*AJIO Data Analysis by using Python (eCommerce) *

Problem Statement

Ajio, a leading fashion eCommerce platform, aims to enhance its customer experience, operational efficiency, and product strategy by leveraging data-driven insights. The challenges aligns in understanding customer buying behavior, optimizing product offerings and identifying issues related to returns, delivery delays, and payment failures.

To address these gaps, an in-depth Exploratory Data Analysis (EDA) of the Ajio dataset—which includes customer, order, product, rating, transaction, delivery, and return data—is essential. This analysis will uncover key trends, patterns, and anomalies, enabling data-backed decisions that improve customer retention, reduce operational inefficiencies for future growth to increase overall profitability.

Objectives

🎯 1. Understand Customer Behavior and Segmentation and improve customer retention.

🎯 2. Optimize inventory, pricing, and product placement strategies.

🎯 3. Enhance supply chain reliability and customer satisfaction.

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

```
#Load the datasets
customers = pd.read_csv("customer.csv")
delivery = pd.read_csv("delivery.csv")
ratings = pd.read_csv("ratings.csv")
orders = pd.read_csv("orders.csv")
products = pd.read_csv("products.csv")
transactions = pd.read_csv("transaction.csv")
returns = pd.read_csv("returns.csv")
```

```
customers.head()
```

	C_ID	C_Name	Gender	Age	City	State	Street_Address	Mobile
0	CS_11000001	Manbir Lala	Male	67	Delhi	Delhi	Park Ave, 163 , Delhi , Delhi - 529675	9607971039
1	CS_11000002	Radhika More	Female	51	Pune	Maharashtra	Elm St, 960 , Pune , Maharashtra - 328062	9109249091
2	CS_11000003	Faqid Halder	Female	57	Bengaluru	Karnataka	Maple St, 71 , Bengaluru , Karnataka - 574209	9129509047
3	CS_11000004	Chandresh Dugar	Female	26	Thane	Maharashtra	2nd St, 557 , Thane , Maharashtra - 329555	9351639395
4	CS_11000005	Logan Soni	Male	24	Ghaziabad	Uttar Pradesh	Pine St, 758 , Ghaziabad , Uttar Pradesh - 119526	9445754174

```
#Check for data types
orders.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 9 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Or_ID       10000 non-null  object
 1   C_ID        10000 non-null  object
 2   P_ID        10000 non-null  object
 3   Order_Date  10000 non-null  object
 4   Order_Time  10000 non-null  object
 5   Qty         10000 non-null  int64
 6   Coupon      10000 non-null  object
 7   DP_ID       10000 non-null  object
 8   Discount    10000 non-null  int64
dtypes: int64(2), object(7)
memory usage: 703.3+ KB
```

```
orders.describe()
```

	Qty	Discount
count	10000.000000	10000.000000
mean	5.513600	9.066700
std	2.882195	11.828941
min	1.000000	0.000000
25%	3.000000	0.000000
50%	6.000000	0.000000
75%	8.000000	15.000000
max	10.000000	50.000000

```
transactions.describe()
```

	Tr_ID	Or_ID	Transaction_Mode	Reward
count	10000	10000	10000	10000
unique	10000	6248	5	2
top	TR_41009984	OR_31004020	Net Banking	No
freq	1	6	2057	5024

```
#To check for duplicates in your dataframe
orders.duplicated().sum()
transactions.duplicated().sum()
customers.duplicated().sum()
ratings.duplicated().sum()
delivery.duplicated().sum()
products.duplicated().sum()
returns.duplicated().sum()
```

```
np.int64(0)
```

```
#If there are any duplicated value then use
orders = orders.drop_duplicates()
transactions = transactions.drop_duplicates()
customers = customers.drop_duplicates()
ratings = ratings.drop_duplicates()
delivery = delivery.drop_duplicates()
products = products.drop_duplicates()
returns = returns.drop_duplicates()
```

```
orders.isnull().sum()
```

	0
Or_ID	0
C_ID	0
P_ID	0
Order_Date	0
Order_Time	0
Qty	0
Coupon	0
DP_ID	0
Discount	0

dtype: int64

```
#assuming quantity has null values

orders["Qty"] = orders["Qty"].fillna(orders["Qty"].median()) #mean() or mode()
orders["Qty"] = orders["Qty"].ffill() #bfill
```

```
customers.head(5)
```

	C_ID	C_Name	Gender	Age	City	State	Street_Address	Mobile
0	CS_11000001	Manbir Lala	Male	67	Delhi	Delhi	Park Ave, 163 , Delhi , Delhi - 529675	9607971039
1	CS_11000002	Radhika More	Female	51	Pune	Maharashtra	Elm St, 960 , Pune , Maharashtra - 328062	9109249091
2	CS_11000003	Faqid Halder	Female	57	Bengaluru	Karnataka	Maple St, 71 , Bengaluru , Karnataka - 574209	9129509047
3	CS_11000004	Chandresh Dugar	Female	26	Thane	Maharashtra	2nd St, 557 , Thane , Maharashtra - 329555	9351639395
4	CS_11000005	Logan Soni	Male	24	Ghaziabad	Uttar Pradesh	Pine St, 758 , Ghaziabad , Uttar Pradesh - 119526	9445754174

```
orders.head(5)
```

	Or_ID	C_ID	P_ID	Order_Date	Order_Time	Qty	Coupon	DP_ID	Discount
0	OR_31000001	CS_11005317	PD_21001301	2024-02-27	22:02:00	1	No Coupon	DV_61000001	0
1	OR_31000002	CS_11000423	PD_21003593	2024-01-21	08:33:31	5	PULL	DV_61000002	25
2	OR_31000003	CS_11001042	PD_21004315	2024-09-22	17:26:05	1	No Coupon	DV_61000002	0
3	OR_31000004	CS_11004079	PD_21007443	2023-05-26	03:15:48	10	AGREEMENT	DV_61000003	10
4	OR_31000005	CS_11009894	PD_21007621	2023-10-26	04:02:44	7	WINDOW	DV_61000001	10

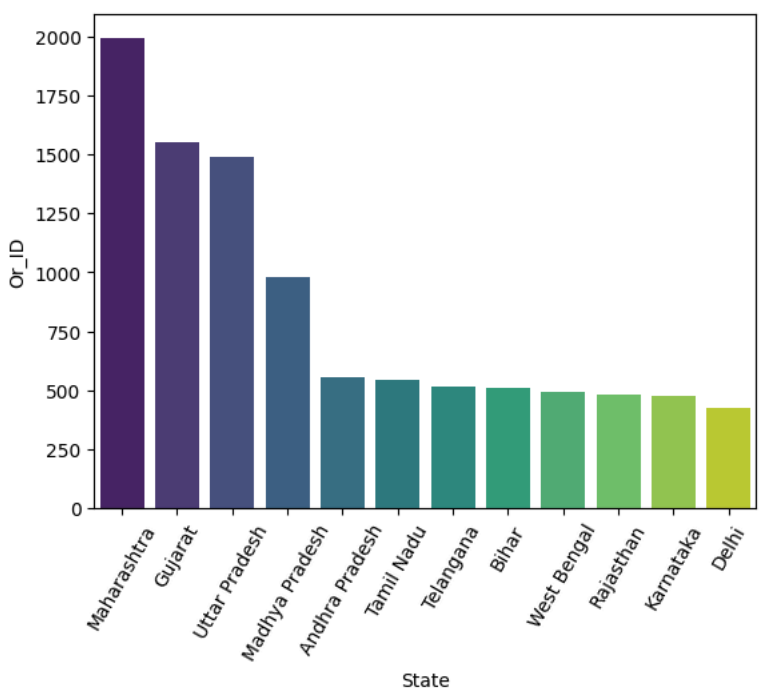
```
#to join two tables together, merge function is used

merged_customer_orders = pd.merge(right = customers, left = orders, how = "inner", on = "C_ID")
merged_customer_orders.head()
```

	Or_ID	C_ID	P_ID	Order_Date	Order_Time	Qty	Coupon	DP_ID	Discount	C_Name	Gender	Age
0	OR_31000001	CS_11005317	PD_21001301	2024-02-27	22:02:00	1	No Coupon	DV_61000001	0	Balvan Mahajan	Male	67
1	OR_31000002	CS_11000423	PD_21003593	2024-01-21	08:33:31	5	PULL	DV_61000002	25	Vincent Sinha	Female	59
2	OR_31000003	CS_11001042	PD_21004315	2024-09-22	17:26:05	1	No Coupon	DV_61000002	0	Yagnesh Narang	Male	44

```
#group by is used to summarise data i.e. it creates pivot tables
gb = merged_customer_orders.groupby("State").agg({"Or_ID":"count"})
gb = gb.sort_values(by = "Or_ID", ascending=False)
gb
```

State	Or_ID
Maharashtra	1994
Gujarat	1551
Uttar Pradesh	1487
Madhya Pradesh	977
Andhra Pradesh	554
Tamil Nadu	541
Telangana	513
Bihar	510
West Bengal	490
Rajasthan	481
Karnataka	478
Delhi	424

```
#Visualize the name of the states from which customers mostly placed orders
sns.barplot(x= gb.index, y= "Or_ID", data = gb, hue= gb.index, palette="viridis")
plt.xticks(rotation = 60)
plt.show()
```



