Assignment for Business Analyst Intern @Jar

Name – Rupam Mal

rupammal2025@gmail.com

Code: https://github.com/RupamMal/JAR-assignment

Sales Analysis:

Part 1: Sales and Profitability Analysis

- Merge the List of Orders and Order Details datasets on the basis of Order ID. Calculate the total sales (Amount) for each category across all orders.
- For each category, calculate the average profit per order and total profit margin (profit as a percentage of Amount).
- Identify the top-performing and underperforming categories based on these metrics. Also, suggest reasons for their performance differences.

In [4]:
```
import pandas as pd
```

In [8]:
```
df_orders =  pd.read_excel("List_of_Orders_1.xlsx")
```

In [10]:
```
df_details =  pd.read_excel("Order_Details_1.xlsx")
```

In [12]:
```
df_target =  pd.read_excel("Sales_target_1.xlsx")
```

In [14]:
```
print(df_orders.head())
```

```
  Order ID Order Date CustomerName          State       City
0  B-25601 2018-04-01       Bharat        Gujarat  Ahmedabad
1  B-25602 2018-04-01        Pearl    Maharashtra       Pune
2  B-25603 2018-04-03        Jahan  Madhya Pradesh     Bhopal
3  B-25604 2018-04-03       Divsha      Rajasthan     Jaipur
4  B-25605 2018-04-05      Kasheen    West Bengal    Kolkata
```

In [16]:
```
df_orders.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 500 entries, 0 to 499
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype         
---  ------        --------------  -----         
 0   Order ID      500 non-null    object        
 1   Order Date    500 non-null    datetime64[ns]
 2   CustomerName  500 non-null    object        
 3   State         500 non-null    object        
 4   City          500 non-null    object        
dtypes: datetime64[ns](1), object(4)
memory usage: 19.7+ KB
```

In [18]:
```
print(df_orders.describe(include='all'))
```

```
       Order ID                      Order Date CustomerName           State  \
count       500                             500          500             500   
unique      500                             NaN          332              19   
top     B-25601                             NaN       Shreya  Madhya Pradesh   
freq          1                             NaN            6             101   
mean        NaN  2018-10-21 11:11:02.400000256          NaN             NaN   
min         NaN            2018-04-01 00:00:00          NaN             NaN   
25%         NaN            2018-07-20 18:00:00          NaN             NaN   
50%         NaN            2018-11-05 12:00:00          NaN             NaN   
75%         NaN            2019-01-25 00:00:00          NaN             NaN   
max         NaN            2019-03-31 00:00:00          NaN             NaN   

          City  
count      500  
unique      24  
top     Indore  
freq        76  
mean       NaN  
min        NaN  
25%        NaN  
50%        NaN  
75%        NaN  
max        NaN  
```

In [20]:
```
print(df_orders['State'].value_counts())
```

```
State
Madhya Pradesh       101
Maharashtra           90
Rajasthan             32
Gujarat               27
Punjab                25
Uttar Pradesh         22
Delhi                 22
West Bengal           22
Karnataka             21
Kerala                16
Bihar                 16
Nagaland              15
Andhra Pradesh        15
Jammu and Kashmir     14
Haryana               14
Himachal Pradesh      14
Goa                   14
Sikkim                12
Tamil Nadu             8
Name: count, dtype: int64
```

In [22]:
```
print(df_details.head())
```

```
  Order ID  Amount  Profit  Quantity     Category      Sub-Category
0  B-25601    1275   -1148         7    Furniture         Bookcases
1  B-25601      66     -12         5     Clothing             Stole
2  B-25601       8      -2         3     Clothing       Hankerchief
3  B-25601      80     -56         4  Electronics  Electronic Games
4  B-25602     168    -111         2  Electronics            Phones
```

In [24]:
```
df_details.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1500 entries, 0 to 1499
Data columns (total 6 columns):
 #   Column        Non-Null Count  Dtype 
---  ------        --------------  ----- 
 0   Order ID      1500 non-null   object
 1   Amount        1500 non-null   int64 
 2   Profit        1500 non-null   int64 
 3   Quantity      1500 non-null   int64 
 4   Category      1500 non-null   object
 5   Sub-Category  1500 non-null   object
dtypes: int64(3), object(3)
memory usage: 70.4+ KB
```

In [26]:
```
print(df_details.describe())
```

```
            Amount       Profit     Quantity
count  1500.000000  1500.000000  1500.000000
mean    287.668000    15.970000     3.743333
std     461.050488   169.140565     2.184942
min       4.000000 -1981.000000     1.000000
25%      45.000000    -9.250000     2.000000
50%     118.000000     9.000000     3.000000
75%     322.000000    38.000000     5.000000
max    5729.000000  1698.000000    14.000000
```

In [28]:
```
print(df_details['Category'].value_counts())
```

```
Category
Clothing       949
Electronics    308
Furniture      243
Name: count, dtype: int64
```

