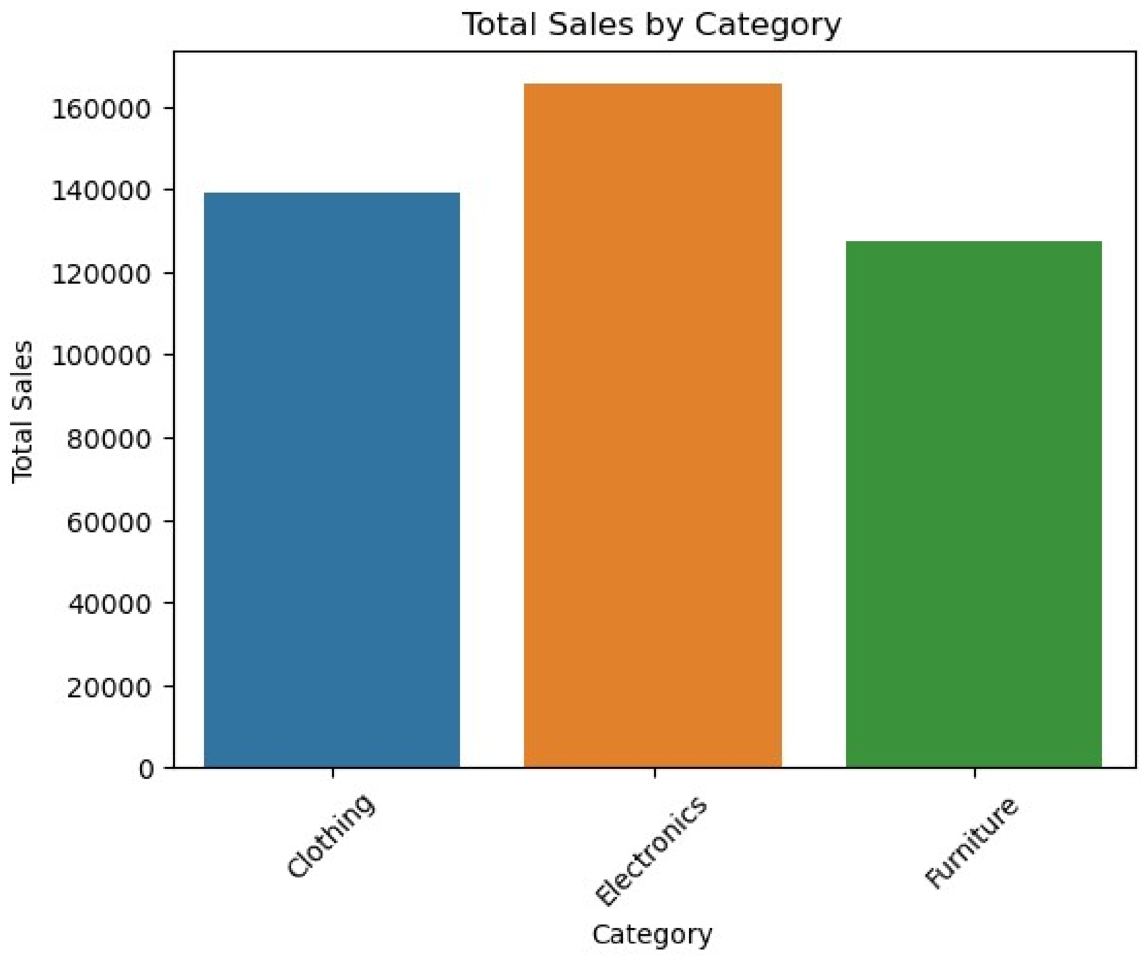
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Load datasets
list_of_orders = pd.read_csv('List_of_Orders_38EF37C2F3.csv')

order_details = pd.read_csv('Order_Details_F6D252B2DD.csv')

print(list_of_orders.head())  # Preview List of Orders
print(order_details.head())  # Preview Order Details

  Order ID  Order Date CustomerName           State       City
0  B-25601  01-04-2018       Bharat         Gujarat  Ahmedabad
1  B-25602  01-04-2018        Pearl     Maharashtra       Pune
2  B-25603  03-04-2018        Jahan  Madhya Pradesh     Bhopal
3  B-25604  03-04-2018       Divsha       Rajasthan     Jaipur
4  B-25605  05-04-2018      Kasheen     West Bengal    Kolkata
  Order ID  Amount  Profit  Quantity     Category      Sub-Category
0  B-25601  1275.0 -1148.0         7    Furniture         Bookcases
1  B-25601    66.0   -12.0         5     Clothing             Stole
2  B-25601     8.0    -2.0         3     Clothing       Hankerchief
3  B-25601    80.0   -56.0         4  Electronics  Electronic Games
4  B-25602   168.0  -111.0         2  Electronics            Phones

