A Project Report

On

“Retail Sales and Profitability Analysis”

By

Mr. Khan Abu Hashim Mohammad Farooque

Professional Program in

Data Analytics

Submitted to
DVOC INSTITUTE
(Grant Road)

Under Guidance of
Prof. _____

`

INDEX

`

List of Figures

`

Executive Summary

This project presents a detailed **retail sales and profitability analysis** using the Superstore dataset. The goal of the study is to identify key performance drivers, understand customer and product behaviour, highlight profit leakage points, and derive data-backed recommendations to improve business performance.
The analysis reveals strong seasonality in monthly sales, with an upward trend in the later months. Technology and Office Supplies dominate sales, while Furniture shows weaker profitability. Sub-category analysis highlights Phones, Chairs, and Storage as strong performers, whereas Tables and Bookcases report heavy losses.

Geographically, the West region contributes the highest sales. Customer analysis shows that a small group of loyal customers generates a large share of total revenue. A key insight discovered is the **negative impact of heavy discounting on profitability**, as shown by the profit–discount scatter plot and correlation heatmap.

The findings from this project provide actionable insights for pricing optimization, product strategy improvements, inventory planning, and customer relationship management.

`

1. Introduction:

1.1 Overview of Retail Sales Analytics

Retail analytics refers to the systematic examination of retail data to uncover meaningful patterns, correlations, and insights. It includes analysing customer behaviour, product demand, pricing strategies, and financial performance. In modern businesses, data-driven decision-making is essential for understanding how products perform, which customers contribute the most value, and how pricing and discounts influence profit margins.

1.2 Purpose of Analysis

This project aims to examine retail sales and profitability trends using Python-based exploratory analysis. By analysing transactions from a retail store dataset, the project uncovers performance trends across categories, sub-categories, customers, and regions.

1.3 Problem Statement

The retail company wants answers to the following:

1. **Sales Trend Analysis:** How do sales fluctuate over months and years?
2. **Category Performance:** Which categories and sub-categories drive revenue and profit?
3. **Product Performance:** Which products generate high profit and which lead to losses?
4. **Regional Insights:** How do different geographic regions contribute to overall sales?
5. **Customer Insights:** Which customers are the most valuable to the business?
6. **Discount Impact:** How strongly do discounts influence profitability?
7. **Correlation Study:** How are key metrics like Sales, Profit, Quantity, and Discount related?

1.4 Aim of the Project

To perform a detailed retail sales and profitability analysis using Python and derive actionable insights that help improve business decision-making.

`

1.5 Objectives

- To clean and prepare the dataset for analysis using Python
- To engineer new features such as Year-Month and Shipping Time
- To analyse monthly sales trends for identifying seasonality
- To compare category-level and sub-category-level performance
- To identify the most profitable and least profitable products
- To study geographic variation through region-wise analysis
- To evaluate customer contribution through top-customer analysis
- To examine the impact of discounting on profit using scatter analysis
- To study correlations between key variables using a heatmap
- To provide recommendations to improve profitability

`

2. About the Data and Libraries

2.1 Dataset Origin and Nature

The Superstore dataset is a well-known retail dataset containing transactional sales records. It represents a real-world business scenario where companies track product-level sales, customer details, order timelines, profitability, and discount information.

2.2 Data Structure

Column Name	Description
Row ID	Unique row identifier for each record
Order ID	Unique identifier for each order
Order Date	Date when the order was placed
Ship Date	Date when the order was shipped
Ship Mode	Type of shipping (Standard, Second Class, etc.)
Customer ID	Unique ID for each customer
Customer Name	Name of the customer
Segment	Customer segment (Consumer, Corporate, Home Office)
Country	Country of the order (mostly United States)
City	City of the customer
State	State of the customer
Postal Code	ZIP/Postal code
Region	Geographic region (West, East, Central, South)
Product ID	Unique ID for each product
Category	Broad product category
Sub-Category	Detailed product sub-category
Product Name	Name of the product
Sales	Total sales value of the order line
Quantity	Number of units sold
Discount	Discount applied to the order line
Profit	Profit earned from the order line

2.3 Statistical Summary

The dataset contains:

- 9,994 rows
- 21 columns
- No significant missing values
- Data spanning multiple years

`

2.4 Google Colab — Cloud-Based Python Environment

What is Google Colab:
Google Colab is a free, cloud-based Jupyter Notebook environment provided by Google. It allows users to write and execute Python code directly in the browser without needing to install any software.
What Google Colab Provides:

- A pre-configured Python environment
- Built-in support for data science libraries (Pandas, NumPy, Matplotlib, Seaborn)
- Easy file upload/download features
- GPU/TPU support (if needed)
- Autosave and versioning using Google Drive

Why Google Colab:
No Installation Required You don't need to install Python or libraries manually. Everything works automatically.

1. Easy File Uploads The dataset was uploaded using:
2. from google.colab import files
3. files.upload()
   This makes data import simple and convenient.
4. Beginner-Friendly Interface Colab provides a notebook-style interface, perfect for step-by-step EDA and visualizations.
5. Supports All Charting Libraries Matplotlib and Seaborn charts render beautifully inside Colab.
6. Cloud Execution(Fast + Stable) The execution happens on Google's cloud servers, so it runs faster even on a low-spec laptop.
7. Integrated Output + Code View Colab combines:
   - Code
   - Output
   - Explanation
     in one place, making it suitable for academic reports.

Where we used Google Colab:

- Cleaning and preprocessing data
- Running EDA code cells efficiently
- Generating all required visualizations
- Keeping the analysis organized in one notebook
- Exporting results easily

Summary:
Google Colab acted as the primary environment for this retail EDA project. It provided a reliable, fast, and user-friendly platform to execute all analysis steps without the need for complex setups.

`

2.5 Libraries Used

1. Pandas — Data Cleaning, Data Preparation, and Analysis

What is Pandas:
Pandas is the most popular Python library for data manipulation, cleaning, and analysis. It provides powerful structures like Data Frames that allow you to load, clean, filter, group, and analyse datasets easily.

