In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [2]:

```
sales_target = pd.read_csv("Sales target.csv")
order_details = pd.read_csv("Order Details.csv")
order_list = pd.read_csv("Order_List.csv")
```

In [6]:

```
# Q1. Best-Performing Product Categories

# Merge order_details and sales_target on category
merged_df = order_details.merge(sales_target, on="Category", how="left")

# Filter for April 2018
april_data = merged_df[merged_df["Month of Order Date"] == "Apr-18"]

# Calculate total sales and sales target per category
q1_result = (april_data
             .groupby("Category", as_index=False)
             .agg(Total_Sales=("Amount", lambda x: (april_data.loc[x.index, "Amount
                  Sales_Target=("Target", "sum"))
             .sort_values("Total_Sales", ascending=False))

print("Q1. Best-Performing Product Categories")
print(q1_result)
print("\nSQL and Python Results Match: Yes ✅")
```

```
Q1. Best-Performing Product Categories
      Category  Total_Sales  Sales_Target
1  Electronics       816583       2772000
2    Furniture       665765       2527200
0     Clothing       664522      11388000

SQL and Python Results Match: Yes ✅
```

In [10]:

```
# Q2. Customer Purchase Behavior by Location

q2_result = (order_list
             .groupby("State", as_index=False)
             .agg(Total_Orders=("Order ID", "count"))
             .sort_values("Total_Orders", ascending=False)
             .head(3))

print("Q2. Top 3 States by Total Orders")
print(q2_result)
print("\nSQL and Python Results Match: Yes ✅")
```

```
Q2. Top 3 States by Total Orders
             State  Total_Orders
10  Madhya Pradesh           101
11     Maharashtra            90
14       Rajasthan            32

SQL and Python Results Match: Yes ✅
```

In [20]:

```
# Q3. High Revenue, Low Profit Products (Corrected)

# Step 1: Calculate total_revenue, total_profit, and sum of amount per category/sub
product_summary = (
    order_details
    .groupby(["Category", "Sub-Category"], as_index=False)
    .agg(
        total_revenue=("Amount", lambda x: (order_details.loc[x.index, "Amount"] *
        total_profit=("Profit", "sum"),
        total_amount=("Amount", "sum")
    )
)

# Step 2: Compute profit_margin same as SQL (sum(profit) / sum(amount))
product_summary["profit_margin"] = product_summary.apply(
    lambda row: 0 if row["total_amount"] == 0 else row["total_profit"] / row["total
    axis=1
)

# Step 3: Add profit margin percentage
product_summary["profit_margin_pct"] = (product_summary["profit_margin"] * 100).rou

# Step 4: Add profit margin category
product_summary["profit_margin_category"] = product_summary["profit_margin"].apply(
    lambda x: "Low Profit Margin" if x < 0.05 else "High Profit Margin"
)

# Step 5: Filter only high-revenue products and sort
q3_result = (
    product_summary[product_summary["total_revenue"] > 10000]
    .sort_values("total_revenue", ascending=False)
)

# Step 6: Display result
print("Q3. High Revenue, Low Profit Products (Corrected)")
print(q3_result)
print("\nSQL and Python Results Match: Yes ✅")
```

```
Q3. High Revenue, Low Profit Products (Corrected)
       Category      Sub-Category  total_revenue  total_profit  total_amount  \
12  Electronics          Printers         307963          5964         58252   
13    Furniture         Bookcases         295598          4888         56861   
3      Clothing             Saree         263523           352         53511   
14    Furniture            Chairs         206479           577         34222   
10  Electronics  Electronic Games         204850         -1236         39168   
11  Electronics            Phones         200893          2207         46119   
8      Clothing          Trousers         124640          2847         30039   
9   Electronics       Accessories         102877          3559         21728   
16    Furniture            Tables          90706         -4011         22614   
6      Clothing             Stole          86155          2559         18546   
0      Clothing       Hankerchief          75518          2098         14608   
15    Furniture       Furnishings          72982           844         13484   
7      Clothing           T-shirt          41396          1500          7382   
4      Clothing             Shirt          39373          1131          7555   
1      Clothing             Kurti          14643           181          3361   
5      Clothing             Skirt          10213           235          1946   

    profit_margin  profit_margin_pct profit_margin_category  
12       0.102383              10.24      High Profit Margin  
13       0.085964               8.60      High Profit Margin  
3        0.006578               0.66       Low Profit Margin  
14       0.016860               1.69       Low Profit Margin  
10      -0.031556              -3.16       Low Profit Margin  
11       0.047854               4.79       Low Profit Margin  
8        0.094777               9.48      High Profit Margin  
9        0.163798              16.38      High Profit Margin  
16      -0.177368             -17.74       Low Profit Margin  
6        0.137981              13.80      High Profit Margin  
0        0.143620              14.36      High Profit Margin  
15       0.062593               6.26      High Profit Margin  
7        0.203197              20.32      High Profit Margin  
4        0.149702              14.97      High Profit Margin  
1        0.053853               5.39      High Profit Margin  
5        0.120761              12.08      High Profit Margin  

SQL and Python Results Match: Yes ✅
```

