Superstore Sales Analysis Project

1. Project Overview

The purpose of this project is to analyze sales and shipping performance using the Superstore dataset. The project focuses on two main areas:

- **Exploratory Data Analysis (EDA)** of sales to uncover key patterns and trends.
- Investigating and addressing the **shipping delay problem** by identifying factors that contribute to delays and proposing solutions to improve shipping efficiency.

2. Purpose of the Project

Sales Analysis: To gain insights into sales performance, product categories, and regional sales distribution to inform decision-making.

Shipping Delay Analysis: To understand the reasons behind shipping delays and provide actionable recommendations to reduce delays and improve customer satisfaction.

3. Dataset Description

The Superstore dataset contains various columns that capture information about orders, customers, sales, and shipping. The key columns include:

- **Order ID**: This is the unique identifier for each order placed by a customer. Multiple items in the same order would share the same Order ID.

- **Order Date**: The date on which the customer placed the order.

- **Ship Date**: The date on which the order was shipped to the customer.

- **Ship Mode**: This column indicates the shipping method used for the order (e.g., Standard Class, Second Class, First Class).

- **Customer ID**: The unique identifier for each customer. It helps in distinguishing between different customers.

- **Customer Name**: The name of the customer who placed the order.
- **Segment**: The customer segment, which might represent categories like Consumer, Corporate, or Home Office, showing which group the customer belongs to.
- **Country**: The country where the order was placed or where the customer is located.
- **City**: The city where the customer is located or where the product was delivered.
- **State**: The state where the customer is located or where the product was delivered.
- **Postal Code**: The postal code for the customer's address.
- **Region**: The geographical region (such as East, West, Central) where the customer is located.
- **Product ID**: The unique identifier for each product sold in the order.
- **Category**: The broad category of the product (e.g., Furniture, Office Supplies, Technology).
- **Sub-Category**: A more specific classification of the product, a subdivision of the Category (e.g., Chairs, Tables, Binders).
- **Product Name**: The specific name of the product sold in the order.
- **Sales**: The total sales amount for the product in the order. This represents the revenue generated from the sale of the item.

4. Project Goals

Sales Analysis Goals:

Identify top-performing products and categories.

Analyze regional sales patterns.

Examine the impact of customer segments on sales performance.

Shipping Delay Goals:

Determine the main causes of shipping delays.

Identify the shipping modes and regions with the longest delays.

Assess the Impact of Shipping Delays on Customer Satisfaction

Understand the Regional and Seasonal Patterns of Delays

5. Exploratory Data Analysis (EDA) of Sales

Asking questions

- what are the top selling products in the superstore?
- what is the sales trend over time (monthly, yearly)?
- which category of products generates the highest Sales?
- which region generates the most sales?
- what is the average Sales for each product category?
- which sub-category of products has the highest demand?
- Total sales values by category and subcategory?
- Which are the most selling products in subcategory?
- Which customer segments are the most profitable ?
- What shipping modes sold the most products?
- Visualize the 'Category' column from the Shipmode column dataset standpoints.
- Which are the Top 10 country by sales?
- Who are the most profitable customers?
- Total sales values by year and month.eProfit/Loss incurred

6. Shipping Delay Problem Analysis

- Are there significant differences in shipping duration and delays between modes, and which mode consistently performs the worst?
- Which regions experience the most shipping delays across different ship modes?
- Are there particular customer segments, regions or product categories that are more affected by delays?
- Are there seasonal trends that affect shipping efficiency? Do certain times of the year experience more delays?
- Does a higher frequency of orders during certain periods (like holidays) correlate with an increase in shipping delays?
- Are there specific regions where shipping delays are more common for certain customer segments (e.g., Consumer, Corporate, Home Office)?

- Which cities experience the most shipping delays?
- Which customers experience the most shipping delays across different cities?
- Different stopped and continued customers and connect it with shipping delay

Analysis Steps

Data Cleaning and preprocessing

Data Cleaning

```
[207]: # Import the necessary libraries
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
```

```
[208]: # Load the dataset
data = pd.read_csv('Superstore Sales Dataset.csv')
data.head()
```

```
[209]: # Check for missing values
data.isnull().sum()
```

```
[209]: Row ID            0
Order ID          0
Order Date        0
Ship Date         0
Ship Mode         0
Customer ID       0
Customer Name     0
Segment           0
Country           0
City              0
State             0
Postal Code      11
Region            0
Product ID        0
Category          0
Sub-Category      0
Product Name      0
Sales             0
dtype: int64
```

```
[210]: data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9800 entries, 0 to 9799
Data columns (total 18 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Row ID          9800 non-null   int64
 1   Order ID        9800 non-null   object
 2   Order Date      9800 non-null   object
 3   Ship Date       9800 non-null   object
 4   Ship Mode       9800 non-null   object
```

[211]:
```python
# Find the rows where 'Postal Code' is missing
missing_postal_code = data[data['Postal Code'].isnull()]
print(missing_postal_code)
```

```
      Row ID        Order ID  Order Date   Ship Date       Ship Mode  \
2234    2235  CA-2018-104066  05/12/2018  10/12/2018  Standard Class
5274    5275  CA-2016-162887  07/11/2016  09/11/2016    Second Class
8798    8799  US-2017-150140  06/04/2017  10/04/2017  Standard Class
9146    9147  US-2017-165505  23/01/2017  27/01/2017  Standard Class
9147    9148  US-2017-165505  23/01/2017  27/01/2017  Standard Class
9148    9149  US-2017-165505  23/01/2017  27/01/2017  Standard Class
9386    9387  US-2018-127292  19/01/2018  23/01/2018  Standard Class
9387    9388  US-2018-127292  19/01/2018  23/01/2018  Standard Class
9388    9389  US-2018-127292  19/01/2018  23/01/2018  Standard Class
9389    9390  US-2018-127292  19/01/2018  23/01/2018  Standard Class
9741    9742  CA-2016-117086  08/11/2016  12/11/2016  Standard Class

     Customer ID     Customer Name      Segment        Country        City  \
2234    QJ-19255      Quincy Jones    Corporate  United States  Burlington
5274    SV-20785  Stewart Visinsky     Consumer  United States  Burlington
8798    VM-21685   Valerie Mitchum  Home Office  United States  Burlington
9146    CB-12535  Claudia Bergmann    Corporate  United States  Burlington
9147    CB-12535  Claudia Bergmann    Corporate  United States  Burlington
9148    CB-12535  Claudia Bergmann    Corporate  United States  Burlington
9386    RM-19375     Raymond Messe     Consumer  United States  Burlington
9387    RM-19375     Raymond Messe     Consumer  United States  Burlington
9388    RM-19375     Raymond Messe     Consumer  United States  Burlington
9389    RM-19375     Raymond Messe     Consumer  United States  Burlington
9741    QJ-19255      Quincy Jones    Corporate  United States  Burlington

        State  Postal Code Region       Product ID         Category  \
2234  Vermont          NaN   East  TEC-AC-10001013       Technology
5274  Vermont          NaN   East  FUR-CH-10000595        Furniture
8798  Vermont          NaN   East  TEC-PH-10002555       Technology
9146  Vermont          NaN   East  TEC-AC-10002926       Technology
9147  Vermont          NaN   East  OFF-AR-10003477  Office Supplies
9148  Vermont          NaN   East  OFF-ST-10001526  Office Supplies
```

[212]:
```python
# After search for postal code for Burlington city  -> 08016
# fill missing values in postal code
data['Postal Code'].fillna('08016', inplace=True)

#check missing values again
```

[213]:
```python
#check for duplicate rows
data.duplicated().sum()         #no duplicted rows
```

[213]:
```
0
```

[214]:
```python
# Ensure correct data types
# Convert 'Order Date' and 'Ship Date' to datetime format
data['Order Date'] = pd.to_datetime(data['Order Date'], dayfirst=True)
data['Ship Date'] = pd.to_datetime(data['Ship Date'], dayfirst=True)
```

[215]:
```python
# Convert 'Postal Code' column  to string    "treated as string data not numeric"
data['Postal Code'] = data['Postal Code'].astype(str)
```

