Sales Analysis

Data Exploration

In [ ]:
```python
# Importing Libraries

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

# Importing Sales Data

sales = pd.read_excel('superstore_sales.xlsx')
```

In [ ]:
```python
# Displaying Data

sales.head()
```

Out[ ]:

	order_id	order_date	ship_date	ship_mode	customer_name	segment	state	countr
0	AG-2011-2040	2011-01-01	2011-01-06	Standard Class	Toby Braunhardt	Consumer	Constantine	Algeri
1	IN-2011-47883	2011-01-01	2011-01-08	Standard Class	Joseph Holt	Consumer	New South Wales	Australi
2	HU-2011-1220	2011-01-01	2011-01-05	Second Class	Annie Thurman	Consumer	Budapest	Hungar
3	IT-2011-3647632	2011-01-01	2011-01-05	Second Class	Eugene Moren	Home Office	Stockholm	Swede
4	IN-2011-47883	2011-01-01	2011-01-08	Standard Class	Joseph Holt	Consumer	New South Wales	Australi

5 rows × 21 columns

In [ ]:
```python
sales.tail()
```

Out[ ]:

	order_id	order_date	ship_date	ship_mode	customer_name	segment	state	cc
51285	CA-2014-115427	2014-12-31	2015-01-04	Standard Class	Erica Bern	Corporate	California	U
51286	MO-2014-2560	2014-12-31	2015-01-05	Standard Class	Liz Preis	Consumer	Souss-Massa-Draâ	Mo
51287	MX-2014-110527	2014-12-31	2015-01-02	Second Class	Charlotte Melton	Consumer	Managua	Nica
51288	MX-2014-114783	2014-12-31	2015-01-06	Standard Class	Tamara Dahlen	Consumer	Chihuahua	M
51289	CA-2014-156720	2014-12-31	2015-01-04	Standard Class	Jill Matthias	Consumer	Colorado	U

5 rows × 21 columns

In [ ]:

```
# Data Shape (Rows / Columns)
sales.shape
```

Out[ ]:

```
(51290, 21)
```

In [ ]:

```
# Displaying Column Names
sales.columns
```

Out[ ]:

```
Index(['order_id', 'order_date', 'ship_date', 'ship_mode', 'customer_name',
       'segment', 'state', 'country', 'market', 'region', 'product_id',
       'category', 'sub_category', 'product_name', 'sales', 'quantity',
       'discount', 'profit', 'shipping_cost', 'order_priority', 'year'],
      dtype='object')
```

In [ ]:

```
# Data Summary
sales.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 51290 entries, 0 to 51289
Data columns (total 21 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   order_id        51290 non-null  object
 1   order_date      51290 non-null  datetime64[ns]
 2   ship_date       51290 non-null  datetime64[ns]
 3   ship_mode       51290 non-null  object
 4   customer_name   51290 non-null  object
 5   segment         51290 non-null  object
 6   state           51290 non-null  object
 7   country         51290 non-null  object
 8   market          51290 non-null  object
 9   region          51290 non-null  object
 10  product_id      51290 non-null  object
 11  category        51290 non-null  object
 12  sub_category    51290 non-null  object
 13  product_name    51290 non-null  object
 14  sales           51290 non-null  float64
 15  quantity        51290 non-null  int64
 16  discount        51290 non-null  float64
 17  profit          51290 non-null  float64
 18  shipping_cost   51290 non-null  float64
 19  order_priority  51290 non-null  object
 20  year            51290 non-null  int64
dtypes: datetime64[ns](2), float64(4), int64(2), object(13)
memory usage: 8.2+ MB
```

In [ ]:
```python
# Checking for Missing Values
sales.isnull().sum()
```

Out[ ]:
```
order_id          0
order_date        0
ship_date         0
ship_mode         0
customer_name     0
segment           0
state             0
country           0
market            0
region            0
product_id        0
category          0
sub_category      0
product_name      0
sales             0
quantity          0
discount          0
profit            0
shipping_cost     0
order_priority    0
year              0
dtype: int64
```

