Data Analyst

E-Commerce Sales Analyzation

By
Jatin Patidar

Summary

In this Python-based E-commerce analysis project, I started by cleaning and understanding the dataset using functions like head(), info(), and describe() to check data structure, data types, and summary statistics. The Order Date and Ship Date columns were converted from object type to datetime format to enable accurate time-based analysis.

From the Order Date, I extracted new features such as Month, Year, and Day of Week, which helped in identifying sales and profit trends over time. After data preparation, I performed data visualization using Python libraries to analyze sales and profit patterns across different time periods, categories, and sub-categories.

The visual insights clearly highlighted seasonal trends, top-performing categories, and profitable sub-categories, helping in better business decision-making related to sales planning, inventory management, and profit optimization.

Context

This project focuses on analyzing E-commerce sales data to understand sales performance, profit trends, and customer purchasing behavior. The dataset includes information such as order dates, sales, profit, categories, sub-categories, and customer segments.

The data was first cleaned and preprocessed using Pandas and NumPy, where missing values were handled, column data types were corrected, and Order Date and Ship Date were converted into datetime format. Time-based features like month, year, and day of the week were extracted to support trend analysis.

For data visualization, both Matplotlib/Seaborn and the Plotly library were used. Plotly was mainly used to create interactive charts such as monthly sales trends, category-wise profit distribution, segment-wise sales analysis, and sub-category comparisons. These interactive dashboards allow users to hover, zoom, and filter data for better understanding.

The insights from this analysis help identify top-performing months, high-profit categories, low-performing sub-categories, and seasonal sales patterns, supporting better business decision-making.

library and Dataframe information with code -

Importing libraries

[2]:
```
import pandas as pd
import plotly.express as px
import plotly.io as pio
import plotly.colors as colors
pio.templates.default = "plotly_white"
```

[3]:
```
data = pd.read_csv(r"C:\Users\a\Downloads\Sample - Superstore.csv", encoding='latin-1')
```

[4]:
```
data.head(5)
```

[4]:

Row ID	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name	Segment	Country	City	...	Postal Code	Region	Product ID	Category	Sub-Category	Product Name
1	CA-2016-152156	11/8/2016	11/11/2016	Second Class	CG-12520	Claire Gute	Consumer	United States	Henderson	...	42420	South	FUR-BO-10001798	Furniture	Bookcases	Bush Somerset Collection Bookcase
2	CA-2016-152156	11/8/2016	11/11/2016	Second Class	CG-12520	Claire Gute	Consumer	United States	Henderson	...	42420	South	FUR-CH-10000454	Furniture	Chairs	Hon Deluxe Fabric Upholstered Stacking Chairs,...
3	CA-2016-138688	6/12/2016	6/16/2016	Second Class	DV-13045	Darrin Van Huff	Corporate	United States	Los Angeles	...	90036	West	OFF-LA-10000240	Office Supplies	Labels	Self-Adhesive Address Labels for Typewriters b...
4	US-2015-108966	10/11/2015	10/18/2015	Standard Class	SO-20335	Sean O'Donnell	Consumer	United States	Fort Lauderdale	...	33311	South	FUR-TA-10000577	Furniture	Tables	Bretford CR4500 Series Slim Rectangular Table

Let’s start by looking at the descriptive statistics of the dataset –

[5]:
```
data.describe()
```

[5]:

	Row ID	Postal Code	Sales	Quantity	Discount	Profit
count	9994.000000	9994.000000	9994.000000	9994.000000	9994.000000	9994.000000
mean	4997.500000	55190.379428	229.858001	3.789574	0.156203	28.656896
std	2885.163629	32063.693350	623.245101	2.225110	0.206452	234.260108
min	1.000000	1040.000000	0.444000	1.000000	0.000000	-6599.978000
25%	2499.250000	23223.000000	17.280000	2.000000	0.000000	1.728750
50%	4997.500000	56430.500000	54.490000	3.000000	0.200000	8.666500
75%	7495.750000	90008.000000	209.940000	5.000000	0.200000	29.364000
max	9994.000000	99301.000000	22638.480000	14.000000	0.800000	8399.976000

