# Superstore Sales analysis

**An insightful, data-driven analysis of superstore sales aimed to enhance performance and enable smarter business decisions.**


**Team members:**

Fatma Esmail Ahmed Elshall

Ola Gamal Mohammed

Neveen Neanaa Waheib

Alaa Atef Mostafa

Noran Mohammed Abdelaziz

**Supervised By:**

Eng/ Ammar Mostafa


April 2025

## Project Idea

Analysis for a big store in united states that distributes various products to multiple states and cities.

In this project, we analyzed a Super Store Sales dataset to gain valuable insights into sales performance, customer behavior, and seasonal trends. The goal was to clean, explore, and visualize the data to support data-driven decision-making.

## Project Pipeline:

1. **Problem definition:**
2. **Data Preparing and understanding.**

3. **Data Cleaning & Processing:**

- Handled missing values.
- Removed duplicate records.
- Adjusted data types for date-related features.
- Dropped unnecessary features .
- Check validate dates.

**4. Exploratory Data Analysis (EDA):**

- Explored the dataset to understand patterns and trends.
- Answered key business questions.

**5. Summarized Insights in a Single Graph (Python):**

- Used Python to visualize key insights effectively.

**6.Created an Interactive Dashboard (Tableau):**

- Designed a comprehensive Tableau dashboard to present findings in an interactive and intuitive way to help businesses optimize their sales strategies.

**7- business recommendations**

- Insightful solutions for business to optimize performance in the future.

**Tools used:**
SQL: Data cleaning and processing
Python: Exploratory data analysis and visualization
Tableau: Interactive Business Dashboard

---

## Project Pipeline:

1. **Problem definition:**

The Superstore has collected several years of sales data but lacks a clear understanding of how its products, customers, and regions contribute to overall performance.
Without a data analysis approach, the store may continue to face inefficient marketing, unbalanced inventory management, and missed sales opportunities.

**The goal of this project** is to analyze the Superstore's historical sales data to uncover key insights that can support better business decisions, improve sales strategies, and enhance customer satisfaction.

---

2. **Data Preparing and understanding:**

The dataset includes detailed information about the store from 2015 to 2018.
It includes detailed information about customer orders, products, sales performance, shipping, and geographical data.
This dataset provides a solid foundation for analyzing trends, identifying key customers and products, and understanding business performance across different segments.

The dataset features were:
[Order ID', 'Order Date', 'Ship Date', 'Ship Mode', 'Customer ID',
'Customer Name', 'Segment', 'Country', 'City', 'State', 'Region',
'Product ID', 'Category', 'Sub-Category', 'Product Name', 'Sales']

| Row ID | Order ID | Order Date | Ship Date | Ship Mode | Customer ID | Customer Name | Segment | Country | City | State | Postal Code | Region |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | CA-2017-152156 | 08/11/2017 | 11/11/2017 | Second Class | CG-12520 | Claire Gute | Consumer | United States | Henderson | Kentucky | 42420.0 | South |
| 2 | CA-2017-152156 | 08/11/2017 | 11/11/2017 | Second Class | CG-12520 | Claire Gute | Consumer | United States | Henderson | Kentucky | 42420.0 | South |
| 3 | CA-2017-138688 | 12/06/2017 | 16/06/2017 | Second Class | DV-13045 | Darrin Van Huff | Corporate | United States | Los Angeles | California | 90036.0 | West |

## Data cleaning using SQL

- Import the dataset from the database schema
- Reading and explore datatypes of features .

```
/*Explore data*/
show columns
from final_project.`superstore sales dataset`;

/*select total number of rows*/
select count(*)
from final_project.`superstore sales dataset`;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

| Row ID | Order ID | Order Date | Ship Date | Ship Mode | Customer ID | Customer Name | Segment | Country | City |
|---|---|---|---|---|---|---|---|---|---|
| 1 | CA-2017-152156 | 08/11/2017 | 11/11/2017 | Second Class | CG-12520 | Claire Gute | Consumer | United States | Henderson |
| 2 | CA-2017-152156 | 08/11/2017 | 11/11/2017 | Second Class | CG-12520 | Claire Gute | Consumer | United States | Henderson |
| 3 | CA-2017-138688 | 12/06/2017 | 16/06/2017 | Second Class | DV-13045 | Darrin Van Huff | Corporate | United States | Los Angeles |
| 4 | US-2016-108966 | 11/10/2016 | 18/10/2016 | Standard Class | SO-20335 | Sean O'Donnell | Consumer | United States | Fort Lauderdale |

Result Grid
Form Editor

Information

**Columns:**

| | |
|---|---|
| Row ID | int |
| Order ID | text |
| Order Date | text |
| Ship Date | text |
| Ship Mode | text |
| Customer ID | text |
| Customer Name | text |
| Segment | text |
| Country | text |
| City | text |
| State | text |
| Postal Code | int |
| Region | text |
| Product ID | text |
| Category | text |
| Sub-Category | text |
| Product Name | text |
| Sales | double |

From the above figures we should

- delete columns as ‘Row ID’ and ‘Postal code’ -we don’t use in analysis stage .

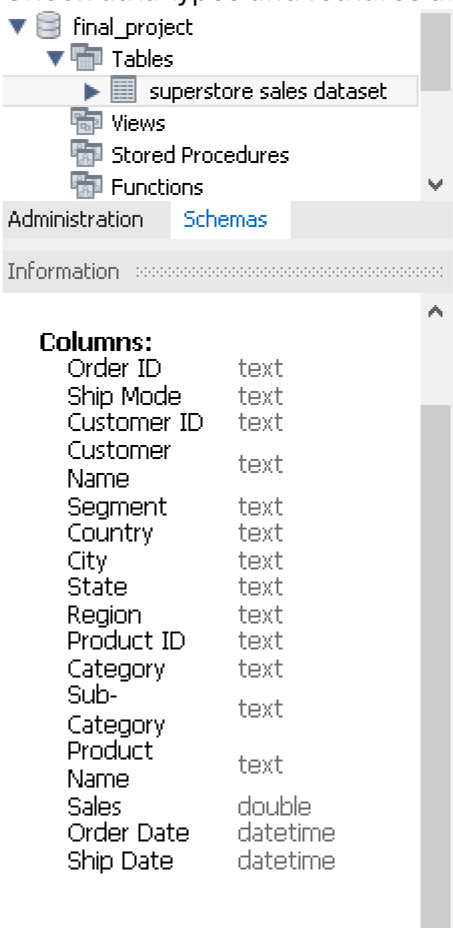
```sql
/* Delete postal code column as we won't use this column in analysis stage*/
/*also I will delete columns 'Row_id'  as we don't use them in analysis stage*/
alter table final_project.`superstore sales dataset`
drop column `Postal Code`,
drop column `Row ID`;
```

- And we will convert the data type of ‘Ship Data’ and ‘Order Data’ from text to Data Time .

```sql
/* change data type of 'ship date' and 'order date' */

/*first  Convert the String Date Format Before Changing Column Type */
/* MySQL expects dates in YYYY-MM-DD format  1-Create a New Temporary Column for Converted Dates */
ALTER TABLE final_project.`superstore sales dataset`
ADD COLUMN `Order_Date_Temp` DATETIME,
ADD COLUMN `Ship_Date_Temp` DATETIME;

SET SQL_SAFE_UPDATES = 0;  /*to turn off safe updates*/
/* 2- Convert Existing String Dates to YYYY-MM-DD Format*/
 UPDATE  final_project.`superstore sales dataset`
SET `Order_Date_Temp` = STR_TO_DATE(`Order Date`, '%d/%m/%Y'),
    `Ship_Date_Temp` = STR_TO_DATE(`Ship Date`, '%d/%m/%Y');
```

```sql
/* 3-Replace the Original Columns with Cleaned Date Columns*/
ALTER TABLE final_project.`superstore sales dataset`
DROP COLUMN `Order Date`,
DROP COLUMN `Ship Date`;

ALTER TABLE final_project.`superstore sales dataset`
CHANGE COLUMN `Order_Date_Temp` `Order Date` DATETIME,
CHANGE COLUMN `Ship_Date_Temp` `Ship Date` DATETIME;
```

- Check data types and features after delete:



- Check number of rows in data

```sql
/*select total number of rows*/
select count(*)
from final_project.`superstore sales dataset`;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

| | count(*) |
|---|---|
| ▶ | 9497 |

- Check nulls in all features:

```sql
/* check for nulls in all columns */
SELECT
    SUM(CASE WHEN `Order ID` IS NULL THEN 1 ELSE 0 END) AS Order_ID_nulls,
    SUM(CASE WHEN `Order Date` IS NULL THEN 1 ELSE 0 END) AS Order_Date_nulls,
    SUM(CASE WHEN `Ship Date` IS NULL THEN 1 ELSE 0 END) AS Ship_Date_nulls,
    SUM(CASE WHEN `Ship Mode` IS NULL THEN 1 ELSE 0 END) AS Ship_Mode_nulls,
    SUM(CASE WHEN `City` IS NULL THEN 1 ELSE 0 END) AS city_nulls,
    SUM(CASE WHEN `Country` IS NULL THEN 1 ELSE 0 END) AS country_nulls,
    SUM(CASE WHEN `Sales` IS NULL THEN 1 ELSE 0 END) AS sales_nulls,
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

| | Order_ID_nulls | Order_Date_nulls | Ship_Date_nulls | Ship_Mode_nulls | city_nulls | country_nulls | sales_nulls | region_nulls | region_nulls | region_nulls |
|---|---|---|---|---|---|---|---|---|---|---|
| ▶ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

All rows are cleaned.

- Check validate dates (if shipping date is before the order date)

```sql
/* check number of rows again after delete duplicates */
select count(*)
from final_project.`superstore sales dataset`;

/*Check validate Dates ( if Ship Date before Order Date)*/
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

| | Order ID | Ship Mode | Customer ID | Customer Name | Segment | Country | City | State | Region | Product ID | Category | Sub-Category | Product Name |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

  - All dates are correct.

- Check duplicates in data :

```
/* check duplicates in data*/
SELECT
    `Order ID`,`Ship Mode`, `Customer ID`, `Customer Name`, `Segment`, `Country`, `City`, `State`, `Region`,
    `Product ID`, `Category`, `Sub-Category`, `Product Name`,`Sales`, `Order Date`, `Ship Date`,
    COUNT(*) AS duplicate_count
FROM final_project.`superstore sales dataset`
GROUP BY
`Order ID`, `Ship Mode`, `Customer ID`, `Customer Name`, `Segment`, `Country`, `City`, `State`, `Region`,
`Product ID`, `Category`, `Sub-Category`, `Product Name`, `Sales`, `Order Date`, `Ship Date`
HAVING COUNT(*) > 1;
```

| Category | Sub-Category | Product Name | Sales | Order Date | Ship Date | duplicate_count |
|---|---|---|---|---|---|---|
| Furniture | Chairs | Global Leather Highback Executive Chair with P... | 281.372 | 2015-04-23 00:00:00 | 2015-04-27 00:00:00 | 2 |

- Delete the duplicated row and save cleaned data.

```
/* drop duplicates */
/* create new table for cleaned data*/
CREATE TABLE final_project.`superstore sales-cleand` AS
SELECT DISTINCT *
FROM final_project.`superstore sales dataset`;
```

- Check duplicated again.

```
SELECT
    `Order ID`,`Ship Mode`, `Customer ID`, `Customer Name`, `Segment`, `Country`, `City`, `State`, `Region`,
    `Product ID`, `Category`, `Sub-Category`, `Product Name`,`Sales`, `Order Date`, `Ship Date`,
    COUNT(*) AS duplicate_count
FROM final_project.`superstore sales-cleand`
GROUP BY
`Order ID`, `Ship Mode`, `Customer ID`, `Customer Name`, `Segment`, `Country`, `City`, `State`, `Region`,
`Product ID`, `Category`, `Sub-Category`, `Product Name`, `Sales`, `Order Date`, `Ship Date`
HAVING COUNT(*) > 1;

/* check number of rows again after delete duplicates */
```

| egment | Country | City | State | Region | Product ID | Category | Sub-Category | Product Name | Sales | Order Date | Ship Date | duplicate_count |
|---|---|---|---|---|---|---|---|---|---|---|---|---|

Now all the data is ready to analysis stage.

## Steps to understand data using python code:

### 1-import libraries:

```
#mathematical libraries
import numpy as np
import pandas as pd

#visualization libraries
import matplotlib.pyplot as plt
import seaborn as sns
# import modules for customize the colors
from matplotlib import cm
from matplotlib.ticker import FuncFormatter
```

### 2-Reading and exploring data

By pandas library

```
#reading data
df=pd.read_csv("/content/drive/MyDrive/DEPI_Graduation_project/Superstore Sales Dataset.csv")
df.head()
```

| | Row ID | Order ID | Order Date | Ship Date | Ship Mode | Customer ID | Customer Name | Segment | Country | City | State | Postal Code | Region | Product ID | Category | Sub Categor |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | CA-2017-152156 | 08/11/2017 | 11/11/2017 | Second Class | CG-12520 | Claire Gute | Consumer | United States | Henderson | Kentucky | 42420.0 | South | FUR-BO-10001798 | Furniture | Bookcase |
| 1 | 2 | CA-2017-152156 | 08/11/2017 | 11/11/2017 | Second Class | CG-12520 | Claire Gute | Consumer | United States | Henderson | Kentucky | 42420.0 | South | FUR-CH-10000454 | Furniture | Chai |
| 2 | 3 | CA-2017-138688 | 12/06/2017 | 16/06/2017 | Second Class | DV-13045 | Darrin Van Huff | Corporate | United States | Los Angeles | California | 90036.0 | West | OFF-LA-10000240 | Office Supplies | Labe |

Information about data and features as datatypes , number of columns and rows and check nulls.

```
df.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
 5   Customer ID    9800 non-null   object
 6   Customer Name  9800 non-null   object
 7   Segment        9800 non-null   object
 8   Country        9800 non-null   object
 9   City           9800 non-null   object
 10  State          9800 non-null   object
 11  Postal Code    9789 non-null   float64
 12  Region         9800 non-null   object
 13  Product ID     9800 non-null   object
 14  Category       9800 non-null   object
 15  Sub-Category   9800 non-null   object
 16  Product Name   9800 non-null   object
```

From the figure ,some features in data have nulls and ,other should delete and data types should be converted.

