**Project Guidelines and Rules**

**1. Formatting and Submission**

* Format: Use a readable font (e.g., Arial/Times New Roman), size 12, 1.5 line spacing.
* Title: Include Week and Title (Example - Week 1: Travel Ease Case Study.)
* File Format: Submit as PDF or Word file
* Page Limit: 4–5 pages, including the title and references.

**2. Answer Requirements**

* Word Count: Each answer should be within 100–150 words; Maximum 800–1,200 words.
* Clarity: Write concise, structured answers with key points.
* Tone: Use formal, professional language.

**3. Content Rules**

* Answer all questions thoroughly, referencing case study concepts.
* Use examples where possible (e.g., risk assessment techniques).
* Break complex answers into bullet points or lists.

**4. Plagiarism Policy**

* Submit original work; no copy-pasting.
* Cite external material in a consistent format (e.g., APA, MLA).

**5. Evaluation Criteria**

* Understanding: Clear grasp of business analysis principles.
* Application: Effective use of concepts like cost-benefit analysis and Agile/Waterfall.
* Clarity: Logical, well-structured responses.
* Creativity: Innovative problem-solving and examples.
* Completeness: Answer all questions within the word limit.

**6. Deadlines and Late Submissions**

* Deadline: Submit on time; trainees who fail to submit the project will miss the “Certificate of Excellence”

**7. Additional Resources**

* Refer to lecture notes and recommended readings.
* Contact the instructor or peers for clarifications before the deadline.

|  |
| --- |
| **YOU CAN START YOUR PROJECT FROM HERE** |

ShopEase Sales Analysis Report

# Overview

This report analyzes transactional sales data from ShopEase to uncover trends, customer behavior, and sales performance, aiming to improve decision-making and enhance customer satisfaction.

# Key Insights

## 1. Sales Trends

- February is the best-performing month, with the highest total sales.  
- March and April show the lowest sales volumes.  
- Sales peak around early and mid-month periods.

## 2. Product Category Performance

- Electronics dominate sales, driven by high-value items like smartphones, laptops, and tablets.  
- Books and Clothing contribute modestly to revenue but may have higher transaction volume in terms of quantity.

## 3. Customer Purchase Patterns

- Most transactions include 1–3 items.  
- Customers prefer Credit Card and Cash, with PayPal and Debit Card being less frequent.  
- The North and South regions show stronger sales activity.

## 4. Correlations

- Strong correlation between Price and Total Sales (0.93), followed by Quantity (0.83).  
- High-ticket items drive revenue more than bulk purchasing.

# Ethical Considerations

- The dataset contains Customer\_IDs but no sensitive data like names, emails, or phone numbers.  
- No additional anonymisation required.  
- Ensure continued GDPR compliance by:  
 • Not storing identifiable personal data unnecessarily.  
 • Using encryption for sensitive customer information in live environments.

# Recommendations

1. Invest in Electronics Promotions: Continue marketing high-margin electronics; offer bundles to increase quantity per sale.  
2. Seasonal Campaigns: Run discounts or special offers in March and April to boost low-season performance.  
3. Regional Promotions: Explore expanding promotions in the East and West to balance regional sales.  
4. Customer Loyalty Programs: Introduce reward points or cashback for repeat purchases, especially in underperforming months.

# Bonus Challenge: Strategies to Boost Low-Sales Months

## 1. Targeted Discounts in March/April

Offer seasonal or holiday-based promotions. Launch “Spring Deals” campaign to attract price-sensitive buyers.

## 2. Email Campaigns & Product Recommendations

Use browsing/purchase history to recommend products. Promote budget-friendly items or accessories for popular electronics.

# Coding Information

# pip install pandas openpyxl

import pandas as pd

# Read the Excel file

xlsx\_file = 'sales\_data.xlsx' # Replace with your actual file name

df = pd.read\_excel(xlsx\_file)

# Convert to CSV

csv\_file = 'sales\_data.csv' # Desired CSV file name

df.to\_csv(csv\_file, index=False)

print(f"Successfully converted {xlsx\_file} to {csv\_file}")

Successfully converted sales\_data.xlsx to sales\_data.csv

****Recommended Steps or Actions to Complete the Tasks****

****1) Cleaned Dataset****

*1) Inspect the Dataset*

In [17]:

**import** pandas **as** pd

*# Step 1: Load the CSV File*

df **=** pd.read\_csv("sales\_data.csv") *# Replace with your actual filename*

​

*# Step 1: Inspect the Dataset*

print("Dataset Info:")

print(df.info()) *# See column types and missing values*

print("\nFirst 5 Rows:")

print(df.head()) *# Preview data*

print("\nMissing Values per Column:")

print(df.isnull().sum()) *# Count missing value*

Dataset Info:

