Half Life – Junior Customer Analytics Assignment

Part 1 – Data Exploration & Cleaning

```
# Importing basic libraries
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd

# Loading customer, transaction, and product datasets
customers = pd.read_csv('/content/customers.csv')
transactions = pd.read_csv('/content/transactions.csv')
products = pd.read_csv('/content/products.csv')

# Display the info for each dataset
print("--- Customers Info ---")
customers.info()
print("\n" + "="*30 + "\n")

print("--- Transactions Info ---")
transactions.info()
print("\n" + "="*30 + "\n")

print("--- Products Info ---")
products.info()
```

```
--- Customers Info ---
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 13 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   customer_id        5000 non-null   object
 1   registration_date  5000 non-null   object
 2   email              4900 non-null   object
 3   first_name         5000 non-null   object
 4   last_name          5000 non-null   object
 5   age                4950 non-null   float64
 6   gender             5000 non-null   object
 7   city               5000 non-null   object
 8   province           5000 non-null   object
 9   country            5000 non-null   object
 10  postal_code        5000 non-null   object
 11  customer_segment   5000 non-null   object
 12  marketing_consent  5000 non-null   bool
```

```
dtypes: bool(1), float64(1), object(11)
memory usage: 473.8+ KB

==============================

--- Transactions Info ---
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 23814 entries, 0 to 23813
Data columns (total 12 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   transaction_id    23814 non-null  object
 1   customer_id       23814 non-null  object
 2   transaction_date  23814 non-null  object
 3   product_id        23814 non-null  object
 4   quantity          23814 non-null  int64
 5   unit_price        23814 non-null  float64
 6   total_amount      23814 non-null  float64
 7   discount_amount   23814 non-null  float64
 8   payment_method    23814 non-null  object
 9   shipping_cost     23814 non-null  float64
 10  order_status      23814 non-null  object
 11  channel           23814 non-null  object
dtypes: float64(4), int64(1), object(7)
memory usage: 2.2+ MB

==============================

--- Products Info ---
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 13 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   product_id      200 non-null    object
 1   product_name    200 non-null    object
 2   category        200 non-null    object
 3   subcategory     200 non-null    object
 4   brand           200 non-null    object
 5   current_price   200 non-null    float64
 6   cost_price      200 non-null    float64
 7   stock_quantity  200 non-null    int64
 8   weight_kg       200 non-null    float64
 9   launch_date     200 non-null    object
 10  is_active       200 non-null    bool
 11  rating          190 non-null    float64
 12  review_count    200 non-null    int64
dtypes: bool(1), float64(4), int64(2), object(6)
memory usage: 19.1+ KB
```