### 3-Data Cleaning & Processing

- Convert “Ship date” and “Order date” from object to Date time

```
# Convert to datetime (since it's in day-first format)
df['Ship Date'] = pd.to_datetime(df['Ship Date'] , dayfirst=True)
df['Order Date'] = pd.to_datetime(df['Order Date'], dayfirst=True)

# Change the format to MM-DD-YYYY
df['Ship Date'] = df['Ship Date'].dt.strftime('%m-%d-%Y')
df['Order Date'] = df['Order Date'].dt.strftime('%m-%d-%Y')

df.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
 5   Customer ID    9800 non-null   object
 6   Customer Name  9800 non-null   object
 7   Segment        9800 non-null   object
 8   Country        9800 non-null   object
 9   City           9800 non-null   object
 10  State          9800 non-null   object
 11  Postal Code    9789 non-null   float64
 12  Region         9800 non-null   object
 13  Product ID     9800 non-null   object
 14  Category       9800 non-null   object
 15  Sub-Category   9800 non-null   object
 16  Product Name   9800 non-null   object
 17  Sales          9800 non-null   float64
dtypes: float64(2), int64(1), object(15)
memory usage: 1.3+ MB
```

-Change the format date to “M-D-Y” to be converted to datetime

```
df['Ship Date'] = pd.to_datetime(df['Ship Date'])
df['Order Date'] = pd.to_datetime(df['Order Date'])
df.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   datetime64[ns]
 3   Ship Date      9800 non-null   datetime64[ns]
 4   Ship Mode      9800 non-null   object
 5   Customer ID    9800 non-null   object
 6   Customer Name  9800 non-null   object
 7   Segment        9800 non-null   object
 8   Country        9800 non-null   object
 9   City           9800 non-null   object
 10  State          9800 non-null   object
 11  Postal Code    9789 non-null   float64
 12  Region         9800 non-null   object
 13  Product ID     9800 non-null   object
 14  Category       9800 non-null   object
 15  Sub-Category   9800 non-null   object
 16  Product Name   9800 non-null   object
 17  Sales          9800 non-null   float64
dtypes: datetime64[ns](2), float64(2), int64(1), object(13)
```

- Delete columns “Row id” and “Postal code” as we won’t use in analysis stage

```
df.drop(columns=['Postal Code','Row ID'],inplace=True)
df.columns
```

```
Index(['Order ID', 'Order Date', 'Ship Date', 'Ship Mode', 'Customer ID',
       'Customer Name', 'Segment', 'Country', 'City', 'State', 'Region',
       'Product ID', 'Category', 'Sub-Category', 'Product Name', 'Sales'],
      dtype='object')
```

- Check duplicates in data

```
# chck duplicates
df[df.duplicated()]
```

| | Order ID | Order Date | Ship Date | Ship Mode | Customer ID | Customer Name | Segment | Country | City | State | Region | Product ID | Category | Sub-Category | Product Name | Sales |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 3406 | US-2015-150119 | 2015-04-23 | 2015-04-27 | Standard Class | LB-16795 | Laurel Beltran | Home Office | United States | Columbus | Ohio | East | FUR-CH-10002965 | Furniture | Chairs | Global Leather Highback Executive Chair with P... | 281.372 |

```
[ ] df.drop_duplicates(inplace =True)
df.shape
```

```
(9799, 16)
```

-Delete one row duplicated in data

- check validate dates (ship date before order date)

```
df[df['Ship Date']< df['Order Date']].value_counts()
```

| Order ID | Order Date | Ship Date | Ship Mode | Customer ID | Customer Name | Segment | Country | City | State | Region | Product ID | Category | Sub-Category | Product Name | Sales | count |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

dtype: int64

-All were correct.

- Remove extra spaces in data as “customer name”

```
#remove extra spaces in data
df = df.applymap(lambda x: x.strip() if isinstance(x, str) else x)
```

```
<ipython-input-12-9f1a00278b8b>:2: FutureWarning: DataFrame.applymap has been deprecated. Use DataFrame.map instead.
  df = df.applymap(lambda x: x.strip() if isinstance(x, str) else x)
```

Now all data was cleaned and ready to analysis stage

- Save the data after cleaning

```
[ ] #save the data after cleaning
df.to_csv("super store_clean.csv")
```

## 4-Exploratory data analysis

Explore insights about:

- Customer sales analysis
- Products analysis
- Region analysis
- Shipping insights
- Trend and seasonal analysis
- Explore correlation between features
- Create insightful python dashboard.

### -customer sales analysis

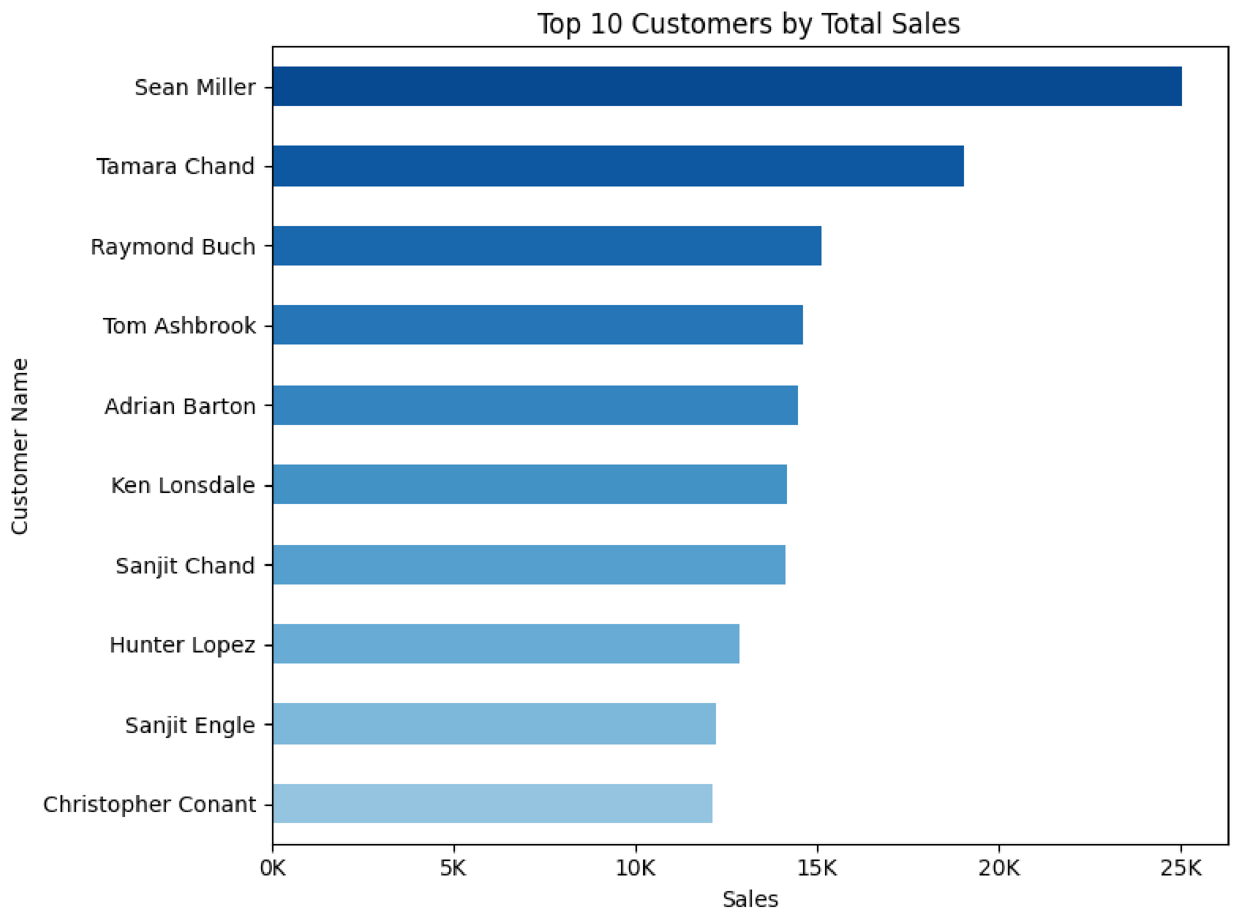
- What is the average sales by customer?

```
df.groupby("Customer Name")["Sales"].sum().mean()
```

```
np.float64(2851.520063934426)
```

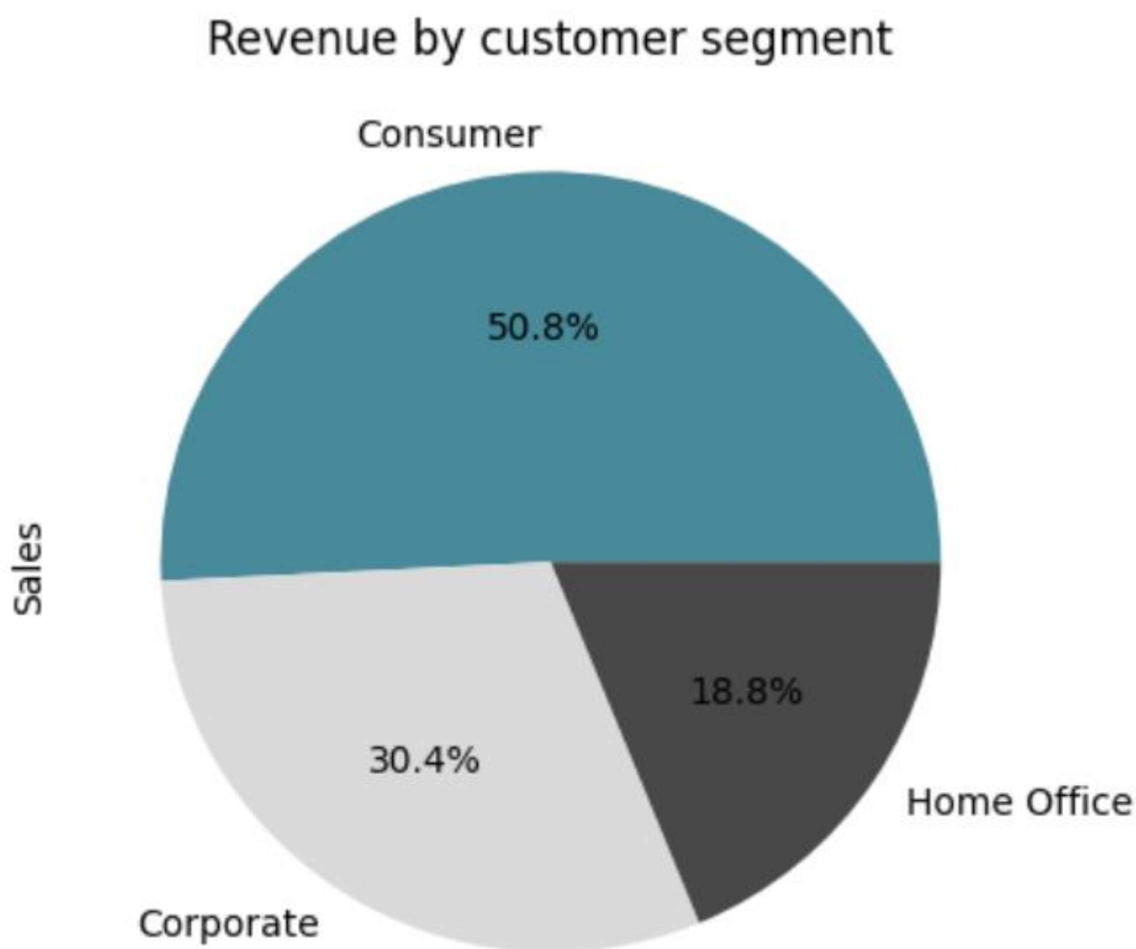
- What is the top customer by total sales?

```
top_customers=df.groupby("Customer Name")["Sales"].sum().nlargest(10).sort_values()
colors = ['#488A99'] * len(top_customers)
alphas = np.linspace(0.4, 1.0, len(top_customers))  # gradient from light to full color

cmap = cm.get_cmap('Blues')  # Use a blue colormap
colors = [cmap(i) for i in np.linspace(0.4, 0.9, len(top_customers))]  # Gradient steps

ax = top_customers.plot(kind='barh', color=colors, figsize=(8, 6))

# Format y-axis to show numbers in 'K'
ax.xaxis.set_major_formatter(FuncFormatter(lambda x, _: f'{int(x/1000)}K'))

plt.title("Top 10 Customers by Total Sales")
plt.xlabel("Sales")
plt.tight_layout()
plt.show()
```


Top 10 Customers by Total Sales
Sean Miller
Tamara Chand
Raymond Buch
Tom Ashbrook
Adrian Barton
Ken Lonsdale
Sanjit Chand
Hunter Lopez
Sanjit Engle
Christopher Conant
Customer Name
0K
5K
10K
15K
20K
25K
Sales

- What is the revenue by customer segment?

```
colors = ['#488A99','#DADADA' ,'#484848']

df.groupby("Segment")["Sales"].sum().sort_values(ascending=False).plot(kind="pie",autopct='%1.1f%%',colors=colors)
plt.title("Revenue by customer segment")
```



```
# What is the total sales revenue?
total_revenue = df['Sales'].sum()
print(f'Total Sales Revenue: ${total_revenue:,.2f}')
```

```
Total Sales Revenue: $2,261,255.41
```