Where we used:

- Load the CSV dataset
- Remove duplicates
- Check missing values
- Convert Order Date & Ship Date into datetime format
- Create new features (Year, Month, Shipping Days, Year-Month)
- Group the data (by Category, Sub-category, Region, Customer, Product)
- Summarize key metrics (Sales, Profit, Discount, Quantity)

Why Pandas:
Because our project required heavy grouping, aggregations, feature engineering, and data cleaning. Without Pandas, none of this structured EDA would be possible.

2. NumPy — Numerical Computation & Support for Pandas

What is Numpy:
NumPy is the core numerical computing library in Python. It supports Pandas internally and is used for mathematical operations and array handling.

Where we used NumPy:
Used for generating index positions for bar charts (np.arange())

- Provides fast vectorized operations
- Powers Pandas calculations internally

Why NumPy:
Even though we did not write many explicit NumPy functions, NumPy is still essential because:

- Pandas is built on top of NumPy
- Chart indexing (np.arange) requires NumPy
- Efficient aggregations and calculations rely on NumPy arrays

`

3. Matplotlib — Base Plotting Library for Python

What is Matplotlib:
Matplotlib is the foundation of all plotting libraries in Python. It gives full control over styling, colors, axes, figure size, ticks, and layout.

Where we used Matplotlib:
Creates the base figure for each chart (plt.figure())

- Customizes axes, labels, titles
- Controls layout and spacing
- Adjusts rotation of x-axis labels
- Controls color, size, and formatting

Even though we used Seaborn for most charts, **Matplotlib is required to control customization**.

Why Matplotlib:
Without Matplotlib, your charts would look raw and unformatted

- Matplotlib improves chart presentation for reports
- Allows high-level formatting required in academic & professional reports

4. Seaborn — Statistical Visualization Library

What is Seaborn:
Seaborn sits on top of Matplotlib and provides **clean, beautiful, and professional charts with very little code**. It is ideal for statistical charts like bar plots, scatterplots, line charts, and heatmaps.

Where we used Seaborn:
Monthly sales line chart

- Category and sub-category charts
- Region-wise and customer-wise bar charts
- Discount vs Profit scatter plot
- Correlation heatmap

Why Seaborn:
High-quality visuals with professional styling

- Built-in themes make the report look clean
- Heatmap visualization is uniquely available in Seaborn
- Perfect for EDA (Exploratory Data Analysis)

Seaborn made your charts look **clean, modern, and presentation-ready**.

`

3. Data Cleaning and Processing

This section describes each step of your code, exactly as performed in the notebook.

3.1 Importing Libraries

Python libraries were imported, and Seaborn styling was set for consistent visualization:

- sns.set() applied Seaborn's default theme
- sns.set_palette("Blues") created a uniform blue color scheme

3.2 Loading the Dataset

The dataset was uploaded using Google Colab's file upload interface, and Pandas loaded the CSV file encoded in latin1.

3.3 Inspecting Data

df.shape and df.isnull().sum() were used to understand dataset size and missing values. No major missing value issues were found.

df.drop_duplicates(inplace=True) removed duplicate entries to ensure data accuracy.

3.4 Converting Date Fields

Both "Order Date" and "Ship Date" were converted to datetime using:

```
df['Order Date'] = pd.to_datetime(df['Order Date'])
df['Ship Date'] = pd.to_datetime(df['Ship Date'])
```

3.5 Feature Engineering

Shipping Days

Shipping time was calculated:

```
df['Ship_Days'] = (df['Ship Date'] - df['Order Date']).dt.days
```

Month and Year Extraction

Year and month were extracted for time-series analysis:

```
df['Year'] = df['Order Date'].dt.year
df['Month'] = df['Order Date'].dt.month_name()
```

Year-Month Column for Trend Analysis

```
df['Year_Month'] = df['Order Date'].dt.to_period('M').astype(str)
```

This allowed grouping by month across multiple years.

3.6 Data Quality Checks

- .describe() was used for statistical summary.
- .head() to preview initial records.
- .nunique() to count unique values across columns.

`

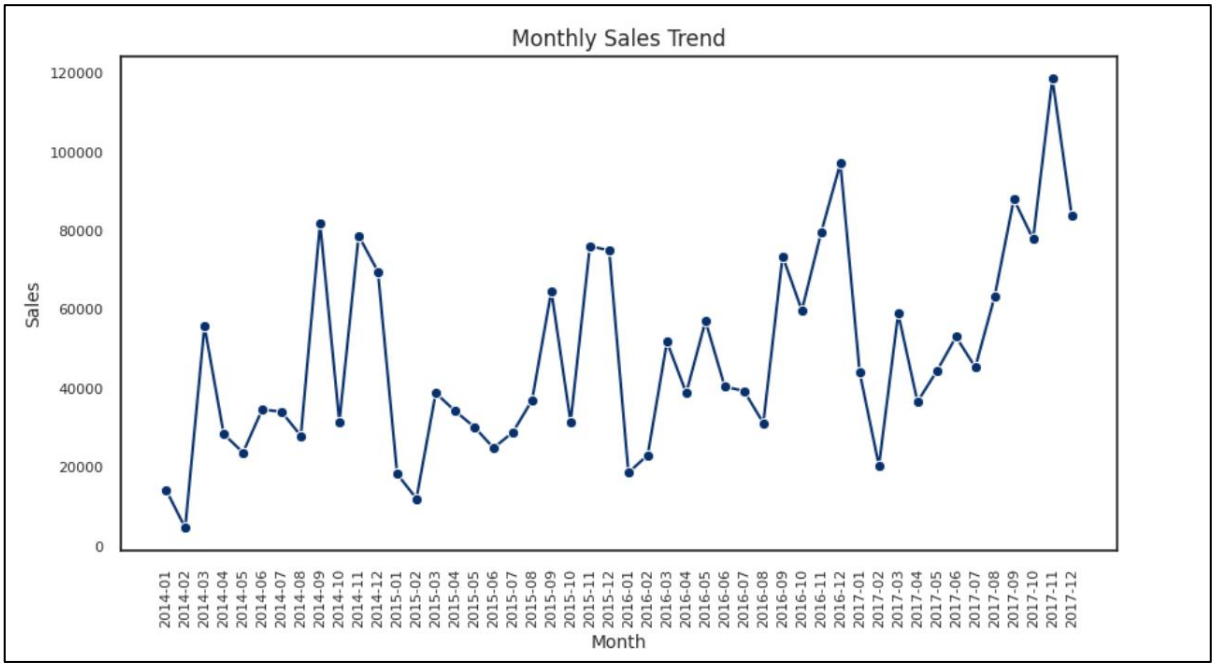
4. Exploratory Data Analysis (EDA)