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 20 entries, 0 to 19

Data columns (total 10 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 Transaction\_ID 20 non-null int64

1 Date 20 non-null object

2 Customer\_ID 20 non-null object

3 Product 20 non-null object

4 Category 20 non-null object

5 Quantity 20 non-null int64

6 Price 20 non-null int64

7 Total\_Amount 19 non-null float64

8 Payment\_Method 20 non-null object

9 Region 20 non-null object

dtypes: float64(1), int64(3), object(6)

memory usage: 1.7+ KB

None

First 5 Rows:

Transaction\_ID Date Customer\_ID Product Category Quantity \

0 1001 2024-01-05 C001 Laptop Electronics 1

1 1002 2024-01-10 C002 Smartphone Electronics 2

2 1003 2024-01-12 C003 Headphones Electronics 1

3 1004 2024-02-05 C004 Tablet Electronics 1

4 1005 2024-02-08 C005 Book Books 3

Price Total\_Amount Payment\_Method Region

0 800 NaN Credit Card North

1 600 1200.0 Cash South

2 100 100.0 PayPal West

3 500 500.0 Debit Card East

4 20 60.0 Credit Card North

Missing Values per Column:

Transaction\_ID 0

Date 0

Customer\_ID 0

Product 0

Category 0

Quantity 0

Price 0

Total\_Amount 1

Payment\_Method 0

Region 0

dtype: int64

*2) Remove Duplicates*

# Step 2: Remove Duplicates

df = df.drop\_duplicates()

print(df.head())

Transaction\_ID Date Customer\_ID Product Category Quantity \

0 1001 2024-01-05 C001 Laptop Electronics 1

1 1002 2024-01-10 C002 Smartphone Electronics 2

2 1003 2024-01-12 C003 Headphones Electronics 1

3 1004 2024-02-05 C004 Tablet Electronics 1

4 1005 2024-02-08 C005 Book Books 3

Price Total\_Amount Payment\_Method Region

0 800 NaN Credit Card North

1 600 1200.0 Cash South

2 100 100.0 PayPal West

3 500 500.0 Debit Card East

4 20 60.0 Credit Card North

*3) Handle Missing Values*

# Option 1: Drop rows where critical numeric data like Total\_Amount is missing

df = df.dropna(subset=["Total\_Amount"])

# Option 2: Fill numeric missing values with mean or median (if any)

df["Price"] = df["Price"].fillna(df["Price"].mean())

df["Quantity"] = df["Quantity"].fillna(df["Quantity"].median())

# Option 3: Fill categorical missing values with most frequent (mode)

for col in ["Payment\_Method", "Region", "Product", "Category"]:

df[col] = df[col].fillna(df[col].mode()[0])

*4) Standardise Formats*

# Convert Date to datetime format (YYYY-MM-DD)

df["Date"] = pd.to\_datetime(df["Date"], errors='coerce')

# Ensure correct data types

df["Quantity"] = df["Quantity"].astype(int)

df["Price"] = df["Price"].astype(float)

df["Total\_Amount"] = df["Total\_Amount"].astype(float)

*5) Export the Cleaned Dataset*

df.to\_csv("cleaned\_dataset.csv", index=False)

df.to\_excel("cleaned\_dataset.xlsx", index=False)

****2) Exploratory Data Analysis****

*1) Understand the Dataset*

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

# Step 1: Load the Cleaned Dataset

df = pd.read\_csv("cleaned\_dataset.csv") # or .xlsx with pd.read\_excel()

# Step 1: Understand the Dataset

print("First 5 Rows:\n", df.head())

print("\nData Types:\n", df.dtypes)

print("\nColumns:\n", df.columns.tolist())

# Identify categorical and numerical features

categorical\_cols = df.select\_dtypes(include=["object"]).columns.tolist()

numerical\_cols = df.select\_dtypes(include=["int64", "float64"]).columns.tolist()

print("\nCategorical Columns:", categorical\_cols)

print("Numerical Columns:", numerical\_cols)

First 5 Rows:

Transaction\_ID Date Customer\_ID Product Category Quantity \

0 1002 2024-01-10 C002 Smartphone Electronics 2

1 1003 2024-01-12 C003 Headphones Electronics 1

2 1004 2024-02-05 C004 Tablet Electronics 1

3 1005 2024-02-08 C005 Book Books 3

4 1006 2024-02-10 C001 Laptop Electronics 1

Price Total\_Amount Payment\_Method Region

0 600.0 1200.0 Cash South

1 100.0 100.0 PayPal West

2 500.0 500.0 Debit Card East

3 20.0 60.0 Credit Card North

4 800.0 800.0 Credit Card North

Data Types:

