In [ ]:
```
#import require library
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [ ]:
```
#Load the dataset
data = pd.read_csv('C:\\Users\\Sakawat Siyam\\Downloads\\archive (3)\\SuperStore_Sa
```

In [ ]:
```
#show the first five row
data.head()
```

Out[ ]:

	Row ID+O6G3A1:R6	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name	Segment	Country	
0	4918	CA-2019-160304	01-01-2019	07-01-2019	Standard Class	BM-11575	Brendan Murry	Corporate	United States	Gaith
1	4919	CA-2019-160304	02-01-2019	07-01-2019	Standard Class	BM-11575	Brendan Murry	Corporate	United States	Gaith
2	4920	CA-2019-160304	02-01-2019	07-01-2019	Standard Class	BM-11575	Brendan Murry	Corporate	United States	Gaith
3	3074	CA-2019-125206	03-01-2019	05-01-2019	First Class	LR-16915	Lena Radford	Consumer	United States	Los
4	8604	US-2019-116365	03-01-2019	08-01-2019	Standard Class	CA-12310	Christine Abelman	Corporate	United States	San

In [ ]:
```
#getting column information
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5901 entries, 0 to 5900
Data columns (total 23 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   Row ID+O6G3A1:R6  5901 non-null   int64  
 1   Order ID          5901 non-null   object 
 2   Order Date        5901 non-null   object 
 3   Ship Date         5901 non-null   object 
 4   Ship Mode         5901 non-null   object 
 5   Customer ID       5901 non-null   object 
 6   Customer Name     5901 non-null   object 
 7   Segment           5901 non-null   object 
 8   Country           5901 non-null   object 
 9   City              5901 non-null   object 
 10  State             5901 non-null   object 
 11  Region            5901 non-null   object 
 12  Product ID        5901 non-null   object 
 13  Category          5901 non-null   object 
 14  Sub-Category      5901 non-null   object 
 15  Product Name      5901 non-null   object 
 16  Sales             5901 non-null   float64
 17  Quantity          5901 non-null   int64  
 18  Profit            5901 non-null   float64
 19  Returns           287 non-null    float64