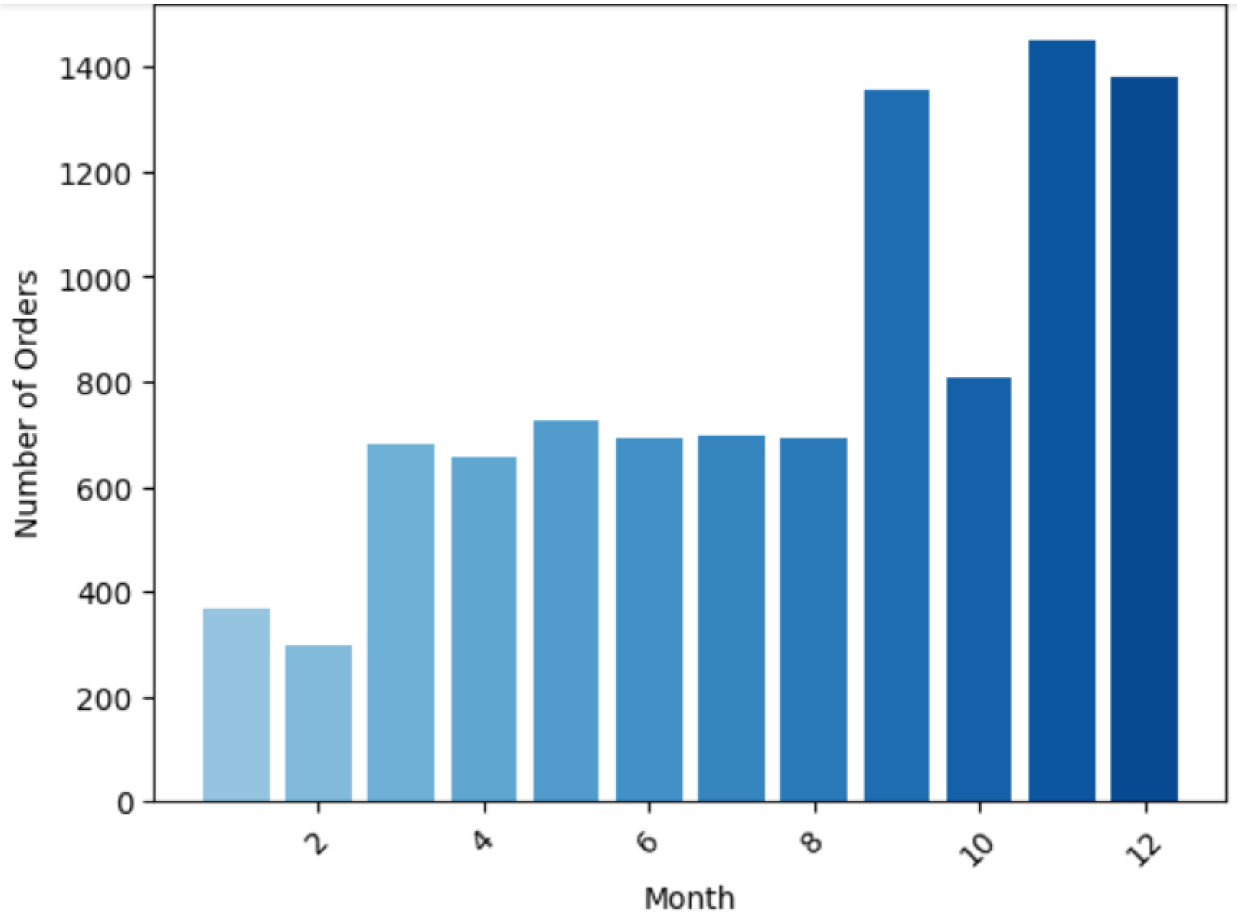
- Making  new column for “Year” and “ Month” for the best trend analysis

```
#make new column for order year
df['order_year']=df['Order Date'].dt.year
# make new column for order month
df['order_month']=df['Order Date'].dt.month

df.shape
```

```
(9799, 18)
```

- Most order month by customer through 4 years

```
#the most order month used by customers

# Count orders by month
month_orders = df['order_month'].value_counts().sort_index()

# Create gradient blue colors
cmap = cm.get_cmap('Blues')
colors = [cmap(i) for i in np.linspace(0.4, 0.9, len(month_orders))]

plt.bar(month_orders.index, month_orders.values, color=colors)
plt.xlabel("Month")
plt.ylabel("Number of Orders")
plt.title("Most Ordered Months by Customers")
plt.xticks(rotation=45) # rotate x axis label
plt.figure()
```




-November is the top in sales months and February is the bottom.

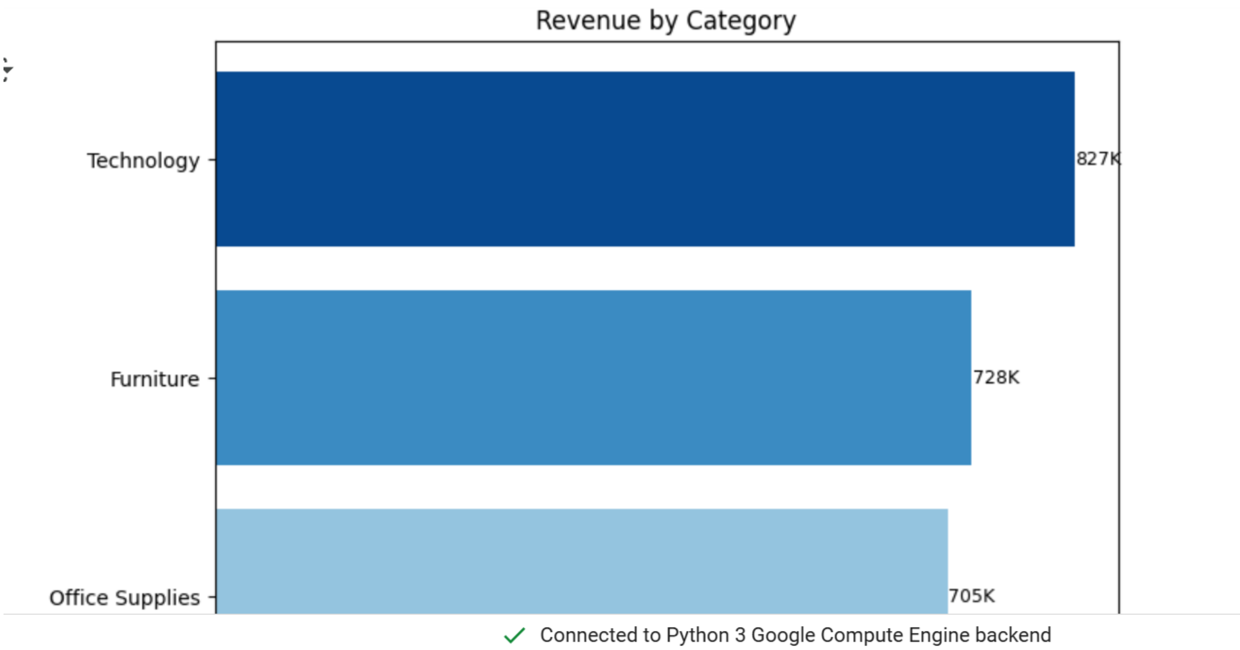
## Explore Product analysis:

- Revenue by category

```
# Which product category generates the highest revenue?
category_revenue = df.groupby('Category')['Sales'].sum().sort_values(ascending=True)

# Create gradient blue colors
cmap = cm.get_cmap('Blues')
colors = [cmap(i) for i in np.linspace(0.4, 0.9, len(category_revenue))]

# Plot
fig, ax = plt.subplots(figsize=(8,6))
bars = ax.barh(category_revenue.index, category_revenue.values, color=colors)

# Format x-axis with 'K'
ax.xaxis.set_major_formatter(FuncFormatter(lambda x, _: f'{int(x/1000)}K'))

# Add value labels next to each bar
for bar in bars:
    width = bar.get_width()
    ax.text(width + 500, bar.get_y() + bar.get_height()/2, f'{int(width/1000)}K',
```


```
# Add value labels next to each bar
for bar in bars:
    width = bar.get_width()
    ax.text(width + 500, bar.get_y() + bar.get_height()/2, f'{int(width/1000)}K',
```


```
            va='center', fontsize=9, color='black')

plt.title("Revenue by Category")
plt.xlabel("Revenue")
plt.show()
```




-Technology represents the most revenue category and office supplies the least revenue.

- Check average sales by category

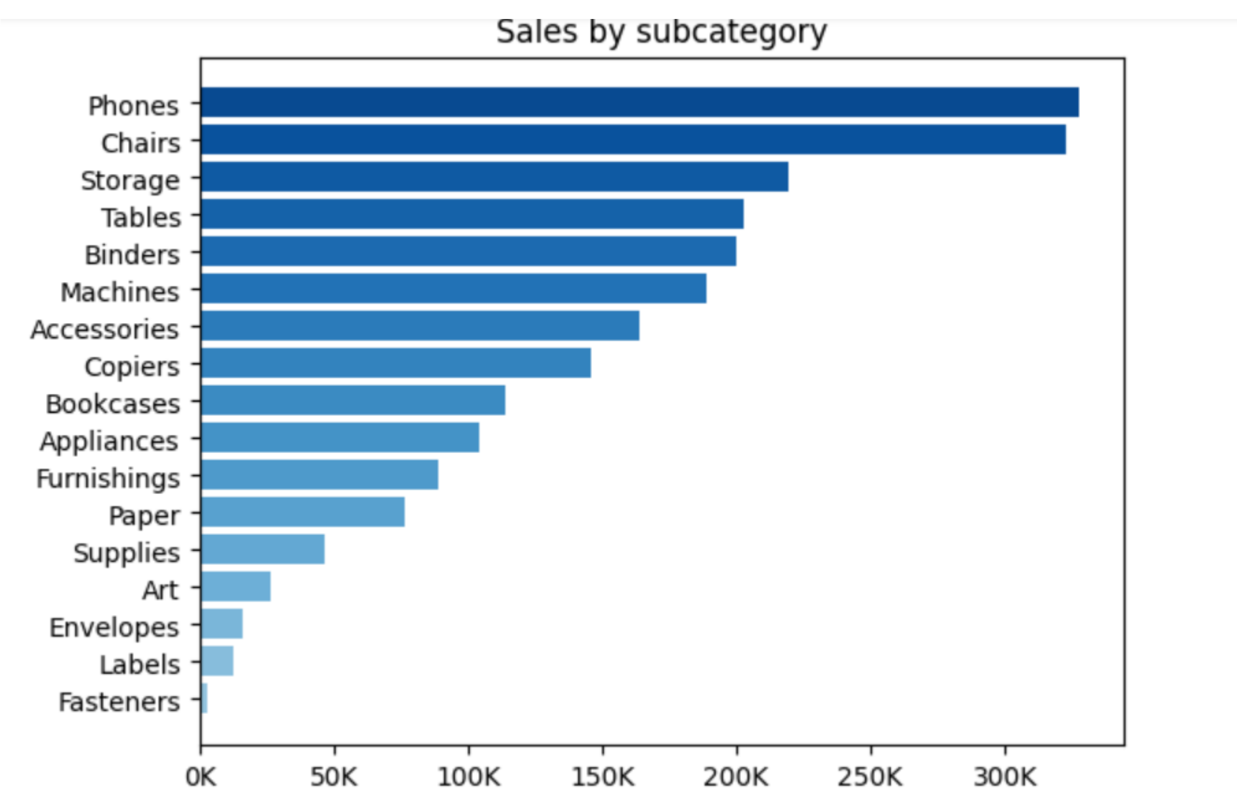
```
#Average sales by categories
df.groupby('Category')['Sales'].sum().mean()
```

```
np.float64(753751.8035666667)
```

Sales by category was almost $754k.

- Sales by subcategory

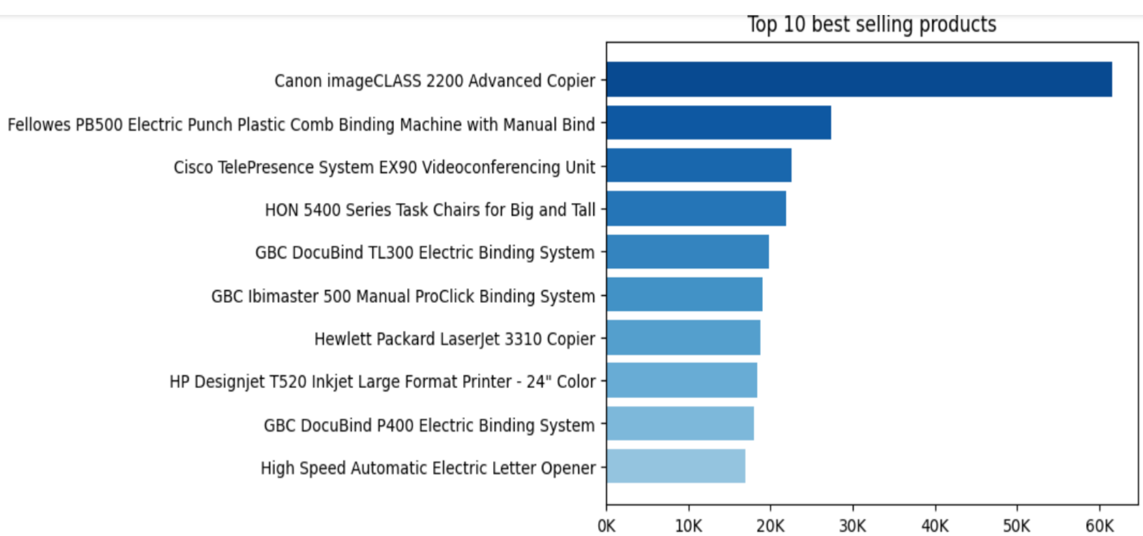
```
# Which sub-category has the highest sales?
subcategory_sales = df.groupby('Sub-Category')['Sales'].sum().sort_values(ascending=True)
# Create gradient blue colors
cmap = cm.get_cmap('Blues')
colors = [cmap(i) for i in np.linspace(0.4, 0.9, len(subcategory_sales))]
fig, ax = plt.subplots()
bars = ax.barh(subcategory_sales.index, subcategory_sales.values, color=colors)

# Format x-axis with 'K'
ax.xaxis.set_major_formatter(FuncFormatter(lambda x, _: f'{int(x/1000)}K'))

plt.title("Sales by subcategory")
plt.show()
```



-Phones represent the highest  sales in sab category and fasteners were the bottom.