In [30]:
```
print(df_details['Sub-Category'].value_counts().head(10))
```

```
Sub-Category
Saree               210
Hankerchief         198
Stole               192
Phones               83
Bookcases            79
Electronic Games     79
T-shirt              77
Printers             74
Chairs               74
Furnishings          73
Name: count, dtype: int64
```

In [32]:
```
print(df_target.head())
```

```
  Month of Order Date   Category  Target
0          2025-04-18  Furniture   10400
1          2025-05-18  Furniture   10500
2          2025-06-18  Furniture   10600
3          2025-07-18  Furniture   10800
4          2025-08-18  Furniture   10900
```

In [34]:
```
df_target.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 36 entries, 0 to 35
Data columns (total 3 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   Month of Order Date  36 non-null     datetime64[ns]
 1   Category             36 non-null     object
 2   Target               36 non-null     int64
dtypes: datetime64[ns](1), int64(1), object(1)
memory usage: 996.0+ bytes
```

In [36]:
```
print(df_target.describe())
```

```
       Month of Order Date        Target
count                   36     36.000000
mean   2025-07-03 18:00:00  12108.333333
min    2025-01-19 00:00:00   9000.000000
25%    2025-04-10 12:00:00  10050.000000
50%    2025-07-03 00:00:00  11450.000000
75%    2025-09-25 12:00:00  14500.000000
max    2025-12-18 00:00:00  16000.000000
std                    NaN   2667.837541
```

In [38]:
```
df_target.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 36 entries, 0 to 35
Data columns (total 3 columns):
 #   Column               Non-Null Count  Dtype         
---  ------               --------------  -----         
 0   Month of Order Date  36 non-null     datetime64[ns]
 1   Category             36 non-null     object        
 2   Target               36 non-null     int64         
dtypes: datetime64[ns](1), int64(1), object(1)
memory usage: 996.0+ bytes
```

In [40]:
```
df_merged = pd.merge(df_orders, df_details, on='Order ID', how='inner')
```

In [42]:
```
print(df_merged.head())
```

```
  Order ID Order Date CustomerName        State       City  Amount  Profit  \
0  B-25601 2018-04-01       Bharat      Gujarat  Ahmedabad    1275   -1148   
1  B-25601 2018-04-01       Bharat      Gujarat  Ahmedabad      66     -12   
2  B-25601 2018-04-01       Bharat      Gujarat  Ahmedabad       8      -2   
3  B-25601 2018-04-01       Bharat      Gujarat  Ahmedabad      80     -56   
4  B-25602 2018-04-01        Pearl  Maharashtra       Pune     168    -111   

   Quantity     Category      Sub-Category
0         7    Furniture         Bookcases
1         5     Clothing             Stole
2         3     Clothing       Hankerchief
3         4  Electronics  Electronic Games
4         2  Electronics            Phones
```

In [62]:
```
df_merged
```

Out[62]:

	Order ID	Order Date	CustomerName	State	City	Amount	Profit	Quantit
0	B-25601	2018-04-01	Bharat	Gujarat	Ahmedabad	1275	-1148	
1	B-25601	2018-04-01	Bharat	Gujarat	Ahmedabad	66	-12	
2	B-25601	2018-04-01	Bharat	Gujarat	Ahmedabad	8	-2	
3	B-25601	2018-04-01	Bharat	Gujarat	Ahmedabad	80	-56	
4	B-25602	2018-04-01	Pearl	Maharashtra	Pune	168	-111	
...	...	...	...	...	...	...	...	
1495	B-26099	2019-03-30	Bhishm	Maharashtra	Mumbai	835	267	
1496	B-26099	2019-03-30	Bhishm	Maharashtra	Mumbai	2366	552	
1497	B-26100	2019-03-31	Hitika	Madhya Pradesh	Indore	828	230	
1498	B-26100	2019-03-31	Hitika	Madhya Pradesh	Indore	34	10	
1499	B-26100	2019-03-31	Hitika	Madhya Pradesh	Indore	72	16	

1500 rows × 11 columns

In [48]:
```
category_sales = df_merged.groupby("Category")["Amount"].sum().reset_index()
category_sales.columns = ["Category", "Total Sales"]
category_sales
```

Out[48]:

	Category	Total Sales
0	Clothing	139054
1	Electronics	165267
2	Furniture	127181

In [52]:
```
category_avg_profit = df_merged.groupby("Category")["Profit"].mean().reset_index
category_avg_profit.columns = ["Category", "Avg Profit per Order"]
```

In [54]:
```
df_merged["Profit Margin %"] = (df_merged["Profit"] / df_merged["Amount"]) * 100
category_profit_margin = df_merged.groupby("Category")["Profit Margin %"].mean()
category_profit_margin.columns = ["Category", "Avg Profit Margin (%)"]
```

In [56]:
```
sales_summary = category_sales.merge(category_avg_profit, on="Category").merge(c
sales_summary
```

Out[56]:

	Category	Total Sales	Avg Profit per Order	Avg Profit Margin (%)
0	Clothing	139054	11.762908	4.132921
1	Electronics	165267	34.071429	-0.622928
2	Furniture	127181	9.456790	-6.788811