[216]:
```python
#  Recheck for missing values and data types and columns after cleaning
missing_values_after_cleaning = data.isnull().sum()
print(missing_values_after_cleaning)
print("_____________________________")
print(data.dtypes)
print("_____________________________")
data.info()
print("_____________________________")
data.columns
```

```
Row ID           0
Order ID         0
Order Date       0
Ship Date        0
Ship Mode        0
Customer ID      0
Customer Name    0
Segment          0
Country          0
City             0
State            0
```

EDA and Sales Analysis Steps

In [359…
```
# Get the top 5 selling products
top_5_selling_products = top_selling_products[:5]
```

In [361…
```
top_5_selling_products
```

Out[361…
```
Product Name
Canon imageCLASS 2200 Advanced Copier                                         61599.824
Fellowes PB500 Electric Punch Plastic Comb Binding Machine with Manual Bind   27453.384
Cisco TelePresence System EX90 Videoconferencing Unit                         22638.480
HON 5400 Series Task Chairs for Big and Tall                                  21870.576
GBC DocuBind TL300 Electric Binding System                                    19823.479
Name: Sales, dtype: float64
```

In [363…
```
# Plot the top 5 selling products using Seaborn for colorful visualization
plt.figure(figsize=(10,6))  # Set the figure size
sns.barplot(x=top_5_selling_products.index, y=top_5_selling_products.values, palette="viridis")

# Add title and labels
plt.title("Top 5 Selling Products", fontsize=16)
plt.xlabel("Product Name", fontsize=14)
plt.ylabel("Total Sales", fontsize=14)

# Rotate x-axis labels for better readability
plt.xticks(rotation=90)

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

In [366…
```
df2.Region.value_counts()
```

Out[366…
```
Region
West       3140
East       2774
Central    2277
South      1598
Name: count, dtype: int64
```

In [368…
```
# Filter data for the specific product
product = df2[df2["Product Name"] == "Canon imageCLASS 2200 Advanced Copier"]

# Group the data by Region and calculate the mean for Sales
region_group = product.groupby(["Region"])[["Sales"]].mean()

# Sort the results from largest to smallest
region_group = region_group.sort_values(by="Sales", ascending=False)

# Plotting the results with a colorful palette
plt.figure(figsize=(10, 6))
sns.barplot(x=region_group.index, y='Sales', data=region_group, palette='viridis')
plt.title('Average Sales by Region for Canon imageCLASS 2200 Advanced Copier')
plt.xlabel('Region')
plt.ylabel('Average Sales')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

Out[374…

	Country	City	State	Region	Sales
0	United States	Henderson	Kentucky	South	261.9600
1	United States	Henderson	Kentucky	South	731.9400
2	United States	Los Angeles	California	West	14.6200
3	United States	Fort Lauderdale	Florida	South	957.5775
4	United States	Fort Lauderdale	Florida	South	22.3680

In [376…
```
for place in df_places.columns:
    print(place, ':', df_places[place].nunique())
```

```
Country : 1
City : 529
State : 48
Region : 4
Sales : 5750
```

In [379…
```
# Group the data by Region and calculate the total sales for each group
grouped_data = df_places.groupby(['Region'], as_index=False).sum()
grouped_data.sort_values(by='Sales', ascending=False, inplace=True)

# Plot the total sales generated by each region
plt.figure(figsize=(10, 5))

# Generate a list of colors based on the number of regions
colors = plt.cm.tab10(range(len(grouped_data)))

plt.bar(grouped_data["Region"], grouped_data['Sales'], color=colors, align='center')
plt.xlabel("Region")
plt.ylabel("Sales")
plt.title("Sales Generated by Region")
plt.xticks(rotation=90)

plt.show()
```

In [380…

```python
# Group the data by State and calculate the total sales for each group
grouped_data = df_places.groupby(['State'], as_index=False).sum()
grouped_data.sort_values(by='Sales', ascending=False, inplace=True)

# Plot the total sales generated by each state
plt.figure(figsize=(22, 10))

# Generate a list of colors using a different colormap
colors = plt.cm.coolwarm(range(len(grouped_data)))  # Using 'coolwarm' colormap

plt.bar(grouped_data["State"], grouped_data["Sales"], color=colors, align="center")
plt.xlabel("State")
plt.ylabel("Sales")
plt.title("Sales Generated by State")
plt.xticks(rotation=90)

plt.show()
```

Sales Generated by State

State

In [382…

```python
# Group the data by City and calculate the total sales for each city
grouped_data = df_places.groupby('City', as_index=False).sum()

# Sort the data by Sales in descending order
grouped_data.sort_values(by='Sales', ascending=False, inplace=True)

# Select the top 5 cities
top_5_cities = grouped_data.head()

# Create the bar plot
plt.figure(figsize=(12, 6))

# Generate a list of colors using a different colormap
colors = plt.cm.plasma(range(len(top_5_cities)))  # Using 'plasma' colormap

plt.bar(top_5_cities['City'], top_5_cities['Sales'], color=colors, align='center')
plt.xlabel("City")
plt.ylabel("Sales")
plt.title("Top 5 Cities by Sales")
plt.xticks(rotation=90)

plt.show()
```

Top 5 Cities by Sales

The best Sales of Products

In [385…

```python
# Group the data by products and category
avg_sales_by_category = df2.groupby('Category')['Sales'].mean()
avg_sales_by_category
```

Out[385…

```
Category
Furniture          348.525277
Office Supplies    119.128041
Technology         456.274096
Name: Sales, dtype: float64
```

In [387…

```python
# Custom color palette
custom_colors = ['#FF5733', '#33FF57', '#3357FF', '#F1C40F', '#8E44AD']  # Red, Green, Blue, Yellow, Purple

# Visualize through a graph
plt.figure(figsize=(12, 6))

# Create a bar plot with custom colors
avg_sales_by_category.plot(kind='bar', color=custom_colors)

plt.title("Average Sales by Product Category (Descending Order)", fontsize=16)
plt.xlabel("Product Category", fontsize=14)
plt.ylabel("Average Sales", fontsize=14)
plt.xticks(rotation=45)
plt.tight_layout()  # Adjust layout to make room for labels
plt.show()
```

In [389…

```python
# Get value counts of 'Sub-Category'
sub_category_counts = df2['Sub-Category'].value_counts()

# Custom color palette
custom_colors = ['#FF5733', '#33FF57', '#3357FF', '#F1C40F', '#8E44AD']  # Red, Green, Blue, Yellow, Purple

# Create a bar plot with custom colors
plt.figure(figsize=(10, 6))
sub_category_counts.plot(kind='bar', color=custom_colors)

plt.title("Value Counts of Sub-Category", fontsize=16)
plt.xlabel("Sub-Category", fontsize=14)
plt.ylabel("Counts", fontsize=14)
plt.xticks(rotation=90)  # Rotate x-axis labels for better visibility
plt.tight_layout()  # Adjust layout to make room for labels
plt.show()
```

Value Counts of Sub-Category

Segment

In [402…

```
# Group the data by 'Segment' and calculate total sales for each segment
total_sales_by_segment = df2.groupby('Segment')['Sales'].sum()

# Visualize through a graph
plt.figure(figsize=(10, 6))

# Custom color palette
custom_colors = ['#FF5733', '#33FF57', '#3357FF']  # Red, Green, Blue

# Create a bar plot with custom colors
total_sales_by_segment.plot(kind='bar', color=custom_colors)

plt.title("Total Sales by Customer Segment", fontsize=16)
plt.xlabel("Customer Segment", fontsize=14)
plt.ylabel("Total Sales", fontsize=14)
plt.xticks(rotation=0)  # Rotate x-axis labels for better visibility
plt.tight_layout()  # Adjust layout to make room for labels

# Add data labels on top of the bars
for index, value in enumerate(total_sales_by_segment):
    plt.text(index, value, f'${value:.2f}', ha='center', va='bottom', fontsize=12)

plt.show()
```

Total Sales by Customer Segment

1e6

In [391…

```
df2.Segment.value_counts()
```

Out[391…

```
Segment
Consumer       5096
Corporate      2948
Home Office    1745
Name: count, dtype: int64
```

In [394…

```
df2['Ship Mode'].value_counts()
```

Out[394…

```
Ship Mode
Standard Class    5849
Second Class      1901
First Class       1501
Same Day           538
Name: count, dtype: int64
```

In [396…

```
pivot_table = df2.pivot_table(index='Segment', columns='Ship Mode', values='Sales', aggfunc='sum')
```

In [398…

```
pivot_table
```

Out[398…

Ship Mode	First Class	Same Day	Second Class	Standard Class
Segment				
Consumer	158104.9470	57452.273	229410.3356	701740.5954
Corporate	102580.0539	45121.323	139045.2908	395465.1671
Home Office	84887.2564	22645.443	80743.3530	235411.3745

What is the sales trend over time (monthly, yearly)?