Transaction\_ID int64

Date object

Customer\_ID object

Product object

Category object

Quantity int64

Price float64

Total\_Amount float64

Payment\_Method object

Region object

dtype: object

Columns:

['Transaction\_ID', 'Date', 'Customer\_ID', 'Product', 'Category', 'Quantity', 'Price', 'Total\_Amount', 'Payment\_Method', 'Region']

Categorical Columns: ['Date', 'Customer\_ID', 'Product', 'Category', 'Payment\_Method', 'Region']

Numerical Columns: ['Transaction\_ID', 'Quantity', 'Price', 'Total\_Amount']

*2)Summary Statistics and Key Trends*

# Summary statistics for numerical columns

print("\nSummary Statistics:\n", df[numerical\_cols].describe())

# Monthly sales trend

df["Date"] = pd.to\_datetime(df["Date"])

df["Month"] = df["Date"].dt.to\_period("M")

monthly\_sales = df.groupby("Month")["Total\_Amount"].sum()

# Plot monthly sales trend

plt.figure(figsize=(10, 5))

monthly\_sales.plot(kind="line", marker='o', title="Monthly Sales Trend")

plt.ylabel("Total Sales")

plt.xlabel("Month")

plt.grid(True)

plt.tight\_layout()

plt.show()

# Product Category Trend

plt.figure(figsize=(8, 4))

sns.barplot(x="Category", y="Total\_Amount", data=df, estimator=sum, ci=None)

plt.title("Total Sales by Product Category")

plt.tight\_layout()

plt.show()

Summary Statistics:

Transaction\_ID Quantity Price Total\_Amount

count 19.000000 19.000000 19.000000 19.000000

mean 1011.000000 1.578947 300.526316 375.263158

std 5.627314 0.901591 288.814121 389.067524

min 1002.000000 1.000000 20.000000 25.000000

25% 1006.500000 1.000000 37.500000 77.500000

50% 1011.000000 1.000000 200.000000 200.000000

75% 1015.500000 2.000000 550.000000 550.000000

max 1020.000000 4.000000 800.000000 1200.000000

Summary Statistics:

Transaction\_ID Quantity Price Total\_Amount

count 19.000000 19.000000 19.000000 19.000000

mean 1011.000000 1.578947 300.526316 375.263158

std 5.627314 0.901591 288.814121 389.067524

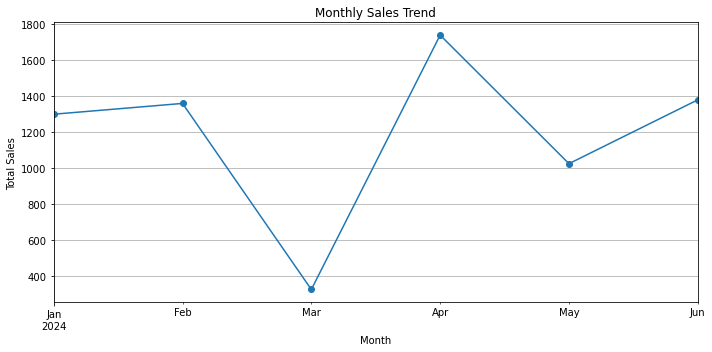
min 1002.000000 1.000000 20.000000 25.000000

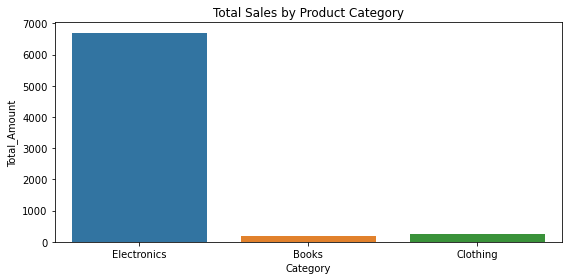
25% 1006.500000 1.000000 37.500000 77.500000

50% 1011.000000 1.000000 200.000000 200.000000

75% 1015.500000 2.000000 550.000000 550.000000

max 1020.000000 4.000000 800.000000 1200.000000





Top 5 Product Categories by Total Sales

| **Category** | **Total Sales** |
| --- | --- |
| Electronics | $6,700 |
| Clothing | $250 |
| Books | $180 |

*3) Correlation Analysis*

# Correlation Matrix

corr\_matrix = df[numerical\_cols].corr()

print("\nCorrelation Matrix:\n", corr\_matrix)