Dataframe information with code -

```
[6]: data.info()
```

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

```
[7]: data.columns
```

```
[7]: Index(['Row ID', 'Order ID', 'Order Date', 'Ship Date', 'Ship Mode',
       'Customer ID', 'Customer Name', 'Segment', 'Country', 'City', 'State',
       'Postal Code', 'Region', 'Product ID', 'Category', 'Sub-Category',
       'Product Name', 'Sales', 'Quantity', 'Discount', 'Profit'],
      dtype='object')
```

convert date columns in pandas-

Converting date columns

[8]:
```
data['Order date'] = pd.to_datetime(data['Order Date'])
data['Ship date'] = pd.to_datetime(data['Ship Date'])
```

[9]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9994 entries, 0 to 9993
Data columns (total 23 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Row ID         9994 non-null   int64
 1   Order ID       9994 non-null   object
 2   Order Date     9994 non-null   object
 3   Ship Date      9994 non-null   object
 4   Ship Mode      9994 non-null   object
 5   Customer ID    9994 non-null   object
 6   Customer Name  9994 non-null   object
 7   Segment        9994 non-null   object
 8   Country        9994 non-null   object
 9   City           9994 non-null   object
 10  State          9994 non-null   object
 11  Postal Code    9994 non-null   int64
 12  Region         9994 non-null   object
 13  Product ID     9994 non-null   object
 14  Category       9994 non-null   object
 15  Sub-Category   9994 non-null   object
 16  Product Name   9994 non-null   object
 17  Sales          9994 non-null   float64
 18  Quantity       9994 non-null   int64
 19  Discount       9994 non-null   float64
 20  Profit         9994 non-null   float64
 21  Order date     9994 non-null   datetime64[ns]
 22  Ship date      9994 non-null   datetime64[ns]
dtypes: datetime64[ns](2), float64(3), int64(3), object(15)
memory usage: 1.8+ MB
```

Adding New Date-Based Columns -

[35]:
```
data['Order Date'] = pd.to_datetime(data['Order Date'], errors='coerce')

# Now you can safely extract parts of the date
data['Order Month'] = data['Order Date'].dt.month
data['Order Year'] = data['Order Date'].dt.year
data['Order Day of Week'] = data['Order Date'].dt.dayofweek
```

[36]:
```
data.head()
```

[36]:

Ship Date	Ship Mode	Customer ID	Customer Name	Segment	Country	City	...	Product Name	Sales	Quantity	Discount	Profit	Order date	Ship date	Order Month	Order Year	Order Day of Week
1/11/2016	Second Class	CG-12520	Claire Gute	Consumer	United States	Henderson	...	Bush Somerset Collection Bookcase	261.9600	2	0.00	41.9136	2016-11-08	2016-11-11	11	2016	1
1/11/2016	Second Class	CG-12520	Claire Gute	Consumer	United States	Henderson	...	Hon Deluxe Fabric Upholstered Stacking Chairs,...	731.9400	3	0.00	219.5820	2016-11-08	2016-11-11	11	2016	1
6/16/2016	Second Class	DV-13045	Darrin Van Huff	Corporate	United States	Los Angeles	...	Self-Adhesive Address Labels for Typewriters b...	14.6200	2	0.00	6.8714	2016-06-12	2016-06-16	6	2016	6
0/18/2015	Standard Class	SO-20335	Sean O'Donnell	Consumer	United States	Fort Lauderdale	...	Bretford CR4500 Series Slim Rectangular Table	957.5775	5	0.45	-383.0310	2015-10-11	2015-10-18	10	2015	6

Monthly Sales Analysis-

```
[93]: sales_by_month = data.groupby('Order Month')['Sales'].sum().reset_index()
      print(sales_by_month)
```

	Order Month	Sales
0	1	94924.8356
1	2	59751.2514
2	3	205005.4888
3	4	137762.1286
4	5	155028.8117
5	6	152718.6793
6	7	147238.0970
7	8	159044.0630
8	9	307649.9457
9	10	200322.9847
10	11	352461.0710
11	12	325293.5035