- Top 10 best selling products

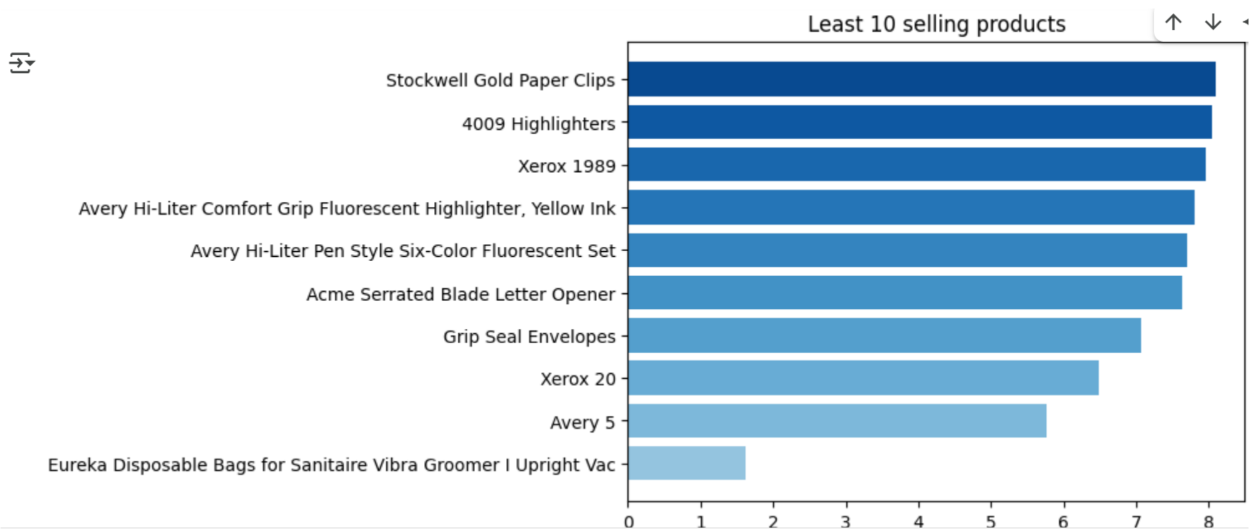
```
# whate  are the top 10 best-selling products?
top_products = df.groupby('Product Name')['Sales'].sum().nlargest(10).sort_values(ascending=True)
# Create gradient blue colors
cmap = cm.get_cmap('Blues')
colors = [cmap(i) for i in np.linspace(0.4, 0.9, len(top_products))]
fig, ax = plt.subplots()
bars = ax.barh(top_products.index, top_products.values, color=colors)

# Format x-axis with 'K'
ax.xaxis.set_major_formatter(FuncFormatter(lambda x, _: f'{int(x/1000)}K'))

plt.title("Top 10 best selling products")
plt.show()
```




-“Canon image class2200 Advanced Copier” was the top sales by about $60k.

- Least selling products

```
# What are the least-selling products?
least_selling_products =  df.groupby('Product Name')['Sales'].sum().sort_values(ascending=True).head(10)
# Create gradient blue colors
cmap = cm.get_cmap('Blues')
colors = [cmap(i) for i in np.linspace(0.4, 0.9, len(least_selling_products))]

plt.barh(least_selling_products.index, least_selling_products.values, color=colors)
plt.title("Least 10 selling products")
plt.show()
```




## Explore sales analysis by region

- Sales by country

```
#best sales by countries
country_sales = df.groupby("Country")["Sales"].sum().sort_values(ascending=False)

# Format in $M
formatted_sales = country_sales.apply(lambda x: f"${x/1_000_000:.2f}M")
formatted_sales
```

| Country | Sales |
|---|---|
| **United States** | $2.26M |

**dtype:** object

Total sales was $2.26M

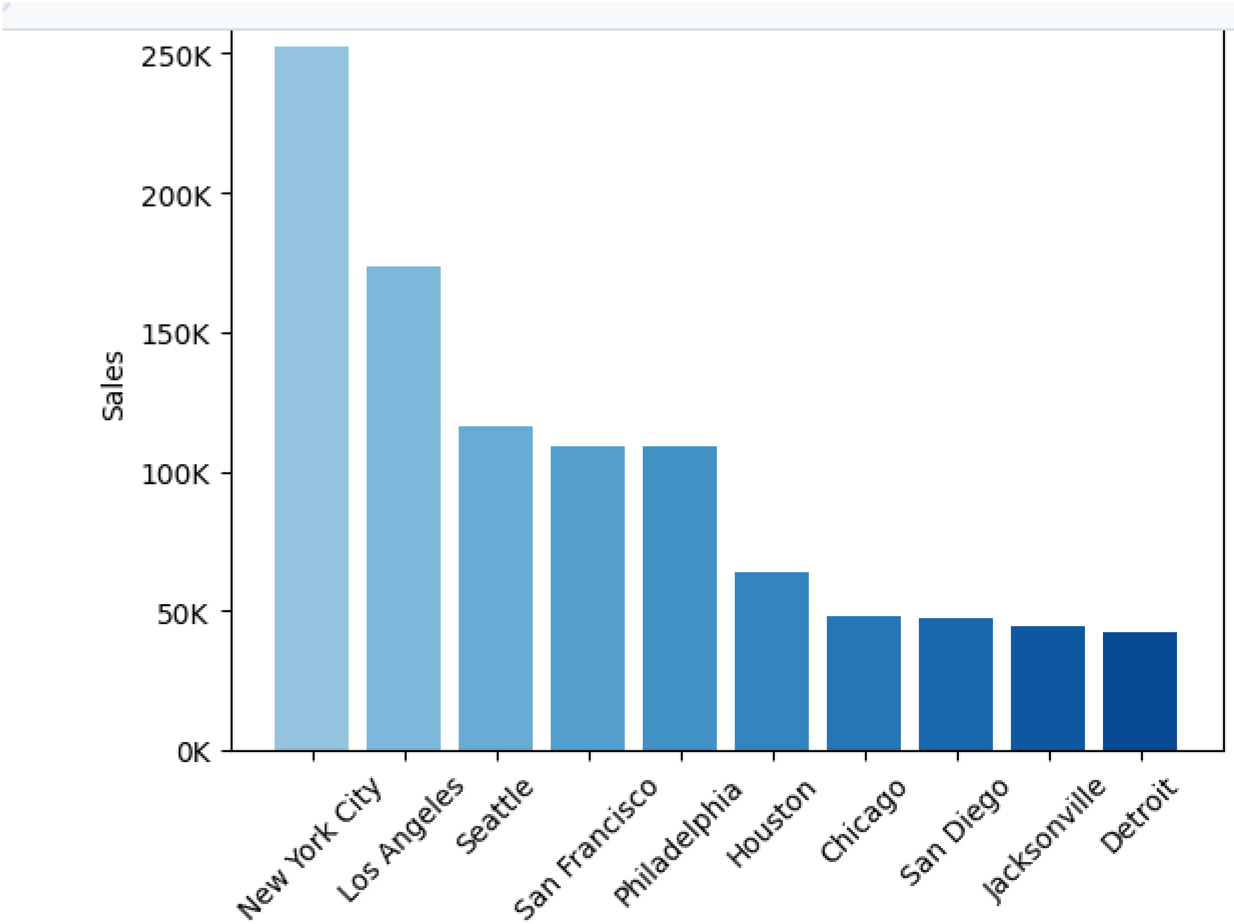
- Descriptive numbers about sales

```python
#sales insights
sales_insights = df["Sales"].agg(['mean', 'min', 'max'])
print(sales_insights)
```

```
mean       230.763895
min          0.444000
max      22638.480000
Name: Sales, dtype: float64
```

- Top 10 cities by sales

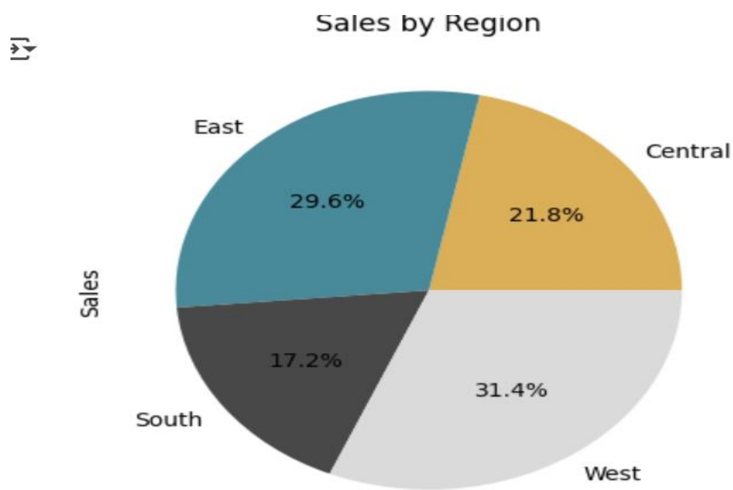
```python
#top 10 cities by sales
Top_cities=df.groupby("City")["Sales"].sum().sort_values(ascending=False).nlargest(10)
# Create gradient blue colors
cmap = cm.get_cmap('Blues')
colors = [cmap(i) for i in np.linspace(0.4, 0.9, len(Top_cities))]
# Create plot
fig, ax = plt.subplots( )
bars = ax.bar(Top_cities.index, Top_cities.values, color=colors)

# Format y-axis numbers with 'K'
ax.yaxis.set_major_formatter(FuncFormatter(lambda x, _: f'{int(x/1000)}K'))

# Rotate x-axis labels
plt.xticks(rotation=45)

ax.set_title("Top 10 Cities with Highest Sales")
ax.set_ylabel("Sales")
plt.show()
```




- New York ($250K) and Los Angeles($170k) were the Top

- Sales across different regions

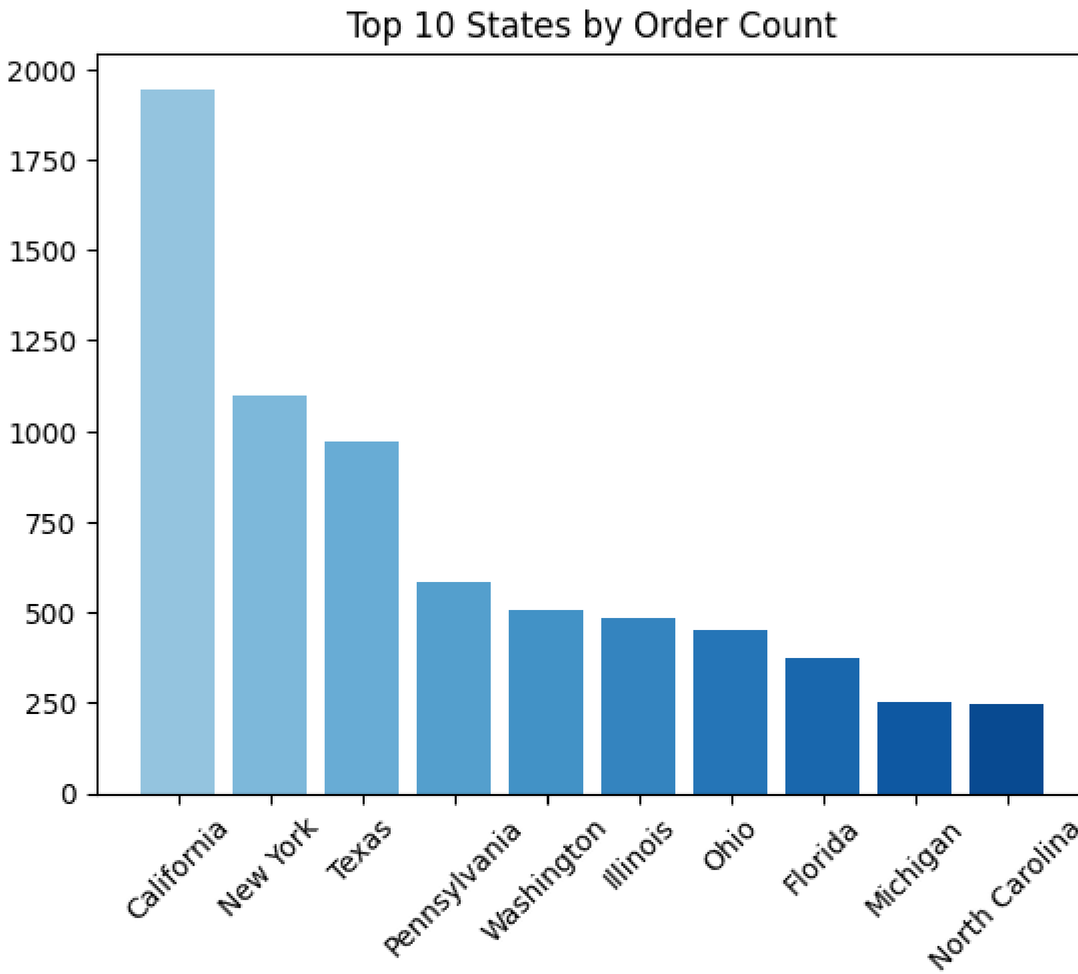
```
# How do sales vary across different regions?
colors=['#DBAE58','#488A99','#484848','#DADADA']
df.groupby("Region")["Sales"].sum().plot(kind="pie",autopct='%1.1f%%',colors=colors)
plt.title("Sales by Region")
```




- West and East were the Top by 60% of the sales so we should pay attention by them .
- While the South was the bottom by 17% .

