E-Commerce Discount Analysis

By - *Sohom Jana*

Problem Statement

Online shopping websites often give discounts to attract customers. But do discounts always increase sales? Sometimes customers buy more, but sometimes discounts can lower the average amount they spend per order.

Objectives

To check if discounts actually increase the average money spent per order.

To compare customer behavior during discount periods vs. regular periods.

To find out if there is an optimal discount range that helps businesses earn more.

Knowing the Dataset

The key columns are:

- **Order_Date** → Date when the order was placed.
- **Time** → Exact time of purchase.
- **Customer_Id** → Unique ID for each customer.
- **Gender** → Male or Female.
- **Device_Type** → Whether the order was placed via Web or Mobile.
- **Customer_Login_type** → Guest or Member.
- **Product_Category & Product** → Type and name of the purchased product.
- **Sales** → Selling price of the product.
- **Quantity** → Number of units purchased.
- **Discount** → Discount given (in decimal, e.g., 0.2 = 20%).
- **Profit** → Profit earned from the order.
- **Shipping_Cost** → Cost of shipping.
- **Order_Priority** → Priority level of the order (High, Medium, Critical, etc.).
- **Payment_method** → Mode of payment (Credit Card, etc.).

Analysis Begins

```
import pandas as pd
import numpy as np
import plotly.express as px
```

```
df = pd.read_csv("/content/E-commerce Dataset.csv")
```

```
df.head()
```

	Order_Date	Time	Aging	Customer_Id	Gender	Device_Type	Customer_Login_type	Product_Category	Product	Sales	Quantity	Discount	Profit	Shipping_Cost	Order_Priority	Pay
0	2018-01-02	10:56:33	8.0	37077	Female	Web	Member	Auto & Accessories	Car Media Players	140.0	1.0	0.3	46.0	4.6	Medium	
1	2018-07-24	20:41:37	2.0	59173	Female	Web	Member	Auto & Accessories	Car Speakers	211.0	1.0	0.3	112.0	11.2	Medium	
2	2018-11-08	08:38:49	8.0	41066	Female	Web	Member	Auto & Accessories	Car Body Covers	117.0	5.0	0.1	31.2	3.1	Critical	
3	2018-04-18	19:28:06	7.0	50741	Female	Web	Member	Auto & Accessories	Car & Bike Care	118.0	1.0	0.3	26.2	2.6	High	
4	2018-08-13	21:18:39	9.0	53639	Female	Web	Member	Auto & Accessories	Tyre	250.0	1.0	0.3	160.0	16.0	Critical	

Next steps: Generate code with df | View recommended plots | New interactive sheet

Data Cleaning and Preprocessing

```
df['Order_Date'] = pd.to_datetime(df['Order_Date'], errors='coerce')
```

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

	0
Order_Date	0
Time	0
Aging	1
Customer_Id	0
Gender	0
Device_Type	0
Customer_Login_type	0
Product_Category	0
Product	0
Sales	1
Quantity	2
Discount	1
Profit	0
Shipping_Cost	1
Order_Priority	2
Payment_method	0

dtype: int64

```
df = df.dropna(subset=['Sales', 'Quantity', 'Discount', 'Shipping_Cost', 'Order_Priority'])
```

```
df['Aging'] = df['Aging'].fillna(df['Aging'].median())
```

```
df = df.reset_index(drop=True)
```

```
df['Total_Order_Value'] = df['Sales'] * df['Quantity'] * (1 - df['Discount'])
```

```
df['Month'] = df['Order_Date'].dt.month
df['DayOfWeek'] = df['Order_Date'].dt.day_name()
```

```
def get_time_of_day(x):
    hour = int(x.split(":")[0])
    if 5 <= hour < 12:
        return "Morning"
    elif 12 <= hour < 17:
        return "Afternoon"
    elif 17 <= hour < 21:
        return "Evening"
    else:
        return "Night"

df['TimeOfDay'] = df['Time'].apply(get_time_of_day)
```