In [ ]:
```python
# Descriptive Statistics of Data
sales.describe()
```

Out[ ]:

	sales	quantity	discount	profit	shipping_cost	year
count	51290.000000	51290.000000	51290.000000	51290.000000	51290.000000	51290.000000
mean	246.490581	3.476545	0.142908	28.641740	26.375818	2012.777208
std	487.565361	2.278766	0.212280	174.424113	57.296810	1.098931
min	0.444000	1.000000	0.000000	-6599.978000	0.002000	2011.000000
25%	30.758625	2.000000	0.000000	0.000000	2.610000	2012.000000
50%	85.053000	3.000000	0.000000	9.240000	7.790000	2013.000000
75%	251.053200	5.000000	0.200000	36.810000	24.450000	2014.000000
max	22638.480000	14.000000	0.850000	8399.976000	933.570000	2014.000000

After initially exploring the data by checking for null values, examining the head and tail, and understanding the column names, I concluded there are no missing values, or additonal steps needed for immediate cleaning or manipulation. This exploration also revealed several key indicators potentially useful for our analysis, such as Sales Amount, Profit, Region, and Quantity. 'Sales' has a wide range, implying varying factors that casuses potential variations in customer spending. Throughout the analysis, we will take a deeper dive into the superstore sales dataset to answer questions that can potentially boost store performance.

Exploratory Data Analysis

- What is the Overall Sales Trend of the Data?

In [ ]:
```
# Sales Start Date
sales['order_date'].min()
```

Out[ ]:
```
Timestamp('2011-01-01 00:00:00')
```

In [ ]:
```
# Sales End Date
sales['order_date'].max()
```

Out[ ]:
```
Timestamp('2014-12-31 00:00:00')
```

This dataset comprises three years of sales data (January 2011 - December 2014), enabling comprehensive trend analysis and seasonality exploration.

In [ ]:
```
# Categorizing the Month/Year from the Data
sales['month_year'] = sales['order_date'].dt.strftime('%Y - %m')
```

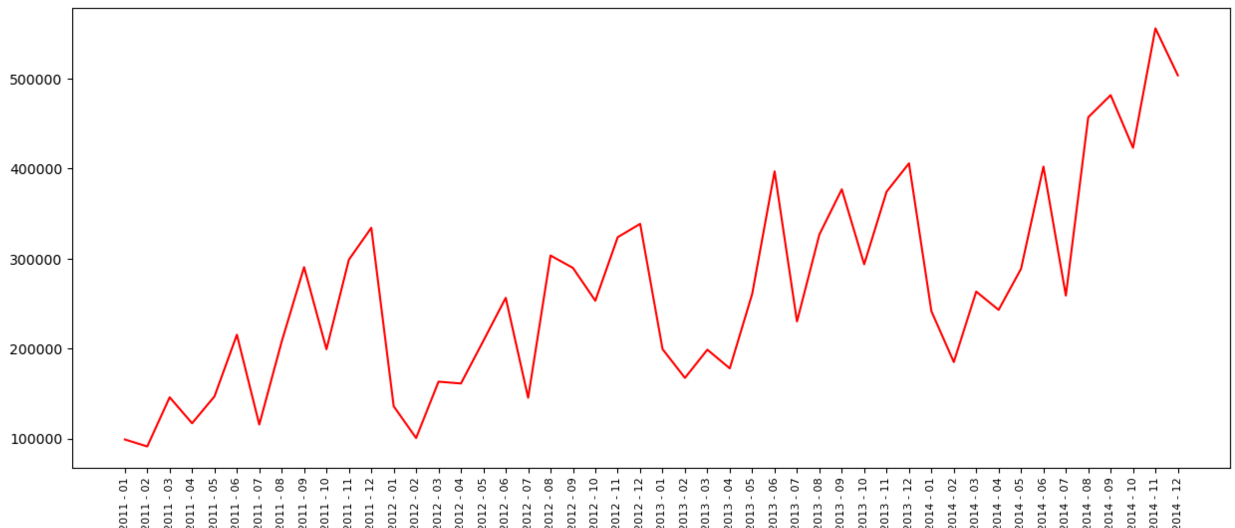
In [ ]:
```
# Month/Year Sales Trends
sales_trend = sales.groupby('month_year').sum(numeric_only=True)['sales'].rese
# Displaying DataFrame
sales_trend.head()
```