In [409…

```
# Convert 'Order Date' to datetime format
df['Order Date'] = pd.to_datetime(df['Order Date'], format='%d/%m/%Y')

# Monthly sales trend
monthly_sales = df.groupby(pd.Grouper(key='Order Date', freq='M'))[['Sales']].sum()

# Yearly sales trend
yearly_sales = df.groupby(pd.Grouper(key='Order Date', freq='Y'))[['Sales']].sum()

# Plotting monthly sales
plt.figure(figsize=(14, 6))

# Monthly Sales Plot
plt.subplot(1, 2, 1)
monthly_sales.plot(kind='line', marker='o', ax=plt.gca())
plt.title('Monthly Sales Trend')
plt.xlabel('Month')
plt.ylabel('Total Sales')
plt.xticks(rotation=45)
plt.grid()

# Yearly Sales Plot
plt.subplot(1, 2, 2)
yearly_sales.plot(kind='bar', ax=plt.gca())
plt.title('Yearly Sales Trend')
plt.xlabel('Year')
plt.ylabel('Total Sales')
plt.xticks(rotation=0)
plt.grid()

plt.tight_layout()
plt.show()
```

Data preparation for shipping delay Problem

Data preparation for shipping delay Problem

[217]:
```python
# Create new columns for year, month, and day from the 'Order Date'
data['order_year'] = data['Order Date'].dt.year
data['order_month'] = data['Order Date'].dt.month
data['order_day'] = data['Order Date'].dt.day


data.head()
```

[217]:

Customer ID	Customer Name	Segment	Country	City	...	Postal Code	Region	Product ID	Category	Sub-Category	Product Name	Sales	order_year	order_month	order_day
CG-12520	Claire Gute	Consumer	United States	Henderson	...	42420.0	South	FUR-BO-10001798	Furniture	Bookcases	Bush Somerset Collection Bookcase	261.9600	2017	11	8
CG-12520	Claire Gute	Consumer	United States	Henderson	...	42420.0	South	FUR-CH-10000454	Furniture	Chairs	Hon Deluxe Fabric Upholstered Stacking Chairs,...	731.9400	2017	11	8
DV-13045	Darrin Van Huff	Corporate	United States	Los Angeles	...	90036.0	West	OFF-LA-10000240	Office Supplies	Labels	Self-Adhesive Address Labels for Typewriters b...	14.6200	2017	6	12
SO-20335	Sean O'Donnell	Consumer	United States	Fort Lauderdale	...	33311.0	South	FUR-TA-10000577	Furniture	Tables	Bretford CR4500 Series Slim Rectangular Table	957.5775	2016	10	11

1.Calculate the Shipping Duration: This will be the difference between the Ship Date and Order Date.
2.Find the Mode (statistics): For each shipping mode, calculate the mode of the shipping duration.
3.Create Shipping Delay Column: Compare the shipping duration with the mode for that shipping mode and flag as "delayed" if the duration exceeds the mode.

[219]:
```python
# Step 1: Calculate shipping Duration
data['Shipping Duration'] = (data['Ship Date'] - data['Order Date']).dt.days

# Step 2: Group by 'Ship Mode' and calculate the mode of shipping duration for each 'Ship Mode'
mode_duration_by_ship_mode = data.groupby('Ship Mode')['Shipping Duration'].agg(lambda x: x.mode()[0]).reset_index()
# Step 3: Sort the results by the shipping mode duration
mode_duration_by_ship_mode = mode_duration_by_ship_mode.sort_values(by='Shipping Duration', ascending=False).reset_index()


mode_duration_by_ship_mode
```

[219]:

	index	Ship Mode	Shipping Duration
0	3	Standard Class	4
1	0	First Class	3
2	2	Second Class	2
3	1	Same Day	0

[220]:
```python
# Create Shipping Delay Column
import numpy as np

# Step 1: Define conditions for shipping delay based on the ship mode and the predefined mode
conditions = [
    (data['Ship Mode'] == 'Standard Class') & (data['Shipping Duration'] > 4),
    (data['Ship Mode'] == 'Second Class') & (data['Shipping Duration'] > 2),
    (data['Ship Mode'] == 'First Class') & (data['Shipping Duration'] > 3),
    (data['Ship Mode'] == 'Same Day') & (data['Shipping Duration'] > 0)
]

# Step 2: Define the corresponding delay calculation for each condition
delay_values = [
    data['Shipping Duration'] - 4,
    data['Shipping Duration'] - 2,
    data['Shipping Duration'] - 3,
    data['Shipping Duration'] - 0
]

# Step 3: Use np.select() to create the 'Shipping Delay' column
data['Shipping Delay'] = np.select(conditions, delay_values, default=0)

# Step 5: View the updated data with 'Shipping Delay' column
data.head()
```

[220]:

:ustomer ID	Customer Name	Segment	Country	City	...	Product ID	Category	Sub-Category	Product Name	Sales	order_year	order_month	order_day	Shipping Duration	Shipping Delay
:G-12520	Claire Gute	Consumer	United States	Henderson	...	FUR-BO-10001798	Furniture	Bookcases	Bush Somerset Collection	261.9600	2017	11	8	3	1

Shipping Delay Problem Analysis (code)

Are there significant differences in shipping duration and delays between modes, and which mode consistently performs the worst?

General overview of performance (average shipping duration).

[222]:
```python
# Step 1: Calculate average shipping duration for each shipping mode
avg_shipping_duration = data.groupby('Ship Mode')['Shipping Duration'].mean().reset_index()

avg_shipping_duration = avg_shipping_duration.sort_values(by='Shipping Duration', ascending=False)

# Step 2: Identify the shipping mode with the highest average shipping duration
worst_mode = avg_shipping_duration.loc[avg_shipping_duration['Shipping Duration'].idxmax()]

# Step 3: Print the average shipping durations and the worst mode
print("Average Shipping Durations by Mode:")
print(avg_shipping_duration)
print("\nShipping Mode that Performs the Worst:")
print(worst_mode)
```

```
Average Shipping Durations by Mode:
        Ship Mode  Shipping Duration
3  Standard Class           5.008363
2    Second Class           3.249211
0     First Class           2.179214
1        Same Day           0.044610

Shipping Mode that Performs the Worst:
Ship Mode            Standard Class
Shipping Duration          5.008363
Name: 3, dtype: object
```


```python
avg_shipping_duration = avg_shipping_duration.sort_values(by='Shipping Duration', ascending=False)

# Step 2: Identify the shipping mode with the highest average shipping duration
worst_mode = avg_shipping_duration.loc[avg_shipping_duration['Shipping Duration'].idxmax()]

# Step 3: Print the average shipping durations and the worst mode
print("Average Shipping Durations by Mode:")
print(avg_shipping_duration)
print("\nShipping Mode that Performs the Worst:")
print(worst_mode)
```

```
Average Shipping Durations by Mode:
        Ship Mode  Shipping Duration
3  Standard Class           5.008363
2    Second Class           3.249211
0     First Class           2.179214
1        Same Day           0.044610

Shipping Mode that Performs the Worst:
Ship Mode            Standard Class
Shipping Duration          5.008363
Name: 3, dtype: object
```


[223]:
```python
#Visualize the average shipping durations with a bar chart
plt.figure(figsize=(8, 6))
plt.bar(avg_shipping_duration['Ship Mode'], avg_shipping_duration['Shipping Duration'], color='lightcoral')
plt.title('Average Shipping Duration by Shipping Mode')
plt.xlabel('Shipping Mode')
plt.ylabel('Average Shipping Duration (Days)')
plt.xticks(rotation=45)
plt.grid(axis='y')
plt.tight_layout()
plt.show()
```