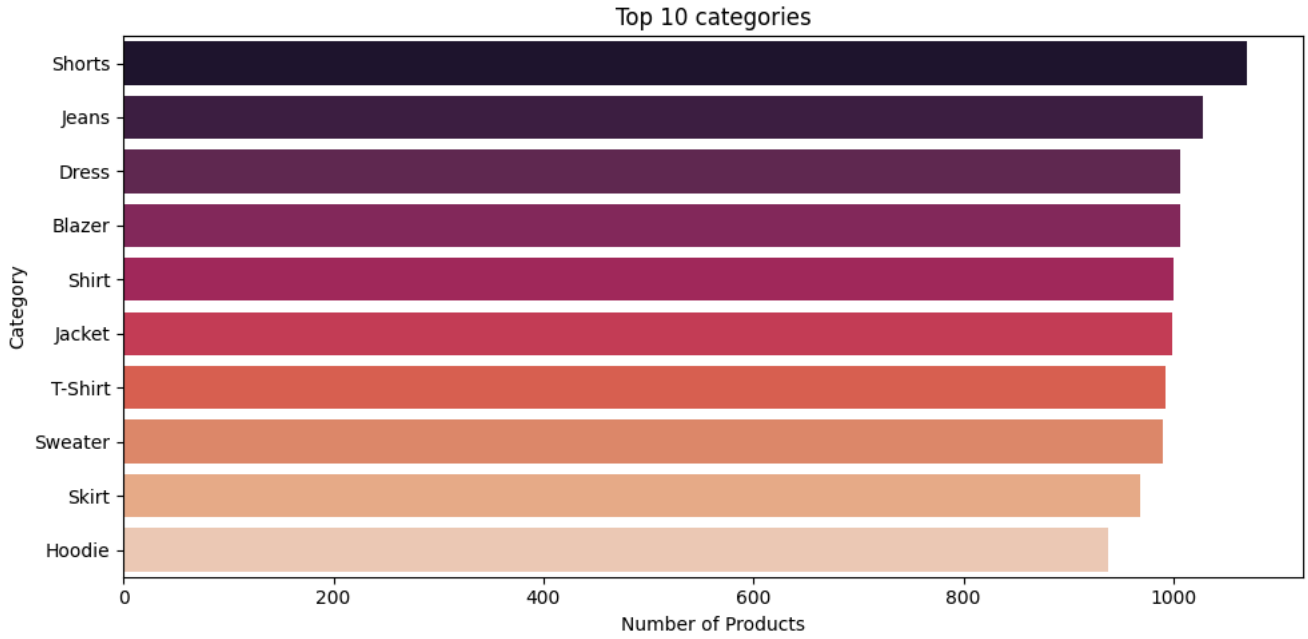
Generate questions regarding EDA analysis on the given dataset for creating ecommerce dashboard.

```
products.head()
```

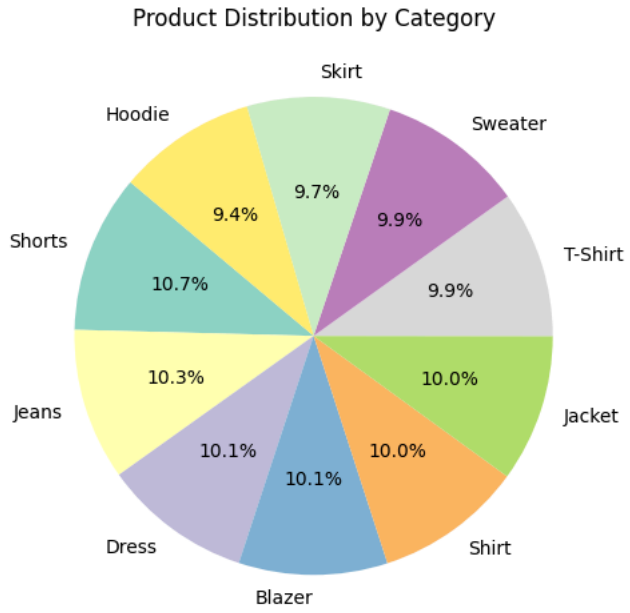
	P_ID	P_Name	Category	Company_Name	Gender	Price
0	PD_21000001	Distressed Stretch Denim Charcoal Faded Jeans	Jeans	Puma	Unisex	1589
1	PD_21000002	Straight Leg Cotton Blend Light Blue Faded Jeans	Jeans	Gap	Men	2211
2	PD_21000003	Single-Breasted Cashmere Gray Houndstooth Blazer	Blazer	Reebok	Unisex	2797
3	PD_21000004	Cropped Knit Dark Green Textured Hoodie	Hoodie	Puma	Men	2160
4	PD_21000005	Formal Silk Olive Green Solid Shirt	Shirt	Levi's	Women	566

```
#Top Top-Selling Product Categories (by count)
top_categories = products['Category'].value_counts().head(10)

plt.figure(figsize=(10,5))
sns.barplot(x=top_categories.values, y=top_categories.index, palette='rocket')
plt.title('Top 10 categories')
plt.xlabel('Number of Products')
plt.ylabel('Category')
plt.tight_layout()
plt.show()
```

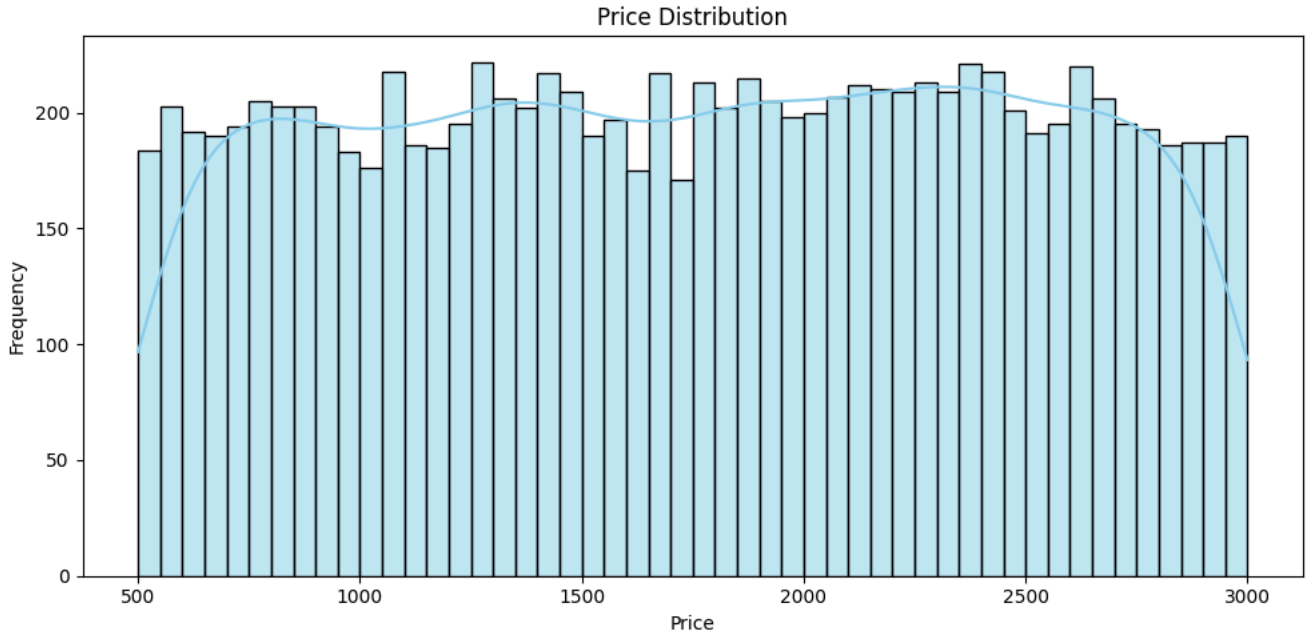
```
/tmp/ipython-input-17-319878112.py:5: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `le

  sns.barplot(x=top_categories.values, y=top_categories.index, palette='rocket')
```




```
#Product Distribution Across Categories
category_counts = products['Category'].value_counts()

plt.figure(figsize=(5,5))
category_counts.plot.pie(autopct='%1.1f%%', startangle=140, colormap='Set3')
plt.ylabel('')
plt.title('Product Distribution by Category')
plt.tight_layout()
plt.show()
```



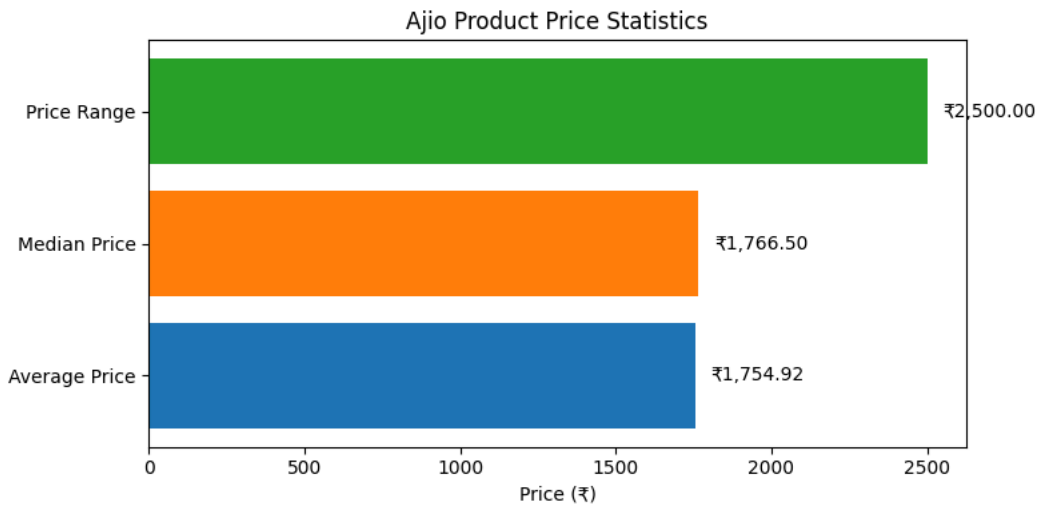

```
#Price distribution

plt.figure(figsize=(10,5))
sns.histplot(products['Price'], bins=50, kde=True, color='skyblue')
plt.title('Price Distribution')
plt.xlabel('Price')
plt.ylabel('Frequency')
plt.tight_layout()
plt.show()
```




```
#Price Statistics (Average, Median, Range)
average_price = products['Price'].mean()
median_price = products['Price'].median()
price_range = products['Price'].max() - products['Price'].min()

print(f"Average Price: ${average_price:.2f}")
print(f"Median Price: ${median_price:.2f}")
print(f"Price Range: ${price_range:.2f}")
```

```
Average Price: $1754.92
Median Price: $1766.50
Price Range: $2500.00
```