In [58]:
```
top_category = sales_summary.loc[sales_summary["Avg Profit Margin (%)"].idxmax()
low_category = sales_summary.loc[sales_summary["Avg Profit Margin (%)"].idxmin()
```

In [60]:
```
print("Best Performing Category:", top_category["Category"])
print("Least Performing Category:", low_category["Category"])
```

```
Best Performing Category: Clothing
Least Performing Category: Furniture
```

Category	Performance Summary	Primary Reason for Performance Difference
Clothing	This is the **most profitable** category with a positive Average Profit Margin of **4.13%**. It generates lower total sales than Electronics but is likely composed of high-volume items with healthy unit profits. This category provides the most stable and reliable contribution to the company's bottom line.	**Positive Profit Margin:** Consists of high-volume, lower-priced goods with well-managed procurement and operational costs, leading to consistent net profitability. **Low Return Rate:** Clothing items may have a relatively low rate of damaged or returned goods compared to fragile electronics or bulky furniture, reducing expenses associated with reverse logistics and restocking.
Electronics	This category generates the **highest total sales** at **165,267**, and the highest average profit per order at 34.07. However, it operates at a slight loss with an average profit margin of **-0.62%**. While popular, its high sales volume barely covers its costs, indicating a need to investigate overheads or pricing.	**High Logistical Costs:** Significant losses due to high expenses for shipping, handling, and storage of bulky items (like chairs and bookcases) that are not covered by the selling price. **High Custom Duties/Taxes:** Electronics are often imported and may be subject to higher customs duties, taxes, or licensing fees, significantly increasing the Cost of Goods Sold (COGS).
Furniture	This is the **least profitable** category with the lowest total sales and a highly negative Average Profit Margin of **-6.79%**. Although the average profit per order is slightly positive at 9.46, the deep negative margin suggests significant losses due to high costs or discounts. High costs like shipping/logistics for bulky items are the likely culprit for the poor overall performance.	**High Overhead/Variable Costs:** Despite high sales, the marginal negative profit is due to high associated expenses (e.g., warranty claims, rapid obsolescence) that erode the gross profit margin. **Slow Inventory Turnover:** High-value, bulky items like furniture generally have a slower inventory turnover rate, incurring higher costs for storage/warehousing over longer periods.

Part 2: Target Achievement Analysis

- Using the Sales Target dataset, calculate the percentage change in target sales for the Furniture category month-over-month.
- Analyse the trends to identify months with significant target fluctuations. Suggest strategies for aligning target expectations with actual performance trends.

In [3]:
```
import pandas as pd
```

In [5]:
```
df_target =  pd.read_excel("Sales_target_1.xlsx")
```

In [7]:
```
df_target["Month of Order Date"] = pd.to_datetime(df_target["Month of Order Date
```

In [9]:
```
df_target = df_target.sort_values("Month of Order Date")
```

In [11]:
```
furniture_sales_target = df_target[df_target["Category"] == "Furniture"].copy()
```

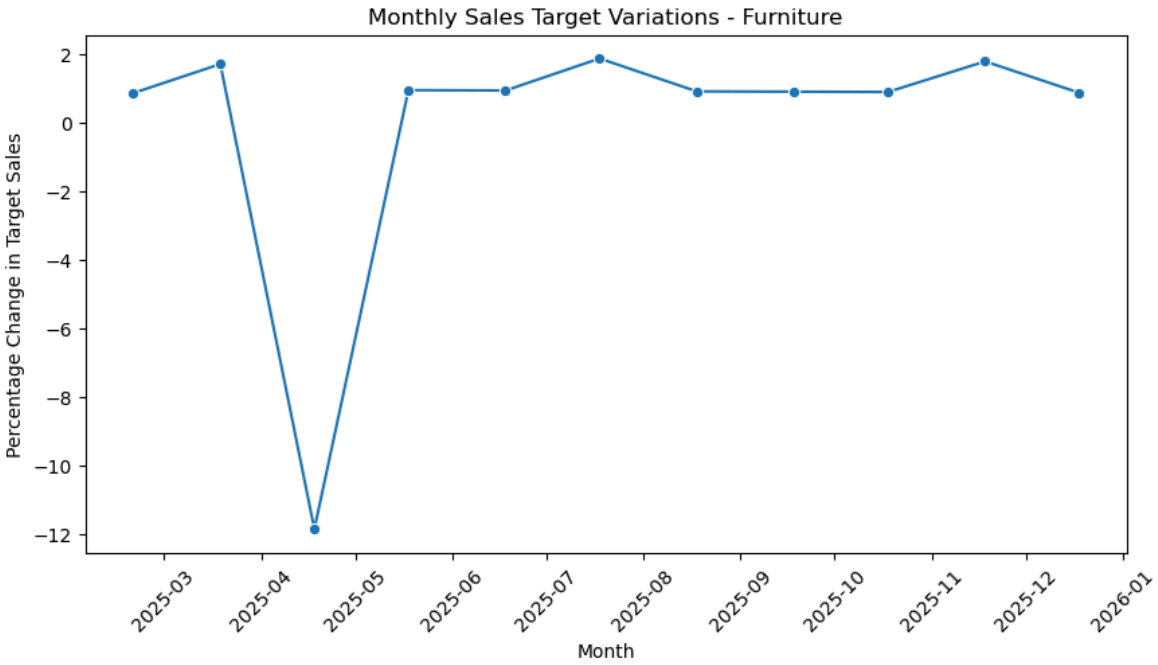
In [13]:
```
furniture_sales_target["Target Change (%)"] = furniture_sales_target["Target"].p
```