Check missing values

```
customers.isnull().sum()
```

```
customer_id            0
registration_date      0
email                100
first_name             0
last_name              0
age                   50
gender                 0
city                   0
province               0
country                0
postal_code            0
customer_segment       0
marketing_consent      0
dtype: int64
```

```
transactions.isnull().sum()
```

```
transaction_id      0
customer_id         0
transaction_date    0
product_id          0
quantity            0
unit_price          0
total_amount        0
discount_amount     0
payment_method      0
shipping_cost       0
order_status        0
channel             0
dtype: int64
```

```
products.isnull().sum()
```

```
product_id         0
product_name       0
category           0
subcategory        0
brand              0
current_price      0
cost_price         0
stock_quantity     0
weight_kg          0
launch_date        0
is_active          0
rating            10
```

```
review_count      0
dtype: int64

# Fill missing ratings with the median
products['rating'] =
products['rating'].fillna(products['rating'].median())
# Fill missing customer age with median
customers['age'] = customers['age'].fillna(customers['age'].median())
# Fill missing emails with 'unknown'
customers['email'] = customers['email'].fillna('Unknown')

# Convert all date columns to datetime objects
products['launch_date'] = pd.to_datetime(products['launch_date'])
customers['registration_date'] =
pd.to_datetime(customers['registration_date'])
transactions['transaction_date'] =
pd.to_datetime(transactions['transaction_date'])

# Keep only completed orders for analysis
completed_txns = transactions[transactions['order_status'] ==
'Completed']

# Define a reference date
reference_date = completed_txns['transaction_date'].max()

# Calculate total spend per customer
total_spend = completed_txns.groupby('customer_id')
['total_amount'].sum().reset_index()
total_spend.rename(columns={'total_amount': 'total_spend'},
inplace=True)

# Calculate number of orders per customer
num_orders = completed_txns.groupby('customer_id')
['transaction_id'].nunique().reset_index()
num_orders.rename(columns={'transaction_id': 'num_orders'},
inplace=True)

# Find the most recent purchase date for each customer
last_purchase = completed_txns.groupby('customer_id')
['transaction_date'].max().reset_index()
last_purchase.rename(columns={'transaction_date':
'last_purchase_date'}, inplace=True)

# Merge all customer-level metrics into one table
customer_stats = total_spend.merge(num_orders, on='customer_id') \
                            .merge(last_purchase, on='customer_id')

# Calculate how many days since the customer last purchased
customer_stats['days_since_last_purchase'] = (
    reference_date - customer_stats['last_purchase_date']
).dt.days
```

```python
# Calculate average order value
customer_stats['avg_order_value'] = (
    customer_stats['total_spend'] / customer_stats['num_orders']
)

tx_cust = transactions.merge(customers, on='customer_id', how='left')

full_data = tx_cust.merge(products, on='product_id', how='left')

full_data.head()
```


```
{"type":"dataframe","variable_name":"full_data"}
```


```python
customer_stats.head()
```


```
{"summary":"{\n  \"name\": \"customer_stats\",\n  \"rows\": 3608,\n
\"fields\": [\n    {\n      \"column\": \"customer_id\",\n
\"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 3608,\n        \"samples\": [\n
\"CUST004205\",\n          \"CUST003060\",\n          \"CUST002858\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"total_spend\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
3082.842253018933,\n        \"min\": 11.08,\n        \"max\":
66854.53,\n        \"num_unique_values\": 2960,\n        \"samples\":
[\n          354.06,\n          1087.52,\n          378.34\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"num_orders\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 12,\n        \"min\": 1,\n
\"max\": 274,\n        \"num_unique_values\": 76,\n
\"samples\": [\n          37,\n          25,\n          6\n        ],\
n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"last_purchase_date\",\n
\"properties\": {\n        \"dtype\": \"date\",\n        \"min\":
\"2023-01-01 17:18:41\",\n        \"max\": \"2024-12-31 22:36:03\",\n
\"num_unique_values\": 3607,\n        \"samples\": [\n
\"2023-09-02 19:47:11\",\n          \"2024-10-14 23:46:43\",\n
\"2024-11-26 09:32:20\"\n        ],\n        \"semantic_type\": \"\",\
n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"days_since_last_purchase\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 190,\n        \"min\": 1,\n
\"max\": 731,\n        \"num_unique_values\": 662,\n
\"samples\": [\n          282,\n          474,\n          292\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"avg_order_value\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
131.3184717050357,\n        \"min\": 11.08,\n        \"max\":
1296.33,\n        \"num_unique_values\": 2965,\n        \"samples\":
[\n          126.354,\n          338.56,\n          250.59\
```

```
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"customer_stats"}

# Save customer analytics dataset for further analysis
customer_stats.to_csv('customer_analytics.csv', index=False)

# Save dataset for further analysis
full_data.to_csv('full_data.csv', index=False)
```

Data Exploration, Cleaning, and Preparation Summary

For this analysis, I first imported the required Python libraries and loaded the three datasets provided: customers.csv, transactions.csv, and products.csv. Before doing any analysis, I explored each dataset separately to understand their structure, data types, and purpose.

Understanding the datasets:

1.Customers dataset: Contains customer-level information such as customer ID, age, email, and registration date.

2.Transactions dataset: Contains transaction-level details including transaction ID, customer ID, product ID, transaction date, order status, quantity, and total amount.

3.Products dataset: Contains product-related information such as product ID, product name, category, subcategory, brand, price, and rating.