- Order by states

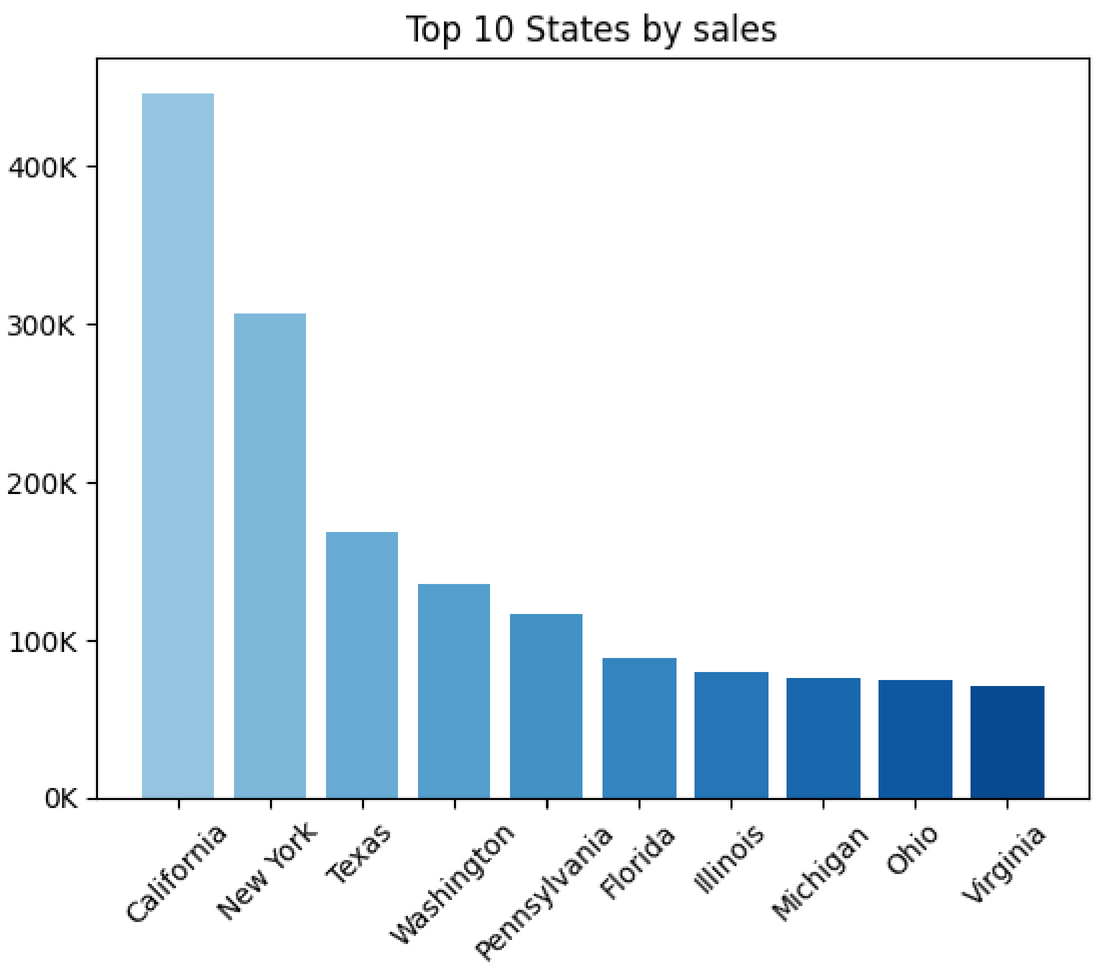
```
# What is the distribution of orders by state?
State_orders=df["State"].value_counts().head(10)  # Top 10 states by order count
#colors
cmap = cm.get_cmap('Blues')
colors = [cmap(i) for i in np.linspace(0.4, 0.9, len(Top_cities))]
plt.bar(State_orders.index , State_orders.values,color=colors)
# Rotate x-axis labels
plt.xticks(rotation=45)
plt.title("Top 10 States by Order Count")
plt.xlabel("State")
#plt.ylabel("Number of Orders")
plt.show()
```




-California represents high state in orders

- What is the top 10 states by sales?

```
#top 10 states by sales
Top_states=df.groupby("State")["Sales"].sum().nlargest(10)
 # Create gradient blue colors
cmap = cm.get_cmap('Blues')
colors = [cmap(i) for i in np.linspace(0.4, 0.9, len(Top_states))]
fig, ax = plt.subplots()
bars = ax.bar(Top_states.index, Top_states.values, color=colors)

# Format x-axis with 'K'
ax.yaxis.set_major_formatter(FuncFormatter(lambda x, _: f'{int(x/1000)}K'))

plt.xticks(rotation=45)  #rotate x axis

plt.title("Top 10 States by sales")
#plt.xlabel("State")
#plt.ylabel("Sales")
```




-California represents the highest state in sales

### Explore shipping analysis

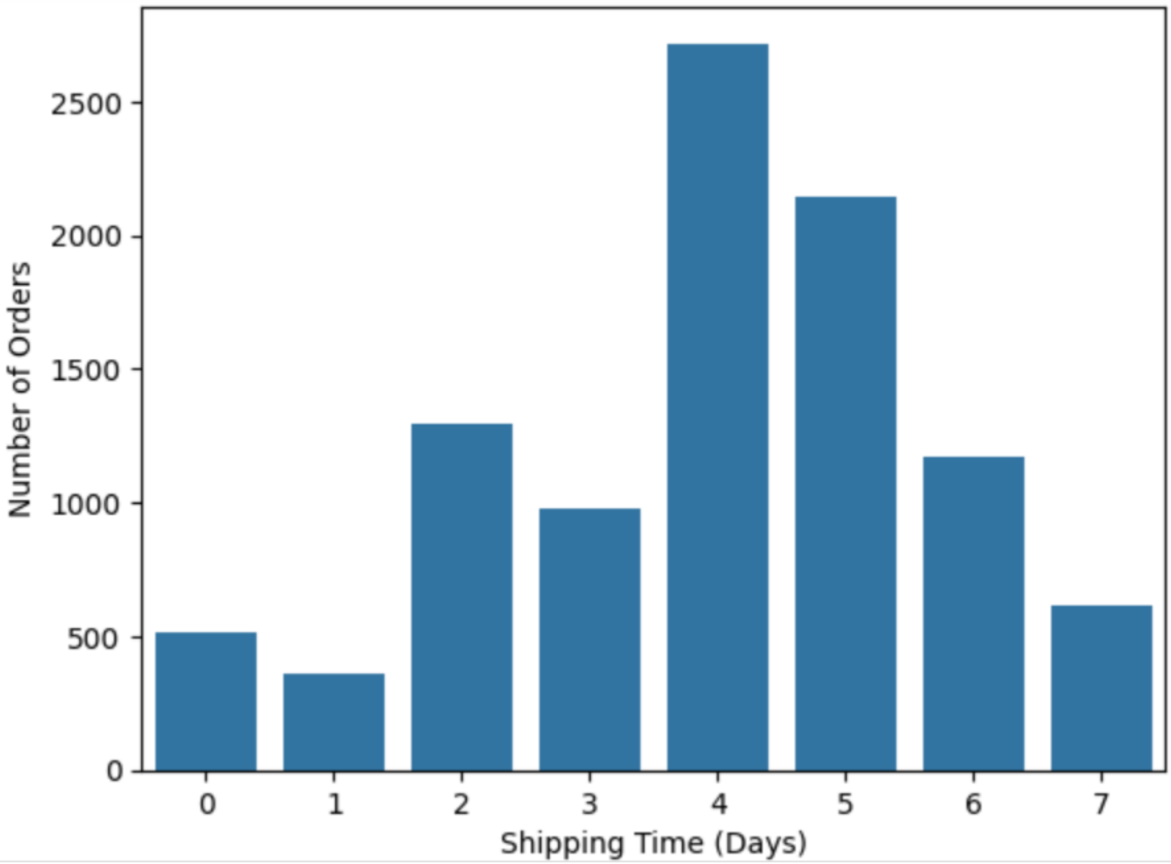
- Average shipping time

```
# calculate time shipping
df["Shipping Time"] = (df["Ship Date"] - df["Order Date"]).dt.days
average_shipping_time = df["Shipping Time"].mean()
print("Average Shipping Time:", average_shipping_time)
```

```
Average Shipping Time: 3.9611184814777016
```

- The average shipping time was 4 days !

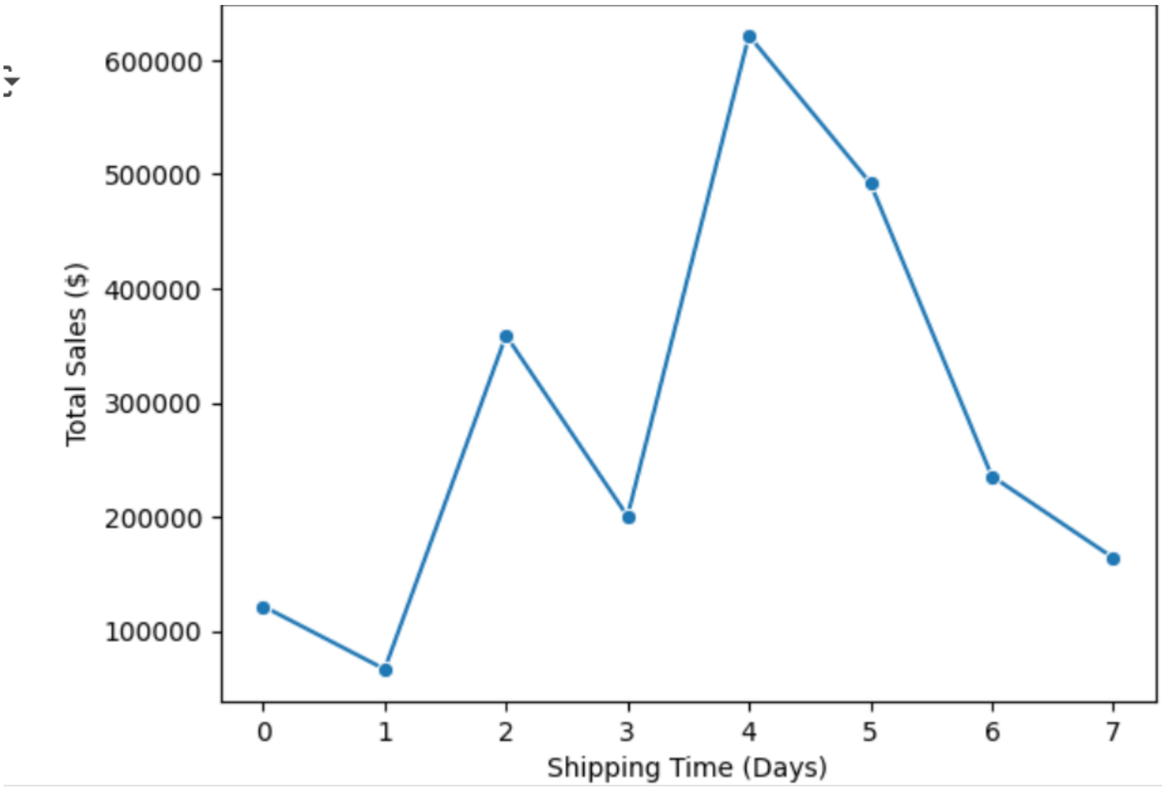
```
#Distribution for shipping time
plt.figure()
sns.barplot(x=df["Shipping Time"].value_counts().index, y=df["Shipping Time"].value_counts().values )
plt.xlabel("Shipping Time (Days)")
plt.ylabel("Number of Orders")
plt.title("Distribution of Shipping Time")
plt.show()
```




-Number of orders reach to more than 2500 order when the shipping time 4 days

- Effect of shipping time on sales

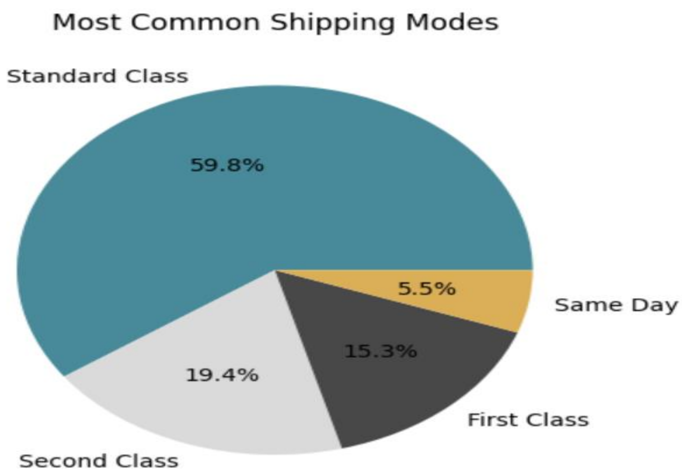
```
# effect of shipping time on sales
plt.figure()
sns.lineplot(x=df.groupby("Shipping Time")["Sales"].sum().index, y=df.groupby("Shipping Time")["Sales"].sum().values, marker="o")
plt.xlabel("Shipping Time (Days)")
plt.ylabel("Total Sales ($)")
plt.title("Effect of Shipping Time on Total Sales")
plt.show()
```




-The sales are very high when shipping time is 4 days and this is normal because the average shipping time was 4 days.

- Explore the shipping mode

```
# the most shipping time used
plt.figure()  #DBAE58     #DADADA
colors=['#488A99','#DADADA','#484848','#DBAE58']
plt.pie(df["Ship Mode"].value_counts(), labels=df["Ship Mode"].value_counts().index, autopct='%1.1f%%',colors=colors)
plt.title("Most Common Shipping Modes")
plt.show()
```

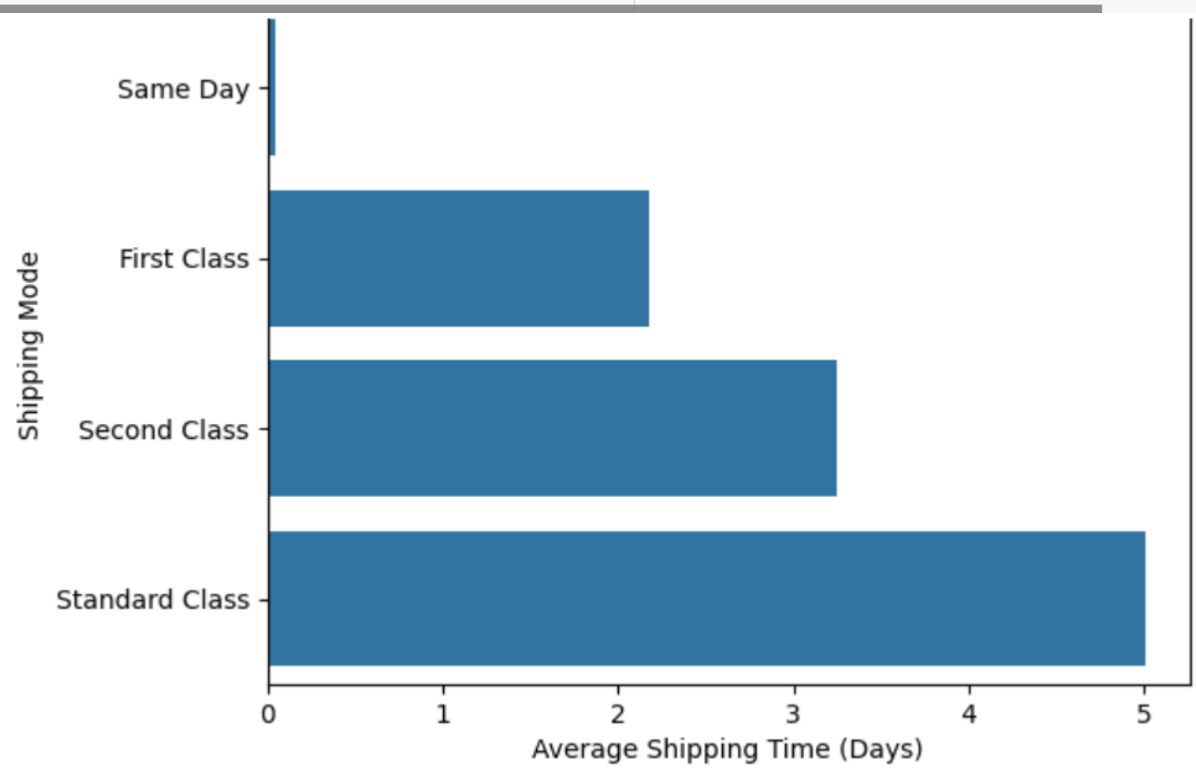



- customers segmented into four groups based on their preferred shipping mode.
- Standard class is the most common shipping mode(60%) of customers used, there for it was economical by them , while the same day orders were the bottom by 5.5% of orders as it was costly in shipping fees for customers.