 20  Payment Mode      5901 non-null   object 
 21  ind1              0 non-null      float64
 22  ind2              0 non-null      float64
dtypes: float64(5), int64(2), object(16)
memory usage: 1.0+ MB
```

In [ ]:
```
# To show all columns and rows
pd.set_option("display.max_columns", None)
pd.set_option("display.max_rows", None)
```

In [ ]:
```
#drop 3 unncessary columns
data.drop(['Row ID+O6G3A1:R6','ind1','ind2'], axis =1,inplace=True)
data.head()
```

Out[ ]:

	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name	Segment	Country	City	Stat
0	CA-2019-160304	01-01-2019	07-01-2019	Standard Class	BM-11575	Brendan Murry	Corporate	United States	Gaithersburg	Marylan
1	CA-2019-160304	02-01-2019	07-01-2019	Standard Class	BM-11575	Brendan Murry	Corporate	United States	Gaithersburg	Marylan
2	CA-2019-160304	02-01-2019	07-01-2019	Standard Class	BM-11575	Brendan Murry	Corporate	United States	Gaithersburg	Marylan
3	CA-2019-125206	03-01-2019	05-01-2019	First Class	LR-16915	Lena Radford	Consumer	United States	Los Angeles	Californi
4	US-2019-116365	03-01-2019	08-01-2019	Standard Class	CA-12310	Christine Abelman	Corporate	United States	San Antonio	Texa

In [ ]:
```
#get statistical infromation
data.describe()
```

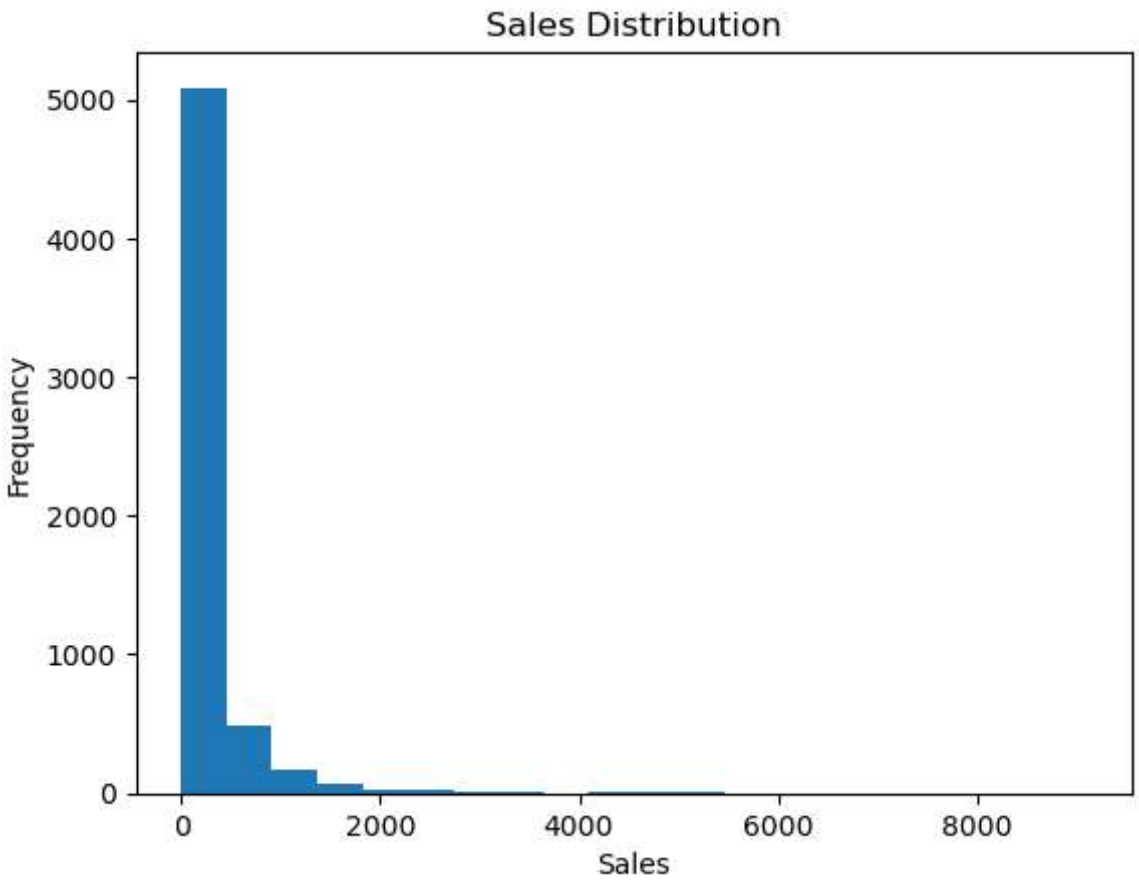
Out[ ]:

	Sales	Quantity	Profit	Returns
count	5901.000000	5901.000000	5901.000000	287.0
mean	265.345589	3.781901	29.700408	1.0
std	474.260645	2.212917	259.589138	0.0
min	0.836000	1.000000	-6599.978000	1.0
25%	71.976000	2.000000	1.795500	1.0
50%	128.648000	3.000000	8.502500	1.0
75%	265.170000	5.000000	28.615000	1.0
max	9099.930000	14.000000	8399.976000	1.0

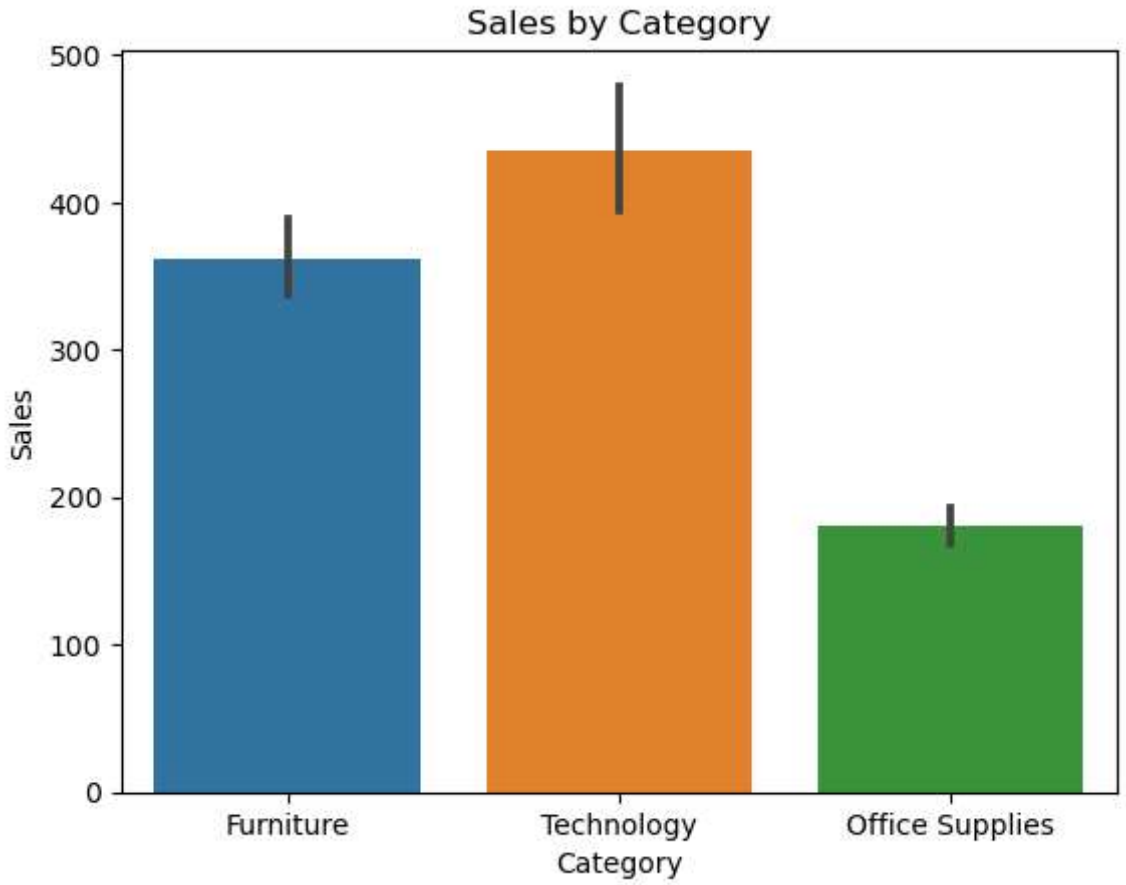
In [ ]:
```
#check the null value
data.isna().sum()
```

Out[ ]:
```
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
Region              0
Product ID          0
Category            0
Sub-Category        0
Product Name        0
Sales               0
Quantity            0
Profit              0
Returns          5614
Payment Mode        0
