Superstore Sales Analysis

Business Intelligence Dashboard Project

This project will analyze customer, product, and regional sales data from a fictional Superstore. The goal is to uncover business insights that will drive revenue and identify inefficiencies such as loss making products, low performing segments, or poor shipping methods.

Tools Used: Python (Pandas, Matplotlib, Seaborn), Jupyter Notebook

In [257…
```python
# Checking to see if I have pandas installed
import pandas as pd
print(pd.__version__)

import matplotlib.pyplot as plt
import seaborn as sns

# Optional: make your plots show up in Jupyter cells
# %matplotlib inline

# # Optional: set a consistent visual style
# sns.set(style="whitegrid")
```

```
2.2.3
```

1. Data Overview

In [258…
```python
# I used that encoding since it is required for the program to run. Proba
df = pd.read_csv("Sample - Superstore.csv", encoding='ISO-8859-1')
df
```

Out[258…

	Row ID	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name	Se
0	1	CA-2016-152156	11/8/2016	11/11/2016	Second Class	CG-12520	Claire Gute	Co
1	2	CA-2016-152156	11/8/2016	11/11/2016	Second Class	CG-12520	Claire Gute	Co
2	3	CA-2016-138688	6/12/2016	6/16/2016	Second Class	DV-13045	Darrin Van Huff	Co
3	4	US-2015-108966	10/11/2015	10/18/2015	Standard Class	SO-20335	Sean O'Donnell	Co
4	5	US-2015-108966	10/11/2015	10/18/2015	Standard Class	SO-20335	Sean O'Donnell	Co
…	…	…	…	…	…	…	…	
9989	9990	CA-2014-110422	1/21/2014	1/23/2014	Second Class	TB-21400	Tom Boeckenhauer	Co
9990	9991	CA-2017-121258	2/26/2017	3/3/2017	Standard Class	DB-13060	Dave Brooks	Co
9991	9992	CA-2017-121258	2/26/2017	3/3/2017	Standard Class	DB-13060	Dave Brooks	Co
9992	9993	CA-2017-121258	2/26/2017	3/3/2017	Standard Class	DB-13060	Dave Brooks	Co
9993	9994	CA-2017-119914	5/4/2017	5/9/2017	Second Class	CC-12220	Chris Cortes	Co

9994 rows × 21 columns

In [259…

```
# How many rows and columns does the dataset have?
```

```
df.shape
```

Out[259…
```
(9994, 21)
```

In [260…
```
# Shows the columns in the dataset.
df.columns
```

Out[260…
```
Index(['Row ID', 'Order ID', 'Order Date', 'Ship Date', 'Ship Mode',
       'Customer ID', 'Customer Name', 'Segment', 'Country', 'City', 'St
ate',
       'Postal Code', 'Region', 'Product ID', 'Category', 'Sub-Categor
y',
       'Product Name', 'Sales', 'Quantity', 'Discount', 'Profit'],
      dtype='object')
```

In [261…
```
# More information on the dataset.
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9994 entries, 0 to 9993
Data columns (total 21 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Row ID         9994 non-null   int64  
 1   Order ID       9994 non-null   object 
 2   Order Date     9994 non-null   object 
 3   Ship Date      9994 non-null   object 
 4   Ship Mode      9994 non-null   object 
 5   Customer ID    9994 non-null   object 
 6   Customer Name  9994 non-null   object 
 7   Segment        9994 non-null   object 
 8   Country        9994 non-null   object 
 9   City           9994 non-null   object 
 10  State          9994 non-null   object 
 11  Postal Code    9994 non-null   int64  
 12  Region         9994 non-null   object 
 13  Product ID     9994 non-null   object 
 14  Category       9994 non-null   object 
 15  Sub-Category   9994 non-null   object 
 16  Product Name   9994 non-null   object 
 17  Sales          9994 non-null   float64
 18  Quantity       9994 non-null   int64  
 19  Discount       9994 non-null   float64
 20  Profit         9994 non-null   float64
dtypes: float64(3), int64(3), object(15)
memory usage: 1.6+ MB
```

In [262…
```
# More calculative description on the dataset.
df.describe()
```

Out[262…

	Row ID	Postal Code	Sales	Quantity	Discount	
count	9994.000000	9994.000000	9994.000000	9994.000000	9994.000000	99
mean	4997.500000	55190.379428	229.858001	3.789574	0.156203	
std	2885.163629	32063.693350	623.245101	2.225110	0.206452	
min	1.000000	1040.000000	0.444000	1.000000	0.000000	-65
25%	2499.250000	23223.000000	17.280000	2.000000	0.000000	
50%	4997.500000	56430.500000	54.490000	3.000000	0.200000	
75%	7495.750000	90008.000000	209.940000	5.000000	0.200000	
max	9994.000000	99301.000000	22638.480000	14.000000	0.800000	83

In [263…

```
# Checking for null values
df.isnull()
```

Out[263…

	Row ID	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name	Segment	Country
0	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False
…	…	…	…	…	…	…	…	…	…
9989	False	False	False	False	False	False	False	False	False
9990	False	False	False	False	False	False	False	False	False
9991	False	False	False	False	False	False	False	False	False
9992	False	False	False	False	False	False	False	False	False
9993	False	False	False	False	False	False	False	False	False

9994 rows × 21 columns

In [264…

```
#
df.isnull().sum()
```

Out[264…
```
Row ID           0
Order ID         0
Order Date       0
Ship Date        0
Ship Mode        0
Customer ID      0
Customer Name    0
Segment          0
Country          0
City             0
State            0
Postal Code      0
Region           0
Product ID       0
Category         0
Sub-Category     0
Product Name     0
Sales            0
Quantity         0
Discount         0
Profit           0
dtype: int64
```

In [265…
```
# Checking for duplicates

df.duplicated()
```

Out[265…
```
0       False
1       False
2       False
3       False
4       False
        ...
9989    False
9990    False
9991    False
9992    False
9993    False
Length: 9994, dtype: bool
```