Average Shipping Duration by Shipping Mode

Average shipping delay

[224]:
```python
delayed_orders = data[data['Shipping Delay'] > 0]

# Calculate the average shipping delay by Ship Mode for orders that were delayed
average_shipping_delay = delayed_orders.groupby('Ship Mode')['Shipping Delay'].mean().reset_index()

# Sort the results for better readability
average_shipping_delay = average_shipping_delay.sort_values(by='Shipping Delay', ascending=False)

# Display the results
print(average_shipping_delay)
```

```
        Ship Mode  Shipping Delay
2    Second Class        2.043852
3  Standard Class        1.683956
0     First Class        1.000000
1        Same Day        1.000000
```

[225]:
```python
# Plotting the average shipping delay by Ship Mode
plt.figure(figsize=(8, 6))
plt.bar(average_shipping_delay['Ship Mode'], average_shipping_delay['Shipping Delay'], color='lightblue')
plt.title('Average Shipping Delay by Ship Mode')
plt.xlabel('Shipping Mode')
plt.ylabel('Average Shipping Delay (Days)')
plt.xticks(rotation=45)
plt.grid(axis='y')

# Show the plot
plt.tight_layout()
plt.show()
```

Average Shipping Delay by Ship Mode

Number of orderes delayed for each ship mode

[226]:
```python
# Count the number of delayed orders by Ship Mode
delayed_orders_count = delayed_orders.groupby('Ship Mode')['Order ID'].count().reset_index()

# Rename the columns for clarity
delayed_orders_count.columns = ['Ship Mode', 'Number of Delayed Orders']

# Sort the results for better readability
delayed_orders_count = delayed_orders_count.sort_values(by='Number of Delayed Orders', ascending=False)

delayed_orders_count

# Another way
#common_ship_mode = delayed_orders['Ship Mode'].value_counts()

# Display the result
#print("Most Common Ship Mode for Delayed Orders:\n", common_ship_mode)
```

[226]:

	Ship Mode	Number of Delayed Orders
3	Standard Class	3509
2	Second Class	1163
1	Same Day	24
0	First Class	1


```python
# Rename the columns for clarity
delayed_orders_count.columns = ['Ship Mode', 'Number of Delayed Orders']

# Sort the results for better readability
delayed_orders_count = delayed_orders_count.sort_values(by='Number of Delayed Orders', ascending=False)

delayed_orders_count

# Another way
#common_ship_mode = delayed_orders['Ship Mode'].value_counts()

# Display the result
#print("Most Common Ship Mode for Delayed Orders:\n", common_ship_mode)
```

[226]:

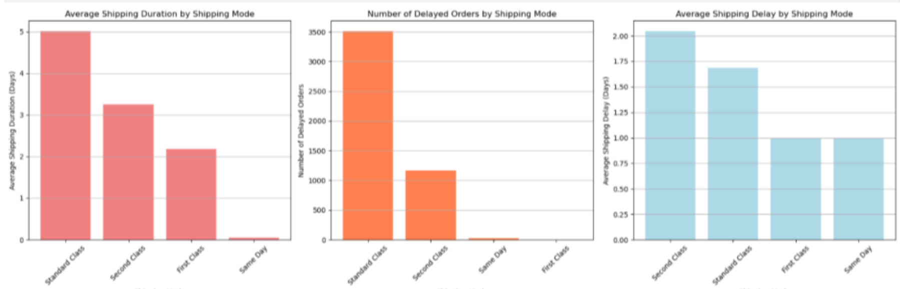
	Ship Mode	Number of Delayed Orders
3	Standard Class	3509
2	Second Class	1163
1	Same Day	24
0	First Class	1

[227]:
```python
# Plotting the number of delayed orders by Ship Mode
plt.figure(figsize=(8, 6))
plt.bar(delayed_orders_count['Ship Mode'], delayed_orders_count['Number of Delayed Orders'], color='coral')
plt.title('Number of Delayed Orders by Ship Mode')
plt.xlabel('Shipping Mode')
plt.ylabel('Number of Delayed Orders')
plt.xticks(rotation=45)
plt.grid(axis='y')

plt.tight_layout()
plt.show()
```

[254]:

```python
import matplotlib.pyplot as plt

# Create a figure and a 1x3 grid of subplots
fig, axes = plt.subplots(1, 3, figsize=(18, 6))  # 1 row, 3 columns

# Subplot 1: Average Shipping Duration by Shipping Mode
axes[0].bar(avg_shipping_duration['Ship Mode'], avg_shipping_duration['Shipping Duration'], color='lightcoral')
axes[0].set_title('Average Shipping Duration by Shipping Mode')
axes[0].set_xlabel('Shipping Mode')
axes[0].set_ylabel('Average Shipping Duration (Days)')
axes[0].tick_params(axis='x', rotation=45)
axes[0].grid(axis='y')

# Subplot 2: Number of Delayed Orders by Shipping Mode
axes[1].bar(delayed_orders_count['Ship Mode'], delayed_orders_count['Number of Delayed Orders'], color='coral')
axes[1].set_title('Number of Delayed Orders by Shipping Mode')
axes[1].set_xlabel('Shipping Mode')
axes[1].set_ylabel('Number of Delayed Orders')
axes[1].tick_params(axis='x', rotation=45)
axes[1].grid(axis='y')

# Subplot 3: Average Shipping Delay by Shipping Mode
axes[2].bar(average_shipping_delay['Ship Mode'], average_shipping_delay['Shipping Delay'], color='lightblue')
axes[2].set_title('Average Shipping Delay by Shipping Mode')
axes[2].set_xlabel('Shipping Mode')
axes[2].set_ylabel('Average Shipping Delay (Days)')
axes[2].tick_params(axis='x', rotation=45)
axes[2].grid(axis='y')

# Adjust the layout to avoid overlap
plt.tight_layout()

# Show the figure with subplots
plt.show()
```


```python
axes[2].bar(average_shipping_delay['Ship Mode'], average_shipping_delay['Shipping Delay'], color='lightblue')
axes[2].set_title('Average Shipping Delay by Shipping Mode')
axes[2].set_xlabel('Shipping Mode')
axes[2].set_ylabel('Average Shipping Delay (Days)')
axes[2].tick_params(axis='x', rotation=45)
axes[2].grid(axis='y')

# Adjust the layout to avoid overlap
plt.tight_layout()

# Show the figure with subplots
plt.show()
```

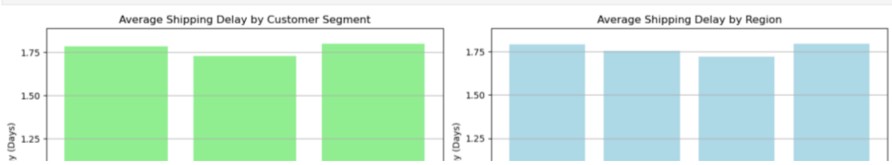
Are there particular customer segments or regions that are more affected by delays?

[228]:
```
#Steps to Analyze:

#Average Shipping Delay by Customer Segment:
#Calculate the average shipping delay for each customer segment.
#Visualize the results using a bar chart to compare how delays impact different segments.

#Average Shipping Delay by Region:
#Calculate the average shipping delay for each region.
#Visualize the results with a bar chart to see how regions are affected by shipping delays.
```

[229]:
```
#Calculate average shipping delay by Customer Segment
avg_delay_by_segment = delayed_orders .groupby('Segment')['Shipping Delay'].mean().reset_index()

# Calculate average shipping delay by Region
avg_delay_by_region = delayed_orders .groupby('Region')['Shipping Delay'].mean().reset_index()

# Plot the results
plt.figure(figsize=(14, 6))

# Subplot 1: Average Shipping Delay by Customer Segment
plt.subplot(1, 2, 1)
plt.bar(avg_delay_by_segment['Segment'], avg_delay_by_segment['Shipping Delay'], color='lightgreen')
plt.title('Average Shipping Delay by Customer Segment')
plt.xlabel('Customer Segment')
plt.ylabel('Average Shipping Delay (Days)')
plt.grid(axis='y')

# Subplot 2: Average Shipping Delay by Region
plt.subplot(1, 2, 2)
plt.bar(avg_delay_by_region['Region'], avg_delay_by_region['Shipping Delay'], color='lightblue')
plt.title('Average Shipping Delay by Region')
plt.xlabel('Region')
plt.ylabel('Average Shipping Delay (Days)')
```


```
avg_delay_by_region = delayed_orders .groupby('Region')['Shipping Delay'].mean().reset_index()

# Plot the results
plt.figure(figsize=(14, 6))

# Subplot 1: Average Shipping Delay by Customer Segment
plt.subplot(1, 2, 1)
plt.bar(avg_delay_by_segment['Segment'], avg_delay_by_segment['Shipping Delay'], color='lightgreen')
plt.title('Average Shipping Delay by Customer Segment')
plt.xlabel('Customer Segment')
plt.ylabel('Average Shipping Delay (Days)')
plt.grid(axis='y')

# Subplot 2: Average Shipping Delay by Region
plt.subplot(1, 2, 2)
plt.bar(avg_delay_by_region['Region'], avg_delay_by_region['Shipping Delay'], color='lightblue')
plt.title('Average Shipping Delay by Region')
plt.xlabel('Region')
plt.ylabel('Average Shipping Delay (Days)')
```


```
plt.grid(axis='y')

# Show the plot
plt.tight_layout()
plt.show()
```



Are there seasonal trends that affect shipping efficiency? Do certain times of the year experience more delays?