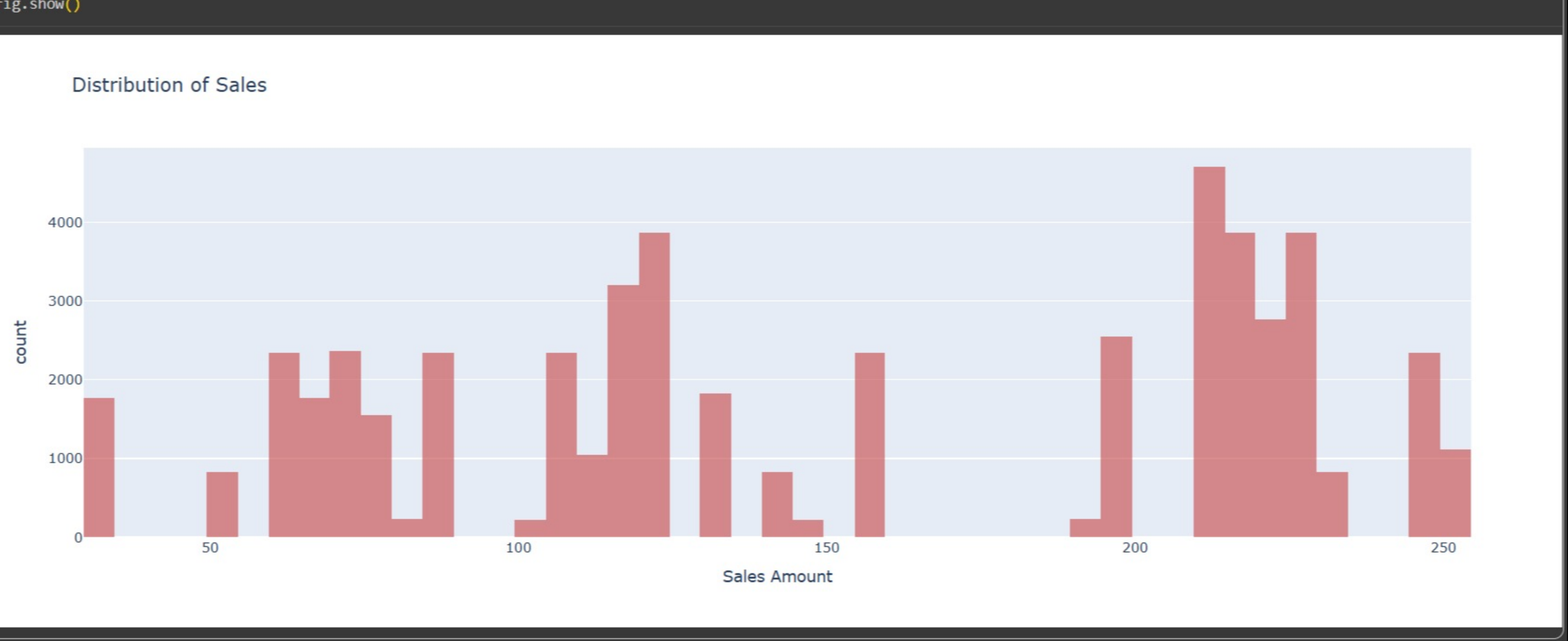
```
df.head()
```

	Order_Date	Time	Aging	Customer_Id	Gender	Device_Type	Customer_Login_type	Product_Category	Product	Sales	Quantity	Discount	Profit	Shipping_Cost	Order_Priority	Pay
0	2018-01-02	10:56:33	8.0	37077	Female	Web	Member	Auto & Accessories	Car Media Players	140.0	1.0	0.3	46.0	4.6	Medium	
1	2018-07-24	20:41:37	2.0	59173	Female	Web	Member	Auto & Accessories	Car Speakers	211.0	1.0	0.3	112.0	11.2	Medium	
2	2018-11-08	08:38:49	8.0	41066	Female	Web	Member	Auto & Accessories	Car Body Covers	117.0	5.0	0.1	31.2	3.1	Critical	
3	2018-04-18	19:28:06	7.0	50741	Female	Web	Member	Auto & Accessories	Car & Bike Care	118.0	1.0	0.3	26.2	2.6	High	
4	2018-08-13	21:18:39	9.0	53639	Female	Web	Member	Auto & Accessories	Tyre	250.0	1.0	0.3	160.0	16.0	Critical	

