Lab 4: Data Visualization and EDA

CPE232 Data Models

---

1. Load all Superstore datasets.

Note: The same datasets used in Lab 3

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
```

In [2]:
```
# Write your code here
superstore_order = pd.read_csv('superstore/superstore_order.csv')
superstore_people = pd.read_csv('superstore/superstore_people.csv')
superstore_return = pd.read_csv('superstore/superstore_return.csv')
```

2. Determine shape of each dataset (print out the results as well).

In [3]:
```
# Write your code here
print(superstore_order.shape)
print(superstore_people.shape)
print(superstore_return.shape)
```

```
(8880, 21)
(4, 2)
(296, 2)
```

3. Show information of the dataset.

In [4]:
```
# Write your code here (superstore_order)
superstore_order.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8880 entries, 0 to 8879
Data columns (total 21 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Row ID         8880 non-null   int64
 1   Order ID       8880 non-null   object
 2   Order Date     8880 non-null   object
 3   Ship Date      8880 non-null   object
 4   Ship Mode      8880 non-null   object
 5   Customer ID    8880 non-null   object
 6   Customer Name  8880 non-null   object
 7   Segment        8880 non-null   object
 8   Country        8880 non-null   object
 9   City           8880 non-null   object
 10  State          8880 non-null   object
 11  Postal Code    8880 non-null   int64
 12  Region         8880 non-null   object
 13  Product ID     8880 non-null   object
 14  Category       8880 non-null   object
 15  Sub-Category   8880 non-null   object
 16  Product Name   8880 non-null   object
 17  Sales          8880 non-null   float64
 18  Quantity       8880 non-null   int64
 19  Discount       8880 non-null   float64
 20  Profit         8880 non-null   float64
dtypes: float64(3), int64(3), object(15)
memory usage: 1.4+ MB
```

In [5]:
```
# Write your code here (superstore_people)
superstore_people.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4 entries, 0 to 3
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Person  4 non-null      object
 1   Region  4 non-null      object
dtypes: object(2)
memory usage: 196.0+ bytes
```

In [6]:
```
# Write your code here (superstore_return)
superstore_return.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 296 entries, 0 to 295
Data columns (total 2 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Returned  296 non-null    object
 1   Order ID  296 non-null    object
dtypes: object(2)
memory usage: 4.8+ KB
```

4. Are there any missing values? If so, in which column?

Ans: ไม่มีค่า Missing Values (Null) ในทุกคอลัมน์ของทั้ง 3 DataFrames

In [7]:
```
superstore_order.isnull().sum()
```

Out[7]:
```
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     0
Region          0
Product ID      0
Category        0
Sub-Category    0
Product Name    0
Sales           0
Quantity        0
Discount        0
Profit          0
dtype: int64
```

In [9]:
```
superstore_people.isnull().sum()
```

Out[9]:
```
Person    0
Region    0
dtype: int64
```

In [10]:
```
superstore_return.isnull().sum()
```

Out[10]:
```
Returned    0
Order ID    0
dtype: int64
```

5.

- 5.1 List unique segments
- 5.2 List unique segments and their corresponding count
- 5.3 Create a pie chart to demonstrate unique segments and their count
- 5.4 Briefly describe what could be interpreted from this pie chart

Note: please create additional cells to answer 5.2 - 5.3

In [11]:
```
# Write your code here (5.1 List unique segments)
print("Unique Segments:", superstore_order["Segment"].unique())
```