4.1 Monthly Sales Trend

```
# 1. Monthly Sales Trend
# Create Year-Month column
df['Year_Month'] = df['Order Date'].dt.to_period('M').astype(str)

# Group by Month
monthly_sales = df.groupby('Year_Month')['Sales'].sum().reset_index()

# Plot
sns.set_style('white') # Set style to white to remove gray background
plt.figure(figsize=(10,5))
sns.lineplot(data=monthly_sales, x='Year_Month', y='Sales', marker='o', color='#08306B')
plt.title("Monthly Sales Trend", fontsize=12)
plt.xlabel("Month", fontsize= 10)
plt.xticks(rotation=90, fontsize=8)
plt.ylabel("Sales",fontsize = 10)
plt.yticks(fontsize=8)
plt.gca().grid(False)
plt.show()
```



(Figure 1)

Objective:

Identify monthly sales performance trends and observe how sales fluctuate over time.

Observation:

The line chart shows several fluctuations throughout the months. Some months show high sales, while others show noticeable dips. Toward the end of the year, especially in November and December, sales generally increase.

Insight:

Seasonal demand and year-end shopping behaviour contribute to higher sales at the end of the year. This suggests that planning inventory and marketing campaigns for year-end peaks can improve business performance.

`

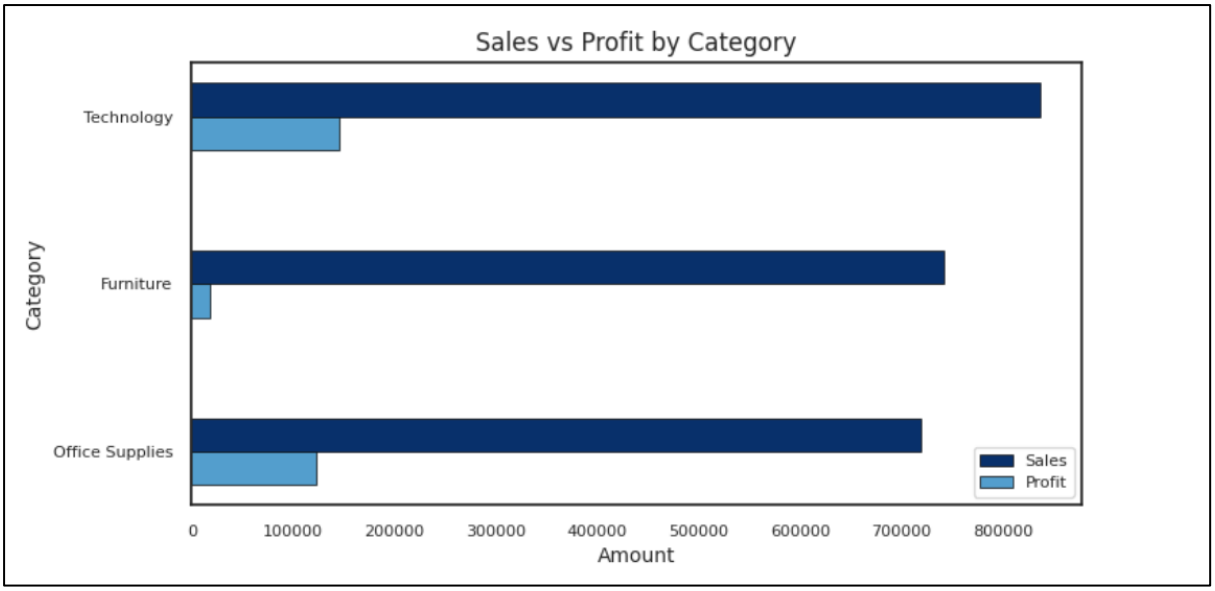
4.2 Sales vs Profit by Category

```
# 2. Sales vs Profit by Category
# Group data
cat_data = df.groupby('Category')[['Sales', 'Profit']].sum().reset_index()

# Sort by Sales in asc order
cat_data = cat_data.sort_values(by='Sales', ascending=True)

# Plot
plt.figure(figsize=(8,4)) # Adjust figure size for better readability of horizontal bars
y = np.arange(len(cat_data['Category']))   # y positions
height = 0.20                               # bar height

plt.barh(y + height/2, cat_data['Sales'], height, label='Sales', color='#08306B',edgecolor='black', linewidth=0.5)
plt.barh(y - height/2, cat_data['Profit'], height, label='Profit', color='#539ECD',edgecolor='black', linewidth=0.5)
plt.title("Sales vs Profit by Category", fontsize = 12)
plt.xlabel("Amount",fontsize = 10)
plt.xticks(fontsize = 8)
plt.ylabel("Category", fontsize = 10)
plt.yticks(y, cat_data['Category'], fontsize = 8)
legend_plt_1 = plt.legend(fontsize=8)
plt.show()
```



(Figure 2)

Objective:
Compare total sales and profit across the three main product categories.