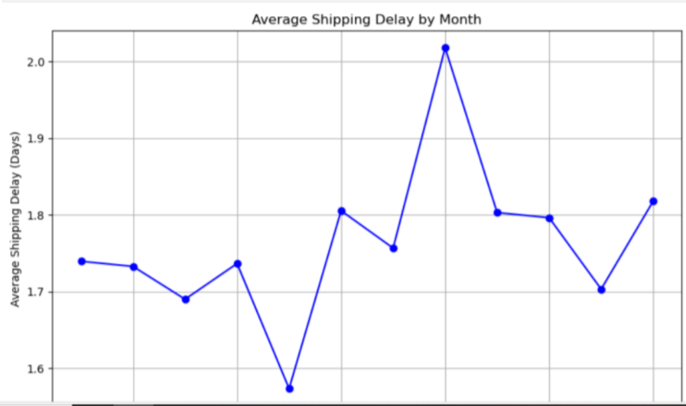
[ ]:
```
# Step 1: Calculate Average Shipping Delay by Month
avg_delay_by_month = delayed_orders.groupby('order_month')['Shipping Delay'].mean().reset_index()

# Step 2: Calculate Average Shipping Delay by Day of the Month
avg_delay_by_day = delayed_orders.groupby('order_day')['Shipping Delay'].mean().reset_index()

# Step 3: Calculate Average Shipping Delay by Quarter
delayed_orders['order_quarter'] = delayed_orders['Order Date'].dt.quarter
avg_delay_by_quarter = delayed_orders.groupby('order_quarter')['Shipping Delay'].mean().reset_index()
```

[231]:
```
# Step 4: Plot the Results

# Create the figure for subplots
plt.figure(figsize=(18, 6))

# Subplot 1: Average Shipping Delay by Month
plt.subplot(1, 3, 1)
avg_delay_by_month.plot(x='order_month', y='Shipping Delay', kind='bar', color='lightcoral', ax=plt.gca())
plt.title('Average Shipping Delay by Month')
plt.xlabel('Month')
plt.ylabel('Average Shipping Delay (Days)')
plt.xticks(rotation=45)
plt.grid(True)

# Subplot 2: Average Shipping Delay by Day of the Month
plt.subplot(1, 3, 2)
plt.bar(avg_delay_by_day['order_day'], avg_delay_by_day['Shipping Delay'], color='lightblue')
plt.title('Average Shipping Delay by Day of the Month')
plt.xlabel('Day of the Month')
```

```python
plt.subplot(1, 3, 1)
avg_delay_by_month.plot(x='order_month', y='Shipping Delay', kind='bar', color='lightcoral', ax=plt.gca())
plt.title('Average Shipping Delay by Month')
plt.xlabel('Month')
plt.ylabel('Average Shipping Delay (Days)')
plt.xticks(rotation=45)
plt.grid(True)

# Subplot 2: Average Shipping Delay by Day of the Month
plt.subplot(1, 3, 2)
plt.bar(avg_delay_by_day['order_day'], avg_delay_by_day['Shipping Delay'], color='lightblue')
plt.title('Average Shipping Delay by Day of the Month')
plt.xlabel('Day of the Month')
plt.ylabel('Average Shipping Delay (Days)')
plt.xticks(rotation=45)
plt.grid(True)

# Subplot 3: Average Shipping Delay by Quarter of the Year
plt.subplot(1, 3, 3)
avg_delay_by_quarter.plot(x='order_quarter', y='Shipping Delay', kind='bar', color='lightgreen', ax=plt.gca())
plt.title('Average Shipping Delay by Quarter')
plt.xlabel('Quarter')
plt.ylabel('Average Shipping Delay (Days)')
plt.xticks(ticks=[0, 1, 2, 3], labels=['Q1', 'Q2', 'Q3', 'Q4'], rotation=0)
plt.grid(True)

# Show the plots
plt.tight_layout()
plt.show()
```

[232]:

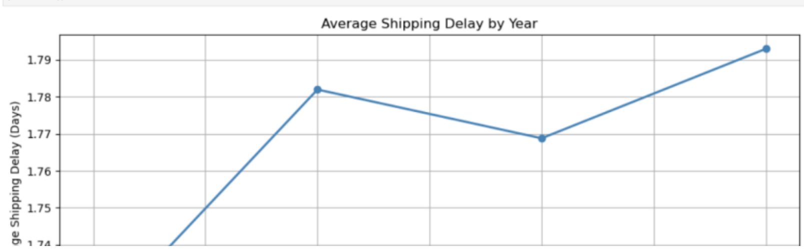
```python
plt.figure(figsize=(10, 6))
plt.plot(avg_delay_by_month['order_month'], avg_delay_by_month['Shipping Delay'], marker='o', color='blue')
plt.title('Average Shipping Delay by Month')
plt.xlabel('Month')
plt.ylabel('Average Shipping Delay (Days)')
plt.xticks(rotation=45)
plt.grid(True)
```



Average Shipping delay over Years

[235]:

```python
# Step 1: Calculate Average Shipping Delay by Year
avg_delay_by_year = delayed_orders.groupby('order_year')['Shipping Delay'].mean().reset_index()

# Step 3: Plot the Results as a Line Chart
plt.figure(figsize=(10, 5))
plt.plot(avg_delay_by_year['order_year'], avg_delay_by_year['Shipping Delay'], marker='o', color='steelblue', linestyle='-', linewidth=2)
plt.title('Average Shipping Delay by Year')
plt.xlabel('Year')
plt.ylabel('Average Shipping Delay (Days)')
plt.xticks(rotation=45)
plt.grid(True)

# Show the plot
plt.tight_layout()
plt.show()
```



Which regions experience the most shipping delays across different ship modes?

[236]:
```python
# Group by Region and Ship Mode, then calculate the average shipping delay
avg_delay_by_region_mode = data.groupby(['Region', 'Ship Mode'])['Shipping Delay'].mean().reset_index()

# Sort the data by average shipping delay in descending order
avg_delay_by_region_mode_sorted = avg_delay_by_region_mode.sort_values(by='Shipping Delay', ascending=False)

# Identify the regions with the highest average shipping delays
top_delays = avg_delay_by_region_mode_sorted.groupby('Ship Mode').head(5)

# Step 3: Plotting
plt.figure(figsize=(14, 8))

# Create a bar plot for each shipping mode
for i, ship_mode in enumerate(top_delays['Ship Mode'].unique()):
    plt.subplot(2, 2, i + 1)
    mode_data = top_delays[top_delays['Ship Mode'] == ship_mode]
    plt.bar(mode_data['Region'], mode_data['Shipping Delay'], color='lightblue')
    plt.title(f'Average Shipping Delay by Region ({ship_mode})')
    plt.xlabel('Region')
    plt.ylabel('Average Shipping Delay (Days)')
    plt.xticks(rotation=45)
    plt.grid(True)

# Show the plots
plt.tight_layout()
plt.show()
```

Does a higher frequency of orders during certain periods (like holidays) correlate with an increase in shipping delays?