In [266…
```
# Checking for sum of duplicates

df.duplicated().sum()
```

Out[266…
```
np.int64(0)
```

In [267…
```
## 2. Data Cleaning
```

In [268…
```
# Converting these columns to date, time format. This helps with analyzin

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

In [269…
```
# More information on the dataset to confirm the last code worked.
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9994 entries, 0 to 9993
Data columns (total 21 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Row ID         9994 non-null   int64
 1   Order ID       9994 non-null   object
 2   Order Date     9994 non-null   datetime64[ns]
 3   Ship Date      9994 non-null   datetime64[ns]
 4   Ship Mode      9994 non-null   object
 5   Customer ID    9994 non-null   object
 6   Customer Name  9994 non-null   object
 7   Segment        9994 non-null   object
 8   Country        9994 non-null   object
 9   City           9994 non-null   object
 10  State          9994 non-null   object
 11  Postal Code    9994 non-null   int64
 12  Region         9994 non-null   object
 13  Product ID     9994 non-null   object
 14  Category       9994 non-null   object
 15  Sub-Category   9994 non-null   object
 16  Product Name   9994 non-null   object
 17  Sales          9994 non-null   float64
 18  Quantity       9994 non-null   int64
 19  Discount       9994 non-null   float64
 20  Profit         9994 non-null   float64
dtypes: datetime64[ns](2), float64(3), int64(3), object(13)
memory usage: 1.6+ MB
```

In [270…
```
# Checking the column names to see if renaming is needed.

df.columns
```

Out[270…
```
Index(['Row ID', 'Order ID', 'Order Date', 'Ship Date', 'Ship Mode',
       'Customer ID', 'Customer Name', 'Segment', 'Country', 'City', 'St
ate',
       'Postal Code', 'Region', 'Product ID', 'Category', 'Sub-Categor
y',
       'Product Name', 'Sales', 'Quantity', 'Discount', 'Profit'],
      dtype='object')
```

In [271…
```
# Changing the column names for a cleaner code.

df.columns = df.columns.str.replace(" ", "_").str.replace("-", "_").str.l
```

In [272…
```
# More information on the dataset to confirm the last code worked.

df.columns
```

Out[272…
```
Index(['row_id', 'order_id', 'order_date', 'ship_date', 'ship_mode',
       'customer_id', 'customer_name', 'segment', 'country', 'city', 'st
ate',
       'postal_code', 'region', 'product_id', 'category', 'sub_categor
y',
       'product_name', 'sales', 'quantity', 'discount', 'profit'],
      dtype='object')
```

In [273…
```
# Setting order_id as the index

df.set_index("order_id", inplace=True)
```

```
# if "order_id" in df.columns:
#     df.set_index("order_id", inplace=True)
# else:
#     print("Column 'order_id' not found in this dataset.")
```

In [274…

```
# Check heads

df.head()
```

Out[274…

order_id	row_id	order_date	ship_date	ship_mode	customer_id	customer_name
CA-2016-152156	1	2016-11-08	2016-11-11	Second Class	CG-12520	Claire Gute
CA-2016-152156	2	2016-11-08	2016-11-11	Second Class	CG-12520	Claire Gute
CA-2016-138688	3	2016-06-12	2016-06-16	Second Class	DV-13045	Darrin Van Huff
US-2015-108966	4	2015-10-11	2015-10-18	Standard Class	SO-20335	Sean O'Donnell
US-2015-108966	5	2015-10-11	2015-10-18	Standard Class	SO-20335	Sean O'Donnell

In [275…

```
# Check tails

df.tail()
```

Out[275…

order_id	row_id	order_date	ship_date	ship_mode	customer_id	customer_name
CA-2014-110422	9990	2014-01-21	2014-01-23	Second Class	TB-21400	Tom Boeckenhauer
CA-2017-121258	9991	2017-02-26	2017-03-03	Standard Class	DB-13060	Dave Brooks
CA-2017-121258	9992	2017-02-26	2017-03-03	Standard Class	DB-13060	Dave Brooks
CA-2017-121258	9993	2017-02-26	2017-03-03	Standard Class	DB-13060	Dave Brooks
CA-2017-119914	9994	2017-05-04	2017-05-09	Second Class	CC-12220	Chris Cortes

In [276…

```
# Rechecking the summary status

df.describe()
```

Out[276…

	row_id	order_date	ship_date	postal_code	
count	9994.000000	9994	9994	9994.000000	9994
mean	4997.500000	2016-04-30 00:07:12.259355648	2016-05-03 23:06:58.571142912	55190.379428	229
min	1.000000	2014-01-03 00:00:00	2014-01-07 00:00:00	1040.000000	0
25%	2499.250000	2015-05-23 00:00:00	2015-05-27 00:00:00	23223.000000	17
50%	4997.500000	2016-06-26 00:00:00	2016-06-29 00:00:00	56430.500000	54
75%	7495.750000	2017-05-14 00:00:00	2017-05-18 00:00:00	90008.000000	209
max	9994.000000	2017-12-30 00:00:00	2018-01-05 00:00:00	99301.000000	22638
std	2885.163629	NaN	NaN	32063.693350	62

In [277…

```
## 3. Exploratory Data Analysis (EDA)
```

In [278…

```
# Category count
df['category'].value_counts()
```

Out[278…
```
category
Office Supplies    6026
Furniture          2121
Technology         1847
Name: count, dtype: int64
```

In [279…
```
df['sub_category'].value_counts().head(17)
```

Out[279…
```
sub_category
Binders        1523
Paper          1370
Furnishings     957
Phones          889
Storage         846
Art             796
Accessories     775
Chairs          617
Appliances      466
Labels          364
Tables          319
Envelopes       254
Bookcases       228
Fasteners       217
Supplies        190
Machines        115
Copiers          68
Name: count, dtype: int64
```

In [280…
```
# Most common states

df['state'].value_counts().head(10)
```

Out[280…
```
state
California        2001
New York          1128
Texas              985
Pennsylvania       587
Washington         506
Illinois           492
Ohio               469
Florida            383
Michigan           255
North Carolina     249
Name: count, dtype: int64
```

4. Sales & Profit Breakdown

In [281…
```
# Total profit by region

df.groupby("region")["profit"].sum()
```

Out[281…
```
region
Central     39706.3625
East        91522.7800
South       46749.4303
West       108418.4489
Name: profit, dtype: float64
```

In [282…
```
# Total sales per category.

df.groupby("category")["sales"].sum()
```

Out[282…
```
category
Furniture          741999.7953
Office Supplies    719047.0320
Technology         836154.0330
Name: sales, dtype: float64
```

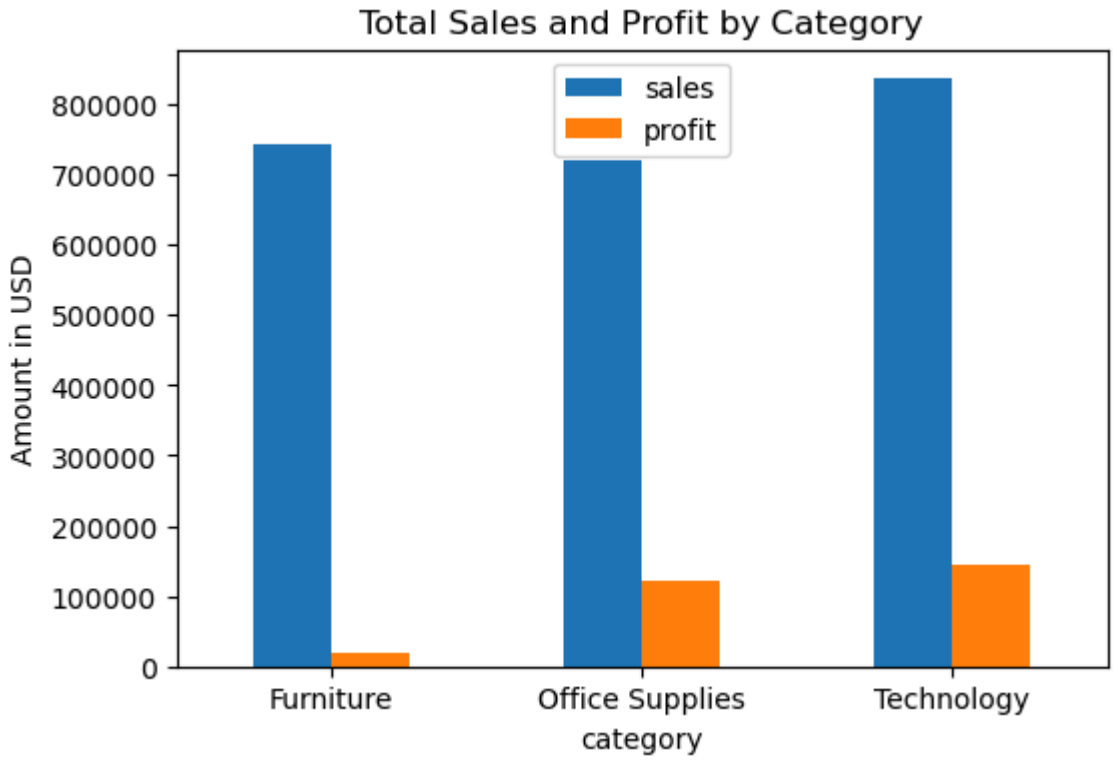
In [283…
```
# Total discount per sub category

df.groupby("sub_category")["discount"].sum()
```

Out[283…
```
sub_category
Accessories     60.80
Appliances      77.60
Art             59.60
Binders        567.00
Bookcases       48.14
Chairs         105.00
Copiers         11.00
Envelopes       20.40
Fasteners       17.80
Furnishings    132.40
Labels          25.00
Machines        35.20
Paper          102.60
Phones         137.40
Storage         63.20
Supplies        14.60
Tables          83.35
Name: discount, dtype: float64
```

In [284…
```
# Average discount per sub category

df.groupby("sub_category")["discount"].mean()
```

Out[284…
```
sub_category
Accessories    0.078452
Appliances     0.166524
Art            0.074874
Binders        0.372292
Bookcases      0.211140
Chairs         0.170178
Copiers        0.161765
Envelopes      0.080315
Fasteners      0.082028
Furnishings    0.138349
Labels         0.068681
Machines       0.306087
Paper          0.074891
Phones         0.154556
Storage        0.074704
Supplies       0.076842
Tables         0.261285
Name: discount, dtype: float64
```

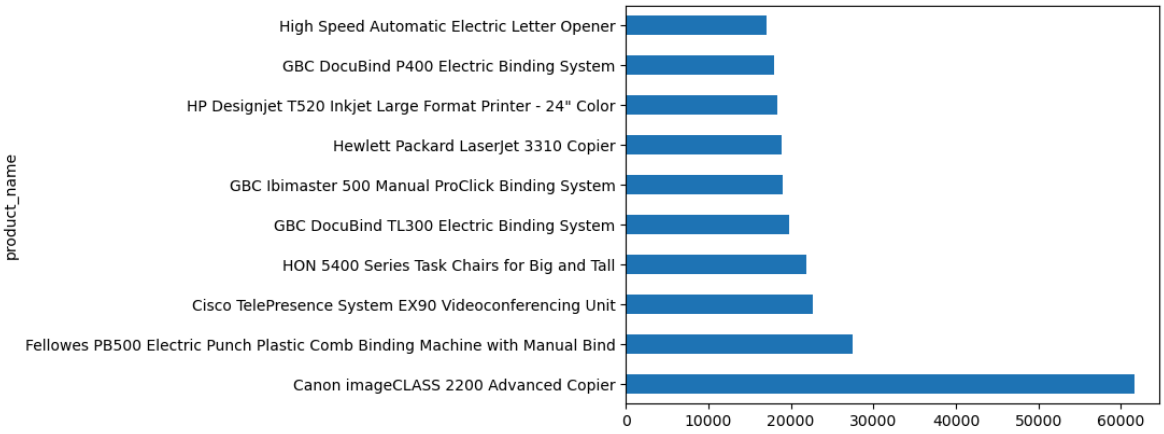
In [285…
```
# Visualisation of the dataset.
# Sales and profit per category

