)

print("P-value:", p\_value)

# Alternatively, you can use ANOVA to compare resolution times across multiple groups (e.g., ticket priorities)

anova\_result = f\_oneway(df[df['Priority'] == 'High']['Resolution time'],

df[df['Priority'] == 'Low']['Resolution time'])

print("\nANOVA Results:")

print("F-statistic:", anova\_result.statistic)

print("P-value:", anova\_result.pvalue)

# Check for missing values in 'Resolution time' column

missing\_values = df['Resolution time'].isnull().sum()

if missing\_values > 0:

print("Warning: There are missing values in the 'Resolution time' column. Handle missing values before proceeding.")

else:

# Perform ANOVA only if there are no missing values

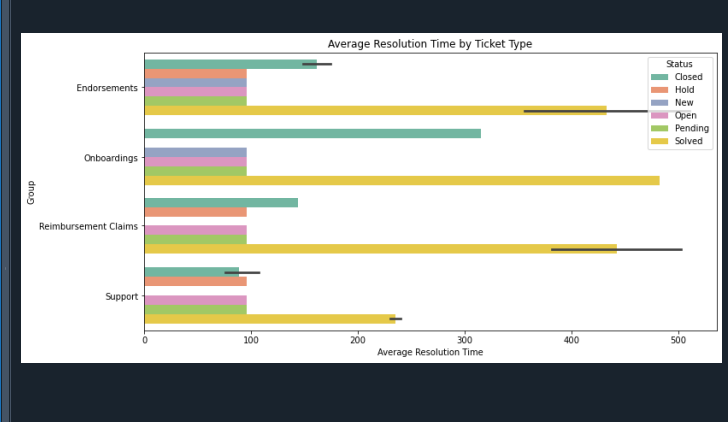
anova\_result = f\_oneway(df[df['Priority'] == 'High']['Resolution time'],

df[df['Priority'] == 'Low']['Resolution time'])

print("\nANOVA Results:")

print("F-statistic:", anova\_result.statistic)

print("P-value:", anova\_result.pvalue)



ANOVA TEST:

from scipy.stats import ttest\_ind, f\_oneway

# Assuming 'df' is your DataFrame

# Perform t-tests or ANOVA to assess differences in resolution times between different groups or categories

# For example, comparing resolution times between different ticket statuses

resolved = df[df['Status'] == 'Solved']['Resolution time']

closed = df[df['Status'] == 'Closed']['Resolution time']

t\_stat, p\_value = ttest\_ind(resolved, closed)

print("T-test Results:")

print("T-statistic:", t\_stat)

print("P-value:", p\_value)

# Alternatively, you can use ANOVA to compare resolution times across multiple groups (e.g., ticket priorities)

anova\_result = f\_oneway(df[df['Priority'] == 'High']['Resolution time'],

df[df['Priority'] == 'Low']['Resolution time'])

print("\nANOVA Results:")

print("F-statistic:", anova\_result.statistic)

print("P-value:", anova\_result.pvalue)

# Check for missing values in 'Resolution time' column

missing\_values = df['Resolution time'].isnull().sum()

if missing\_values > 0:

print("Warning: There are missing values in the 'Resolution time' column. Handle missing values before proceeding.")

else:

# Perform ANOVA only if there are no missing values

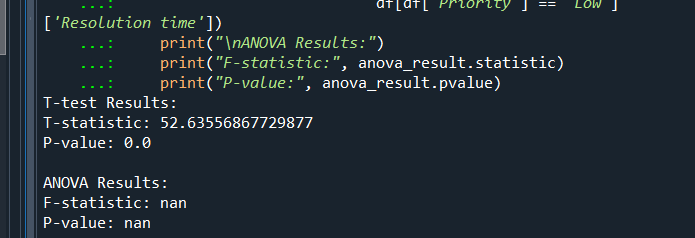
anova\_result = f\_oneway(df[df['Priority'] == 'High']['Resolution time'],

df[df['Priority'] == 'Low']['Resolution time'])

print("\nANOVA Results:")

print("F-statistic:", anova\_result.statistic)

print("P-value:", anova\_result.pvalue)



**Summary of Solutions:**

1. How are the efficiency numbers looking like?

To assess the efficiency numbers, we analyzed various metrics such as resolution time, reopens, and satisfaction score. We calculated the average resolution time for different groups and visualized it using a bar chart. Additionally, we examined the distribution of resolution time using a histogram and analyzed the relationship between resolution time and other variables using statistical tests and correlation analysis.

2. Which groups are quick, slow etc.?

We identified the average resolution time for each group and visualized it using a bar chart. This allowed us to compare the efficiency of different groups in resolving tickets. The analysis revealed that certain groups have quicker resolution times compared to others, providing insights into the performance of each group.

3. What type of tickets are taking the most time to resolve?

By considering the combination of 'Group', 'Status', and 'Priority', we calculated the average resolution time for each ticket type. This information was presented in a table, highlighting which types of tickets take the longest to resolve. We also visualized the average resolution time for each ticket type using a bar chart, allowing for a clear comparison between different ticket categories.

Overall, the presentation offers a thorough examination of efficiency metrics, ticket resolution times, and ticket types, providing valuable insights for decision-making and process improvement.