```
fig = px.line(
    sales_by_month,
    x='Order Month',
    y='Sales',
    title='Monthly Sales Analysis'
)
fig.show()
```

Key Finding -

- **This graph shows total sales for each month.**
- **It helps identify seasonal sales trends.**
- **The highest sales month indicates peak customer demand.**
- **The lowest sales month highlights periods that need promotional strategies.**

Sales Analysis by Category -

```
[96]: Sales_by_category = data.groupby('Category')['Sales'].sum().reset_index()
```

```
[97]: Sales_by_category
```

[97]:

	Category	Sales
0	Furniture	741999.7953
1	Office Supplies	719047.0320
2	Technology	836154.0330

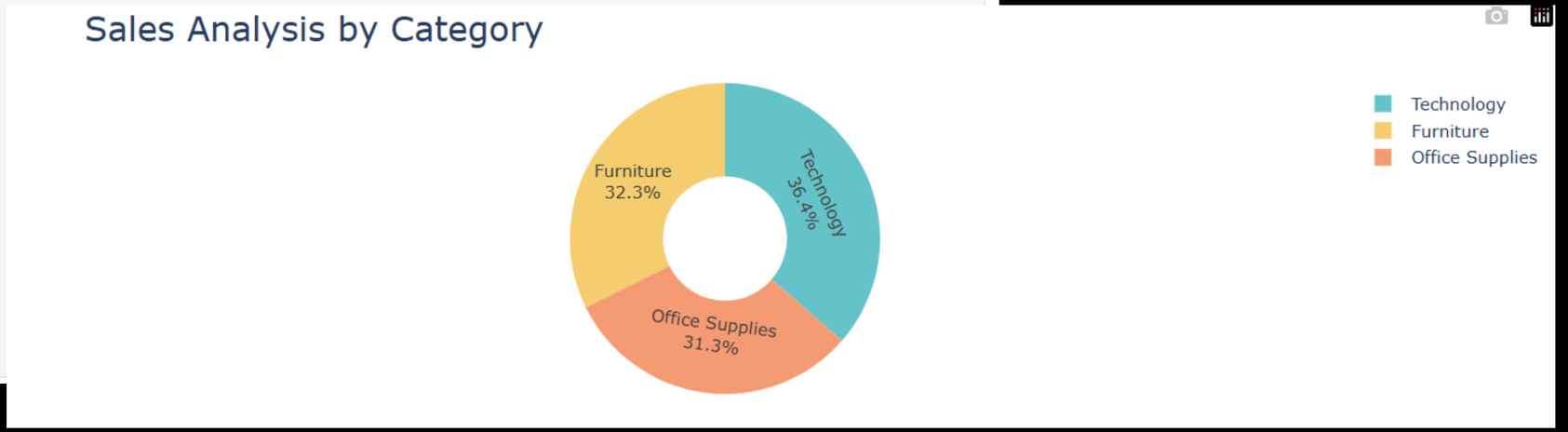
```
[98]: Sales_by_category = data.groupby('Category')['Sales'].sum().reset_index()

fig = px.pie(
    Sales_by_category,
    values='Sales',
    names='Category',
    hole=0.4,
    color_discrete_sequence=px.colors.qualitative.Pastel
)

fig.update_traces(
    textposition='inside',
    textinfo='percent+label'
)

fig.update_layout(
    title_text='Sales Analysis by Category',
    title_font=dict(size=24)
)

fig.show()
```



Key Finding -

- **This visualization compares sales across different product categories.**
- **It shows which category contributes the most and least to total sales.**
- **High-performing categories are key revenue drivers.**
- **Low-performing categories may need improvement or better marketing.**

Sales Analysis by Sub Category -

```
[100]: Sales_by_subcategory = data.groupby('Sub-Category')['Sales'].sum().reset_index()
       Sales_by_subcategory
```

[100]:

	Sub-Category	Sales
0	Accessories	167380.3180
1	Appliances	107532.1610
2	Art	27118.7920
3	Binders	203412.7330
4	Bookcases	114879.9963
5	Chairs	328449.1030
6	Copiers	149528.0300
7	Envelopes	16476.4020
8	Fasteners	3024.2800
9	Furnishings	91705.1640
10	Labels	12486.3120
11	Machines	189238.6310
12	Paper	78479.2060
13	Phones	330007.0540
14	Storage	223843.6080
15	Supplies	46673.5380
16	Tables	206965.5320

```
Sales_by_subcategory = (
    data.groupby('Sub-Category')['Sales']
    .sum()
    .reset_index()
    .sort_values(by='Sales', ascending=False))
fig = px.bar(
    Sales_by_subcategory,
    x='Sub-Category',
    y='Sales',
    title="Sales Analysis by Sub-Category (Sorted by Sales)")
fig.show()
```

Key Finding -

- **This graph gives a detailed breakdown of sales within each category.**
- **It helps identify top-selling and low-selling products.**
- **Useful for product-level analysis and inventory planning.**

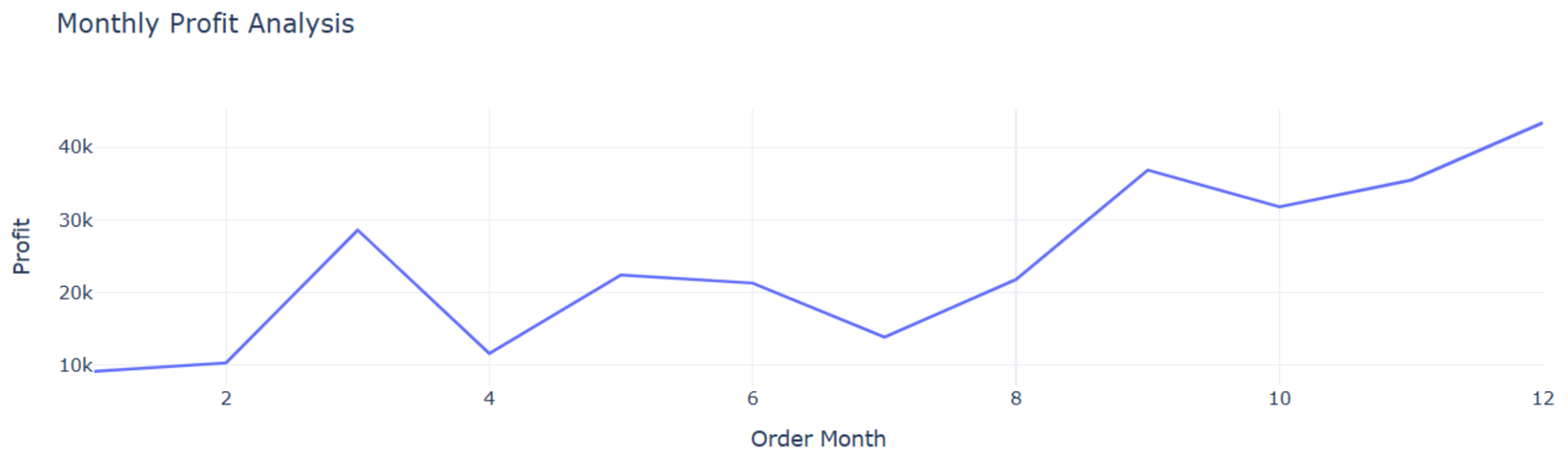
Monthly Profit Analysis -

```
[37]: Profit_by_month = data.groupby('Order Month')['Profit'].sum().reset_index()
      Profit_by_month
```

[37]:

	Order Month	Profit
0	1	9134.4461
1	2	10294.6107
2	3	28594.6872
3	4	11587.4363
4	5	22411.3078
5	6	21285.7954
6	7	13832.6648
7	8	21776.9384
8	9	36857.4753
9	10	31784.0413
10	11	35468.4265
11	12	43369.1919