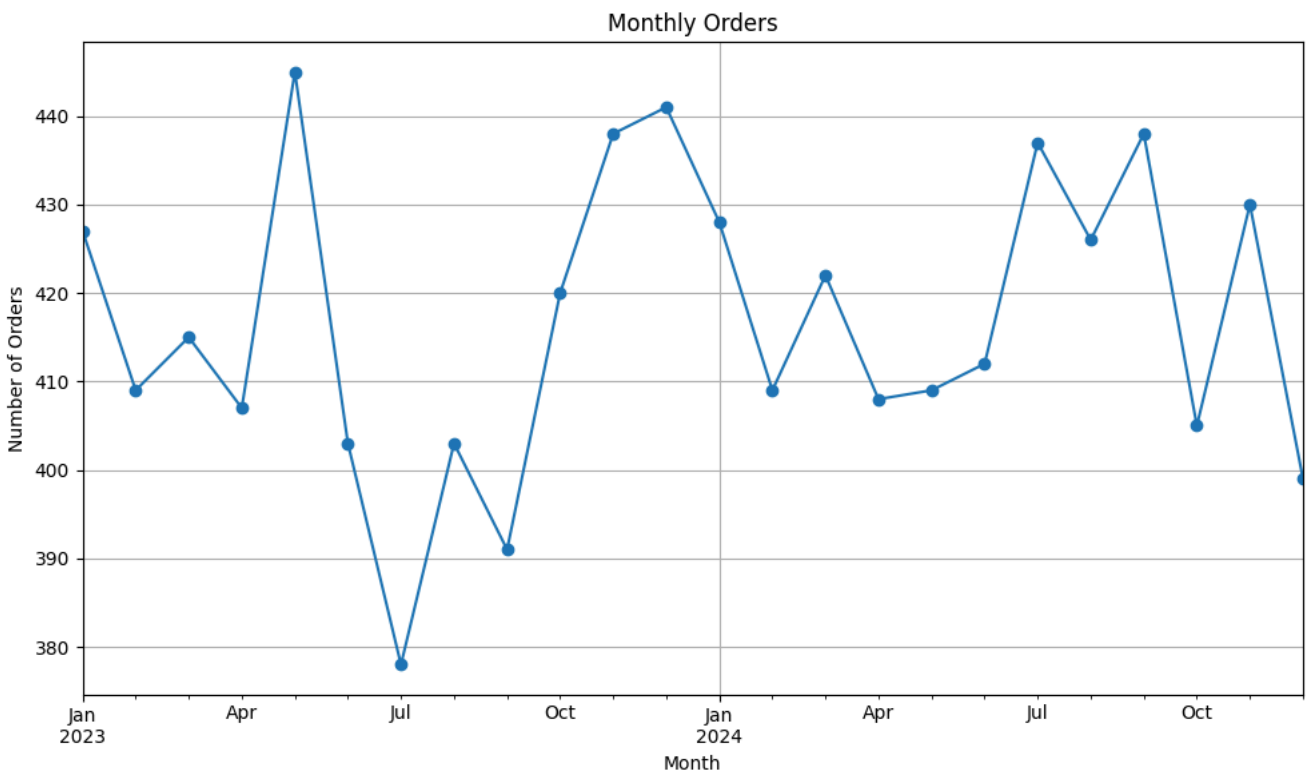
```
# Price Statistics Visualization (Bar Plot)
labels = ['Average Price', 'Median Price', 'Price Range']
values = [average_price, median_price, price_range]

# Plotting
plt.figure(figsize=(8, 4))
bars = plt.barh(labels, values, color=['#1f77b4', '#ff7f0e', '#2ca02c'])

# Add value labels to bars
for bar in bars:
    plt.text(bar.get_width() + 50, bar.get_y() + bar.get_height()/2,
             f"₹{bar.get_width():,.2f}", va='center')

plt.title('Ajio Product Price Statistics')
plt.xlabel('Price (₹)')
plt.tight_layout()
plt.show()
```



```
orders.info()
```

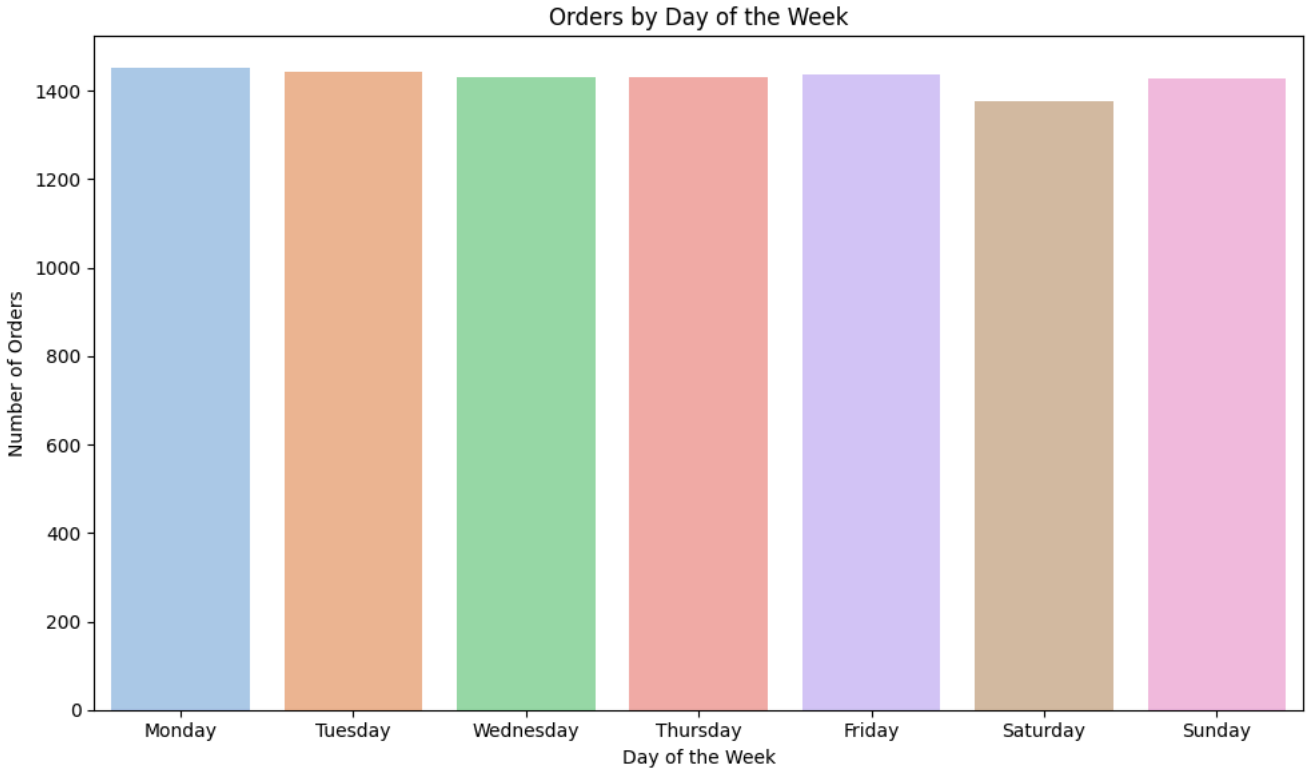
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 9 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   Or_ID       10000 non-null  object
 1   C_ID        10000 non-null  object
 2   P_ID        10000 non-null  object
 3   Order_Date  10000 non-null  object
 4   Order_Time  10000 non-null  object
 5   Qty         10000 non-null  int64 
 6   Coupon      10000 non-null  object
 7   DP_ID       10000 non-null  object
 8   Discount    10000 non-null  int64 
dtypes: int64(2), object(7)
memory usage: 703.3+ KB
```

```
# Total Number of Orders Over Time
orders['Order_Date'] = pd.to_datetime(orders['Order_Date'])
orders_monthly = orders.groupby(orders['Order_Date'].dt.to_period("M")).size()
orders_monthly.index = orders_monthly.index.to_timestamp()

plt.figure(figsize=(10,6))
orders_monthly.plot(marker='o', linestyle='-')
plt.title("Monthly Orders")
plt.xlabel("Month")
plt.ylabel("Number of Orders")
plt.grid(True)
plt.tight_layout()
plt.show()
```