```
Unique Segments: ['Consumer' 'Corporate' 'Home Office']
```

In [12]:
```
# Write your code here (5.2 List unique segments and their corresponding count)
segment_counts = pd.crosstab(superstore_order['Segment'], columns='count').reset
segment_counts
```

Out[12]:

col_0	Segment	count
0	Consumer	4613
1	Corporate	2673
2	Home Office	1594

In [13]:
```
# Write your code here (5.3 Create a pie chart)
fig = px.pie(segment_counts, values='count', names='Segment', title='Segment Dis
fig.show()
```

Answer for the question 5.4

Ans: The pie chart shows the distribution of unique segments in the superstore_order dataset. It is divided into three segments including Consumer, Corporate, and Home Office. The Consumer segment has the highest percentage, followed by Corporate and Home Office.

6.

- 6.1 List unique states
- 6.2 List top-10 unique states and their corresponding count
- 6.3 Create a bar chart (vertical) to demonstrate the count of top-10 unique states
- 6.4 Based on 6.2, also include the total sales of these states (show your result as a dataframe)
- 6.5 Using the result from 6.4, if you were the owner of this superstore, what information could be interpreted from this result?

Note: please create additional cells to answer 6.2 - 6.4

In [14]:
```
# Write your code here (6.1 List unique states)
print("Unique States:", superstore_order["State"].unique())
```

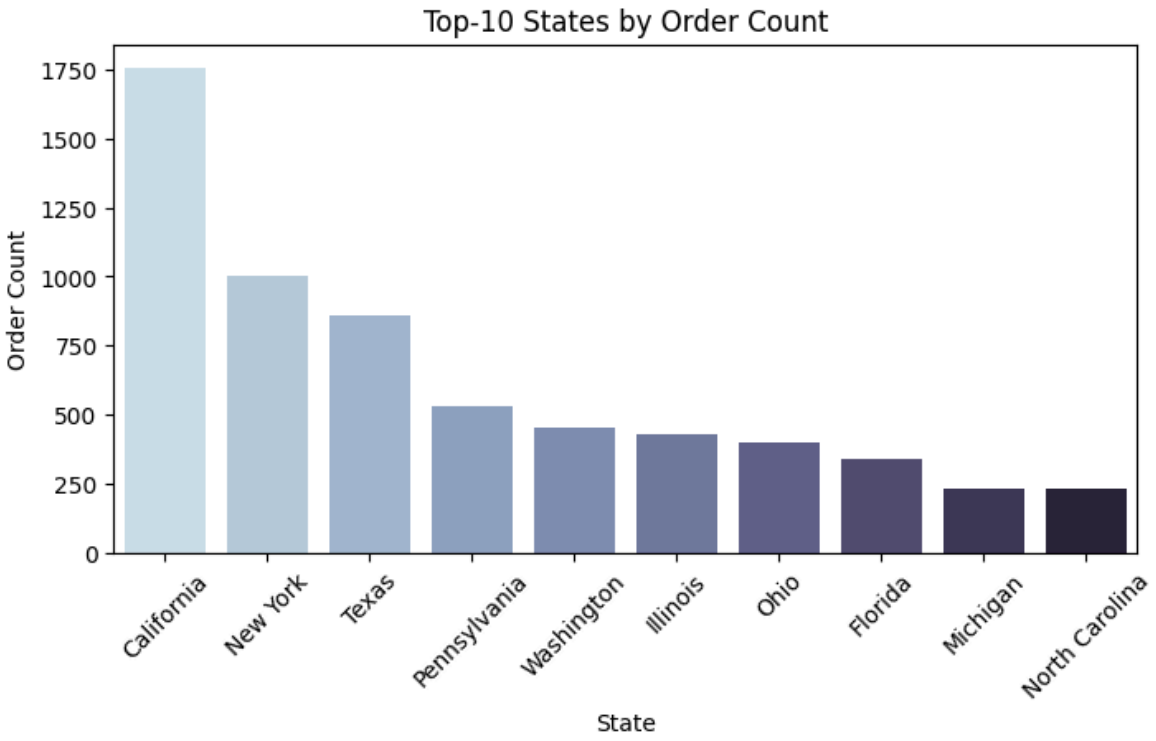
```
Unique States: ['Kentucky' 'California' 'Florida' 'North Carolina' 'Washington'
'Texas'
 'Wisconsin' 'Utah' 'Nebraska' 'Pennsylvania' 'Illinois' 'Minnesota'
 'Michigan' 'Delaware' 'Indiana' 'New York' 'Arizona' 'Virginia'
 'Tennessee' 'Alabama' 'South Carolina' 'Oregon' 'Colorado' 'Iowa' 'Ohio'
 'Missouri' 'Oklahoma' 'New Mexico' 'Louisiana' 'Connecticut' 'New Jersey'
 'Massachusetts' 'Georgia' 'Nevada' 'Rhode Island' 'Mississippi'
 'Arkansas' 'Montana' 'New Hampshire' 'Maryland' 'District of Columbia'
 'Kansas' 'Vermont' 'Maine' 'South Dakota' 'Idaho' 'North Dakota'
 'Wyoming' 'West Virginia']
```

In [15]:
```
# Write your code here (6.2 List top-10 unique states and their corresponding co
top10_state = pd.crosstab(superstore_order['State'], columns='count')\
                .sort_values('count', ascending=False).head(10).reset_index()
top10_state
```

Out[15]:

col_0	State	count
0	California	1754
1	New York	1001
2	Texas	860
3	Pennsylvania	531
4	Washington	452
5	Illinois	427
6	Ohio	396
7	Florida	339
8	Michigan	230
9	North Carolina	229

In [15]:
```
# Write your code here (6.3 Create a bar chart vertical to demonstrate the count
plt.figure(figsize=(8, 4))
sns.barplot(top10_state, x='State', y='count', hue='State',
            palette=sns.color_palette("ch:s=.25,rot=-.25", n_colors=len(top10_st
plt.title("Top-10 States by Order Count")
plt.xlabel("State")
plt.ylabel("Order Count")
plt.xticks(rotation=45)
plt.show()
```

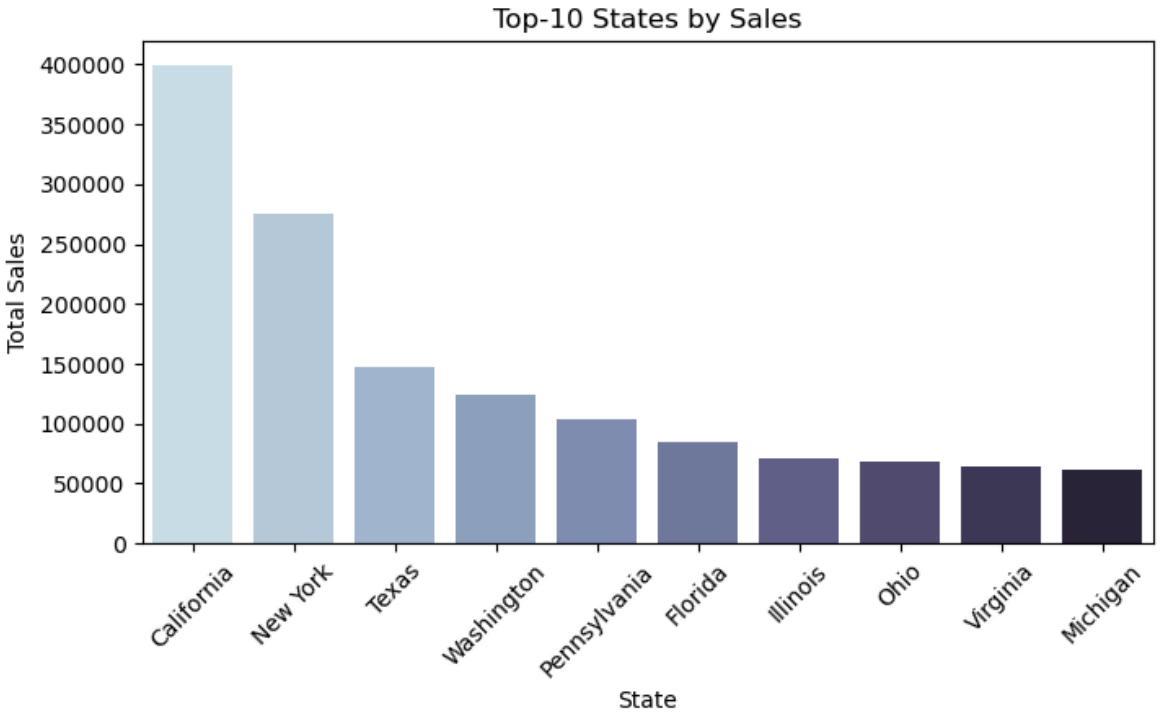


In [16]:
```
# Write your code here (6.4 Based on 6.2, also include the total sales of these
top10_state_sales = superstore_order.groupby("State").agg({'Sales': 'sum', 'Stat
                                    .sort_values('Sales', ascending=False).head(
                                    .rename(columns={'Sales': 'total_sales', 'St
top10_state_sales
```

Out[16]:

State	total_sales	count
California	399195.4555	1754
New York	274866.8190	1001
Texas	147855.0282	860
Washington	124497.7780	452
Pennsylvania	103852.5210	531
Florida	84083.0880	339
Illinois	71456.1780	427
Ohio	67924.2140	396
Virginia	64632.5200	191
Michigan	62147.6960	230

In [19]:
```
plt.figure(figsize=(8, 4))
sns.barplot(top10_state_sales, x=top10_state_sales.index, y='total_sales',
            hue='State', palette=sns.color_palette("ch:s=.25,rot=-.25", n_colors
            legend=False)
plt.title("Top-10 States by Sales")
plt.xlabel("State")
plt.ylabel("Total Sales")
plt.xticks(rotation=45)
plt.show()
```



Answer for the question 6.5

Ans: จากข้อมูล พบว่า California มีออเดอร์และยอดขายสูงสุด รองลงมาคือ New York และ Texas ควรขยายตลาดเพิ่มเติมในรัฐเหล่านี้ ส่วน Michigan และ North Carolina มียอดขายต่ำ อาจต้องใช้กลยุทธ์ส่ง

เสริมการขายเพิ่มเติม ขณะที่ Washington และ Florida มีออเดอร์สูงแต่ยอดขายไม่สอดคล้อง อาจต้องปรับกลยุทธ์การตลาด

7.

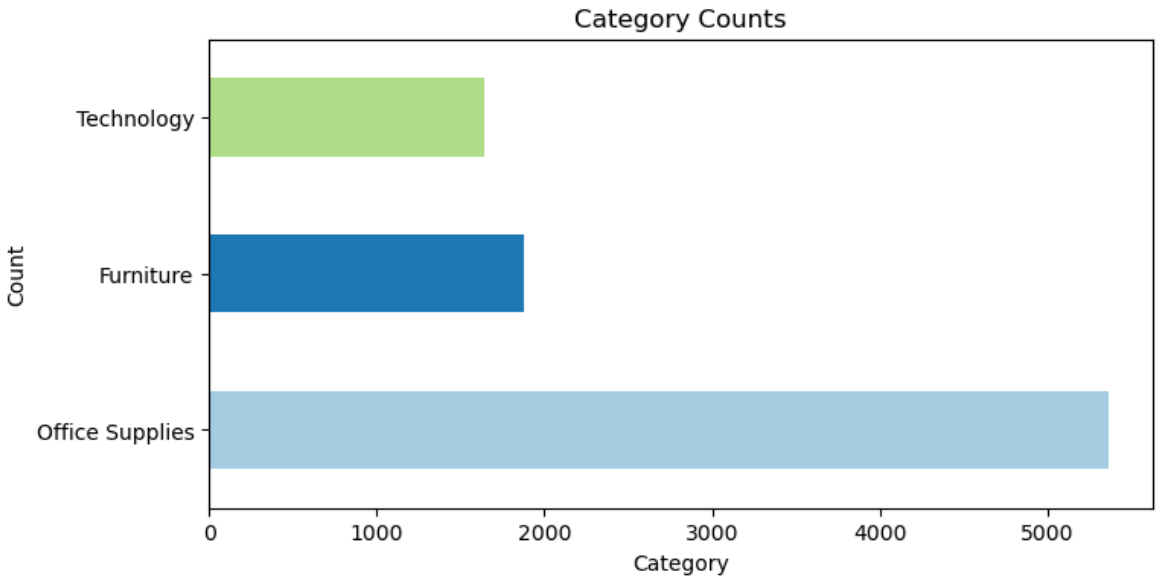
- 7.1 List unique categories
- 7.2 Create a bar chart (horizontal) to demonstrate the proportion of these categories
- 7.3 Compute the ratio of these categories in percentage and print the results

Note: please create additional cells to answer 7.2 - 7.3

In [17]:
```
# Write your code here (7.1 List unique categories)
print("Unique Categories:", superstore_order["Category"].unique())
```