```
]: Profit_by_month = data.groupby('Order Month')['Profit'].sum().reset_index()
   fig = px.line(Profit_by_month, x = 'Order Month' , y='Profit', title = 'Monthly Profit Analysis')
   fig.show()
```



Key Finding -

- **This graph shows how profit changes month by month.**
- **It identifies which month generated the highest and lowest profit.**
- **Highlights the difference between high sales and high profit months.**
- **Useful for cost control and discount planning.**

Profit by Category -

```
[38]: Profit_by_category = data.groupby('Category')['Profit'].sum().reset_index()
      Profit_by_category
```

[38]:

	Category	Profit
0	Furniture	18451.2728
1	Office Supplies	122490.8008
2	Technology	145454.9481

```
[107]: Profit_by_category = data.groupby('Category')['Profit'].sum().reset_index()

fig = px.pie(
    Profit_by_category,
    values='Profit',
    names='Category',
    hole=0.5,
    color_discrete_sequence=px.colors.qualitative.Pastel)


fig.update_traces(
    textposition='inside',
    textinfo='percent+label'
)

fig.update_layout(
    title_text='Profit Analysis by Category',
    title_font=dict(size=24),
    title_x=0.5
)

fig.show()
```

Key Finding -

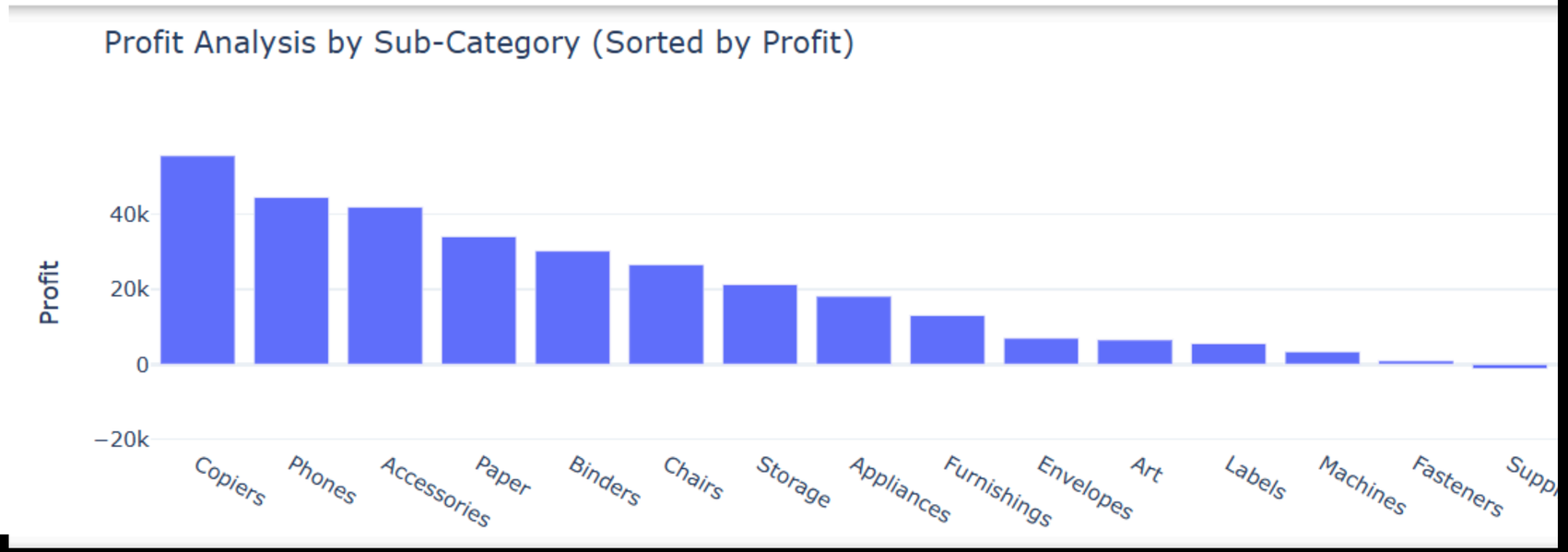
- This visualization compares profit across product categories.
- It shows that higher sales do not always mean higher profit.
- Helps identify categories with better profit margins.
- Supports pricing and cost optimization decisions.

Profit by Sub-Category -