In [17]:
```
furniture_sales_target
```

Out[17]:

	Month of Order Date	Category	Target	Target Change (%)
9	2025-01-19	Furniture	11500	NaN
10	2025-02-19	Furniture	11600	0.869565
11	2025-03-19	Furniture	11800	1.724138
0	2025-04-18	Furniture	10400	-11.864407
1	2025-05-18	Furniture	10500	0.961538
2	2025-06-18	Furniture	10600	0.952381
3	2025-07-18	Furniture	10800	1.886792
4	2025-08-18	Furniture	10900	0.925926
5	2025-09-18	Furniture	11000	0.917431
6	2025-10-18	Furniture	11100	0.909091
7	2025-11-18	Furniture	11300	1.801802
8	2025-12-18	Furniture	11400	0.884956

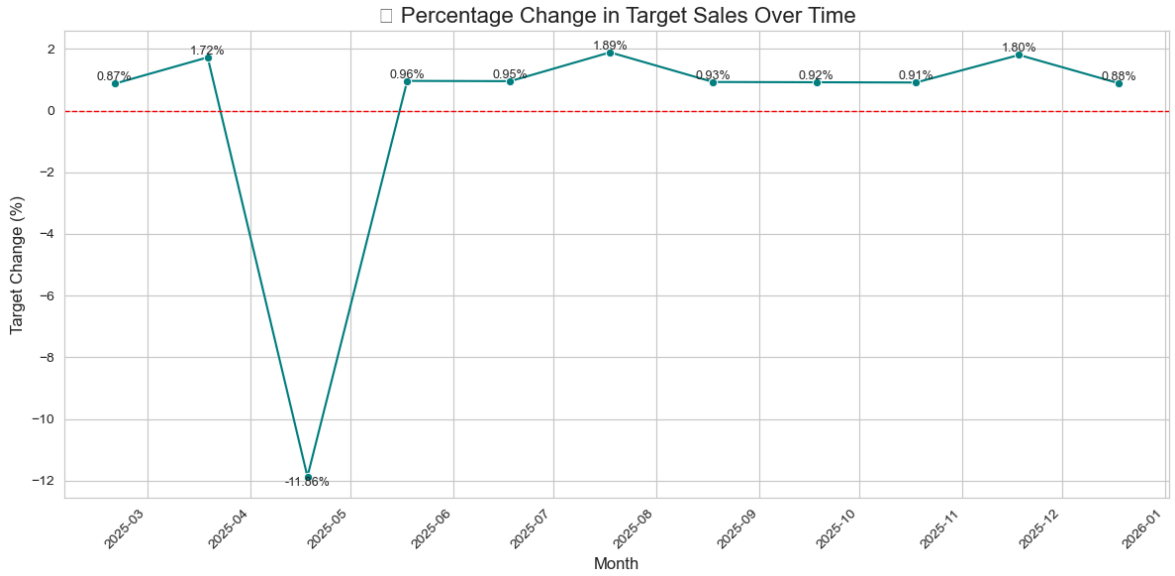
In [29]:
```
plt.figure(figsize=(10, 5))
sns.lineplot(x=furniture_sales_target["Month of Order Date"], y=furniture_sales_
plt.xlabel("Month")
plt.ylabel("Percentage Change in Target Sales")
plt.title("Monthly Sales Target Variations - Furniture")
plt.xticks(rotation=45)
plt.show()
```