```
#
orders['day_of_week'] = orders['Order_Date'].dt.day_name()
day_order_counts = orders['day_of_week'].value_counts().reindex(['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Su

plt.figure(figsize=(10,6))
sns.barplot(x=day_order_counts.index, y=day_order_counts.values, palette = 'pastel')
plt.title("Orders by Day of the Week")
plt.xlabel("Day of the Week")
plt.ylabel("Number of Orders")
plt.tight_layout()
plt.show()
```

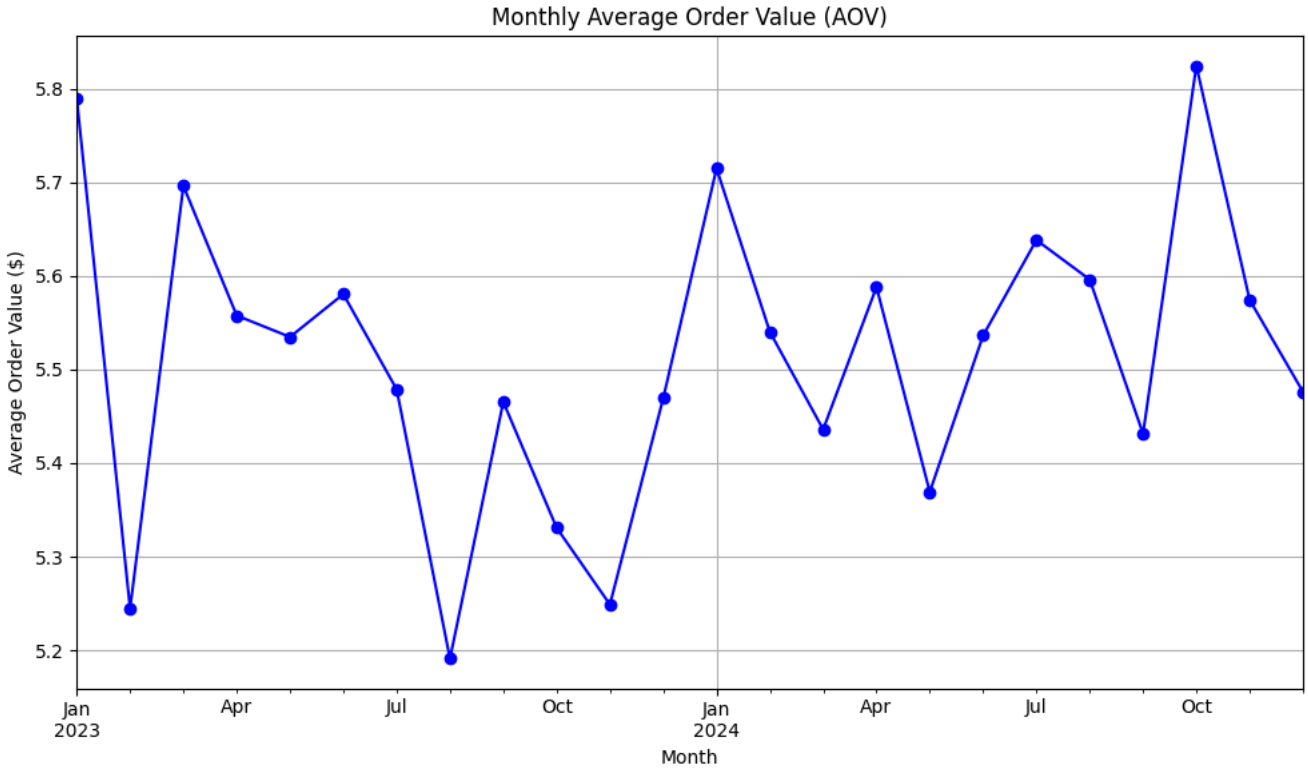
```
/tmp/ipython-input-53-147989365.py:6: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `le

  sns.barplot(x=day_order_counts.index, y=day_order_counts.values, palette = 'pastel')
```




```
#Average Order Value (AOV)
aov = orders['Qty'].mean()
print("Average ORder Value (AOV): ${:.2f}".format(aov))
```

```
Average ORder Value (AOV): $5.51
```

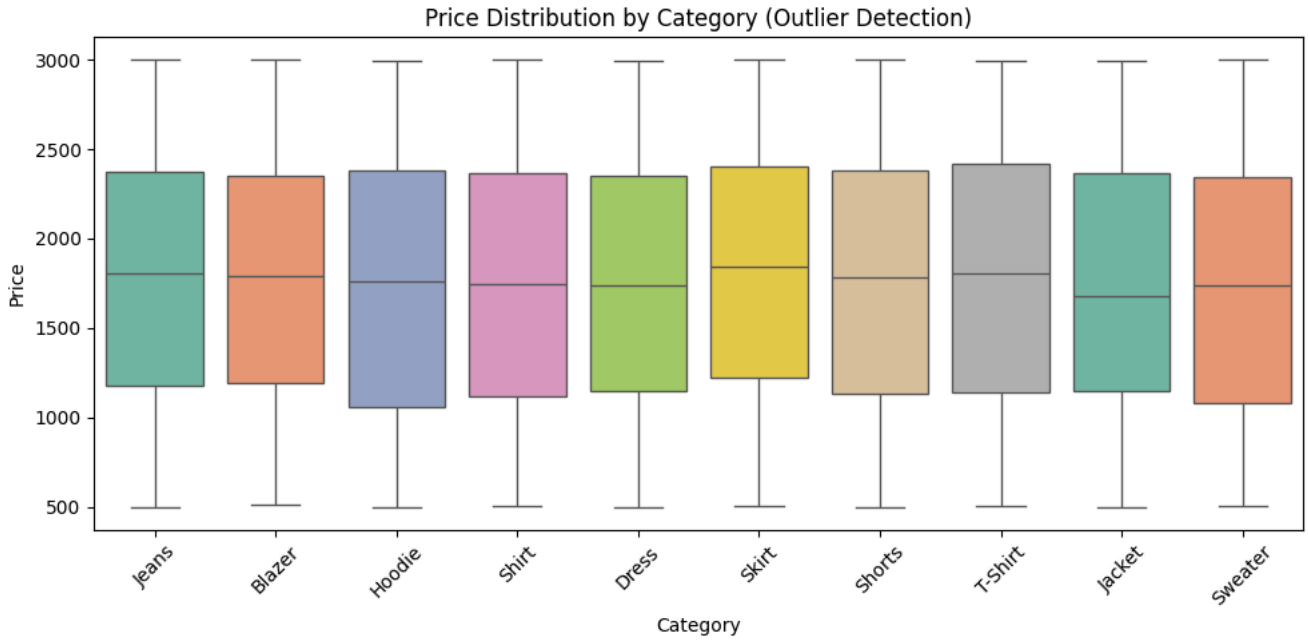
```
#Distribution of monthly Average Order Value (AOV)
aov_monthly = orders.groupby(orders['Order_Date'].dt.to_period("M"))['Qty'].mean()
aov_monthly.index = aov_monthly.index.to_timestamp()

plt.figure(figsize=(10,6))
aov_monthly.plot(marker='o', linestyle='-', color='blue')
plt.title("Monthly Average Order Value (AOV)")
plt.xlabel("Month")
plt.ylabel("Average Order Value ($)")
plt.grid(True)
plt.tight_layout()
plt.show()
```




```
orders['discount_percentage'] = ((products['Price'] - orders['Discount']) / products['Price']) * 100
```

```
#Check the outliers in Price
plt.figure(figsize=(10,5))
sns.boxplot(x='Category', y='Price', data=products, palette='Set2')
plt.xticks(rotation=45)
plt.title('Price Distribution by Category (Outlier Detection)')
plt.tight_layout()
plt.show()
```

```
/tmp/ipython-input-25-3911932655.py:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `le

  sns.boxplot(x='Category', y='Price', data=products, palette='Set2')
```




```
customers.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 8 columns):
 #   Column          Non-Null Count  Dtype
```

```
 ---  ------          --------------  -----
 0   C_ID            10000 non-null  object
 1   C_Name          10000 non-null  object
 2   Gender          10000 non-null  object
 3   Age             10000 non-null  int64
 4   City            10000 non-null  object
 5   State           10000 non-null  object
 6   Street_Address  10000 non-null  object
 7   Mobile          10000 non-null  int64
dtypes: int64(2), object(6)
memory usage: 625.1+ KB
```

```
#Total unique customers
unique_customers = customers['C_ID'].nunique()
print(f"Total unique customers: {unique_customers}")
```

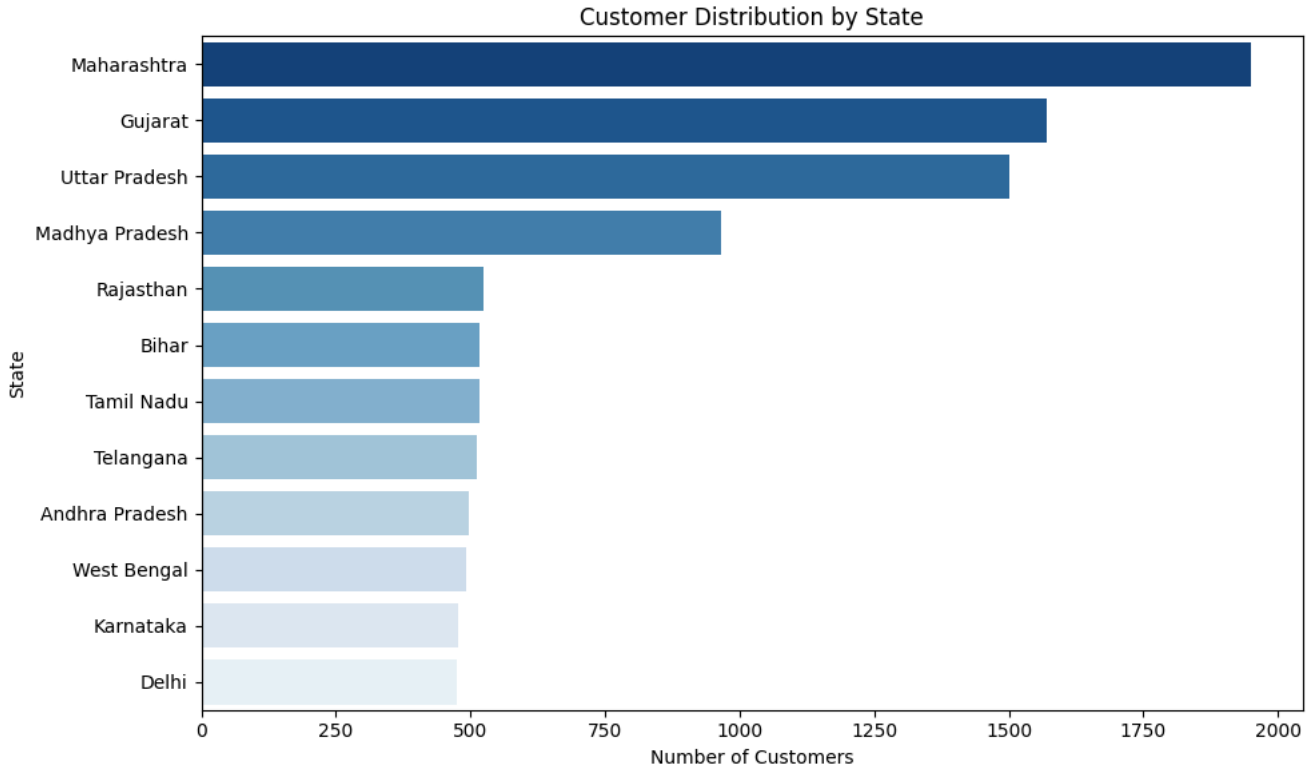
```
Total unique customers: 10000
```