```
Profit_by_subcategory = data.groupby('Sub-Category')['Profit'].sum().reset_index()
Profit_by_subcategory
```

	Sub-Category	Profit
0	Accessories	41936.6357
1	Appliances	18138.0054
2	Art	6527.7870
3	Binders	30221.7633
4	Bookcases	-3472.5560
5	Chairs	26590.1663
6	Copiers	55617.8249
7	Envelopes	6964.1767
8	Fasteners	949.5182
9	Furnishings	13059.1436
10	Labels	5546.2540
11	Machines	3384.7569
12	Paper	34053.5693
13	Phones	44515.7306
14	Storage	21278.8264
15	Supplies	-1189.0995
16	Tables	-17725.4811

```
Profit_by_subcategory = (
    data.groupby('Sub-Category')['Profit']
    .sum()
    .reset_index()
    .sort_values(by='Profit', ascending=False))
fig = px.bar(
    Profit_by_subcategory,
    x='Sub-Category',
    y='Profit',
    title="Profit Analysis by Sub-Category (Sorted by Profit)")
fig.show()
```



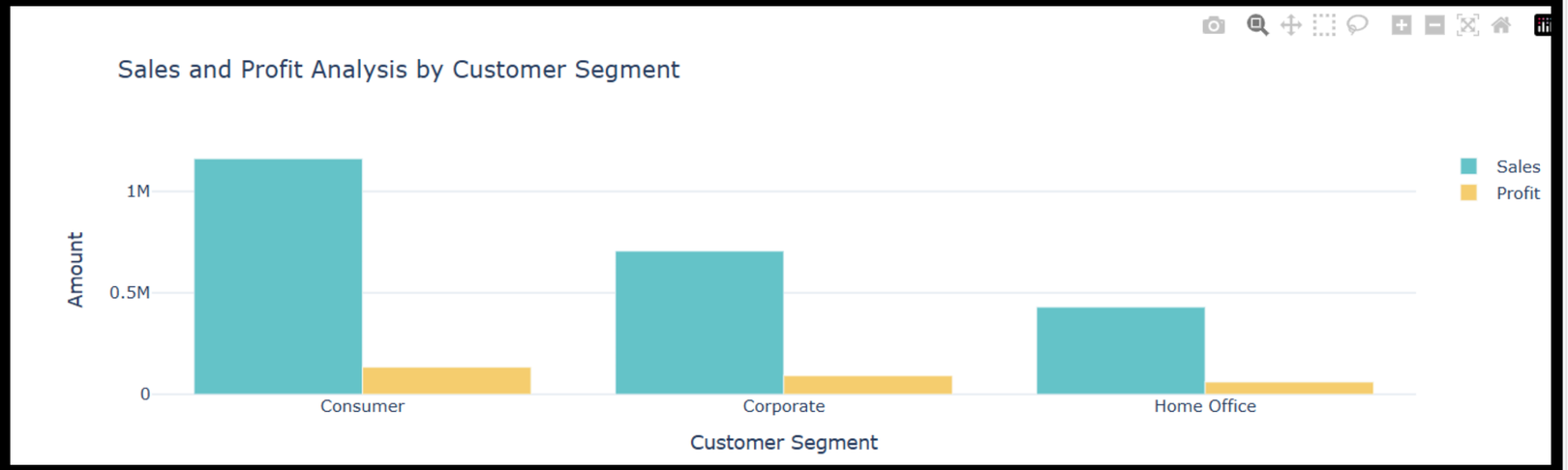
Key Finding -

- **This graph analyzes profit at the sub-category level.**
- **It helps identify loss-making or low-profit products.**
- **Useful for deciding which products to improve, continue, or remove.**
- **Supports strategic product portfolio decisions.**

Sales and Profit - Customer Segment -