# Perform a Left Join to include all rows from List of Orders
merged_data = pd.merge(list_of_orders, order_details, on='Order ID',
how='left')

# Preview the merged data
print(merged_data.head())

  Order ID  Order Date CustomerName        State       City  Amount
Profit  \
0  B-25601  01-04-2018       Bharat      Gujarat  Ahmedabad  1275.0 -
1148.0
1  B-25601  01-04-2018       Bharat      Gujarat  Ahmedabad    66.0
-12.0
2  B-25601  01-04-2018       Bharat      Gujarat  Ahmedabad     8.0
-2.0
3  B-25601  01-04-2018       Bharat      Gujarat  Ahmedabad    80.0
-56.0
4  B-25602  01-04-2018        Pearl  Maharashtra       Pune   168.0  -
111.0

   Quantity     Category      Sub-Category
0       7.0    Furniture         Bookcases
1       5.0     Clothing             Stole
2       3.0     Clothing       Hankerchief
3       4.0  Electronics  Electronic Games
4       2.0  Electronics            Phones
```

Calculate Total Sales for Each Category

```
# Group by Category and calculate the total sales
total_sales_by_category = merged_data.groupby('Category')
['Amount'].sum().reset_index()

# Rename columns for clarity
total_sales_by_category.columns = ['Category', 'Total Sales']

# Print the result
print(total_sales_by_category)
```

```
      Category  Total Sales
0     Clothing     139054.0
1  Electronics     165267.0
2    Furniture     127181.0
```

```
total_sales_by_category.to_csv('Total_Sales_By_Category.csv',
index=False)

# Create a bar plot
sns.barplot(x='Category', y='Total Sales',
data=total_sales_by_category)
plt.title('Total Sales by Category')
plt.xticks(rotation=45)
plt.show()
```

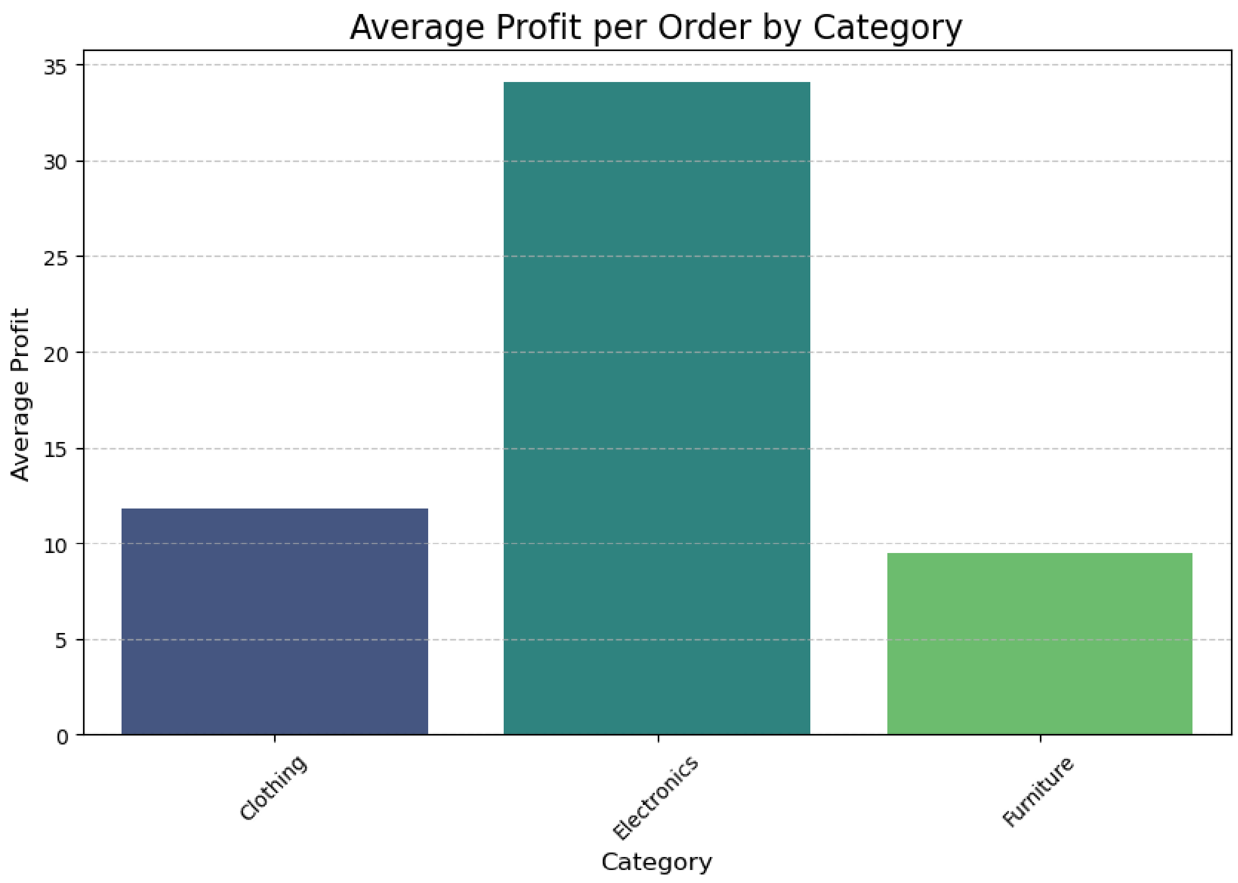


```
 #Average Profit per Order
avg_profit_by_category = merged_data.groupby('Category')
['Profit'].mean().reset_index()
avg_profit_by_category.columns = ['Category', 'Average Profit']

# Display the result
print(avg_profit_by_category)
```

```
      Category  Average Profit
0     Clothing       11.762908
1  Electronics       34.071429
2    Furniture        9.456790
```

```
# Profit Margin
merged_data['Profit Margin'] = (merged_data['Profit'] /
merged_data['Amount']) * 100
profit_margin_by_category = merged_data.groupby('Category')['Profit
Margin'].mean().reset_index()
profit_margin_by_category.columns = ['Category', 'Average Profit
Margin (%)']
```

```
# Combine Metrics
category_performance = pd.merge(total_sales_by_category,
avg_profit_by_category, on='Category')
category_performance = pd.merge(category_performance,
profit_margin_by_category, on='Category')

# Sort and Identify Performance
top_categories = category_performance.sort_values(by='Total Sales',
ascending=False)
underperforming_categories =
category_performance.sort_values(by='Average Profit Margin (%)')

# Print Results
print("Category Performance Summary:")
print(category_performance)

Category Performance Summary:
      Category  Total Sales  Average Profit  Average Profit Margin (%)
0     Clothing     139054.0       11.762908                   4.132921
1  Electronics     165267.0       34.071429                  -0.622928
2    Furniture     127181.0        9.456790                  -6.788811

print("\nTop Performing Categories:")
print(top_categories)


Top Performing Categories:
      Category  Total Sales  Average Profit  Average Profit Margin (%)
1  Electronics     165267.0       34.071429                  -0.622928
0     Clothing     139054.0       11.762908                   4.132921
2    Furniture     127181.0        9.456790                  -6.788811

print("\nUnderperforming Categories:")
print(underperforming_categories)


Underperforming Categories:
      Category  Total Sales  Average Profit  Average Profit Margin (%)
2    Furniture     127181.0        9.456790                  -6.788811
1  Electronics     165267.0       34.071429                  -0.622928
0     Clothing     139054.0       11.762908                   4.132921

# Plot the data using seaborn
plt.figure(figsize=(10, 6))  # Set the figure size
sns.barplot(x='Category', y='Average Profit',
data=avg_profit_by_category, palette='viridis')

# Add titles and labels
plt.title('Average Profit per Order by Category', fontsize=16)
plt.xlabel('Category', fontsize=12)
plt.ylabel('Average Profit', fontsize=12)
plt.xticks(rotation=45, fontsize=10)  # Rotate x-axis labels for
better readability
plt.grid(axis='y', linestyle='--', alpha=0.7)  # Add horizontal
gridlines

# Show the plot
plt.show()
```



```
plt.savefig('average_profit_by_category.png', dpi=300)

<Figure size 640x480 with 0 Axes>
```

Part 2: Target Achievement Analysis

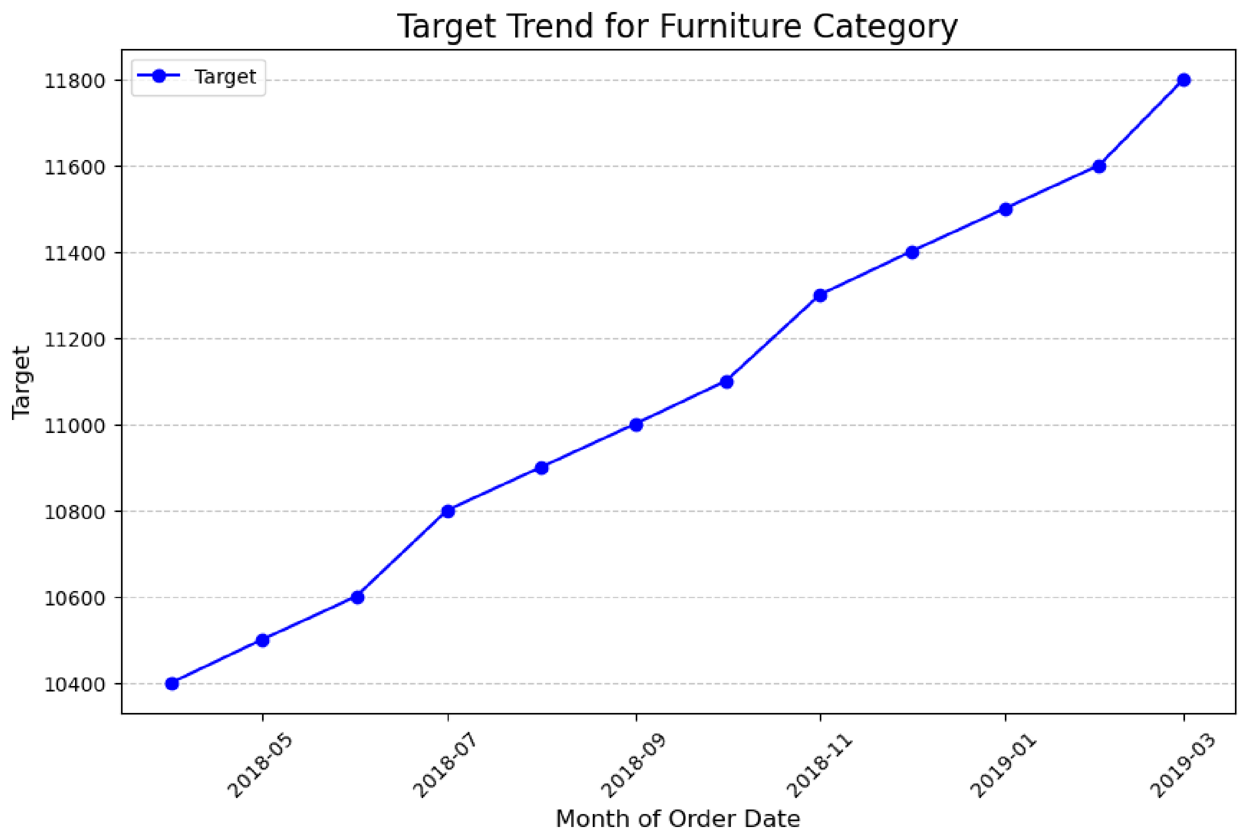
```
# Load and filter dataset
sales_target = pd.read_csv('Sales_target_1C8295CCDE.csv')
furniture_sales = sales_target[sales_target['Category'] ==
'Furniture']

print(furniture_sales.head(10))
```

```
  Month of Order Date   Category   Target
0              Apr-18  Furniture  10400.0
1              May-18  Furniture  10500.0
2              Jun-18  Furniture  10600.0
3              Jul-18  Furniture  10800.0
4              Aug-18  Furniture  10900.0
5              Sep-18  Furniture  11000.0
6              Oct-18  Furniture  11100.0
7              Nov-18  Furniture  11300.0
8              Dec-18  Furniture  11400.0
9              Jan-19  Furniture  11500.0

# Ensure 'Month of Order Date' is in datetime format and sort
furniture_sales['Month of Order Date'] =
pd.to_datetime(furniture_sales['Month of Order Date'], format='%b-%y')
furniture_sales = furniture_sales.sort_values(by='Month of Order
Date')

# Display the sorted data
print(furniture_sales.head(10))

  Month of Order Date   Category   Target
0          2018-04-01  Furniture  10400.0
1          2018-05-01  Furniture  10500.0
2          2018-06-01  Furniture  10600.0
3          2018-07-01  Furniture  10800.0
4          2018-08-01  Furniture  10900.0
5          2018-09-01  Furniture  11000.0
6          2018-10-01  Furniture  11100.0
7          2018-11-01  Furniture  11300.0
8          2018-12-01  Furniture  11400.0
9          2019-01-01  Furniture  11500.0

C:\Users\Minnat Alam\AppData\Local\Temp\
ipykernel_18540\1045223797.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  furniture_sales['Month of Order Date'] =
pd.to_datetime(furniture_sales['Month of Order Date'], format='%b-%y')