df.groupby("category")[["sales", "profit"]].sum().plot(kind="bar", figsiz
plt.title("Total Sales and Profit by Category")
plt.ylabel("Amount in USD")
plt.xticks(rotation=0)
plt.show()
```



Technology had the highest profit margins but Furniture made frequent losses despite high sales.

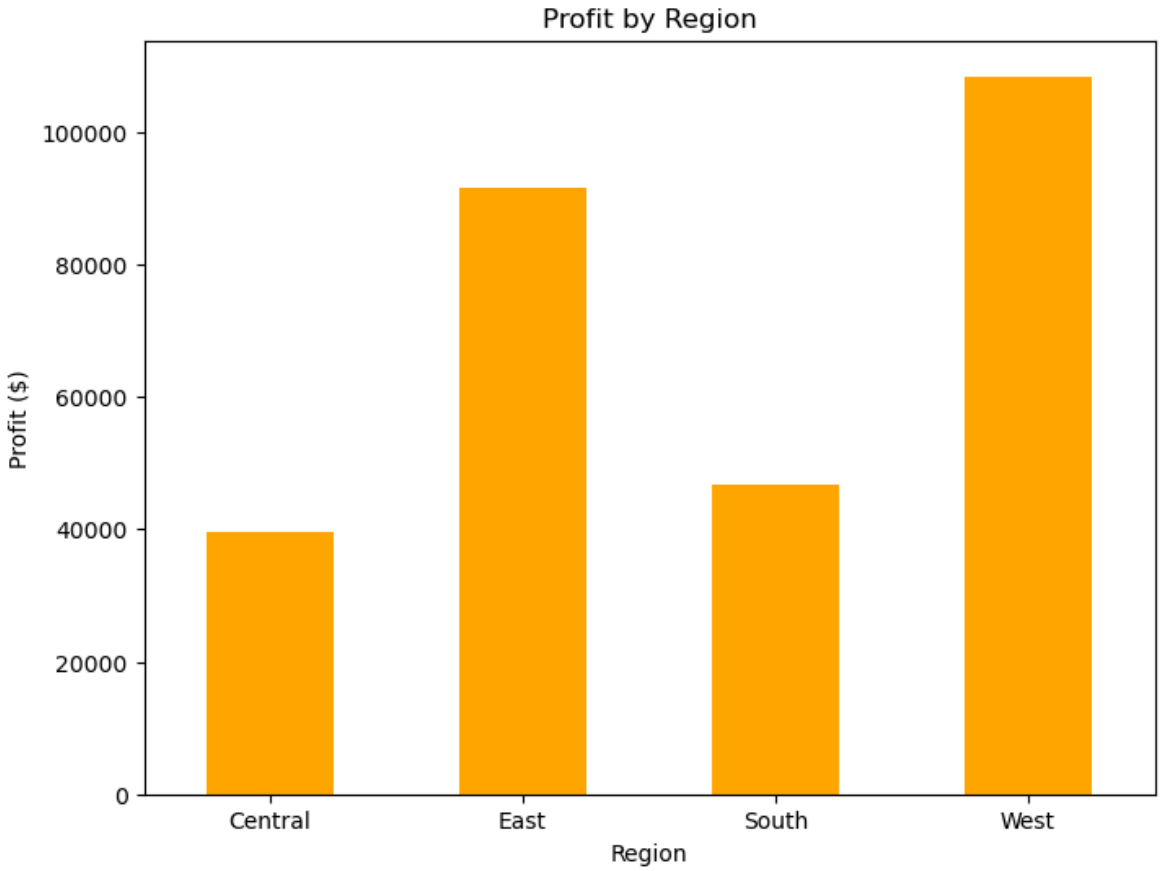
In [286…
```
# Top ten sold products .

top_products = df.groupby("product_name")["sales"].sum().sort_values(asce
top_products.plot(kind='barh')
```

Out[286… <Axes: ylabel='product_name'>



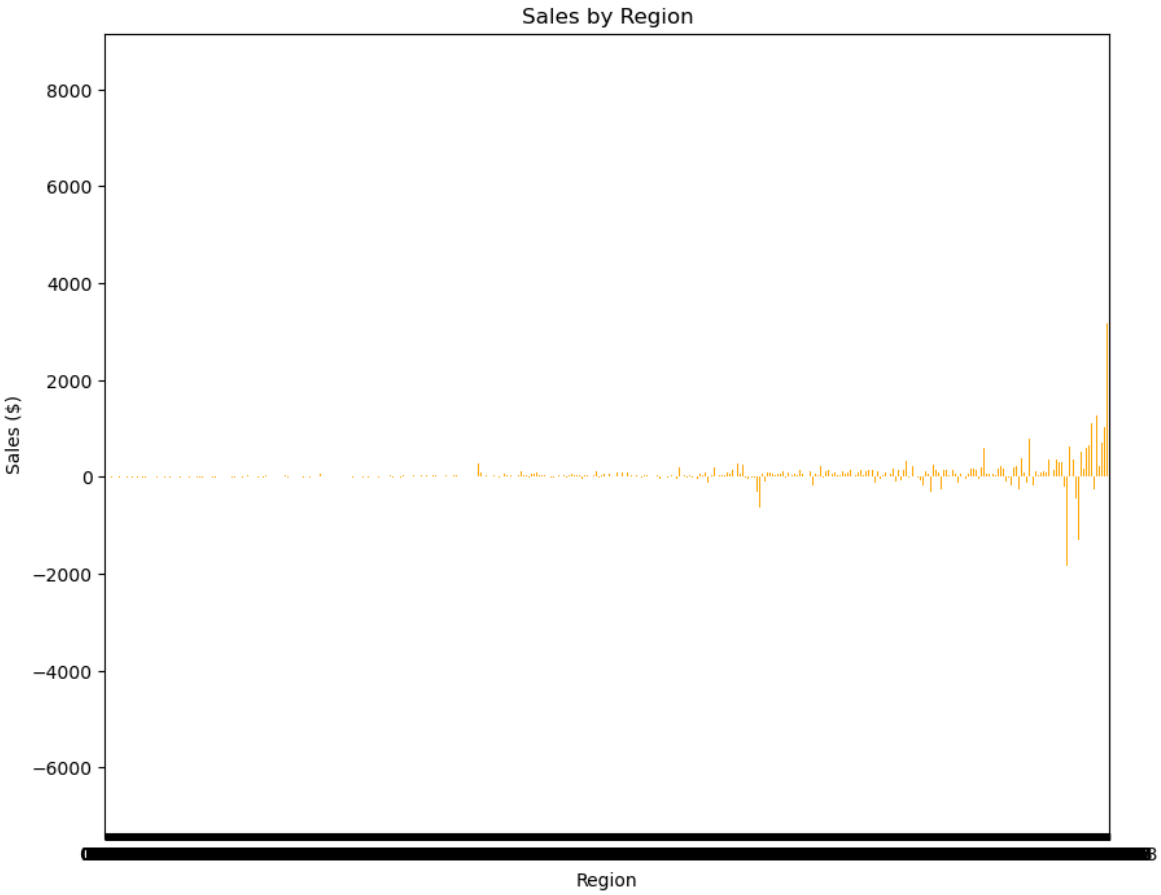
In [287…