- We also should pay more attention for the second class shipping mode -it shape about 20% of the orders.

- What is the fast-shipping type used?

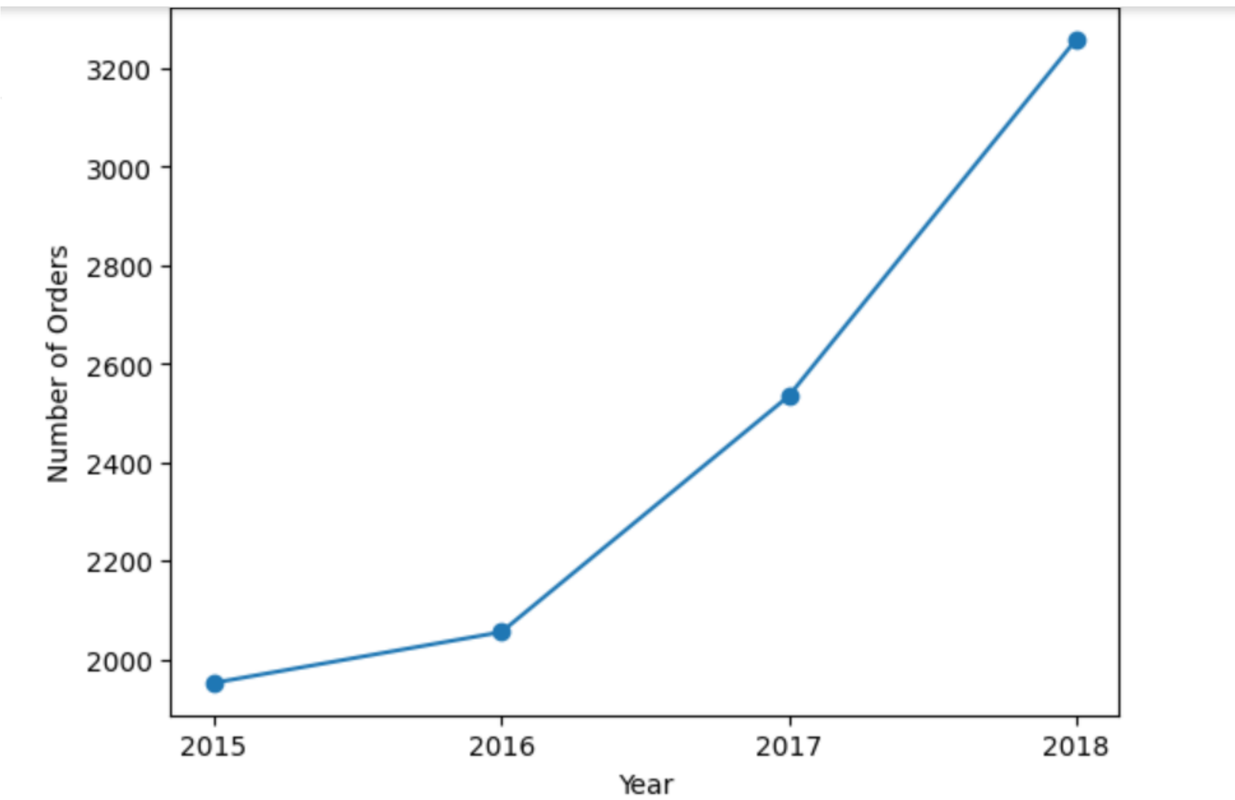
```
# The fast shipping type used
plt.figure()
sns.barplot(y=df.groupby("Ship Mode")["Shipping Time"].mean().sort_values().index, x=df.groupby("Ship Mode")["Shipping Time"].mean().sort_values().va
plt.ylabel("Shipping Mode")
plt.xlabel("Average Shipping Time (Days)")
plt.title("Average Shipping Time by Mode")
plt.show()
```



### Trend and seasonal analysis

- Number of orders for each year

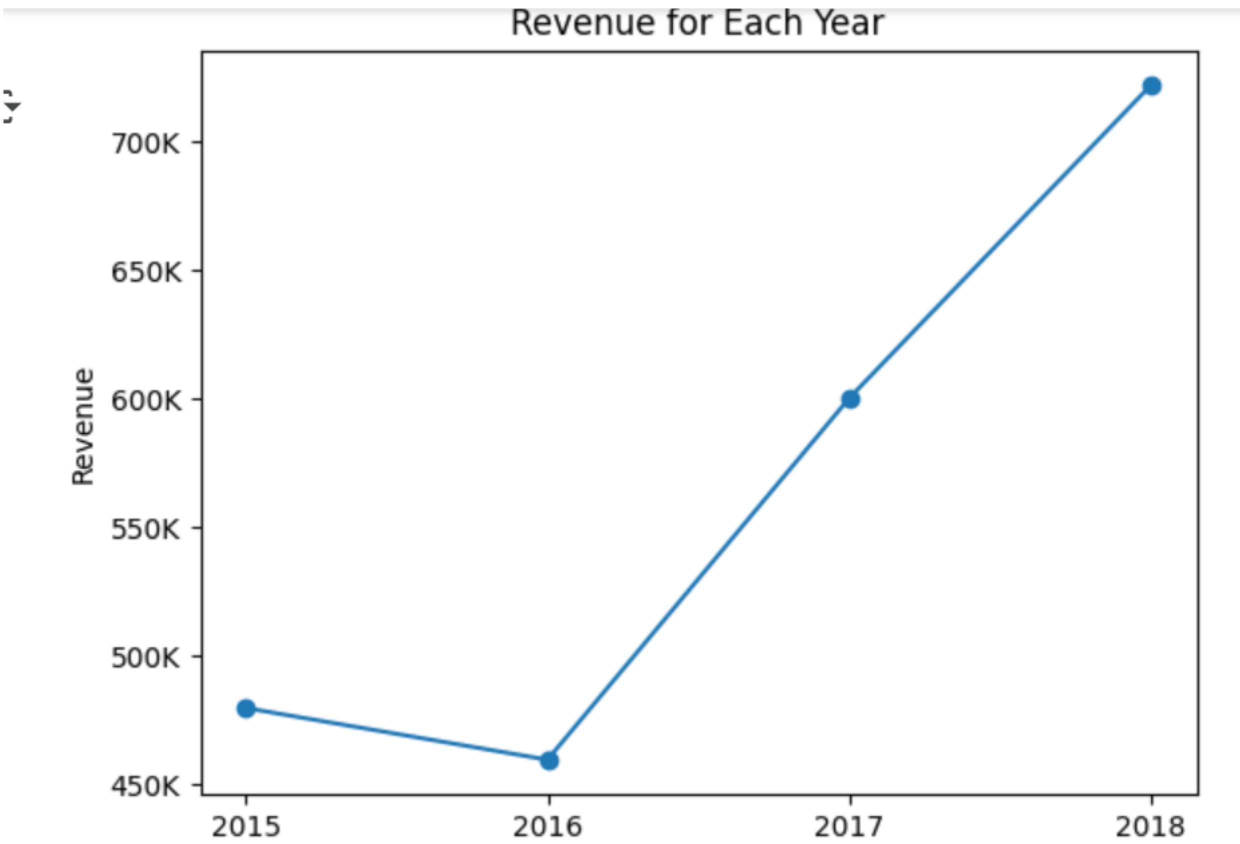
```
# number of orders for each year
order_year=df['order_year'].value_counts().sort_index()
order_year.plot(kind="line" , marker='o')
plt.xlabel("Year")
plt.ylabel("Number of Orders")
plt.xticks(order_year.index)
plt.show()
```




- It is noticeable that the number of orders increases gradually each year, but this does not necessarily mean that total sales are higher, as it depends on the quantity and price of each product ordered.

- Revenue for each year

```python
# revenue for each year
revenue_year = df.groupby("order_year")["Sales"].sum()

ax = revenue_year.plot(kind="line", marker='o')
plt.xlabel("Year")
plt.ylabel("Revenue")  # Changed y-label
plt.title("Revenue for Each Year")
plt.xticks(revenue_year.index)

# Format the y-axis to display values in thousands (K)
formatter = FuncFormatter(lambda x, pos: f'{x / 1000:,.0f}K')
ax.yaxis.set_major_formatter(formatter)

plt.show()
```




- The revenue is very high in 2018 and low in 2016.

- Check first and last month by sales data

```
# check first and last months in sales in dataset
df[df['order_year']==2015 ]['order_month'].min()
```

```
1
```

```
[ ] # check last month in sales
df[df['order_year']==2018]['order_month'].max()
```

```
12
```

- Create visuals for seasonal analysis

Made "order date" as the index to make better time series analysis with python.

```
#sales trend by quarters

# Set 'Order Date' as index
# important because time series analysis requires the date to be the index for resampling and trend analysis.

df.set_index('Order Date', inplace=True)
```

```python
import matplotlib.pyplot as plt
from matplotlib.ticker import FuncFormatter
import seaborn as sns
import numpy as np

# Set base blue color
blue = "#488A99"

fig, axes = plt.subplots(2, 2, figsize=(12, 8))

# 1. Sales trend over years
year_trend = df.groupby(df['order_year'])['Sales'].sum()
year_trend.plot(kind='line', ax=axes[0, 0], marker='o', linewidth=2, color=blue)

axes[0, 0].set_title("Total Sales from 2015-2018")
axes[0, 0].set_ylabel("Sales (K)")
axes[0, 0].yaxis.set_major_formatter(FuncFormatter(lambda x, _: f'{int(x/1000)}K'))

# Manually set x-ticks to avoid float fractions
axes[0, 0].set_xticks(year_trend.index)
axes[0, 0].set_xticklabels([str(int(year)) for year in year_trend.index])
```

```python
# 2. Quarterly sales trend
df['Sales'].resample('Q').sum().plot(kind='line', marker='o', linestyle='-', ax=axes[0, 1], color=blue)
axes[0, 1].set_title("Quarterly Sales Trend")
axes[0, 1].set_ylabel("Sales (K)")
axes[0, 1].yaxis.set_major_formatter(FuncFormatter(lambda x, _: f'{int(x/1000)}K'))

# 3. Monthly sales trend comparison
df['Year'] = df.index.year
df['Month'] = df.index.month
df_monthly_trend = df.groupby(['Year', 'Month'])['Sales'].sum().unstack(level=0)
df_monthly_trend.plot(marker='o', linestyle='-', ax=axes[1, 0])  # Keep default color for comparison
axes[1, 0].set_title("Monthly Sales Trend Comparison (2015-2018)")
axes[1, 0].set_xlabel("Month")
axes[1, 0].set_ylabel("Sales (K)")
axes[1, 0].set_xticks([1, 4, 8, 12])
axes[1, 0].set_xticklabels(['Jan', 'Apr', 'Aug', 'Dec'])
axes[1, 0].yaxis.set_major_formatter(FuncFormatter(lambda x, _: f'{int(x/1000)}K'))
```

```python
# 4. Effect of shipping time on sales
shipping_sales = df.groupby("Shipping Time")["Sales"].sum()
sns.lineplot(x=shipping_sales.index, y=shipping_sales.values,
             marker="o", ax=axes[1, 1], color=blue)
axes[1, 1].set_title("Effect of Shipping Time on Total Sales")
axes[1, 1].set_xlabel("Shipping Time (Days)")
axes[1, 1].set_ylabel("Sales (K)")
axes[1, 1].yaxis.set_major_formatter(FuncFormatter(lambda x, _: f'{int(x/1000)}K'))

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


**Summary about this chart:**

**Trend By years:**

**-** Sales increased significantly from **2016 to 2018**.

- **2016** experienced the lowest sales ($460K), followed by a strong recovery in **2017** and peaking in **2018** ($720K).

- This suggests improved performance and possibly better strategies or market conditions in the later years.

**Trend By quarters trend:**

-Sales tend to fluctuate each quarter, but overall growth is visible toward the end.

-There's a noticeable peak in late 2018, reflecting strong year-end performance.

-Regular dips and rises could be linked to seasonal or promotional events (e.g., back to school, holidays).

**Trend By Monthly Sales Trend:**

- Across all years, **sales peak in November**, likely due to year-end sales like **White Friday** or holiday shopping.
- **February** tends to show the **lowest sales**, possibly due to fewer shopping occasions.
- In general, **2018** outperformed all other years across most months, showing strong monthly performance consistency.

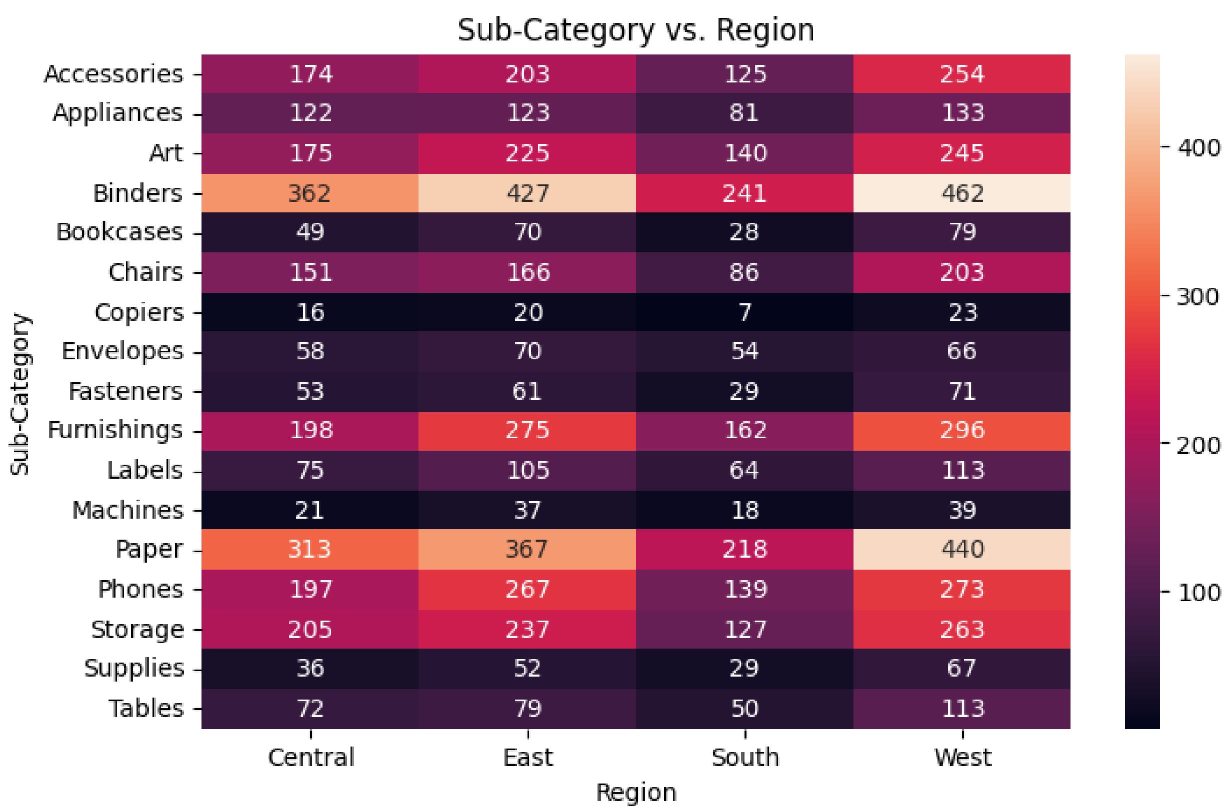
**Effect of Shipping Time on Sales:**

- Most sales occurred when shipping time was around 4 days, as shipping fees were suitable.
- Sales drop sharply for very fast (0–1 day) -as it was very cost in fees .
- long shipping durations (6–7 days) – we should work on this problem as long waits reduce customer satisfaction.

### Explore correlation between features

- Correlation between subcategory and Shipping mode

```
#correlation between product sub-category and shipping mode?
Subcategory_Region = pd.crosstab(df['Sub-Category'], df['Region'])
print(Subcategory_Region)

plt.figure(figsize=(8, 5))
sns.heatmap(Subcategory_Region, annot=True, fmt='d')
plt.title('Sub-Category vs. Region')
plt.xlabel('Region')
plt.ylabel('Sub-Category')
plt.show()
```




- From figure "Binders" were the most subcategory sold by Regions and "Copiers were the bottom sales by regions.

- "Bookcases", "Machines" , "Supplies" and "Copiers" were the least group Subcategories by sales .

- "Binders" and "Paper" were the Top

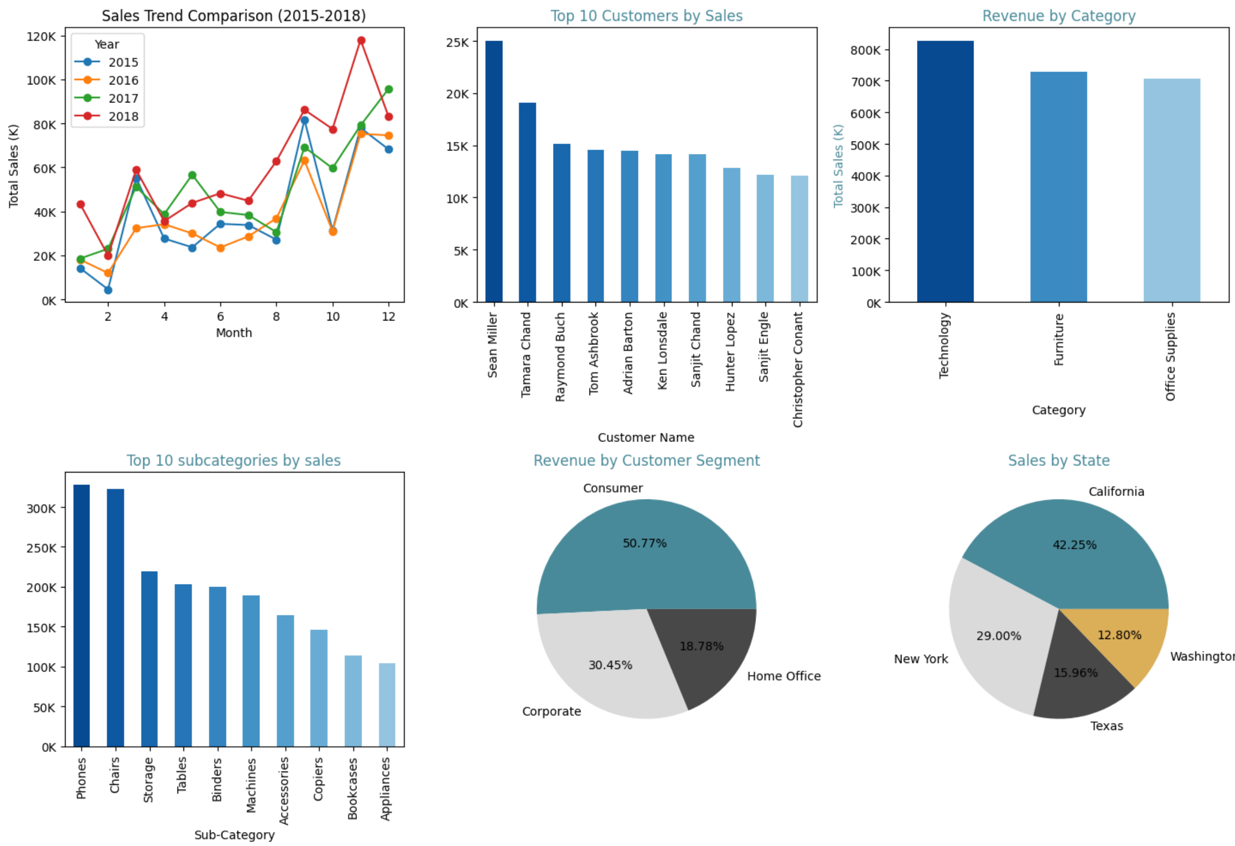
## 5.Create dashboard with the most important information

Summrized  most important information in a dashboard using python

```
# Define custom colors
custom_color = "#488A99"
pie_colors = ['#488A99', '#DADADA', '#484848', '#DBAE58']
cmap = cm.get_cmap('Blues')  # Blue color gradient

# Data aggregations
top_customers = df.groupby("Customer Name")["Sales"].sum().nlargest(10)
cat_sales = df.groupby("Category")["Sales"].sum().sort_values(ascending=False)
sub_sales = df.groupby("Sub-Category")["Sales"].sum().nlargest(10)

# Set reversed gradient colors (darker for high values, lighter for low)
customer_colors = [cmap(i) for i in np.linspace(0.9, 0.4, len(top_customers))]
category_colors = [cmap(i) for i in np.linspace(0.9, 0.4, len(cat_sales))]
subcat_colors = [cmap(i) for i in np.linspace(0.9, 0.4, len(sub_sales))]

# Create a 2x3 subplot layout
fig, axes = plt.subplots(2, 3, figsize=(15, 10))

# 1. Monthly Sales Trend Comparison
df_monthly_trend.plot(marker='o', linestyle='-', ax=axes[0, 0])
axes[0, 0].set_title("Sales Trend Comparison (2015-2018)")
axes[0, 0].set_ylabel("Total Sales (K)")
axes[0, 0].yaxis.set_major_formatter(FuncFormatter(lambda x, _: f'{int(x/1000)}K'))
axes[0, 0].set_xlabel("Month")
```

```python
# 2. Top 10 Customers by Sales
top_customers.plot(kind="bar", color=customer_colors, ax=axes[0, 1])
axes[0, 1].set_title("Top 10 Customers by Sales", color=custom_color)
axes[0, 1].yaxis.set_major_formatter(FuncFormatter(lambda x, _: f'{int(x/1000)}K'))

# 3. Revenue by Category
cat_sales.plot(kind="bar", color=category_colors, ax=axes[0, 2])
axes[0, 2].set_title("Revenue by Category", color=custom_color)
axes[0, 2].set_ylabel("Total Sales (K)", color=custom_color)
axes[0, 2].yaxis.set_major_formatter(FuncFormatter(lambda x, _: f'{int(x/1000)}K'))

# 4. Top Subcategories by Sales
sub_sales.plot(kind="bar", color=subcat_colors, ax=axes[1, 0])
axes[1, 0].set_title("Top 10 subcategories by sales", color=custom_color)
axes[1, 0].yaxis.set_major_formatter(FuncFormatter(lambda x, _: f'{int(x/1000)}K'))

# 5. Revenue by Segment - Pie Chart
df.groupby("Segment")["Sales"].sum().plot(
    kind="pie", autopct="%.2f%%", colors=pie_colors, ax=axes[1, 1])
axes[1, 1].set_title("Revenue by Customer Segment", color=custom_color)
axes[1, 1].set_ylabel("")
```

```python
# 6. Sales by State - Pie Chart
df.groupby("State")["Sales"].sum().nlargest(4).plot(
    kind="pie", autopct="%.2f%%", colors=pie_colors, ax=axes[1, 2])
axes[1, 2].set_title("Sales by State", color=custom_color)
axes[1, 2].set_ylabel("")

