This analysis of the Olist Brazilian Ecommerce Dataset and report is completed by Aloysius Ang for the ST Engineering Technical Assessment.

Imported Libraries

```
import pandas as  pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
import os
import nltk
from nltk.sentiment import SentimentIntensityAnalyzer
from nltk.corpus import stopwords
from wordcloud import WordCloud
```

Importing the Data

```
# Dictionary to store DataFrames
dataframes = {}

for file in os.listdir('.'):
    if file.endswith('.csv'):
        var_name = os.path.splitext(file)[0]
        var_name = var_name.replace('olist_', '').replace('_dataset', '')
        dataframes[var_name] = pd.read_csv(file)
        print(f"Loaded {file} into DataFrame: {var_name}")

# Access DataFrames
customers = dataframes['customers']
```

First, I imported the datasets using a for loop and stored them in a dictionary. This is to make data analysis easier. I also imported each dataset manually as seen in the code below.

```
# Importing each dataset manually so that vs code will not give an error when
using the variable name
customers= pd.read_csv('olist_customers_dataset.csv')
geolocation= pd.read_csv('olist_geolocation_dataset.csv')
order_items= pd.read_csv('olist_order_items_dataset.csv')
order_payments= pd.read_csv('olist_order_payments_dataset.csv')
```

```
order_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')
product_category_name_translation=
pd.read_csv('product_category_name_translation.csv')
```

Data Understanding

```
for key in dataframes:
    print("Data:",key)
    display(dataframes[key].head())
```

The code above is mainly to display the columns each table has. This will make it easier to know which tables to merge later on for analysis.

```
for key in dataframes:
    print("\n Data Frame:", key)
    dataframes[key].info()
```

By using the above code, I found that there are null values in some columns of some datasets. However, I did not touch them first as I know I will be joining the tables together later and tampering with the data now may give an inaccurate analysis later.

Analysis for Customer Retention

The main purpose of this analysis is to improve customer retention. To accomplish this, I first separated the customers who purchased items only once and customers who bought items more than once, which I classify as repeating customers. I also identified the customers who are inactive (6 months of inactivity after their latest order). Then, I analysed their behaviours and identified factors which would make them stay.

Identifying Repeat Customers

```
full_customers_orders = customers.merge(orders,  how='left', on='customer_id')
```

First, I merged the Customers and Orders datasets together. I used a left join to ensure all the customer records are kept, in case there are some customers who did not purchase anything before.

```
print("Number of duplicated rows:",
full_customers_orders.duplicated(subset=['order_id',
'customer_unique_id']).sum())
null_counts = full_customers_orders.isnull().sum()
print(null_counts)
```

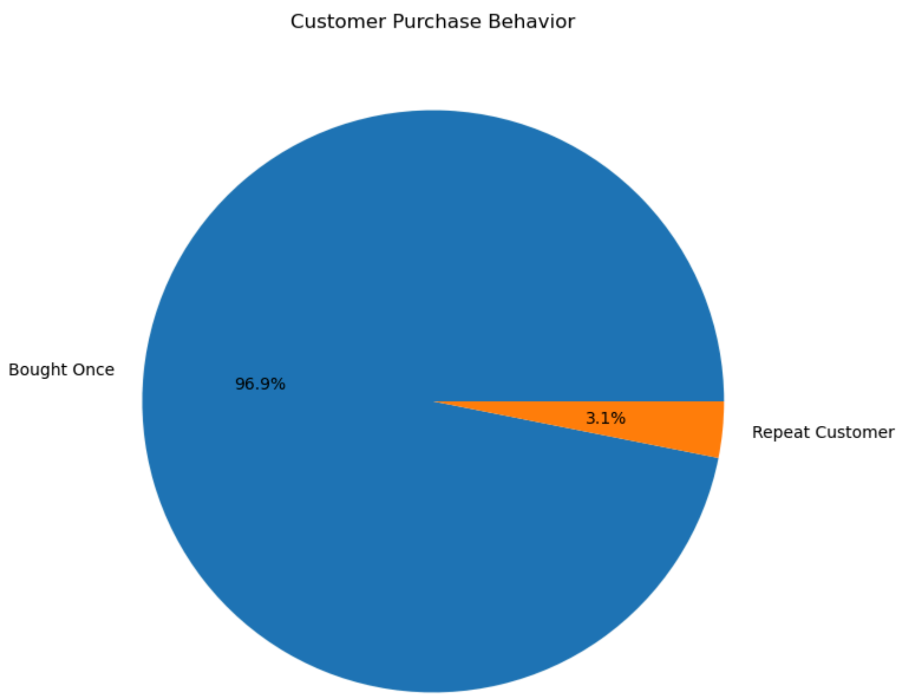
Next, I checked for duplicates and null values. There were no duplicates and the null values were only in the ‘order_approved_at’, ‘order_delivered_carrier_date’ and ‘order_delivered_customer_date’ columns, which may be irrelevant to what I wanted to do for now which is to identify repeat customers, thus I left them alone for now. The lack of null values also show that there are no customers who did not purchase anything. Every customer purchased something before.

```
# Group by 'customer_unique_id' to count the number of orders for each customer
number_customer_orders =
full_customers_orders.groupby('customer_unique_id')['order_id'].count().reset_ind
ex()

# Rename the column for clarity
number_customer_orders.rename(columns={'order_id': 'order_count'}, inplace=True)
# Define customer categories
def classify_customer(order_count):
    if order_count == 0:
        return 'Did Not Buy'
    elif order_count == 1:
        return 'Bought Once'
    else:
        return 'Repeat Customer'

# Apply the classification function
number_customer_orders['customer_category'] =
number_customer_orders['order_count'].apply(classify_customer)

# Count customers in each category
customer_distribution =
number_customer_orders['customer_category'].value_counts()
customer_distribution
```

I then find the number of times each customer ordered something and use that to find the repeated customers.

Output:

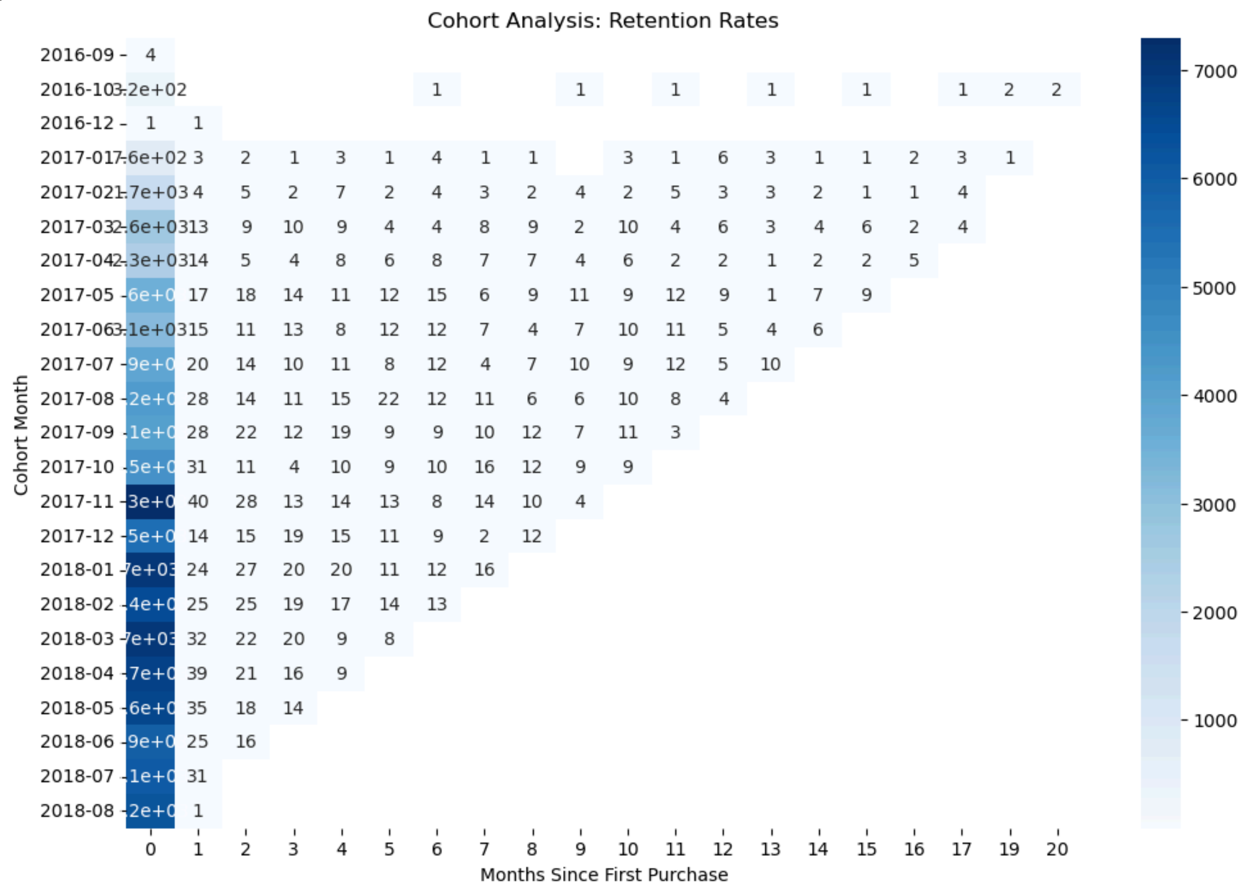
```
customer_category
Bought Once        93099
Repeat Customer     2997
```



Plotting a cohort analysis chart

```
# Ensure datetime columns are in datetime format
customers_orders_cleaned['order_purchase_timestamp'] =
pd.to_datetime(customers_orders_cleaned['order_purchase_timestamp'])

# Create a cohort_month for each customer based on their first purchase
customers_orders_cleaned['cohort_month'] =
customers_orders_cleaned.groupby('customer_unique_id')['order_purchase_timestamp'
].transform('min').dt.to_period('M')

# Create order_month to track each order's purchase month
customers_orders_cleaned['order_month'] =
customers_orders_cleaned['order_purchase_timestamp'].dt.to_period('M')
# Calculate the difference in months between the order and the cohort
customers_orders_cleaned['cohort_index'] =
(customers_orders_cleaned['order_month'] -
customers_orders_cleaned['cohort_month']).apply(lambda x: x.n)
# Group data by cohort_month and cohort_index
```

```
cohort_data = customers_orders_cleaned.groupby(['cohort_month',
'cohort_index']).agg(
    unique_customers=('customer_unique_id', 'nunique')
).reset_index()

# Pivot the data for heatmap
cohort_pivot = cohort_data.pivot(index='cohort_month', columns='cohort_index',
values='unique_customers')
# Divide each row by the first month's unique customers
cohort_size = cohort_pivot.iloc[:, 0]
retention_rates = cohort_pivot.divide(cohort_size, axis=0)

plt.figure(figsize=(12, 8))
sns.heatmap(cohort_pivot, annot=True, cmap="Blues")
plt.title('Cohort Analysis: Retention Rates')
plt.ylabel('Cohort Month')
plt.xlabel('Months Since First Purchase')
plt.show()
```



We can see that over time, the number of customers is falling. This could be due to the fact that the store was closing as well.

Analysis for repeat customers

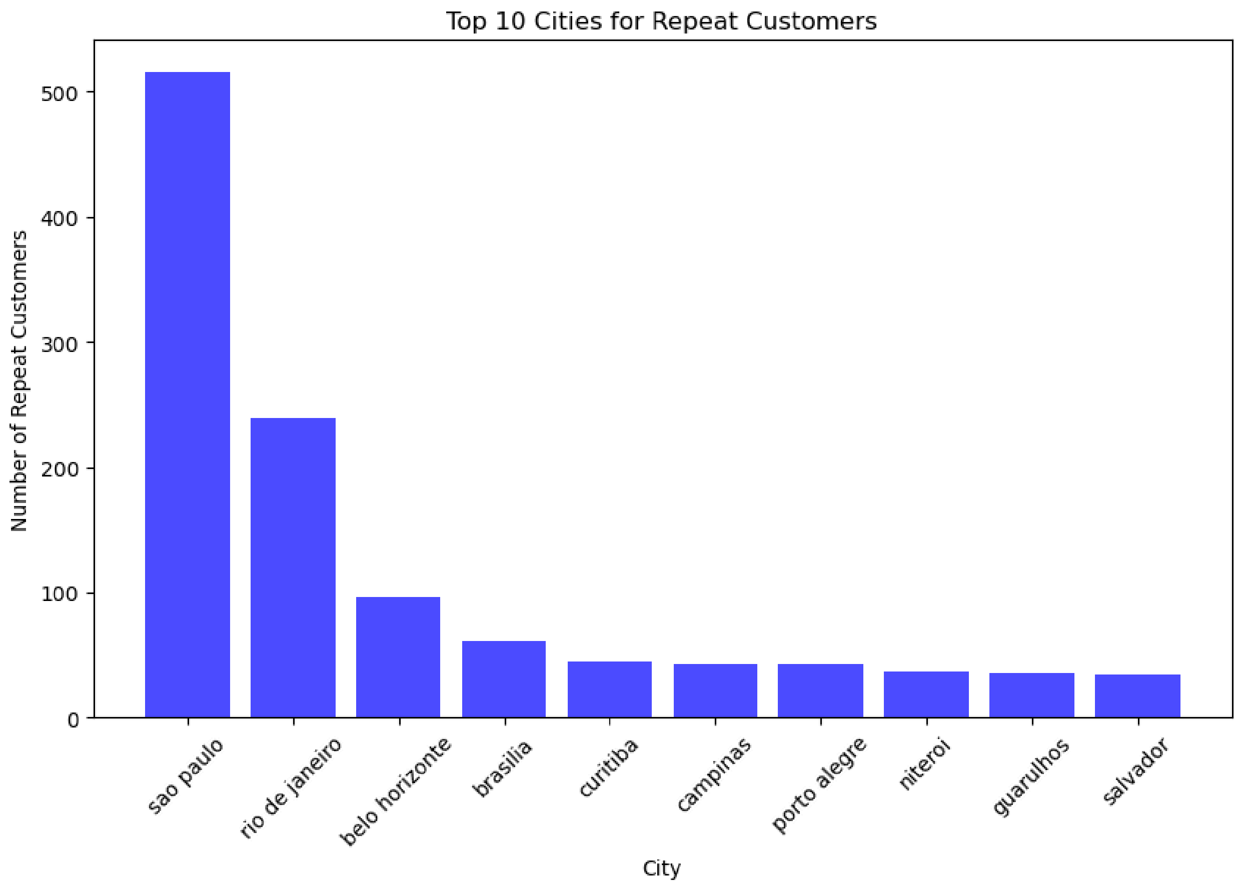
Getting the dataset for repeat customers

```
# Identify Repeat
customer_purchase_counts = customers_orders.groupby('customer_unique_id').size()
repeat_customers_ids = customer_purchase_counts[customer_purchase_counts >
1].index

# Split the Data
repeat_customers =
customers_orders[customers_orders['customer_unique_id'].isin(repeat_customers_ids
)]
repeat_unique_customers =
repeat_customers.drop_duplicates(subset='customer_unique_id') # Getting the
unique customers ID because they are duplicated
```

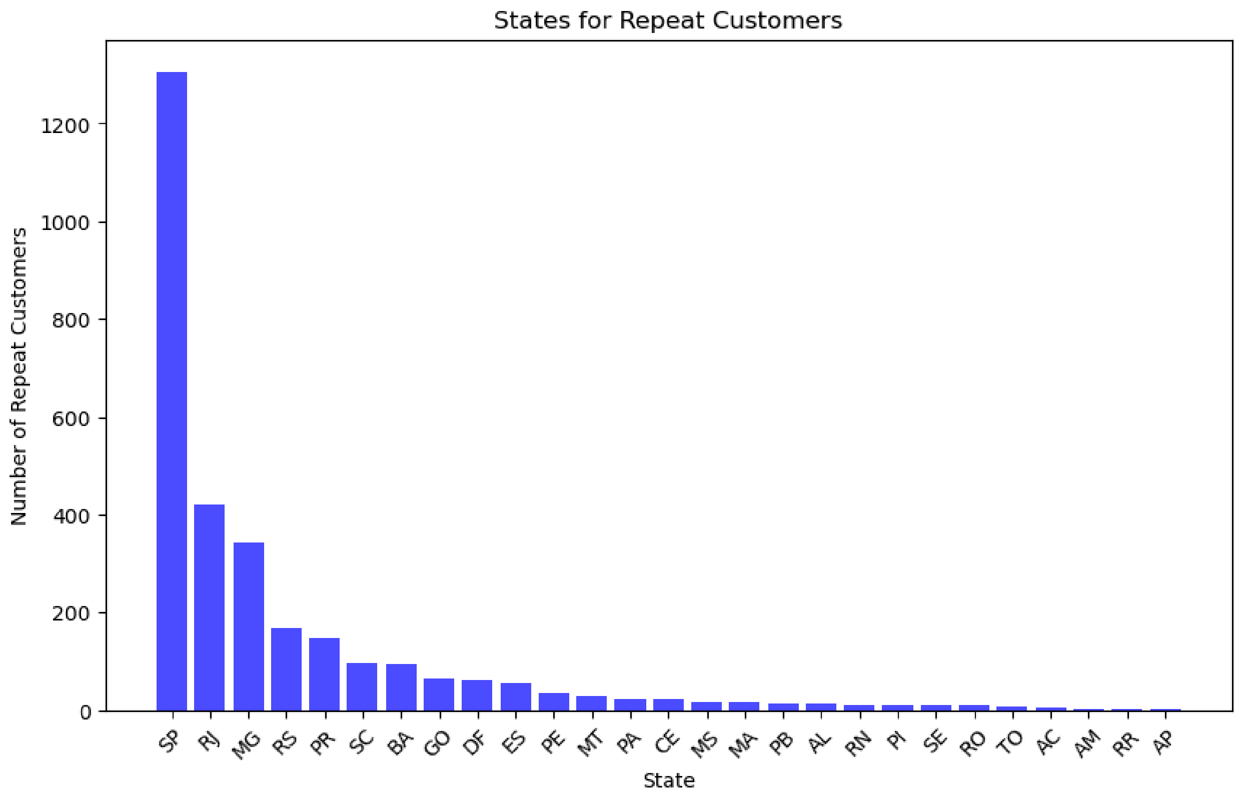
The above code is to get the dataset for repeat customers. Repeat_customers dataset contains all the orders made by the customers. In order to find out where most of them live, I first have to find the unique customer IDs and then find where they live. This is stored in the repeat_unique_customers dataset.

Finding the location of where the repeat customers stay

```
# Repeat Customers City
plt.figure(figsize=(10, 6))
plt.bar(repeat_city_counts.index, repeat_city_counts.values, color='blue',
alpha=0.7)
plt.title('Top 10 Cities for Repeat Customers')
plt.xlabel('City')
plt.ylabel('Number of Repeat Customers')
plt.xticks(rotation=45)
plt.show()
```



```
# Repeat Customers State
plt.figure(figsize=(10, 6))
plt.bar(repeat_state_counts.index, repeat_state_counts.values, color='blue',
alpha=0.7)
plt.title('States for Repeat Customers')
plt.xlabel('State')
plt.ylabel('Number of Repeat Customers')
plt.xticks(rotation=45)
plt.show()
```



From the graphs, we can tell which cities most of the repeat customers are from. Hence, the company will know where to allocate their resources to increase the chances of customers buying again.

Understanding the reviews from the repeat customers

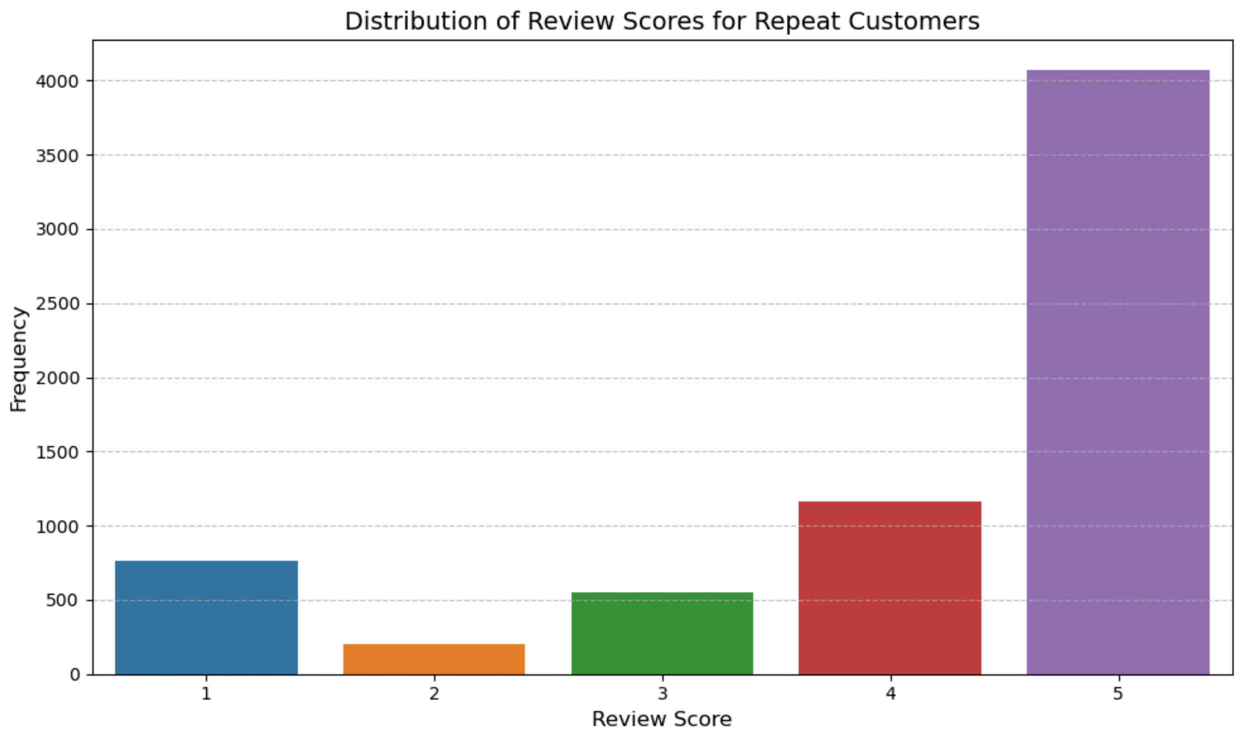
```
repeat_customers_reviews= repeat_customers.merge(order_reviews,  how='inner',
on='order_id')
```

I merged the repeat_customers table with order_reviews table.

Distribution of review scores

```
# Plot the distribution of review scores
plt.figure(figsize=(10, 6))
sns.countplot(data=repeat_customers_reviews, x='review_score',
order=sorted(repeat_customers_reviews['review_score'].unique()))
plt.title('Distribution of Review Scores', fontsize=14)
plt.xlabel('Review Score', fontsize=12)
plt.ylabel('Frequency', fontsize=12)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()
```

Next I plotted the distribution of review scores. I made sure that the review labels are in order in the plot.

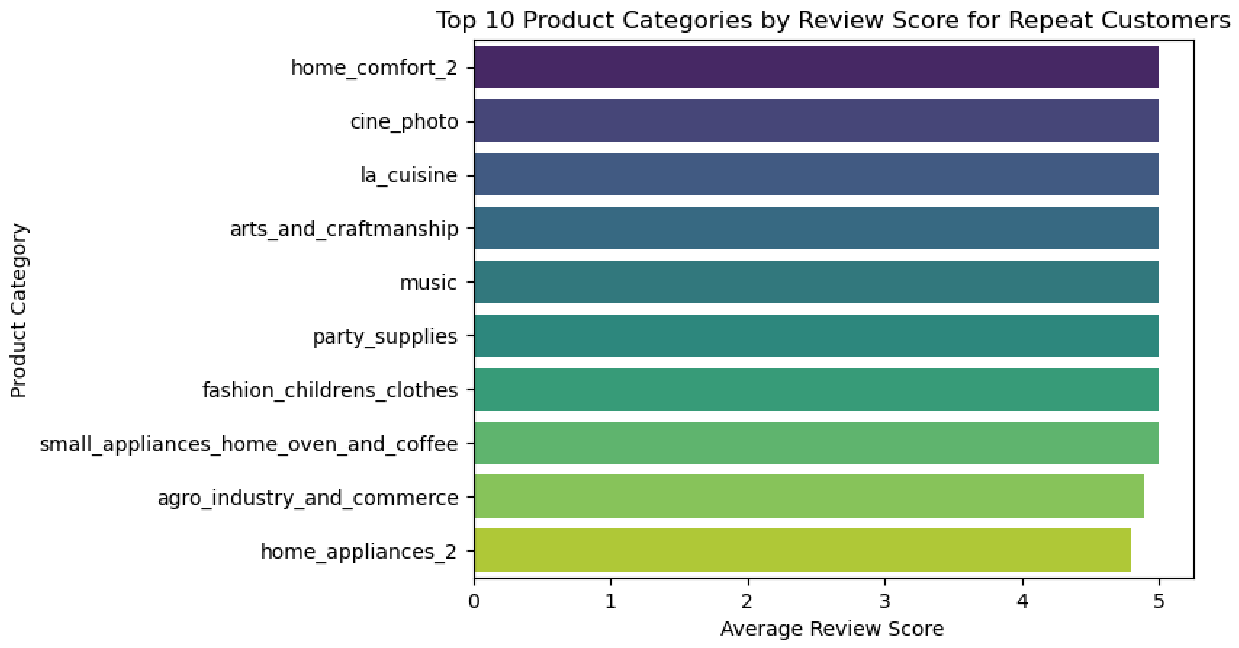


Finding top and bottom categories by review score

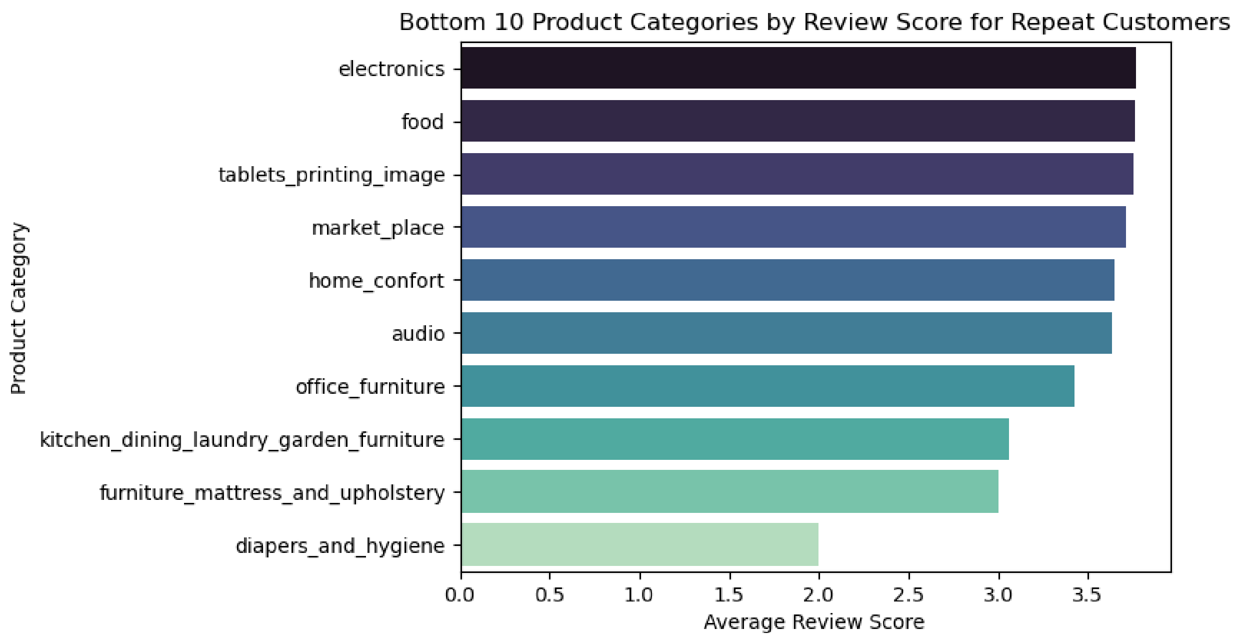
```
repeat_customers_reviews_orderItems = repeat_customers_reviews.merge(order_items,
how='inner', on='order_id')
repeat_customers_reviews_orderItems_products=
repeat_customers_reviews_orderItems.merge(products_translated,  how='inner',
on='product_id')
repeat_customers_reviews_orderItems_products
```

Merging datasets to get the required columns

```
# Finding the product category with the highest review score
# Group by product_category_name and calculate the average review score for each
product category
average_reviews_per_category = (

repeat_customers_reviews_orderItems_products.groupby('product_category_name_engli
sh')['review_score']
    .mean()
    .sort_values(ascending=False)
)

# Convert the result to a DataFrame
average_reviews_per_category_df = average_reviews_per_category.reset_index()
average_reviews_per_category_df.columns = ['product_category_name_english',
'average_review_score']

# Getting the top categories with the highest reviews
top_categories_by_reviews = average_reviews_per_category_df.head(10)

# Plot
sns.barplot(x='average_review_score', y='product_category_name_english',
data=top_categories_by_reviews, palette='viridis')
plt.title('Top 10 Product Categories by Review Score')
plt.xlabel('Average Review Score')
plt.ylabel('Product Category')
plt.show()
```



```
# Getting the bottom categories
bottom_categories_by_reviews = average_reviews_per_category_df.tail(10)

# Plot
sns.barplot(
    x='average_review_score',
    y='product_category_name_english',
    data=bottom_categories_by_reviews,
    palette='mako'
)
plt.title('Bottom 10 Product Categories by Review Score')
plt.xlabel('Average Review Score')
plt.ylabel('Product Category')
plt.show()
```

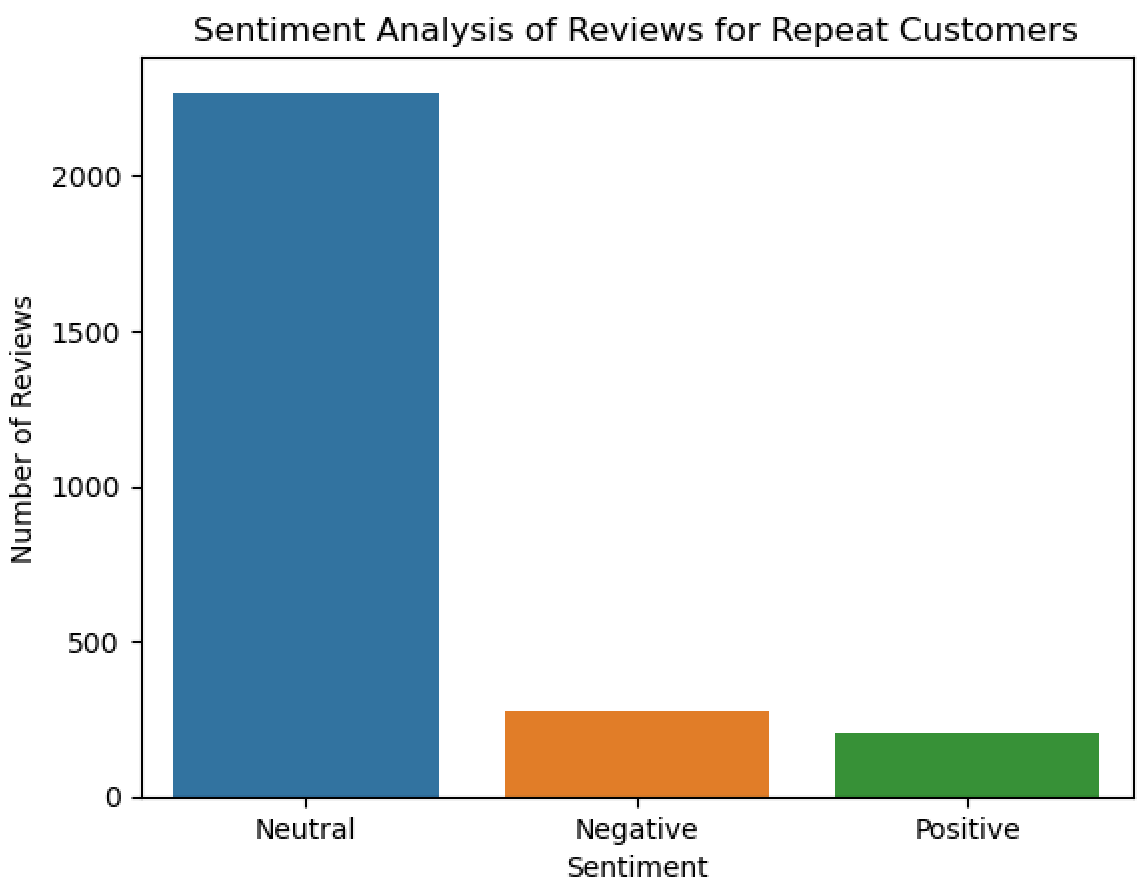


Conducting sentiment analysis

```
# Removing null values in the message column
repeat_customers_reviews_no_missing_review_content =
repeat_customers_reviews.dropna(subset=['review_comment_message'])
# Download VADER lexicon
nltk.download('vader_lexicon')

# Initialize VADER Sentiment Analyzer
sia = SentimentIntensityAnalyzer()

# Analyze sentiment
def analyze_sentiment(text):
    sentiment = sia.polarity_scores(text)
    return sentiment["compound"], sentiment["pos"], sentiment["neu"],
sentiment["neg"]

# Apply the function and store results
repeat_customers_reviews_no_missing_review_content[["sentiment_score",
"positive", "neutral", "negative"]] =
repeat_customers_reviews_no_missing_review_content["review_comment_message"].appl
y(
    lambda x: pd.Series(analyze_sentiment(x))
)

# Classify sentiment based on sentiment score
def classify_sentiment(score):
    if score > 0.05:
        return "Positive"
    elif score < -0.05:
        return "Negative"
    else:
        return "Neutral"

repeat_customers_reviews_no_missing_review_content["sentiment_category"] =
repeat_customers_reviews_no_missing_review_content["sentiment_score"].apply(class
ify_sentiment)
```

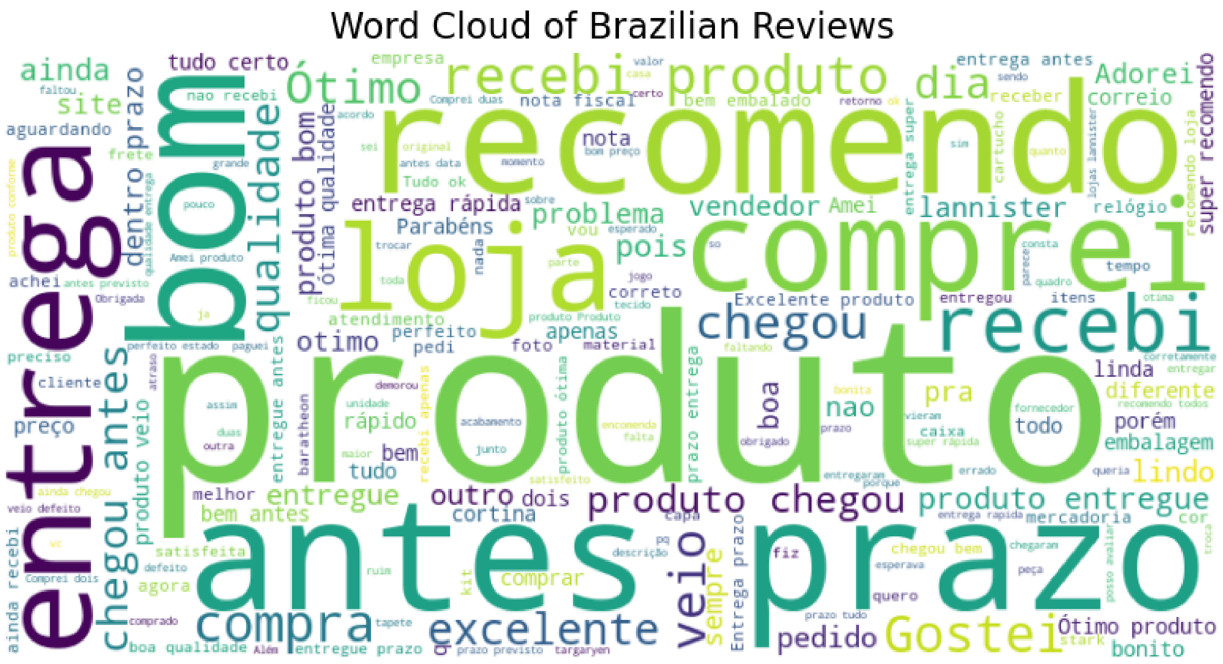
I first removed the null review messages. Then I calculated the sentiment scores and classified the reviews based on sentiment score.

```
# Number of each sentiment category
sentiment_counts =
repeat_customers_reviews_no_missing_review_content["sentiment_category"].value_co
unts()

# Plot
sns.barplot(x=sentiment_counts.index, y=sentiment_counts.values)
plt.title("Sentiment Analysis of Reviews")
plt.xlabel("Sentiment")
plt.ylabel("Number of Reviews")
plt.show()
```



Wordcloud

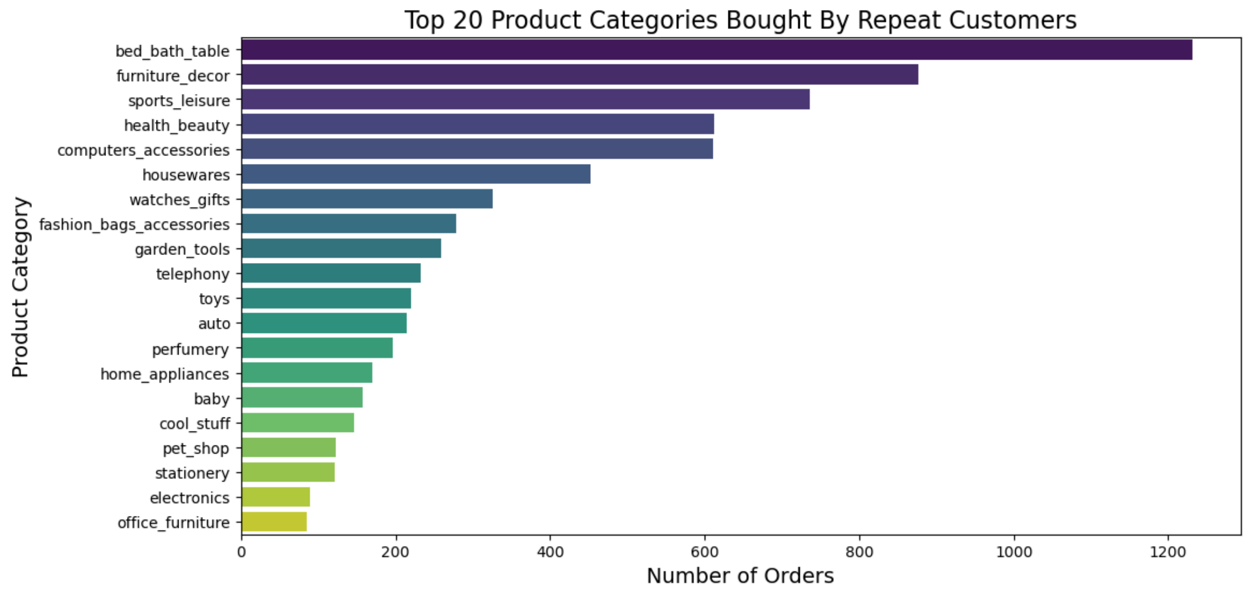
```
# Download Portuguese stopwords
nltk.download("stopwords")
stop_words = set(stopwords.words("portuguese"))

# Combining all the reviews together
all_reviews = "
".join(repeat_customers_reviews_no_missing_review_content["review_comment_message
"])

# Removing stopwords
filtered_words = " ".join([word for word in all_reviews.split() if word.lower()
not in stop_words])

# Word cloud
wordcloud = WordCloud(
    width=800,
    height=400,
    background_color="white",
    colormap="viridis",
    stopwords=stop_words
).generate(filtered_words)

# Display the word cloud
plt.figure(figsize=(10, 5))
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis("off")
plt.title("Word Cloud of Brazilian Reviews", fontsize=16)
plt.show()
```

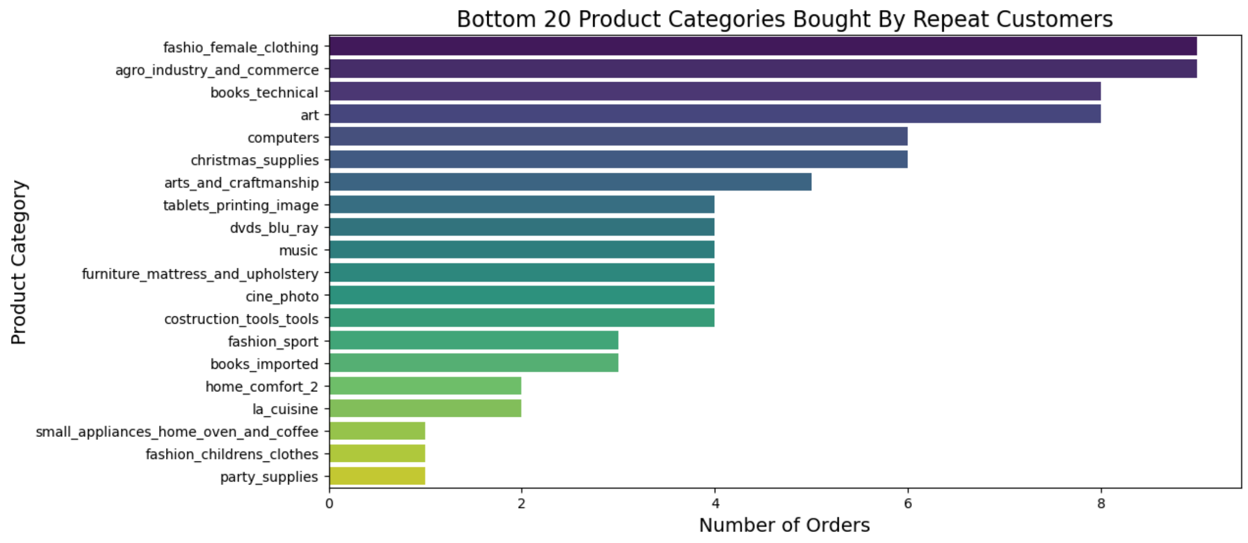


Insights from review analysis of repeated customers

Most of the review scores are positive. The store is better able to understand which product categories repeating customers like and dislike. Interestingly, most of the bottom 10 categories performed decently. All except 1 have an average review score of more than or equal to 3, which is average, not bad. This means that customers are generally satisfied with their purchase.
The sentiment analysis shows that most of the reviews are neutral. However, there are more negative reviews than positive ones, showing that customers are more likely to comment when receiving a bad product.

Finding the most and least bought product category by repeat customers



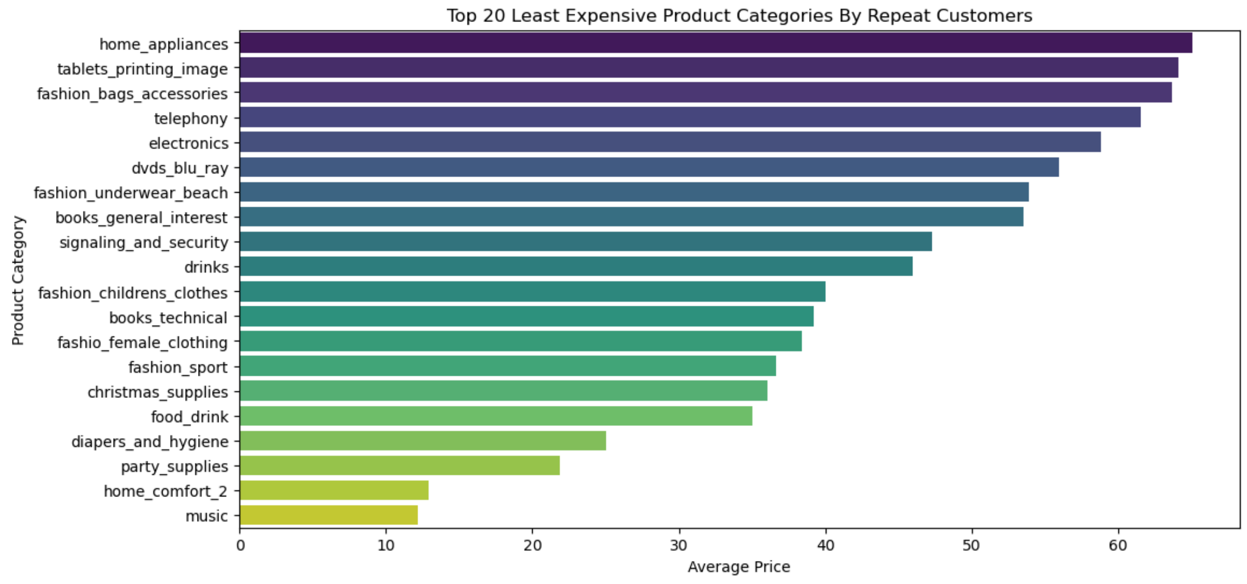


An interesting observation to take note is that the product categories are not in the same order as when they are ordered by review scores. Since the number of orders are directly related to sales, it may be wiser for the company to focus more on these product categories instead in order to boost customer retention.

Finding the least and most expensive product category bought by repeat customers

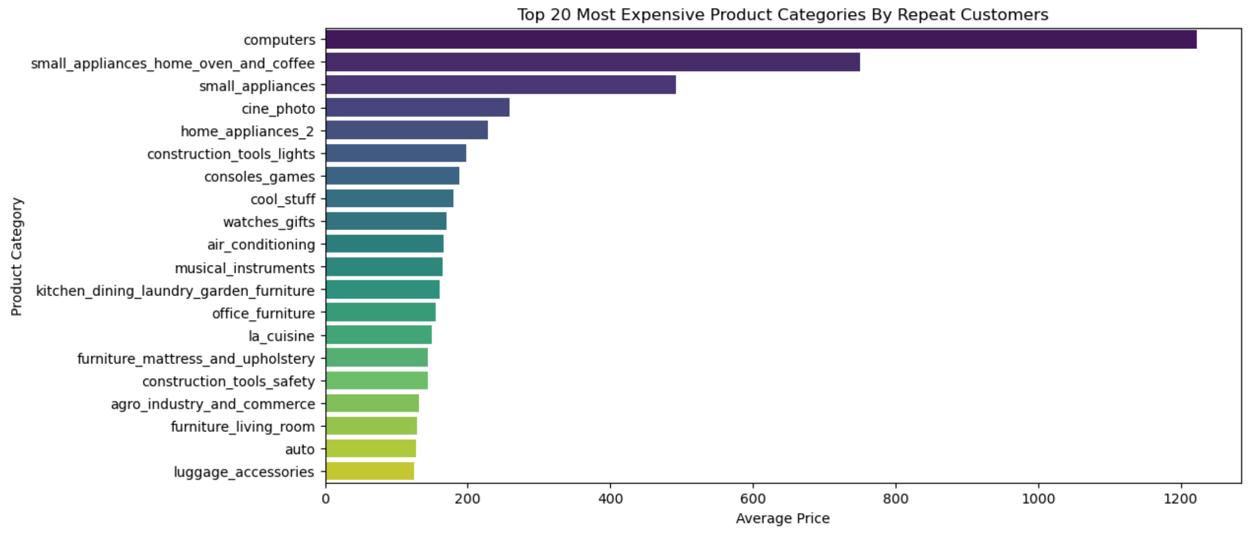
```
# Calculate the average price per category
avg_price_per_category_bottom = (

repeat_customers_reviews_orderItems_products.groupby("product_category_name")["pr
ice"]
```

```
    .mean()
    .sort_values(ascending=False)
    .tail(20)
)

# Plotting the bottom 20 most expensive categories
plt.figure(figsize=(12, 6))
sns.barplot(
    x=avg_price_per_category_bottom.values,
    y=avg_price_per_category_bottom.index,
    palette="viridis",
)
plt.title("Top 20 Least Expensive Product Categories")
plt.xlabel("Average Price")
plt.ylabel("Product Category")
plt.show()
```



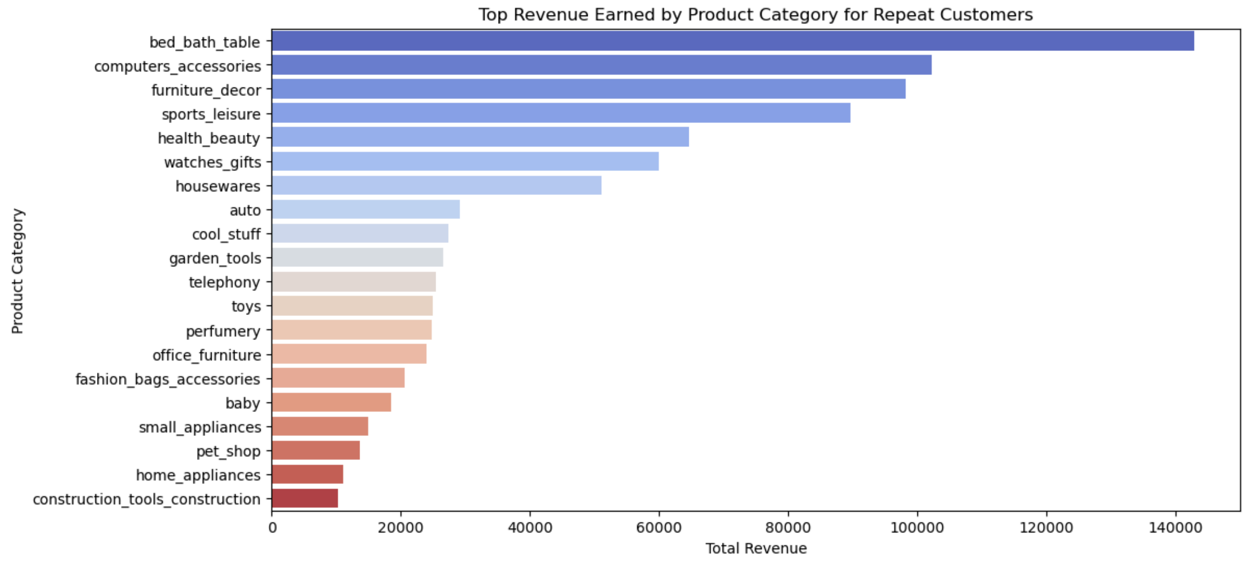
Another interesting observation is that price may not be the main factor contributing to why the repeat customers like to buy the top product categories. This shows the company should really promote those categories to boost retention.

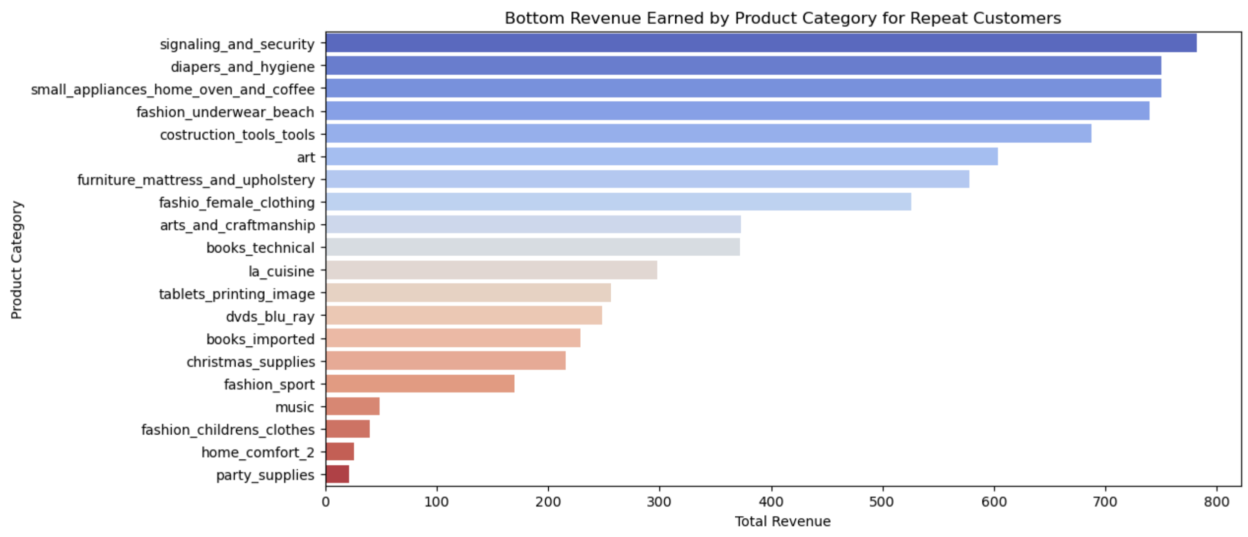
Finding the most expensive product category bought by repeat customers



On the other hand, some product categories appear in both the most expensive chart and the bottom categories purchased chart. This could indicate that price could be one of the reasons why these product categories were not sold as much.

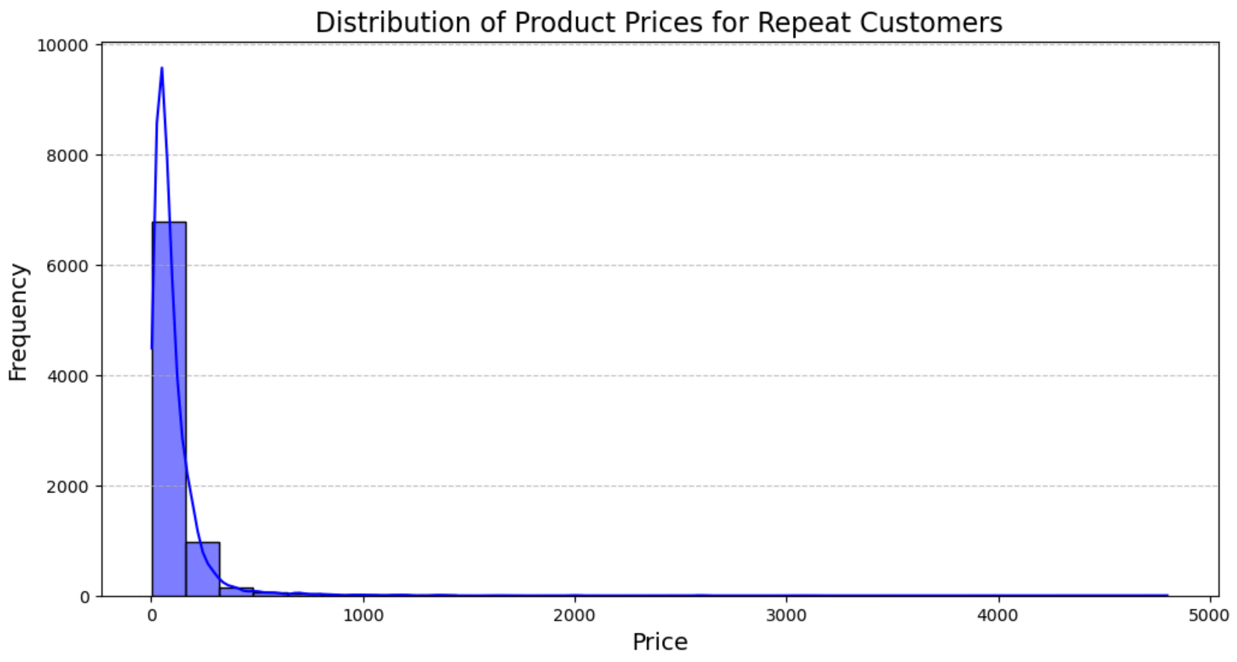
Finding the product categories with the highest and lowest revenue by repeat customers

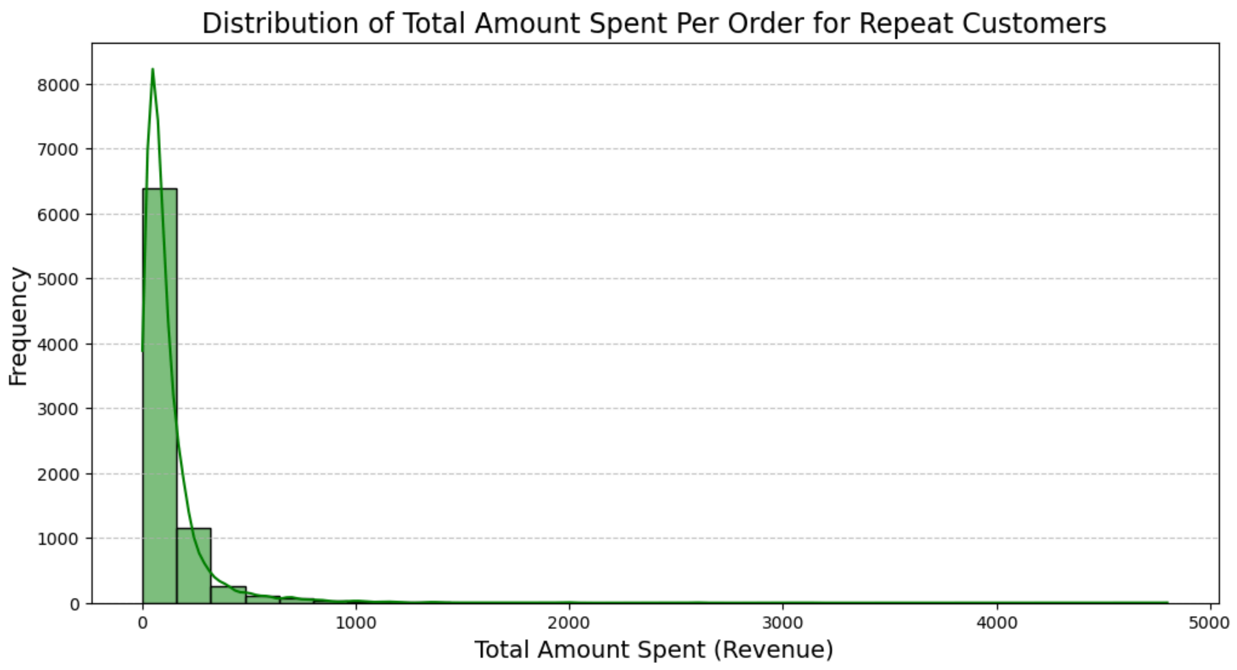




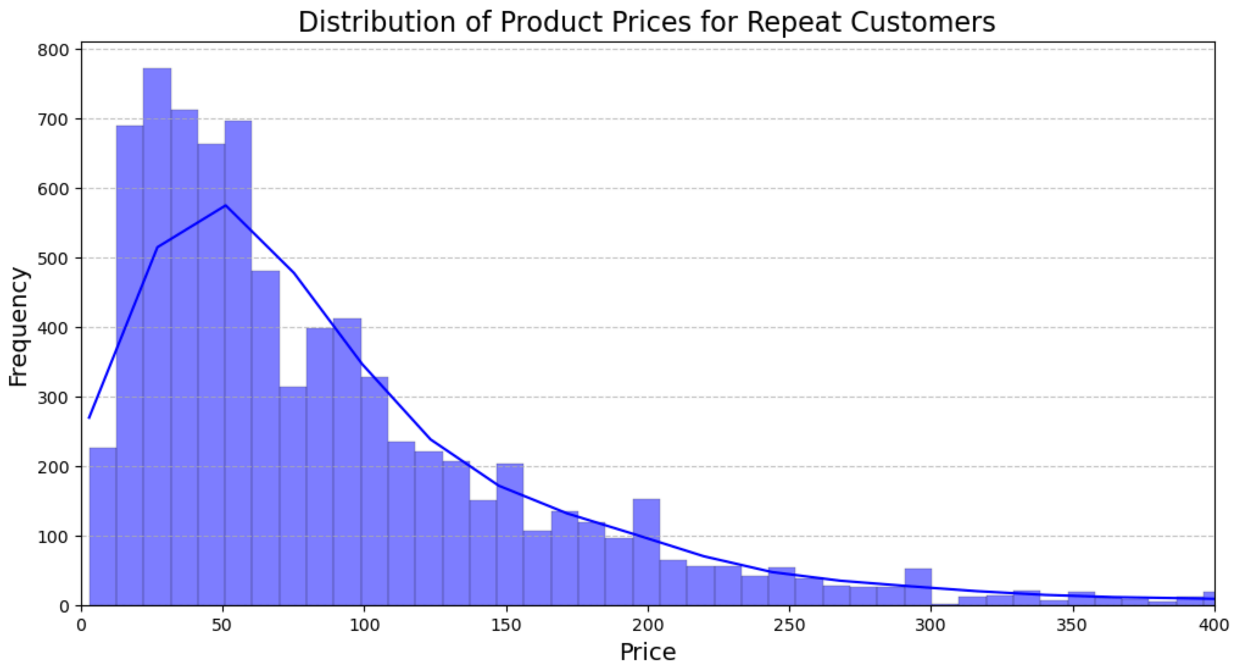
The company should take into consideration these product categories as well, especially the top earning ones, since they are the categories that bring in the most revenue by the repeating customers.
Most products in the top revenue chart also appear in the most bought categories chart, indicating that these are the categories that repeating customers really like.

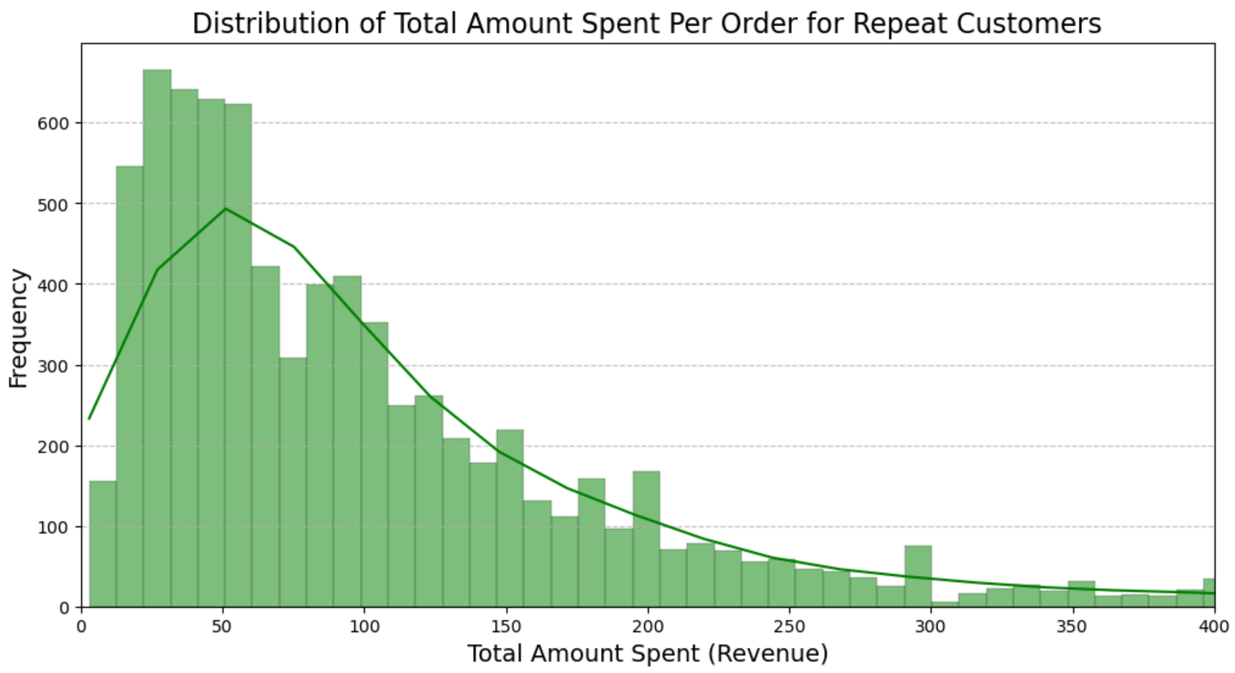
Distribution of the prices of items bought and total amount spent per order by repeat customers





Limiting the x axis to 400





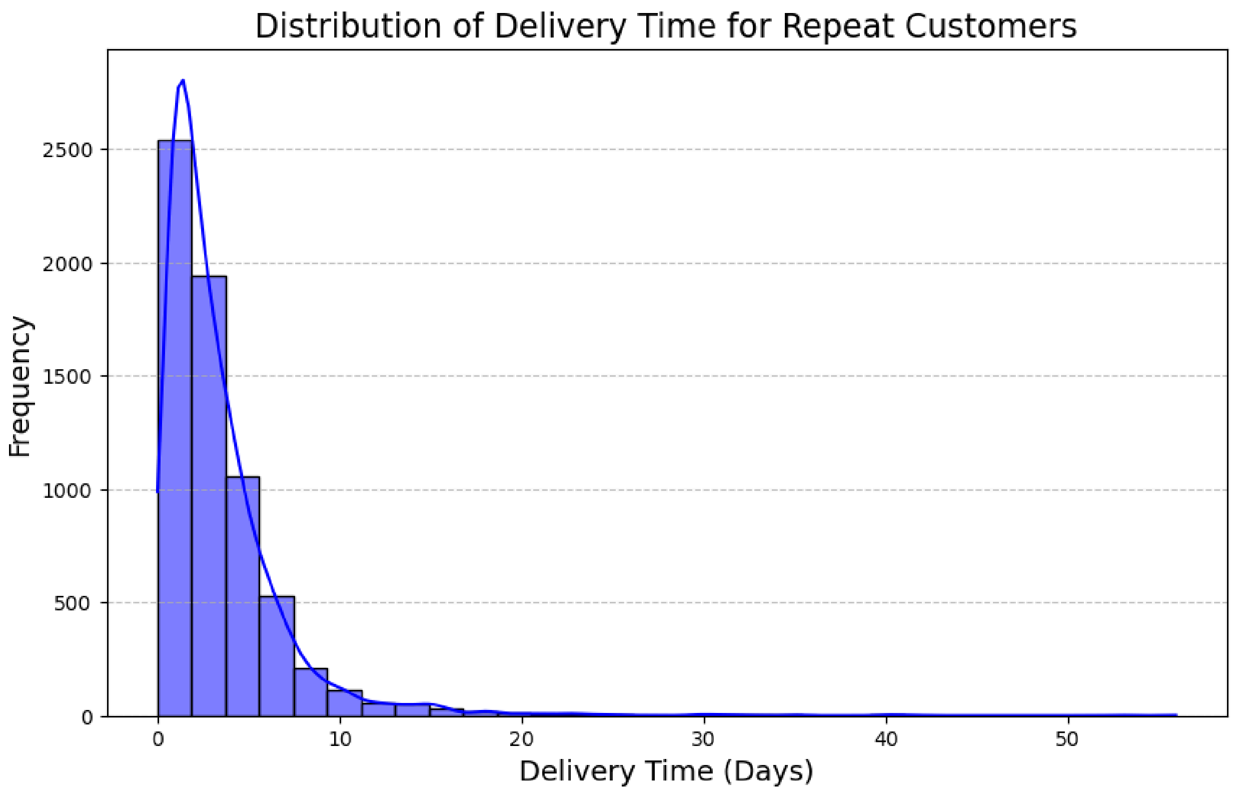
We can see that most customers do not spend much per order. Hence the company can focus on cheaper items first.

Analysing delivery times of repeat customers

In order to calculate delivery times, I will be using the order_delivered_carrier_date column. This is because it has lesser null values than the order_delivered_customer_date column and because customers may not report that they have received the item.

```
# Removing rows where courier dates are null
repeat_customers_reviews_no_missing_carrier_dates =
repeat_customers_reviews.dropna(subset=['order_delivered_carrier_date'])
# Converting order delivered date and time placed to datetime format
repeat_customers_reviews_no_missing_carrier_dates['order_delivered_carrier_date']
= pd.to_datetime(

repeat_customers_reviews_no_missing_carrier_dates['order_delivered_carrier_date']
)
repeat_customers_reviews_no_missing_carrier_dates['order_purchase_timestamp'] =
pd.to_datetime(
    repeat_customers_reviews_no_missing_carrier_dates['order_purchase_timestamp']
)

# Calculating the delivery time
repeat_customers_reviews_no_missing_carrier_dates['delivery_time'] = (
repeat_customers_reviews_no_missing_carrier_dates['order_delivered_carrier_date']
-
    repeat_customers_reviews_no_missing_carrier_dates['order_purchase_timestamp']
).abs()

# Convert the result to days
repeat_customers_reviews_no_missing_carrier_dates['delivery_time'] = (

repeat_customers_reviews_no_missing_carrier_dates['delivery_time'].dt.total_secon
ds() / 86400
) # 60 * 60 * 24

# Calculating the statistics of delivery_time
print(repeat_customers_reviews_no_missing_carrier_dates['delivery_time'].describe
())

# Plot a histogram for delivery_time
plt.figure(figsize=(10, 6))
sns.histplot(repeat_customers_reviews_no_missing_carrier_dates['delivery_time'],
             bins=30, kde=True, color='blue', edgecolor='black')
plt.title('Distribution of Delivery Time', fontsize=16)
plt.xlabel('Delivery Time (Days)', fontsize=14)
plt.ylabel('Frequency', fontsize=14)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()
```

```
count    6579.000000
mean        3.362996
std         3.479129
min        -0.041030
25%         1.210596
50%         2.363079
75%         4.287633
max        55.946528
```



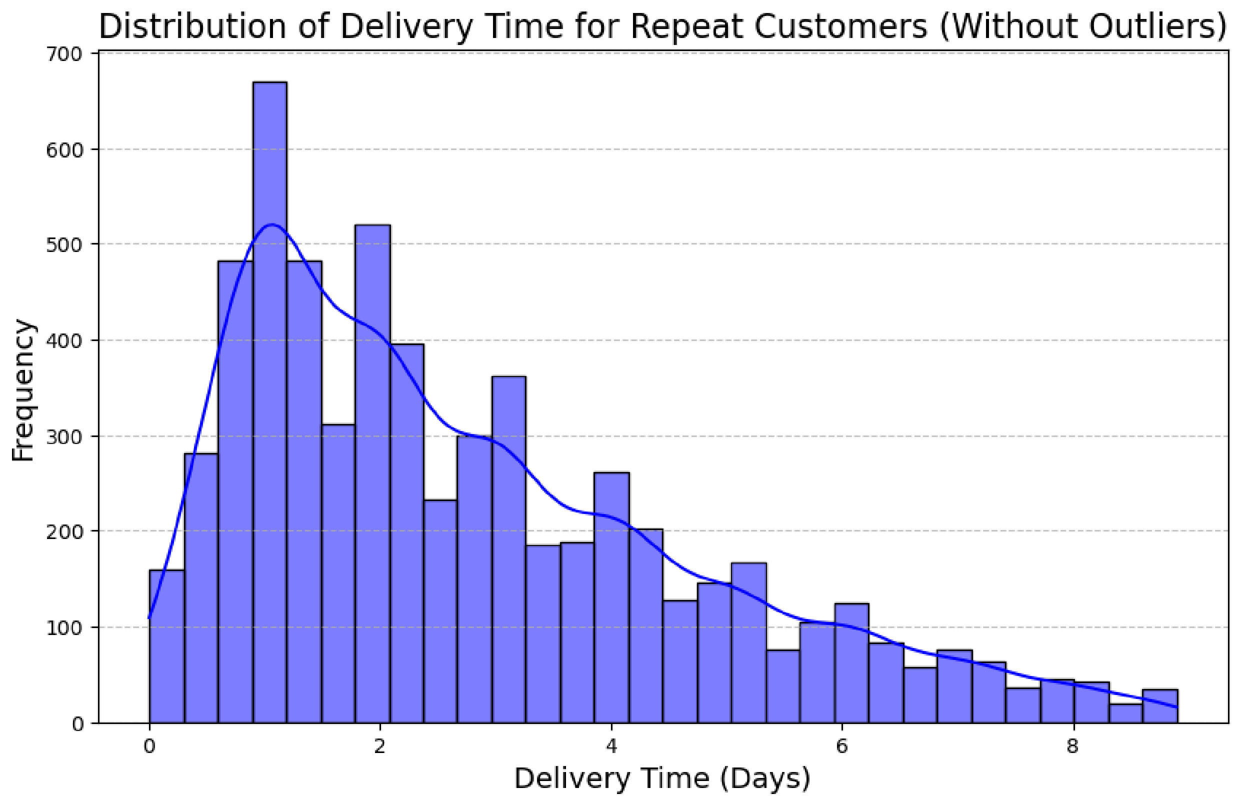
Removing outliers

There is a delivery time where it took 55 days. I will now remove the outliers and plot again.

```
# Find Q1 and Q3
Q1 =
repeat_customers_reviews_no_missing_carrier_dates['delivery_time'].quantile(0.25)
Q3 =
repeat_customers_reviews_no_missing_carrier_dates['delivery_time'].quantile(0.75)

# Find the Interquartile Range
IQR = Q3 - Q1

# Define bounds for outliers
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

# Filter the data to remove outliers
filtered_df = repeat_customers_reviews_no_missing_carrier_dates[
    (repeat_customers_reviews_no_missing_carrier_dates['delivery_time'] >=
lower_bound) &
```

```
    (repeat_customers_reviews_no_missing_carrier_dates['delivery_time'] <=
upper_bound)
]
# Calculating the statistics of delivery_time
print(filtered_df['delivery_time'].describe())

# Plot the histogram
plt.figure(figsize=(10, 6))
sns.histplot(filtered_df['delivery_time'], bins=30, kde=True, color='blue',
edgecolor='black')
plt.title('Distribution of Delivery Time (Without Outliers)', fontsize=16)
plt.xlabel('Delivery Time (Days)', fontsize=14)
plt.ylabel('Frequency', fontsize=14)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()
```

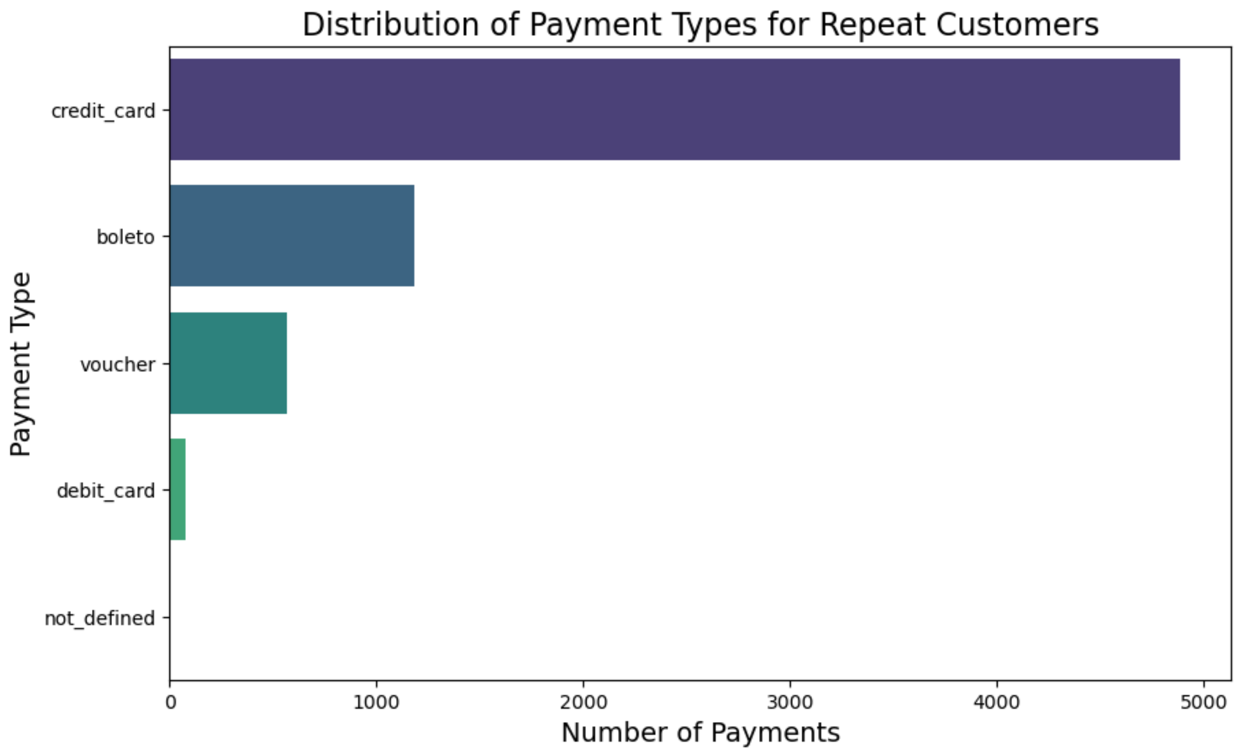
```
count    6237.000000
mean        2.777880
std         1.973935
min         0.000984
25%         1.172488
50%         2.219618
75%         3.982106
max         8.895058
```



We see that the delivery time is quite good, with most orders delivered within 4 days and a median of 2 days. This is perhaps why customers are willing to buy again.