Observation:
Technology shows strong sales and high profit. Office Supplies displays moderate profit with decent sales volume. Furniture has high sales but very low profit compared to the rest.

Insight:
Technology is the most profitable category. Furniture may need pricing or discount policy adjustments because it generates revenue but fails to convert into profit.

`

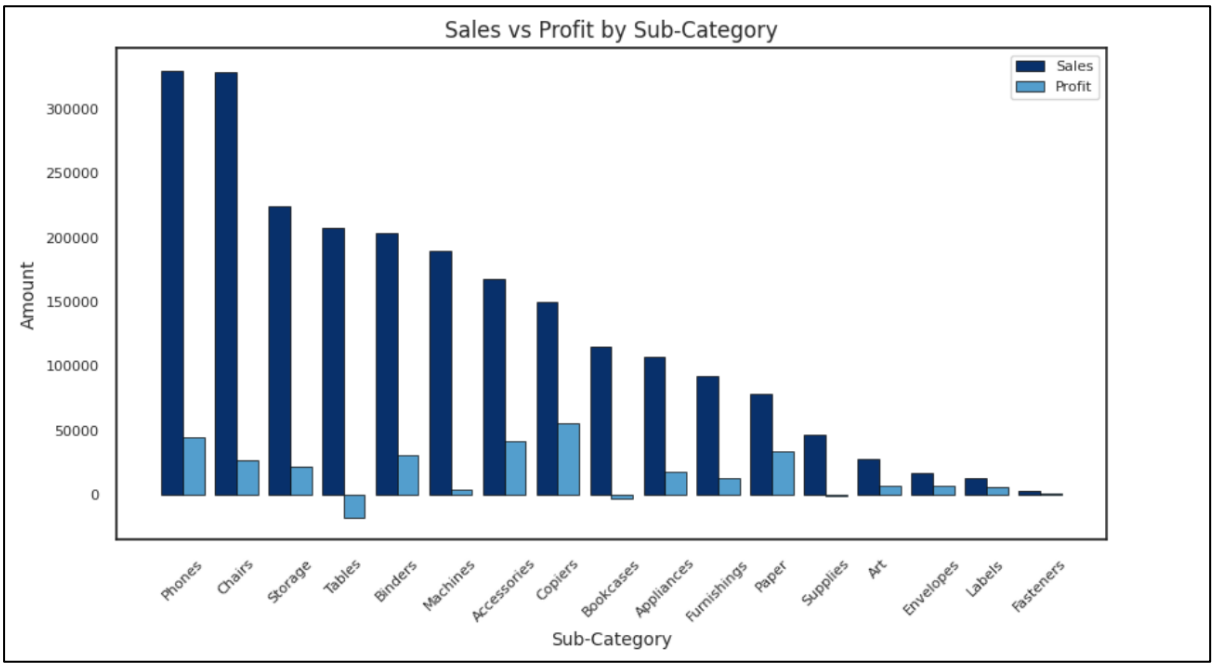
4.3 Sales vs Profit by Sub-Category

```
# 3. Sales vs Profit by Sub Category
# Group data
sub_data = df.groupby('Sub-Category')[['Sales', 'Profit']].sum().reset_index()

# Sort by Sales (looks cleaner)
sub_data = sub_data.sort_values('Sales', ascending=False)

# Positions
x = np.arange(len(sub_data['Sub-Category']))
width = 0.4  # width of bars

# Plot
plt.figure(figsize=(10,5))
plt.bar(x - width/2, sub_data['Sales'], width, label='Sales', color='#08306B',edgecolor='black', linewidth=0.5)
plt.bar(x + width/2, sub_data['Profit'], width, label='Profit', color='#539ECD',edgecolor='black', linewidth=0.5)
plt.title("Sales vs Profit by Sub-Category",fontsize = 12)
plt.xlabel("Sub-Category",fontsize = 10)
plt.xticks(x, sub_data['Sub-Category'], rotation=45,fontsize = 8)
plt.ylabel("Amount",fontsize = 10)
plt.yticks(fontsize = 8)
legend_plt_1 = plt.legend(fontsize=8)
plt.show()
```



(Figure 3)

Objective:

Analyse performance at a more detailed sub-category level to identify specific profitable or loss-making product lines.

Observation:

Phones, Chairs, and Storage have high sales. However, categories like Tables and Bookcases show negative profit despite considerable sales volume

Insight:

Loss-making sub-categories need immediate review. These products are dragging down overall profitability and may require price changes or reduced discounting.