```
# Profit per region

df.groupby("region")["profit"].sum().plot(kind='bar', color='orange', fig
plt.title("Profit by Region")
plt.ylabel("Profit ($)")
plt.xlabel("Region")
plt.xticks(rotation=0)
plt.show()
```



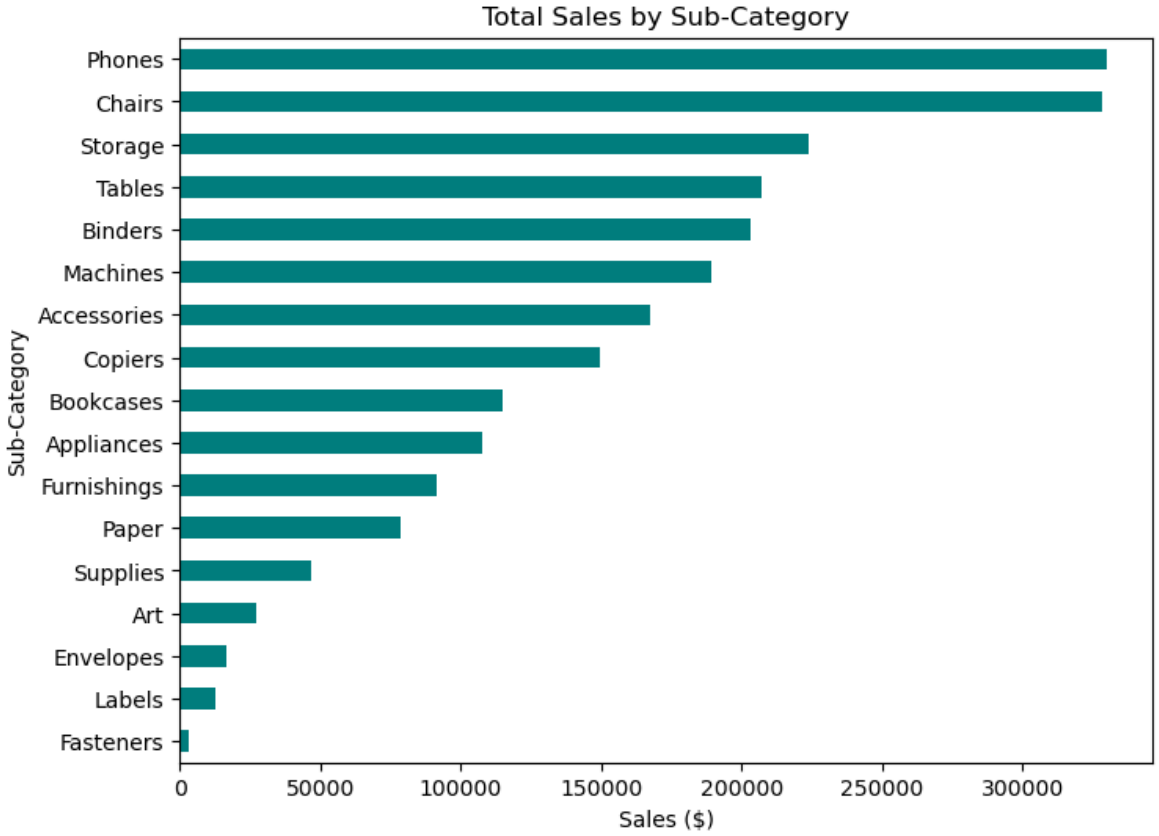
In [288…