dtype: int64
```

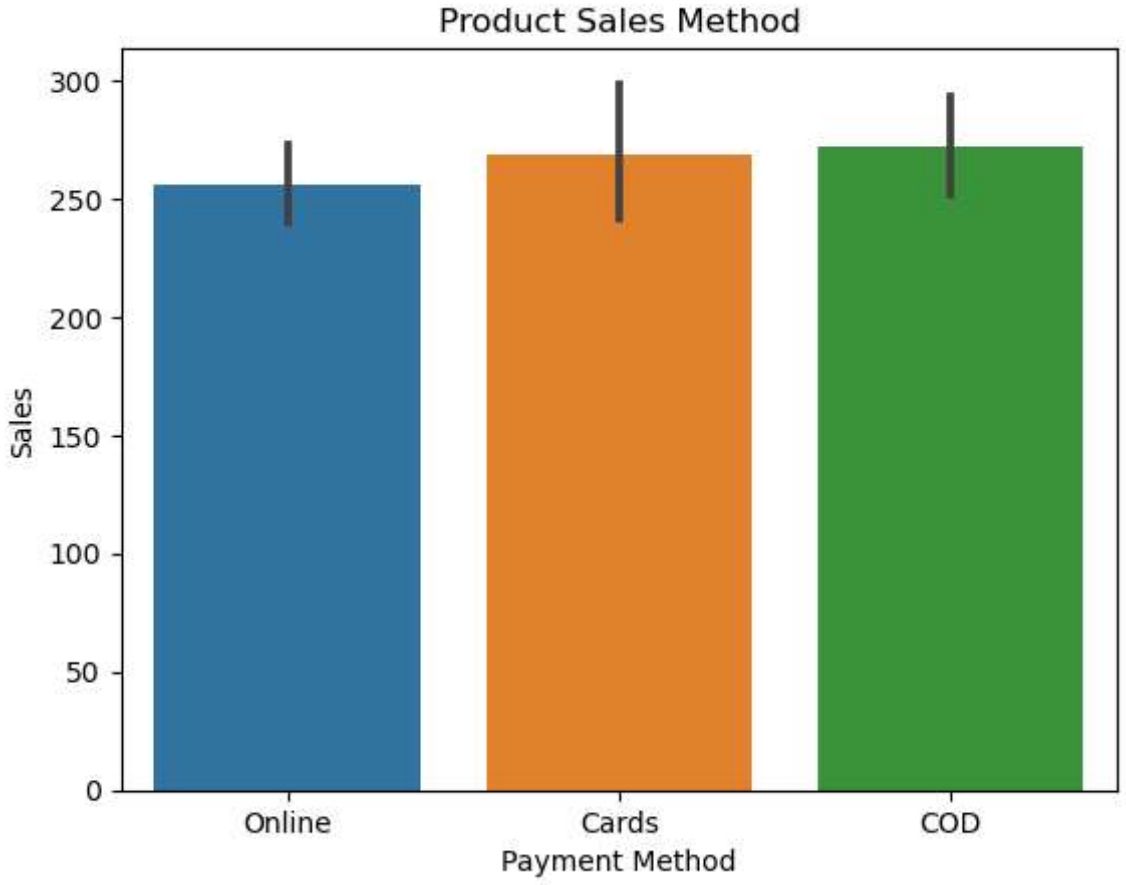
In [ ]:
```
#Histogram of sales
plt.hist(data['Sales'],bins=20)
plt.xlabel('Sales')
plt.ylabel('Frequency')
plt.title('Sales Distribution')
plt.show()
```



In [ ]:
```
# Bar chart of Sales by Category
sns.barplot(x='Category', y='Sales', data=data)
plt.xlabel('Category')
plt.ylabel('Sales')
plt.title('Sales by Category')
plt.show()
```



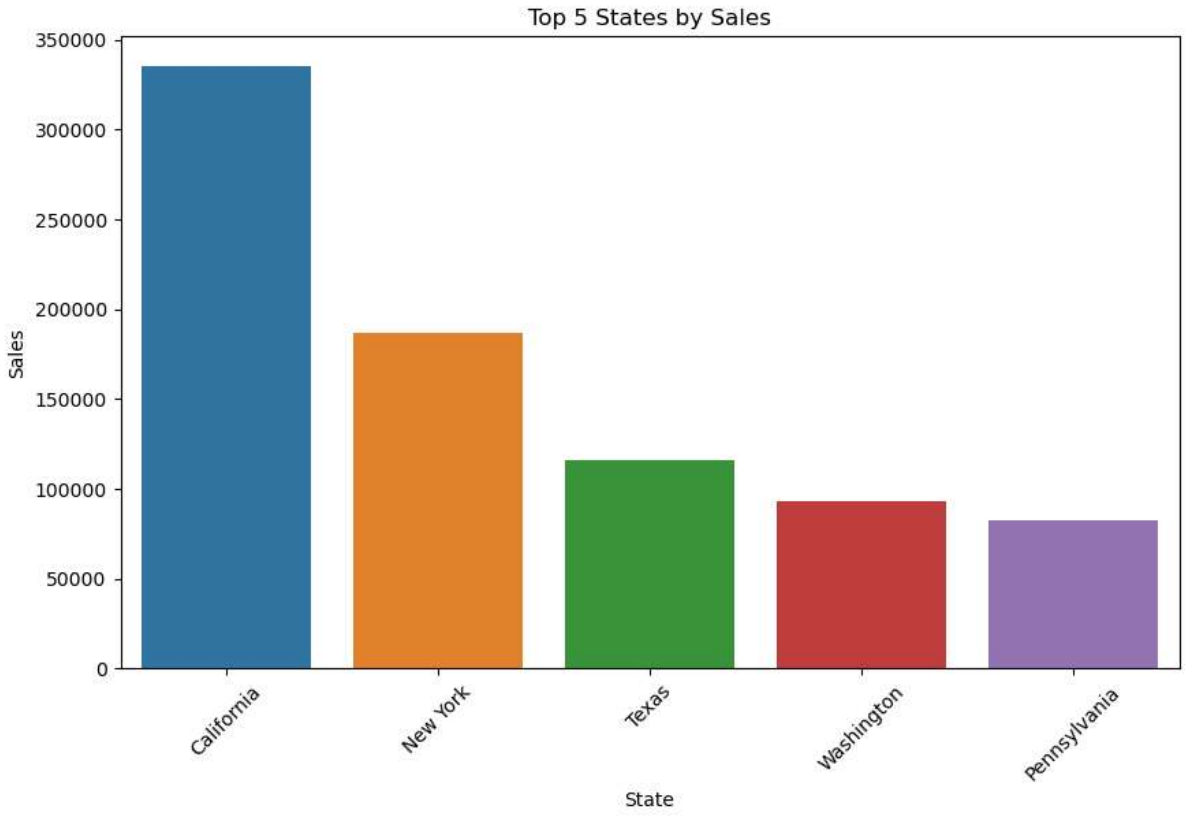
In [ ]:
```
# Bar chart of Sales by Category
sns.barplot(x='Payment Mode', y='Sales', data=data)
plt.xlabel('Payment Method')
plt.ylabel('Sales')
plt.title('Product Sales Method')
plt.show()
```



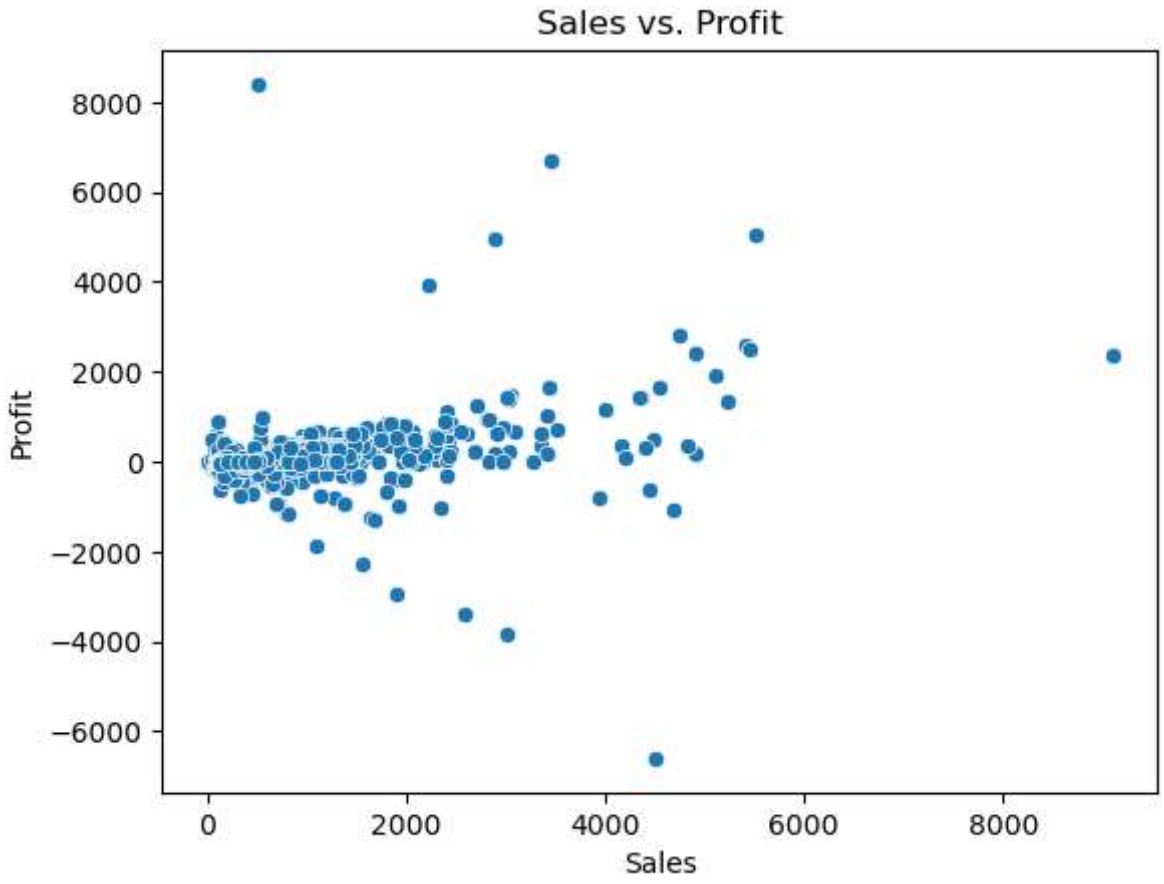
In [ ]:
```
# Bar chart of Sales by Category
sns.barplot(x='Ship Mode', y='Sales', data=data)
plt.xlabel('Ship Mode')
plt.ylabel('Sales')
plt.title('Sales by Shipping Mode')
plt.show()
```

In [ ]:
```
# Count and sort the states by total sales
top_states = data.groupby('State')['Sales'].sum().sort_values(ascending=False).head

# Create a bar chart for the top 5 states
plt.figure(figsize=(10, 6))
sns.barplot(x=top_states.index, y=top_states.values)
plt.xlabel('State')
plt.ylabel('Sales')
plt.title('Top 5 States by Sales')
plt.xticks(rotation=45)
plt.show()
```

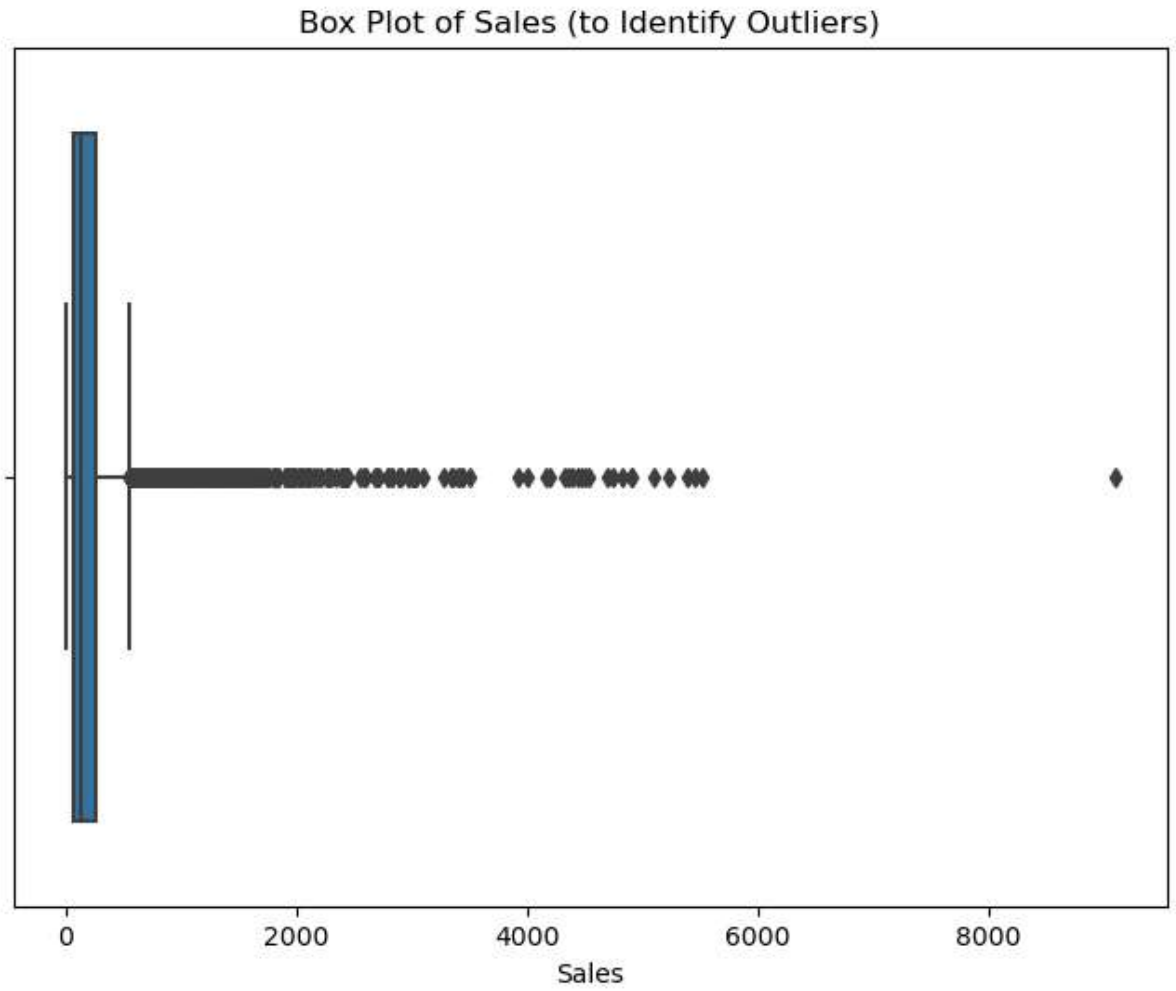


In [ ]:
```
# Scatter plot of Sales vs. Profit
sns.scatterplot(x='Sales', y='Profit', data=data)
plt.xlabel('Sales')
plt.ylabel('Profit')
plt.title('Sales vs. Profit')
plt.show()
```



In [ ]:
```
#create boxplot
plt.figure(figsize=(8, 6))
```