# Adjust layout to prevent overlap
plt.tight_layout()
plt.show()
```




## 6- Tableau dashboard

Created interactive insightful analysis include:

-Sales analysis dashboard

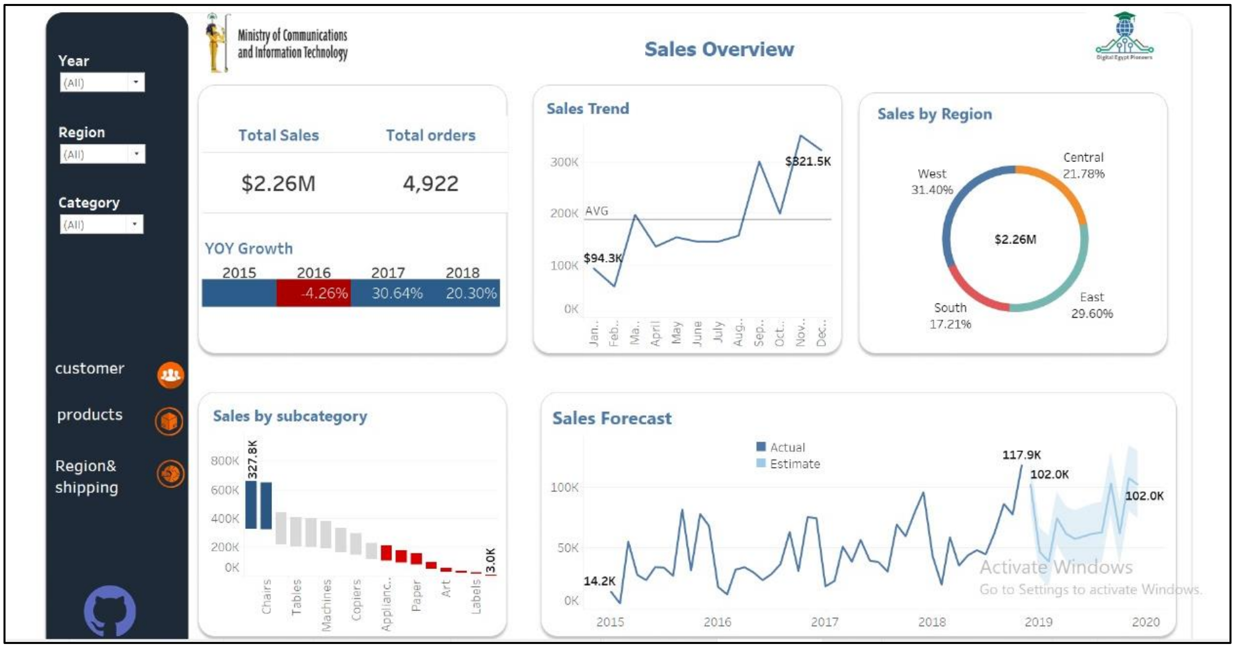
-make sales Forecast

- Product analysis dashboard

-Customer analysis dashboard

-Regional and shipping insights dashboard

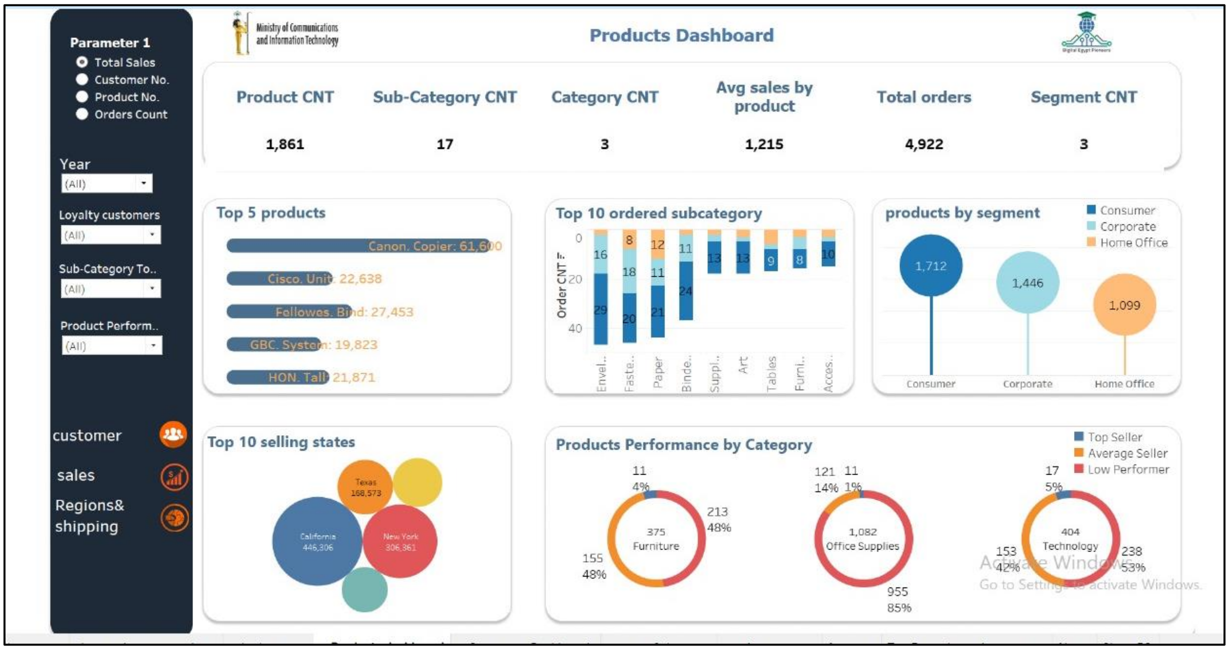
- **Sales dashboard**




This dashboard presents an overview of overall sales performance, trends, and forecasts:

- Total sales: $2.26M with 4,922 total orders
- Year-over-year growth: Notable increase in 2017 (30.64%) and 2018 (20.30%)
- Monthly sales trend shows peaks in November and December
- Regional sales: West leads (31.4%), followed by East and Central
- Sales by subcategory: Chairs and Tables dominate
- Forecasting shows growth opportunities in future months

- **Product dashboard**




We categorized products based on total sales performance as follows:

- *Top Sellers*: Top 25% of products by sales amount

- *Low Performers*: Products with sales below the overall average (approximately 18% of total sales )

- *Regular Products*: All other products

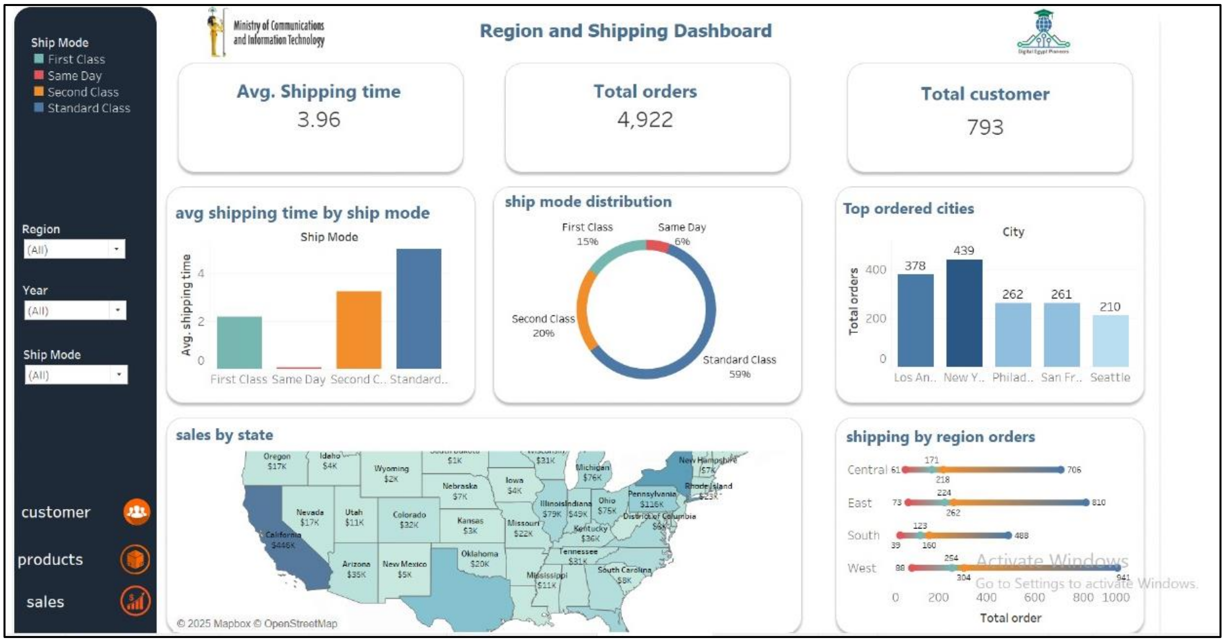
The analysis revealed that *around 72% of products* are underperforming in terms of sales.

- **Customer analysis dashboard**


This dashboard focuses on customer segmentation, loyalty, and performance. It highlights:

- Total of 793 unique customers and their distribution across 49 states and 529 cities
- Customer loyalty classification: 3% Loyal, 54% Regular, 43% Occasional
- Customer segments by region (Consumer, Corporate, Home Office)
- Sales trends by customer segment and year
- Top 10 customers by order value
- Parameters showing customer behavior patterns

- **Region and shipping insights dashboard**




This dashboard provides a comprehensive overview of shipping performance across different regions and ship modes. It displays key metrics such as the average shipping time (3.96 days), total orders (4,922), and total customers (793). Visuals include:

- Average shipping time by ship mode

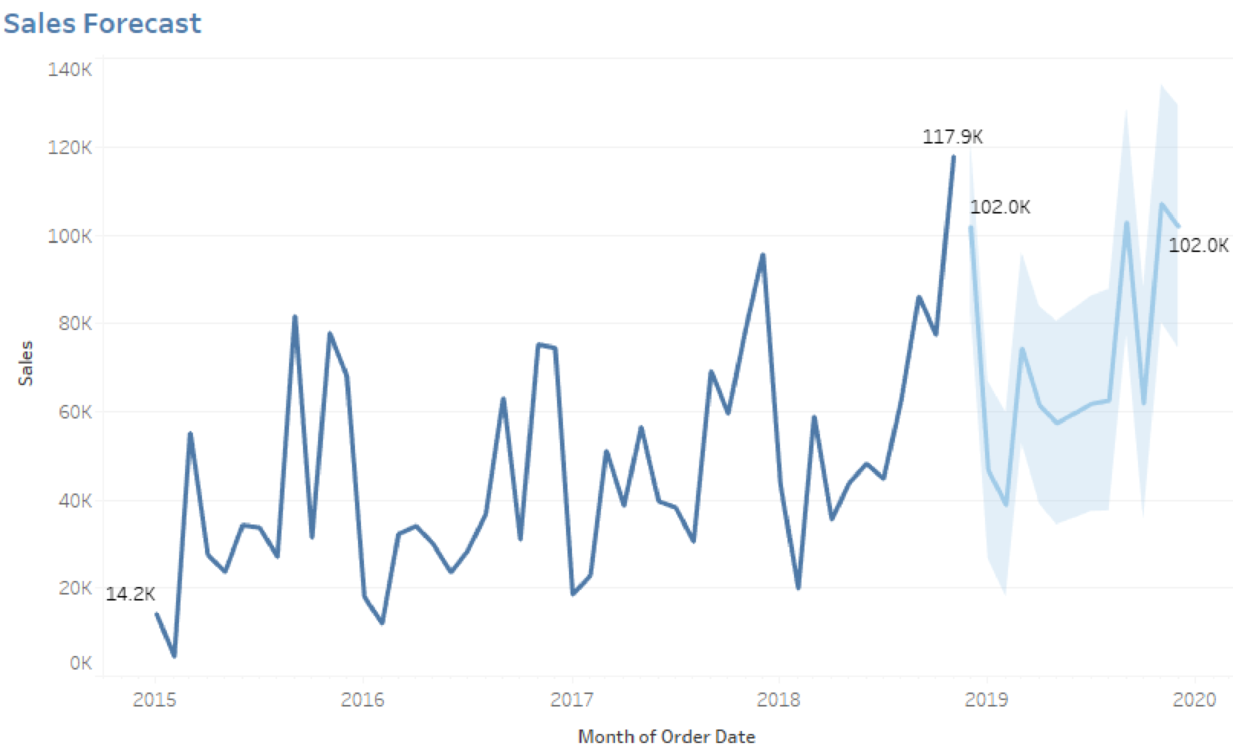
- Ship mode distribution (Standard Class is the most used at 59%)

- Top ordering cities (New York and Los Angeles lead)

- Shipping distribution by region (highest from the West and East)

- Sales performance across US states

## Sales Forecast




### Sales Forecast

| Forecast i.. | 2015 | 2016 | 2017 | 2018 | 2019 |
|---|---|---|---|---|---|
| **Actual** | 479.6K | 459.4K | 600.2K | | |
| **Estimate** | | | | 673.4K | 671.9K |

**Stability Prediction:**

- The forecast for 2019 is slightly lower than 2018 but very close, suggesting that the market may be reaching a maturity phase or expecting consistent performance without aggressive expansion.

- The sales forecast shows a strong recovery after a slight decline in 2016, with projections indicating continued growth and eventual stabilization.

- These trends support the effectiveness of strategic adjustments made in 2017, and the forecasts suggest confidence in maintaining high performance into 2019.

## Insights after analysis

- **Total sales** were $2.26M
- Average sales by category were 753.751 k
- It is noticeable that the number of **orders increases** gradually each year, but this does not necessarily mean that total sales are higher, as it depends on the quantity and price of each product ordered.
- **Sales increase** in "March", "September" and become highest peak at "November" also **sales deacrsed** in "February" and "October" in each year. I believe this increase is due to the beginning of a new season, such as summer, sales increase as customer purchases more items.
- Additionally, in September, the start of a new school year contributes to higher sales, while in November, the rise is driven by White Friday discounts.
- **Revenue for each year** increased gradually except year 2016 the sales decreased by 4.2% from 2015.
- **The bottom year** in sales was 2016 (460K), after this year the sales increase gradually and reach the Top in 2018(722K ).
- **"Consumer" Segment** have 50% of Purchases so we should pay attention by this segment for a better future of our business.
- **Technology products** were the top category in sales and **"office supplies"** were the bottom.
- "Phones" were **the Top** sales(327K).
- "California", "New York" , "Texas" were **the Top states** by sales.
- "New York","Los Angeles" and" Seattle" were **the Top Cities** by sales.
- Top our **loyalty Customers** were: " Sean Miller", Tamara Chand"," Raymond Buch".

- **60%** of **shipping modes** preferred by customers was "Standard Class" there is because it was suitable in money.
- **Average shipping time** was 4 days from complete the order to deliver the customer.

- **Total Sales**: $2.26M
- **Average Sales** by Category: $753.75K

**Key Insights:**

- The number of orders increased gradually each year. However, this did not always result in higher total sales, as sales depend on both product quantity and price.

**Sales Trends by Month:**

- Sales tend to peak in March, September, and November, with November showing the highest sales—likely due to White Friday discounts.
- Sales typically drop in February and October each year.
- The increase in March may be related to seasonal purchases, while September growth aligns with the back-to-school season.

**Yearly Revenue Performance:**

- Revenue increased steadily except in 2016, which saw a 4.2% decrease compared to 2015.
- The lowest sales year was 2016 (460K), while2018recordedthehighestsales(722K).

**Customer Segments & Preferences:**

- The "Consumer" segment contributed to 50% of all purchases. This group should be a key focus for future marketing and sales strategies.

**Preferred Shipping Mode:**

- 60% of shipping modes preferred by customers was "Standard Class" —likely due to its cost-effectiveness.

**Average Shipping Time:**

- Approximately 4 days from order completion to delivery.

**Top Performers:**

- Top Product Category: Technology
- Lowest Sales Category: Office Supplies
- Top-Selling Product: Phones ($327K)
- Top States by Sales: California, New York, Texas
- Top Cities by Sales: New York, Los Angeles, Seattle
- Top Loyal Customers: Sean Miller, Tamara Chand, Raymond Buch

---

## 7- Business Recommendations

**Focus on the Consumer Segment**

- The Consumer segment accounts for 50% of total sales. Enhancing customer service, offering loyalty programs, and targeted promotions for this segment could significantly boost revenue.

**Capitalize on Peak Months**

- November, March, and September consistently show high sales. Plan seasonal campaigns and promotions ahead of these months.

**Marketing Campaigns**

- Consider stocking up inventory before these peaks and running marketing campaigns aligned with seasonal demand (e.g., Back-to-School in September, Black/White Friday in November).

**Study and enhance the Drop in February and October**

- Develop strategies to boost sales during slower months. Options include limited-time discounts, product bundles, or free shipping offers in February and October.

**Improve Revenue Consistency**

- 2016 saw a 4.2% drop in sales. Analyzing causes (e.g., supply chain, product mix, or external factors) and creating risk mitigation plans can help maintain consistent growth.

**Expand Best-Selling Categories**

- Technology (especially Phones) is the top-performing category. Consider introducing new tech products, accessories, or exclusive deals to boost upselling and cross-selling.

**Optimize Shipping Strategy**

- 60% of orders used Standard Class shipping. Explore options to make faster shipping more affordable, or offer free standard shipping thresholds to increase cart sizes.

**Invest in Top Regions**

- States like California, New York, and Texas, and cities such as New York City, Los Angeles, and Seattle generate the most revenue. Focus advertising, inventory, and service optimization efforts here to further capitalize.

**Leverage Top Customers**

- Loyal customers like Sean Miller, Tamara Chand, Raymond Buch (to 20 top selling customers) have strong purchasing power. Consider exclusive VIP programs, early access to sales, or referral rewards for this group.

**Diversify Product Offerings in Low-Performing Categories**

- "Office Supplies" is the weakest category. Review customer needs, improve product variety, or create value bundles to attract more purchases in this segment.

**Shorten Delivery Time (if possible)**

- Current average delivery is 4 days. Faster delivery could improve customer satisfaction and retention—especially for premium or urgent product lines.

**Project links :**

- **Github repository:**
  **https://github.com/fatma-elshall/DEPI_gradution_project**

## Contents