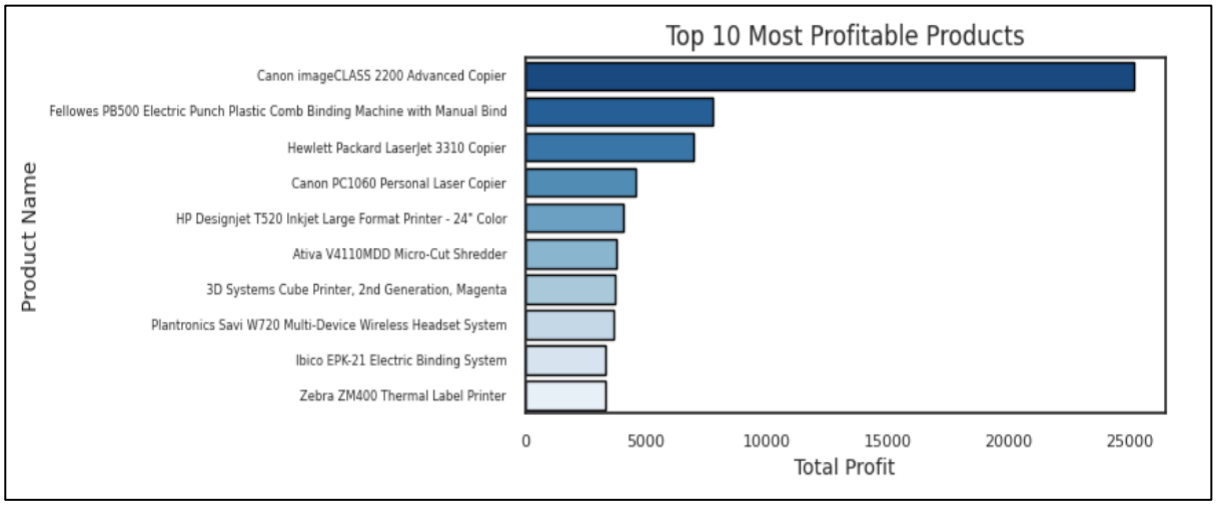
`

4.4 Top 10 Most Profitable Products

```
# 4. Top 10 Most profitable products and least products making loss
# Group by Product Name and sum Profit
product_profit = df.groupby('Product Name')['Profit'].sum().reset_index()

# Sort for most profitable products (top 10)
most_profitable_products = product_profit.sort_values(by='Profit', ascending=False).head(10)

# Plotting Most Profitable Products
plt.figure(figsize=(6, 3))
sns.barplot(x='Profit', y='Product Name', data=most_profitable_products, hue='Product Name', palette='Blues_r',
            legend=False,edgecolor='black', linewidth=1.0)
plt.title('Top 10 Most Profitable Products')
plt.xlabel('Total Profit', fontsize = 10)
plt.xticks(fontsize = 8)
plt.ylabel('Product Name', fontsize = 10)
plt.yticks(fontsize = 6)
plt.show()
```



(Figure 4)

Objective:

Identify which individual products contribute most to business profitability.

Observation:

Most top profitable products belong to technology or office-related items like copiers and shredders. These products show consistently high profit margins.

Insight:

These high-margin products should be prioritized in marketing and inventory planning. Promoting these items can directly increase company profitability.

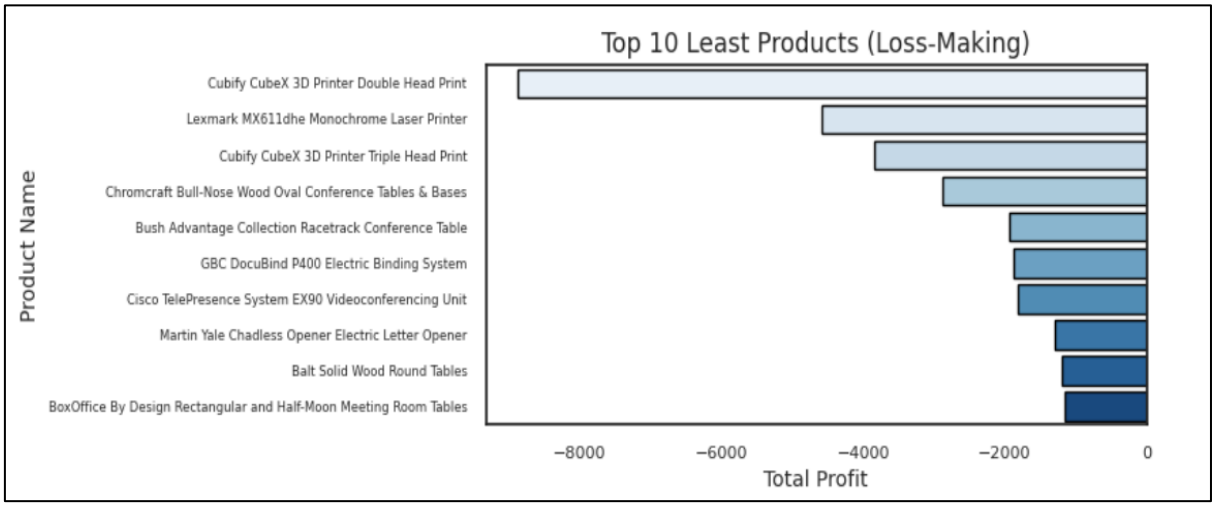
`

4.5 Top 10 Least Products Loss Making

```
# 5. Plotting Least Profitable Products
# Group by Product Name and sum Profit
product_profit = df.groupby('Product Name')['Profit'].sum().reset_index()

# Sort for least products loss Making (top 10)
least_profitable_products = product_profit.sort_values(by='Profit', ascending=True).head(10)

plt.figure(figsize=(6, 3))
sns.barplot(x='Profit', y='Product Name', data=least_profitable_products, hue='Product Name', palette='Blues',
            legend=False,edgecolor='black', linewidth=1.0)
plt.title('Top 10 Least Products (Loss-Making)', fontsize = 12)
plt.xlabel('Total Profit', fontsize = 10)
plt.xticks(fontsize = 8)
plt.ylabel('Product Name', fontsize = 10)
plt.yticks(fontsize = 6)
plt.show()
```



(Figure 5)

Objective:

Identify products that generate negative profit and need attention.

Observation:

Many furniture-related products (especially tables and bookcases) appear in the loss-making list. These show significantly negative profit values.

Insight:

These products either have high shipping costs, low margins, or excessive discounts. They are major contributors to loss and should be re-evaluated.