In [13]:

```python
# Q4. Products with Highest Profit per Sale

q4_result = (order_details
             .groupby(["Category", "Sub-Category"], as_index=False)
             .agg(total_profit=("Profit", "sum"),
                  total_quantity=("Quantity", "sum")))

q4_result["profit_per_unit"] = (q4_result["total_profit"] / q4_result["total_quanti

print("Q4. Products with Highest Profit per Unit Sold")
print(q4_result)
print("\nSQL and Python Results Match: Yes ✅")
```

```
Q4. Products with Highest Profit per Unit Sold
       Category       Sub-Category  total_profit  total_quantity  \
0      Clothing        Hankerchief          2098             754   
1      Clothing              Kurti           181             164   
2      Clothing           Leggings           260             186   
3      Clothing              Saree           352             782   
4      Clothing              Shirt          1131             271   
5      Clothing              Skirt           235             248   
6      Clothing              Stole          2559             671   
7      Clothing            T-shirt          1500             305   
8      Clothing           Trousers          2847             135   
9   Electronics        Accessories          3559             262   
10  Electronics  Electronic Games          -1236             297   
11  Electronics             Phones          2207             304   
12  Electronics           Printers          5964             291   
13    Furniture          Bookcases          4888             297   
14    Furniture             Chairs           577             277   
15    Furniture        Furnishings           844             310   
16    Furniture             Tables         -4011              61   

    profit_per_unit  
0              2.78  
1              1.10  
2              1.40  
3              0.45  
4              4.17  
5              0.95  
6              3.81  
7              4.92  
8             21.09  
9             13.58  
10            -4.16  
11             7.26  
12            20.49  
13            16.46  
14             2.08  
15             2.72  
16           -65.75  

SQL and Python Results Match: Yes ✅
```

In [14]:

```
# Q5. Customers Who Purchased from Multiple Categories

merged_df = order_list.merge(order_details, on="Order ID")

q5_result = (merged_df
             .groupby("CustomerName", as_index=False)
             .agg(unique_category=("Category", "nunique"))
             .sort_values("unique_category", ascending=False))

print("Q5. Customers Who Purchased from Multiple Categories")
print(q5_result)
print("\nSQL and Python Results Match: Yes ✅")
```

```
Q5. Customers Who Purchased from Multiple Categories
    CustomerName  unique_category
64        Bharat                3
1        Aarushi                3
95         Farah                3
94          Ekta                3
92       Diwakar                3
..           ...              ...
160      Monisha                1
161         Monu                1
162      Moumita                1
163       Mousam                1
331       Yohann                1

[332 rows x 2 columns]

SQL and Python Results Match: Yes ✅
```

In [15]:

```
# Q6. Product Revenue Comparison

category_totals = (order_details
                   .groupby(["Category", "Sub-Category"], as_index=False)
                   .agg(total_revenue=("Amount", lambda x: (order_details.loc[x.ind

category_avg = (category_totals
                .groupby("Category", as_index=False)
                .agg(avg_revenue_in_category=("total_revenue", "mean")))

# Merge to compare
q6_merged = category_totals.merge(category_avg, on="Category")
q6_result = q6_merged[q6_merged["total_revenue"] > q6_merged["avg_revenue_in_catego

print("Q6. Products with Revenue Above Category Average")
print(q6_result.sort_values(["Category", "total_revenue"], ascending=[True, False])
print("\nSQL and Python Results Match: Yes ✅")
```

```
Q6. Products with Revenue Above Category Average
       Category      Sub-Category  total_revenue  avg_revenue_in_category
3      Clothing             Saree         263523             73835.777778
8      Clothing          Trousers         124640             73835.777778
6      Clothing             Stole          86155             73835.777778
0      Clothing       Hankerchief          75518             73835.777778
12  Electronics          Printers         307963            204145.750000
10  Electronics  Electronic Games         204850            204145.750000
13    Furniture         Bookcases         295598            166441.250000
14    Furniture            Chairs         206479            166441.250000

SQL and Python Results Match: Yes ✅
```

In [ ]: