Final Project Report

Name : Vanitha S

Batch No : TN_DA_FNB03

Contact Number : 9500708068

Email ID : svanitha694@gmail.com

Project Title : Sales Performance and Customer Behaviour Analysis in E – Commerce (2024 – 2025)

Project Domain : Sales & E – Commerce / Business Analytics

Dataset Link :

https://drive.google.com/file/d/1IXfVsu3aneZ3nBqO5SZ8NsoGSvoaIYbp/view?usp=sharing

Cleaned Dataset Link :

https://drive.google.com/file/d/1dw6tVRJAn_iv6DNAqV4oetW27BSpeQZu/view?usp=sharing

Mentor Name : Kumaran M

Submission Date : 9th December 2025

Sales Performance and Customer Behaviour Analysis in E – Commerce (2024 – 2025)

Domain:

Sales & E – Commerce / Business Analytics

Objective:

1. To identify meaningful patterns and trends in online customer purchasing behaviour.
2. To clean, transform, and pre-process the dataset for accurate sales and business analysis.
3. To analyze sales performance across products, categories, regions, and time periods.
4. To create clear visualizations that highlight revenue trends, customer preferences, and product demand.
5. To discover top-selling and low-selling items to support inventory and marketing strategies.
6. To provide actionable business insights that help improve sales, customer engagement, and overall e-commerce growth.

Outcome:

1. A cleaned and well-processed e-commerce dataset ready for analysis.
2. Sales trends and customer behaviour patterns identified clearly.
3. Meaningful visualizations showing revenue, product demand, and user activity.
4. Top-selling and low-selling products discovered for business focus.
5. Insights to support better decision-making in sales and marketing strategies.

Dataset Information:

Source: Deepdatalake - Online Retail & E-Commerce Dataset

https://deepdatalake.com/details.php?dataset_id=41

Year / Timeline: MARCH 2024 to MARCH 2025

Dataset Description:

customer_id - Unique ID for each customer.

order_date - Date on which the order was placed.

product_id - Unique ID for each product.

category_id - ID representing the product category.

category_name - Name of the product category (e.g., Electronics, Fashion).

product_name - Name of the product ordered.

quantity - Number of units purchased in that order.

price - Price of the product per unit.

payment_method - Mode of payment used (Card, COD, UPI, etc.).

city - City from where the order was placed.

review_score - Customer rating given for the product (1–5 scale).

gender - Gender of the customer (M/F).

age - Age of the customer.

Type of Analysis:

1.Descriptive Analysis:

- Total customers, orders, products
- Top selling categories
- Average order value
- Most used payment method
- City-wise sales
- Average review score
- Age & gender distribution of buyers

2. Diagnostic Analysis:

- Why some categories sell more?
- Why some cities have higher orders?
- Why review scores are low for some products?
- Why certain payment methods are preferred?

3. Predictive Analysis (optional if forecasting required using basic stats):

- Predict which category may get more orders in future
- Predict customer age group likely to buy more
- Estimate future order volume (simple trend)

4. Prescriptive Analysis (recommendations for business decisions):

- Recommend which products should be promoted more
- Suggest best city for marketing campaigns
- Suggest discount ideas for low-selling categories
- Improve product quality based on review scores
- Recommend fastest payment method for conversion

Stages for DA Project

Stage 1 – Problem Definition and Dataset Selection

Problem Definition

The growth of e-commerce has created large volumes of customer, product, and order data. Businesses often struggle to understand buying patterns, customer preferences, sales performance, and product demand. Without proper analysis, companies cannot take effective decisions related to pricing, marketing, stock management, or customer satisfaction.

This project aims to analyze the e-commerce dataset to identify key trends, customer behaviour patterns, product performance, and factors influencing sales and ratings. The insights will help in improving business decisions such as which products to promote, which cities to target, and how to enhance customer satisfaction.

Dataset Selection

- The dataset selected for this project is an E-commerce Sales Dataset (2024–2025) containing customer, order, product, and review information. It includes essential fields such as:

- Customer demographics (age, gender, city)

- Order details (order ID, date, quantity, price)

- Product details (category, ratings, payment method)

- Transaction details (total amount, payment type)

- **The dataset was chosen because:**

- It provides realistic business-level information suitable for analysis.

- It contains 13 meaningful columns, making it moderate and easy to work with.

- It supports multiple analytical techniques such as descriptive, diagnostic, and prescriptive analysis.

- It helps identify sales trends, customer behaviour, and product performance.

- It is ideal for creating visualizations and insights for decision-making.

- Import libraries, load dataset
- Dataset description (rows, columns, features)
- Initial EDA (head, info, describe, shape, null checks)

Code:

```
# importing libraries

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import os
import requests
from bs4 import BeautifulSoup
from urllib.parse import urljoin
import time
from google.colab import drive
!pip install requests
```

Output:

```
Requirement already satisfied: requests in /usr/local/lib/python3.12/dist-packages (2.32.4)
Requirement already satisfied: charset_normalizer<4,>=2 in /usr/local/lib/python3.12/dist-packages (from requests) (3.4.4)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.12/dist-packages (from requests) (3.11)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.12/dist-packages (from requests) (2.5.0)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.12/dist-packages (from requests) (2025.11.12)
```

Code:

```
!pip install requests beautifulsoup4
```

Output:

```
Requirement already satisfied: requests in /usr/local/lib/python3.12/dist-packages (2.32.4)
Requirement already satisfied: beautifulsoup4 in /usr/local/lib/python3.12/dist-packages (4.13.5)
Requirement already satisfied: charset_normalizer<4,>=2 in /usr/local/lib/python3.12/dist-packages (from requests) (3.4.4)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.12/dist-packages (from requests) (3.11)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.12/dist-packages (from requests) (2.5.0)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.12/dist-packages (from requests) (2025.11.12)
Requirement already satisfied: soupsieve>1.2 in /usr/local/lib/python3.12/dist-packages (from beautifulsoup4) (2.8)
Requirement already satisfied: typing-extensions>=4.0.0 in /usr/local/lib/python3.12/dist-packages (from beautifulsoup4) (4.15.0)
```

Code:

```
!pip install selenium
```

Output:

```
Collecting selenium
  Downloading selenium-4.38.0-py3-none-any.whl.metadata (7.5 kB)
Requirement already satisfied: urllib3<3.0,>=2.5.0 in /usr/local/lib/python3.12/dist-packages (from urllib3[socks]<3.0,>=2.5.0->selenium) (2.5.0)
Collecting trio<1.0,>=0.31.0 (from selenium)
  Downloading trio-0.32.0-py3-none-any.whl.metadata (8.5 kB)
Collecting trio-websocket<1.0,>=0.12.2 (from selenium)
  Downloading trio_websocket-0.12.2-py3-none-any.whl.metadata (5.1 kB)
Requirement already satisfied: certifi>=2025.10.5 in /usr/local/lib/python3.12/dist-packages (from selenium) (2025.11.12)
Requirement already satisfied: typing_extensions<5.0,>=4.15.0 in /usr/local/lib/python3.12/dist-packages (from selenium) (4.15.0)
Requirement already satisfied: websocket-client<2.0,>=1.8.0 in /usr/local/lib/python3.12/dist-packages (from selenium) (1.9.0)
Requirement already satisfied: attrs>=23.2.0 in /usr/local/lib/python3.12/dist-packages (from trio<1.0,>=0.31.0->selenium) (25.4.0)
Requirement already satisfied: sortedcontainers in /usr/local/lib/python3.12/dist-packages (from trio<1.0,>=0.31.0->selenium) (2.4.0)
Requirement already satisfied: idna in /usr/local/lib/python3.12/dist-packages (from trio<1.0,>=0.31.0->selenium) (3.11)
Collecting outcome (from trio<1.0,>=0.31.0->selenium)
  Downloading outcome-1.3.0.post0-py2.py3-none-any.whl.metadata (2.6 kB)
Requirement already satisfied: sniffio>=1.3.0 in /usr/local/lib/python3.12/dist-packages (from trio<1.0,>=0.31.0->selenium) (1.3.1)
Collecting wsproto>=0.14 (from trio-websocket<1.0,>=0.12.2->selenium)
  Downloading wsproto-1.3.2-py3-none-any.whl.metadata (5.2 kB)
Requirement already satisfied: pysocks!=1.5.7,<2.0,>=1.5.6 in /usr/local/lib/python3.12/dist-packages (from urllib3[socks]<3.0,>=2.5.0->selenium) (1.7.1)
Requirement already satisfied: h11<1,>=0.16.0 in /usr/local/lib/python3.12/dist-packages (from wsproto>=0.14->trio-websocket<1.0,>=0.12.2->selenium) (0.16.0)
Downloading selenium-4.38.0-py3-none-any.whl (9.7 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 9.7/9.7 MB 66.1 MB/s eta 0:00:00
Downloading trio-0.32.0-py3-none-any.whl (512 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 512.0/512.0 kB 28.0 MB/s eta 0:00:00
Downloading trio_websocket-0.12.2-py3-none-any.whl (21 kB)
Downloading outcome-1.3.0.post0-py2.py3-none-any.whl (10 kB)
Downloading wsproto-1.3.2-py3-none-any.whl (24 kB)
Installing collected packages: wsproto, outcome, trio, trio-websocket, selenium
Successfully installed outcome-1.3.0.post0 selenium-4.38.0 trio-0.32.0 trio-websocket-0.12.2 wsproto-1.3.2
```

Code:

```
drive.mount('/content/drive')
```

Output:

```
Mounted at /content/drive
```

Code:

```
df = pd.read_csv('/content/drive/MyDrive/Entri - Main Project/Online retail and E-commerce 24,25.csv')
print(df)
```

Output:

```
     customer_id  order_date  product_id  category_id       category_name  \
0          13542  2024-12-17         784           10         Electronics
1          23188  2024-06-01         682           50   Sports & Outdoors
2          55098  2025-02-04         684           50   Sports & Outdoors
3          65208  2024-10-28         204           40  Books & Stationery
4          63872  2024-05-10         202           20             Fashion
..           ...         ...         ...          ...                 ...
995        67967  2024-05-04         965           40  Books & Stationery
996        99828  2024-09-12         510           40  Books & Stationery
997        92290  2024-11-06         445           10         Electronics
998        61427  2024-09-17         410           10         Electronics
999        20658  2024-11-06         177           40  Books & Stationery

    product_name  quantity   price    payment_method           city  \
0     Smartphone         2  373.36       Credit Card  New Oliviaberg
1    Soccer Ball         5  299.34       Credit Card    Port Matthew
2           Tent         5   23.00       Credit Card      West Sarah
3     Story Book         2  230.11     Bank Transfer  Hernandezburgh
4          Skirt         4  176.72       Credit Card    Jenkinshaven
..           ...       ...     ...               ...             ...
995     Notebook         3  495.24  Cash on Delivery      Hodgemouth
996   Story Book         5  427.73       Credit Card     Douglastown
997   Smartphone         5  354.64     Bank Transfer  New Amberville
998       Laptop         4  221.54  Cash on Delivery        New Sean
999          Pen         3  196.97  Cash on Delivery    North Kelsey

     review_score gender  age
0             1.0      F   56
1             NaN      M   59
2             5.0      F   64
3             5.0      M   34
4             1.0      F   33
..            ...    ...  ...
995           NaN    NaN   30
996           3.0      F   72
997           NaN      M   49
998           3.0      M   71
999           1.0      M   34

[1000 rows x 13 columns]
```

Code:

```
# Dataset description
print("Records: ",df.shape[0])
print("Columns: ",df.shape[1])
print("Feature: ",df.columns)
```

Output:

```
Records:  1000
Columns:  13
Feature:  Index(['customer_id', 'order_date', 'product_id', 'category_id',
       'category_name', 'product_name', 'quantity', 'price', 'payment_method',
       'city', 'review_score', 'gender', 'age'],
      dtype='object')
```

Code:

```
# Initial EDA
df.head() # first 5 records
```

Output:

	customer_id	order_date	product_id	category_id	category_name	product_name	quantity	price	payment_method	city	review_score	gender	age
0	13542	2024-12-17	784	10	Electronics	Smartphone	2	373.36	Credit Card	New Oliviaberg	1.0	F	56
1	23188	2024-06-01	682	50	Sports & Outdoors	Soccer Ball	5	299.34	Credit Card	Port Matthew	NaN	M	59
2	55098	2025-02-04	684	50	Sports & Outdoors	Tent	5	23.00	Credit Card	West Sarah	5.0	F	64
3	65208	2024-10-28	204	40	Books & Stationery	Story Book	2	230.11	Bank Transfer	Hernandezburgh	5.0	M	34
4	63872	2024-05-10	202	20	Fashion	Skirt	4	176.72	Credit Card	Jenkinshaven	1.0	F	33

Code:

```
# last 5 rows
df.tail()
```

Output:

	customer_id	order_date	product_id	category_id	category_name	product_name	quantity	price	payment_method	city	review_score	gender	age
995	67967	2024-05-04	965	40	Books & Stationery	Notebook	3	495.24	Cash on Delivery	Hodgemouth	NaN	NaN	30
996	99828	2024-09-12	510	40	Books & Stationery	Story Book	5	427.73	Credit Card	Douglastown	3.0	F	72
997	92290	2024-11-06	445	10	Electronics	Smartphone	5	354.64	Bank Transfer	New Amberville	NaN	M	49
998	61427	2024-09-17	410	10	Electronics	Laptop	4	221.54	Cash on Delivery	New Sean	3.0	M	71
999	20658	2024-11-06	177	40	Books & Stationery	Pen	3	196.97	Cash on Delivery	North Kelsey	1.0	M	34

Code:

```
#size - Total data count
print(df.size)
print(df.index)
```

Output:

```
20000
RangeIndex(start=0, stop=1000, step=1)
```

Code:

```
# info
df.info()
```

Output:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 13 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   customer_id     1000 non-null   int64  
 1   order_date      1000 non-null   object 
 2   product_id      1000 non-null   int64  
 3   category_id     1000 non-null   int64  
 4   category_name   1000 non-null   object 
 5   product_name    1000 non-null   object 
 6   quantity        1000 non-null   int64  
 7   price           1000 non-null   float64
 8   payment_method  1000 non-null   object 
 9   city            1000 non-null   object 
 10  review_score    799 non-null    float64
 11  gender          897 non-null    object 
 12  age             1000 non-null   int64  
dtypes: float64(2), int64(5), object(6)
memory usage: 101.7+ KB
```

Code:

```
df.dtypes
```

Output:

	0
customer_id	int64
order_date	object
product_id	int64
category_id	int64
category_name	object
product_name	object
quantity	int64
price	float64
payment_method	object
city	object
review_score	float64
gender	object
age	int64

dtype: object

Code:

```
# Discribe
df.describe()
```

Output:

	customer_id	product_id	category_id	quantity	price	review_score	age
count	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	799.000000	1000.000000
mean	55490.723000	540.726000	30.030000	2.947000	251.850660	3.992491	46.382000
std	25910.185857	261.737704	14.370303	1.413573	139.194688	1.239469	16.569992
min	10201.000000	100.000000	10.000000	1.000000	10.720000	1.000000	18.000000
25%	33857.000000	311.750000	20.000000	2.000000	128.525000	3.000000	32.000000
50%	54619.500000	542.500000	30.000000	3.000000	250.220000	4.000000	47.000000
75%	77848.500000	770.750000	40.000000	4.000000	366.467500	5.000000	61.000000
max	99923.000000	995.000000	50.000000	5.000000	499.500000	5.000000	75.000000

Code:

```
df.describe(include='object')
```

Output:

	order_date	category_name	product_name	payment_method	city	gender
count	1000	1000	1000	1000	1000	897
unique	342	5	25	3	962	2
top	2024-08-12	Sports & Outdoors	Yoga Mat	Cash on Delivery	Port Matthew	M
freq	10	211	51	374	3	457

Code:

```
# Shape

print("Shape (Rows, Columns)")
print(df.shape)
```

Output:

```
Shape (Rows, Columns)
(1000, 13)
```

Code:

```
# columnswise unique number of values coounts

df.nunique()
```

Output:

	0
customer_id	1000
order_date	342
product_id	605
category_id	5
category_name	5
product_name	25
quantity	5
price	991
payment_method	3
city	962
review_score	5
gender	2
age	58

dtype: int64

Code:

```
df.value_counts()
```

Output:

customer_id	order_date	product_id	category_id	category_name	product_name	quantity	price	payment_method	city	review_score	gender	age	count
99909	2024-08-11	233	10	Electronics	Tablet	1	474.06	Cash on Delivery	Sheltonmouth	5.0	F	37	1
98186	2024-08-25	606	50	Sports & Outdoors	Running Shoes	1	134.35	Bank Transfer	Lake Brianbury	4.0	F	47	1
98151	2024-05-14	707	50	Sports & Outdoors	Yoga Mat	2	75.60	Cash on Delivery	West Robertmouth	4.0	M	45	1
98074	2024-08-03	463	30	Home & Living	Vase	4	494.66	Cash on Delivery	East Jason	4.0	F	47	1
97829	2024-10-09	640	30	Home & Living	Pillow	5	143.03	Credit Card	Lake Alison	4.0	M	38	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...
10792	2024-04-21	898	50	Sports & Outdoors	Running Shoes	2	31.51	Credit Card	East Lisaview	5.0	F	49	1
10539	2025-03-17	142	40	Books & Stationery	Story Book	1	34.64	Bank Transfer	Anthonyport	3.0	F	44	1
10486	2024-09-13	297	20	Fashion	Shirt	2	274.10	Cash on Delivery	Port Allisonfort	4.0	F	54	1
10299	2024-11-27	971	20	Fashion	Dress	4	203.94	Cash on Delivery	Teresaville	5.0	F	33	1
10254	2024-09-10	318	30	Home & Living	Vase	1	70.93	Cash on Delivery	Port Christine	3.0	M	73	1

721 rows × 1 columns

dtype: int64

Code:

```
# Null checks

print("--- Null Checks (Missing Values) ---")
print(df.isnull().sum().to_markdown(numalign="left", stralign="left"))
```

Output:

```
--- Null Checks (Missing Values) ---
|                | 0   |
|:---------------|:----|
| customer_id    | 0   |
| order_date     | 0   |
| product_id     | 0   |
| category_id    | 0   |
| category_name  | 0   |
| product_name   | 0   |
| quantity       | 0   |
| price          | 0   |
| payment_method | 0   |
| city           | 0   |
| review_score   | 201 |
| gender         | 103 |
| age            | 0   |
```

Stage 2 – Data Cleaning and Pre-processing

- Handle missing values (impute or drop)
- Handle duplicates
- Treat outliers if required
- Check skewness and apply transformations
- Convert data types if needed
- Feature transformations (date parts, derived fields if required for analysis)

Code:

```
# Impute Missing values
df['review_score'] = df['review_score'].fillna(df['review_score'].median())
df['gender'] = df['gender'].fillna(df['gender'].mode()[0])
# Check for missing values in the entire dataset
missing_values = df.isnull().sum()
missing_values
```

Output:

	0
customer_id	0
order_date	0
product_id	0
category_id	0
category_name	0
product_name	0
quantity	0
price	0
payment_method	0
city	0
review_score	0
gender	0
age	0

dtype: int64

Code:

```
# Outlier Treatment (IQR Method)

num_cols = df.select_dtypes(include=['int64','float64']).columns

for col in num_cols:
    Q1 = df[col].quantile(0.25)
    Q3 = df[col].quantile(0.75)
    IQR = Q3 - Q1
    lower = Q1 - 1.5 * IQR
    upper = Q3 + 1.5 * IQR
    df[col] = np.where(df[col] < lower, lower,
                       np.where(df[col] > upper, upper, df[col]))

print("\nOutlier treatment done!!.")
```

Output:

```
Outlier treatment done!!.
```

Code:

```
df.dtypes
```

Output:

	0
customer_id	float64
order_date	object
product_id	float64
category_id	float64
category_name	object
product_name	object
quantity	float64
price	float64
payment_method	object
city	object
review_score	float64
gender	object
age	float64

dtype: object

Code:

```
# data type conversion

df['customer_id'] = df['customer_id'].astype(int)
df['product_id'] = df['product_id'].astype(int)
df['category_id'] = df['category_id'].astype(int)
df['order_date'] = pd.to_datetime(df['order_date'])
df['age'] = df['age'].astype(int)

print("\nUpdated Data Types:\n", df.dtypes)
```

Output:

```
Updated Data Types:
 customer_id                int64
order_date        datetime64[ns]
product_id                 int64
category_id                int64
category_name             object
product_name              object
quantity                 float64
price                    float64
payment_method            object
city                      object
review_score             float64
gender                    object
age                        int64
dtype: object
```

Code:

```
df.head()
```

Output:

	customer_id	order_date	product_id	category_id	category_name	product_name	quantity	price	payment_method	city	review_score	gender	age
0	13542	2024-12-17	784	10	Electronics	Smartphone	2.0	373.36	Credit Card	New Oliviaberg	2.5	F	56
1	23188	2024-06-01	682	50	Sports & Outdoors	Soccer Ball	5.0	299.34	Credit Card	Port Matthew	4.0	M	59
2	55098	2025-02-04	684	50	Sports & Outdoors	Tent	5.0	23.00	Credit Card	West Sarah	5.0	F	64
3	65208	2024-10-28	204	40	Books & Stationery	Story Book	2.0	230.11	Bank Transfer	Hernandezburgh	5.0	M	34
4	63872	2024-05-10	202	20	Fashion	Skirt	4.0	176.72	Credit Card	Jenkinshaven	2.5	F	33

Code:

```
# Feature Engineering (Date Parts)
df['Year'] = df['order_date'].dt.year
df['Month'] = df['order_date'].dt.month
df['Day'] = df['order_date'].dt.day
df['Weekday'] = df['order_date'].dt.day_name()
print(df)
```

Output:

	customer_id	order_date	product_id	category_id	category_name	product_name	quantity	price	payment_method	city	review_score	gender	age	Year	Month	Day	Weekday
0	13542	2024-12-17	784	10	Electronics	Smartphone	2.0	373.36	Credit Card	New Oliviaberg	2.5	F	56	2024	12	17	Tuesday
1	23188	2024-06-01	682	50	Sports & Outdoors	Soccer Ball	5.0	299.34	Credit Card	Port Matthew	4.0	M	59	2024	6	1	Saturday
2	55098	2025-02-04	684	50	Sports & Outdoors	Tent	5.0	23.00	Credit Card	West Sarah	5.0	F	64	2025	2	4	Tuesday
3	65208	2024-10-28	204	40	Books & Stationery	Story Book	2.0	230.11	Bank Transfer	Hernandezburgh	5.0	M	34	2024	10	28	Monday
4	63872	2024-05-10	202	20	Fashion	Skirt	4.0	176.72	Credit Card	Jenkinshaven	2.5	F	33	2024	5	10	Friday
..	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
995	67967	2024-05-04	965	40	Books & Stationery	Notebook	3.0	495.24	Cash on Delivery	Hodgemouth	4.0	M	30	2024	5	4	Saturday
996	99828	2024-09-12	510	40	Books & Stationery	Story Book	5.0	427.73	Credit Card	Douglastown	3.0	F	72	2024	9	12	Thursday
997	92290	2024-11-06	445	10	Electronics	Smartphone	5.0	354.64	Bank Transfer	New Amberville	4.0	M	49	2024	11	6	Wednesday
998	61427	2024-09-17	410	10	Electronics	Laptop	4.0	221.54	Cash on Delivery	New Sean	3.0	M	71	2024	9	17	Tuesday
999	20658	2024-11-06	177	40	Books & Stationery	Pen	3.0	196.97	Cash on Delivery	North Kelsey	2.5	M	34	2024	11	6	Wednesday

[1000 rows x 17 columns]

Code:

```
#count of unique value
df['Weekday'].value_counts()
```

Output:

Weekday	count
Friday	150
Thursday	148
Tuesday	148
Wednesday	146
Monday	142
Saturday	136
Sunday	130

dtype: int64

Code:

```
df['review_score'].value_counts()
```

Output:

review_score	count
4.0	399
5.0	382
2.5	110
3.0	109

dtype: int64

Code:

```
def categorize_rating(score):
    if score >= 4.5:
        return 'Excellent'
    elif score >= 3:
        return 'Good'
    elif score >= 2:
        return 'Average'
    else:
        return 'Poor'

df['Rating_Category'] = df['review_score'].apply(categorize_rating)

print(df[['review_score', 'Rating_Category']])
```

Output:

	review_score	Rating_Category
0	2.5	Average
1	4.0	Good
2	5.0	Excellent
3	5.0	Excellent
4	2.5	Average
..	...	...
995	4.0	Good
996	3.0	Good
997	4.0	Good
998	3.0	Good
999	2.5	Average

[1000 rows x 2 columns]

Code:

```
df['age'].value_counts()
```

Output:

age	count
39	24
63	24
48	23
68	23
34	22
44	22
43	22
61	22
26	21
50	21
65	21
23	21
24	20
64	20
29	20
56	20

Code:

```
def age_group(age):
    if 18 <= age <= 30:
        return 'Young Adult'
    elif 31 <= age <= 58:
        return 'Adult'
    elif 59 <= age <= 75:
        return 'Senior'
    else:
        return 'Out of Range'
df['Age_Group'] = df['age'].apply(age_group)
print(df[['age', 'Age_Group']])
```

Output:

```
      age    Age_Group
0      56        Adult
1      59       Senior
2      64       Senior
3      34        Adult
4      33        Adult
..    ...          ...
995    30  Young Adult
996    72       Senior
997    49        Adult
998    71       Senior
999    34        Adult

[1000 rows x 2 columns]
```

Code:

```
# Skewness check

print("\nSkewness of numerical columns:\n", df[num_cols].skew())
```

Output:

```
Skewness of numerical columns:
 customer_id     0.014653
product_id       0.010938
category_id      0.002850
quantity         0.061980
price            0.038536
review_score    -0.541437
age             -0.016843
dtype: float64
```

Code:

```
df.info()
```

Output:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 19 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   customer_id      1000 non-null   int64
 1   order_date       1000 non-null   datetime64[ns]
 2   product_id       1000 non-null   int64
 3   category_id      1000 non-null   int64
 4   category_name    1000 non-null   object
 5   product_name     1000 non-null   object
 6   quantity         1000 non-null   float64
 7   price            1000 non-null   float64
 8   payment_method   1000 non-null   object
 9   city             1000 non-null   object
 10  review_score     1000 non-null   float64
 11  gender           1000 non-null   object
 12  age              1000 non-null   int64
 13  Year             1000 non-null   int32
 14  Month            1000 non-null   int32
 15  Day              1000 non-null   int32
 16  Weekday          1000 non-null   object
 17  Rating_Category  1000 non-null   object
 18  Age_Group        1000 non-null   object
dtypes: datetime64[ns](1), float64(3), int32(3), int64(4), object(8)
memory usage: 136.8+ KB
```

Stage 3 – EDA and Visualizations

- Univariate Analysis → distribution of single variables (countplot, histogram, boxplot)
- Bivariate Analysis → relation between two variables (scatterplot, barplot, correlation heatmap)
- Multivariate Analysis → relation among 3+ variables (pairplot, grouped analysis, pivot tables, advanced plots)
- Interpretation MUST with every visualization
- Focus on business story not just charts
- Each chart must be in seperate cells.

Code:

```
from google.colab import files

df.to_csv("cleaned_dataset.csv", index=False)
files.download("cleaned_dataset.csv")
print(f"\n cleaned-dataset downloaded successfully!!")
```

Output:

```
cleaned-dataset downloaded successfully!!
```

Code:

```
df.columns
```

Output:

```
Index(['customer_id', 'order_date', 'product_id', 'category_id',
       'category_name', 'product_name', 'quantity', 'price', 'payment_method',
       'city', 'review_score', 'gender', 'age', 'Year', 'Month', 'Day',
       'Weekday', 'Rating_Category', 'Age_Group'],
      dtype='object')
```

1. SALES ANALYSIS CHARTS

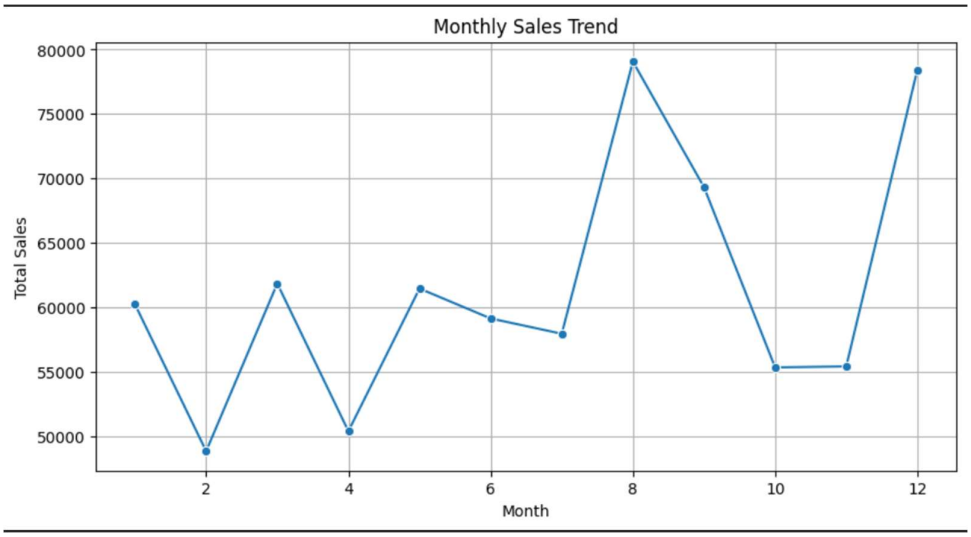
(Uses: quantity, price, Month, Day, Weekday, city, payment_method)

Code:

```
# Total Sales by Month (Line Chart)

df['Sales'] = df['quantity'] * df['price']
monthly_sales = df.groupby('Month')['Sales'].sum().reset_index()
plt.figure(figsize=(10,5))
```

```
sns.lineplot(data=monthly_sales, x='Month', y='Sales', marker='o')
plt.title("Monthly Sales Trend")
plt.xlabel("Month")
plt.ylabel("Total Sales")
plt.grid(True)
plt.show()
```

Output:



Interpretation of your above chart

Explain the Chart

- This chart shows the total sales for each month plotted as a line.

- The line connects monthly sales values and shows how sales increased or decreased across the months.

What is it saying?

- It is showing the sales trend over time.

- Which months had high sales

- Which months had low sales

- Whether sales are increasing, decreasing, or fluctuating

- It tells the overall growth or decline pattern across the year.

What features you used?

- Month → As the time period (x-axis)

- Sales → Total sales amount (y-axis)

- quantity × price → To calculate actual sales

- groupby('Month') → To get monthly total sales

- Line Chart → To show trend over time

- marker='o' → To highlight each month's data point

From this what you are showing us?

- How sales change from one month to another

- The overall trend (upward / downward / seasonal)

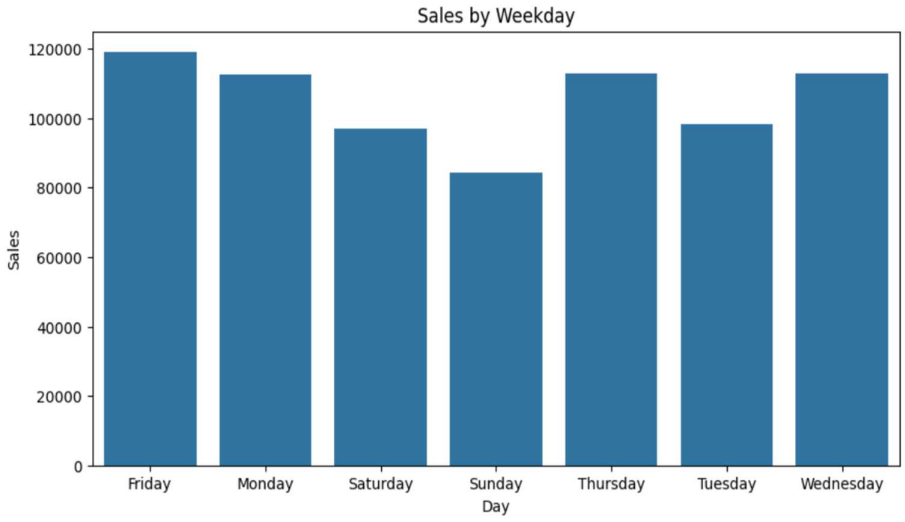
- The best-performing and worst-performing months

- This helps understand when sales peak and when they dip, useful for planning inventory, marketing, and seasonal strategies.

Code:

```
# Sales by Weekday (Bar Chart)

weekday_sales = df.groupby('Weekday')['Sales'].sum().reset_index()
plt.figure(figsize=(10,5))
sns.barplot(data=weekday_sales, x='Weekday', y='Sales')
plt.title("Sales by Weekday")
plt.xlabel("Day")
plt.ylabel("Sales")
plt.show()
```

Output:



Interpretation of your above chart

Explain the Chart

- This chart shows the total sales made on each weekday.

- Every bar represents how much sales happened on that specific day.

What is it saying?

- It shows which days have high sales and which days have low sales.

- You can easily see the best-performing day (tallest bar) and the slowest day (shortest bar).

What features you used?

- Weekday → Used as the category (x-axis)

- Sales → Total sales amount for each day (y-axis)

- Groupby() → To calculate total sales per weekday

- Bar Chart → To visually compare sales across days

From this what you are showing us?

- The sales pattern across the week

- Which day performs the best

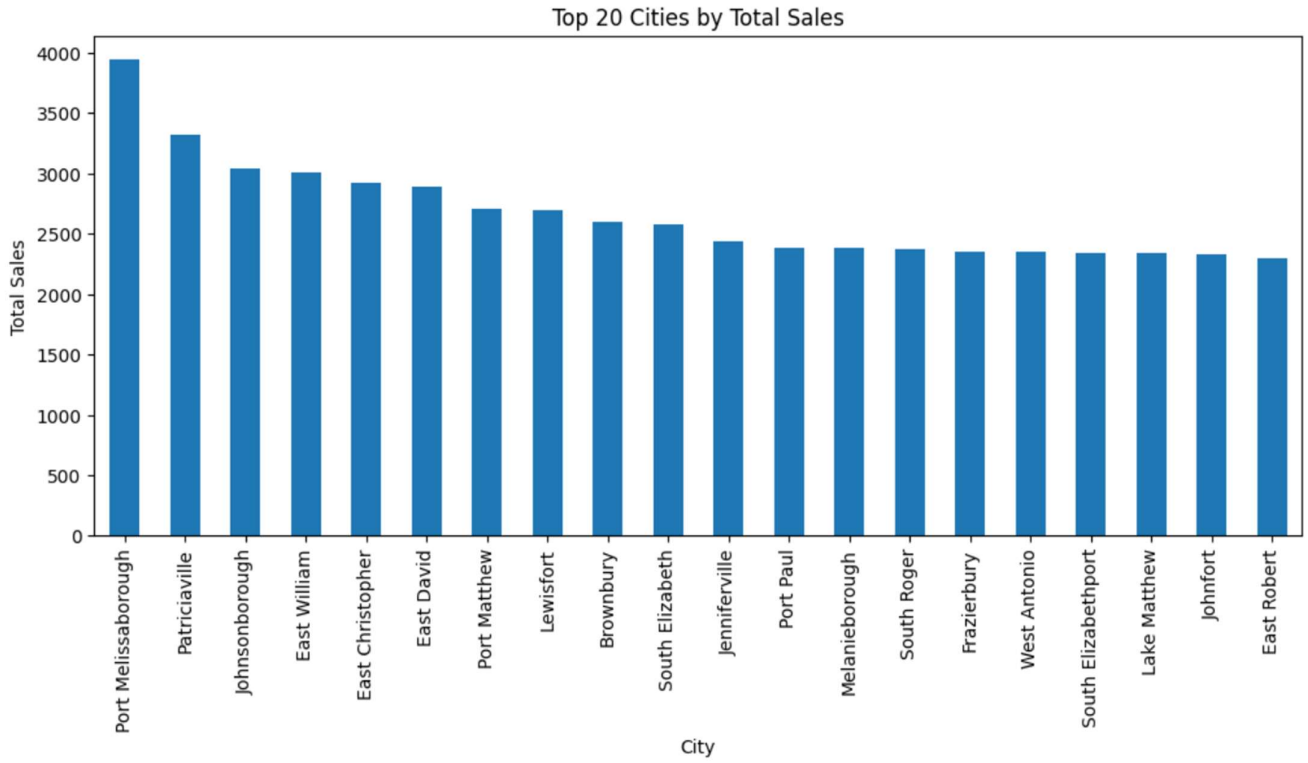
- Which day needs improvement

- The overall weekly trend in customer activity

Code:

```
# Sales by Top 20 City (Bar Chart)

city_sales = df.groupby('city')['Sales'].sum().sort_values(ascending=False)

plt.figure(figsize=(12,5))
city_sales.head(20).plot(kind='bar')
plt.title("Top 20 Cities by Total Sales")
plt.xlabel("City")
plt.ylabel("Total Sales")
plt.xticks(rotation=90)
plt.show()
```

Output:



Interpretation of your above chart

Explain the Chart

- This bar chart shows the total sales for each city.

- Only the top 20 cities with the highest total sales are displayed.

- Each bar represents a city, and its height shows how much total revenue that city generated.

What is it saying?

- Which cities are generating the most revenue

- How sales compare across different cities

- Key markets contributing most to the business

What features you used?

- city → To group orders

- Sales → Total sales per city

- groupby() + sum() → To calculate total sales

- sort_values(ascending=False) → To get the top 20 cities

- Bar chart → To visually compare total sales

- xticks rotation → To make city names readable

From this what you are showing us?

- The cities with the highest total revenue

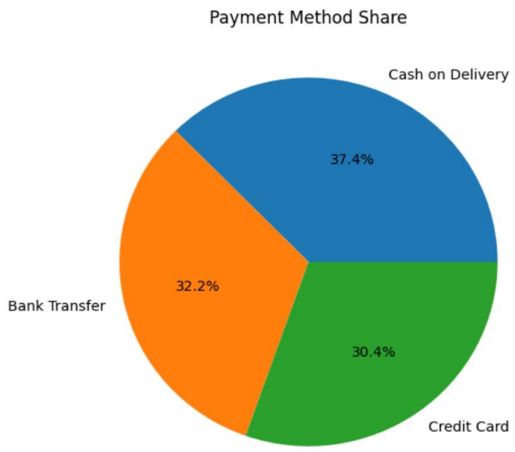
- Potential key markets for focus

- Insights for sales strategy, marketing, and resource planning

Code:

```
# Payment Method Distribution (Pie Chart)

payment_counts = df['payment_method'].value_counts()

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

payment_counts.plot.pie(autopct="%1.1f%%")

plt.title("Payment Method Share")

plt.ylabel("")

plt.show()
```

Output:



Interpretation of your above chart

Explain the Chart

- This pie chart shows the percentage share of each payment method used by customers.

- Each slice represents how often a particular payment method was used.

What is it saying?

- It tells us which payment method is most preferred and which ones are less frequently used.

- The bigger the slice, the more customers use that payment method.

What features you used?

- payment_method column → To identify types of payments

- value_counts() → To count how many times each method was used

- Pie chart → To represent share/percentage

- autopct → To show the percentage on the chart

From this what you are showing us?

- The distribution of payment methods

- The most popular and least popular payment types

- Customer payment preference trends

- This helps understand which payment option customers choose most often.

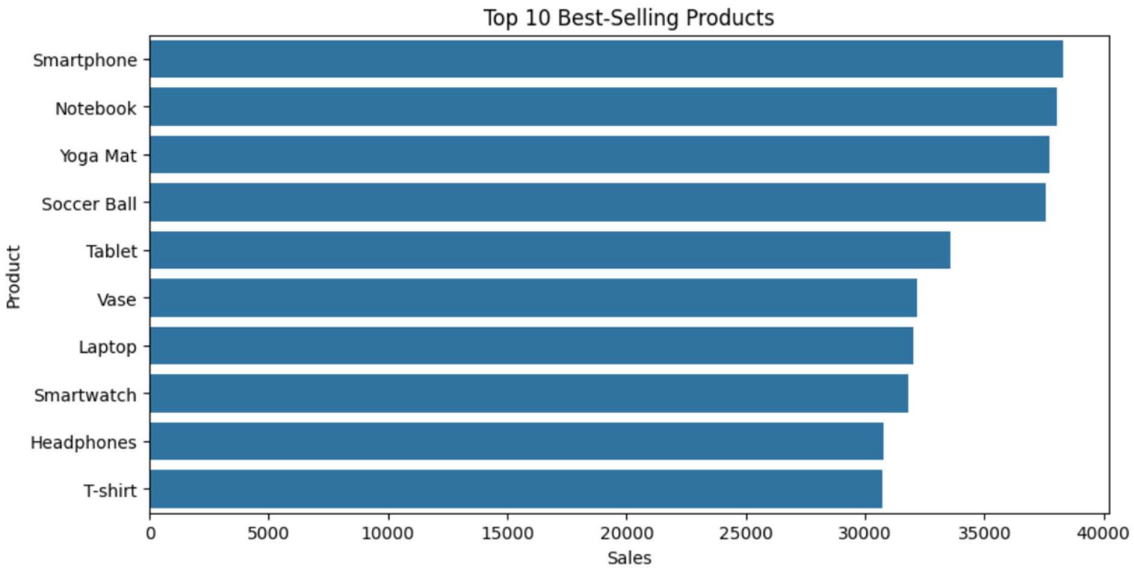
2. PRODUCT ANALYSIS CHARTS

(Uses: product_id, product_name, category_name)

Code:

```
# Top 10 Best-selling Products (Bar Chart)

product_sales =
df.groupby('product_name')['Sales'].sum().nlargest(10).reset_index()

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

sns.barplot(data=product_sales, x='Sales', y='product_name')

plt.title("Top 10 Best-Selling Products")

plt.xlabel("Sales")

plt.ylabel("Product")

plt.show()
```

Output:



Interpretation of your above chart

Explain the Chart

- This bar chart shows the top 10 products with the highest total sales.

- Each bar represents one product, and the length of the bar shows how much sales that product generated.

What is it saying?

- Which products are the best performers

- Which products generate the most revenue

- The ranking of the top 10 products based on total sales

- The product at the top of the chart is the number one best-selling item.

What features you used?

- product_name → To group products

- Sales → Total sales value

- groupby() + sum() → To calculate total sales per product

- nlargest(10) → To pick the top 10 highest-selling products

- Bar chart → To compare product sales visually

From this what you are showing us?

- The top 10 highest revenue-generating products

- Which products customers buy the most

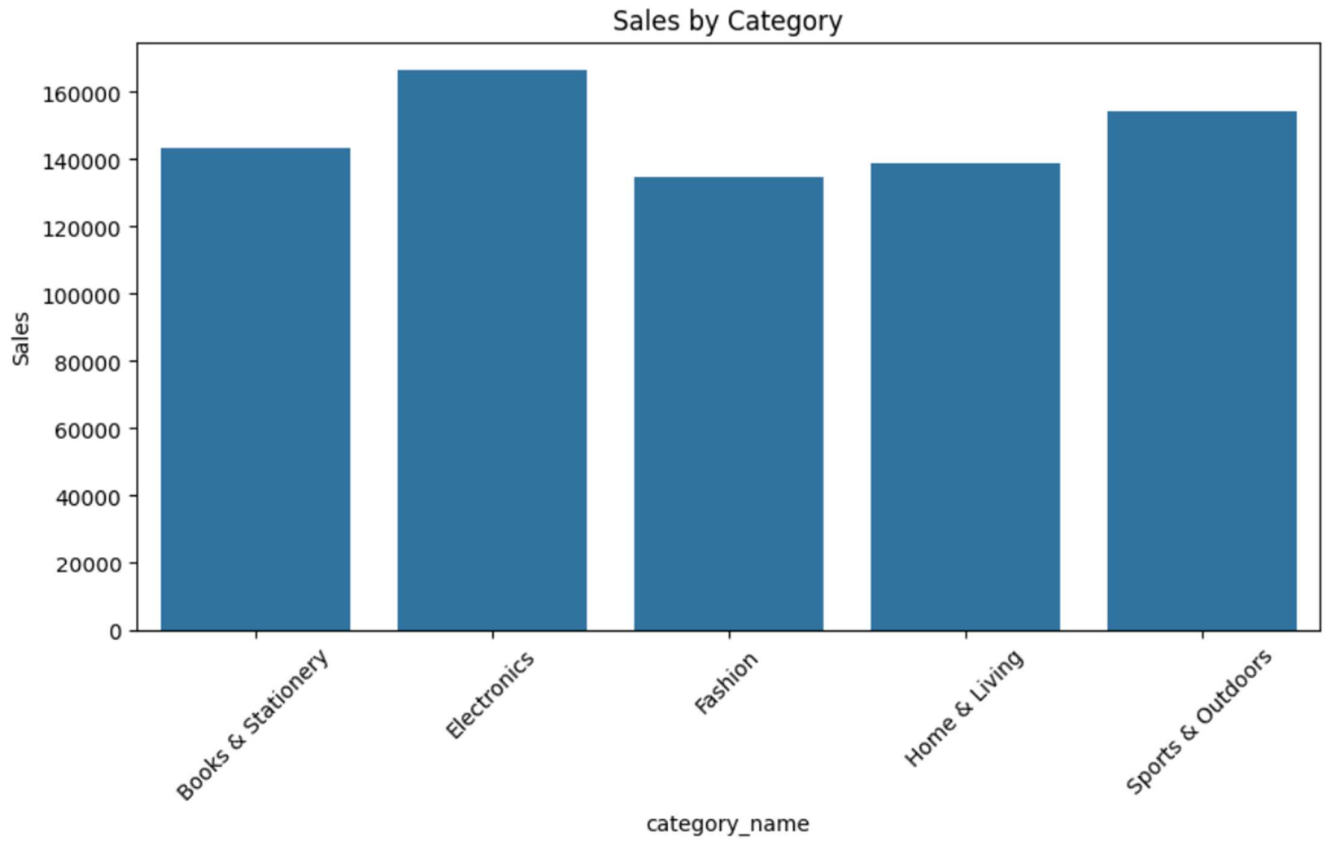
- Which items contribute the most to the company's sales

- Useful insights for inventory, marketing, and product strategy

Code:

```
# Category-wise Sales Contribution (Bar Chart)

category_sales = df.groupby('category_name')['Sales'].sum().reset_index()

plt.figure(figsize=(10,5))
sns.barplot(data=category_sales, x='category_name', y='Sales')
plt.xticks(rotation=45)
plt.title("Sales by Category")
plt.show()
```

Output:



Interpretation of your above chart

Explain the Chart

- This bar chart shows the total sales for each product category.

- Each bar represents a category, and the height shows how much sales that category generated.

What is it saying?

- Which categories are top performers

- Which categories bring high or low revenue

- How sales are distributed across different product types

- The tallest bar represents the highest-selling category.

What features you used?

- category_name → To identify product categories

- Sales → Total sales amount

- groupby() + sum() → To calculate total sales per category

- Bar chart → To compare sales across categories

- xticks rotation → To make category names readable

From this what you are showing us?

- The sales performance of each category

- Which categories bring the most revenue

- Insights to help in inventory planning, marketing, and business focus

3. CUSTOMER ANALYSIS CHARTS

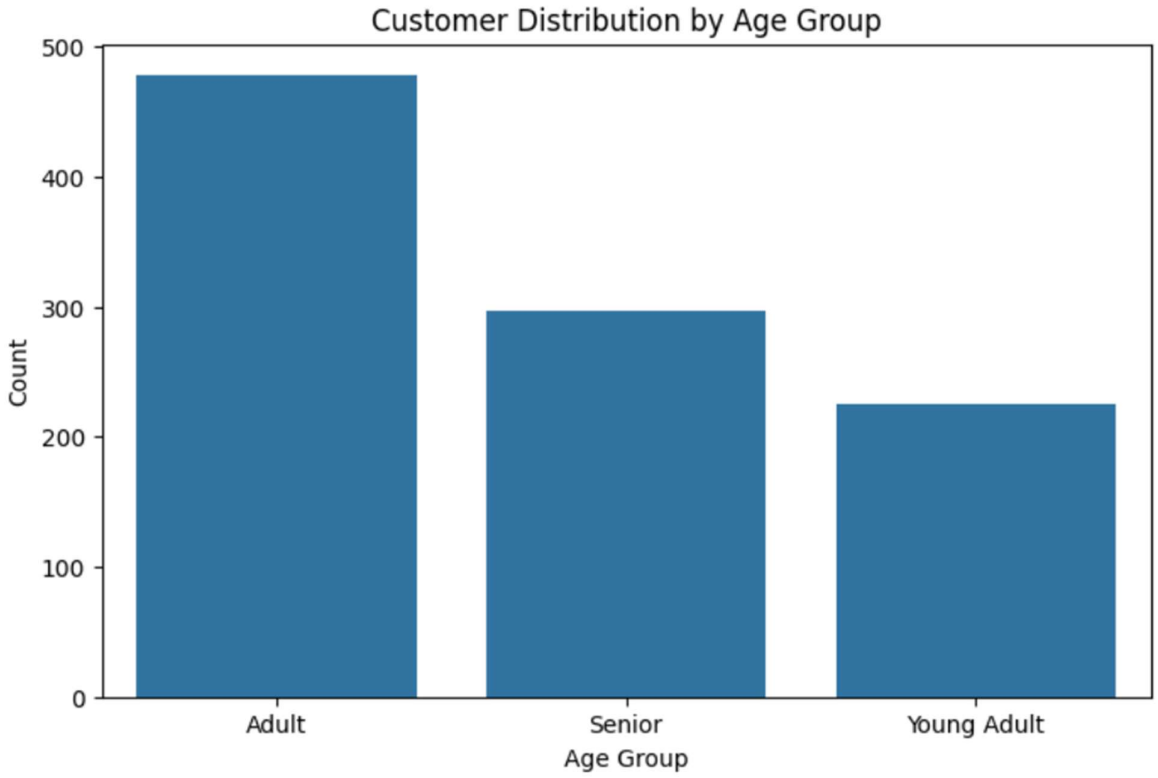
(Uses: customer_id, gender, age, Age_Group, city)

Code:

```
# Customer Count by Age Group (Bar Chart)

plt.figure(figsize=(8,5))
sns.countplot(data=df, x='Age_Group')
plt.title("Customer Distribution by Age Group")
plt.xlabel("Age Group")
```

```
plt.ylabel("Count")
plt.show()
```

Output:



Interpretation of your above chart

Explain the Chart

- This bar chart shows how many customers belong to each Age Group category.

- Each bar represents the count of customers in that age group.

What is it saying?

- Which age group has the highest number of customers

- Which age group is least represented

- The overall age distribution pattern of your customers

- This helps identify your main customer demographic.

What features you used?

- Age_Group → Categorized age ranges (e.g., 18–25, 26–35, etc.)
- Countplot → To show how many customers fall into each group
- Figure size & labels → To make the chart readable

From this what you are showing us?

- The customer count per age group

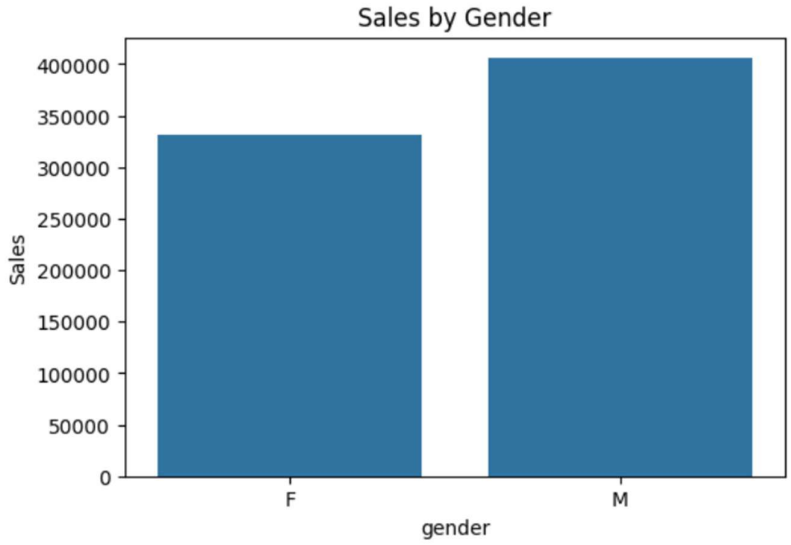
- Which age groups dominate your customer base

- Useful insights for targeted marketing and product planning

Code:

```
# Sales by Gender (Bar Chart)

gender_sales = df.groupby('gender')['Sales'].sum().reset_index()
plt.figure(figsize=(6,4))
sns.barplot(data=gender_sales, x='gender', y='Sales')
plt.title("Sales by Gender")
plt.show()
```

Output:



Interpretation of your above chart

Explain the Chart

- This bar chart shows the total sales for each gender.

- Each bar represents how much revenue was generated by male or female customers.

What is it saying?

- Which gender contributes more to total sales

- The comparison between male and female customer purchases

What features you used?

- gender → To group customers

- Sales → Total sales value

- groupby() + sum() → To calculate total sales per gender

- Bar chart → To visually compare sales

From this what you are showing us?

- Sales distribution across genders

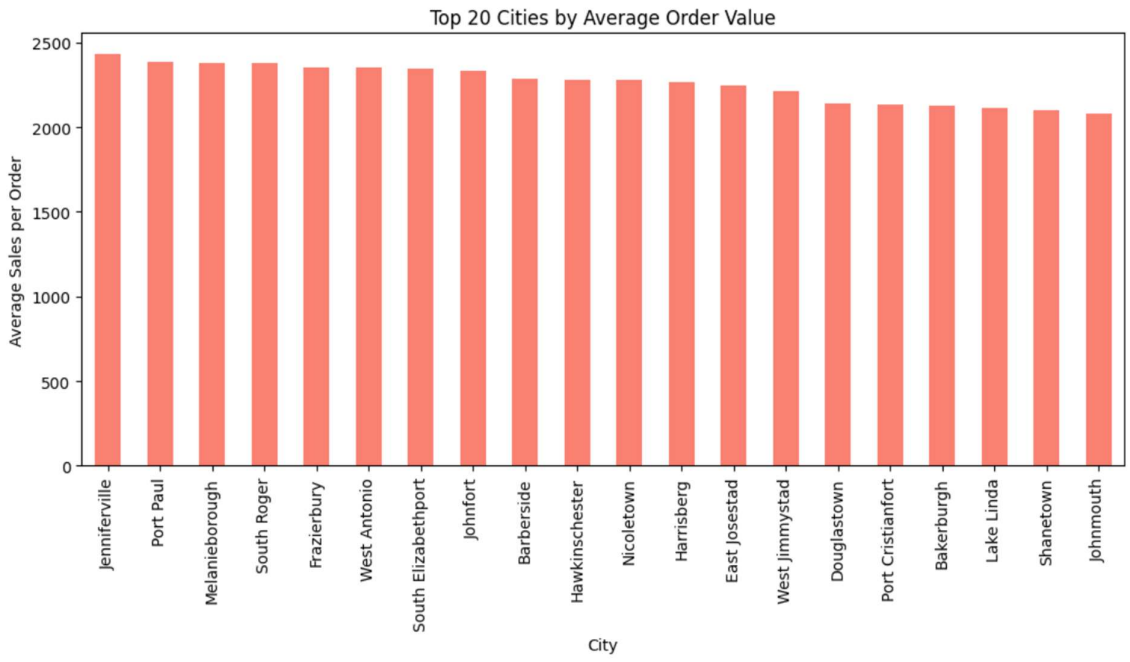
- Which gender is the more profitable customer segment

- Helps in marketing and targeting strategies

Code:

```
# Average Order Value by City (Bar Chart)
avg_sales_city = df.groupby('city')['Sales'].mean().sort_values(ascending=False)

plt.figure(figsize=(12,5))
avg_sales_city.head(20).plot(kind='bar', color='salmon')
plt.title("Top 20 Cities by Average Order Value")
plt.xlabel("City")
plt.ylabel("Average Sales per Order")
plt.xticks(rotation=90)
plt.show()
```

Output:



Interpretation of your above chart

Explain the Chart

- This bar chart shows the average sales per order for each city.

- Only the top 20 cities with the highest average order value are displayed.

- Each bar represents a city, and the height shows its average sales per order.

What is it saying?

- Which cities have the highest spending per order

- How average order value varies across different cities

- Where customers are spending more on average

What features you used?

- city → To group orders

- Sales → Total sales per order

- groupby() + mean() → To calculate average order value per city

- sort_values(ascending=False) → To get the highest spending cities first

- Bar chart → To visually compare average sales

- xticks rotation → To make city names readable

From this what you are showing us?

- Cities with the highest average order value

- Potential target markets for premium products

- Insights for marketing, promotions, and resource allocation

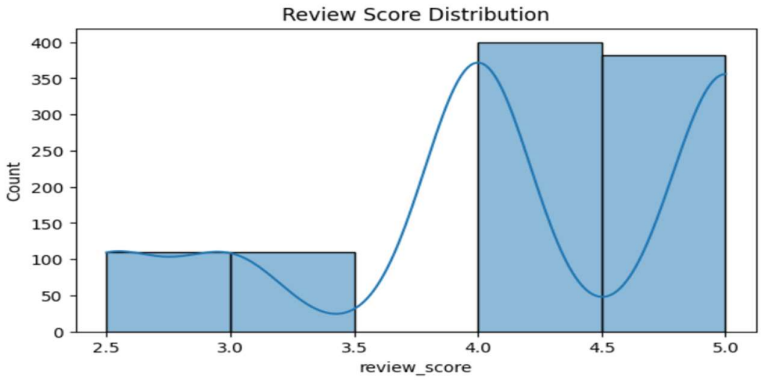
4. REVIEW & RATING ANALYSIS

(Uses: review_score, Rating_Category)

Code:

```
# Review Score Distribution (Histogram)
plt.figure(figsize=(7,4))
sns.histplot(df['review_score'], bins=5, kde=True)
plt.title("Review Score Distribution")
plt.show()
```

Output:



Interpretation of your above chart

Explain the Chart

- This histogram shows how customer review scores are distributed.

- The x-axis shows review scores (e.g., 1–5), and the y-axis shows how many reviews fall into each score.

- The KDE curve smooths the distribution to show the overall trend.

What is it saying?

- Which review scores are most common

- Whether customers generally give high or low ratings

- The overall pattern of customer satisfaction

What features you used?

- review_score → Customer ratings

- Histogram → To show frequency distribution

- bins=5 → To group scores into 5 ranges (1,2,3,4,5)

- kde=True → To add a smooth density curve

From this what you are showing us?

- How customers rate products overall

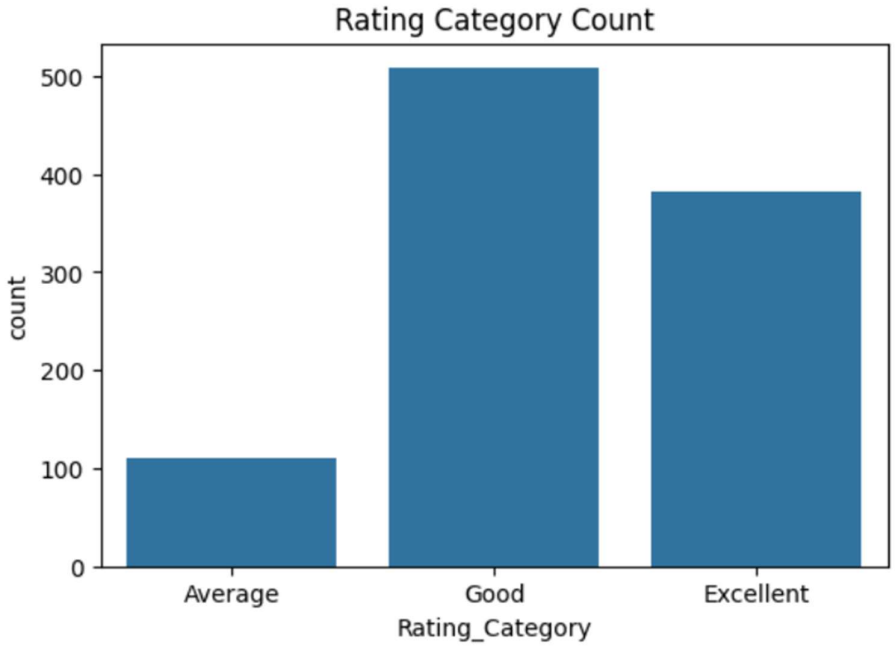
- Whether the ratings are skewed toward high or low scores

- Useful insights for product improvement and customer satisfaction analysis

Code:

```
# Rating Category vs Count (Bar Chart)

plt.figure(figsize=(6,4))
sns.countplot(data=df, x='Rating_Category')
plt.title("Rating Category Count")
plt.show()
```

Output:



Interpretation of your above chart

Explain the Chart

- This bar chart shows the number of reviews in each rating category (e.g., Good, Bad, Excellent).

- Each bar represents how many reviews fall into that category.

What is it saying?

- Which rating category is most common

- Which category is least common

- The overall customer satisfaction distribution

What features you used?

- Rating_Category → Categorized review scores

- Countplot → To show the frequency of each category

- Figure size & labels → To make the chart readable

From this what you are showing us?

- The distribution of review categories

- Which rating (Good, Bad, Excellent) is dominant

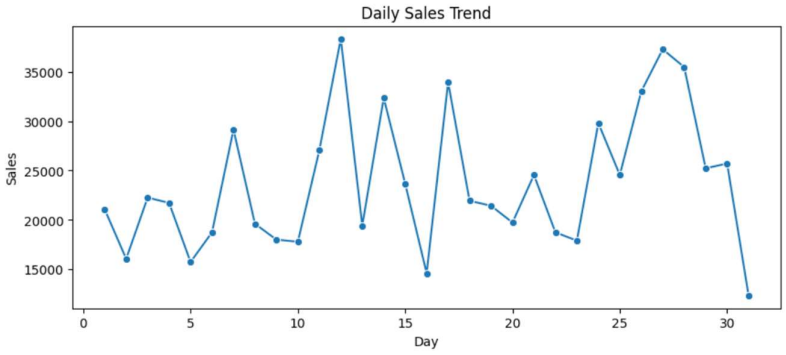
- Insights for product quality perception and improvement

5. TIME-BASED ANALYSIS

Code:

```
# Daily Sales Trend (Line Chart)

daily_sales = df.groupby('Day')['Sales'].sum().reset_index()

plt.figure(figsize=(10,4))
sns.lineplot(data=daily_sales, x='Day', y='Sales', marker='o')
plt.title("Daily Sales Trend")
plt.show()
```

Output:



Interpretation of your above chart

Explain the Chart

- This line chart shows the total sales for each day.

- Each point represents total sales for that day, and the line shows the trend across days.

What is it saying?

- How sales change day by day

- Which days have high sales and which have low sales

- The overall daily sales pattern or trend

What features you used?

- Day → Day of the week or specific date (x-axis)

- Sales → Total sales per day (y-axis)

- groupby() + sum() → To calculate total daily sales

- Line chart with marker → To show trend and highlight daily values

From this what you are showing us?

- The day-to-day variation in sales

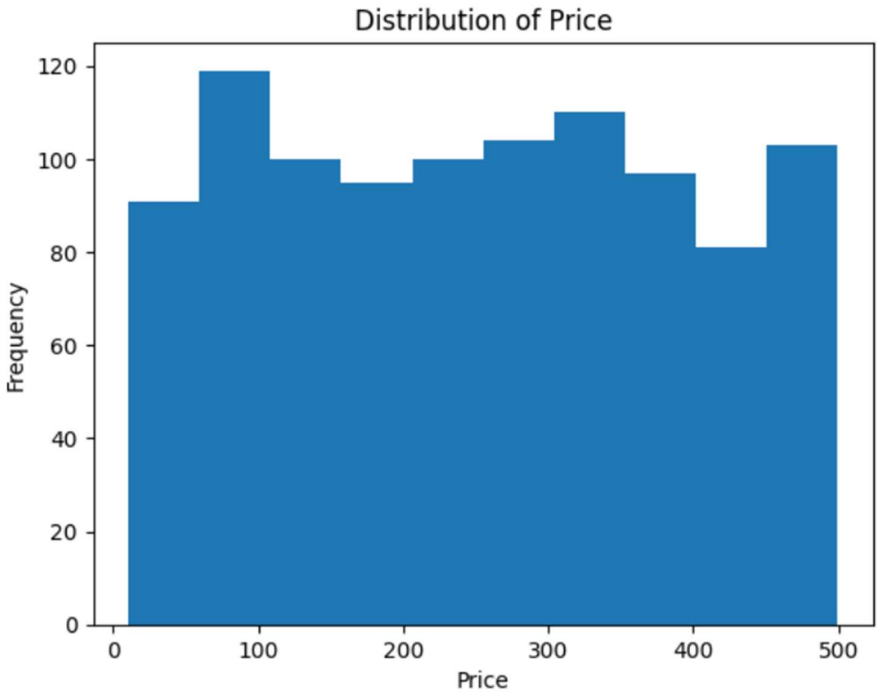
- Days with peak or low sales

- Useful insights for daily planning, staffing, and promotions

Code:

```
# Histogram – Price Distribution
plt.hist(df['price'])
plt.title("Distribution of Price")
plt.xlabel("Price")
plt.ylabel("Frequency")
plt.show()
```

Output:



Interpretation of your above chart

Explain the Chart

- This histogram shows how product prices are distributed in your dataset.

- The x-axis represents price ranges, and the y-axis shows how many products fall into each range.

What is it saying?

- Which price ranges are most common

- Whether products are mostly low-priced, mid-priced, or high-priced

- The overall pricing pattern in your dataset

What features you used?

- price → Product price values

- Histogram → To show frequency distribution

- xlabel & ylabel → To label axes

- title → To describe the chart

From this what you are showing us?

- How product prices are spread

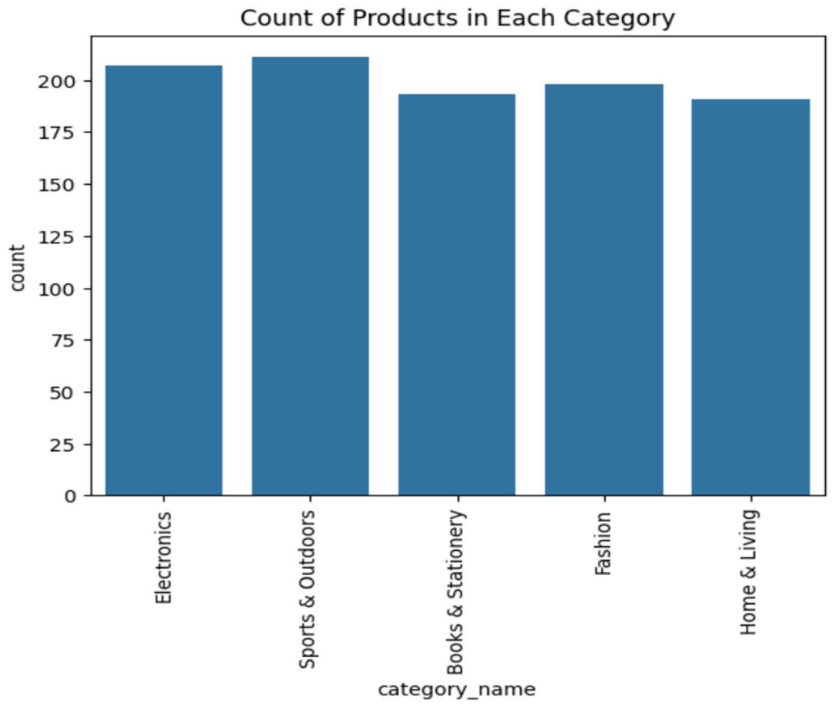
- The most frequent price ranges

- Insights for pricing strategy and product positioning

Code:

```
# Countplot – Category Distribution
sns.countplot(x='category_name', data=df)

plt.xticks(rotation=90)

plt.title("Count of Products in Each Category")

plt.show()
```

Output:



Interpretation of your above chart

Explain the Chart

- This bar chart shows the number of products in each category.

- Each bar represents a category, and its height shows how many products belong to that category.

What is it saying?

- Which categories have more products available

- Which categories have fewer products

- The overall product distribution across categories

What features you used?

- category_name → To group products by category

- Countplot → To show the frequency of products in each category

- xticks rotation → To make category names readable

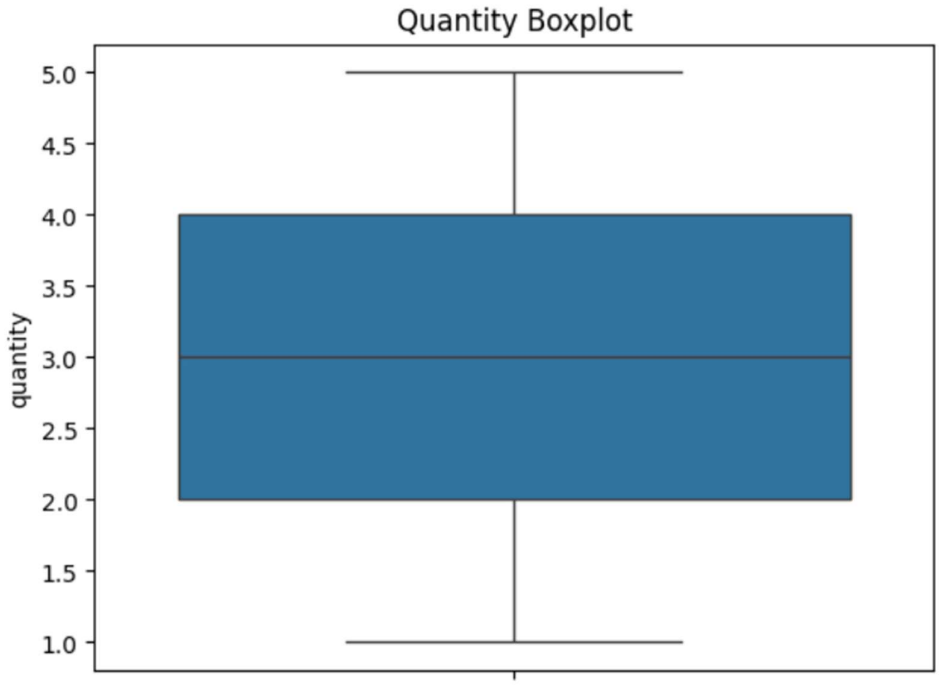
From this what you are showing us?

- The distribution of products across categories
- Insights into inventory focus and category diversity

Code:

```
# Boxplot – Quantity Outliers
sns.boxplot(y=df['quantity'])
plt.title("Quantity Boxplot")
plt.show()
```

Output:



Interpretation of your above chart

Explain the Chart

- This boxplot shows the distribution of the quantity of products in orders.

- It visualizes the median, quartiles, and possible outliers in the data.

What is it saying?

- The typical range of quantity ordered (interquartile range)

- The median quantity

- Any outliers (orders with unusually high or low quantities)

What features you used?

- quantity → Number of items in each order

- Boxplot → To show distribution, median, quartiles, and outliers

From this what you are showing us?

- How quantities vary across orders

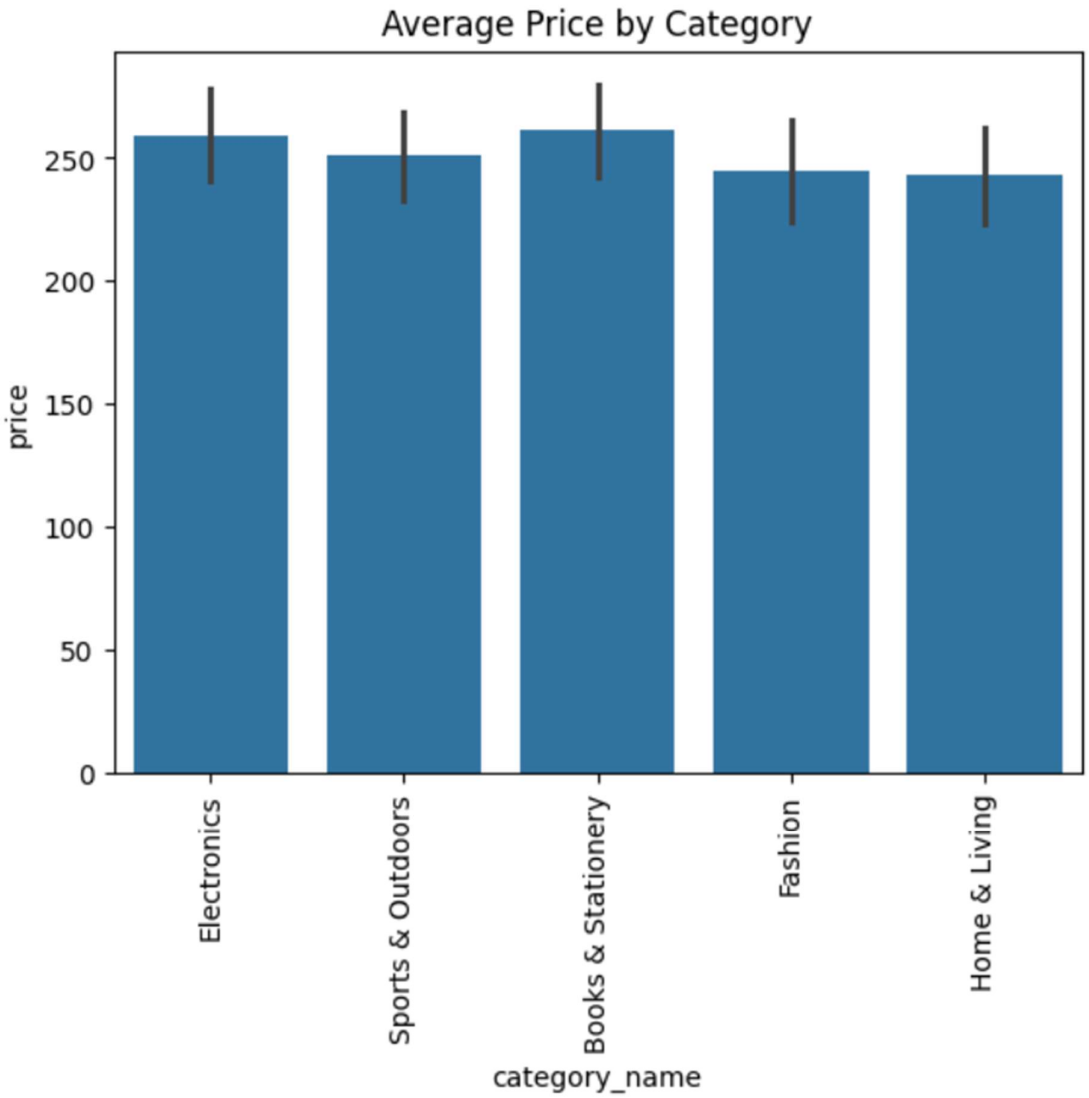
- The usual order size

- Potential unusual large or small orders that may need attention

Code:

```
## BIVARIATE ANALYSIS (Two variables)

# Barplot — Category vs Average Price
sns.barplot(x='category_name', y='price', data=df)
plt.xticks(rotation=90)
plt.title("Average Price by Category")
plt.show()
```

Output:



Interpretation of your above chart

Explain the Chart

- This bar chart shows the average product price in each category.

- Each bar represents a category, and its height shows the average price of products in that category.

What is it saying?

- Which categories have higher-priced products

- Which categories have lower-priced products

- The pricing trend across different categories

What features you used?

- category_name → To group products by category

- price → Product prices

- Barplot → To show average price per category

From this what you are showing us?

- Average pricing per category

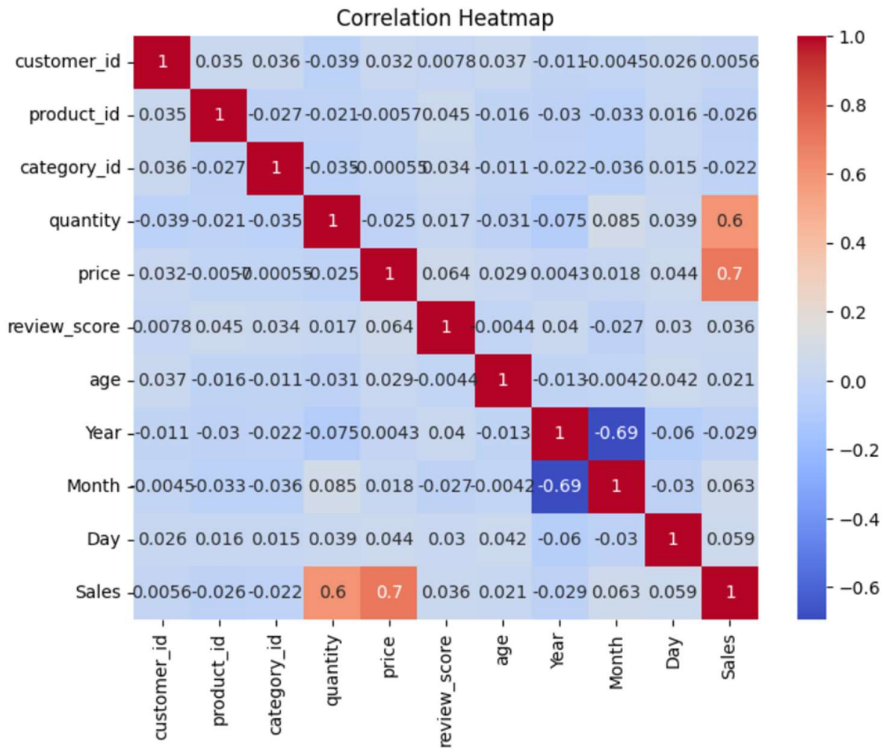
- Insights for pricing strategy and category comparison

- Which categories may be premium vs budget-focused

Code:

```
# Heatmap – Correlation

plt.figure(figsize=(8,6))

sns.heatmap(df.corr(numeric_only=True), annot=True, cmap="coolwarm")

plt.title("Correlation Heatmap")

plt.show()
```

Output:



Interpretation of your above chart

Explain the Chart

- This heatmap shows the correlation between all numerical columns in your dataset.

- Each cell represents how strongly two variables are related (positively or negatively).

- Positive correlation (closer to +1) → When one increases, the other also increases.

- Negative correlation (closer to –1) → When one increases, the other decreases.

- Values near 0 → Very weak or no relationship.

- The color scale (cool to warm) makes it easy to see high and low correlations.

What is it saying?

- Which features are strongly related

- Which variables affect or move with each other

- Whether features have patterns or dependencies

- For example, Sales may show a strong correlation with quantity or price.

What features you used?

- df.corr(numeric_only=True) → Calculates correlation between numeric columns

- Heatmap → To visualize correlation values

- annot=True → Shows the numbers inside the cells

- cmap="coolwarm" → Color gradient for correlation strength

From this what you are showing us?

- The relationship strength between all numerical variables

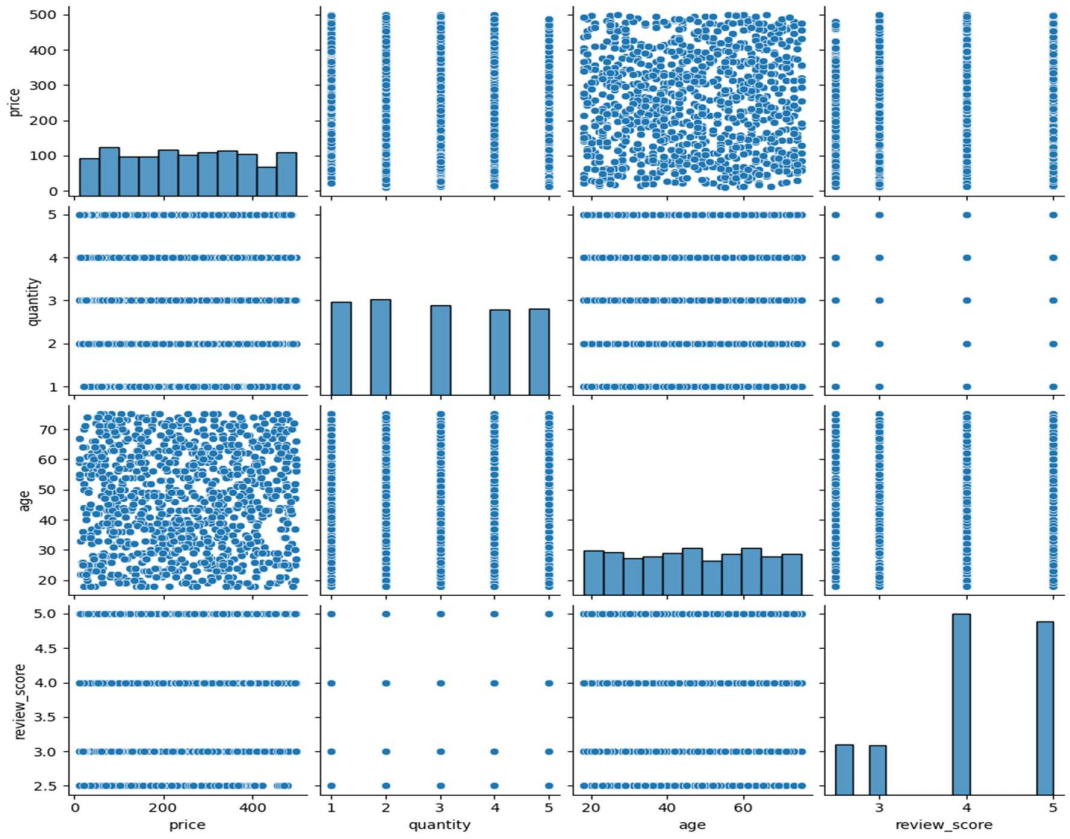
- Which features are important or dependent

- Useful insights for feature selection, modeling, and understanding data behavior

Code:

```
## MULTIVARIATE ANALYSIS

# Pairplot (multi-variable relationship)
sns.pairplot(df[['price', 'quantity', 'age', 'review_score']])
plt.show()
```

Output:



Interpretation of your above chart

Explain the Chart

- This pairplot shows the relationship between multiple numerical variables: price, quantity, age, and review_score.

- It creates scatter plots for every pair of variables and histograms on the diagonal.

What is it saying?

- How two variables relate to each other (e.g., price vs quantity)

- Whether there are patterns, trends, or clusters

- The distribution of each variable

- Possible correlations or unusual values

- This gives a full picture of how the variables behave together.

What features you used?

- price, quantity, age, review_score → Numerical columns selected

- pairplot() → Automatically creates scatter plots + histograms

- Matplotlib/Seaborn → For visualization

From this what you are showing us?

- The overall relationship between key numeric features

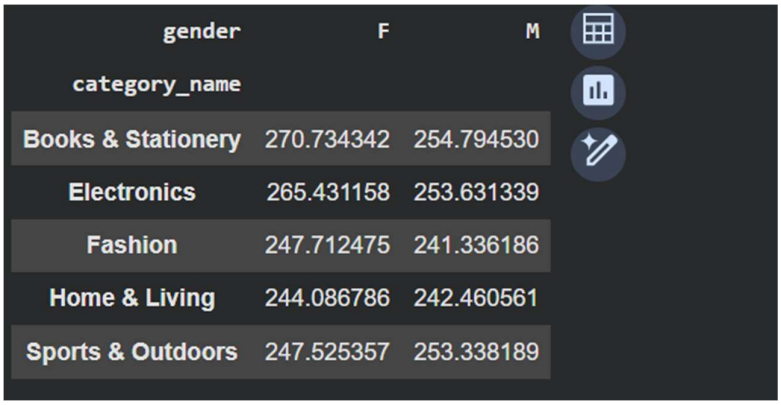
- How each variable impacts or relates to the others

- Insights for data patterns, trends, and model preparation

Code:

```
# Pivot Table – Category vs Gender vs Avg Price

pivot_table = df.pivot_table(values='price', index='category_name',

columns='gender', aggfunc='mean')

pivot_table
```

Output:



gender	F	M
category_name		
Books & Stationery	270.734342	254.794530
Electronics	265.431158	253.631339
Fashion	247.712475	241.336186
Home & Living	244.086786	242.460561
Sports & Outdoors	247.525357	253.338189

Interpretation of your above chart

Explain the Pivot Table

- This pivot table shows the average product price for each category, split by gender.
- Rows = category_name
- Columns = gender (Male/Female)
- Values = average price
- It helps compare how pricing differs for each gender within each category.

What is it saying?

- Which categories have higher average prices
- How male vs female customers differ in average product price
- Categories where one gender spends more than the other

What features you used?

- pivot_table() → To summarize data
- values='price' → The measure used
- index='category_name' → Rows
- columns='gender' → Columns
- aggfunc='mean' → To calculate average

From this what you are showing us?

- Average price comparison between genders for each category
- Insights for targeted pricing, marketing, and product strategy
- Understanding which categories have gender-based price differences

Dashboards:

Dashboard 1: Executive Summary Dashboard:

Objective of the Dashboard

The Executive Summary Dashboard provides a high-level overview of the company's overall performance in terms of **sales, quantity, orders, customer purchase behavior, category-wise contribution, payment mode preference, and best-performing products**.

This dashboard is designed for senior leadership to make quick decisions based on monthly and category trends.

Key Features of the Dashboard

1. Monthly Sales Trend (Line Chart)

- Shows how total sales changed month-by-month.
- Highlights peak and low-performing months.
- Helps identify seasonal or promotional impact on revenue.

2. Category-wise Sales Distribution (Pie Chart)

- Breaks down revenue contribution from each product category.
- Shows which categories drive maximum income.

3. Quantity Sold by Month (Bar Chart)

- Displays volume movement of products monthly.
- Useful for inventory and supply-chain planning.

4. Payment Method Distribution

- Shows customer payment preferences across:
    - Cash on Delivery
    - Bank Transfer
    - Credit Card

5. Top 10 Products by Total Sales (Bar Chart)

- Highlights the highest-selling products.
- Helps identify star performers and low-performing items.

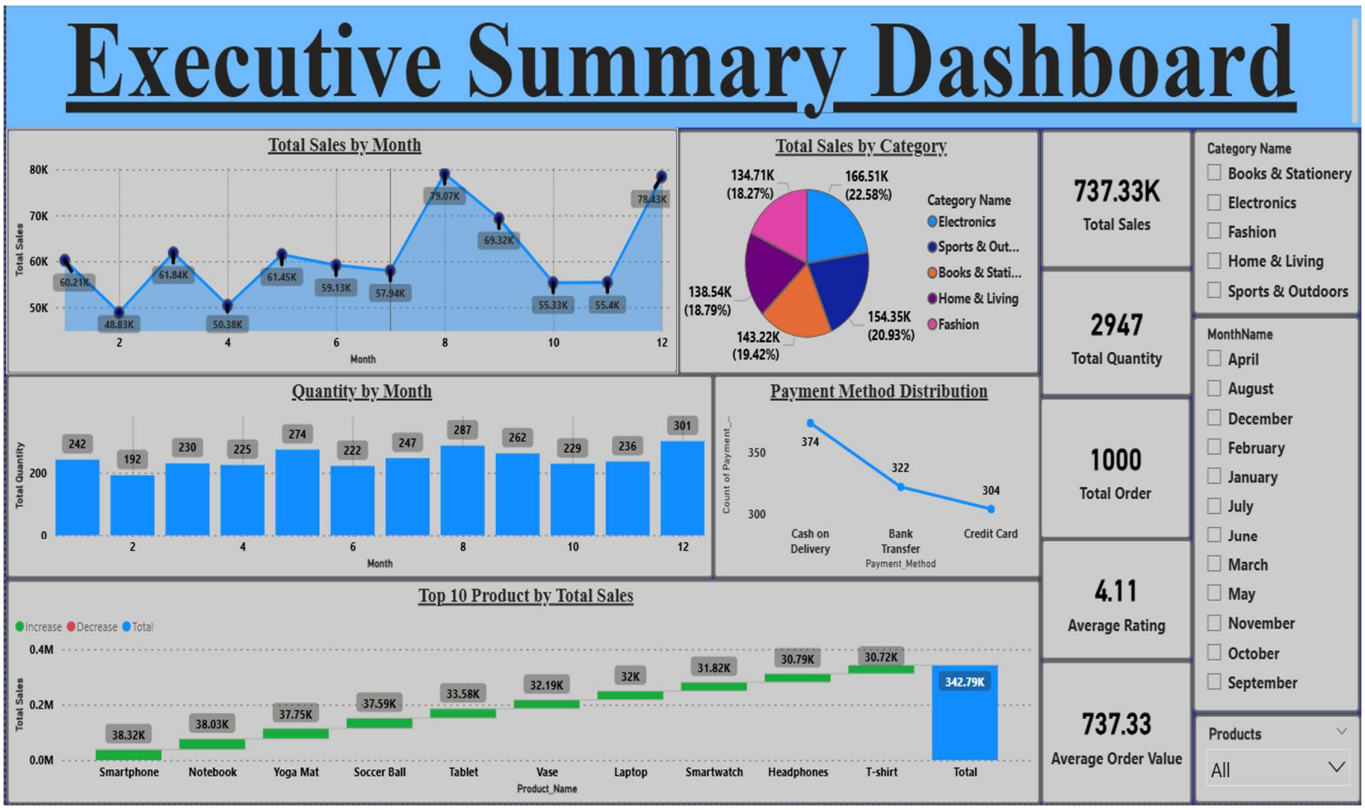
6. KPI Cards

- **Total Sales** (737.33K)
- **Total Quantity Sold** (2947)
- **Total Orders** (1000)
- **Average Rating** (4.11)
- **Average Order Value** (737.33)

7. Interactive Filters

- Filter by:
  - Category Name
  - Month
  - Product

These allow users to drill down and analyze specific segments.



Insights from the Dashboard

1. Strong Year-End Performance

- Sales and quantity peak in **August** and **December**.
- December shows the **maximum sales + maximum quantity**.

Suggests seasonal demand or successful year-end promotions.

2. Electronics Drives the Business

Electronics category contributes **22.58%** of total revenue, the highest among all categories.

Indicates strong customer preference for tech products.

3. Cash on Delivery Dominates

COD accounts for the highest number of transactions (**374**).

Customers prefer **risk-free payment options**.

4. Top 10 Products Contribute a Major Portion of Revenue

Products like:

- Smartphones
- Notebook
- Yoga Mat
- Soccer Ball
  contribute significantly to total sales.

These products should remain priority items for stock and marketing.

5. Average Rating of 4.11 Indicates High Satisfaction

A consistent rating above 4 means:

- Good product quality
- Positive customer experience
- Strong brand trust

6. Balanced Category Contribution

Sales contributions from all five categories are evenly spread (18%–22.5%).

Business does not depend on a single category alone—**low risk**.

Final Summary

Overall, the business performance is strong with:

- Healthy total sales of **737.33K**
- Stable customer satisfaction (avg rating = **4.11**)
- High demand during year-end months
- Balanced category distribution
- Strong contribution from top-performing products
- COD as the preferred payment mode

This indicates a well-rounded business performance with scope for strategic improvements.

Recommendations

Focus on High-Performing Months

- Plan marketing campaigns in **August & December**.
- Offer festive combos & discounts.

Increase Stock for Top Products

Products like Smartphones, Yoga Mats, Soccer Balls, and Notebooks should be prioritized.

Encourage Digital Payments

Provide incentives such as:

- 5% cashback for credit card payments
- Wallet discounts

This can reduce COD risk.

Improve Low-Performing Months

February & April need additional:

- Promotions
- Targeted advertising
- New product launches

Category Strategy

- Boost product variety in Electronics & Sports.
- Expand Home & Living for higher sales growth.

Dashboard 2: Customer Analytics Dashboard

Objective of the Dashboard

The purpose of this Customer Analytics Dashboard is to analyze customer demographics, sales performance, purchasing behaviour, and geographical trends to support data-driven business decisions.

Key Features of the Dashboard

Total Customer Overview

- **1000 total customers**
- **Average Age:** 46.38 years
- **Cities Covered:** 962
- **Gender Distribution:**
  - **Male:** 560
  - **Female:** 440

Age Distribution Analysis

- Histogram comparing male and female customers across age groups.
- Shows customer count in age bins such as 10–20, 20–30, 30–40, 40–50, 50–60, 60–70, and above.

Total Sales by Gender

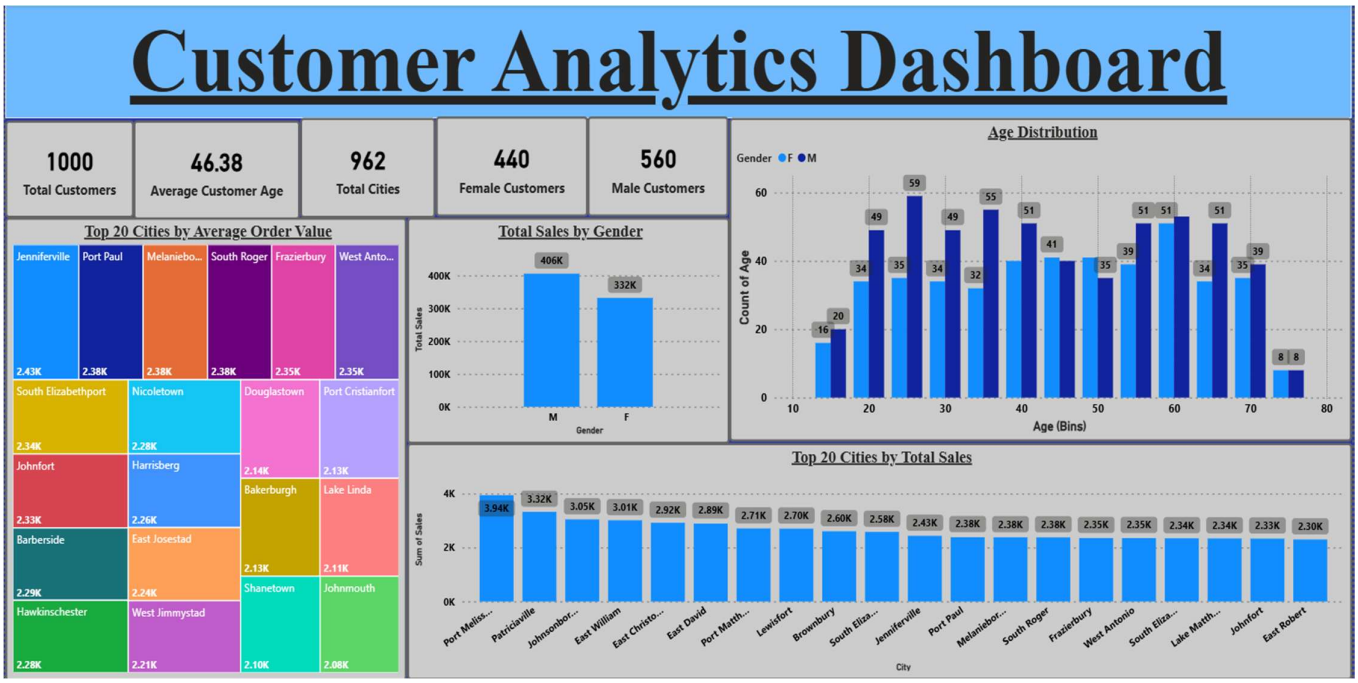
- Bar chart representing the total revenue contributed by male and female customers.
- Helps identify which gender drives more sales.

Top 20 Cities by Average Order Value (AOV)

- Treemap showing the highest-spending cities.
- Highlights cities with stronger purchasing power.

Top 20 Cities by Total Sales

- Horizontal bar chart showcasing cities that contribute the most sales volume.
- Helps to identify revenue-driving regions.



Insights from the Dashboard

1. Customer Demographics

- The business caters to **1000 customers** spread across **962 cities**.
- The **average customer age is 46.38**, indicating the major customer base is middle-aged.
- Gender split:
  - **Male (56%)**
  - **Female (44%)**

Male customers slightly dominate the customer base.

2. Sales by Gender

- **Male customers generated ~406K sales**, while
- **Female customers generated ~332K sales**.

Male customers contribute a higher share of total sales, showing strong revenue potential from this segment.

3. Age Distribution Insights

- Highest customer concentration is in the age ranges:
  - **30–40 years**
  - **40–50 years**
  - **50–60 years**
- Lowest activity is observed in age ranges:
  - **10–20 years**
  - **70–80 years**

Middle-aged customers (30–60 years) form the core buyer group.

4. Top Cities by Average Order Value (AOV)

Cities with the highest AOV (2.10K – 2.43K range):

- Jenniferville
- Port Paul
- Melanieborough
- South Roger
- Frazierbury

These cities show higher spending power and strong purchasing behaviour.

5. Top Cities by Total Sales

Top revenue-generating cities (Sales 2.30K – 3.94K):

- Port Melissa
- PatriciaVille
- Johnsonboro
- East William
- East Christo
- East David

Sales are not limited to a single region; the business has a well-spread revenue footprint.

Final Summary

The Customer Analytics Dashboard provides a clear understanding of customer behavior and sales performance:

- Strong geographic reach **across 962 cities.**
- Majority revenue contribution from male customers.
- Core buying age group: 30–60 years.
- Stable and consistent AOV across top 20 cities.

- Multiple cities contribute significantly to sales—not dependent on one or two locations.

Recommendations

Marketing

- Focus targeted campaigns on **middle-aged groups (30–60 years)**.
- Create gender-specific promotions to increase both male and female engagement.

Regional Strategy

- Expand promotional efforts in mid-performing cities with potential for growth.
- Strengthen customer loyalty programs in top AOV cities.

Business Growth

- Introduce personalized recommendations and retention strategies for high-value customers.
- Monitor city-wise performance monthly to optimize marketing spend.

Dashboard 3: Product & Ratings Dashboard

Objective of the Dashboard

The purpose of the Product & Ratings Dashboard is to analyze product performance across categories, understand customer rating patterns, and evaluate how product ratings influence total sales. This helps the business identify high-performing products, underperforming segments, and customer satisfaction trends.

Key Features of the Dashboard

Product Overview Metrics

- **Total Categories:** 5
- **Average Category Price:** 251.85
- **Highest Product Sales Amount:** 38.32K
- **Lowest Product Sales Amount:** 18.23K
- **Average Rating (across all products):** 4.11

Total Quantity by Category

A horizontal bar chart displaying the total quantity sold per category:

- **Electronics:** 648

- **Sports & Outdoors:** 625
- **Fashion:** 564
- **Home & Living:** 563
- **Books & Stationery:** 547

Electronics leads both in demand and volume.

Average Category Price

- Books & Stationery – **261**
- Electronics – **259**
- Sports & Outdoors – **251**
- Fashion – **245**
- Home & Living – **243**

Books & Stationery has the highest average price, Home & Living the lowest.

Product-Level Performance Table

Shows:

- Category Name
- Product Name
- Average Rating
- Total Quantity
- Total Sales

This helps identify:

- High-selling products
- Best-rated items
- Products needing improvement

Sales & Review Comparison by Rating Category

Bar + line combination chart highlights:

- **Total Sales by Rating Category**
- **Review Count by Rating Category**

Values:

- **Good:** 372K total sales, 508 reviews
- **Excellent:** 291K total sales, 382 reviews
- **Average:** 74K total sales, 110 reviews

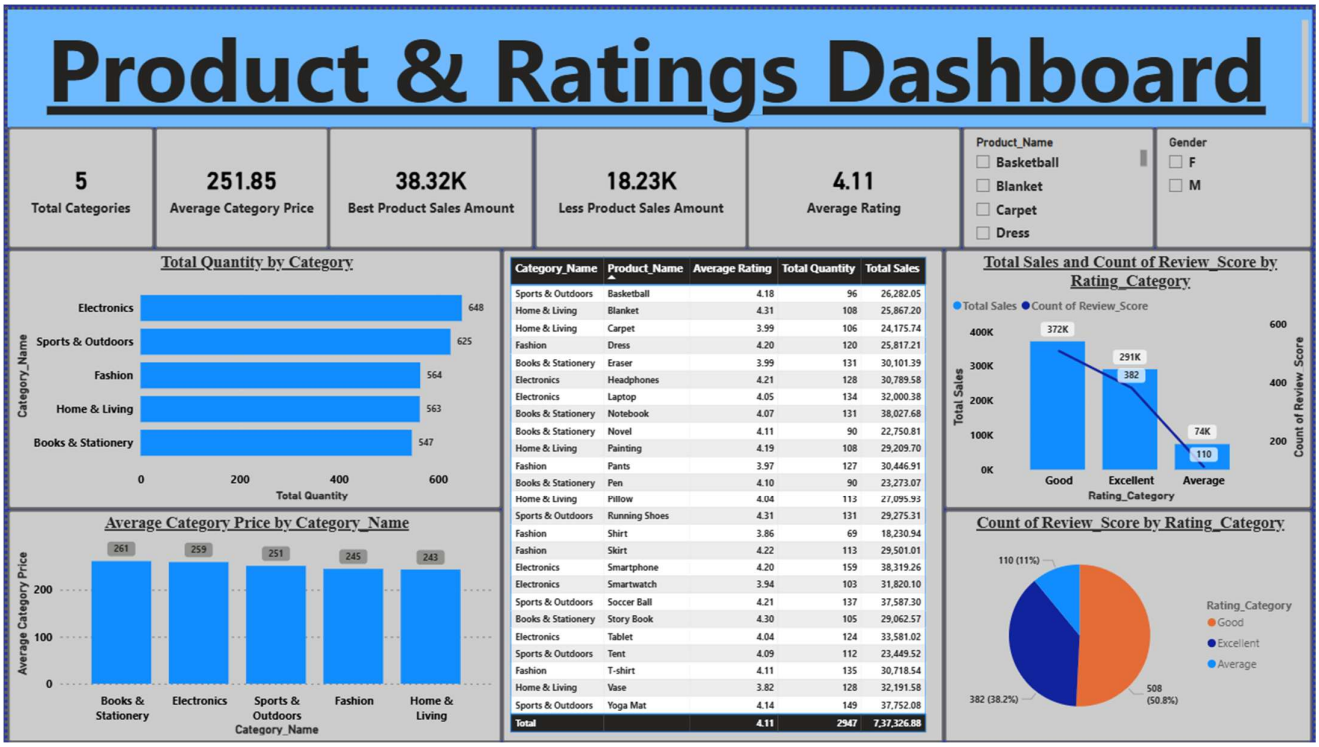
Higher ratings correlate directly with higher sales.

Review Score Distribution

Pie chart showing percentage of reviews:

- Good: **50.8%**
- Excellent: **38.2%**
- Average: **11%**

Majority of the customer base rates products positively.



Category_Name	Product_Name	Average Rating	Total Quantity	Total Sales
Sports & Outdoors	Basketball	4.18	96	26,282.05
Home & Living	Blanket	4.31	108	25,867.20
Home & Living	Carpet	3.99	106	24,175.74
Fashion	Dress	4.20	120	25,817.21
Books & Stationery	Eraser	3.99	131	30,101.39
Electronics	Headphones	4.21	128	30,789.58
Electronics	Laptop	4.05	134	32,000.38
Books & Stationery	Notebook	4.07	131	38,027.68
Books & Stationery	Novel	4.11	90	22,750.81
Home & Living	Painting	4.19	108	29,209.70
Fashion	Pants	3.97	127	30,446.91
Books & Stationery	Pen	4.10	90	23,273.07
Home & Living	Pillow	4.04	113	27,095.93
Sports & Outdoors	Running Shoes	4.31	131	29,275.31
Fashion	Shirt	3.86	69	18,230.94
Fashion	Skirt	4.22	113	29,501.01
Electronics	Smartphone	4.20	159	38,319.26
Electronics	Smartwatch	3.94	103	31,820.10
Sports & Outdoors	Soccer Ball	4.21	137	37,587.30
Books & Stationery	Story Book	4.30	105	29,062.57
Electronics	Tablet	4.04	124	33,581.02
Sports & Outdoors	Tent	4.09	112	23,449.52
Fashion	T-shirt	4.11	135	30,718.54
Home & Living	Vase	3.82	128	32,191.58
Sports & Outdoors	Yoga Mat	4.14	149	37,752.08
Total		4.11	2947	7,37,326.88

Insights From the Dashboard

1. Electronics Leads in Demand

With **648 units sold**, Electronics is the best-performing category in total quantity.
This indicates customer preference for gadgets and tech products.

2. Positive Customer Satisfaction

- Average rating across all products = **4.11**
- Over **89% of reviews** fall under *Good* or *Excellent*.

Customers are generally satisfied with product quality.

3. Higher Ratings = Higher Sales

- **Good-rated products** generated **372K**, the highest.
- **Excellent-rated products** contributed **291K**, also significant.
- **Average-rated products** contributed only **74K**.

Rating quality strongly impacts selling potential.

4. Top-Selling Products

From the product table:

- Basketball
- Blanket
- Carpet
- Dress
- Headphones
- Laptop

These items should be prioritized for inventory, marketing, and promotions.

5. Price Variation Across Categories

- Prices remain within a close range (243 to 261).
- Books & Stationery is priced slightly higher on average.

Pricing strategy is consistent and balanced.

Final Summary

The Product & Ratings Dashboard demonstrates strong performance across product categories:

- Electronics and Sports & Outdoors are top-selling categories.
- Majority of customers give good or excellent ratings.
- Strong correlation between product ratings and total sales.
- Product pricing is stable across categories.
- Several individual products (like Basketball, Blanket, Laptop) contribute significantly to revenue.

Recommendations

Improve Performance of Low-Rated Products

- Analyze negative reviews for products rated "Average".
- Improve quality or customer experience around these products.

Promote Best-Selling Items

- Increase visibility, ads, or discount bundles for:
    - Basketball
    - Blanket
    - Laptop
    - Headphones

Category-Level Strategy

- Boost inventory for high-demand categories (Electronics, Sports & Outdoors)
- Introduce new product lines in low-quantity categories like Books & Stationery.

Maintain High Customer Satisfaction

- Encourage customers to leave reviews.
- Use loyalty rewards for highly-rated products.

Future Enhancement

1. Advanced Predictive Analytics

- Forecast customer churn

- Predict future revenue trends

- Fraud probability detection

2. Automated Alerts & Notifications

- Alerts for sudden revenue drops

- Notifications for high churn–risk customers

- Seasonal or weekend performance alerts

3. Drill-Through & Deep-Dive Analysis

- Country-wise revenue breakdown

- Monthly, weekly, and seasonal trend insights

4. AI-Powered Insights

- Automatic trend explanations

- AI-driven business suggestions

5. Customer & Product Segmentation

- Age-based customer groups

- Risk-based customer clusters

- Geography-based clusters

6. Recommendation Systems

- Best product/category to promote

- Regions needing fraud monitoring

- Months requiring stronger marketing push

Conclusion

- This analysis provides a comprehensive understanding of customer behavior, product performance, and overall business trends. The dashboards highlight key patterns in sales, ratings, demographics, and product categories, enabling data-driven decision-making. Insights such as top-performing products, monthly sales fluctuations, customer distribution, and rating trends help identify growth opportunities and areas that need improvement.

- By transforming raw data into meaningful visual insights, the project supports better strategic planning in marketing, inventory management, customer engagement, and product development. With the recommended future enhancements—such as predictive analytics, segmentation, and AI-based insights—the analysis can evolve into a more advanced decision-support system.

- Overall, this project demonstrates the value of data analytics in understanding business performance and driving actionable outcomes.