```
Sales_Profit_by_Segment= data.groupby('Segment').agg({'Sales': 'sum', 'Profit':'sum'}).reset_index()
Sales_Profit_by_Segment
```

	Segment	Sales	Profit
0	Consumer	1.161401e+06	134119.2092
1	Corporate	7.061464e+05	91979.1340
2	Home Office	4.296531e+05	60298.6785




```
Sales_Profit_by_Segment = data.groupby('Segment').agg({
    'Sales': 'sum',
    'Profit': 'sum'
}).reset_index()

color_palette = colors.qualitative.Pastel

fig = go.Figure()

fig.add_trace(go.Bar(
    x=Sales_Profit_by_Segment['Segment'],
    y=Sales_Profit_by_Segment['Sales'],
    name='Sales',
    marker_color=color_palette[0]
))

fig.add_trace(go.Bar(
    x=Sales_Profit_by_Segment['Segment'],
    y=Sales_Profit_by_Segment['Profit'],
    name='Profit',
    marker_color=color_palette[1]
))

fig.update_layout(
    title='Sales and Profit Analysis by Customer Segment',
    xaxis_title='Customer Segment',
    yaxis_title='Amount',
    barmode='group'
)

fig.show()
```

Key Finding -

- **The Consumer segment generates the highest sales and profit.**
- **The Corporate segment shows moderate and stable performance.**
- **The Home Office segment has the lowest sales and profit.**
- **Business growth is mainly driven by the Consumer segment.**

Sales to Profit Ratio -

```
Sales_profit_by_segment = data.groupby('Segment').agg({
    'Sales': 'sum',
    'Profit': 'sum'
}).reset_index()

Sales_profit_by_segment['Sales_to_Profit_Ratio'] = (
    Sales_profit_by_segment['Sales'] / Sales_profit_by_segment['Profit']
)

print(Sales_profit_by_segment[['Segment', 'Sales_to_Profit_Ratio']])
```

```
       Segment  Sales_to_Profit_Ratio
0     Consumer               8.659471
1    Corporate               7.677245
2  Home Office               7.125416
```

Key Insights

- Sales show clear monthly trends, indicating seasonality in customer purchasing behavior.
- Certain product categories consistently contribute the highest sales, making them key revenue drivers.
- Some sub-categories generate low sales but high profit, while others have high sales with low profit margins.
- Monthly profit does not always increase with monthly sales, showing the impact of discounts and costs.
- A few sub-categories show low or negative profit, indicating pricing or cost issues.
- Time-based analysis using extracted Month and Day of Week helped identify peak and slow periods

Recommendations

- Focus marketing and inventory planning on high-sales and high-profit months to maximize revenue.
- Improve pricing, promotions, or cost control for low-profit sub-categories.
- Increase visibility and stock for high-performing categories and sub-categories.
- Re-evaluate discount strategies during high-sales months to protect profit margins.
- Plan special offers and campaigns during low-sales months to boost demand.
- Use day-of-week insights to optimize staffing, promotions, and delivery planning.

Business Value

These insights and recommendations support data-driven decision-making, helping improve sales performance, profitability, and overall business efficiency.

Conclusion

This analysis helped to understand how the store is performing in terms of sales and profit. By checking monthly sales, we identified the month with the highest sales and the month with the lowest sales. Monthly profit analysis also showed which month was the most profitable.

Sales analysis by category and sub-category showed that some products sell very well, while others have low sales. The same pattern was seen in profit, where a few categories and sub-categories bring good profit and others give less profit or even losses.

Customer segment analysis showed that different customer groups behave differently. Some segments give high sales, while some give better profit. The sales-to-profit ratio showed that high sales do not always mean high profit, mainly due to discounts and costs.

Overall, this analysis helps the business focus more on high-performing products, profitable customer segments, and the best sales months, while improving or controlling low-performing areas.

Thank You

FOR YOUR ATTENTION

jatinpatidar717@gmail.com