# Calculate percentage change
furniture_sales['Pct_Change'] = furniture_sales['Target'].pct_change()
* 100

# Identify significant fluctuations
significant_fluctuations =
furniture_sales[abs(furniture_sales['Pct_Change']) > 20]
```

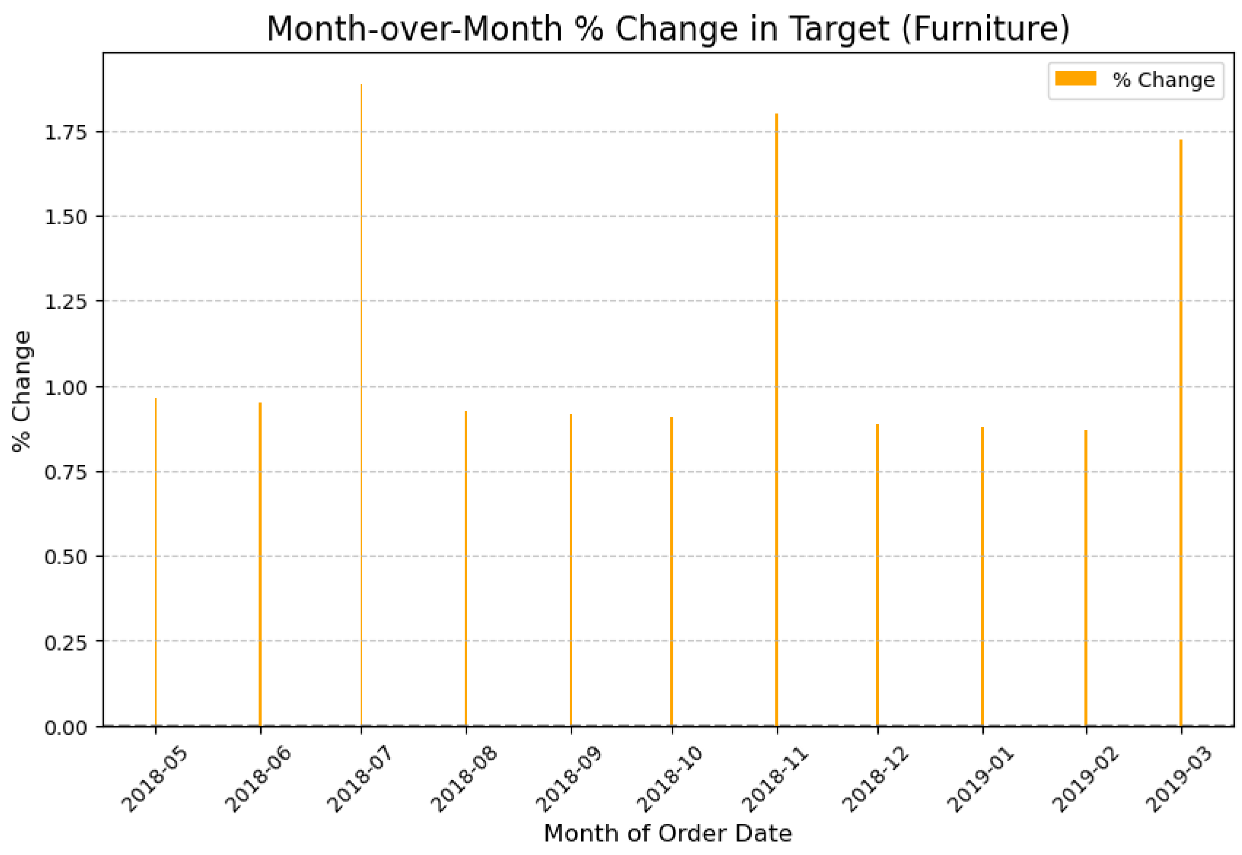
```
print("Months with significant target fluctuations:")
print(significant_fluctuations[['Month of Order Date', 'Target',
'Pct_Change']])
```

```
Months with significant target fluctuations:
Empty DataFrame
Columns: [Month of Order Date, Target, Pct_Change]
Index: []
```

```
# Plot target sales trend
plt.figure(figsize=(10, 6))
plt.plot(furniture_sales['Month of Order Date'],
furniture_sales['Target'], marker='o', label='Target', color='blue')
plt.title('Target Trend for Furniture Category', fontsize=16)
plt.xlabel('Month of Order Date', fontsize=12)
plt.ylabel('Target', fontsize=12)
plt.xticks(rotation=45)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.legend()
plt.show()
```



```
# Plot percentage change trend
plt.figure(figsize=(10, 6))
plt.bar(furniture_sales['Month of Order Date'],
furniture_sales['Pct_Change'], color='orange', label='% Change')
plt.title('Month-over-Month % Change in Target (Furniture)',
fontsize=16)
plt.xlabel('Month of Order Date', fontsize=12)
plt.ylabel('% Change', fontsize=12)
plt.axhline(0, color='gray', linestyle='--')
plt.xticks(rotation=45)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.legend()
plt.show()
```



Part 3: Regional Performance Insights