Out[ ]:

	month_year	sales
0	2011 - 01	98898.48886
1	2011 - 02	91152.15698
2	2011 - 03	145729.36736
3	2011 - 04	116915.76418
4	2011 - 05	146747.83610

In [ ]:

```
# Visualizing Sales Trends
# Adjusting Figure Size
plt.figure(figsize=(15,6))
# Visualizing Data (Line Graph)
plt.plot(sales_trend['month_year'], sales_trend['sales'], color = 'red')
# Rotating X-Axis Labels
plt.xticks(rotation = 'vertical', size = 8)
# Displaying a Clean Graph
plt.show()
```



Deeper analysis on the sales data reveals a recurring yearly pattern: predictable dips in February and July followed by consistent rebounds and positive growth. This poses an opportunity to utilize this analysis to strategically plan workforce needs, experiment with pricing, and diversify product offerings. Ultimately, leveraging this data empowers this store to navigate and anticipate sales fluctuations to achieve sustainable growth.

In [ ]:

```
# Sales - Day of Week
sales['day_of_week'] = sales['order_date'].dt.weekday
week_sales_trend = (sales.groupby('day_of_week').sum(numeric_only=True)['sales
week_sales_trend = week_sales_trend.sort_values('sales', ascending=False)

# Renaming Column Names - Easier Readability
week_sales_trend.columns = ['Day', 'Sales']
days = {0: 'Monday', 1: 'Tuesday', 2: 'Wednesday', 3: 'Thursday',
          4: 'Friday', 5: 'Saturday', 6: 'Sunday'}
week_sales_trend['Day'] = week_sales_trend['Day'].map(days)
week_sales_trend
```

Out[ ]:

	Day	Sales
4	Friday	2.322848e+06
1	Tuesday	2.268417e+06
3	Thursday	2.245837e+06
0	Monday	2.235913e+06
2	Wednesday	2.169218e+06
5	Saturday	1.177964e+06
6	Sunday	2.223045e+05

Weekly sales data reveals a distinct trend, which shows Fridays dominating revenue followed by Tuesdays and Thursdays. This suggests targeted weekend promotions and optimized staffing on Fridays and adjacent days could significantly boost revenue and improve customer experience.

- What is the Total Sales Amount Per Region?

In [ ]:

```python
# Group data by state and sum sales
sales_by_region = sales.groupby('region')['sales'].sum().reset_index()

# Sort in Ascending Order
sales_by_region = sales_by_region.sort_values('sales' , ascending= False)

# Percentage Per Region
total_sales = sales_by_region["sales"].sum()

# Calculate sales percentage by region
sales_by_region["sales_Percentage"] = sales_by_region["sales"] / total_sales *
sales_by_region
```

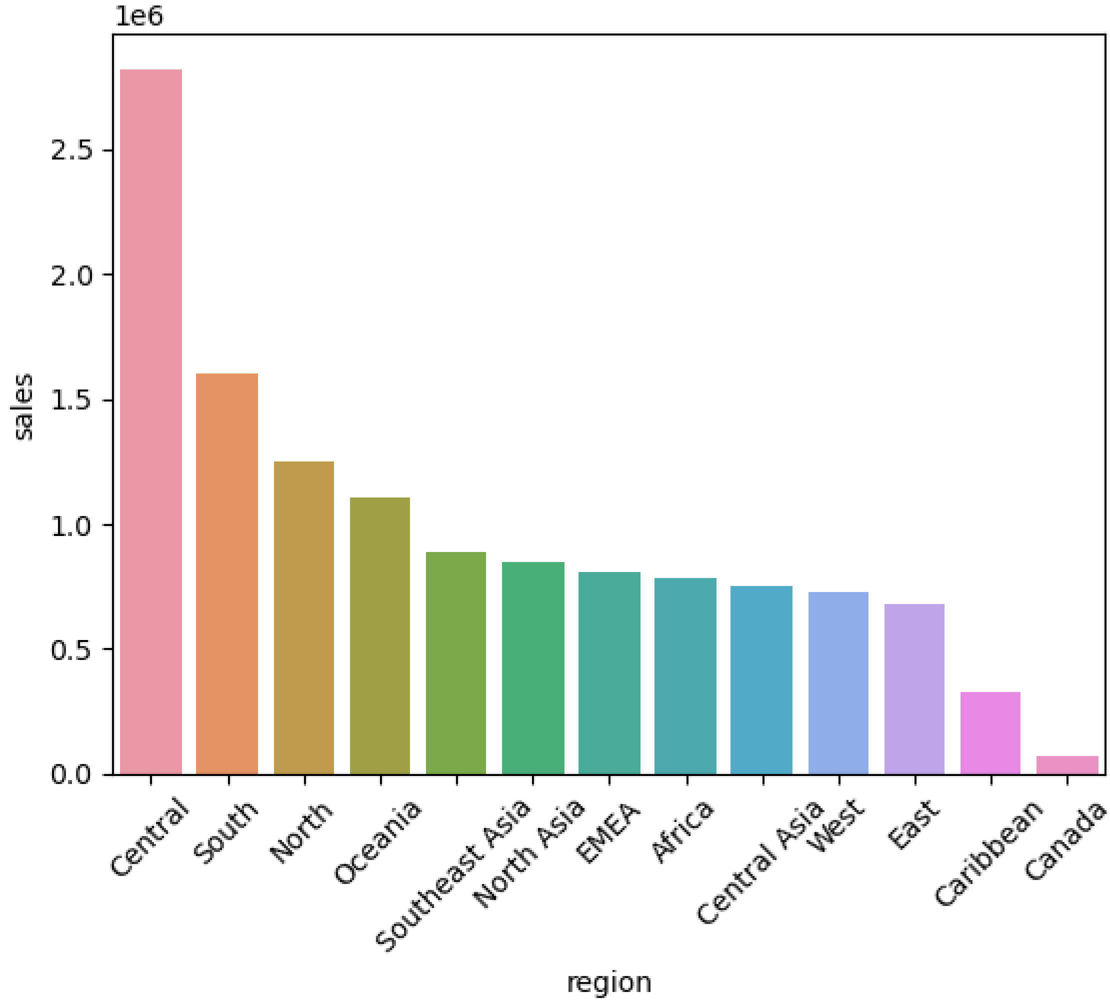
Out[ ]:

	region	sales	sales_Percentage
3	Central	2.822303e+06	22.323924
10	South	1.600907e+06	12.662897
7	North	1.248166e+06	9.872774
9	Oceania	1.100185e+06	8.702270
11	Southeast Asia	8.844232e+05	6.995634
8	North Asia	8.483098e+05	6.709983
5	EMEA	8.061613e+05	6.376596
0	Africa	7.837732e+05	6.199510
4	Central Asia	7.528266e+05	5.954728
12	West	7.254578e+05	5.738246
6	East	6.787812e+05	5.369042
2	Caribbean	3.242809e+05	2.565005
1	Canada	6.692817e+04	0.529390

In [ ]:

```
# Visualize Data
sns.barplot(x='region', y='sales', data=sales_by_region)

# Adjust X-Labels
plt.xticks(rotation=45)
plt.show()
```



The Central region dominates with 22.3% of total sales, while Canada and the Caribbean contribute 2.5% and 0.5%, respectively. Analyzing and addressing these significant disparities, particularly in regions with less than 5% of total sales like Cananda and Caribbean, could unlock substantial growth potential. By leveraging data-driven insights on customer preferences and product sales, along with targeted market research to understand local dynamics, we can develop strategies to improve sales in these underperforming areas. This presents unique opportunities for expansion and achieving a more balanced growth across all regions.