`

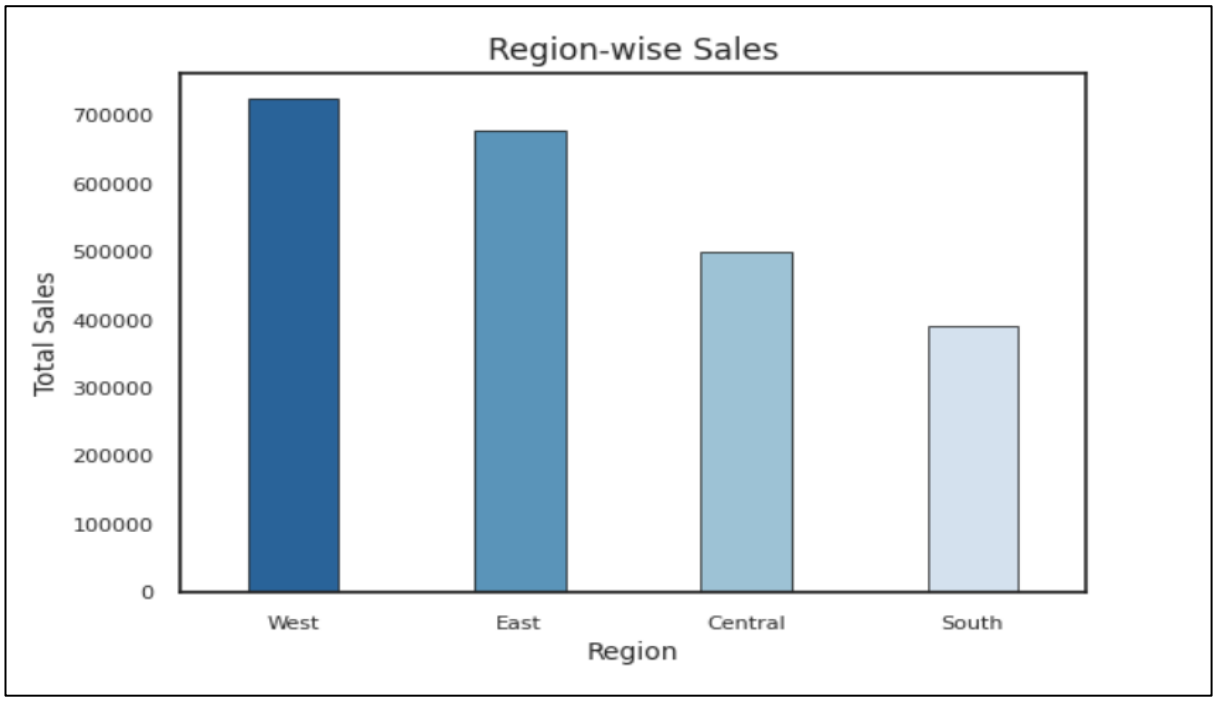
4.6 Region-wise Sales

```
# 6. Sales by Region

# Group by Region and sum Sales
region_sales = df.groupby('Region')['Sales'].sum().reset_index()

# Sort for better visualization (optional, but cleaner)
region_sales = region_sales.sort_values('Sales', ascending=False)

# Plot the bar chart
plt.figure(figsize=(6, 4))
sns.barplot(x='Region', y='Sales', data=region_sales, hue='Region', palette='Blues_r',
            legend=False,edgecolor='black',linewidth=0.5, width=0.4)
plt.title('Region-wise Sales', fontsize = 12)
plt.xlabel('Region',fontsize = 10)
plt.xticks(fontsize= 8)
plt.ylabel('Total Sales',fontsize = 10)
plt.yticks(fontsize= 8)
plt.show()
```



(Figure 6)

Objective:
Analyse total sales contributions of different regions.

Observation:
The West region shows the highest sales. The South region performs the lowest among the four regions.

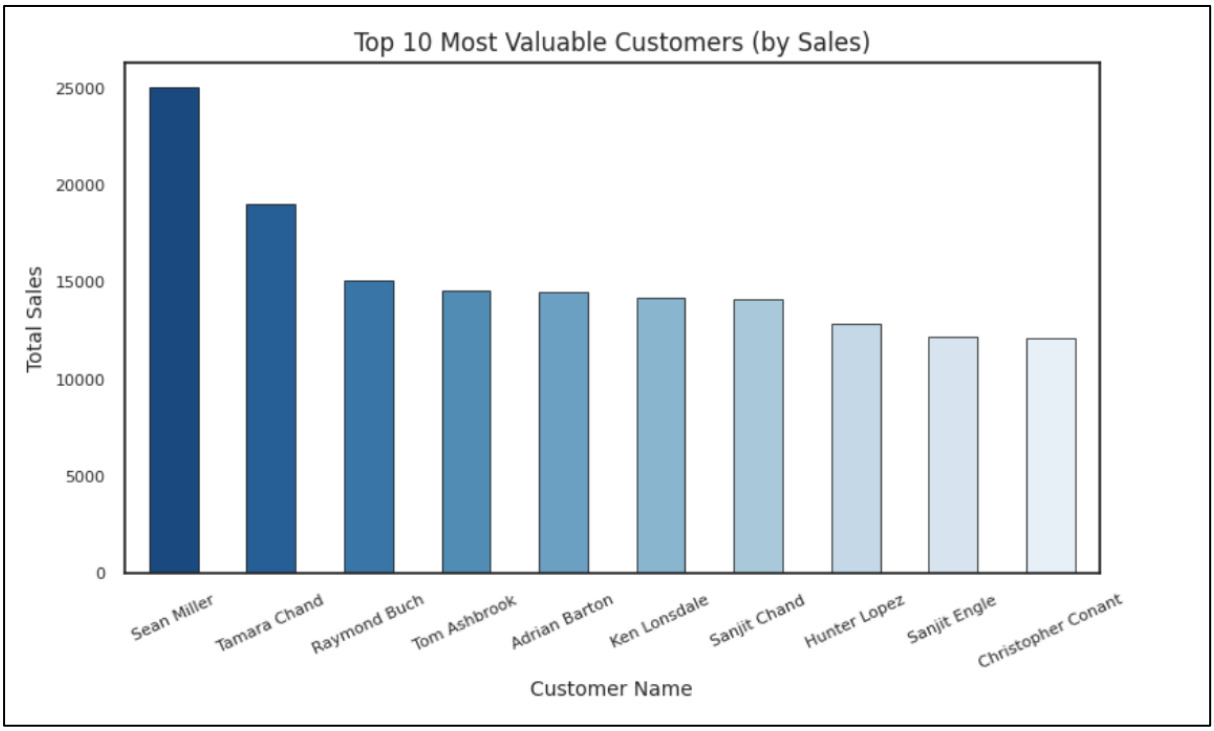
Insight:
The West region can be targeted for growth expansion. The South region may require marketing improvement or promotional strategies to improve performance.