```
# Load datasets
list_of_orders = pd.read_csv('List_of_Orders_38EF37C2F3.csv')

print(list_of_orders.head())
```

```
  Order ID  Order Date CustomerName           State       City
0  B-25601  01-04-2018       Bharat         Gujarat  Ahmedabad
1  B-25602  01-04-2018        Pearl     Maharashtra       Pune
2  B-25603  03-04-2018        Jahan  Madhya Pradesh     Bhopal
```

```
3  B-25604  03-04-2018       Divsha       Rajasthan     Jaipur
4  B-25605  05-04-2018      Kasheen     West Bengal    Kolkata
```

```python
# Top 5 states by order count
state_order_counts = list_of_orders.groupby('State')['Order
ID'].count().reset_index()
state_order_counts.columns = ['State', 'Order Count']
top_5_states = state_order_counts.sort_values(by='Order Count',
ascending=False).head(5)

# Filter for top 5 states
top_5_state_names = top_5_states['State'].tolist()
filtered_orders =
list_of_orders[list_of_orders['State'].isin(top_5_state_names)]

# Calculate order count, total sales, and average profit
state_performance = merged_data.groupby('State').agg(
    Order_Count=('Order ID', 'count'),
    Total_Sales=('Amount', 'sum'),
    Average_Profit=('Profit', 'mean')
).reset_index()

# Get the top 5 states by order count
top_5_states = state_performance.sort_values(by='Order_Count',
ascending=False).head(5)

print("Top 5 States by Order Count:")
print(top_5_states)
```

```
Top 5 States by Order Count:
             State  Order_Count  Total_Sales  Average_Profit
10  Madhya Pradesh          340     105140.0       16.326471
11     Maharashtra          290      95348.0       21.296552
4          Gujarat           87      21058.0        5.344828
2            Delhi           74      22531.0       40.364865
14       Rajasthan           74      21149.0       16.986486
```

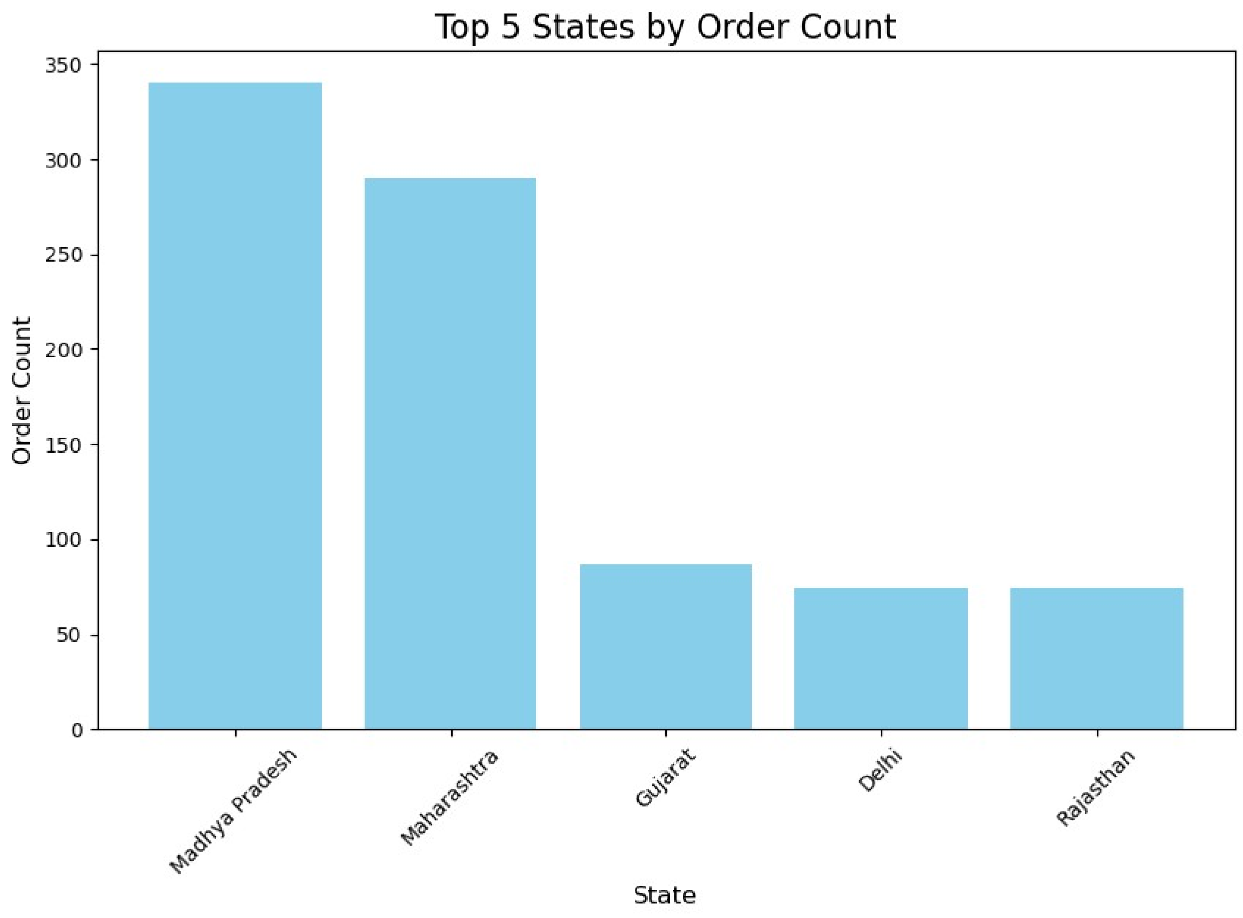
```python
# Filter for Furniture category
furniture_target = sales_target[sales_target['Category'] ==
'Furniture']