```
# Profit per region

df.groupby("sales")["profit"].sum().plot(kind='bar', color='orange', figs
plt.title("Sales by Region")
plt.ylabel("Sales ($)")
plt.xlabel("Region")
plt.xticks(rotation=0)
plt.show()
```



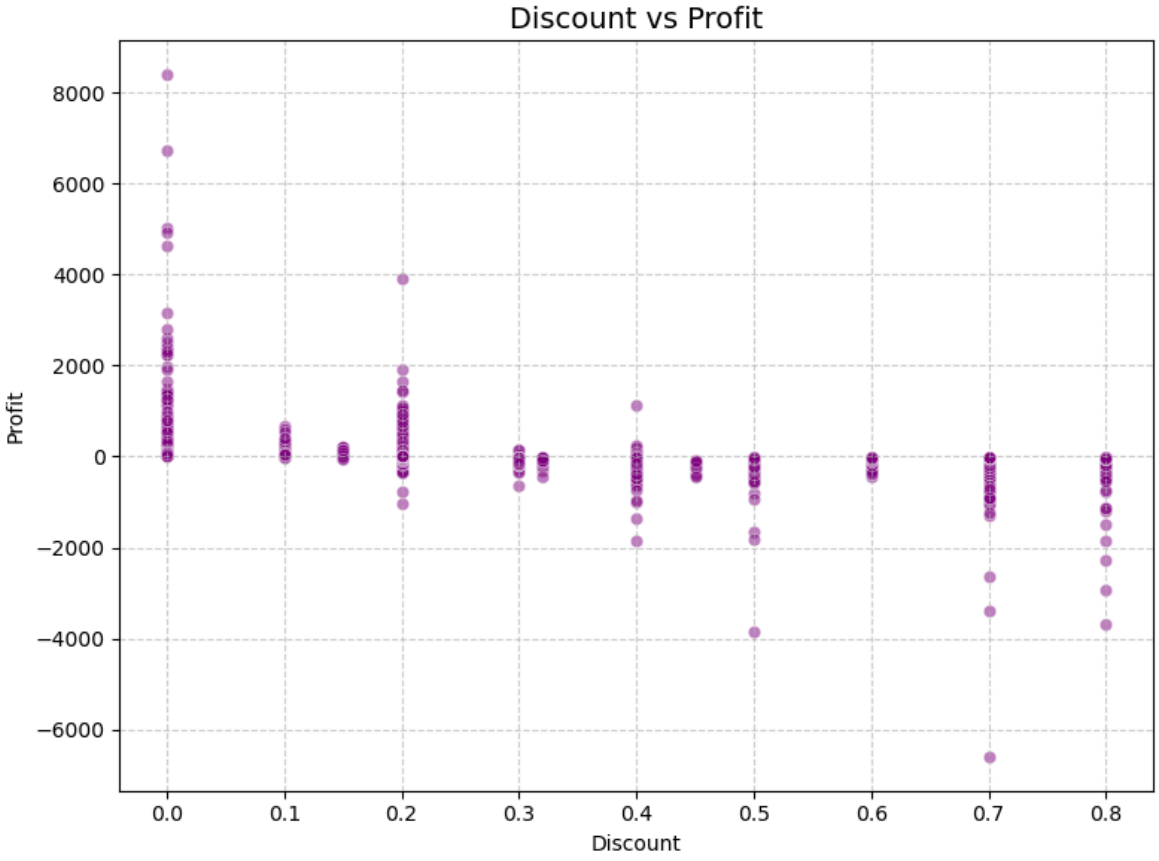
In [289…

```
# Visualisation of sales per sub-category.

df.groupby("sub_category")["sales"].sum().sort_values().plot(kind="barh",
plt.title("Total Sales by Sub-Category")
plt.xlabel("Sales ($)")
plt.ylabel("Sub-Category")
# plt.xticks(rotation=77)
plt.show()
```



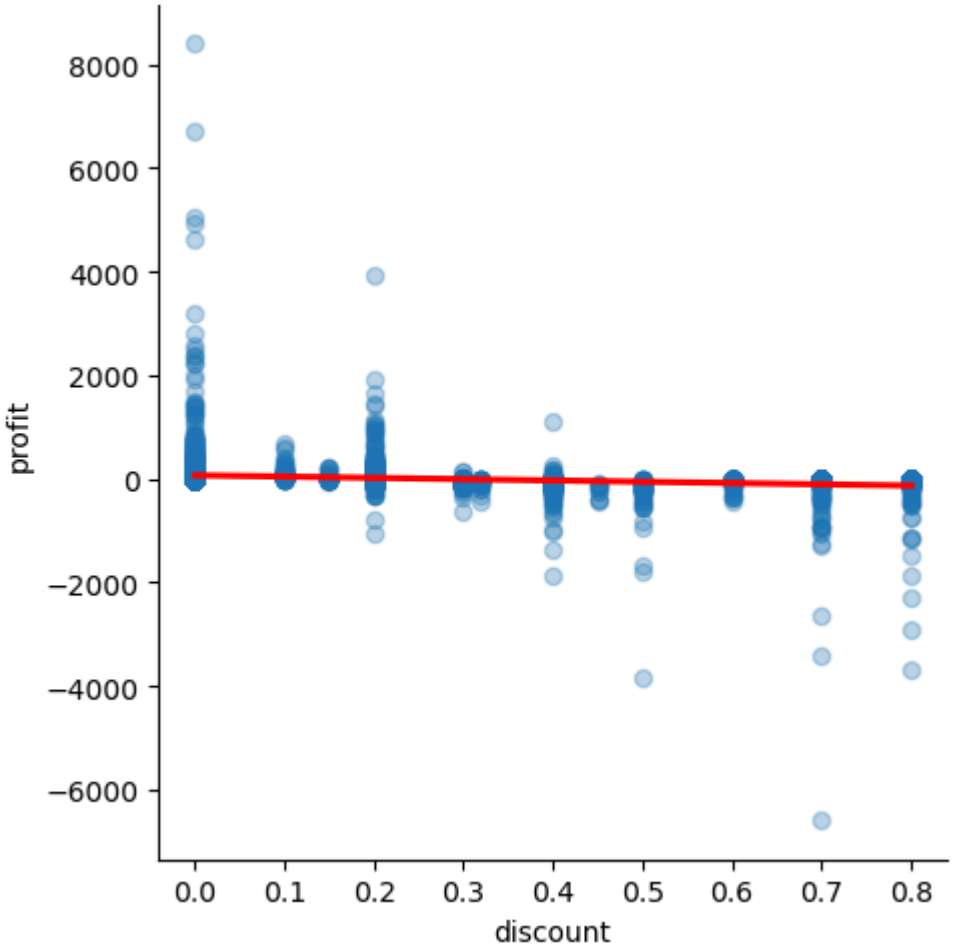
In [290…

```
# Would discounts affect profit?