In [33]:
```
data = {
    'Month of Order Date': ['2025-01-19', '2025-02-19', '2025-03-19', '2025-04-1
                            '2025-05-18', '2025-06-18', '2025-07-18', '2025-08-1
                            '2025-09-18', '2025-10-18', '2025-11-18', '2025-12-1
    'Category': ['Furniture'] * 12,
    'Target': [11500, 11600, 11800, 10400, 10500, 10600, 10800, 10900, 11000, 11
    'Target Change (%)': [np.nan, 0.869565, 1.724138, -11.864407, 0.961538, 0.95
                          1.886792, 0.925926, 0.917431, 0.909091, 1.801802, 0.88
}
df = pd.DataFrame(data)

df['Month of Order Date'] = pd.to_datetime(df['Month of Order Date'])

df_plot = df.dropna(subset=['Target Change (%)']).copy()

sns.set_style("whitegrid")
plt.figure(figsize=(12, 6))

line = sns.lineplot(
    x='Month of Order Date',
    y='Target Change (%)',
    data=df_plot,
    marker='o',
    color='teal'
)

plt.title('🎯 Percentage Change in Target Sales Over Time', fontsize=16)
plt.xlabel('Month', fontsize=12)
plt.ylabel('Target Change (%)', fontsize=12)
plt.xticks(rotation=45, ha='right')

plt.axhline(0, color='red', linestyle='--', linewidth=1)

for x, y in zip(df_plot['Month of Order Date'], df_plot['Target Change (%)']):
    plt.text(x, y, f'{y:.2f}%',
             ha='center',
             va='bottom' if y >= 0 else 'top',
             fontsize=9)
```

```
plt.tight_layout()
plt.show()
```

```
C:\Users\Rupam\AppData\Local\Temp\ipykernel_24300\1300105528.py:40: UserWarning: Glyph 127919 (\N{DIRECT HIT}) missing from current font.
  plt.tight_layout()
C:\Users\Rupam\AppData\Roaming\Python\Python312\site-packages\IPython\core\pylabtools.py:170: UserWarning: Glyph 127919 (\N{DIRECT HIT}) missing from current font.
  fig.canvas.print_figure(bytes_io, **kw)
```



Month	Target Change (%)	Trend Observation
Mar 2025 to Apr 2025	-11.86%	**Most Significant Fluctuation**: This is the largest and only major decrease, indicating a planned or necessary reduction in the target after Q1.
Feb 2025 to Mar 2025	1.72%	**Highest Positive Increase (Q1)**: The target increased by a noticeable amount leading into the start of the typical spring/Q2 selling season.
Jun 2025 to Jul 2025	1.89%	**Highest Positive Increase (Overall)**: This spike suggests an expected surge in sales, perhaps aligning with mid-year promotions or seasonal shifts (e.g., summer furniture).
All Other Months	+0.87% to +0.96%	**Steady, Modest Growth**: The target generally exhibits a consistent, small month-over-month increase, suggesting a foundational expectation of continuous improvement.

Strategies for Target Alignment with Actual Performance Trends:

1. **Adopt a Seasonal Forecasting Model**: Use models that incorporate historical cyclical demand (e.g., the April dip and July spike) rather than relying on a simple, fixed monthly growth factor.

2. **Conduct Root Cause Analysis on Major Fluctuations:** Investigate the reason behind the significant -11.86% target decreases in April 2025 to determine if it was a necessary adjustment based on prior overestimation or an accurate market-driven expectation.

3. **Integrate Historical Sales Achievement Rate into Target Setting:** Adjust future targets based on the previous year's Sales Achievement Rate (Actual Sales div Target). If a month consistently missed its goal (e.g., 90% achievement), the new target should be adjusted or paired with a resource increase to bridge the gap.

4. **Differentiate Target-Setting by Sub-Category/Segment:** Analyse and set targets for individual sub-segments within Furniture (e.g., dining, office). This prevents high volatility in a small sub-category from overly skewing the overall target expectation for the entire category.

5. **Implement a Flex Budget for Q1/Q2:** Introduce a dynamic "flex budget" for the volatile March-April period. This budget can be reallocated mid-quarter to either reward over-performance or apply pressure to slow periods, making targets more responsive to real-time performance.

Part 3: Regional Performance Insights

- From the List of Orders dataset, identify the top 5 states with the highest order count. For each of these states, calculate the total sales and average profit.
- Highlight any regional disparities in sales or profitability. Suggest regions or cities that should be prioritized for improvement.

In [1]:
```
import pandas as pd
```

In [3]:
```
df_orders =  pd.read_excel("List_of_Orders_1.xlsx")
```

In [5]:
```
df_details =  pd.read_excel("Order_Details_1.xlsx")
```

In [7]:
```
df_target =  pd.read_excel("Sales_target_1.xlsx")
```

In [11]:
```
merged_df = pd.merge(df_details, df_orders, on="Order ID")
```

In [13]:
```
merged_df
```

Out[13]:

	Order ID	Amount	Profit	Quantity	Category	Sub-Category	Order Date	CustomerName
0	B-25601	1275	-1148	7	Furniture	Bookcases	2018-04-01	Bharat
1	B-25601	66	-12	5	Clothing	Stole	2018-04-01	Bharat
2	B-25601	8	-2	3	Clothing	Hankerchief	2018-04-01	Bharat
3	B-25601	80	-56	4	Electronics	Electronic Games	2018-04-01	Bharat
4	B-25602	168	-111	2	Electronics	Phones	2018-04-01	Pearl
...	...	...	...	...	...	...	...	...
1495	B-26099	835	267	5	Electronics	Phones	2019-03-30	Bhishm
1496	B-26099	2366	552	5	Clothing	Trousers	2019-03-30	Bhishm
1497	B-26100	828	230	2	Furniture	Chairs	2019-03-31	Hitika
1498	B-26100	34	10	2	Clothing	T-shirt	2019-03-31	Hitika
1499	B-26100	72	16	2	Clothing	Shirt	2019-03-31	Hitika