# Ensure the Month column is in datetime format
furniture_target['Month of Order Date'] =
pd.to_datetime(furniture_target['Month of Order Date'], format='%b-
%y')

# Sort by month and calculate percentage change
furniture_target = furniture_target.sort_values(by='Month of Order
Date')
furniture_target['Pct_Change'] =
furniture_target['Target'].pct_change() * 100
```

```
print("Furniture Target Sales with Percentage Change:")
print(furniture_target)
```

```
Furniture Target Sales with Percentage Change:
   Month of Order Date   Category   Target  Pct_Change
0           2018-04-01  Furniture  10400.0         NaN
1           2018-05-01  Furniture  10500.0    0.961538
2           2018-06-01  Furniture  10600.0    0.952381
3           2018-07-01  Furniture  10800.0    1.886792
4           2018-08-01  Furniture  10900.0    0.925926
5           2018-09-01  Furniture  11000.0    0.917431
6           2018-10-01  Furniture  11100.0    0.909091
7           2018-11-01  Furniture  11300.0    1.801802
8           2018-12-01  Furniture  11400.0    0.884956
9           2019-01-01  Furniture  11500.0    0.877193
10          2019-02-01  Furniture  11600.0    0.869565
11          2019-03-01  Furniture  11800.0    1.724138

C:\Users\Minnat Alam\AppData\Local\Temp\
ipykernel_18540\1926538683.py:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  furniture_target['Month of Order Date'] =
pd.to_datetime(furniture_target['Month of Order Date'], format='%b-
%y')
```

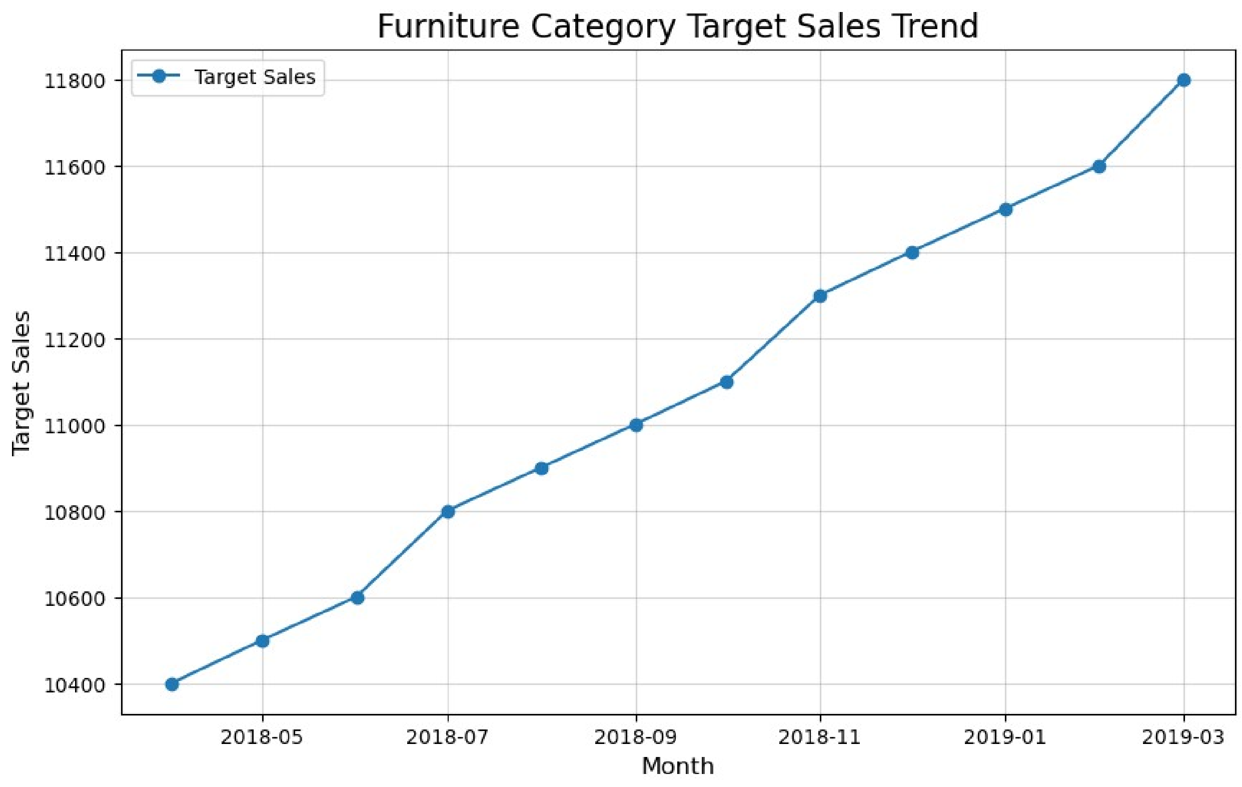
Regional Performance