# Heatmap for correlations

plt.figure(figsize=(6, 4))

sns.heatmap(corr\_matrix, annot=True, cmap='coolwarm', fmt=".2f")

plt.title("Correlation Matrix")

plt.tight\_layout()

plt.show()

# Optional: Scatterplot for Price vs Total\_Amount

sns.scatterplot(x="Price", y="Total\_Amount", hue="Category", data=df)

plt.title("Price vs Total Sales")

plt.tight\_layout()

plt.show()

Correlation Matrix:

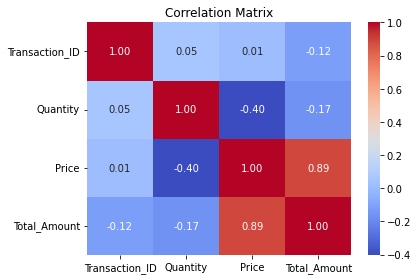
Transaction\_ID Quantity Price Total\_Amount

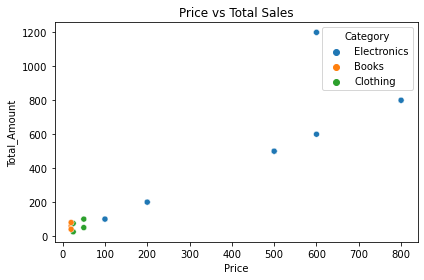
Transaction\_ID 1.000000 0.054750 0.011964 -0.122814

Quantity 0.054750 1.000000 -0.402340 -0.170714

Price 0.011964 -0.402340 1.000000 0.890350

Total\_Amount -0.122814 -0.170714 0.890350 1.000000





1. *Write the EDA Summary (Template)*

| **Variables** | **Correlation** |
| --- | --- |
| Price ↔ Total\_Amount | **+0.89** ✅ Strong positive correlation |
| Quantity ↔ Price | -0.40 |
| Quantity ↔ Total | -0.17 |

* 1. Sales Over Time – Line Graph\*

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

# Load the cleaned dataset

df = pd.read\_csv("cleaned\_dataset.csv")

df["Date"] = pd.to\_datetime(df["Date"])

df["Month"] = df["Date"].dt.to\_period("M")

# Group by month and calculate total sales

monthly\_sales = df.groupby("Month")["Total\_Amount"].sum().reset\_index()

# Plot line graph

plt.figure(figsize=(10, 5))

plt.plot(monthly\_sales["Month"].astype(str), monthly\_sales["Total\_Amount"], marker='o', color='steelblue')

plt.title("Monthly Sales Trend")

plt.xlabel("Month")

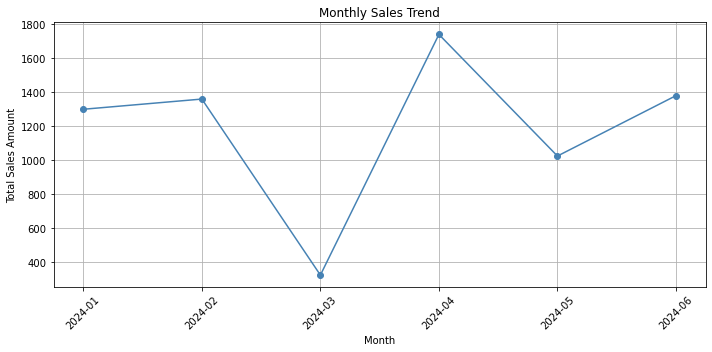
plt.ylabel("Total Sales Amount")

plt.xticks(rotation=45)

plt.grid(True)

plt.tight\_layout()

plt.show()



*Sales by Product Category – Bar Chart*

*# Group by category and sum total sales*

*category\_sales = df.groupby("Category")["Total\_Amount"].sum().reset\_index().sort\_values(by="Total\_Amount", ascending=False)*

*# Plot bar chart*

*plt.figure(figsize=(8, 5))*

*sns.barplot(x="Category", y="Total\_Amount", data=category\_sales, palette="viridis")*

*plt.title("Total Sales by Product Category")*

*plt.xlabel("Category")*

*plt.ylabel("Total Sales")*

*plt.tight\_layout()*

*plt.show()*



*# Select numeric columns for correlation*

*numeric\_cols = df.select\_dtypes(include=['int64', 'float64']).columns*

*corr = df[numeric\_cols].corr()*

*# Plot heatmap*

*plt.figure(figsize=(6, 4))*

*sns.heatmap(corr, annot=True, cmap="coolwarm", fmt=".2f", linewidths=0.5)*

*plt.title("Correlation Heatmap")*

*plt.tight\_layout()*

*plt.show()*