After understanding the datasets, I checked for missing values in each file. I found a small number of missing values in product ratings and customer age. Since only a few values were missing and these fields were not the main focus of the analysis, I chose to fill them using the median rather than the mean, as the median is less affected by extreme values. For missing customer email values, I replaced them with “Unknown” instead of removing records, because it is important to retain all customers for analysis. No rows were removed during cleaning. I also converted all date-related columns into datetime format to support time-based analysis later.

After cleaning, I prepared the data for analysis by creating two datasets for different purposes:

1.Transaction-level dataset (full_data): Created by merging transactions with customer and product information. This dataset is used for sales and product-level analysis, such as revenue trends and category performance.

2.Customer-level analytics dataset(customer_analytics): Created by aggregating transaction data to calculate total spend, number of orders, average order value, and days since last purchase for each customer. These features summarize customer purchasing behavior and are commonly used in customer segmentation and marketing analysis.

Part 2 – Basic Customer Segmentation

```
# Calculate thresholds using percentiles
spend_75 = customer_stats['total_spend'].quantile(0.75)
spend_40 = customer_stats['total_spend'].quantile(0.40)
```

```
orders_75 = customer_stats['num_orders'].quantile(0.75)
recency_75 = customer_stats['days_since_last_purchase'].quantile(0.75)

# Create a new column for customer segment
def assign_segment(row):
    if (row['total_spend'] >= spend_75) and (row['num_orders'] >=
orders_75):
        return 'High-Value / Loyal'
    elif row['days_since_last_purchase'] >= recency_75:
        return 'Inactive'
    elif row['total_spend'] >= spend_40:
        return 'Regular'
    else:
        return 'Occasional'

customer_stats['customer_segment'] =
customer_stats.apply(assign_segment, axis=1)

customer_stats['customer_segment'].value_counts()
```

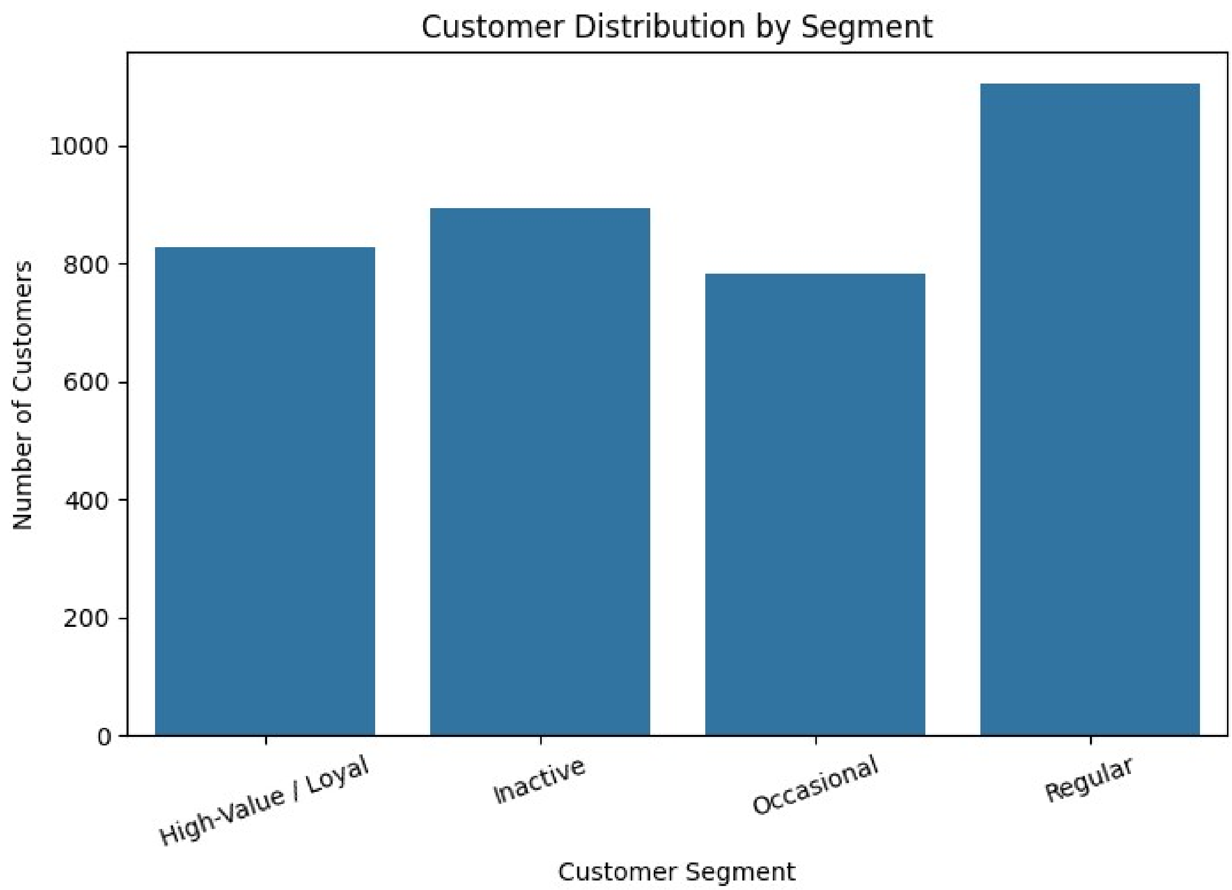
```
customer_segment
Regular               1103
Inactive               894
High-Value / Loyal     828
Occasional             783
Name: count, dtype: int64
```

```
segment_summary = (
    customer_stats
    .groupby('customer_segment')
    .agg(
        number_of_customers=('customer_id', 'count'),
        avg_spend=('total_spend', 'mean'),
        avg_orders=('num_orders', 'mean')
    )
    .reset_index()
)

segment_summary
```


```
{"summary":"{\n  \"name\": \"segment_summary\",\n  \"rows\": 4,\n
\"fields\": [\n    {\n      \"column\": \"customer_segment\",\n
\"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 4,\n        \"samples\": [\n
\"Inactive\",\n          \"Regular\",\n          \"High-Value /
Loyal\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"number_of_customers\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 141,\n        \"min\": 783,\n
\"max\": 1103,\n        \"num_unique_values\": 4,\n
\"samples\": [\n          894,\n          1103,\n          828\n
```

```
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"avg_spend\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
1927.5331540285435,\n        \"min\": 281.42029374201786,\n
\"max\": 4369.32695652174,\n        \"num_unique_values\": 4,\n
\"samples\": [\n          399.84447427293065,\n
1031.4705258386218,\n          4369.32695652174\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"avg_orders\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
7.6573461057880925,\n        \"min\": 1.6409395973154361,\n
\"max\": 17.632850241545892,\n        \"num_unique_values\": 4,\n
\"samples\": [\n          1.6409395973154361,\n
4.116047144152312,\n          17.632850241545892\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    }\n  ]\n}","type":"dataframe","variable_name":"segment_summary"}
```