```
#Distribution of customers by state, gender and age group
state_counts = customers['State'].value_counts()
gender_counts = customers['Gender'].value_counts()
bins = [0, 18, 25, 35, 50, 65, 100]
labels = ['<18', '18-24', '25-34', '35-49', '50-64', '65+']
customers['Age_Group'] = pd.cut(customers['Age'], bins=bins, labels=labels, right=False)
age_group_counts = customers['Age_Group'].value_counts().sort_index()
print(state_counts, gender_counts, age_group_counts, sep="\n\n")
```

```
State
Maharashtra       1949
Gujarat           1571
Uttar Pradesh     1501
Madhya Pradesh     965
Rajasthan          525
Bihar              518
Tamil Nadu         517
Telangana          512
Andhra Pradesh     498
West Bengal        493
Karnataka          477
Delhi              474
Name: count, dtype: int64

Gender
Male      5080
Female    4920
Name: count, dtype: int64

Age_Group
<18         0
18-24    1337
25-34    1839
35-49    2870
50-64    2793
65+      1161
Name: count, dtype: int64
```

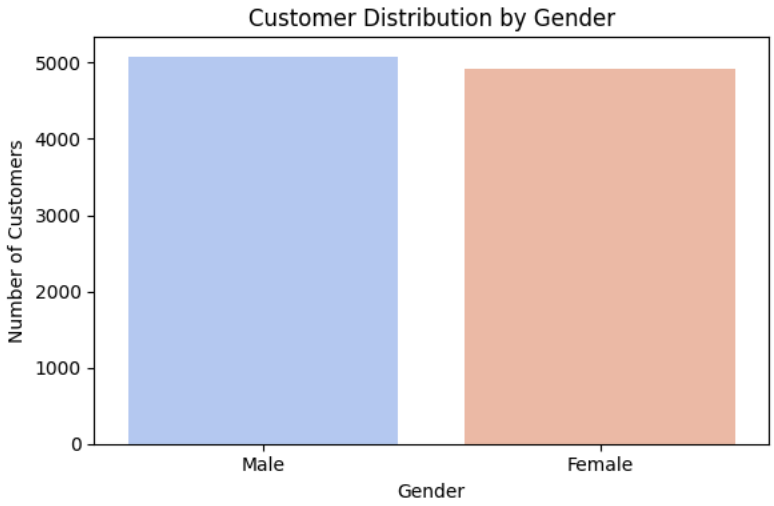
```
#Visualiza distribution of customers based on State
plt.figure(figsize=(10,6))
state_counts = customers['State'].value_counts()
sns.barplot(x=state_counts.values, y=state_counts.index, palette='Blues_r')
plt.title('Customer Distribution by State')
plt.xlabel('Number of Customers')
plt.ylabel('State')
plt.tight_layout()
plt.show()
```

```
/tmp/ipython-input-29-3553594688.py:4: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `le

  sns.barplot(x=state_counts.values, y=state_counts.index, palette='Blues_r')
```




```
#Visualiza distribution of customers based on Gender
plt.figure(figsize=(6, 4))
gender_counts = customers['Gender'].value_counts()
sns.barplot(x=gender_counts.index, y=gender_counts.values, palette='coolwarm')
plt.title('Customer Distribution by Gender')
plt.ylabel('Number of Customers')
plt.xlabel('Gender')
plt.tight_layout()
plt.show()
```

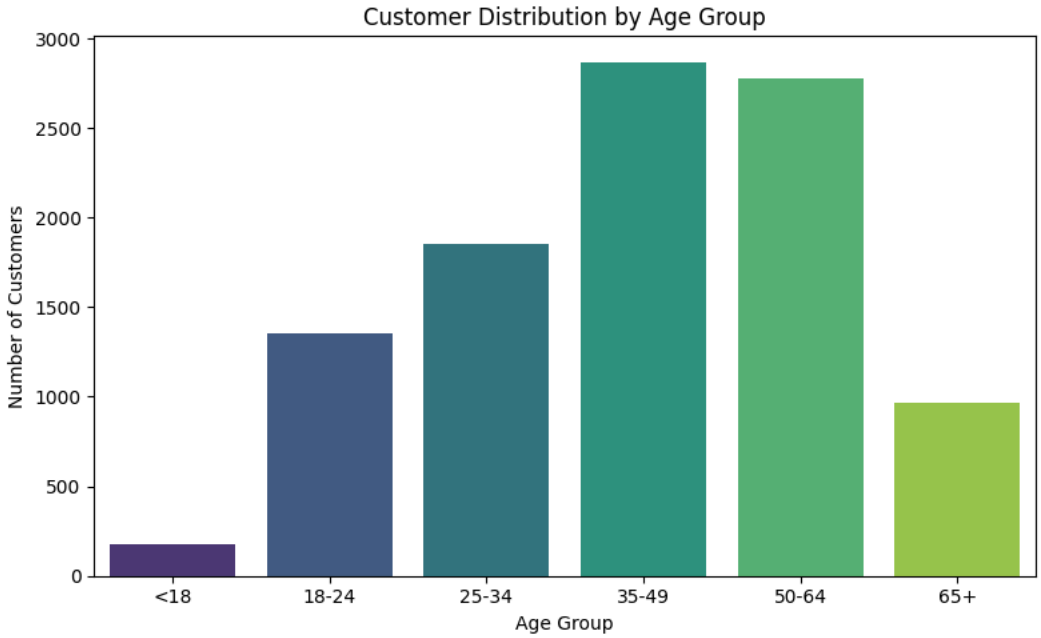
```
/tmp/ipython-input-179-296349430.py:4: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `le

  sns.barplot(x=gender_counts.index, y=gender_counts.values, palette='coolwarm')
```




```
#Visualiza distribution of customers based on age group
bins = [0, 18, 25, 35, 50, 65, 100]
labels = ['<18', '18-24', '25-34', '35-49', '50-64', '65+']
customers['age_group'] = pd.cut(customers['Age'], bins=bins, labels=labels)

plt.figure(figsize=(8, 5))
age_group_dist = customers['age_group'].value_counts().sort_index()
```

```
sns.barplot(x=age_group_dist.index, y=age_group_dist.values, palette='viridis')
plt.title('Customer Distribution by Age Group')
plt.ylabel('Number of Customers')
plt.xlabel('Age Group')
plt.tight_layout()
plt.show()
```

```
/tmp/ipython-input-33-398848677.py:8: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `le

  sns.barplot(x=age_group_dist.index, y=age_group_dist.values, palette='viridis')
```




```
#Repeat purchase rate
customer_orders = orders.groupby('C_ID').size()
repeat_customers = customer_orders[customer_orders > 1].count()
total_customers = customers['C_ID'].nunique()
repeat_purchase_rate = repeat_customers / total_customers

print("Repeat purchase rate:", round(repeat_purchase_rate * 100, 2), "%")
```

```
Repeat purchase rate: 26.33 %
```

```
#Top 10 cities/states by customers count
top_cities = customers['City'].value_counts().head(10)