plt.figure(figsize=(8, 6))
sns.scatterplot(x='discount', y='profit', data=df, alpha=0.5, color='purp
plt.title('Discount vs Profit', fontsize=14)
plt.xlabel('Discount')
plt.ylabel('Profit')
plt.grid(True, linestyle='--', alpha=0.6)
plt.tight_layout()
plt.show()
```



In [291…
```
# Trend line to visualise the relationship between discounts and profits.

sns.lmplot(x='discount', y='profit', data=df, scatter_kws={'alpha':0.3},
```

Out[291…
```
<seaborn.axisgrid.FacetGrid at 0x148d65810>
```



5. Shipping & Discounts

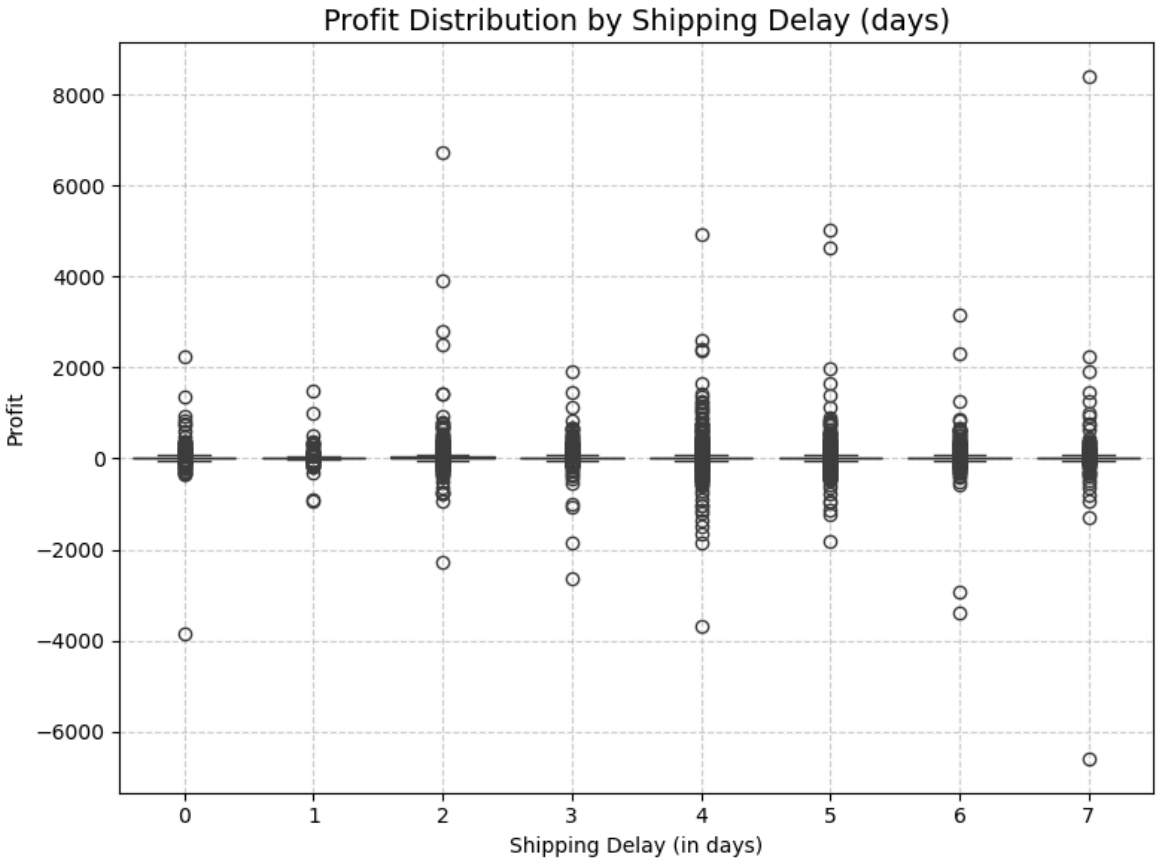
In [292…
```
# Create a new column for shipping delay in days
df['shipping_delay'] = (df['ship_date'] - df['order_date']).dt.days
```

In [293…
```
# Confirming to see if the last code worked

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 9994 entries, CA-2016-152156 to CA-2017-119914
Data columns (total 21 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   row_id          9994 non-null   int64
 1   order_date      9994 non-null   datetime64[ns]
 2   ship_date       9994 non-null   datetime64[ns]
 3   ship_mode       9994 non-null   object
 4   customer_id     9994 non-null   object
 5   customer_name   9994 non-null   object
 6   segment         9994 non-null   object
 7   country         9994 non-null   object
 8   city            9994 non-null   object
 9   state           9994 non-null   object
 10  postal_code     9994 non-null   int64
 11  region          9994 non-null   object
 12  product_id      9994 non-null   object
 13  category        9994 non-null   object
 14  sub_category    9994 non-null   object
 15  product_name    9994 non-null   object
 16  sales           9994 non-null   float64
 17  quantity        9994 non-null   int64
 18  discount        9994 non-null   float64
 19  profit          9994 non-null   float64
 20  shipping_delay  9994 non-null   int64
dtypes: datetime64[ns](2), float64(3), int64(4), object(12)
memory usage: 1.9+ MB
```

In [294…

```
# Does the profit vary with shipping time?