```
Unique Categories: ['Furniture' 'Office Supplies' 'Technology']
```

In [20]:
```
# Write your code here (7.2 Create a bar chart horizontal to demonstrate the cou
category_count = superstore_order["Category"].value_counts()
plt.figure(figsize=(8, 4))
category_count.plot(kind='barh', color=plt.cm.Paired.colors)
plt.title("Category Counts")
plt.xlabel("Category")
plt.ylabel("Count")
plt.show()
```



In [21]:
```
# Write your code here (7.3 Compute the ratio of these categories in percentage
category_count = pd.DataFrame(category_count)
category_count['ratio'] = category_count['count'] / category_count['count'].sum(
category_count
```

Out[21]:

Category	count	ratio
Office Supplies	5360	60.360360
Furniture	1880	21.171171
Technology	1640	18.468468

8. Update the type of all columns that contain dates to *datetime* and show information after an update.

- Order Date, Ship Date

In [22]:
```
superstore_order.head(5)
```

Out[22]:

	Row ID	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name	Segment	Co
0	1	CA-2016-152156	08/11/2016	11/11/2016	Second Class	CG-12520	Claire Gute	Consumer	U
1	2	CA-2016-152156	08/11/2016	11/11/2016	Second Class	CG-12520	Claire Gute	Consumer	U
2	3	CA-2016-138688	12/06/2016	16/06/2016	Second Class	DV-13045	Darrin Van Huff	Corporate	U
3	4	US-2015-108966	11/10/2015	18/10/2015	Standard Class	SO-20335	Sean ODonnell	Consumer	U
4	5	US-2015-108966	11/10/2015	18/10/2015	Standard Class	SO-20335	Sean ODonnell	Consumer	U

5 rows × 21 columns

In [23]:
```
superstore_order[['Order Date', 'Ship Date']].info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8880 entries, 0 to 8879
Data columns (total 2 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   Order Date  8880 non-null   object
 1   Ship Date   8880 non-null   object
dtypes: object(2)
memory usage: 138.9+ KB
```

In [24]:
```
# write your code here
for col in ['Order Date', 'Ship Date']:
    superstore_order[col] = pd.to_datetime(superstore_order[col], format='%d/%m/