print("Top 10 Cities:\n", top_cities)
```

```
Top 10 Cities:
 City
Vadodara     531
Surat        529
Jaipur       525
Kanpur       520
Patna        518
Chennai      517
Hyderabad    512
Lucknow      512
Ahmedabad    511
Mumbai       506
Name: count, dtype: int64
```

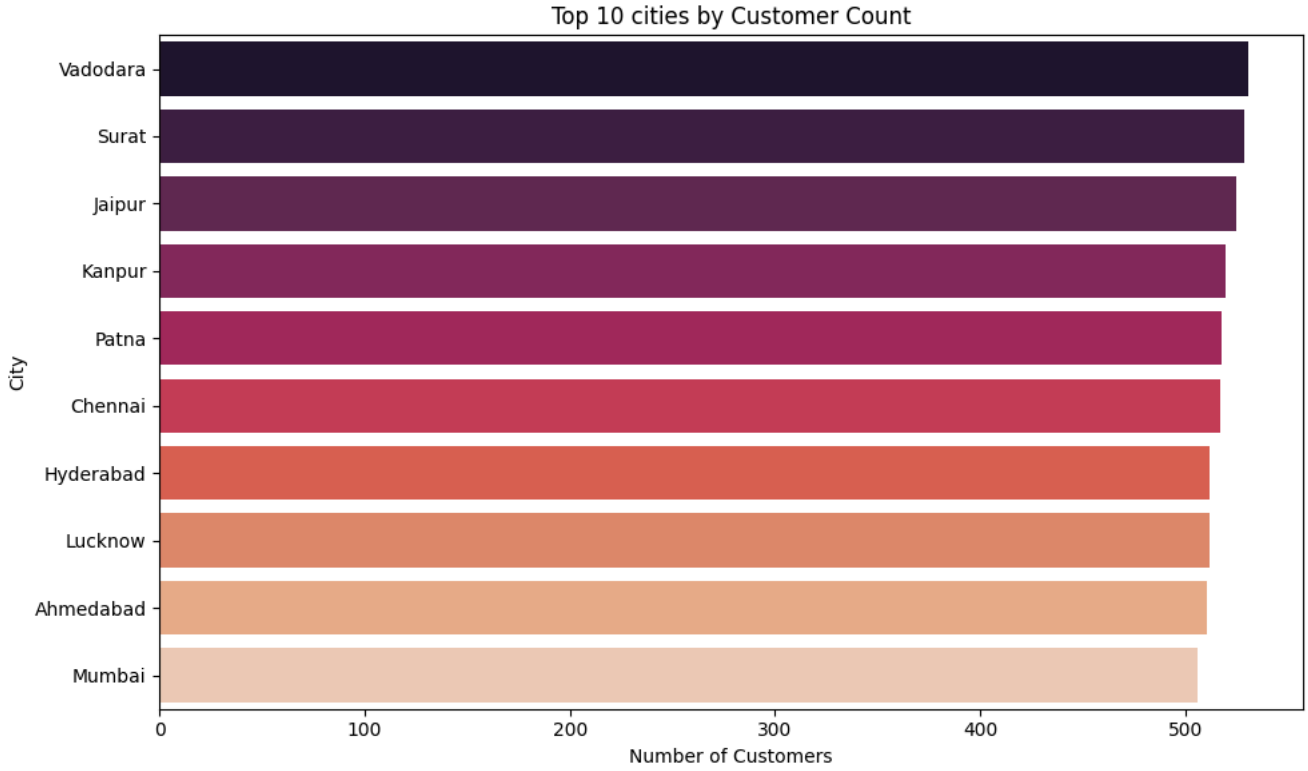
```
top_states = customers['State'].value_counts().head(10)
print("\nTop 10 States:\n", top_states)
```

```

Top 10 States:
 State
Maharashtra       1949
Gujarat           1571
Uttar Pradesh     1501
Madhya Pradesh     965
Rajasthan          525
Bihar              518
Tamil Nadu         517
Telangana          512
Andhra Pradesh     498
```

```
West Bengal        493
Name: count, dtype: int64
```

```
#Visualiza Top 10 cities/states by customers count
plt.figure(figsize=(10,6))
top_cities = customers['City'].value_counts().head(10)
sns.barplot(x=top_cities.values, y=top_cities.index, palette='rocket')
plt.title('Top 10 cities by Customer Count')
plt.xlabel('Number of Customers')
plt.ylabel('City')
plt.tight_layout()
plt.show()
```

```
/tmp/ipython-input-37-992330865.py:4: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `le

  sns.barplot(x=top_cities.values, y=top_cities.index, palette='rocket')
```

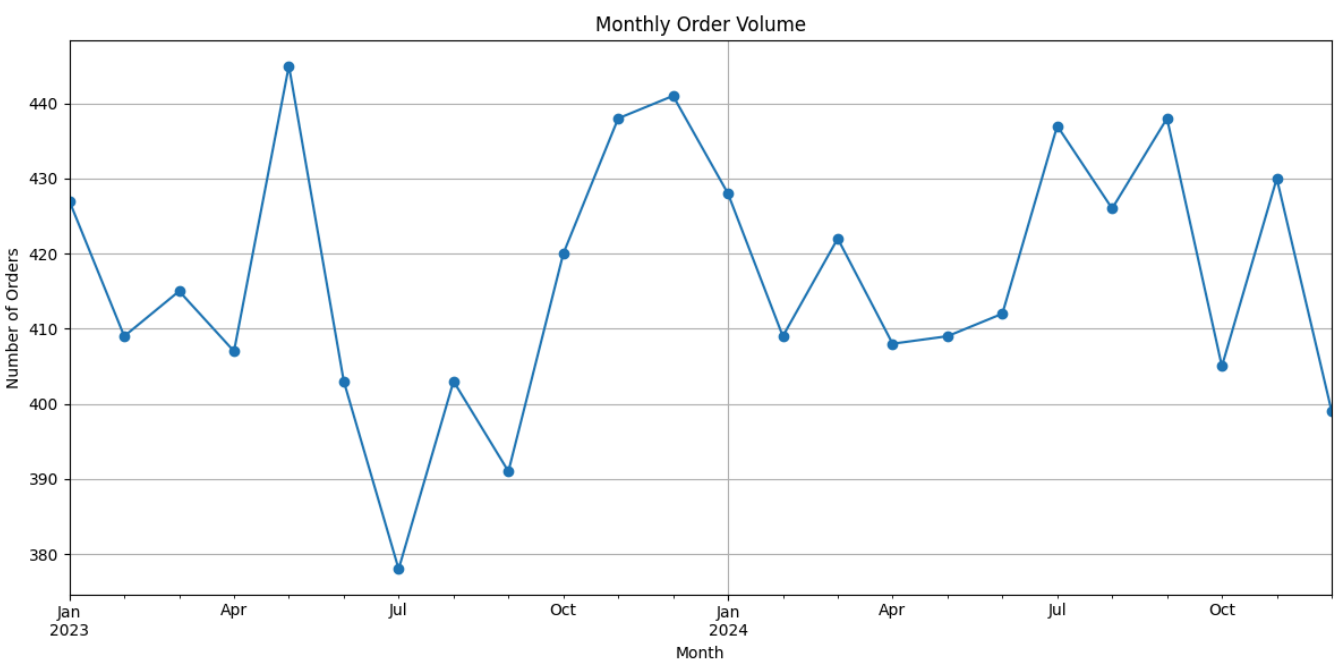



```
orders.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 13 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   Or_ID                10000 non-null  object
 1   C_ID                 10000 non-null  object
 2   P_ID                 10000 non-null  object
 3   Order_Date           10000 non-null  datetime64[ns]
 4   Order_Time           10000 non-null  object
 5   Qty                  10000 non-null  int64
 6   Coupon               10000 non-null  object
 7   DP_ID                10000 non-null  object
 8   Discount             10000 non-null  int64
 9   discount_percentage  10000 non-null  float64
 10  day_of_week          10000 non-null  object
 11  hour                 10000 non-null  int32
 12  day                  10000 non-null  object
dtypes: datetime64[ns](1), float64(1), int32(1), int64(2), object(8)
memory usage: 976.7+ KB
```

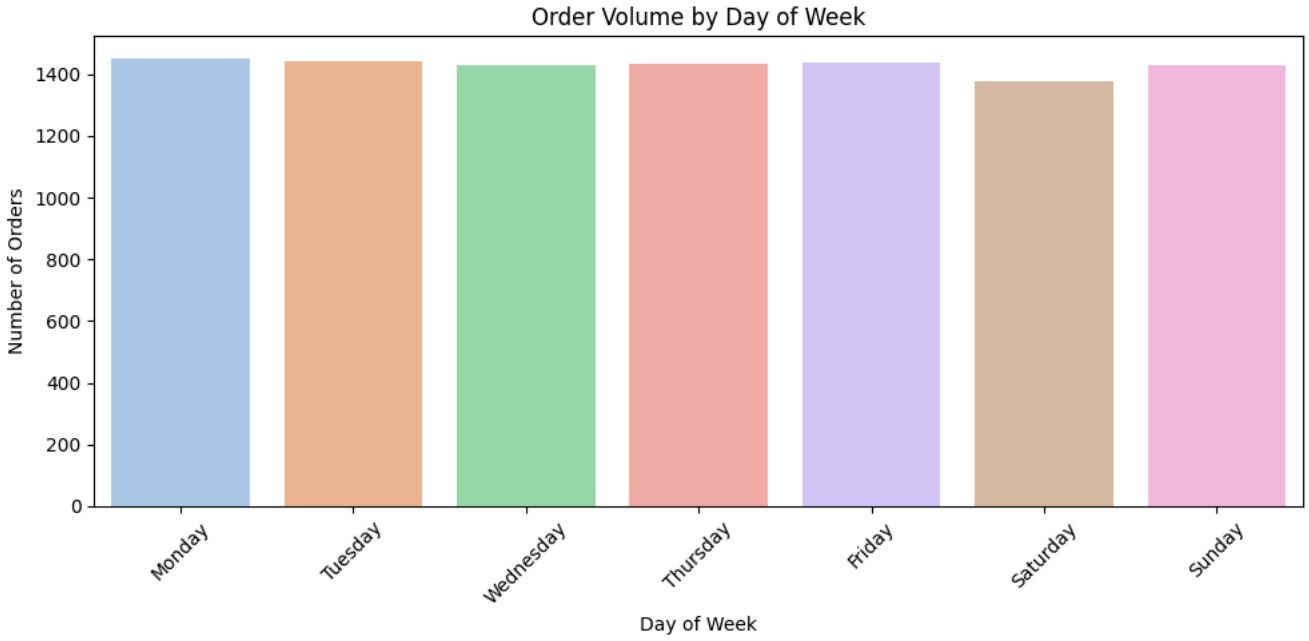
```
#Total Number of Orders Over Time
orders['Order_Date'] = pd.to_datetime(orders['Order_Date'])
orders_monthly = orders.groupby(orders['Order_Date'].dt.to_period("M")).size()
orders_monthly.index = orders_monthly.index.to_timestamp()

plt.figure(figsize=(12, 6))
orders_monthly.plot(marker='o', linestyle='-')
plt.title("Monthly Order Volume")
plt.xlabel("Month")
```