1500 rows × 10 columns

In [17]:
```
top_states = merged_df["State"].value_counts().head(5).index.tolist()
statewise_df = merged_df[merged_df["State"].isin(top_states)]
```

In [19]:
```
state_performance = statewise_df.groupby("State").agg(
Order_Count=("Order ID", "count"),
Total_Sales=("Amount", "sum"),
Avg_Profit=("Profit", "mean")
).reset_index()
```

In [21]:
```
print(state_performance)
```

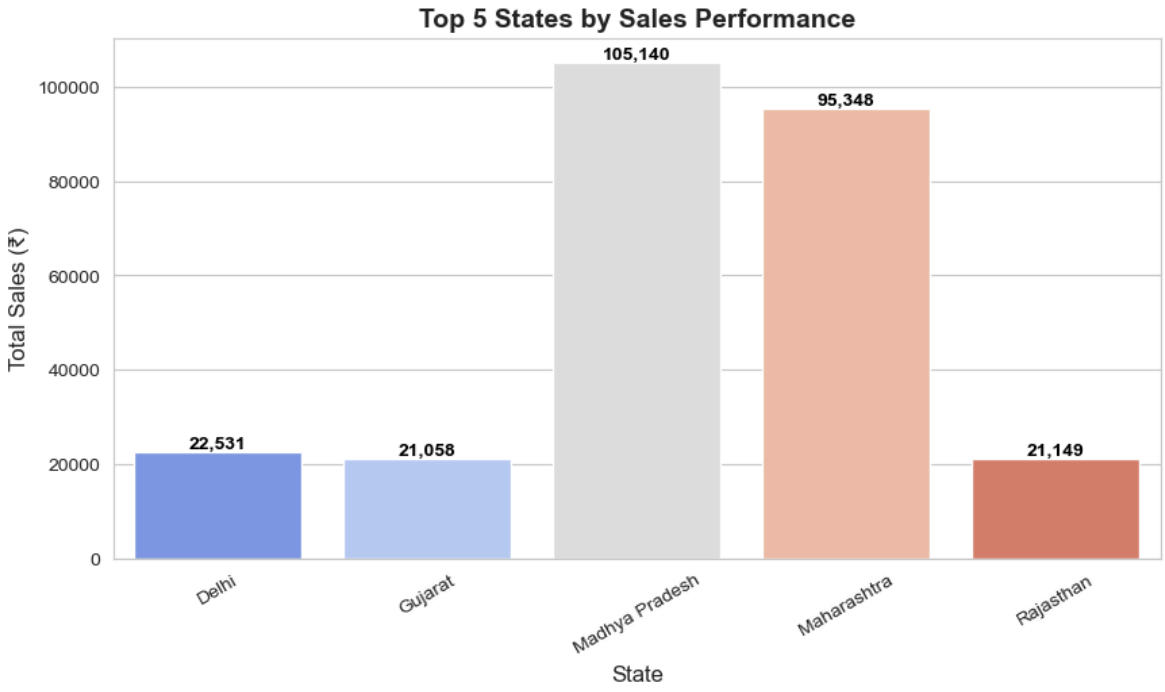
```
           State  Order_Count  Total_Sales  Avg_Profit
0          Delhi           74        22531   40.364865
1        Gujarat           87        21058    5.344828
2  Madhya Pradesh          340       105140   16.326471
3    Maharashtra           290        95348   21.296552
4      Rajasthan           74        21149   16.986486
```

In [23]:
```
import seaborn as sns
import matplotlib.pyplot as plt
```

In [25]:
```
sns.set_style("whitegrid")
```

In [33]:
```
plt.figure(figsize=(10, 5))
ax = sns.barplot(x="State", y="Total_Sales", data=state_performance, palette="co

plt.xlabel("State", fontsize=12)
plt.ylabel("Total Sales (₹)", fontsize=12)
plt.title("Top 5 States by Sales Performance", fontsize=14, fontweight='bold')

plt.xticks(rotation=30)

for p in ax.patches:
    ax.annotate(f'{p.get_height():,.0f}',
(p.get_x() + p.get_width() / 2., p.get_height()),
ha='center', va='bottom', fontsize=10, color='black', fontweight='bold')
# Show the plot
plt.show()
```

```
C:\Users\Rupam\AppData\Local\Temp\ipykernel_24764\3911232467.py:2: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v
0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effe
ct.

  ax = sns.barplot(x="State", y="Total_Sales", data=state_performance, palette="c
oolwarm")
```


Top 5 States by Sales Performance
Total Sales (₹)
0
20000
40000
60000
80000
100000
22,531
21,058
105,140
95,348
21,149
Delhi
Gujarat
Madhya Pradesh
Maharashtra
Rajasthan
State

In [ ]:

1. Sales Disparity

- **Leading Regions: Madhya Pradesh** and **Maharashtra** are the dominant regions, accounting for the highest Total Sales (₹105,140 and ₹95,348, respectively) and Order Counts (340 and 290, respectively).
- **Laggard Regions: Delhi**, **Gujarat**, and **Rajasthan** show a significant volume disparity, with Total Sales clustering around ₹21,000 to ₹22,500. Delhi and Rajasthan have the lowest number of orders at 74 each.