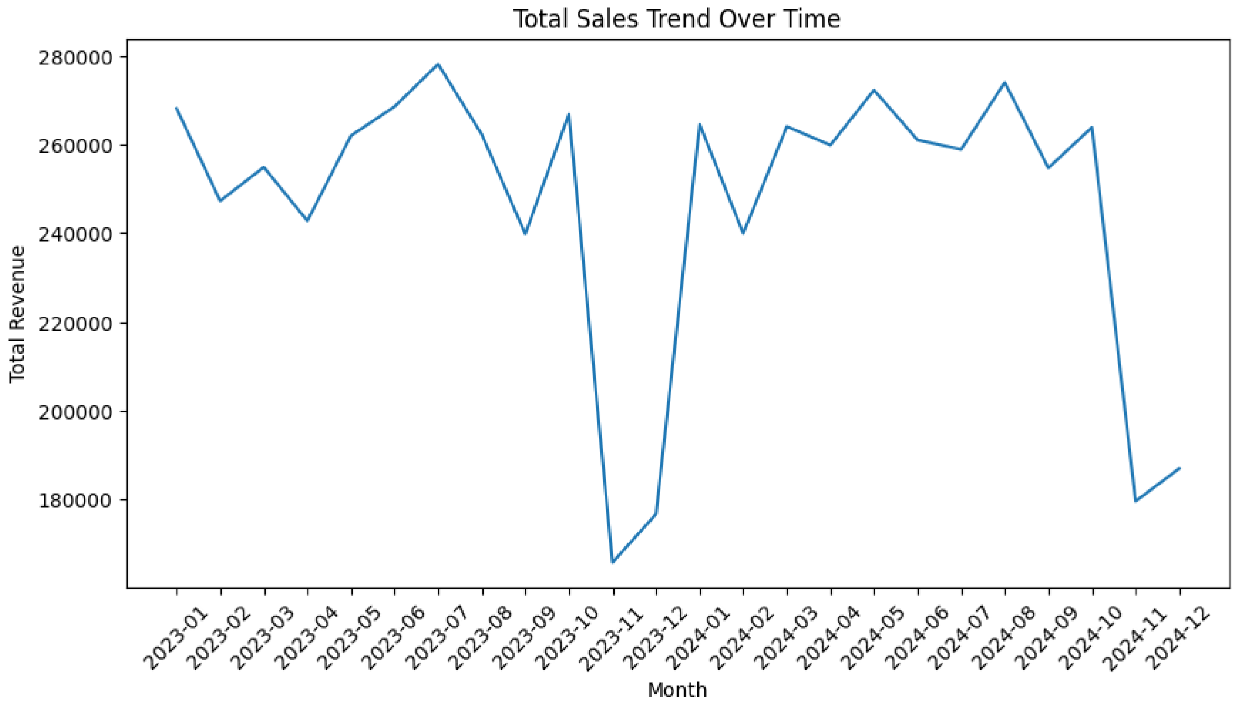

```
plt.figure(figsize=(8,5))
sns.barplot(
    data=segment_summary,
    x='customer_segment',
    y='number_of_customers'
)
plt.title('Customer Distribution by Segment')
plt.xlabel('Customer Segment')
plt.ylabel('Number of Customers')
plt.xticks(rotation=20)
plt.show()
```



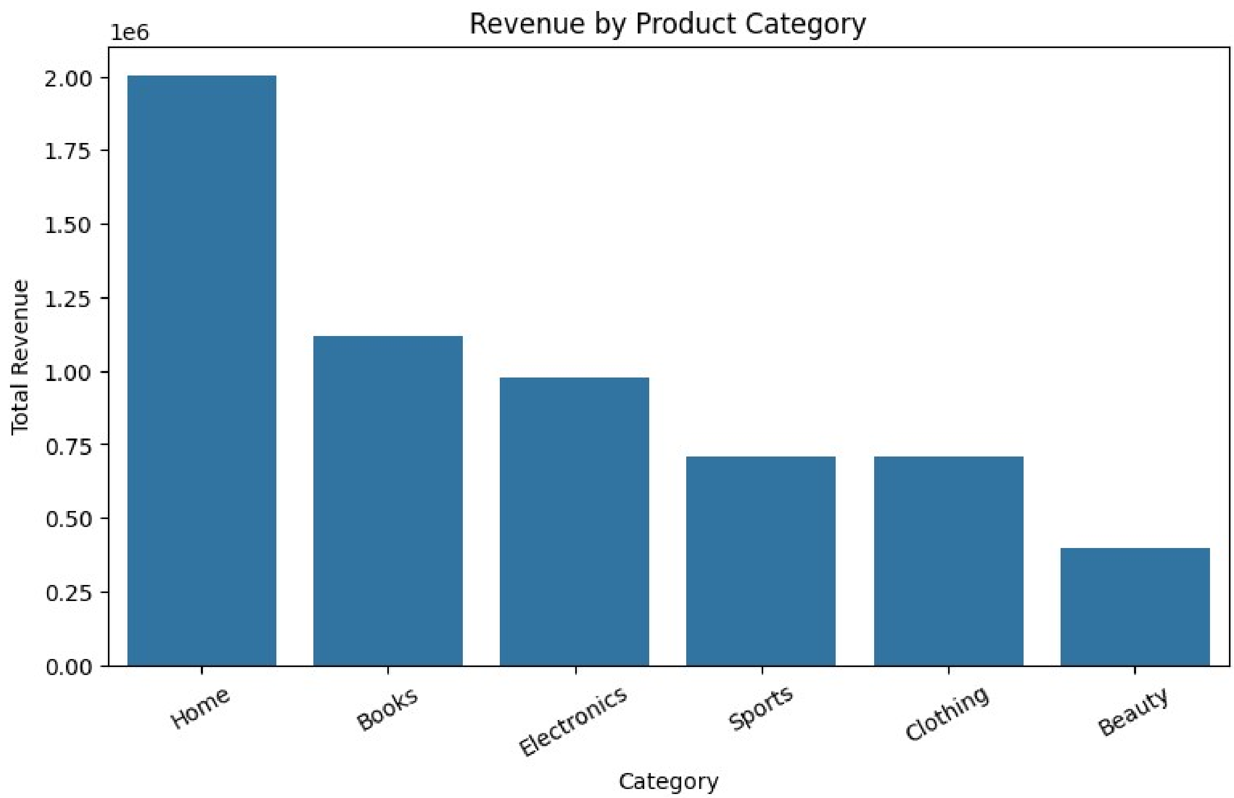
Part 3 – Basic Sales Analysis

```
full_data['revenue'] = full_data['quantity'] * full_data['unit_price']

#Sales Trend Over Time
# Create month column
full_data['order_month'] =
full_data['transaction_date'].dt.to_period('M').astype(str)

monthly_sales = (
    full_data
    .groupby('order_month')
    .agg(
        total_revenue=('revenue', 'sum'),
        number_of_orders=('transaction_id', 'nunique')
    )
    .reset_index()
)
plt.figure(figsize=(10,5))
sns.lineplot(data=monthly_sales, x='order_month', y='total_revenue')
plt.title('Total Sales Trend Over Time')
plt.xlabel('Month')
plt.ylabel('Total Revenue')
```