```
plt.ylabel("Number of Orders")
plt.grid(True)
plt.tight_layout()
plt.show()
```




```
#Trend: Orders by Day of Week / Time of Day
orders['day_of_week'] = orders['Order_Date'].dt.day_name()
day_order_counts = orders['day_of_week'].value_counts().reindex(
    ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday'])

plt.figure(figsize=(10, 5))
sns.barplot(x=day_order_counts.index, y=day_order_counts.values, palette='pastel')
plt.title("Order Volume by Day of Week")
plt.ylabel("Number of Orders")
plt.xlabel("Day of Week")
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

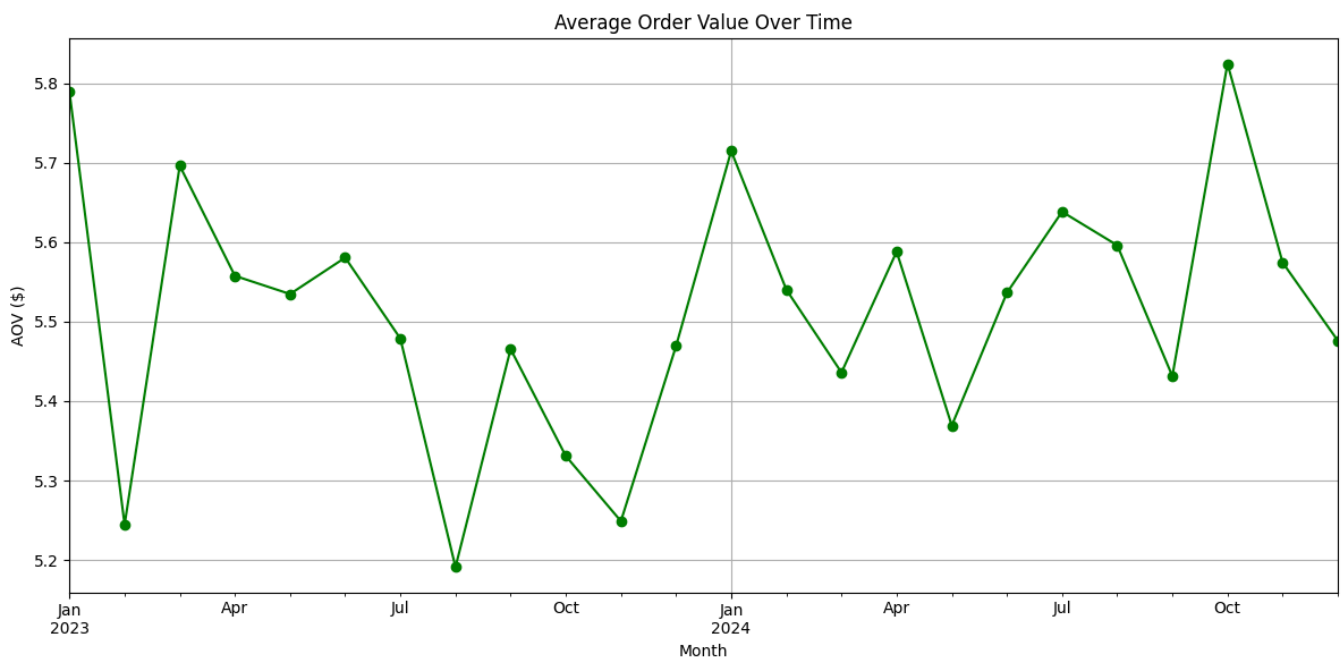
```
/tmp/ipython-input-176-1821224035.py:7: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `le

  sns.barplot(x=day_order_counts.index, y=day_order_counts.values, palette='pastel')
```




```
#Average Order Value (AOV)
aov = orders['Qty'].mean()
print("Average Order Value (AOV): ${:.2f}".format(aov))
```

```
Average Order Value (AOV): $5.51
```

```
#Visualize AOV by Month:
aov_monthly = orders.groupby(orders['Order_Date'].dt.to_period("M"))['Qty'].mean()
aov_monthly.index = aov_monthly.index.to_timestamp()

plt.figure(figsize=(12, 6))
aov_monthly.plot(marker='o', linestyle='-', color='green')
plt.title("Average Order Value Over Time")
plt.xlabel("Month")
plt.ylabel("AOV ($)")
plt.grid(True)
plt.tight_layout()
plt.show()
```




```
ratings.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 4 columns):
 #   Column                  Non-Null Count  Dtype 
---  ------                  --------------  ----- 
 0   R_ID                    10000 non-null  object
 1   Or_ID                   10000 non-null  object
 2   Prod_Rating             10000 non-null  int64 
 3   Delivery_Service_Rating  10000 non-null  int64 
dtypes: int64(2), object(2)
memory usage: 312.6+ KB
```

```
ratings.head(5)
```

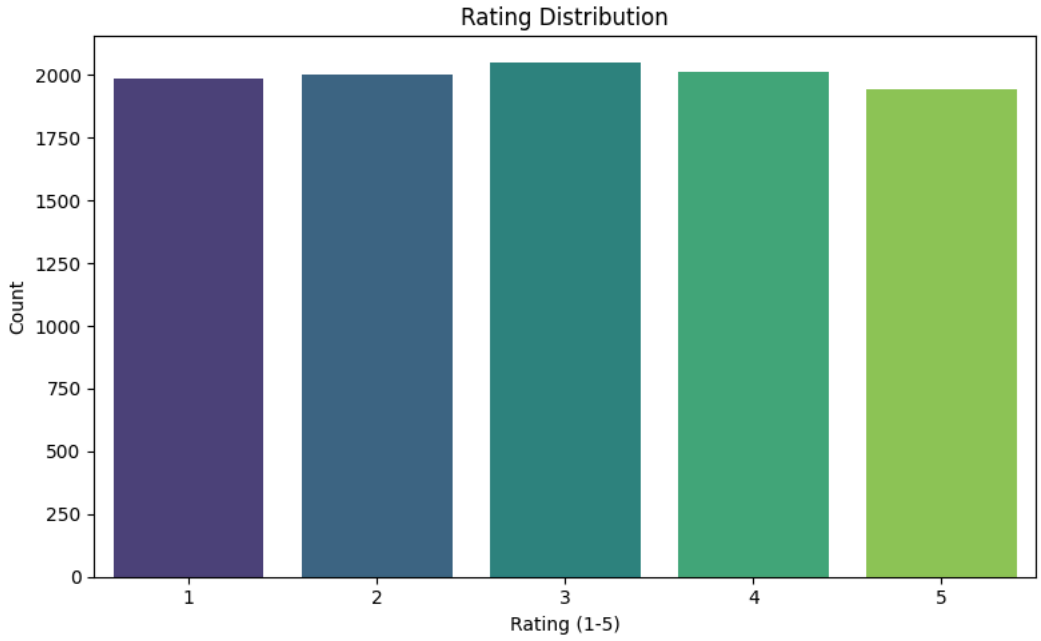
	R_ID	Or_ID	Prod_Rating	Delivery_Service_Rating
0	RT_101000001	OR_31009479	4	5
1	RT_101000002	OR_31001385	2	2
2	RT_101000003	OR_31005731	5	1
3	RT_101000004	OR_31000188	1	3
4	RT_101000005	OR_31005904	3	1

```
avg_rating = ratings['Prod_Rating'].mean()
print("Average Rating: {:.2f}".format(avg_rating))
```

```
Average Rating: 2.99
```

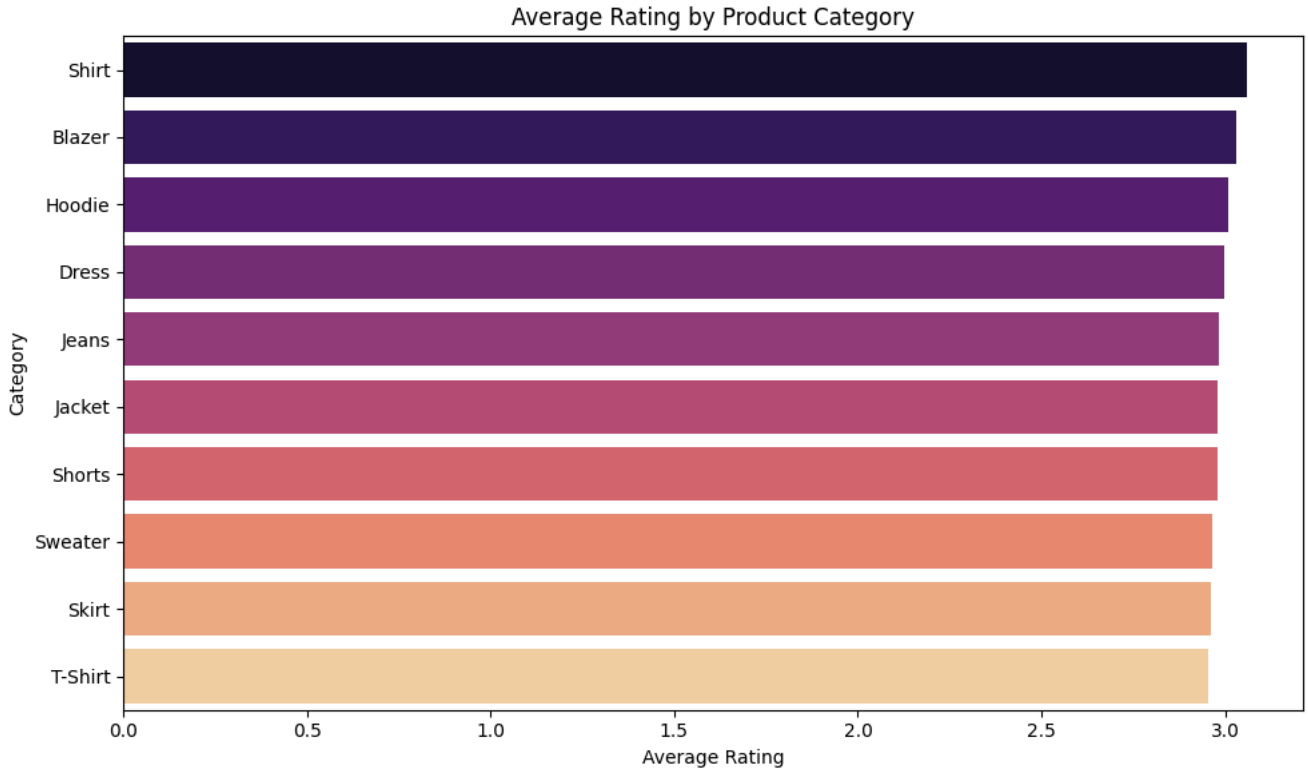
```
#Rating Distribution
plt.figure(figsize=(8, 5))
sns.countplot(x='Prod_Rating', data=ratings, palette='viridis')
plt.title("Rating Distribution")
plt.xlabel("Rating (1-5)")
plt.ylabel("Count")
plt.tight_layout()
plt.show()
```

```
/tmp/ipython-input-64-1678084318.py:2: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `le

  sns.countplot(x='Prod_Rating', data=ratings, palette='viridis')
```




```
#Average rating by product category
merged_orders_products = orders.merge(products[['P_ID', 'Category']], on='P_ID', how='left')
ratings_with_category = ratings.merge(merged_orders_products[['Or_ID', 'Category']], on='Or_ID', how='left')

avg_rating_by_category = ratings_with_category.groupby('Category')['Prod_Rating'].mean().sort_values(ascending=False)

plt.figure(figsize=(10, 6))
sns.barplot(x=avg_rating_by_category.values, y=avg_rating_by_category.index, palette='magma')
plt.title("Average Rating by Product Category")
plt.xlabel("Average Rating")
plt.ylabel("Category")
plt.tight_layout()
plt.show()
```

```
/tmp/ipython-input-67-545903006.py:8: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `le

  sns.barplot(x=avg_rating_by_category.values, y=avg_rating_by_category.index, palette='magma')
```

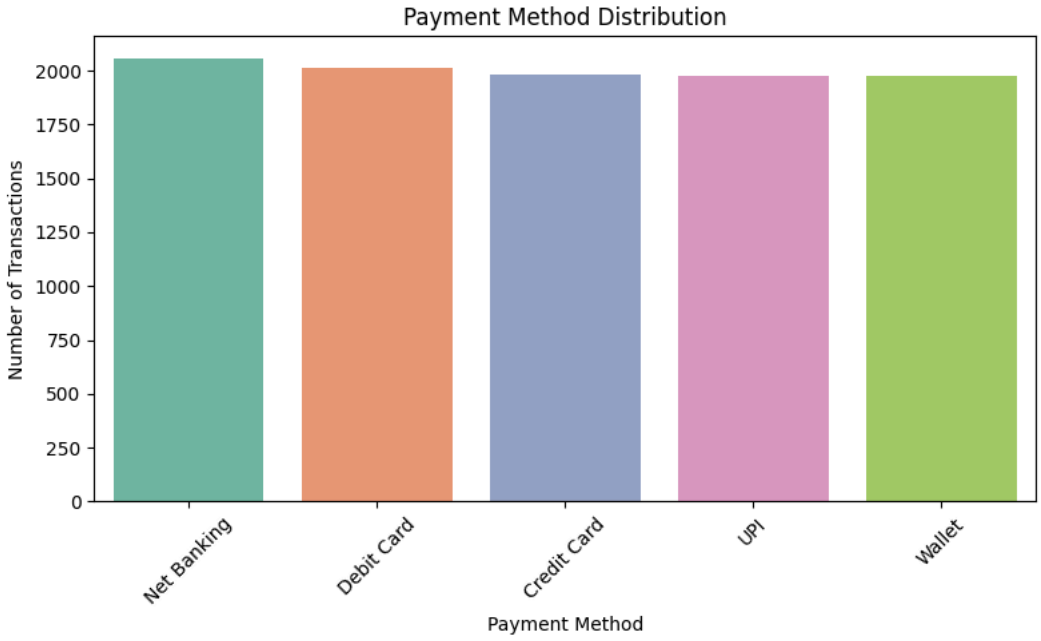



```
transactions.head()
```

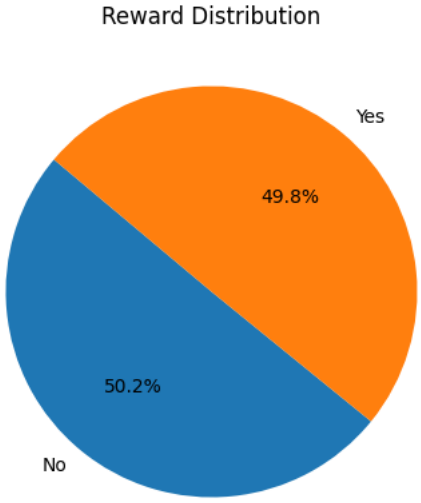
	Tr_ID	Or_ID	Transaction_Mode	Reward
0	TR_41000001	OR_31002037	Wallet	No
1	TR_41000002	OR_31008376	Wallet	Yes
2	TR_41000003	OR_31002152	UPI	No
3	TR_41000004	OR_31009239	Credit Card	Yes
4	TR_41000005	OR_31002891	Debit Card	No

```
#Mostly Used Payment Methods
payment_counts = transactions['Transaction_Mode'].value_counts()

plt.figure(figsize=(8, 5))
sns.barplot(x=payment_counts.index, y=payment_counts.values, palette='Set2')
plt.title("Payment Method Distribution")
plt.xlabel("Payment Method")
plt.ylabel("Number of Transactions")
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

```
/tmp/ipython-input-166-2876887792.py:5: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `le

  sns.barplot(x=payment_counts.index, y=payment_counts.values, palette='Set2')
```

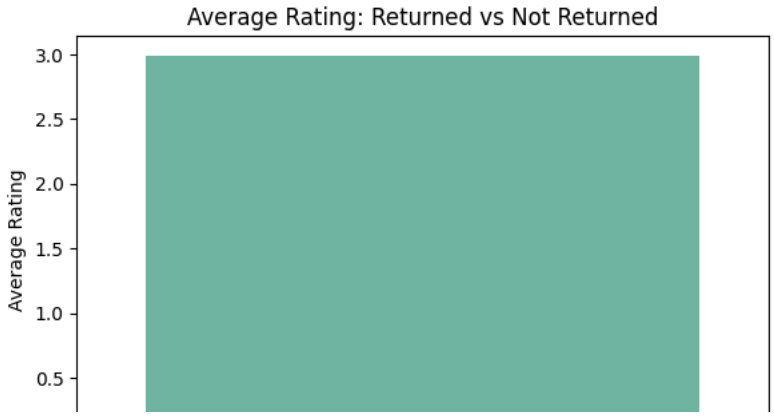



```
reward_counts = transactions['Reward'].value_counts()

plt.figure(figsize=(8, 5))
plt.pie(x=reward_counts.values, labels=reward_counts.index, autopct='%1.1f%%', startangle=140, colors=['#1f77b4', '#ff7f0e'])
plt.title("Reward Distribution")
plt.show()
```




```
ratings_returns = ratings.merge(returns[['Or_ID', 'Return_Refund']], on='Or_ID', how='left')

# Compare average rating for returned vs non-returned
ratings_returns['Return_Refund'] = ratings_returns['Return_Refund'].fillna(0)
avg_rating_returned = ratings_returns.groupby('Return_Refund')['Prod_Rating'].mean()

plt.figure(figsize=(6, 4))
sns.barplot(x=avg_rating_returned.index.map({0: 'Not Returned', 1: 'Returned'}),
            y=avg_rating_returned.values, palette='Set2')
plt.title("Average Rating: Returned vs Not Returned")
plt.ylabel("Average Rating")
plt.tight_layout()
plt.show()
```

```
/tmp/ipython-input-73-4022979002.py:8: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `le

  sns.barplot(x=avg_rating_returned.index.map({0: 'Not Returned', 1: 'Returned'}),
```




```
returns.head()
```

	RT_ID	Or_ID	Return_Reason	Return_Refund	Dates
0	RR_301000001	OR_31004141	Wrong Item Shipped	Approved	2023-04-08
1	RR_301000002	OR_31008145	Late Delivery	Rejected	2024-05-13
2	RR_301000003	OR_31005212	Wrong Item Shipped	Approved	2023-11-01
3	RR_301000004	OR_31006318	Late Delivery	Rejected	2024-05-16
4	RR_301000005	OR_31007423	Defective Product	Approved	2023-06-10

```
#Top and Bottom Rated Products
```