superstore_order[['Order Date', 'Ship Date']].info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8880 entries, 0 to 8879
Data columns (total 2 columns):
 #   Column      Non-Null Count  Dtype         
---  ------      --------------  -----         
 0   Order Date  8880 non-null   datetime64[ns]
 1   Ship Date   8880 non-null   datetime64[ns]
dtypes: datetime64[ns](2)
memory usage: 138.9 KB
```

9. Create a new column "Processing time day" to show number of days taken to ship an order and show your result in a dataframe format.

Hint: The duration starts as soon as the item has been ordered and ends once the order has successfully shipped.

In [25]:
```
# write your code here
superstore_order["Processing time day"] = (superstore_order["Ship Date"] - super
superstore_order[['Order Date', 'Ship Date', 'Processing time day']].head()
```

Out[25]:

	Order Date	Ship Date	Processing time day
0	2016-11-08	2016-11-11	3
1	2016-11-08	2016-11-11	3
2	2016-06-12	2016-06-16	4
3	2015-10-11	2015-10-18	7
4	2015-10-11	2015-10-18	7

10. Based on the result in 9.

- 10.1 How many orders are there that take more than 5 days to process?
- 10.2 Show the top 5 rows (expected output should contain these columns: Order ID, Order Date, Ship Date, Processing time day, Quantity)
- 10.3 Plot the histogram based on the column Quantity

Note: please create additional cells to answer 10.2 - 10.3

In [26]:
```
# Write your code here (10.1)
count = (superstore_order["Processing time day"] > 5).sum()
print(count)
```

```
1656
```

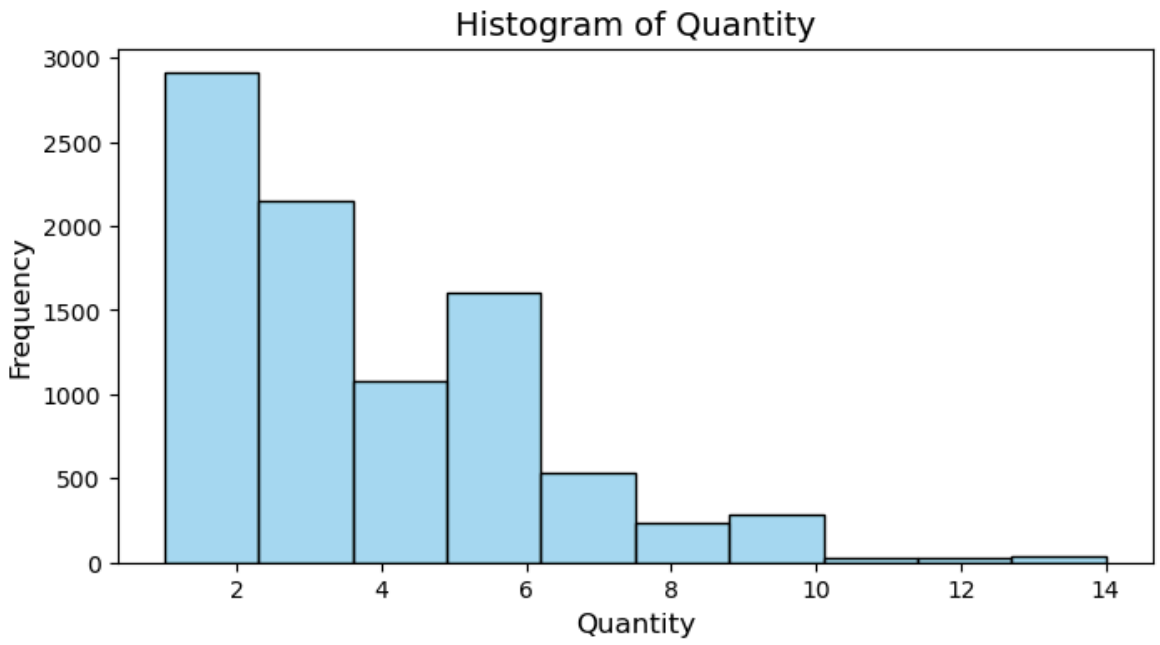
In [27]:
```
# Write your code here (10.2)
cols = ["Order ID", "Order Date", "Ship Date", "Processing time day", "Quantity"
superstore_order[cols].head(5)
```

Out[27]:

	Order ID	Order Date	Ship Date	Processing time day	Quantity
0	CA-2016-152156	2016-11-08	2016-11-11	3	2
1	CA-2016-152156	2016-11-08	2016-11-11	3	3
2	CA-2016-138688	2016-06-12	2016-06-16	4	2
3	US-2015-108966	2015-10-11	2015-10-18	7	5
4	US-2015-108966	2015-10-11	2015-10-18	7	2

In [28]:
```
# Write your code here (10.3)
plt.figure(figsize=(8, 4))
sns.histplot(superstore_order["Quantity"], bins=10, color="skyblue")