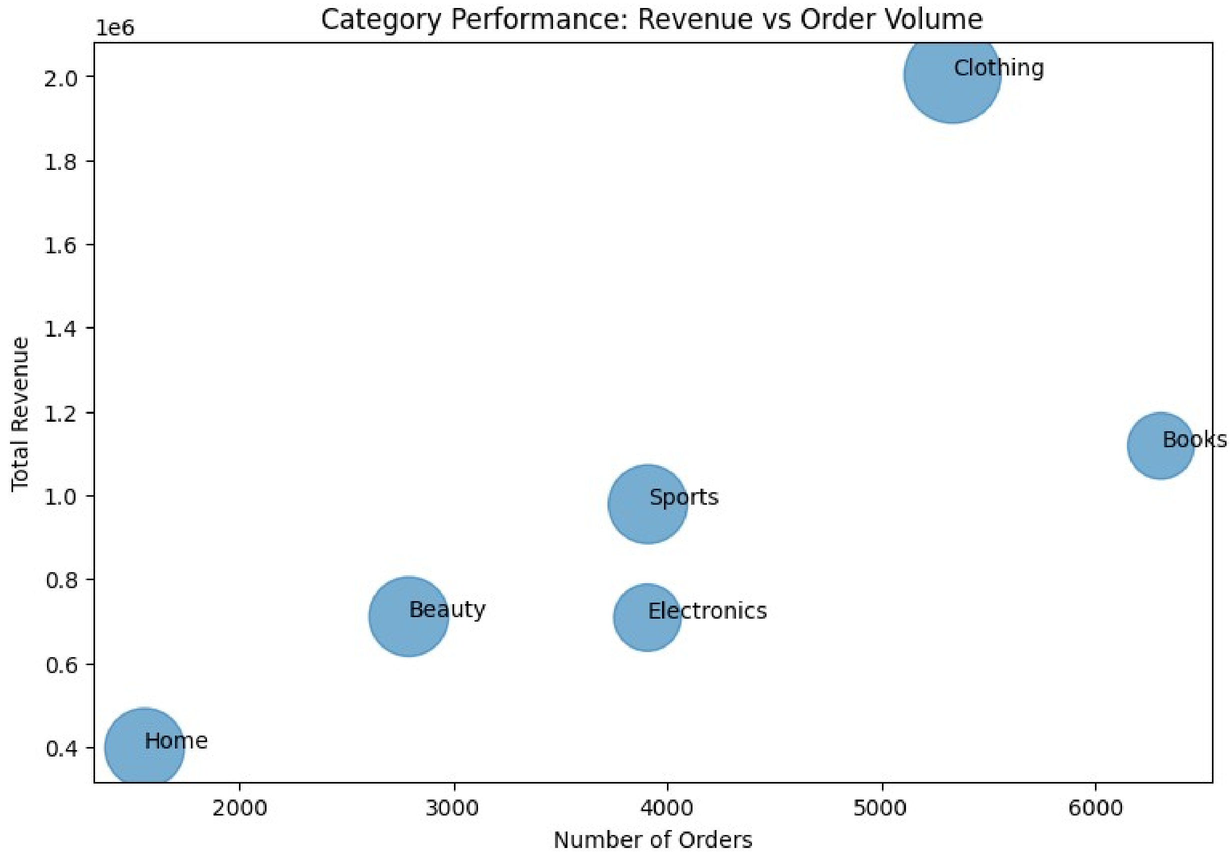
```
plt.xticks(rotation=45)
plt.show()
```




```
#Best Performing Product Categories
category_sales = (
    full_data
    .groupby('category')
    .agg(
        total_revenue=('revenue', 'sum'),
        number_of_orders=('transaction_id', 'nunique'),
        avg_order_value=('revenue', 'mean')
    )
    .reset_index()
    .sort_values(by='total_revenue', ascending=False)
)
plt.figure(figsize=(9,5))
sns.barplot(data=category_sales, x='category', y='total_revenue')
plt.title('Revenue by Product Category')
plt.xlabel('Category')
plt.ylabel('Total Revenue')
plt.xticks(rotation=30)
plt.show()
```




```
plt.figure(figsize=(9,6))
plt.scatter(
    category_sales['number_of_orders'],
    category_sales['total_revenue'],
    s=category_sales['avg_order_value'] * 5,
    alpha=0.6
)

for i, txt in enumerate(category_sales['category']):
    plt.annotate(txt, (
        category_sales['number_of_orders'][i],
        category_sales['total_revenue'][i]
    ))

plt.xlabel('Number of Orders')
plt.ylabel('Total Revenue')
plt.title('Category Performance: Revenue vs Order Volume')
plt.show()
```