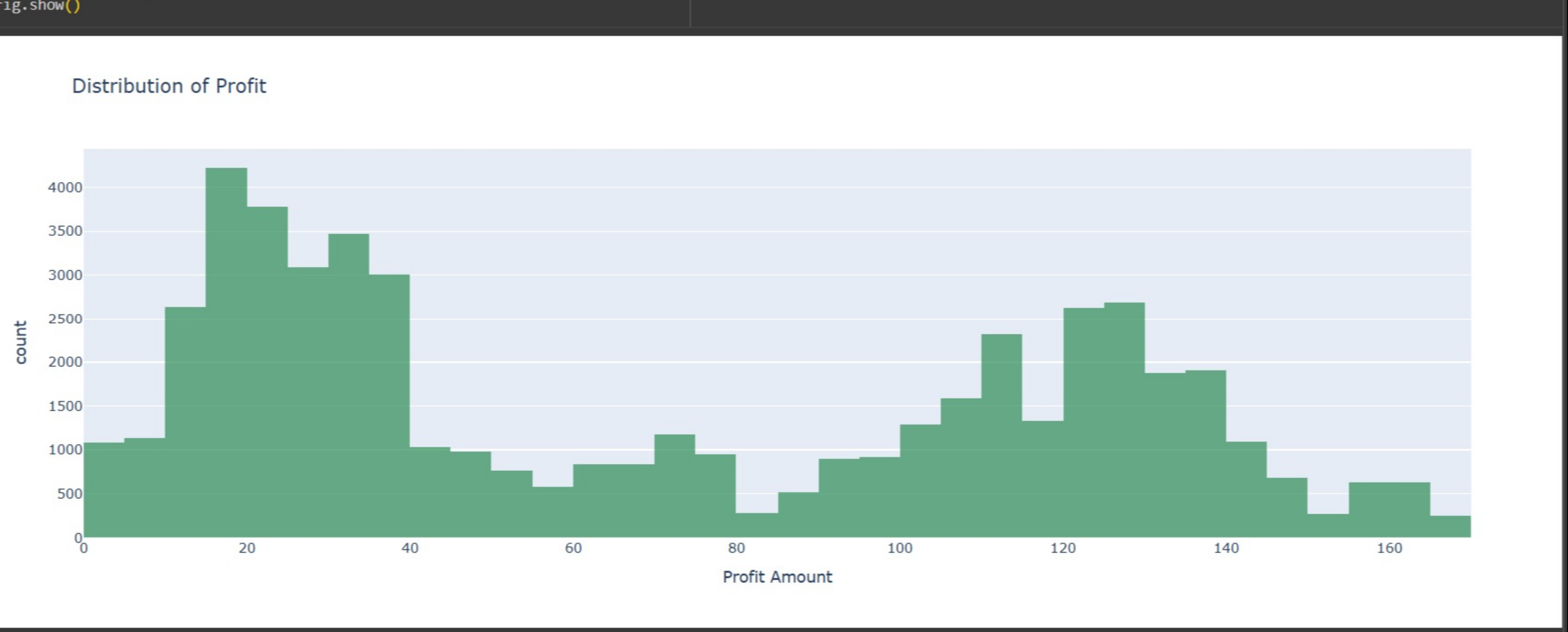
Next steps: Generate code with df | View recommended plots | New interactive sheet

Analysis with Plots

```
# Distribution of Sales
fig = px.histogram(df, x="Sales", nbins=50, title="Distribution of Sales",
                   labels={"Sales": "Sales Amount"},
                   color_discrete_sequence=['indianred'])
fig.update_traces(opacity=0.7)
fig.show()
```