`

4.7 Top 10 Most Valuable Customers

```
# 7. Most Valuable Customer by Sales
# Group by Customer and sort
top_customers = df.groupby('Customer Name')['Sales'].sum().sort_values(ascending=False).head(10)

# Plot
plt.figure(figsize=(8,5))
sns.barplot(y=top_customers.values, x=top_customers.index, hue=top_customers.index,
            palette ="Blues_r",edgecolor='black', linewidth=0.5, width=0.5)
plt.title("Top 10 Most Valuable Customers (by Sales)", fontsize= 12)
plt.xlabel("Customer Name", fontsize = 10)
plt.xticks(rotation=25, fontsize = 8)
plt.ylabel("Total Sales", fontsize = 10)
plt.yticks(fontsize = 8)
plt.tight_layout()
plt.show()
```



(Figure 7)

Objective:
Identify customers who generate the highest sales.

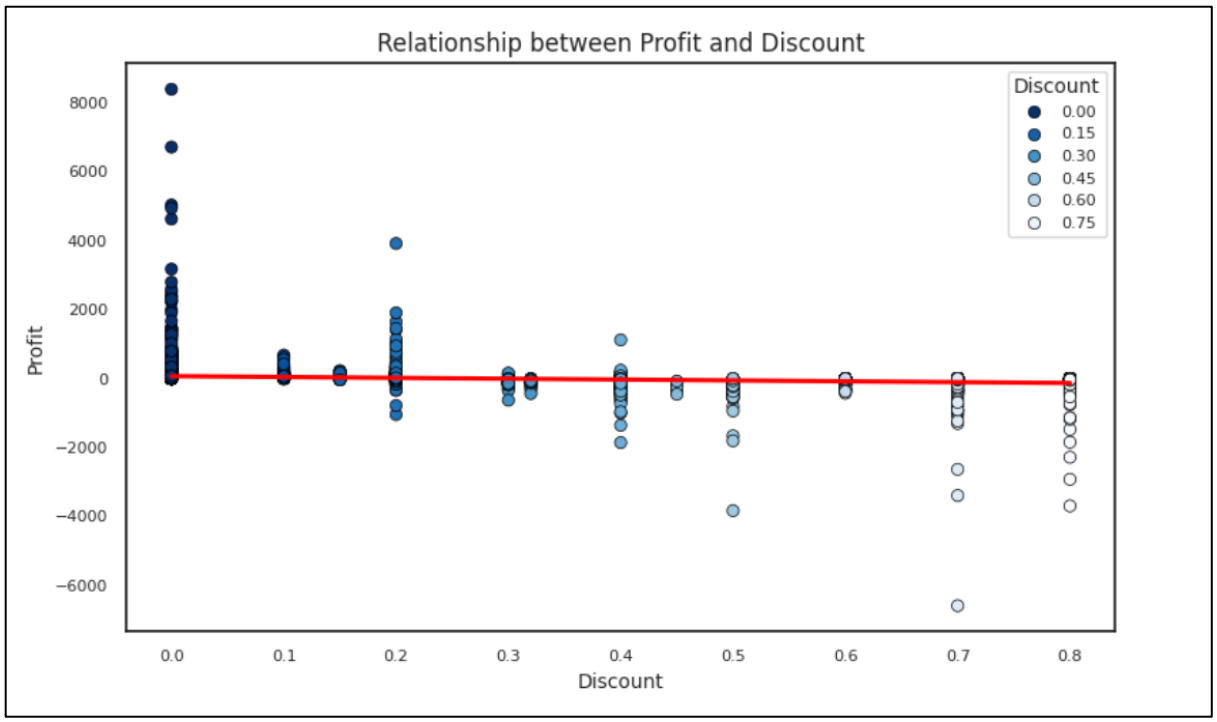
Observation:
A small group of customers contribute a significantly higher amount of sales compared to the rest.

Insight:
These customers are highly valuable. Retention strategies like loyalty benefits or exclusive offers should focus on these individuals.

`

4.8 Relationship Between Profit and Discount

```
# 8. Relationship between Profit and Discount
plt.figure(figsize=(8, 5))
sns.scatterplot(x='Discount', y='Profit', data=df, hue='Discount', palette ='Blues_r',edgecolor='black', linewidth=0.5)
sns.regplot(x=df['Discount'], y=df['Profit'], scatter=False, color = 'red')
plt.title("Relationship between Profit and Discount", fontsize = 12)
plt.xlabel('Discount', fontsize = 10)
plt.xticks(fontsize = 8)
plt.ylabel('Profit', fontsize = 10)
plt.yticks(fontsize = 8)
legend_plt_1 = plt.legend(fontsize=8)
legend_plt_1.set_title('Discount', prop={'size': 10})
plt.tight_layout()
plt.show()
```



(Figure 8)

Objective:
Understand how increasing discount affects profit.

Observation:
As discount increases, profit decreases sharply. Higher discounts are clearly linked to negative profit values.

Insight:
The company's discount strategy is hurting profitability. Discounts above a certain threshold should be controlled or avoided.