plt.xlabel("Quantity", fontsize=12)
plt.ylabel("Frequency", fontsize=12)
plt.title("Histogram of Quantity", fontsize=14)
plt.show()
```

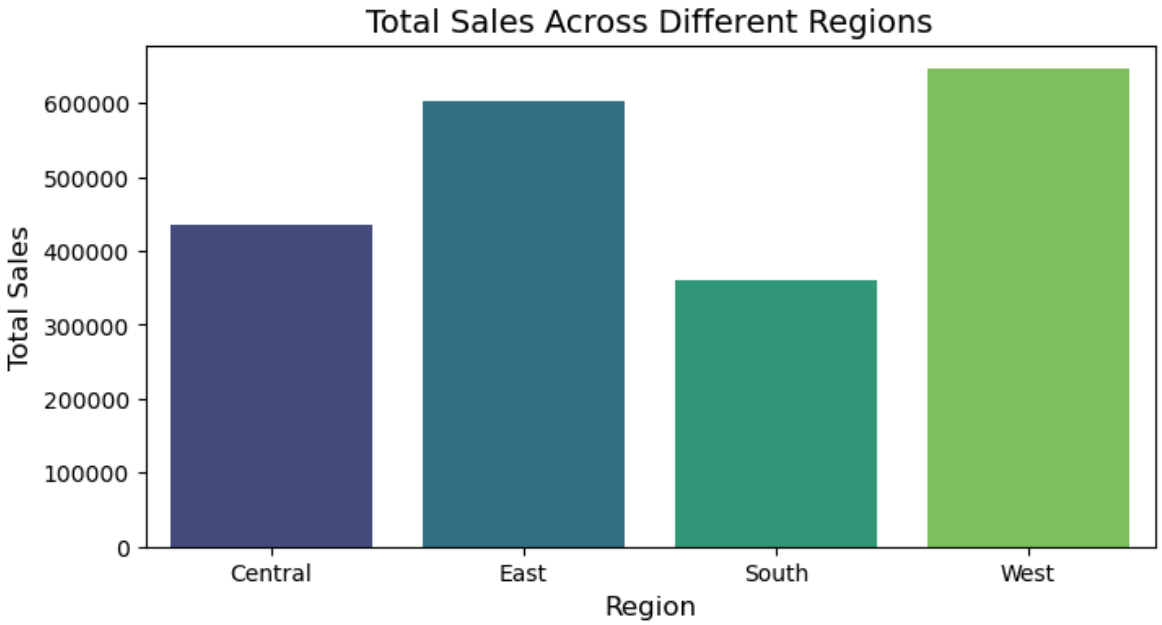


11. Total sales compare across different regions

- 11.1 Create a bar chart to visualize.

In [29]:
```
# Write your code here (11.1)
sales_by_region = superstore_order.groupby("Region")["Sales"].sum().reset_index(
```

```
plt.figure(figsize=(8, 4))
sns.barplot(data=sales_by_region, x="Region", y="Sales", hue='Region', palette='
plt.xlabel("Region", fontsize=12)
plt.ylabel("Total Sales", fontsize=12)
plt.title("Total Sales Across Different Regions", fontsize=14)
plt.show()
```



- 11.2 How do total sales compare across different regions? Explain in as much detail as possible.

  Ans: ยอดขายรวมเรียงลำดับภูมิภาคจากมากไปน้อยได้ดังนี้ West, East, Central, South

12.Which states have the highest number of returns? Use a horizontal bar chart.

Ans: California

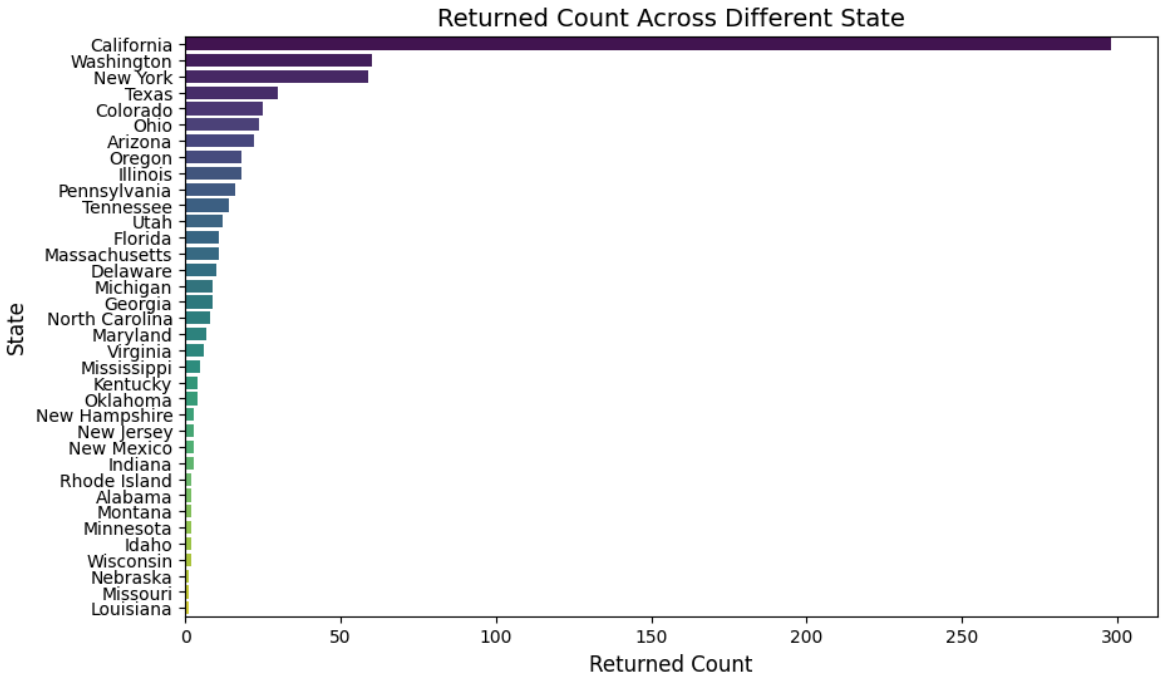
In [30]:
```
# Write your code here (12)
merged_df = pd.merge(superstore_return, superstore_order, on='Order ID', how='in
filtered_df = merged_df[ merged_df['Returned'] == 'Yes' ]
return_by_state = filtered_df.groupby('State').agg({'Returned': 'count'})\
                                .sort_values(by='Returned', ascending=False).res
return_by_state
```

Out[30]:

	State	Returned
0	California	298
1	Washington	60
2	New York	59
3	Texas	30
4	Colorado	25
5	Ohio	24
6	Arizona	22
7	Oregon	18
8	Illinois	18
9	Pennsylvania	16
10	Tennessee	14
11	Utah	12
12	Florida	11
13	Massachusetts	11
14	Delaware	10
15	Michigan	9
16	Georgia	9
17	North Carolina	8
18	Maryland	7
19	Virginia	6
20	Mississippi	5
21	Kentucky	4
22	Oklahoma	4
23	New Hampshire	3
24	New Jersey	3
25	New Mexico	3
26	Indiana	3
27	Rhode Island	2
28	Alabama	2
29	Montana	2
30	Minnesota	2
31	Idaho	2
32	Wisconsin	2

	State	Returned
33	Nebraska	1
34	Missouri	1
35	Louisiana	1

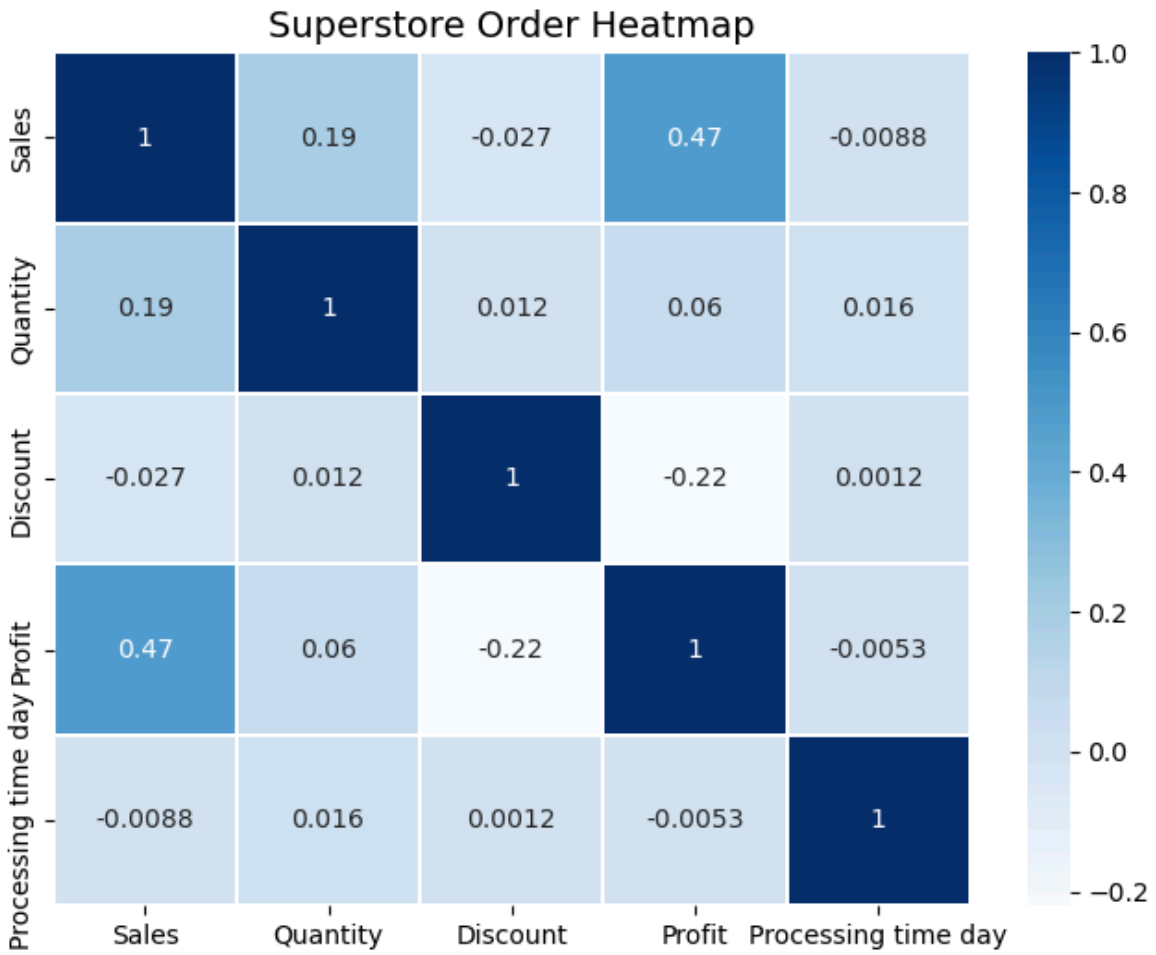
In [31]:
```
plt.figure(figsize=(10, 6))
sns.barplot(data=return_by_state, x='Returned', y='State', hue='State', palette=
plt.xlabel('Returned Count', fontsize=12)
plt.ylabel('State', fontsize=12)
plt.title('Returned Count Across Different State', fontsize=14)
plt.show()
```



13. What is the correlation between numerical variables in the superstore_order dataset? Use a heatmap

Hint: Use seaborn to create a heatmap :)

In [31]:
```
# Write your code here (13)
cols = ['Sales', 'Quantity', 'Discount', 'Profit','Processing time day']
# คำนวณ correlation matrix
correlation_matrix = superstore_order[cols].corr()
# สร้าง heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(correlation_matrix, cmap='Blues', linewidths=0.30, annot=True)
plt.title('Superstore Order Heatmap', fontsize=14)
plt.show()
```



14. Create a USA State-Level to visualize total sales per state.

- The darkest color represents the highest total sales.
- The lightest color represents the lowest total sales.
- Use a continuous gradient scale (e.g., dark blue to light blue, dark red to light red, or any custom gradient of your choice).

Hint: Use plotly.express

In [32]:
```
state_code = pd.read_csv('https://raw.githubusercontent.com/plotly/datasets/mast
us_state_map = state_code[['code', 'state']]

states_dict = {}
for i, row in us_state_map.iterrows():
    states_dict[row['state']] = row['code']
print(states_dict)
```

```
{'Alabama': 'AL', 'Alaska': 'AK', 'Arizona': 'AZ', 'Arkansas': 'AR', 'Californi
a': 'CA', 'Colorado': 'CO', 'Connecticut': 'CT', 'Delaware': 'DE', 'Florida': 'F
L', 'Georgia': 'GA', 'Hawaii': 'HI', 'Idaho': 'ID', 'Illinois': 'IL', 'Indiana':
'IN', 'Iowa': 'IA', 'Kansas': 'KS', 'Kentucky': 'KY', 'Louisiana': 'LA', 'Maine':
'ME', 'Maryland': 'MD', 'Massachusetts': 'MA', 'Michigan': 'MI', 'Minnesota': 'M
N', 'Mississippi': 'MS', 'Missouri': 'MO', 'Montana': 'MT', 'Nebraska': 'NE', 'Ne
vada': 'NV', 'New Hampshire': 'NH', 'New Jersey': 'NJ', 'New Mexico': 'NM', 'New
York': 'NY', 'North Carolina': 'NC', 'North Dakota': 'ND', 'Ohio': 'OH', 'Oklahom
a': 'OK', 'Oregon': 'OR', 'Pennsylvania': 'PA', 'Rhode Island': 'RI', 'South Caro
lina': 'SC', 'South Dakota': 'SD', 'Tennessee': 'TN', 'Texas': 'TX', 'Utah': 'U
T', 'Vermont': 'VT', 'Virginia': 'VA', 'Washington': 'WA', 'West Virginia': 'WV',
'Wisconsin': 'WI', 'Wyoming': 'WY'}
```

In [33]:
```
sales_per_state = superstore_order.groupby('State')['Sales'].sum().reset_index()
sales_per_state['code'] = sales_per_state['State'].map(states_dict)

# Create the choropleth map using Plotly
px.choropleth(sales_per_state,
                    locations='code',
                    locationmode='USA-states',
                    color='Sales',
                    color_continuous_scale="Reds",
                    labels={'Sales': 'Total Sales'},
                    title='Total Sales per State in the USA',
                    scope='usa')
```

14.2 Answer the following questions:

1. Which state has the highest total sales?
2. How do sales anomalies affect the gradient color shading on the map?
3. If you change the color scale, does it impact readability? Why or why not?

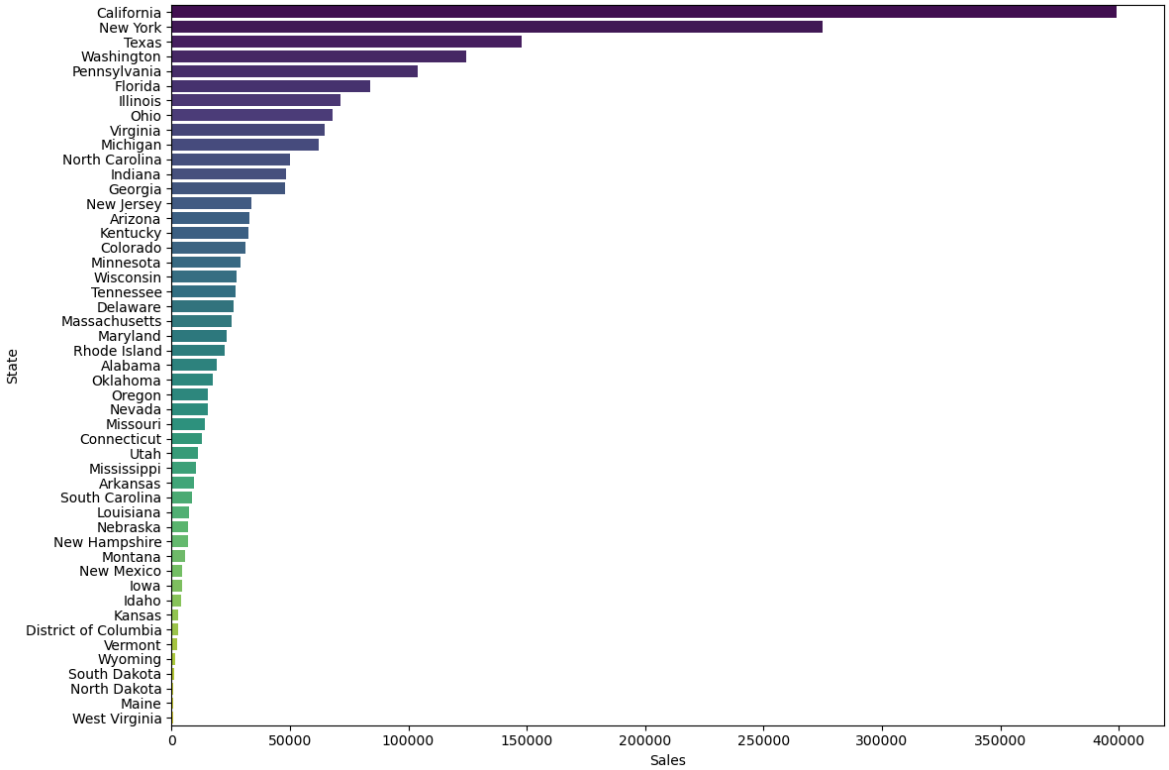
Ans:

1. California
2. จาก barplot ของ total sales per state ด้านล่าง สังเกตได้ว่า sales ส่วนใหญ่ไปกองกัยที่ California และ New York ทำให้สีของพื้นที่ใน state map ด้านบน มีพื้นที่ที่มีสีเข้มแค่ 2 จุดนั่นก็คือ California และ New York และทำให้สีที่แสดงถึง sales ของ state ที่เหลือไม่ชัดเจน (cutoff ประมาณ 100k) เนื่องจากมีค่าห่างจากทั้ง 2 top sales state มาก
3. หากเปลี่ยนช่วงสี (color scale) จะช่วยให้สามารถมองเห็นความแตกต่างของ total sales ในรัฐอื่น ๆ นอกเหนือจาก California และ New York ได้ดีขึ้น หากตั้งค่า upper bound ต่ำกว่าค่าสูงสุด (maximum sales) ของ California

เมื่อกำหนด upper bound ประมาณ 100K หรือต่ำกว่า รัฐที่มียอดขายระดับกลางและต่ำจะเริ่มแสดงความแตกต่างของเฉดสีได้ชัดเจนขึ้น แทนที่จะถูกกลืนไปกับเฉดสีอ่อนทั้งหมด ทำให้ อ่านค่าและเปรียบเทียบข้อมูลระหว่างรัฐได้ง่ายขึ้น

In [34]:
```
sorted_sales = sales_per_state.sort_values(by='Sales', ascending=False).reset_in

plt.figure(figsize=(12, 8))
sns.barplot(data=sorted_sales, y='State', x='Sales', hue='State', palette='virid
plt.tight_layout()
plt.show()
```

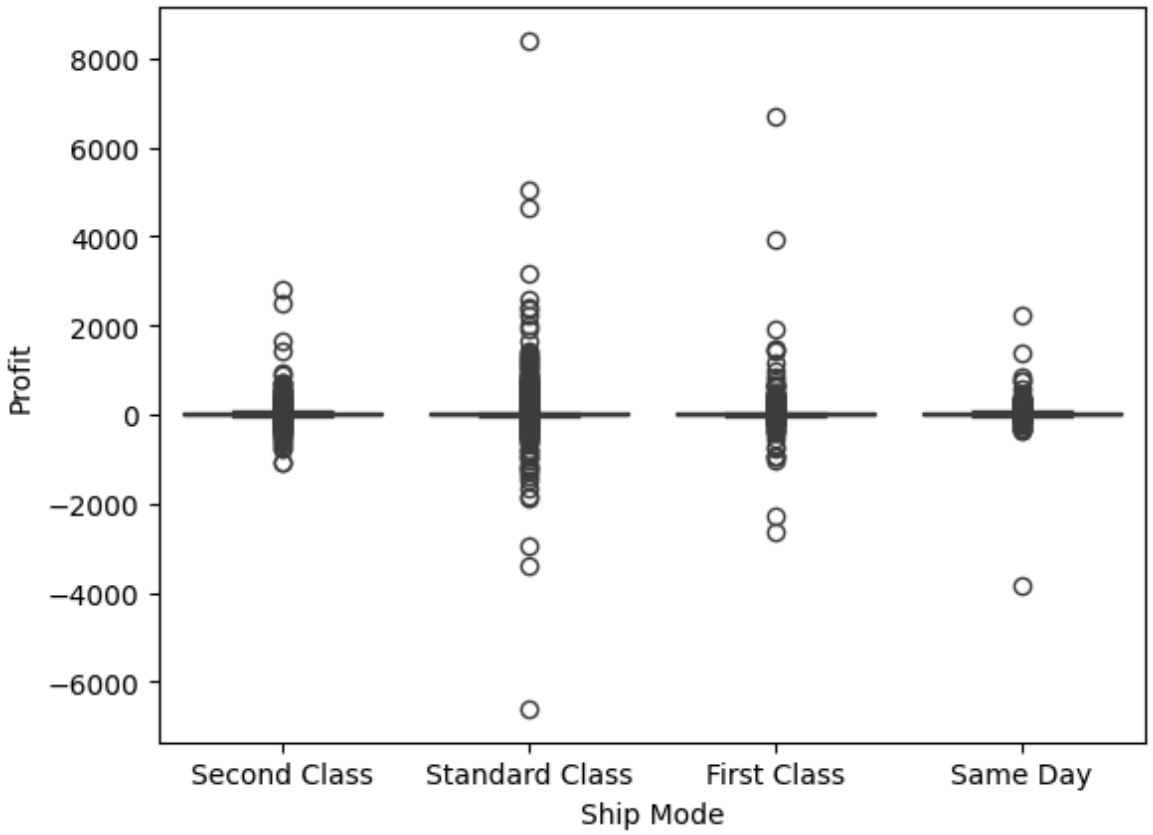


In [46]:
```
px.choropleth(sales_per_state,
                    locations='code',
                    locationmode='USA-states',
                    color='Sales',
                    color_continuous_scale="Reds",
                    range_color=(0, 80000),
                    labels={'Sales': 'Total Sales'},
                    title='Total Sales per State in the USA',
                    scope='usa')
```

15. Create a box plot to compare the different shipping modes based on total profit.

In [47]:
```
#Write your code here (15)
sns.boxplot(data=superstore_order, x='Ship Mode', y='Profit')
```

Out[47]: <Axes: xlabel='Ship Mode', ylabel='Profit'>



15.2 Which shipping mode has the highest median profit?

Ans: Second Class

In [35]:
```
superstore_order.groupby('Ship Mode').agg({'Profit': 'median'})\
    .sort_values('Profit', ascending=False)\
    .rename(columns={'Profit': 'Median Profit'})
```

Out[35]:

Ship Mode	Median Profit
Second Class	9.76080
First Class	8.79255
Same Day	8.43490
Standard Class	8.31040

[BONUS 20 pts] Determine the percentage of customers who:

- B1)returned the product once
- B2) returned the product at least once
- B3) never returned the product
- Finally, Plot a comparison of B2 and B3

Note: please create additional cells to answer the above points

B1

- Everyone in superstore_return.csv returned the product so, we don't have to filter the 'Returned' value.

In [36]:
```
superstore_return['Returned'].unique()
```

Out[36]:
```
array(['Yes'], dtype=object)
```

In [38]:
```
merged_df = pd.merge(superstore_order, superstore_return, how='left', on='Order
merged_df.head()
```

Out[38]:

	Row ID	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name	Segment	Country	
0	1	CA-2016-152156	2016-11-08	2016-11-11	Second Class	CG-12520	Claire Gute	Consumer	United States	Her
1	2	CA-2016-152156	2016-11-08	2016-11-11	Second Class	CG-12520	Claire Gute	Consumer	United States	Her
2	3	CA-2016-138688	2016-06-12	2016-06-16	Second Class	DV-13045	Darrin Van Huff	Corporate	United States	
3	4	US-2015-108966	2015-10-11	2015-10-18	Standard Class	SO-20335	Sean ODonnell	Consumer	United States	Lau
4	5	US-2015-108966	2015-10-11	2015-10-18	Standard Class	SO-20335	Sean ODonnell	Consumer	United States	Lau

5 rows × 23 columns

In [39]:
```
# count method automatically handle null and doesn't count it
returnByCustomer = merged_df.groupby('Customer ID').agg({'Returned': 'count'})
return_once = len(returnByCustomer[ returnByCustomer['Returned'] == 1 ])
ratio = round(return_once / len(returnByCustomer) * 100, 2)

print('Returned once customer count:', return_once)
print('Returned once ratio:', ratio, '%')
```

```
Returned once customer count: 62
Returned once ratio: 7.86 %
```

B2

In [40]:
```
return_count = len(returnByCustomer[ returnByCustomer['Returned'] >= 1 ])
ratio = round(return_count / len(returnByCustomer) * 100, 2)

print('Returned at least once customer count:', return_count)
print('Returned at least once ratio:', ratio, '%')
```

```
Returned at least once customer count: 222
Returned at least once ratio: 28.14 %
```

B3

In [41]:
```
no_return = len(returnByCustomer[ returnByCustomer['Returned'] == 0 ])
ratio = round(no_return / len(returnByCustomer) * 100, 2)

print('No returned customer count:', no_return)
print('No returned ratio:', round(no_return / len(returnByCustomer) * 100, 2), '
```

```
No returned customer count: 567
No returned ratio: 71.86 %
```