```
import matplotlib.pyplot as plt

# Bar chart for top 5 states by order count
plt.figure(figsize=(10, 6))
plt.bar(top_5_states['State'], top_5_states['Order_Count'],
color='skyblue')
plt.title('Top 5 States by Order Count', fontsize=16)
plt.xlabel('State', fontsize=12)
plt.ylabel('Order Count', fontsize=12)
plt.xticks(rotation=45)
plt.show()
```



Target Achievement Trend

```
# Line chart for target sales
plt.figure(figsize=(10, 6))
plt.plot(furniture_target['Month of Order Date'],
furniture_target['Target'], marker='o', label='Target Sales')
plt.title('Furniture Category Target Sales Trend', fontsize=16)
plt.xlabel('Month', fontsize=12)
plt.ylabel('Target Sales', fontsize=12)
plt.grid(alpha=0.5)
plt.legend()
plt.show()
```



App Exploration:

```
Five Effective and User-Friendly Features
Automated Micro-Savings:

Automatically rounds up transactions and invests the spare change in
digital gold, making saving seamless for users.
Helps inculcate a savings habit without requiring active effort.
Intuitive User Interface (UI):

Clean and simple design with minimal distractions.
Easy navigation, ensuring first-time users can understand the features
quickly.
Low Investment Threshold:

Users can start investing with as little as ₹10, making it accessible
to a broad audience.
Real-Time Gold Prices:

Transparent display of live gold rates ensures users can track their
investments and make informed decisions.
Instant Notifications and Insights:

Immediate updates on savings and investment progress keep users
engaged and motivated to save more.
```

```
Five Areas for Improvement
Limited Investment Options:

Currently focuses only on digital gold. Expanding to mutual funds,
stocks, or fixed deposits could attract a more diverse audience.
Lack of Financial Education:

The app could include interactive tutorials or tools to educate users
about the benefits and risks of investing in digital gold and other
assets.
Personalized Goals and Recommendations:

Adding goal-setting features (e.g., save for a trip or gadget) and
tailored advice based on user behavior could enhance engagement.
Customer Support:

Availability of live chat or 24/7 support would improve user
confidence, especially for new investors unfamiliar with digital gold.
Reward Mechanisms:

Incorporating gamification or reward systems (e.g., badges, cashback)
for reaching savings milestones could make the app more engaging.
```

```
  Cell In[60], line 12
    Users can start investing with as little as ₹10, making it
accessible to a broad audience.
                                                ^
SyntaxError: invalid character '₹' (U+20B9)
```

Product Exploration:

```
New Business Opportunities for Jar App
Expand into Mutual Funds, SIPs, and Goal-Based Savings: Offer
Systematic Investment Plans (SIPs), mutual funds, and recurring
savings options for specific goals (e.g., vacations, education),
leveraging automation to simplify investments.

Introduce Micro-Insurance and Micro-Credit Services: Provide
affordable health, term, or accident insurance plans, and offer small,
instant loans or credit lines based on user savings and spending
patterns to address short-term financial needs.

Personalized Financial Insights and Coaching: Use AI-driven
recommendations and insights tailored to user behavior, helping them
improve financial literacy and make smarter financial decisions.

Integrate Broader Investment Options: Expand beyond digital gold to
include ETFs, bonds, or fixed deposits, allowing users to diversify
```

```
their investments within the app.

Gamification and Rewards for Engagement: Implement gamified elements
like milestones, badges, or cashback rewards for achieving savings
goals, enhancing user engagement and retention.
```