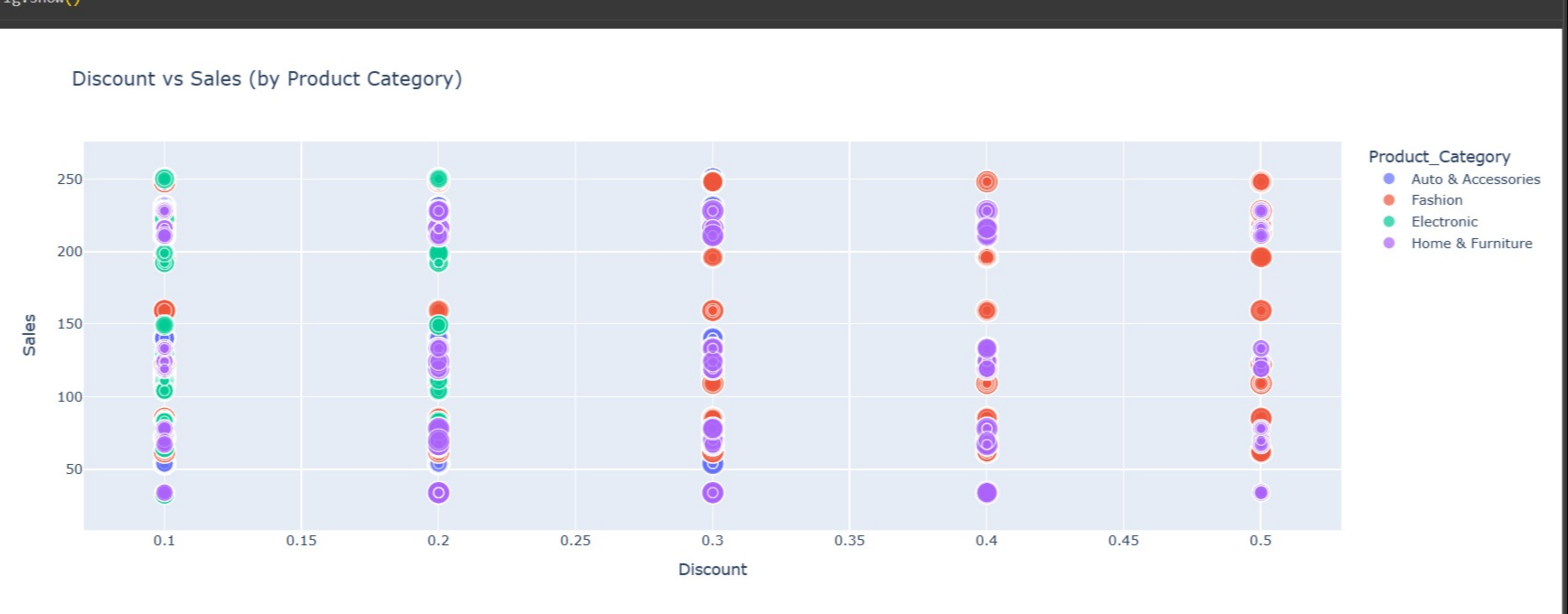



```
# Distribution of Profit
fig = px.histogram(df, x="Profit", nbins=50, title="Distribution of Profit",
                   labels={"Profit": "Profit Amount"},
                   color_discrete_sequence=['seagreen'])
fig.update_traces(opacity=0.7)
fig.show()
```

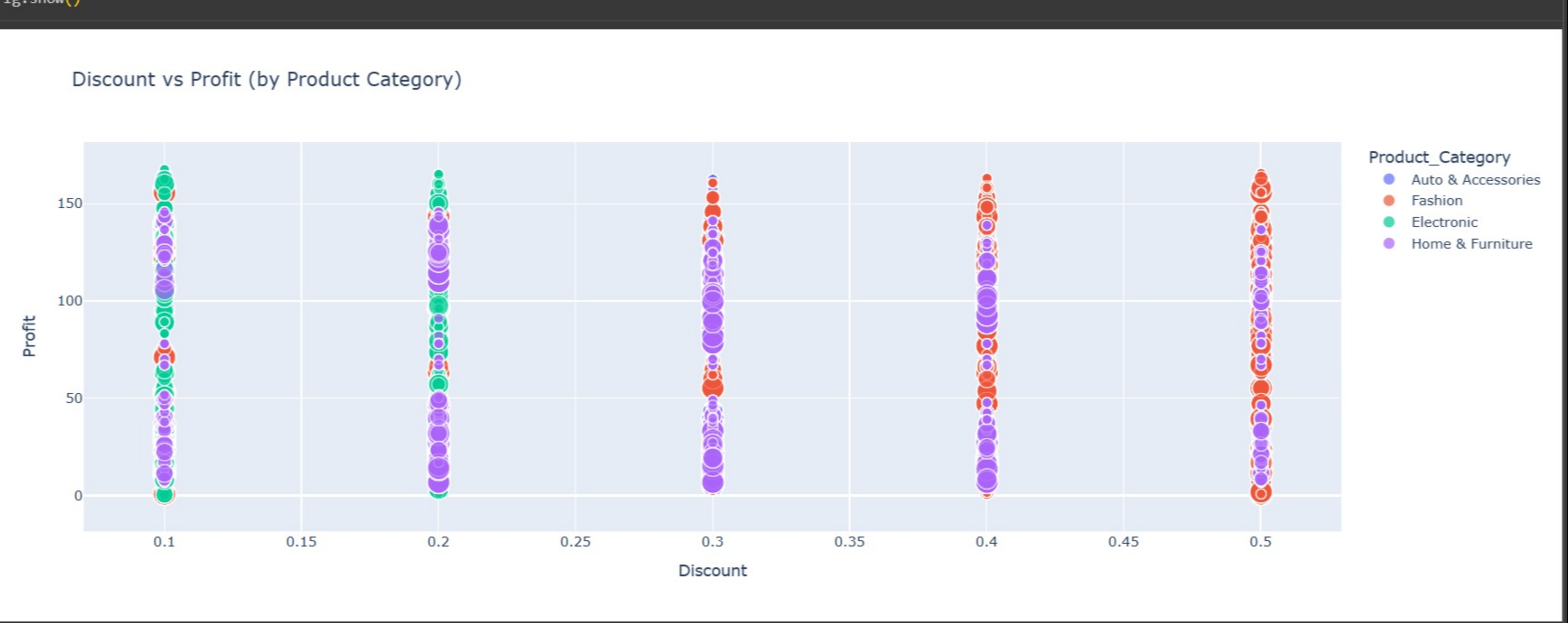



```
# Distribution of Discounts
fig = px.histogram(df, x="Discount", nbins=20, title="Distribution of Discounts",
                   labels={"Discount": "Discount Rate"},
                   color_discrete_sequence=['royalblue'])
fig.update_traces(opacity=0.7)
fig.show()
```


```
#  IMP: Discount vs Sales
fig = px.scatter(df, x="Discount", y="Sales",
                 size="Quantity", color="Product_Category",
                 hover_data=["Profit", "Product"],
                 title="Discount vs Sales (by Product Category)")
fig.show()
```

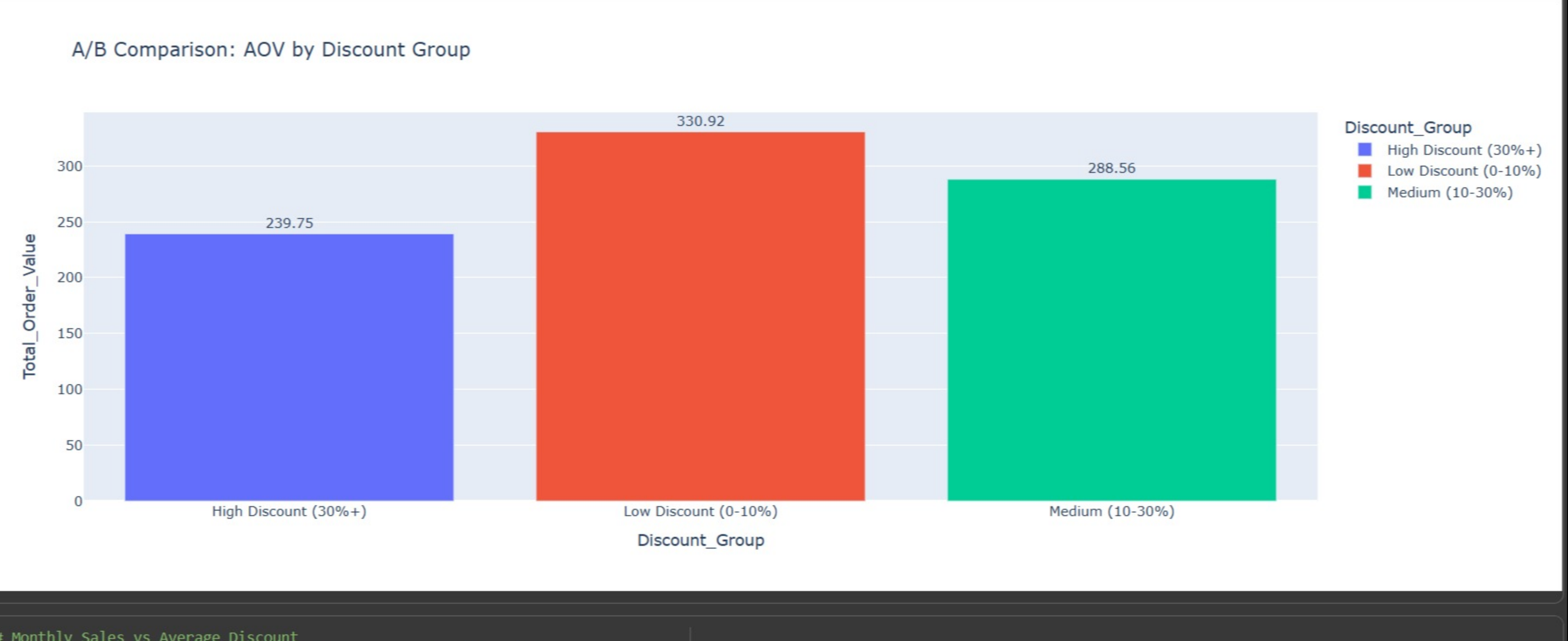



```
# IMP: Discount vs Profit
fig = px.scatter(df, x="Discount", y="Profit",
                 size="Quantity", color="Product_Category",
                 hover_data=["Sales", "Product"],
                 title="Discount vs Profit (by Product Category)")
fig.show()
```

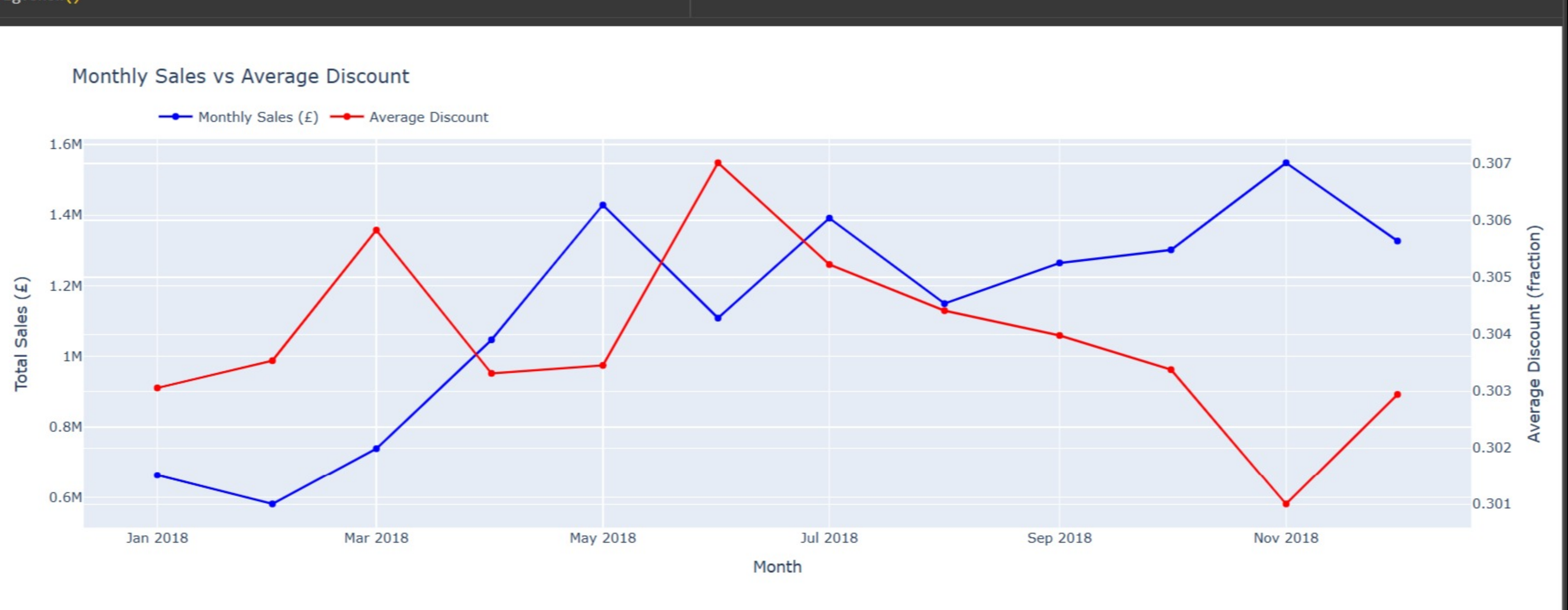



```
# IMP: Average Order Value by Discount ranges
df['Discount_Bucket'] = pd.cut(df['Discount'], bins=[0,0.1,0.2,0.3,0.5,1.0],
                               labels=["0-10%", "10-20%", "20-30%", "30-50%", "50%+"])

fig = px.box(df, x="Discount_Bucket", y="Total_Order_Value",
             color="Discount_Bucket",
             title="Average Order Value Across Discount Buckets",
             labels={"Total_Order_Value": "Order Value"})
fig.show()
```


```
# Define A/B groups: Low discount (<=10%) vs High discount (>=30%)
df['Discount_Group'] = np.where(df['Discount'] <= 0.1, "Low Discount (0-10%)",
                                np.where(df['Discount'] >= 0.3, "High Discount (30%+)", "Medium (10-30%)"))

# Compare Average Order Value by Discount Group
fig = px.bar(df.groupby("Discount_Group")["Total_Order_Value"].mean().reset_index(),
             x="Discount_Group", y="Total_Order_Value", color="Discount_Group",
             text="Total_Order_Value", title="A/B Comparison: AOV by Discount Group")
fig.update_traces(texttemplate='%{text:.2f}', textposition='outside')
fig.show()
```




```
# Monthly Sales vs Average Discount
monthly = df.groupby(df['Order_Date'].dt.to_period("M")).agg({
    "Total_Order_Value": "sum",
    "Discount": "mean"
}).reset_index()

monthly['Order_Date'] = monthly['Order_Date'].astype(str)

# Dual-axis plot with Plotly
import plotly.graph_objects as go

fig = go.Figure()

# Line for Sales
fig.add_trace(go.Scatter(x=monthly['Order_Date'], y=monthly['Total_Order_Value'],
                         mode='lines+markers', name='Monthly Sales (£)', line=dict(color='blue')))

# Line for Discount
fig.add_trace(go.Scatter(x=monthly['Order_Date'], y=monthly['Discount'],
                         mode='lines+markers', name='Average Discount', line=dict(color='red'), yaxis='y2'))

# Layout with two axes
fig.update_layout(
    title="Monthly Sales vs Average Discount",
    xaxis=dict(title="Month"),
    yaxis=dict(title="Total Sales (£)", side="left"),
    yaxis2=dict(title="Average Discount (fraction)", overlaying="y", side="right"),
    legend=dict(x=0.05, y=1.1, orientation="h")
)

fig.show()
```



Conclusion:

- Discounts mainly fall in the **10–30% range**.
- **Sales rise** with discounts but **profit drops** sharply at high levels.
- **Best discount range: 10–20%** → highest average order value (AOV).
- Very high discounts (>40%) reduce both profit and AOV.