2. Profitability Disparity

- **Highest Profitability: Delhi** stands out with the highest Average Profit per order at ₹40.36.
- **Lowest Profitability: Gujarat** shows the poorest performance, with an Average Profit of only ₹5.34. This is a severe margin issue, suggesting high costs or heavy discounting in this region.
- **Volume vs. Margin:** Despite having the highest sales volume, **Madhya Pradesh**'s average profit (₹16.33) is lower than both Maharashtra (₹21.30) and the low-volume Delhi region (₹40.36).

Prioritization for Improvement

Prioritization should be split into efforts to address the profit issue and efforts to address the volume issue.

Priority 1: Profit Improvement (Focus on Gujarat)

- **Region: Gujarat**
- **Issue:** Critically low Average Profit (₹5.34).
- **Suggested Action:** This region should be the top priority for a margin review. Focus on identifying the causes of low profitability, such as high shipping/logistics costs, excessive discounts, or high cost of procurement/poor product mix in the state.

Priority 2: Volume Growth (Focus on Delhi & Rajasthan)

- **Regions: Delhi** and **Rajasthan**
- **Issue:** Low Order Count (74 each) and low Total Sales.
- **Suggested Action:** These regions are ripe for sales expansion. **Delhi** is particularly promising as it already demonstrates very strong profitability (₹40.36 Avg. Profit). Campaigns and sales efforts should be boosted here to leverage the high margins and increase total revenue contribution. For Rajasthan, sales efforts should aim for volume while maintaining the current margin (₹16.99 Avg. Profit).

Secondary Focus: Margin Review in Top State (Madhya Pradesh)

- **Region: Madhya Pradesh**
- **Issue:** Highest sales volume but moderate average profit (₹16.33), which is lower than Maharashtra's (₹21.30).

- **Suggested Action:** Even a small increase in the average profit margin in this high-volume state would yield a large increase in total company profit. Investigate whether operational efficiencies or a slight price/discount adjustment can bring its average profit closer to Maharashtra's.

Question 2 : (10 marks)

App Exploration:

Explore the features and user experience of the Jar app. Highlight five things you found particularly effective and user-friendly. Additionally, identify five areas where improvements could be made, providing your reasoning for each suggestion.

Jar is an Indian fintech startup that offers a micro-savings and investment app. It allows users to save and invest small amounts of money, starting from ₹10, in digital gold on a daily, weekly, or monthly basis. It has over 1.5 crore users.

About Jar Users- Jar's users are young Indian adults, typically in the age range of 25–35.
1. They have a regular source of income.
2. Living in urban or semi-urban areas.
3. Comfortable using digital financial services.
4. Desire to Save and Invest, but do not know where to start.
6. Have little to no financial knowledge beyond FDs.
7. Finally, people were worried about liquidity – they wanted their money to be instantly accessible.

How does Jar works- Jar allows you to save small amounts of money and invest the savings into digital gold, which can subsequently be liquidated or converted to physical gold. The app does this by checking your transaction SMS and prompting you to round off spends and invest the delta. There is also an option to invest discrete amounts as and when you want.

Things I found particularly effective and user-friendly:

1.Simple and Intuitive Interface: Jar presents users with a gamified but familiar scenario. Whenever the user makes an investment, they are allowed to 'spin a jar' for rewards. In parallel, they receive a notification congratulating them on their investment. This double-incentive has created an early and effective flywheel that is keeping users investing with Jar so far. The minimal and functional design language (e.g., confined in cards, minimal text) makes the investment process unintimidating and easy to navigate for beginners. This aligns well with its mission of fostering a daily savings habit for a wide audience.

2. Gamification and Rewards: The app often includes elements like the "Wheel of Savings," and "Magic Hat" where users can win cash-back or double their savings after a transaction and from magic hat they can get special cards. This element introduces a fun and engaging psychological loop. It rewards the act of saving/spending, making the otherwise mundane financial process feel like a positive game, which helps in boosting engagement and repeat transactions.

3. Automated "Round-Off" Savings: The app automatically rounds up a user's UPI or online transactions to the nearest ₹10 or ₹5 and invests the "spare change" into digital gold. This feature is highly effective because it makes saving effortless and habitual. It leverages existing spending habits to create an investment habit.

4. Quick and Seamless Account Setup: The onboarding process is extremely fast, requiring minimal information (primarily a phone number) and is completely paperless. It takes average 45-60 seconds. This low-friction setup removes the biggest barrier to entry for new users, especially those intimidated by traditional financial institutions' extensive KYC requirements. A user can go from downloading the app to making their first investment in under a minute.