```
sns.boxplot(x=data['Sales'])
plt.xlabel('Sales')
plt.title('Box Plot of Sales (to Identify Outliers)')
plt.show()
```



In [ ]:
```
# Calculate the IQR (Interquartile Range)
Q1 = data['Sales'].quantile(0.25)
Q3 = data['Sales'].quantile(0.75)
IQR = Q3 - Q1

# Define the lower and upper bounds for outliers
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

# Identify and count outliers
outliers = data[(data['Sales'] < lower_bound) | (data['Sales'] > upper_bound)]
num_outliers = len(outliers)

print(f'Number of outliers: {num_outliers}')
```

```
Number of outliers: 631
```

In [ ]:
```
# Calculate total sales and profit
total_sales = data['Sales'].sum()
total_profit = data['Profit'].sum()

print(f'Total Sales: ${total_sales:.2f}')
print(f'Total Profit: ${total_profit:.2f}')
```

```
Total Sales: $1565804.32
Total Profit: $175262.11
```

In [ ]:
```
#Group Data by a Specific Attribute (e.g., Region)
grouped_by_region = data.groupby('Region')[['Sales', 'Profit']].sum()
print(grouped_by_region)
```

```
               Sales      Profit
Region
Central  341007.5242  27450.0071
East     450234.6660  53400.4243
South    252121.0810  26551.7163
West     522441.0520  67859.9582
```

In [ ]:
```
#Group Data by Another Attribute (e.g., Customer Segment)
grouped_by_segment = data.groupby('Segment')[['Sales', 'Profit']].sum()
print(grouped_by_segment)
```