plt.figure(figsize=(8, 6))
sns.boxplot(x='shipping_delay', y='profit', data=df)
plt.title('Profit Distribution by Shipping Delay (days)', fontsize=14)
plt.xlabel('Shipping Delay (in days)')
plt.ylabel('Profit')
plt.xticks(rotation=0)
plt.grid(True, linestyle='--', alpha=0.6)
plt.tight_layout()
plt.show()
```



In [295…
```
# Average profit by the shipping delay range.
# The rows with missing or negative delays will be excluded to ensure acc
```

In [296…
```
# Describe this column

df['shipping_delay'].describe()
```

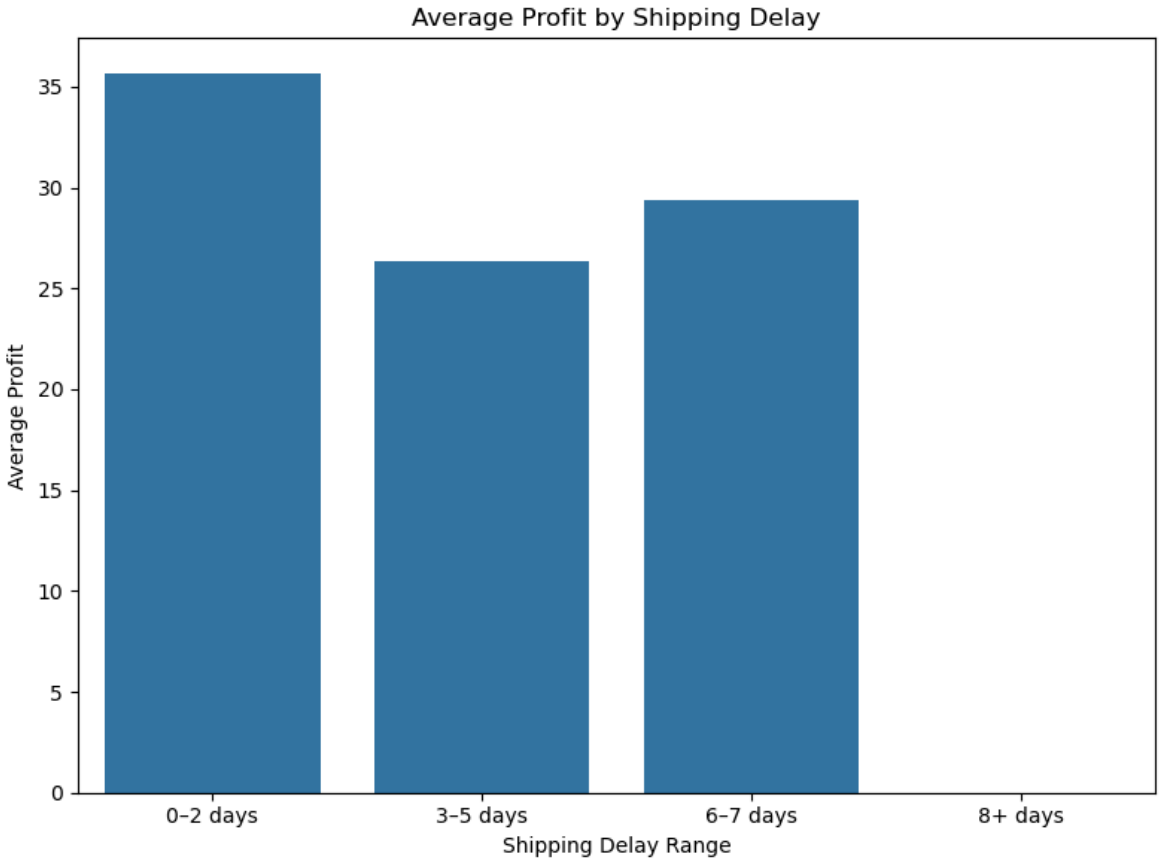
Out[296…
```
count    9994.000000
mean        3.958175
std         1.747567
min         0.000000
25%         3.000000
50%         4.000000
75%         5.000000
max         7.000000
Name: shipping_delay, dtype: float64
```

In [297…
```
# Create delay buckets with fixed bin range

df['delay_bucket'] = pd.cut(
    df['shipping_delay'],
    bins=[0, 2, 5, 7, 15],  # 15 is safely higher than your max of 7
    labels=['0–2 days', '3–5 days', '6–7 days', '8+ days'])
```

In [298…
```
# Group by bucket and compute average profit

# delay_profit = df.groupby('delay_bucket')['profit'].mean().reset_index(
# /var/folders/yh/2rmbgwj12sv2byn6m4jxwy680000gn/T/ipykernel_3740/3049045
# delay_profit = df.groupby('delay_bucket')['profit'].mean().reset_index(
# Did not use that cause of the error message

delay_profit = df.groupby('delay_bucket', observed=True)['profit'].mean()
```

In [299…

```
# Plotting the result

plt.figure(figsize=(8,6))
sns.barplot(x='delay_bucket', y='profit', data=delay_profit)
plt.title('Average Profit by Shipping Delay')
plt.xlabel('Shipping Delay Range')
plt.ylabel('Average Profit')
plt.tight_layout()
plt.show()
```



6. Insights & Recommendations

Analyzing Sales and Profitability Trends in Superstore Data (2014–2017)

Project Overview:

This project analyzes four years(2014 - 2017) of sales and profit data from a global superstore. The goal was to identify key performance patterns across product categories, regions, and customer segments. The objective is to identify where the business is most profitable, where it’s losing money, changes that could drive profit, and what products or regions need closer attention.

Tools Used:

Python (Pandas, Matplotlib, Seaborn) Jupyter Notebook Superstore Sales Dataset

Key Analyses Performed:

Descriptive statistics Missing value checks Exploratory analysis Sales and profit comparison by category and region Average discount per sub_category Scatterplot of discount vs profit

Key Findings (Insights):

“Technology” had the highest profit margins in all the regions. "Furniture” had high sales but lower profits, including some losses in "Tables", probably due to high discounts. The South region had the lowest profit overall despite the consistent sales. Orders with higher discounts(above 30%) were often associated with negative profits. Over 5 days, shipping delays were linked with reduced profits.

Recommendations

Reduce discounting(max 30%), especially on unprofitable sub-categories like tables. Re-evaluate logistics and marketing spend in the South region. Prioritize shipping, logistics improvements in regions with frequent delays Invest more in top-performing categories like Technology and Phones. Use the customer segment and product category cross-analysis for targeted campaigns.

Note

This project has helped strengthen my understanding of exploratory data analysis, data storytelling using Python, and how to turn raw sales data into actionable business insights.

In [ ]: