Name: Minh Trinh - Data Analyst Intern

Assignment by: FPT Corporation

Section #1: Data Exploration:

Explore, clean (if necessary) and describe the dataset

Section #2: Business Acumen

Main strategy of the company is to maximize GMV and optimize spending

● Identify and measure at least three key suitable metrics to analyse the company performance and deliver insights for improvement

● Explain your rationale behind those metrics

● Provide supporting analysis and visualization to communicate your ideas to the CEO

● Which additional data/dataset do you think the company should collect and why

How you will complete the tasks

1. You can use any analysis tool to complete the tasks
2. Expected Output:

- PDF/PowerPoint file shows your analysis
- Technical work (SQL, Python, R, Power BI) along with a description of the steps you take to solve it

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.impute import KNNImputer
```

In [2]:
```
customers = pd.read_csv('olist_customers_dataset.csv')
geo = pd.read_csv("olist_geolocation_dataset.csv")
items = pd.read_csv("olist_order_items_dataset.csv")
payments = pd.read_csv("olist_order_payments_dataset.csv")
reviews = pd.read_csv("olist_order_reviews_dataset.csv")
orders = pd.read_csv("olist_orders_dataset.csv")
products = pd.read_csv("olist_products_dataset.csv")
sellers = pd.read_csv("olist_sellers_dataset.csv")
translation = pd.read_csv("product_category_name_translation.csv")
```

In [3]:
```
#Data Description

from IPython.display import display, HTML
```

```
# Assuming you have the following dataframes: items, payments, reviews, orde

# Define a function to format the dataframe as an HTML table
def display_table(df):
    display(HTML(df.to_html()))

# Print the description of the 'items' dataframe
print("Items Description:")
display_table(items.describe())

# Print the description of the 'payments' dataframe
print("Payments Description:")
display_table(payments.describe())

# Print the description of the 'reviews' dataframe
print("Reviews Description:")
display_table(reviews.describe())

# Print the description of the 'products' dataframe
print("Products Description:")
display_table(products.describe())
```

```
Items Description:
```

	order_item_id	price	freight_value
count	112650.000000	112650.000000	112650.000000
mean	1.197834	120.653739	19.990320
std	0.705124	183.633928	15.806405
min	1.000000	0.850000	0.000000
25%	1.000000	39.900000	13.080000
50%	1.000000	74.990000	16.260000
75%	1.000000	134.900000	21.150000
max	21.000000	6735.000000	409.680000

```
Payments Description:
```

	payment_sequential	payment_installments	payment_value
count	103886.000000	103886.000000	103886.000000
mean	1.092679	2.853349	154.100380
std	0.706584	2.687051	217.494064
min	1.000000	0.000000	0.000000
25%	1.000000	1.000000	56.790000
50%	1.000000	1.000000	100.000000
75%	1.000000	4.000000	171.837500
max	29.000000	24.000000	13664.080000

```
Reviews Description:
```

	review_score
count	99224.000000
mean	4.086421
std	1.347579
min	1.000000
25%	4.000000
50%	5.000000
75%	5.000000
max	5.000000

```
Products Description:
```

	product_name_lenght	product_description_lenght	product_photos_qty	product_weig
count	32341.000000	32341.000000	32341.000000	32949.00
mean	48.476949	771.495285	2.188986	2276.47
std	10.245741	635.115225	1.736766	4282.03
min	5.000000	4.000000	1.000000	0.00
25%	42.000000	339.000000	1.000000	300.00
50%	51.000000	595.000000	1.000000	700.00
75%	57.000000	972.000000	3.000000	1900.00
max	76.000000	3992.000000	20.000000	40425.00

In [4]:
```
# Select columns to impute
products_to_impute = products[['product_name_lenght', 'product_description_
                               'product_length_cm', 'product_height_cm', 'p

# Initialize the KNNImputer
knn_imputer = KNNImputer(n_neighbors=5)

# Apply the imputer to the DataFrame
products_imputed = pd.DataFrame(knn_imputer.fit_transform(products_to_impute

# Merge the imputed columns back into the original dataset
products[products_to_impute.columns] = products_imputed

products['product_category_name'].fillna("Others", inplace=True)

na_counts = products.isna().sum()

print(na_counts)
```

```
product_id                    0
product_category_name         0
product_name_lenght           0
product_description_lenght    0
product_photos_qty            0
product_weight_g              0
product_length_cm             0
product_height_cm             0
product_width_cm              0
dtype: int64
```

In [5]:
```
#Merge the datasets together into one big dataset, which includes all neces
df = pd.merge(customers, orders, on='customer_id')
```

```
#df = pd.merge(df, geo, left_on='customer_zip_code_prefix', right_on = "geol
#df = pd.merge(df, sellers, left_on='geolocation_zip_code_prefix', right_on
#df = pd.merge(df, reviews, on='order_id')
df = pd.merge(df, items, on='order_id')
#df = pd.merge(df, sellers, on='seller_id')
#df = pd.merge(df, payments, on='order_id')
#df = pd.merge(df, products, on='product_id')

df = df.drop_duplicates()


df
```

Out[5]:

	customer_id	customer_unique_id	custome
0	06b8999e2fba1a1fbc88172c00ba8bc7	861eff4711a542e4b93843c6dd7febb0	
1	18955e83d337fd6b2def6b18a428ac77	290c77bc529b7ac935b93aa66c333dc3	
2	4e7b3e00288586ebd08712fdd0374a03	060e732b5b29e8181a18229c7b0b2b5e	
3	b2b6027bc5c5109e529d4dc6358b12c3	259dac757896d24d7702b9acbbff3f3c	
4	4f2d8ab171c80ec8364f7c12e35b23ad	345ecd01c38d18a9036ed96c73b8d066	
...	...	...	
112645	17ddf5dd5d51696bb3d7c6291687be6f	1a29b476fee25c95fbafc67c5ac95cf8	
112646	e7b71a9017aa05c9a7fd292d714858e8	d52a67c98be1cf6a5c84435bd38d095d	
112647	5e28dfe12db7fb50a4b2f691faecea5e	e9f50caf99f032f0bf3c55141f019d99	
112648	56b18e2166679b8a959d72dd06da27f9	73c2643a0a458b49f58cea58833b192e	
112649	274fa6071e5e17fe303b9748641082c8	84732c5050c01db9b23e19ba39899398	

112650 rows × 18 columns

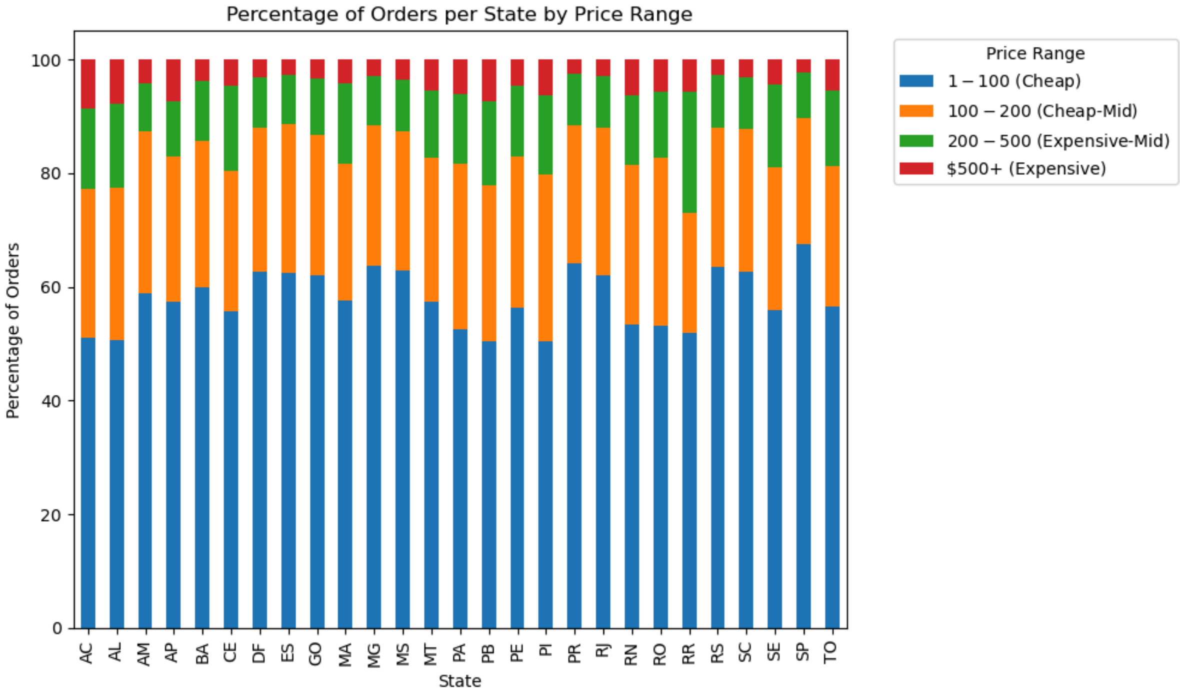
Visualization 1

In [6]:

```
# Define bins and labels for price ranges
bins = [0,100,200, 500, float('inf')]
labels = ['$1-$100 (Cheap)', '$100-$200 (Cheap-Mid)', '$200-$500 (Expensive-

# Create a new column with the price ranges
df['price_range'] = pd.cut(df['price'], bins=bins, labels=labels, right=Fals

# Group by 'seller_state' and 'price_range' and count number of orders
customer_state_price_range = df.groupby(['customer_state', 'price_range']).s

# Pivot the table to have price ranges as columns
pivot_table = customer_state_price_range.pivot(index='customer_state', colum

# Calculate the percentage of orders in each price range for each state
```

```
pivot_table_percentage = pivot_table.div(pivot_table.sum(axis=1), axis=0) *

# Plotting the results with a larger figure size
plt.figure(figsize=(15, 8))
ax = pivot_table_percentage.plot(kind='bar', stacked=True, figsize=(12, 6))
plt.xlabel("State")
plt.ylabel("Percentage of Orders")
plt.title("Percentage of Orders per State by Price Range")

# Place the legend outside of the plot
plt.legend(title="Price Range", bbox_to_anchor=(1.05, 1), loc='upper left')
plt.tight_layout(rect=[0, 0, 0.85, 1])

plt.show()
```

```
<Figure size 1500x800 with 0 Axes>
```



Rationale #1

The rationale behind using the metric of order counts by price category is to understand the purchasing patterns and customer preferences based on different price ranges. By analyzing the number of orders in each price category, businesses can gain insights into the demand for products at different price points and make informed decisions regarding pricing strategies, product development, and marketing efforts.

My Interpretation:

The graph shows the distribution of orders across different price categories. It indicates that approximately 60% of the orders fall into the low-cost category (1-100), while the number of orders decreases as the price of the products increases. This trend can be attributed to consumer preferences for more affordable options and the higher demand for products at lower price points.

The data suggests that customers are more inclined to purchase products in the lower price range, which aligns with the general consumer behavior of seeking value for money. As the price increases, the demand for products decreases, indicating that customers may be more selective or price-sensitive when it comes to higher-priced items.

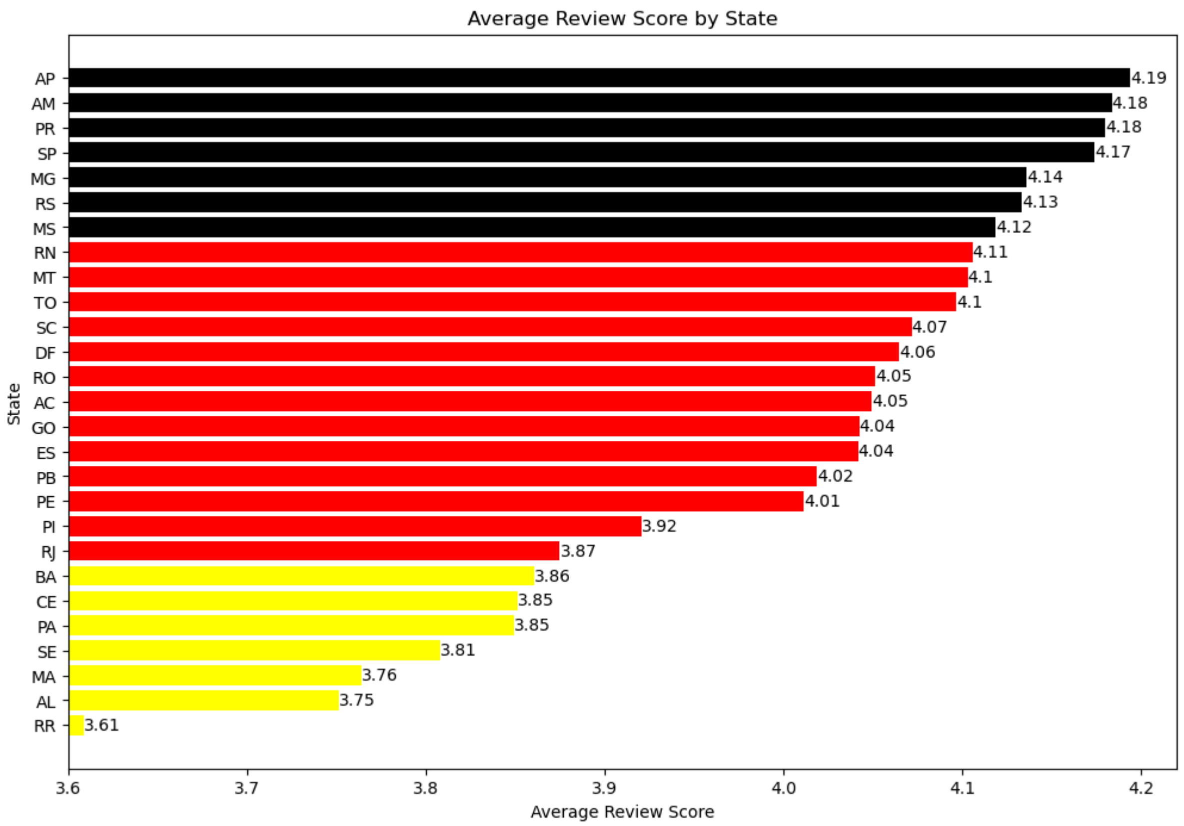
In [7]:
```
# Group by 'customer_state' and count number of orders
orders_per_state = df.groupby('customer_state').size().reset_index(name='or

# Order the table by 'order_count' in descending order
orders_per_state = orders_per_state.sort_values(by='order_count', ascending=

# Reset the index
orders_per_state = orders_per_state.reset_index(drop=True)

# Set the index to start from 1
orders_per_state.index = orders_per_state.index + 1

# Display the table
print(orders_per_state)
```

```
   customer_state  order_count
1              SP        47449
2              RJ        14579
3              MG        13129
4              RS         6235
5              PR         5740
6              SC         4176
7              BA         3799
8              DF         2406
9              GO         2333
10             ES         2256
11             PE         1806
12             CE         1478
13             PA         1080
14             MT         1055
15             MA          824
16             MS          819
17             PB          602
18             PI          542
19             RN          529
20             AL          444
21             SE          385
22             TO          315
23             RO          278
24             AM          165
25             AC           92
26             AP           82
27             RR           52
```

Visualization 2

In [8]:
```
import numpy as np
# Merge orders with customers to get state information
orders_customers = pd.merge(orders, customers, on='customer_id')

# Merge the resulting dataset with reviews to get review scores
orders_customers_reviews = pd.merge(orders_customers, reviews, on='order_id

# Group by state and calculate the average review score
state_review_scores = orders_customers_reviews.groupby('customer_state')['re
state_review_scores.columns = ['State', 'Average Review Score']
state_review_scores = state_review_scores.sort_values(by='Average Review Sco
```

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

plt.figure(figsize=(12, 8))

# Define the quartiles
q1 = np.percentile(state_review_scores['Average Review Score'], 25)
q3 = np.percentile(state_review_scores['Average Review Score'], 75)

# Plot the bars with different colors for each quartile segment
plt.barh(state_review_scores['State'], state_review_scores['Average Review S
         color=np.where(state_review_scores['Average Review Score'] < q1, 'y
                        np.where(state_review_scores['Average Review Score']

plt.xlabel('Average Review Score')
plt.ylabel('State')
plt.title('Average Review Score by State')
plt.gca().invert_yaxis()

# Set the y-axis limits
plt.xlim(3.6, 4.22)

for i, score in enumerate(state_review_scores['Average Review Score']):
    plt.text(score, i, str(round(score, 2)), ha='left', va='center')

plt.show()

#GERMANY FLAG
```



In [9]:
```
# Merge reviews with orders by order_id
reviews_orders = pd.merge(reviews, orders, on='order_id')

# Merge orders with items by order_id
orders_items = pd.merge(orders, items, on='order_id')

# Merge all datasets with products by product_id
merged_data = pd.merge(pd.merge(reviews_orders, orders_items, on='order_id'
```

```
# Group by product weight and calculate the average review score
weight_review_scores = merged_data.groupby(pd.cut(merged_data['product_weigh
weight_review_scores.columns = ['Weight Category', 'Average Review Score']
weight_review_scores = weight_review_scores.sort_values(by='Average Review S
weight_review_scores = weight_review_scores.reset_index(drop=True)

# Display the average review scores by weight category
print(weight_review_scores)
```

```
  Weight Category  Average Review Score
0          medium              4.059937
1           light              4.038472
2           heavy              4.005866
```

Weight categories correlation with Reviews:

Categorizing products into weight categories (light, medium, and heavy) allows for a meaningful comparison of average review scores across different weight ranges. This categorization helps to simplify the analysis and provides a clear framework for understanding the relationship between weight and review scores.

The average review score serves as a metric to assess customer satisfaction. By calculating the average review score for each weight category, we can determine how customers perceive products of different weights and identify any patterns or trends in customer satisfaction based on weight.

My interpretation on this is that, the differences in average review scores between the weight categories are relatively small. It is evident to conclude that the weight of products is not statistically significant to measure the satisfaction of customers towards the products.

In [10]:

```
# Merge orders with items by order_id
orders_items = pd.merge(orders, items, on='order_id')

# Merge all datasets with products by product_id
merged_data = pd.merge(pd.merge(reviews_orders, orders_items, on='order_id')

# Group by price and calculate the average review score
price_review_scores = merged_data.groupby(pd.cut(merged_data['price'], bins=
price_review_scores.columns = ['Price Category', 'Average Review Score']
price_review_scores = price_review_scores.sort_values(by='Average Review Sco
price_review_scores = price_review_scores.reset_index(drop=True)

# Group by price and calculate the count of orders
price_order_counts = merged_data.groupby(pd.cut(merged_data['price'], bins=
price_order_counts.columns = ['Price Category', 'Order Count']
price_order_counts = price_order_counts.sort_values(by='Order Count', ascend
price_order_counts = price_order_counts.reset_index(drop=True)

# Display the average review scores by price category
print(price_review_scores)
print('_______________________________________')
print(price_order_counts)

total_order_counts = price_order_counts['Order Count'].sum()
print("Total Order Counts:", total_order_counts)
```

```
  Price Category  Average Review Score
0      cheap-mid              4.056862
1  expensive-mid              4.032790
2          cheap              4.024688
3      expensive              4.003140
_______________________________________
  Price Category  Order Count
0          cheap        72222
1      cheap-mid        26837
2  expensive-mid        10125
3      expensive         3185
Total Order Counts: 112369
```

Rationale #2

The average review score is used as a metric to assess customer satisfaction in the context of evaluating the overall satisfaction level of customers. By analyzing the average review scores across different price ranges, businesses can gain insights into the relationship between price categories and customer satisfaction.

This analysis helps businesses understand how price influences customer perceptions and satisfaction levels. Furthermore, it provides valuable information for decision-making processes related to product development, marketing strategies, and customer experience enhancements. By understanding the impact of price on customer satisfaction through the analysis of average review scores, businesses can make informed decisions to optimize their offerings and improve customer satisfaction.

My Interpretation

Average Review Score:

The "cheap-mid" price category has the highest average review score, indicating that customers are generally satisfied with products in this price range while the "expensive" price category has the lowest average review score, suggesting that customers may have slightly lower satisfaction levels with products in this higher price range.

Order Counts:

The "cheap" price category has the highest number of orders, indicating that products in the lower price range are more popular and have a higher demand among customers while the "expensive" price category has the lowest number of orders, suggesting a more limited customer base for products in this higher price range.

These findings provide insights into the relationship between price, customer satisfaction, and customer demand. Customers appear to be generally satisfied with products in the mid-range price categories, while products in the lower price range have a higher demand. However, it's important to note that the differences in average review scores and order counts between price categories are relatively small.

In [11]:
```
# Merge orders with items by order_id
orders_items = pd.merge(orders, items, on='order_id')
```

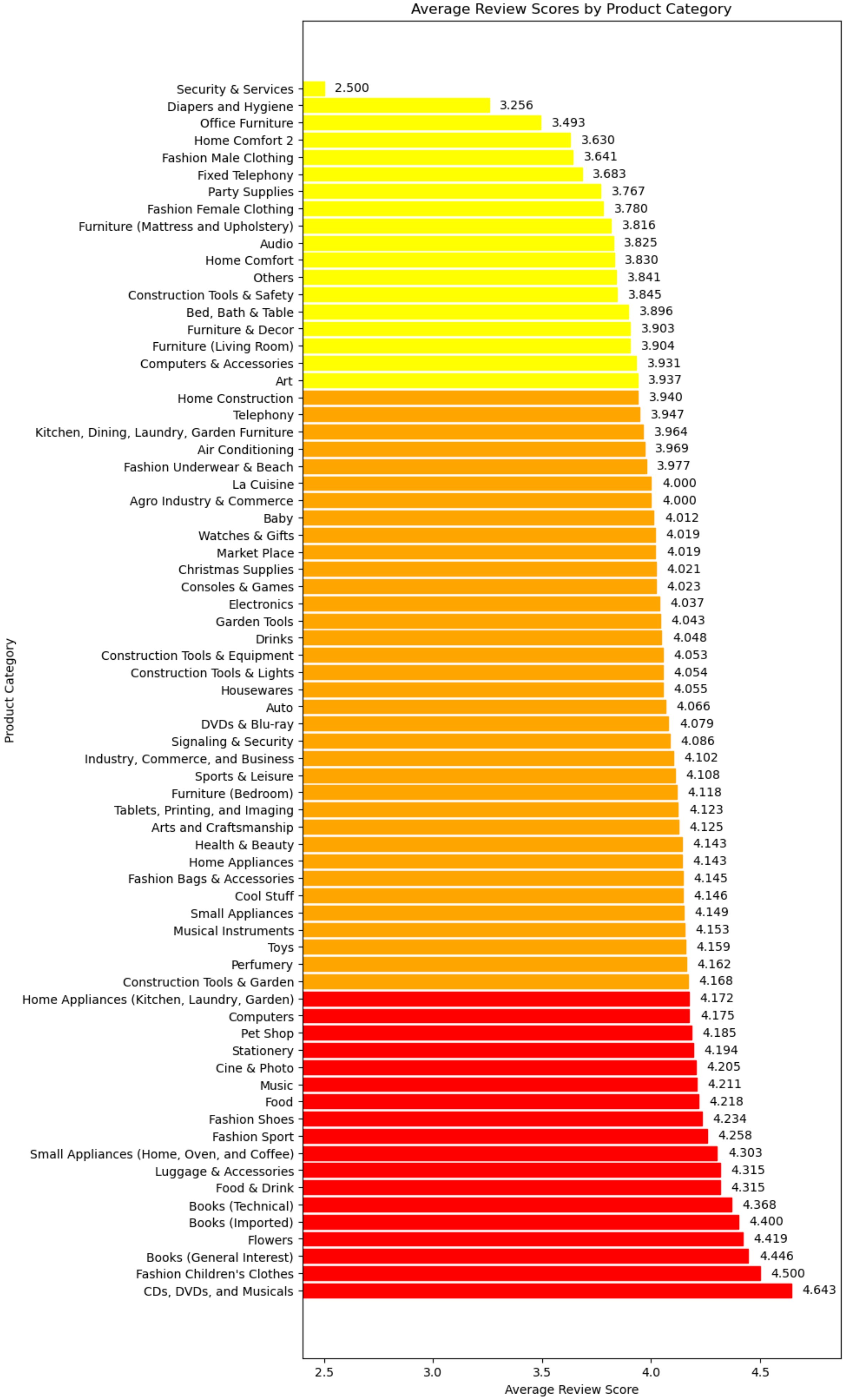
```
# Merge all datasets with products by product_id
merged_data = pd.merge(pd.merge(reviews_orders, orders_items, on='order_id')

# Merge with product_category_name_translation by product_category_name
merged_data = pd.merge(merged_data, translation, left_on='product_category_n

# Replace the values in the 'product_category_name_english' column with thei
merged_data['product_category_name_english'] = merged_data['product_category
    'sports_leisure': 'Sports & Leisure',
    'computers_accessories': 'Computers & Accessories',
    'garden_tools': 'Garden Tools',
    'bed_bath_table': 'Bed, Bath & Table',
    'toys': 'Toys',
    'home_confort': 'Home Comfort',
    'small_appliances': 'Small Appliances',
    'health_beauty': 'Health & Beauty',
    'pet_shop': 'Pet Shop',
    'cool_stuff': 'Cool Stuff',
    'electronics': 'Electronics',
    'baby': 'Baby',
    'luggage_accessories': 'Luggage & Accessories',
    'housewares': 'Housewares',
    'watches_gifts': 'Watches & Gifts',
    'auto': 'Auto',
    'telephony': 'Telephony',
    'fashion_bags_accessories': 'Fashion Bags & Accessories',
    'perfumery': 'Perfumery',
    'furniture_decor': 'Furniture & Decor',
    'home_appliances_2': 'Home Appliances',
    'food_drink': 'Food & Drink',
    'musical_instruments': 'Musical Instruments',
    'stationery': 'Stationery',
    'books_imported': 'Books (Imported)',
    'office_furniture': 'Office Furniture',
    'books_general_interest': 'Books (General Interest)',
    'construction_tools_construction': 'Construction Tools & Equipment',
    'books_technical': 'Books (Technical)',
    'construction_tools_safety': 'Construction Tools & Safety',
    'art': 'Art',
    'home_appliances': 'Home Appliances (Kitchen, Laundry, Garden)',
    'computers': 'Computers',
    'christmas_supplies': 'Christmas Supplies',
    'audio': 'Audio',
    'Others': 'Others',
    'industry_commerce_and_business': 'Industry, Commerce, and Business',
    'furniture_living_room': 'Furniture (Living Room)',
    'consoles_games': 'Consoles & Games',
    'market_place': 'Market Place',
    'drinks': 'Drinks',
    'kitchen_dining_laundry_garden_furniture': 'Kitchen, Dining, Laundry, Ga
    'music': 'Music',
    'furniture_bedroom': 'Furniture (Bedroom)',
    'la_cuisine': 'La Cuisine',
    'signaling_and_security': 'Signaling & Security',
    'home_construction': 'Home Construction',
    'food': 'Food',
    'small_appliances_home_oven_and_coffee': 'Small Appliances (Home, Oven,
    'air_conditioning': 'Air Conditioning',
    'cine_photo': 'Cine & Photo',
    'fashion_shoes': 'Fashion Shoes',
    'agro_industry_and_commerce': 'Agro Industry & Commerce',
    'furniture_mattress_and_upholstery': 'Furniture (Mattress and Upholstery
    'home_comfort_2': 'Home Comfort 2',
    'fashion_underwear_beach': 'Fashion Underwear & Beach',
```

```
    'construction_tools_lights': 'Construction Tools & Lights',
    'dvds_blu_ray': 'DVDs & Blu-ray',
    'costruction_tools_tools': 'Construction Tools & Garden',
    'fashion_male_clothing': 'Fashion Male Clothing',
    'fixed_telephony': 'Fixed Telephony',
    'costruction_tools_garden': 'Construction Tools & Garden',
    'fashion_female_clothing': 'Fashion Female Clothing',
    'fashion_sport': 'Fashion Sport',
    'nan': 'Not Available',
    'tablets_printing_image': 'Tablets, Printing, and Imaging',
    'cds_dvds_musicals': 'CDs, DVDs, and Musicals',
    'flowers': 'Flowers',
    'diapers_and_hygiene': 'Diapers and Hygiene',
    'party_supplies': 'Party Supplies',
    'fashion_childrens_clothes': "Fashion Children's Clothes",
    'arts_and_craftmanship': 'Arts and Craftsmanship',
    'security_and_services': 'Security & Services'
})


# Group by product category and calculate the average review score
category_review_scores = merged_data.groupby('product_category_name_english
category_review_scores.columns = ['Product Category', 'Average Review Score
category_review_scores = category_review_scores.sort_values(by='Average Revi
category_review_scores = category_review_scores.reset_index(drop=True)

# Group by product category and calculate the count of orders
category_order_counts = merged_data.groupby('product_category_name_english')
category_order_counts.columns = ['Product Category', 'Order Count']
category_order_counts = category_order_counts.sort_values(by='Order Count',
category_order_counts = category_order_counts.reset_index(drop=True)

product_categories = merged_data['product_category_name_english'].unique()
```

Visualization 3

In [12]:

```
# Create a horizontal bar chart for average review scores
plt.figure(figsize=(8, 20))
bars = plt.barh(category_review_scores['Product Category'], category_review_
plt.xlabel('Average Review Score')
plt.ylabel('Product Category')
plt.title('Average Review Scores by Product Category')
plt.xlim(2.4, 4.87)

for bar in bars:
    plt.text(bar.get_width() + 0.05, bar.get_y() + bar.get_height() / 2,
             f'{bar.get_width():.3f}', va='center')
min_value = min(category_review_scores['Average Review Score'])
q1_value = category_review_scores['Average Review Score'].quantile(0.25)
q3_value = category_review_scores['Average Review Score'].quantile(0.75)
max_value = max(category_review_scores['Average Review Score'])

for bar in bars:
    if bar.get_width() < q1_value:
        bar.set_color('yellow')
    elif bar.get_width() < q3_value:
        bar.set_color('orange')
    else:
        bar.set_color('red')

plt.show()
```


Average Review Scores by Product Category
Security & Services 2.500
Diapers and Hygiene 3.256
Office Furniture 3.493
Home Comfort 2 3.630
Fashion Male Clothing 3.641
Fixed Telephony 3.683
Party Supplies 3.767
Fashion Female Clothing 3.780
Furniture (Mattress and Upholstery) 3.816
Audio 3.825
Home Comfort 3.830
Others 3.841
Construction Tools & Safety 3.845
Bed, Bath & Table 3.896
Furniture & Decor 3.903
Furniture (Living Room) 3.904
Computers & Accessories 3.931
Art 3.937
Home Construction 3.940
Telephony 3.947
Kitchen, Dining, Laundry, Garden Furniture 3.964
Air Conditioning 3.969
Fashion Underwear & Beach 3.977
La Cuisine 4.000
Agro Industry & Commerce 4.000
Baby 4.012
Watches & Gifts 4.019
Market Place 4.019
Christmas Supplies 4.021
Consoles & Games 4.023
Electronics 4.037
Garden Tools 4.043
Drinks 4.048
Construction Tools & Equipment 4.053
Construction Tools & Lights 4.054
Housewares 4.055
Auto 4.066
DVDs & Blu-ray 4.079
Signaling & Security 4.086
Industry, Commerce, and Business 4.102
Sports & Leisure 4.108
Furniture (Bedroom) 4.118
Tablets, Printing, and Imaging 4.123
Arts and Craftsmanship 4.125
Health & Beauty 4.143
Home Appliances 4.143
Fashion Bags & Accessories 4.145
Cool Stuff 4.146
Small Appliances 4.149
Musical Instruments 4.153
Toys 4.159
Perfumery 4.162
Construction Tools & Garden 4.168
Home Appliances (Kitchen, Laundry, Garden) 4.172
Computers 4.175
Pet Shop 4.185
Stationery 4.194
Cine & Photo 4.205
Music 4.211
Food 4.218
Fashion Shoes 4.234
Fashion Sport 4.258
Small Appliances (Home, Oven, and Coffee) 4.303
Luggage & Accessories 4.315
Food & Drink 4.315
Books (Technical) 4.368
Books (Imported) 4.400
Flowers 4.419
Books (General Interest) 4.446
Fashion Children's Clothes 4.500
CDs, DVDs, and Musicals 4.643
Product Category
2.5
3.0
3.5
4.0
4.5
Average Review Score

.

. . .

.

Rationale #3

The product categories may consist of similar types of products or products with similar characteristics. If the products within a category have consistent quality or features, it can lead to more consistent review scores and lower variability. Customers may have similar expectations when purchasing products within a specific category. If the products consistently meet or exceed these expectations, it can result in more consistent review scores and lower variability. The range of possible review scores may be relatively narrow, leading to lower variability.

My Interpretation

High Average Review Scores

The top product categories with high average review scores include "CDs, DVDs and Musicals" (4.643), "Fashion Childrens Clothes" (4.500), and "Books (General Interest)" (4.446). These categories seem to have products that are well-received by customers, as indicated by the high average review scores.

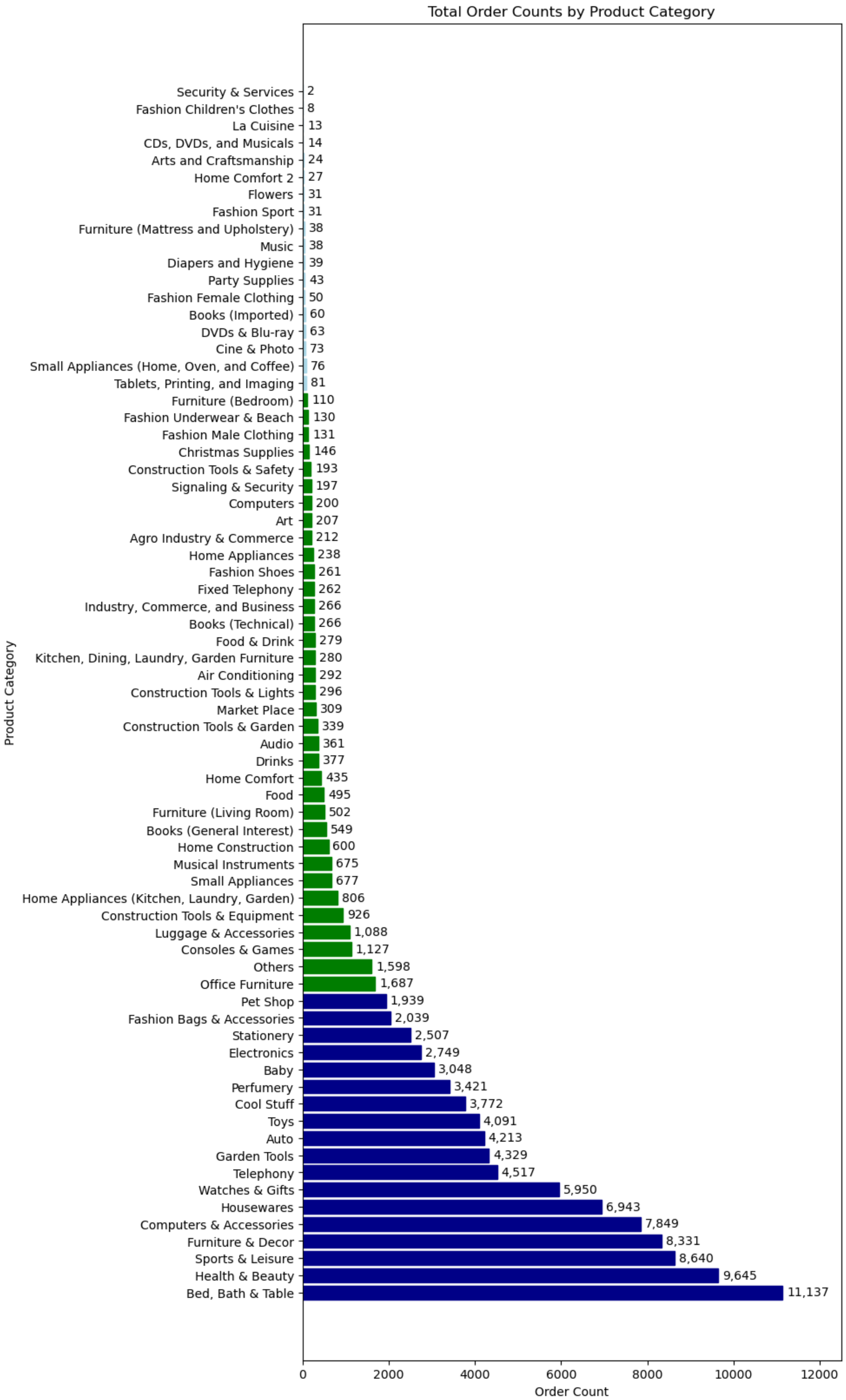
Low Average Review Scores

On the other hand, the product categories with relatively low average review scores include "Diapers & Hygiene" (3.256), "Office Furniture" (3.493), and "Security & Services" (2.500). These categories may have products that are not as well-received by customers, as indicated by the lower average review scores. Please notice that the range of possible review scores may be relatively narrow, leading to lower variability.

Visualization 4

In [13]:
```
# Create a horizontal bar chart for total order counts
plt.figure(figsize=(8, 20))
bars = plt.barh(category_order_counts['Product Category'], category_order_co
plt.xlabel('Order Count')
plt.ylabel('Product Category')
plt.title('Total Order Counts by Product Category')
plt.xlim(0, 12500)

for bar in bars:
    plt.text(bar.get_width() + 100, bar.get_y() + bar.get_height() / 2,
             f'{int(bar.get_width()):,}', va='center')
min_value = min(category_order_counts['Order Count'])
q1_value = category_order_counts['Order Count'].quantile(0.25)
q3_value = category_order_counts['Order Count'].quantile(0.75)
max_value = max(category_order_counts['Order Count'])
for bar in bars:
    if bar.get_width() < q1_value:
        bar.set_color('lightblue')
    elif bar.get_width() < q3_value:
        bar.set_color('green')
    else:
```

```
        bar.set_color('darkblue')
plt.show()
```



Rationale 4

In terms of market demand, higher order counts indicate a stronger market demand for products within those specific categories. This information is valuable for businesses as it allows them to identify popular product categories and allocate resources effectively. The order counts also serve as a reflection of customer preferences and buying behavior. Categories with higher order counts signify greater customer interest in purchasing products from those particular categories. These order counts can provide valuable insights to guide marketing and promotional strategies. By focusing on categories with lower order counts, businesses can develop targeted marketing campaigns to increase sales and attract a larger customer base.

My Interpretation

The top product categories with the highest order counts include "Bed, Bath & Tab;e" (11,137), "Health & Beauty" (9,645), and "Sports & Leisure" (8,640). These categories seem to have a strong market demand, indicating that customers are actively purchasing products from these categories.

The product categories with lower order counts include "Furniture (Mattress and Upholstery)" (38), "Music" (38), and "flowers" (31). These categories may have a relatively lower market demand compared to the top categories.

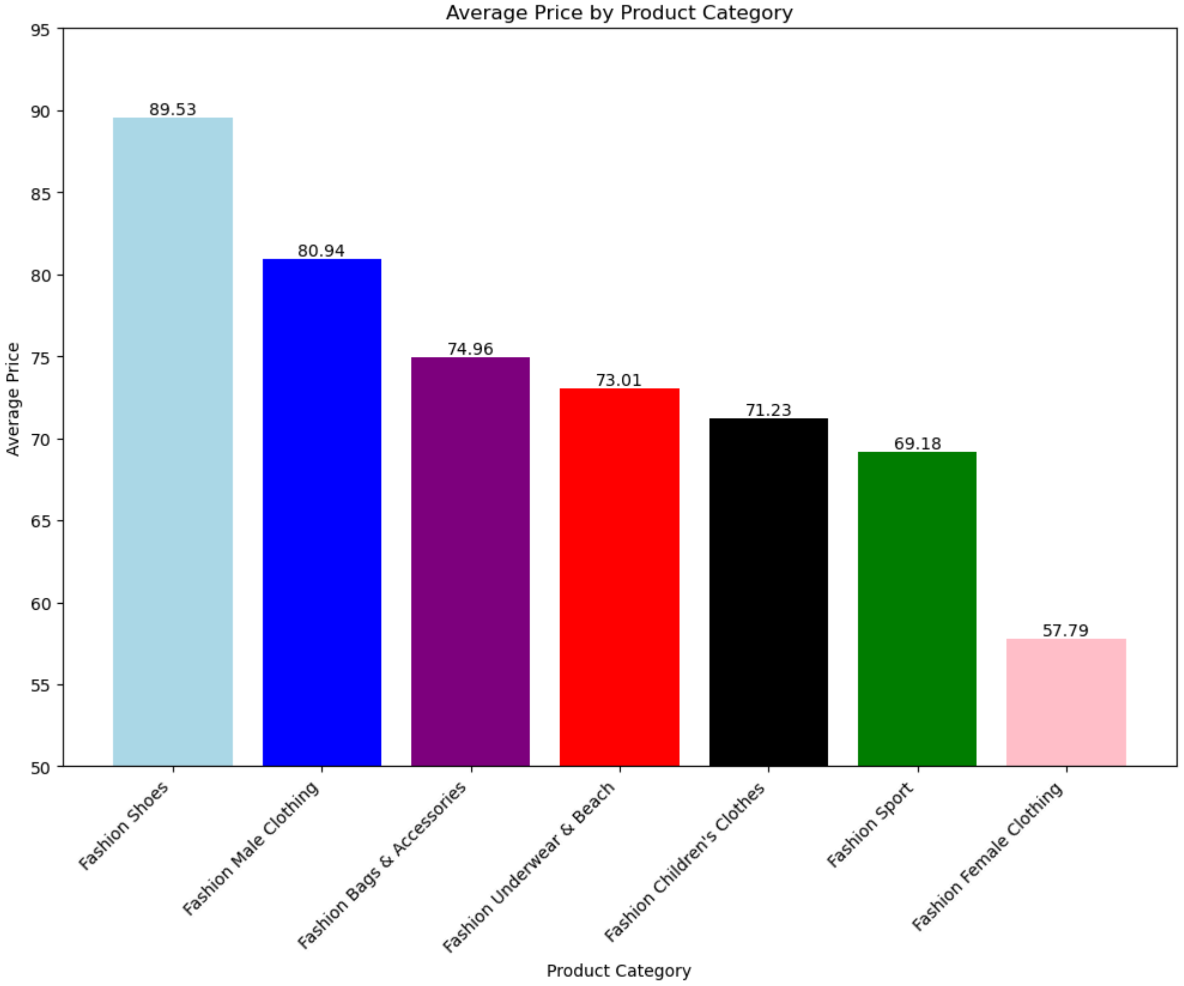
The order counts can provide valuable insights for businesses to identify popular product categories and allocate resources accordingly. Higher order counts indicate a greater demand for products within those categories, which can guide businesses in making informed decisions about inventory management and resource allocation.

In [14]:
```
# Exclude rows with missing values in 'product_category_name_english'
merged_data_clean = merged_data.dropna(subset=['product_category_name_englis

# Filter the cleaned merged_data to include only rows where 'product_categor
fashion_data = merged_data_clean[merged_data_clean['product_category_name_en

# Group by product category and calculate the average price and average revi
fashion_stats = fashion_data.groupby('product_category_name_english').agg({
fashion_stats.columns = ['Product Category', 'Average Price', 'Average Revie
fashion_stats = fashion_stats.sort_values(by='Average Review Score', ascendi
fashion_stats = fashion_stats.reset_index(drop=True)

# Count number of order IDs for each fashion category
fashion_order_counts = fashion_data.groupby('product_category_name_english')
fashion_order_counts.columns = ['Product Category', 'Number of Orders']

# Merge fashion_stats and fashion_order_counts
fashion_stats = fashion_stats.merge(fashion_order_counts, on='Product Catego

# Sort by average review score in descending order
fashion_stats = fashion_stats.sort_values(by='Average Review Score', ascendi

print(fashion_stats)
```

```
                 Product Category  Average Price  Average Review Score  \
0  Fashion Children's Clothes      71.231250              4.500000
1               Fashion Sport      69.177419              4.258065
2               Fashion Shoes      89.532835              4.233716
3  Fashion Bags & Accessories      74.956557              4.144679
4   Fashion Underwear & Beach      73.012692              3.976923
5     Fashion Female Clothing      57.788800              3.780000
6       Fashion Male Clothing      80.937557              3.641221

   Number of Orders
0                 8
1                31
2               261
3              2039
4               130
5                50
6               131
```

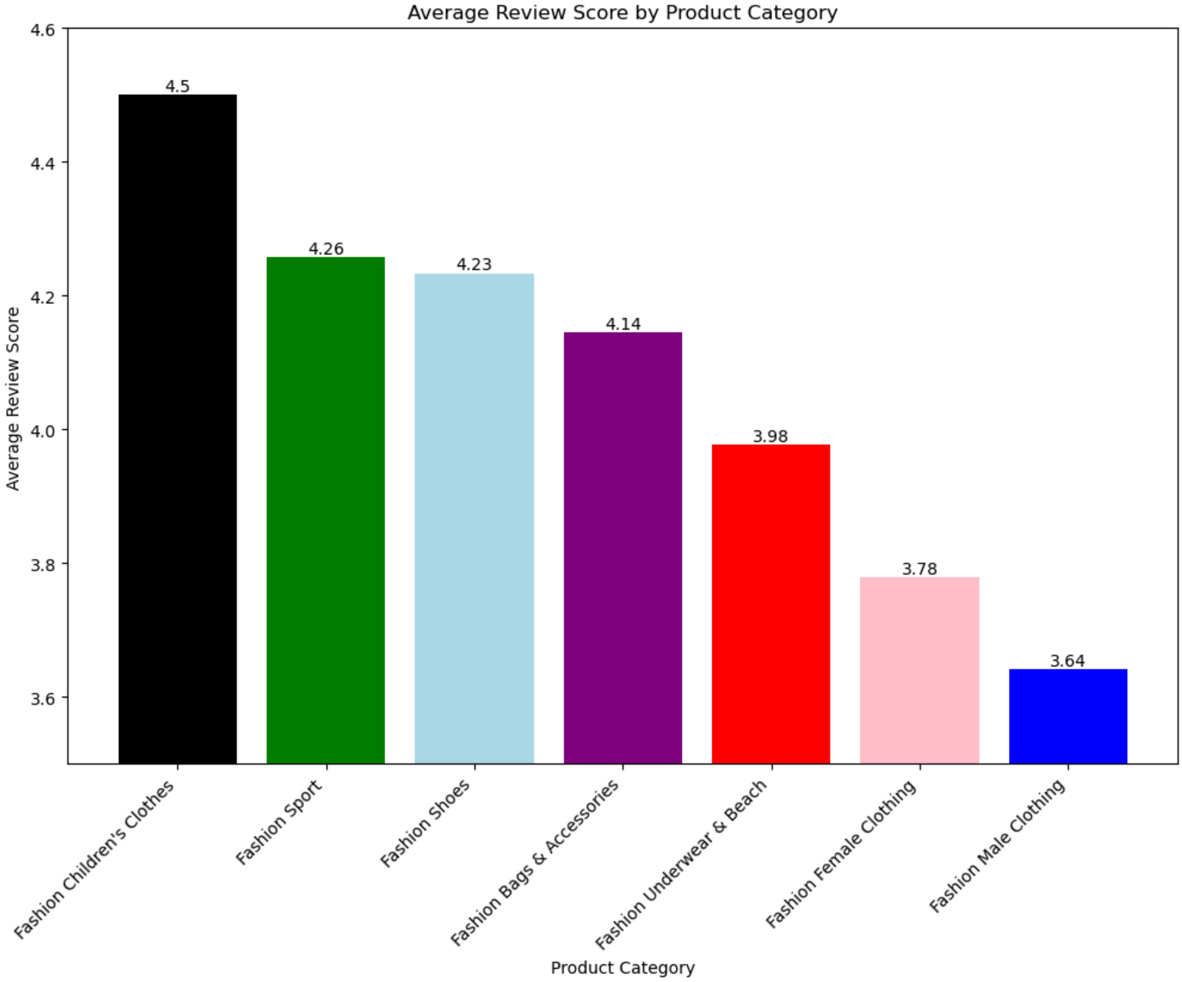
Visualization 5

In [15]:
```
# Sort the fashion_stats dataframe by 'Average Price' in descending order
fashion_stats_sorted = fashion_stats.sort_values('Average Price', ascending=

# Create a list of colors for each bar
colors = ['lightblue', 'blue', 'purple', 'red', 'black', 'green', 'pink']

# Create a bar graph for average price with different colors
plt.figure(figsize=(12, 8))
bars = plt.bar(fashion_stats_sorted['Product Category'], fashion_stats_sorte
plt.xlabel('Product Category')
plt.ylabel('Average Price')
plt.title('Average Price by Product Category')
plt.xticks(rotation=45, ha='right')

# Set the y-axis limit from 50 to 95
plt.ylim(50, 95)

# Add ranking on top of each bar
for bar in bars:
    yval = bar.get_height()
    plt.text(bar.get_x() + bar.get_width() / 2, yval, round(yval, 2), ha='ce

plt.show()
```



Visualization 6

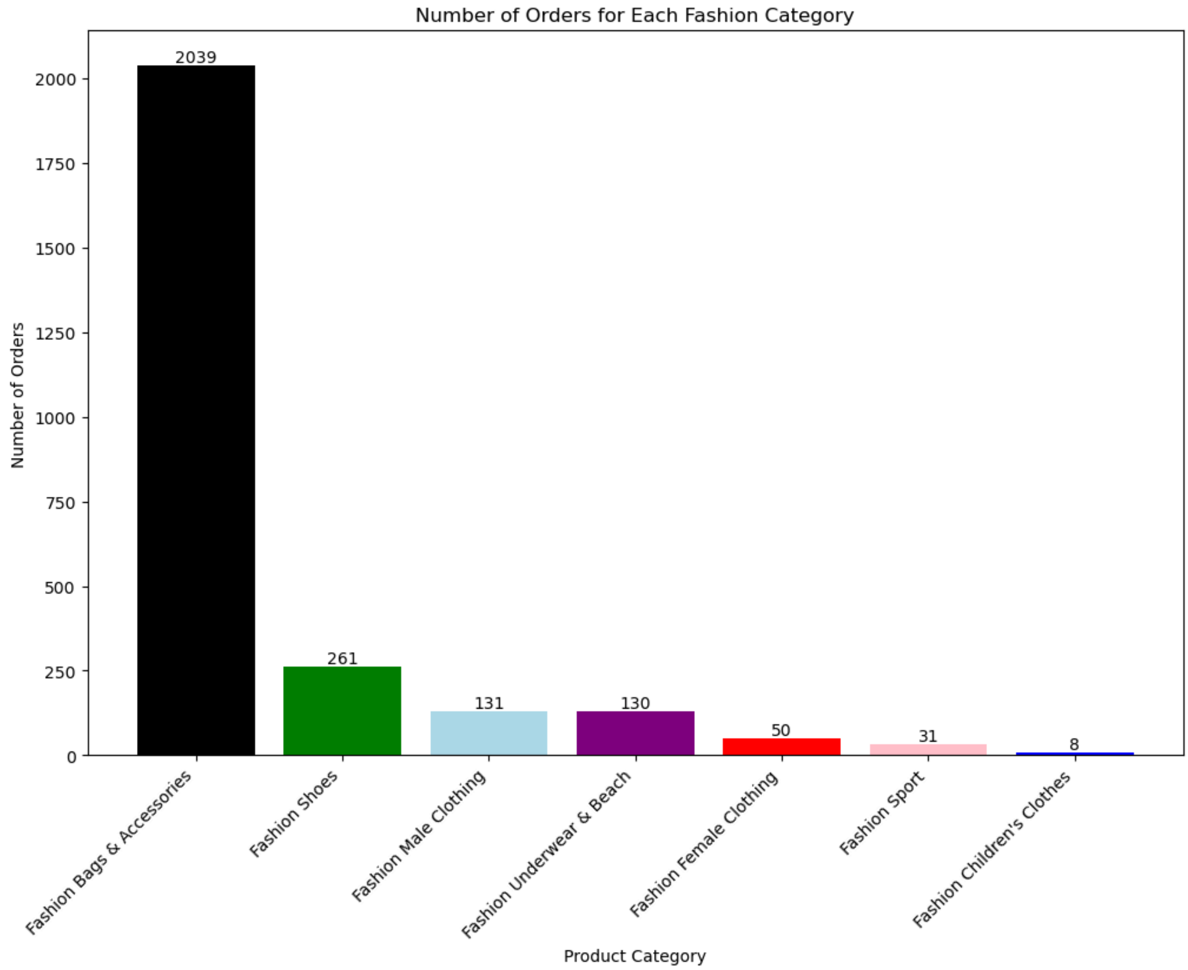
In [16]:
```
# Create a list of colors for each bar
colors = ['black', 'green', 'lightblue', 'purple', 'red', 'pink', 'blue']

# Create a bar graph for average review score with different colors
plt.figure(figsize=(12, 8))
bars = plt.bar(fashion_stats['Product Category'], fashion_stats['Average Rev
plt.xlabel('Product Category')
plt.ylabel('Average Review Score')
plt.title('Average Review Score by Product Category')
plt.xticks(rotation=45, ha='right')

# Set the y-axis limit from 3.5 to 4.6
plt.ylim(3.5, 4.6)

# Add ranking on top of each bar
for bar in bars:
    yval = bar.get_height()
    plt.text(bar.get_x() + bar.get_width() / 2, yval, round(yval, 2), ha='ce

plt.show()
```



Visualization 7

In [17]:

```
# Count number of order IDs for each fashion category
fashion_order_counts = fashion_data.groupby('product_category_name_english')
fashion_order_counts.columns = ['Product Category', 'Number of Orders']

# Sort by number of orders in descending order
fashion_order_counts = fashion_order_counts.sort_values(by='Number of Orders

# Create a list of colors for each bar
colors = ['black', 'green', 'lightblue', 'purple', 'red', 'pink', 'blue']

# Plot the number of orders with different colors for each bar
plt.figure(figsize=(12, 8))
bars = plt.bar(fashion_order_counts['Product Category'], fashion_order_coun
plt.xlabel('Product Category')
plt.ylabel('Number of Orders')
plt.title('Number of Orders for Each Fashion Category')

# Add labels on top of each bar
for bar in bars:
    yval = bar.get_height()
    plt.text(bar.get_x() + bar.get_width() / 2, yval, str(int(yval)), ha='ce

plt.xticks(rotation=45, ha='right')
plt.show()
```



Rationale 5, 6 & 7

Average Price and Average Review Score

First, drop rows from the dataset that have missing values in the 'product_category_name_english' column. This step ensures that only complete and valid data is used for further analysis. Then, filter the cleaned dataset to include only rows where the 'product_category_name_english' column contains the word "fashion". This step narrows down the dataset to focus specifically on fashion-related products.

Finally, calculate the average price, average review score and total number of orders for each category. This step provides summary statistics of fashion products in different categories.

My Interpretation

Fashion Children Clothes: This category has an average price of 71.23 and an average review score of 4.5 out of 5. It suggests that children's clothing in the fashion industry is priced moderately and generally receives positive reviews. Fashion Sport: The average price for fashion sport products is 69.18, and the average review score is 4.26. This indicates that sportswear in the fashion industry is relatively affordable and tends to have high customer satisfaction.

Fashion Sports: Fashion shoes have a higher average price of 89.53, with an average review score of 4.23. This suggests that fashion shoes are priced higher compared to other categories, but they still receive favorable reviews from customers.

Fashion Shoes: The average price for fashion bags and accessories is 74.96, and the average review score is 4.14. It implies that bags and accessories in the fashion industry are moderately priced and generally well-received by customers.

Fashion Underwear & Beach: This category has an average price of 73.01 and an average review score of 3.98. It indicates that underwear and beachwear in the fashion industry are priced reasonably, but the average review score is slightly lower compared to other categories.

Fashion Male Clothing: Fashion male clothing has the highest average price among the categories, with 80.94, but the lowest average review score of 3.64. This suggests that male clothing in the fashion industry tends to be priced higher, but it may have room for improvement in terms of customer satisfaction.

Fashion Bag and Accessories is a popular category with a mid-average price of $74.96 and an average review score of 4.14. However, what sets it apart is the significantly high number of orders it receives. With a total of 2,039 orders, it surpasses the second most ordered item, fashion shoes, which has only 261 orders. This finding highlights the strong demand for Fashion Bag and Accessories among customers. Despite having a higher average price compared to other fashion categories, it continues to attract a large number of orders. This suggests that customers are willing to invest in these products, indicating their popularity and appeal in the market.

Additional data/dataset I think the company should collect and why?

In the review dataset, it has been observed that a significant number of reviews do not include the review title. This omission presents a potential opportunity to further examine the variables within the dataset in order to gain a more comprehensive understanding of the customers' perspectives on the products. Relying solely on the review score may not provide sufficient depth of insight. The comment should also be English, instead of Spanish.

To enhance the analysis, additional variables such as the manufacturer or brand of the product and the country of production can be incorporated. This would enable a comparative analysis to determine which country produces the best products. Furthermore, it would be valuable to conduct a comprehensive analysis of clothing items from renowned brands such as Nike, Adidas, UNIQLO, H&M, Louis Vuitton, Versace, Dolce & Gabbana and others. This approach would facilitate a more extensive and insightful examination of the data.