Sales Analysis – Key Findings & Insights

In this sales analysis, I used three main metrics revenue, number of orders, and average order value (AOV) to understand how sales perform over time and across product categories.

1. Sales Trend Over Time

The monthly sales trend shows that revenue is generally stable across most months, with values staying around a similar range. However, there are a few noticeable changes. For example, sales drop sharply around October–November 2023, before recovering strongly in early 2024. Another decline is visible around November 2024, followed by a slight recovery in December 2024. These patterns suggest that sales fluctuations are likely influenced by seasonal factors, promotions, or changes in customer demand rather than long-term performance issues.

1. Revenue by Product Category

When analyzing revenue by product category, it is clear that Home is the strongest revenue-generating category, followed by Books and Electronics. Categories such as Sports and Clothing contribute a moderate amount, while Beauty generates the lowest total revenue.

This shows that customer spending is concentrated in a few categories rather than evenly distributed across all products.

1. Revenue vs Order Volume (Category Comparison)

Comparing revenue with order volume reveals an important insight. Books has a high number of orders but lower total revenue compared to Home, which generates much higher revenue with fewer orders. On the other hand, Clothing shows relatively high revenue with a moderate number of orders, indicating a higher average order value. This confirms that more orders do not always mean higher revenue.

One surprising finding was how strongly revenue is driven by a small number of categories, especially Home, while categories like Books rely more on volume rather than value.

Key Recommendation

The business should prioritize marketing and promotions for Home and Clothing, as these categories generate higher revenue per order. At the same time, strategies such as bundles or upselling could be tested in Books and Beauty to increase their average order value and overall contribution.