```
                   Sales      Profit
Segment
Consumer      753002.1291  81338.5875
Corporate     509743.1262  57805.7991
Home Office   303059.0679  36117.7193
```

In [ ]:
```
#Which region has the highest total sales and profit
#Group the data by region and calculate the total sales and profit for each region
region_sales_profit = data.groupby('Region')[['Sales', 'Profit']].sum()

# Find the region with the highest total sales
region_highest_sales = region_sales_profit['Sales'].idxmax()

# Find the region with the highest total profit
region_highest_profit = region_sales_profit['Profit'].idxmax()

# Print the results
print(f"Region with the highest total sales: {region_highest_sales}")
print(f"Region with the highest total profit: {region_highest_profit}")
```

```
Region with the highest total sales: West
Region with the highest total profit: West
```

In [ ]:
```
#Is there a specific customer segment that generates more profit than others?

# Group the data by customer segment and calculate the total profit for each segmer
segment_profit = data.groupby('Segment')['Profit'].sum()

# Find the customer segment with the highest total profit
segment_highest_profit = segment_profit.idxmax()

# Print the results
print(f"Customer segment with the highest total profit: {segment_highest_profit}")
```

```
Customer segment with the highest total profit: Consumer
```

In [ ]:
```
#Are there any regions or segments with particularly low sales or profit?

# Calculate the minimum sales and profit for each region
region_min_sales = data.groupby('Region')['Sales'].min()
region_min_profit = data.groupby('Region')['Profit'].min()

# Define a threshold for low sales or profit (you can adjust this threshold)
sales_threshold = 1000  # Adjust as needed
profit_threshold = 100  # Adjust as needed

# Identify regions with low sales or profit
regions_with_low_sales = region_min_sales[region_min_sales < sales_threshold]
regions_with_low_profit = region_min_profit[region_min_profit < profit_threshold]
```

```
# Print the results
print("Regions with low sales:")
print(regions_with_low_sales)

print("\nRegions with low profit:")
print(regions_with_low_profit)
```

```
Regions with low sales:
Region
Central    0.836
East       1.504
South      2.214
West       1.408
Name: Sales, dtype: float64

Regions with low profit:
Region
Central   -2929.4845
East      -6599.9780
South     -3839.9904
West      -3399.9800
Name: Profit, dtype: float64
```

In [ ]:
```
#For Identifying Customer Segments with Low Sales or Profit


# Calculate the minimum sales and profit for each customer segment
segment_min_sales = data.groupby('Segment')['Sales'].min()
segment_min_profit = data.groupby('Segment')['Profit'].min()

# Define a threshold for low sales or profit (you can adjust this threshold)
sales_threshold = 1000  # Adjust as needed
profit_threshold = 100  # Adjust as needed

# Identify customer segments with low sales or profit
segments_with_low_sales = segment_min_sales[segment_min_sales < sales_threshold]
segments_with_low_profit = segment_min_profit[segment_min_profit < profit_threshold

# Print the results
print("Customer segments with low sales:")
print(segments_with_low_sales)

print("\nCustomer segments with low profit:")
print(segments_with_low_profit)
```

```
Customer segments with low sales:
Segment
Consumer       1.192
Corporate      0.836
Home Office    1.408
Name: Sales, dtype: float64

Customer segments with low profit:
Segment
Consumer      -6599.9780
Corporate     -3839.9904
Home Office   -3399.9800
Name: Profit, dtype: float64
```

In [ ]:
```
#Calculate return rates or other key performance indicators (KPIs)
# Calculate the total sales and the total returns
total_sales = data['Sales'].sum()
total_returns = data['Returns'].sum()

# Calculate the return rate as a percentage
```

```
return_rate = (total_returns / total_sales) * 100

# Print the results
print(f"Total Sales: ${total_sales:.2f}")
print(f"Total Returns: ${total_returns:.2f}")
print(f"Return Rate: {return_rate:.2f}%")
```

```
Total Sales: $1565804.32
Total Returns: $287.00
Return Rate: 0.02%
```