5. Real-Time Gold Price Tool and Secured Vaulting for Gold: The app provides a dedicated "Gold price tool" that offers real-time updates on 24K gold rates, often refreshing every minute, and sometimes even showing prices based on purity (e.g., 18k, 22k, 24k) or weight. Giving users real-time transparency on the live price of gold is crucial for building trust. And also, the digital gold purchased by the user is stored in bank-grade, world-class lockers/vaults and is typically insured by top banks, all at no extra cost to the user.

Areas for Improvement and Suggestions:

1.Bad Quality Graphics and Illustrations: Numerous illustrations and graphics, particularly the "dirt board" in the goal-setting screen, are of low resolution and poor graphical quality, which detracts from the overall user experience and professionalism of the app.

Suggestion - Implement a full-scale visual refresh initiative with a focus on consistency, high-resolution assets, and a polished design system. Where you should introduce Subtle Animations and Micro-interactions.

2. Lack of Financial Education & Insights: Many users don’t understand digital gold or investment risks.

Suggestion- Add educational blogs, videos, and AI-driven spending insights to help users make smarter investment choices. Jar can Collab with Investopedia academy, NCFE or Varsity by Zerodha and can make a platform like “Knowledge Jar” to give financial literacy to beginners.

3. Slow Withdrawal Process and Bad Customer Support Response Time: From Play store review of Jar app, I saw that some users experience delays when withdrawing funds from the app and some users report delays in customer support responses, especially for fund withdrawals.

Suggestion- Improve support by adding 24/7 live chat or a faster query resolution system and Improve transaction speed and introduce instant withdrawal options for greater flexibility. These also affects user experience.

4. No Much Goal-Based Saving Plans: The app lacks custom savings goals (e.g., saving for a vacation, wedding, education, self-gifting).

Suggestion- Allow users to set financial goals, track progress, and get reminders for consistent savings.

5. Data Aggregation and Financial Reporting: The app's strength is saving spare change, but it often lacks sophisticated reporting on a user's overall spending *pattern* that led to those savings. Users lack an easy way to export their gold purchase history for tax or personal record-keeping purposes.

Suggestion- Provide users with a monthly email or in-app report detailing: "Total Gold Saved This Month," "Top 3 Spending Categories That Triggered Savings," and the Gold Purchase Ledger.

Question 3: (10 marks)

Product Exploration:
The Jar app provides users with an innovative way to save and invest in digital gold, starting with as little as ₹10. It automates savings and investments, making financial planning seamless and accessible. As the first Made-in-India app to pioneer such a solution, Jar has successfully created a niche in automated savings and investment.

Building on its strong foundation and leveraging its existing user base and trust, what are some new business opportunities Jar could venture into, to expand its offerings and enhance user engagement? Discuss how the app can utilize its strengths, such as automation, a user-friendly design, and established credibility, to seamlessly integrate these new services, deepen its value proposition, and achieve significant milestones in the financial ecosystem.

Five Effective & User-Friendly Features:

The Jar app has successfully built a niche in automated savings and digital gold investment, but there are many opportunities to expand its offerings. Below are five new business opportunities that could enhance user engagement and financial inclusion.

1. Micro-Investments in Mutual Funds & Stocks:

How It Works: Allow users to invest spare change into mutual funds, index funds, or stocks, just like digital gold.

Why It's Beneficial: Expands investment options, making wealth-building more accessible.

2. "Digital Piggy Bank" for Kids (Co-Saving Accounts):

How It Works: A parent (Jar user) can open a sub-account for their child (Minor Account) called a "Kid's Gullak." The parent can link a small, recurring auto-debit (e.g., a "daily allowance") or use the spare-change feature to save into this specific account. All transactions are controlled and monitored by the parent.

Why It's Beneficial: This taps into the emotional need of parents to save for their children. It creates an early customer acquisition. It also locks in the parent with a high-retention feature that increases the total assets under management (AUM) within the Jar ecosystem.

3. Automated Bill Payments & Smart Budgeting:

How It Works: Jar identifies recurring bills (utilities, rent, subscriptions) via linked accounts. It reserves required funds automatically and pays the bills on time. If cash is short, it can auto-sell a small amount of gold to cover the payment, preventing defaults.

Why It's Beneficial: It solves a core user pain point (missed payments). This increases daily app relevance and allows Jar to introduce a monthly subscription fee for the Auto-Pay Premium service.

4. Integration of Educational Modules – Knowledge Jar:

How It Works: Provide financial literacy courses, webinars, and investment tutorials within the app.

Why It's Beneficial: Empowers users with financial knowledge, improving their investment decisions. And the SEO of Jar can be improved by backlinking some financial education websites like – Investopedia, Zerodha Varsity, NCFE, Upsurge.

5. AI-Based Personalized Financial Advisor – Jar Wisdom:

How It Works: Jar Wisdom analyses savings, spending, and goals to provide personalized, conversational financial advice. It calculates risk scores and suggests the optimal balance between gold, SIPs, and other assets based on user data.

Why It's Beneficial: Solves the core user question: "What do I do next?" This democratic, data-driven advice is the basis for a "Jar Prime" subscription tier, guaranteeing a high-margin, recurring revenue stream by providing a value-add beyond simple saving.