- Top 10 Products (Sales)

In [ ]:
```
# Grouping Product Name Column and Importing it into a DataFrame
prod_sales = pd.DataFrame(sales.groupby('product_name').sum(numeric_only=True))


# Sort DataFramein Ascending Order
prod_sales = prod_sales.sort_values('sales', ascending = False)


# Top 10 Products By Sales
prod_sales[:10]
```

Out[ ]:

product_name	sales
Apple Smart Phone, Full Size	86935.7786
Cisco Smart Phone, Full Size	76441.5306
Motorola Smart Phone, Full Size	73156.3030
Nokia Smart Phone, Full Size	71904.5555
Canon imageCLASS 2200 Advanced Copier	61599.8240
Hon Executive Leather Armchair, Adjustable	58193.4841
Office Star Executive Leather Armchair, Adjustable	50661.6840
Harbour Creations Executive Leather Armchair, Adjustable	50121.5160
Samsung Smart Phone, Cordless	48653.4600
Nokia Smart Phone, with Caller ID	47877.7857

- Top 10 Products (Quantity)

In [ ]:

```
# Grouping By Product Quantity
prod_quant_sold = pd.DataFrame(sales.groupby('product_name').sum(numeric_only=

# Sort DataFrame Ascending Order
prod_quant_sold = prod_quant_sold.sort_values('quantity', ascending= False)

# Top 10 Most Sold Products
prod_quant_sold[:10]
```

Out[ ]:

product_name	quantity
Staples	876
Cardinal Index Tab, Clear	337
Eldon File Cart, Single Width	321
Rogers File Cart, Single Width	262
Sanford Pencil Sharpener, Water Color	259
Stockwell Paper Clips, Assorted Sizes	253
Avery Index Tab, Clear	252
Ibico Index Tab, Clear	251
Smead File Cart, Single Width	250
Stanley Pencil Sharpener, Water Color	242

Apple Smartphones lead total sales in revenue generation, while Staples and Clear Index tabs dominate in quantity of items sold. Higher prices contribute significantly to revenue dominance, despite potentially lower sales volume. From the analysis, we can infer that

Apple's brand image attracts customers willing to pay more for quality. On the contrary, the staples are essential office supplies often bought in bulk, or regularly replenished, which causes an increased quantity of these items sold. Also, afforable price points make them accessible to a larger customer base, further contributing to their high sale volume.

- What are the most Profitable Categories and Subcategories ?

In [ ]:

```
# Grouping Subcategories with Categories based on Profit
cat_profit = pd.DataFrame(sales.groupby(['category', 'sub_category']).sum(nume

# Sort By Cateogry and Profit
cat_profit.sort_values(['category', 'profit'], ascending = False )
```

Out[ ]:

category	sub_category	profit
Technology	Copiers	258567.54818
	Phones	216717.00580
	Accessories	129626.30620
	Machines	58867.87300
Office Supplies	Appliances	141680.58940
	Storage	108461.48980
	Binders	72449.84600
	Paper	59207.68270
	Art	57953.91090
	Envelopes	29601.11630
	Supplies	22583.26310
	Labels	15010.51200
	Fasteners	11525.42410
Furniture	Bookcases	161924.41950
	Chairs	141973.79750
	Furnishings	46967.42550
	Tables	-64083.38870

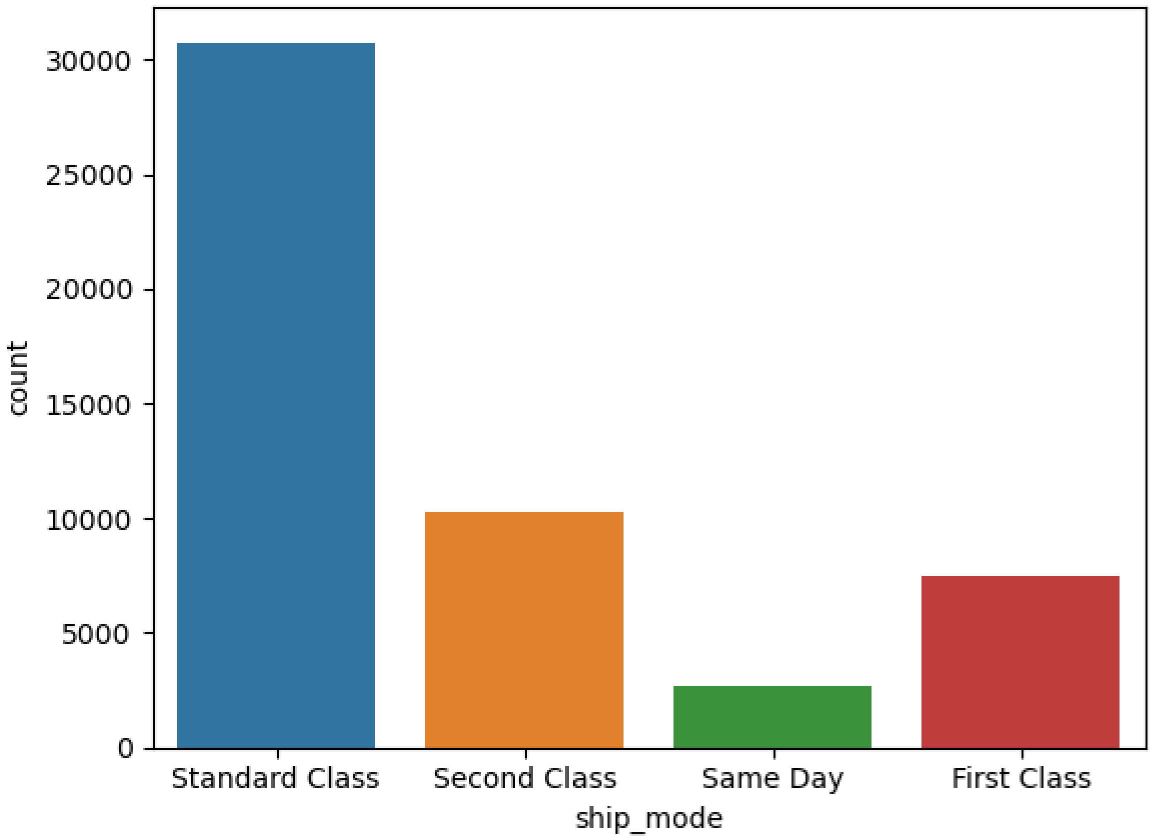
Our data reveals a contrast between high-performing technology products and furniture, particularly tables. While technology drives total profits, tables represent a significant loss, accounting for a $64,083.40 loss. Additional analysis behind the cause of the 64,083.40 loss of tabel profits will be needed to improve sales and boost overall store performance.

- What is the Preferred Shippign Method ?

In [ ]:
```
# Visualizing Popular Shipping Methods

sns.countplot(x = 'ship_mode' , data = sales)
```

Out[ ]:
```
<Axes: xlabel='ship_mode', ylabel='count'>
```



Data reveals a clear preference for Standard Class shipping , while Same Day shipping is the least preferred method. This could be attributed to pricing associated with these options, as Standard Class shipping methods are more cost effective compared to Same Day shipping methods. This presents a strategic opportunity to leverage incentives and explore diverse delivery options, which could potentially increase sales without major resource investment.