[263]:
```python
# Calculate the number of orders for all data
monthly_orders = data.groupby('order_month').agg(num_orders=('Order ID', 'count')).reset_index()

# Calculate the average shipping delay for delayed orders
monthly_delay = delayed_orders.groupby('order_month').agg(avg_shipping_delay=('Shipping Delay', 'mean')).reset_index()

# Merge the two results to have both num_orders and avg_shipping_delay in the same dataframe
monthly_summary = pd.merge(monthly_orders, monthly_delay, on='order_month', how='left')

monthly_summary
```

[263]:

	order_month	num_orders	avg_shipping_delay
0	1	366	1.739583
1	2	297	1.732919
2	3	680	1.690141
3	4	657	1.736686
4	5	725	1.573620
5	6	691	1.805556
6	7	697	1.756839
7	8	693	2.018018
8	9	1354	1.803101
9	10	809	1.796392
10	11	1449	1.703281
11	12	1382	1.818047

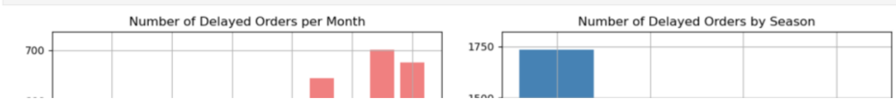
[264]:
```python
# Step 2: Plot the results

# Create a figure
plt.figure(figsize=(12, 6))

# Subplot 1: Number of Orders
plt.subplot(1, 2, 1)
plt.bar(monthly_summary['order_month'], monthly_summary['num_orders'], color='coral')
plt.title('Number of Orders per Month')
plt.xlabel('Month')
plt.ylabel('Number of Orders')
plt.xticks(rotation=45)
plt.grid()

# Subplot 2: Average Shipping Delay
plt.subplot(1, 2, 2)
plt.plot(monthly_summary['order_month'], monthly_summary['avg_shipping_delay'], marker='o', color='orange')
plt.title('Average Shipping Delay per Month')
plt.xlabel('Month')
plt.ylabel('Average Shipping Delay (Days)')
plt.xticks(rotation=45)
plt.grid()

# Show the plots
plt.tight_layout()
plt.show()
```

Are delays more frequent during specific months or seasons?

[239]:
```python
# Step 1: Calculate the number of delayed orders per month
monthly_delays = delayed_orders.groupby('order_month').size().reset_index(name='num_delayed_orders')

# Step 2: Add season column based on order_month
season_mapping = {
    1: 'Winter', 2: 'Winter', 3: 'Spring',
    4: 'Spring', 5: 'Spring', 6: 'Summer',
    7: 'Summer', 8: 'Summer', 9: 'Fall',
    10: 'Fall', 11: 'Fall', 12: 'Winter'
}
monthly_delays['season'] = monthly_delays['order_month'].map(season_mapping)

# Step 3: Calculate the number of delayed orders by season
seasonal_delays = monthly_delays.groupby('season')['num_delayed_orders'].sum().reset_index()

# Step 4: Plot the results

# Create a figure
plt.figure(figsize=(12, 6))

# Bar plot for monthly delays
plt.subplot(1, 2, 1)
plt.bar(monthly_delays['order_month'], monthly_delays['num_delayed_orders'], color='lightcoral')
plt.title('Number of Delayed Orders per Month')
plt.xlabel('Month')
plt.ylabel('Number of Delayed Orders')
plt.xticks(rotation=45)
plt.grid()

# Bar plot for seasonal delays
plt.subplot(1, 2, 2)
```

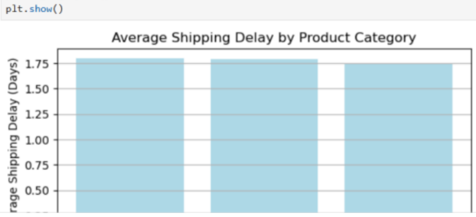

```python
# Step 3: Calculate the number of delayed orders by season
seasonal_delays = monthly_delays.groupby('season')['num_delayed_orders'].sum().reset_index()

# Step 4: Plot the results

# Create a figure
plt.figure(figsize=(12, 6))

# Bar plot for monthly delays
plt.subplot(1, 2, 1)
plt.bar(monthly_delays['order_month'], monthly_delays['num_delayed_orders'], color='lightcoral')
plt.title('Number of Delayed Orders per Month')
plt.xlabel('Month')
plt.ylabel('Number of Delayed Orders')
plt.xticks(rotation=45)
plt.grid()

# Bar plot for seasonal delays
plt.subplot(1, 2, 2)
```


```python
plt.bar(seasonal_delays['season'], seasonal_delays['num_delayed_orders'], color='steelblue')
plt.title('Number of Delayed Orders by Season')
plt.xlabel('Season')
plt.ylabel('Number of Delayed Orders')
plt.grid()

# Show the plots
plt.tight_layout()
plt.show()
```



How does the shipping delay vary with different product categories?

[266]:
```python
# Filter out rows where Shipping Delay is 0 (i.e., no delays)
delayed_orders = data[data['Shipping Delay'] > 0]

# Step 1: Group by Product Category and calculate average shipping delay
avg_delay_by_category = delayed_orders.groupby('Category')['Shipping Delay'].mean().reset_index()

# Step 2: Sort values for better visualization
avg_delay_by_category = avg_delay_by_category.sort_values(by='Shipping Delay', ascending=False)

# Step 3: Plot the results
plt.figure(figsize=(6, 4))
plt.bar(avg_delay_by_category['Category'], avg_delay_by_category['Shipping Delay'], color='lightblue')
plt.title('Average Shipping Delay by Product Category')
plt.xlabel('Product Category')
plt.ylabel('Average Shipping Delay (Days)')
plt.xticks(rotation=45)
plt.grid(axis='y')
plt.tight_layout()
plt.show()
```



Are there specific regions where shipping delays are more common for certain customer segments (e.g., Consumer, Corporate, Home Office)?

[241]:
```python
# Step 1: Group by Region and Customer Segment, and calculate average shipping delay
avg_delay_by_region_segment = (
    delayed_orders.groupby(['Region', 'Segment'])['Shipping Delay']
    .mean()
    .reset_index()
)

# Step 2: Pivot the data for better visualization
pivot_avg_delay = avg_delay_by_region_segment.pivot(index='Region', columns='Segment', values='Shipping Delay')

pivot_avg_delay
```

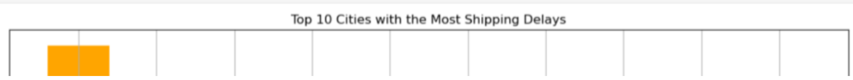
[241]:

Segment	Consumer	Corporate	Home Office
Region			
Central	1.822262	1.679525	1.885965
East	1.816143	1.631579	1.776744
South	1.712264	1.734694	1.716667
West	1.766879	1.845511	1.774059

[267]:
```python
# Step 3: Plot the results
plt.figure(figsize=(8, 6))
pivot_avg_delay.plot(kind='bar', color=['lightblue', 'lightgreen', 'lightcoral'], figsize=(12, 6))
plt.title('Average Shipping Delay by Region and Customer Segment')
plt.xlabel('Region')
plt.ylabel('Average Shipping Delay (Days)')
plt.xticks(rotation=45)
plt.grid(axis='y')
```

Which cities experience the most shipping delays?

[245]:
```python
# Step 2: Group by 'City' to get the count of delayed orders and average shipping delay
city_delay_analysis = delayed_orders.groupby('City').agg(
    num_delayed_orders=('Order ID', 'count'),      # Count the number of delayed orders
    avg_shipping_delay=('Shipping Delay', 'mean')  # Calculate average shipping delay
).reset_index()

# Step 3: Sort by the number of delayed orders to find the cities with the most delays
top_cities_with_delays = city_delay_analysis.sort_values(by='num_delayed_orders', ascending=False).head(10)

# Step 4: Visualize the results

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

# Bar plot for the top 10 cities with the most delayed orders
plt.bar(top_cities_with_delays['City'], top_cities_with_delays['num_delayed_orders'], color='orange')

# Add title and labels
plt.title('Top 10 Cities with the Most Shipping Delays')
plt.xlabel('City')
plt.ylabel('Number of Delayed Orders')
plt.xticks(rotation=45)
plt.grid(True)
plt.tight_layout()

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



Top 10 States with the highest Average shipping delays?

[247]:
```python
# Step 1: Filter out rows where Shipping Delay is 0 (i.e., no delays)
delayed_orders = data[data['Shipping Delay'] > 0]

# Step 2: Group by 'State' to calculate the average shipping delay
state_avg_delay = delayed_orders.groupby('State').agg(
    avg_shipping_delay=('Shipping Delay', 'mean')  # Average shipping delay per state
).reset_index()

# Step 3: Sort the states by average shipping delay
top_states_by_avg_delay = state_avg_delay.sort_values(by='avg_shipping_delay', ascending=False).head(10)

# Step 4: Plot the results
plt.figure(figsize=(12, 6))

# Bar plot for average shipping delay
plt.bar(top_states_by_avg_delay['State'], top_states_by_avg_delay['avg_shipping_delay'], color='purple')

# Add titles and labels
plt.title('Top 10 States with the Highest Average Shipping Delay')
plt.xlabel('State')
plt.ylabel('Average Shipping Delay (Days)')

# Rotate the x-axis labels for better readability
plt.xticks(rotation=45)

# Add gridlines for clarity
plt.grid(True)

# Show the plot
plt.tight_layout()
plt.show()
```

Which customers experience the most shipping delays across different cities?

```
[248]: # Step 1: Filter out rows where Shipping Delay is 0 (i.e., no delays)
delayed_orders = data[data['Shipping Delay'] > 0]

# Step 2: Group by 'Customer Name' and 'City' to calculate the total and average shipping delay
customer_city_delay = delayed_orders.groupby(['Customer Name', 'City']).agg(
    total_shipping_delay=('Shipping Delay', 'sum'),  # Total shipping delay per customer per city
    avg_shipping_delay=('Shipping Delay', 'mean'),   # Average shipping delay per customer per city
    num_delayed_orders=('Shipping Delay', 'count')   # Number of delayed orders per customer per city
).reset_index()

# Step 3: Sort the customers by total shipping delay
top_customers_by_delay = customer_city_delay.sort_values(by='total_shipping_delay', ascending=False).head(10)

# Step 4: Plot the results
plt.figure(figsize=(12, 6))

# Bar plot for total shipping delay for top 10 customers across cities
plt.barh(top_customers_by_delay['Customer Name'] + ' (' + top_customers_by_delay['City'] + ')',
         top_customers_by_delay['total_shipping_delay'], color='lightblue')

# Add titles and labels
plt.title('Top 10 Customers with the Most Shipping Delays Across Different Cities')
plt.xlabel('Total Shipping Delay (Days)')
plt.ylabel('Customer (City)')

# Rotate the y-axis labels for better readability
plt.yticks(rotation=0)

# Add gridlines for clarity
plt.grid(True)

# Show the plot
```

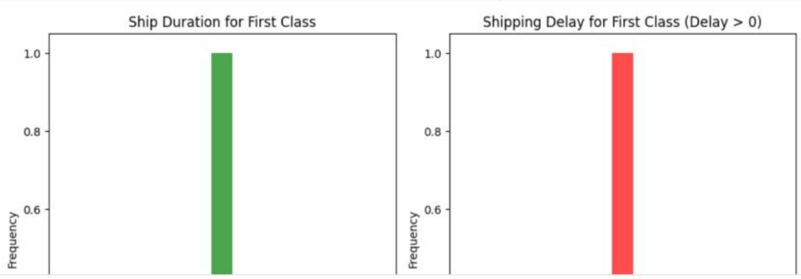
```
•[282]: ######################   Analysis For Customers ##########################################

# Step 1: Identify the last purchase year for each customer
latest_purchase = data.groupby('Customer ID')['order_year'].max().reset_index()
latest_purchase.columns = ['Customer ID', 'Last_Purchase_Year']

# Step 2: Define the cutoff year to determine "stopped" and "continued" customers
cutoff_year = 2018
stopped_customers = latest_purchase[latest_purchase['Last_Purchase_Year'] < cutoff_year]
continued_customers = latest_purchase[latest_purchase['Last_Purchase_Year'] >= cutoff_year]

# Step 3: Filter the dataset for stopped and continued customers
stopped_customers_orders = data[data['Customer ID'].isin(stopped_customers['Customer ID'])]
continued_customers_orders = data[data['Customer ID'].isin(continued_customers['Customer ID'])]

# Step 4: Calculate the average shipping delay for both stopped and continued customers
avg_delay_stopped = stopped_customers_orders['Shipping Delay'].mean()
avg_delay_continued = continued_customers_orders['Shipping Delay'].mean()

# Ensure the average shipping delays are greater than zero
avg_delay_stopped = max(avg_delay_stopped, 1)
avg_delay_continued = max(avg_delay_continued, 1)

#       : Visualize the comparison of average shipping delays for stopped and continued customers
delay_data = pd.DataFrame({
    'Customer_Type': ['Stopped', 'Continued'],
    'Avg_Shipping_Delay': [avg_delay_stopped, avg_delay_continued]
})

plt.figure(figsize=(8,6))
plt.bar(delay_data['Customer_Type'], delay_data['Avg_Shipping_Delay'], color=['red', 'green'])
plt.xlabel('Customer Type')
plt.ylabel('Average Shipping Delay (Days)')
plt.title('Comparison of Shipping Delays: Stopped vs Continued Customers')
plt.show()
```

```
})

plt.figure(figsize=(8,6))
plt.bar(delay_data['Customer_Type'], delay_data['Avg_Shipping_Delay'], color=['red', 'green'])
plt.xlabel('Customer Type')
plt.ylabel('Average Shipping Delay (Days)')
plt.title('Comparison of Shipping Delays: Stopped vs Continued Customers')
plt.show()

# Step 5: Analyze customer retention over time (for stopped and continued customers)
stopped_customers_per_year = stopped_customers_orders.groupby('order_year')['Customer ID'].nunique()
continued_customers_per_year = continued_customers_orders.groupby('order_year')['Customer ID'].nunique()

# Step 6: Plot trends in customer retention over time
plt.figure(figsize=(10,6))
plt.plot(stopped_customers_per_year.index, stopped_customers_per_year, label='Stopped Customers', color='red', marker='o')
plt.plot(continued_customers_per_year.index, continued_customers_per_year, label='Continued Customers', color='green', marker='o')
plt.ylabel('Number of Customers')
plt.xlabel('Year')
plt.title('Customer Retention Over Time: Stopped vs Continued Customers')
plt.legend()
plt.show()
```

```
# Step 8: Analyze shipping delay distributions for stopped and continued customers
plt.figure(figsize=(12, 6))

# Shipping Delay for Stopped Customers
plt.subplot(1, 2, 1)
plt.hist(stopped_customers_orders['Shipping Delay'], bins=15, color='salmon', alpha=0.7)
plt.title('Shipping Delay for Stopped Customers')
plt.xlabel('Days')
plt.ylabel('Frequency')

# Shipping Delay for Continued Customers
plt.subplot(1, 2, 2)
plt.hist(continued_customers_orders['Shipping Delay'], bins=15, color='green', alpha=0.7)
plt.title('Shipping Delay for Continued Customers')
plt.xlabel('Days')
plt.ylabel('Frequency')

plt.tight_layout()
plt.show()
```

```python
import matplotlib.pyplot as plt

# Let's assume your DataFrame is called df and has 'Ship Mode', 'Ship Duration', and 'Shipping Delay' columns.

# List of ship modes to iterate over
ship_modes = ['First Class', 'Same Day', 'Second Class', 'Standard Class']

# Create individual plots for each ship mode
for mode in ship_modes:
    # Filter the dataset for the current shipping mode and for Shipping Delay > 0
    df_mode = df[(df['Ship Mode'] == mode) & (df['Shipping Delay'] > 0)]

    # Create a figure for the current mode
    plt.figure(figsize=(10,6))

    # Plot Shipping Duration for this mode
    plt.subplot(1, 2, 1)
    plt.hist(df_mode['Ship Duration'], bins=15, color='g', alpha=0.7)
    plt.title(f'Ship Duration for {mode}')
    plt.xlabel('Days')
    plt.ylabel('Frequency')

    # Plot Shipping Delay for this mode (only for delays > 0)
    plt.subplot(1, 2, 2)
    plt.hist(df_mode['Shipping Delay'], bins=15, color='r', alpha=0.7)
    plt.title(f'Shipping Delay for {mode} (Delay > 0)')
    plt.xlabel('Days')
    plt.ylabel('Frequency')

    # Show the plots
    plt.tight_layout()
```

```python
    plt.ylabel('Frequency')