`

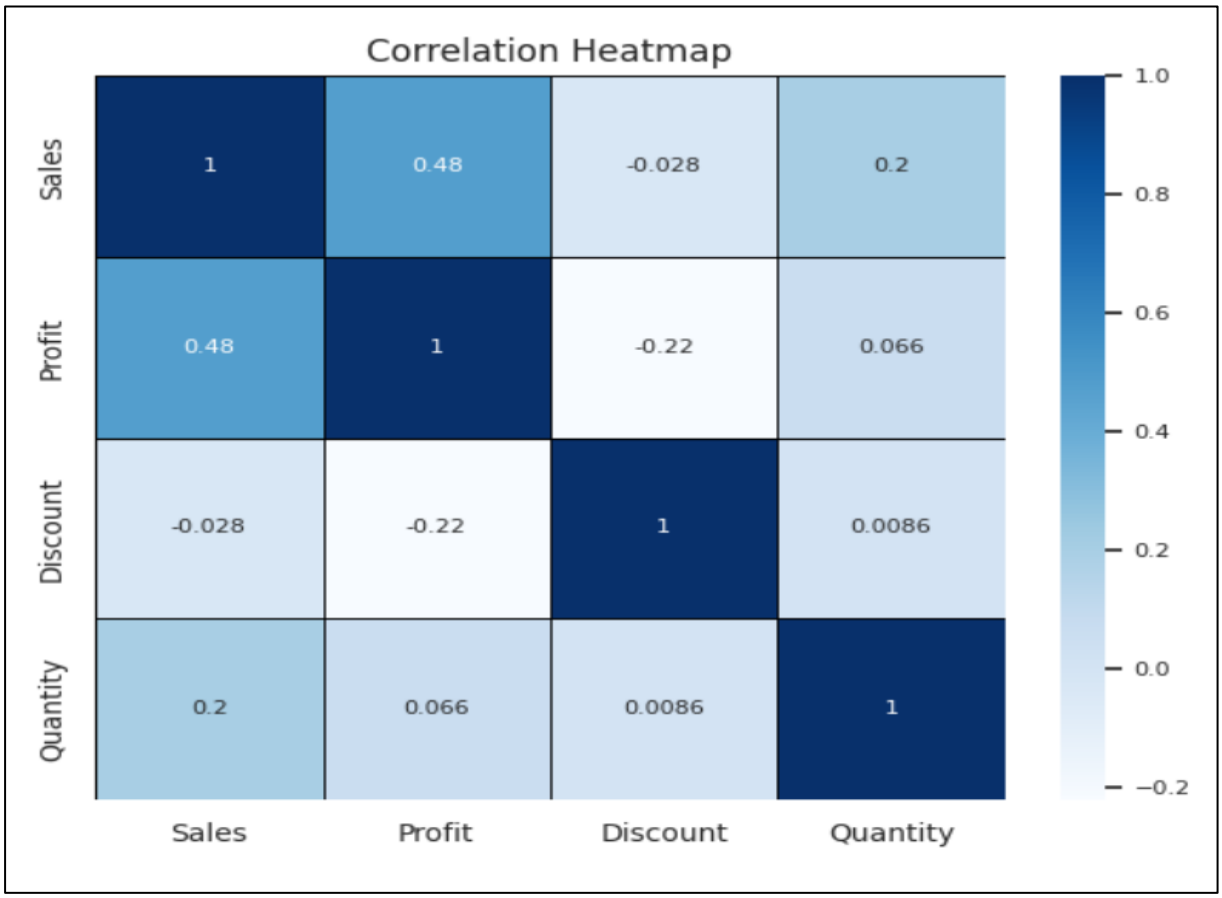
4.9 Correlation Heatmap

```
# 10. Correlation Heatmap
corr = df[['Sales','Profit','Discount','Quantity']].corr()

plt.figure(figsize=(6,5))
heatmap_plot = sns.heatmap(corr, annot=True, cmap='Blues', linewidths=0.5,linecolor='black', annot_kws={"size": 8})
plt.title("Correlation Heatmap",fontsize = 12)
plt.xticks(fontsize = 10)
plt.yticks(fontsize = 10)

# Get the colorbar object and set its tick label font size
cbar = heatmap_plot.collections[0].colorbar
cbar.ax.tick_params(labelsize=8)

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



(Figure 9)

Objective:
Check relationships between Sales, Profit, Discount, and Quantity.

Observation:
Discount shows a strong negative correlation with Profit. Sales and Profit have a weak correlation due to discount influence.

Insight:
The biggest factor affecting profit is discounting. The business should revise its discount strategy before focusing on increasing sales.

`

5. Key Findings

- Finding 1 — Technology is the key profit driver
  High margins + high demand make it the strongest segment.
- Finding 2 — Furniture category is dragging overall profitability
  High sales but extremely low margins → dangerous pattern.
- Finding 3 — Sub-categories like Tables & Bookcases are loss centre's
  Immediate corrective actions needed.
- Finding 4 — Seasonality heavily influences retail sales
  November/December peak must be leveraged.
- Finding 5 — West region dominates sales
  Strong geographic opportunity.
- Finding 6 — South region underperforms
  Needs marketing and strategy focus.
- Finding 7 — Revenue relies on a small group of customers
  Customer retention and personalization mandatory.
- Finding 8 — Discount policies are harming profitability
  The biggest insight in the entire project.

`

6. Recommendation

1. Modify discount strategy immediately
   - Reduce discounts above 20%
   - Remove discounting entirely for loss-making items
   - Implement intelligent discount rules

2. Revisit Furniture pricing
   - Review procurement and logistics cost
   - Increase price or remove low-margin furniture

3. Promote high-profit products

Focus on Copiers, Phones, Chairs, and Shredders.

4. Improve performance in the South region
   - Targeted marketing campaigns
   - Pricing adjustments
   - Regional promotions

5. Apply customer segmentation

Reward high-value customers with:
   - Loyalty points
   - Early access to offers
   - Priority delivery

6. Reduce dependence on low-profit items

Shift inventory more heavily towards profitable segments.

`

7. Conclusion

This report provided a comprehensive analysis of retail sales performance using Python. The results highlight the importance of discount management, product mix optimization, customer targeting, and regional sales strategy. The findings can guide business decisions and help the retail organization boost profit margins while maintaining healthy sales growth.

8. Future Work

Future enhancements may include:

- Predictive forecasting models (ARIMA, Prophet, LSTM)
- Customer segmentation using clustering
- Product recommendation systems
- Power BI dashboard for management
- RFM customer scoring
- Dynamic pricing optimization

`