    # Plot Shipping Delay for this mode (only for delays > 0)
    plt.subplot(1, 2, 2)
    plt.hist(df_mode['Shipping Delay'], bins=15, color='r', alpha=0.7)
    plt.title(f'Shipping Delay for {mode} (Delay > 0)')
    plt.xlabel('Days')
    plt.ylabel('Frequency')

    # Show the plots
    plt.tight_layout()
    plt.show()
```




```python
# Step 1: Calculate the number of purchases per customer
customer_orders = df.groupby('Customer ID')['Order ID'].count().reset_index()
customer_orders.columns = ['Customer ID', 'Order Count']

# Step 2: Filter customers who experienced shipping delays
delayed_customers = df[df['Shipping Delay'] > 0]

# Step 3: Merge delayed customers with the customer orders to find one-time customers with delays
delayed_customer_orders = delayed_customers.merge(customer_orders, on='Customer ID')

# Step 4: Separate one-time customers from repeat customers
one_time_customers = delayed_customer_orders[delayed_customer_orders['Order Count'] == 1]
repeat_customers = delayed_customer_orders[delayed_customer_orders['Order Count'] > 1]

# Step 5: Compare average shipping delays
avg_delay_one_time = one_time_customers['Shipping Delay'].mean()
avg_delay_repeat = repeat_customers['Shipping Delay'].mean()

print(f"Average shipping delay for one-time customers: {avg_delay_one_time}")
print(f"Average shipping delay for repeat customers: {avg_delay_repeat}")

# Step 6: Visualize the impact of shipping delays on customer loyalty
import seaborn as sns
import matplotlib.pyplot as plt

sns.barplot(x=['One-Time Customers', 'Repeat Customers'], y=[avg_delay_one_time, avg_delay_repeat])
plt.title('Impact of Shipping Delay on Customer Loyalty')
plt.ylabel('Average Shipping Delay (Days)')
plt.show()
```

```python
# Step 2: Filter customers who experienced shipping delays
delayed_customers = df[df['Shipping Delay'] > 0]

# Step 3: Merge delayed customers with the customer orders to find one-time customers with delays
delayed_customer_orders = delayed_customers.merge(customer_orders, on='Customer ID')

# Step 4: Separate one-time customers from repeat customers
one_time_customers = delayed_customer_orders[delayed_customer_orders['Order Count'] == 1]
repeat_customers = delayed_customer_orders[delayed_customer_orders['Order Count'] > 1]

# Step 5: Compare average shipping delays
avg_delay_one_time = one_time_customers['Shipping Delay'].mean()
avg_delay_repeat = repeat_customers['Shipping Delay'].mean()

print(f"Average shipping delay for one-time customers: {avg_delay_one_time}")
print(f"Average shipping delay for repeat customers: {avg_delay_repeat}")

# Step 6: Visualize the impact of shipping delays on customer loyalty
import seaborn as sns
import matplotlib.pyplot as plt

sns.barplot(x=['One-Time Customers', 'Repeat Customers'], y=[avg_delay_one_time, avg_delay_repeat])
plt.title('Impact of Shipping Delay on Customer Loyalty')
plt.ylabel('Average Shipping Delay (Days)')
plt.show()
```

```
Average shipping delay for one-time customers: nan
Average shipping delay for repeat customers: 1.7693947144075022
```


```python
import pandas as pd
import matplotlib.pyplot as plt

# Assuming you already have 'Shipping Delay' and 'Ship Date' in your DataFrame (data)

# Extract month from the 'Ship Date'
data['Month'] = pd.DatetimeIndex(data['Ship Date']).month # Changed 'df' to 'data'

# Group the data by month and calculate the average shipping delay for each month
shipping_delay_by_month = data.groupby('Month')['Shipping Delay'].sum() # Changed 'df' to 'data'

# Plot the shipping delays over months
plt.figure(figsize=(10,6))
plt.plot(shipping_delay_by_month.index, shipping_delay_by_month.values, marker='o', linestyle='-', color='b')

# Labels and title
plt.xlabel('Month')
plt.ylabel('Total Shipping Delay (Days)')
plt.title('Total Shipping Delay by Month')
plt.grid(True)
plt.xticks(ticks=range(1, 13), labels=['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec'])

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

Is there a correlation between total orders and shipping delays for different shipping modes?

```python
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Assuming 'df' is your DataFrame and contains the necessary columns:
# - 'Ship Mode': to indicate the shipping mode
# - 'Shipping Delay': where positive values indicate delayed orders

# Step 1: Filter the DataFrame to include only rows with shipping delays greater than 0
delayed_orders = df[df['Shipping Delay'] > 0]

# Step 2: Create a new DataFrame to calculate order volume by Ship Mode
order_frequency = delayed_orders.groupby(['Ship Mode', 'Shipping Delay']).size().reset_index(name='Order Frequency')

# Step 3: Visualize the order frequency against shipping delays for each shipping mode
plt.figure(figsize=(12, 6))
sns.scatterplot(data=order_frequency, x='Shipping Delay', y='Order Frequency', hue='Ship Mode', palette='Set2')
plt.title('Order Frequency vs. Shipping Delay by Ship Mode')
plt.xlabel('Shipping Delay (days)')
plt.ylabel('total orders')
plt.grid()
plt.show()

# Step 4: Calculate the correlation coefficient between Shipping Delay and Order Frequency
correlation = order_frequency[['Shipping Delay', 'Order Frequency']].corr().iloc[0, 1]
print(f"Correlation coefficient between Order Frequency and Shipping Delay: {correlation:.2f}")
```

Conclusion and Recommendations

Sales Insights:

- Sales Trend: Sales peaked in 2018 with steady growth starting from 2015, despite a slight dip in 2016.
- Top Products: The best-selling products were Canon, Fellowes, Cisco, HON, and GVC.
- Regional Performance: The West region led in sales, followed by the East, Central, and South.
- Top Cities: New York City generated the highest sales, while East Orange and Memphis had the lowest technology sales.
- Product Categories: Technology products generated the highest average sales, followed by furniture and office supplies.
- Customer Segments: The consumer segment led in sales, followed by corporate and home office.
- Shipping Mode: Standard class was the most-used shipping option across all segments.

Recommendations:

- Focus marketing efforts on the South region and low-performing cities like East Orange and Memphis.
- Expand the range of technology products to drive more sales.
- Offer bundle deals or promotions for top products like Canon and Fellowes.
- Provide personalized offers to retain high-performing consumer and corporate segments.
- Incentivize the use of standard shipping with free or discounted options for certain order values.

Optimizing Shipping Efficiency: Key Insights and Strategic Recommendations

1.Seasonal Shipping Delays: Shipping delays are highest in August and December, likely due to holiday rushes and vacations in the U.S. also sees significant delays.

2.Regional Impact: The Central and West regions experience the most shipping delays, especially for Standard Class and Same Day deliveries. The Home Office segment in Central and Corporate segment in the South are most affected.

3.Shipping Mode Analysis: SecondClass has the highest delay rate, followed by Standard Class.First Class and Same Day shipping have similar and lower delays.

4.Customer Segments & Product Categories: Home Office customers are the most affected by delays. Furniture and Technology have the longest shipping delays among productcategories.

5.Customer Loyalty: Continued customers experience similar delays as stopped customers but place more frequent orders.

Factors such as customer service, product quality, pricing, and competition may have influenced stopped customers more than shipping delays alone.

Recommendations:

1.Improve Holiday & Peak Season Logistics: Implement strategies to manage peak seasons (August and December) by offering faster shipping options, adjusting inventory, and hiring seasonal staff.

2.Focus on Problematic Regions: Prioritize resolving delays in Central and West regions by improving infrastructure or offering special shipping incentives.

3.Optimize Shipping Modes: Consider incentivizing First Class and Same Day shipping options to reduce the load on Standard Class, which experiences the most delays.

4.Targeted Customer Retention Strategies: For stopped customers, improve communication, loyalty programs, and provide better customer service to reduce churn. Engage continued customers through loyalty incentives despite shipping delays.

5.Gather Customer Feedback: Collect feedback from both stopped and continued customers to better understand their concerns about shipping delays, service quality, and purchasing behavior. Use this feedback to create more personalized solutions and improve loyalty.

6.Enhance Communication: Provide real-time updates on delays and offer compensation (e.g., discounts) for late deliveries to maintain customer trust.