Analysing payment types of repeat customers

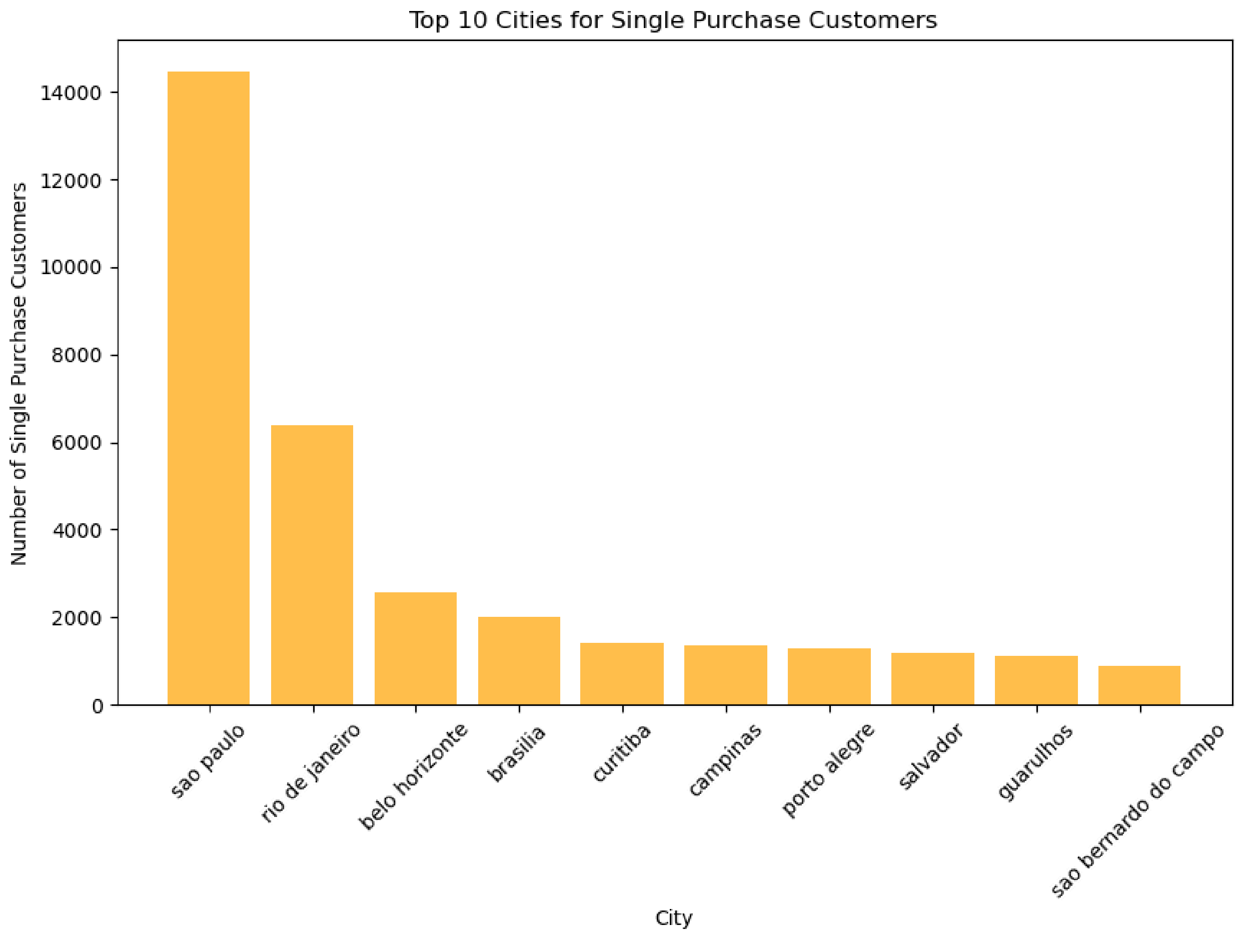
```
# Count the number of times each payment type appears
payment_type_counts = repeat_customers_payments["payment_type"].value_counts()

# Plot
plt.figure(figsize=(10, 6))
sns.barplot(x=payment_type_counts.values, y=payment_type_counts.index,
palette="viridis")
plt.title('Distribution of Payment Types for Repeat Customers', fontsize=16)
plt.xlabel('Number of Payments', fontsize=14)
plt.ylabel('Payment Type', fontsize=14)
plt.show()
```

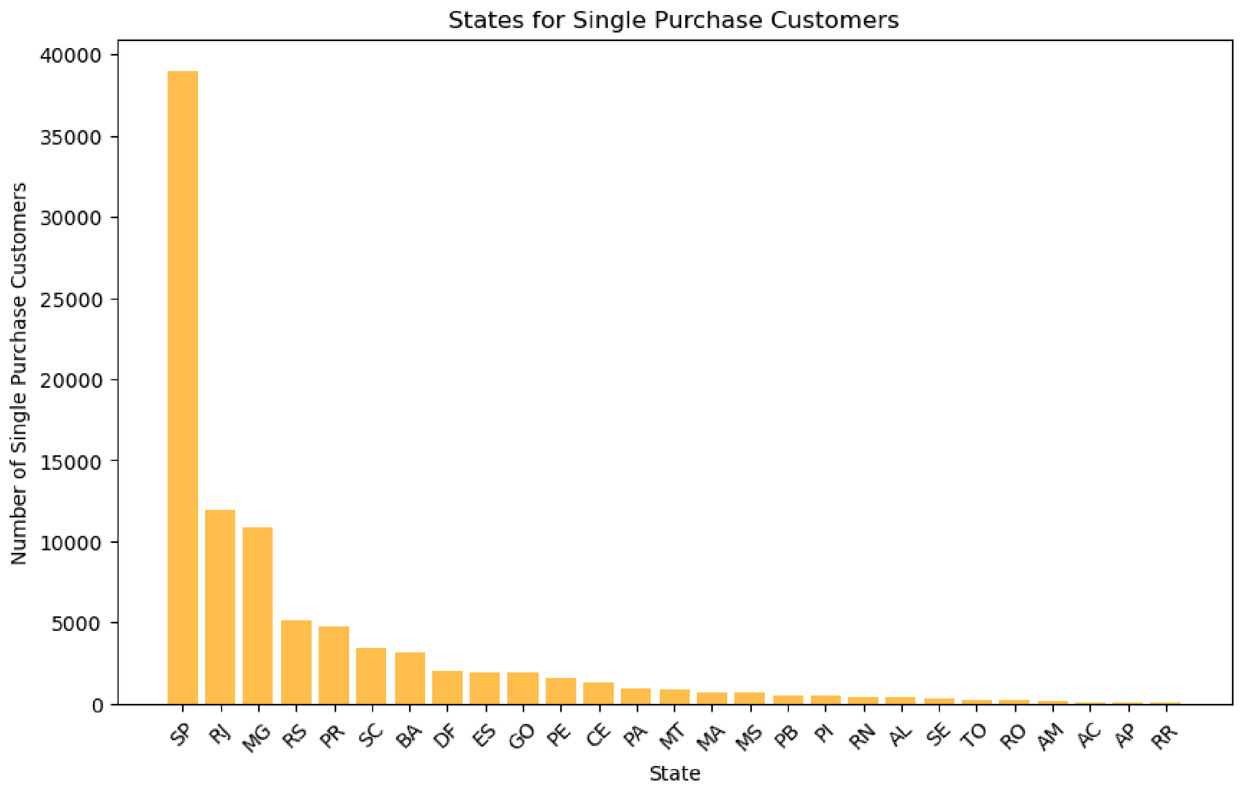


Most of the orders made were paid by credit card. I will analyse the payment types of inactive customers as well. Perhaps some of them were unable to pay by credit card which led to them leaving.

Analysis for single purchase customers

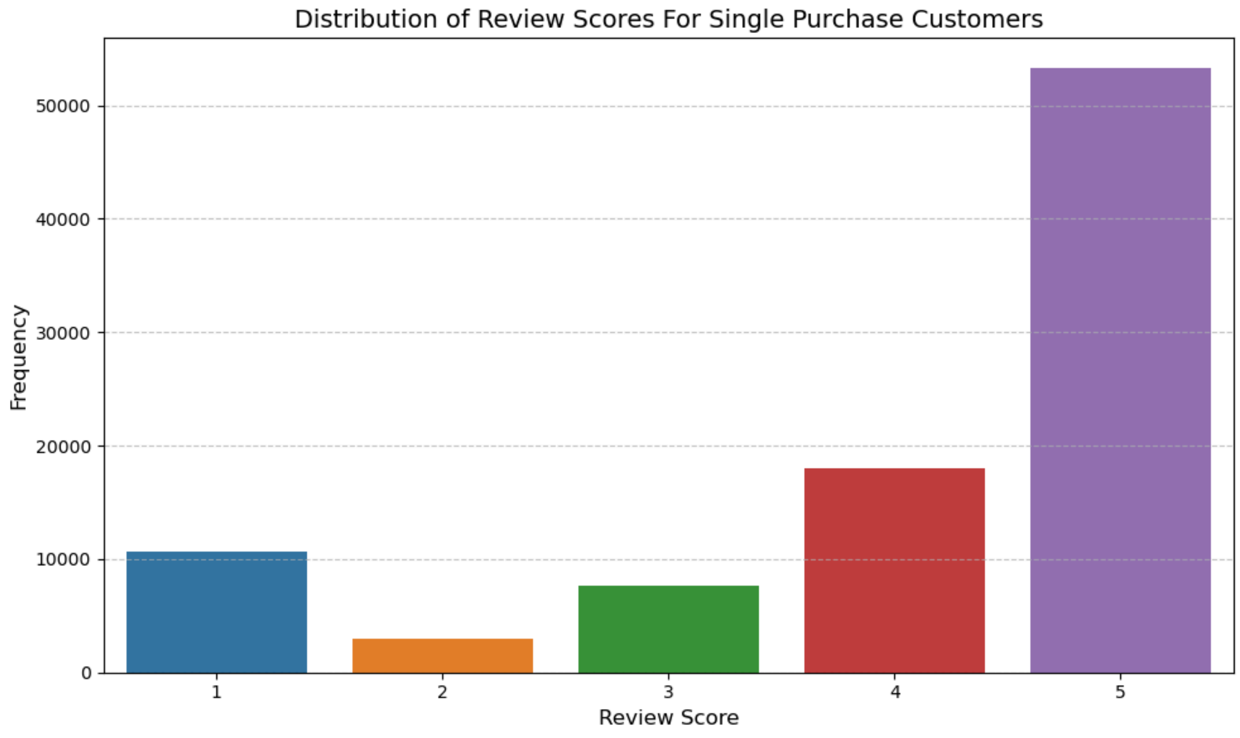
Finding the location where single purchase customers stay



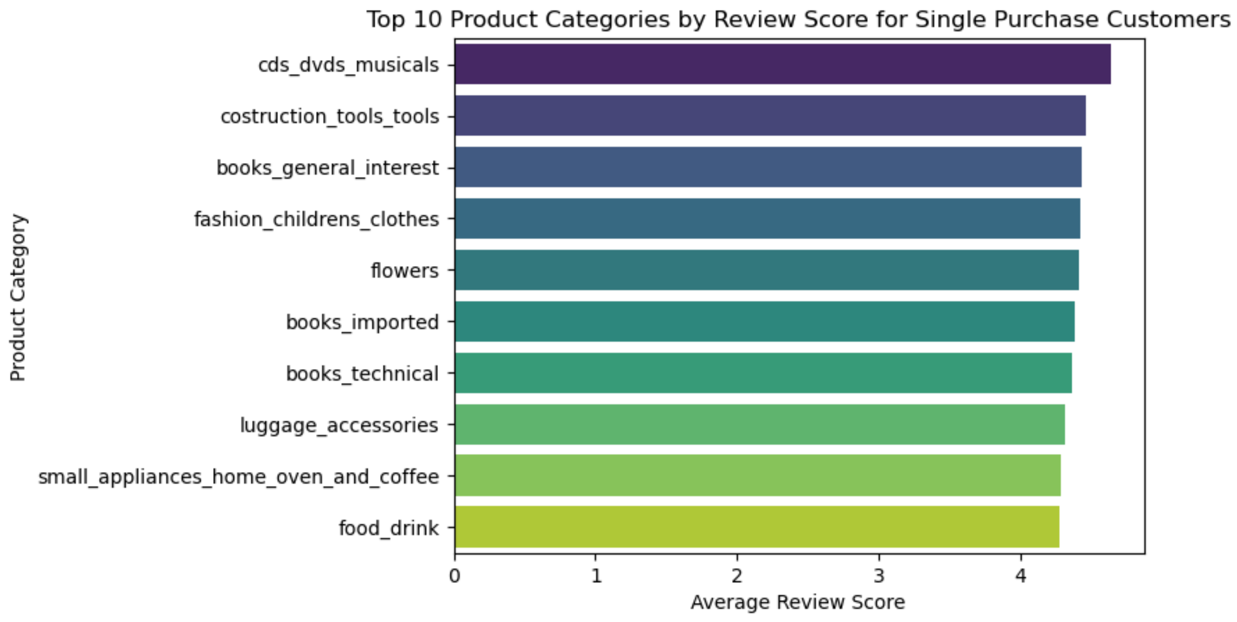


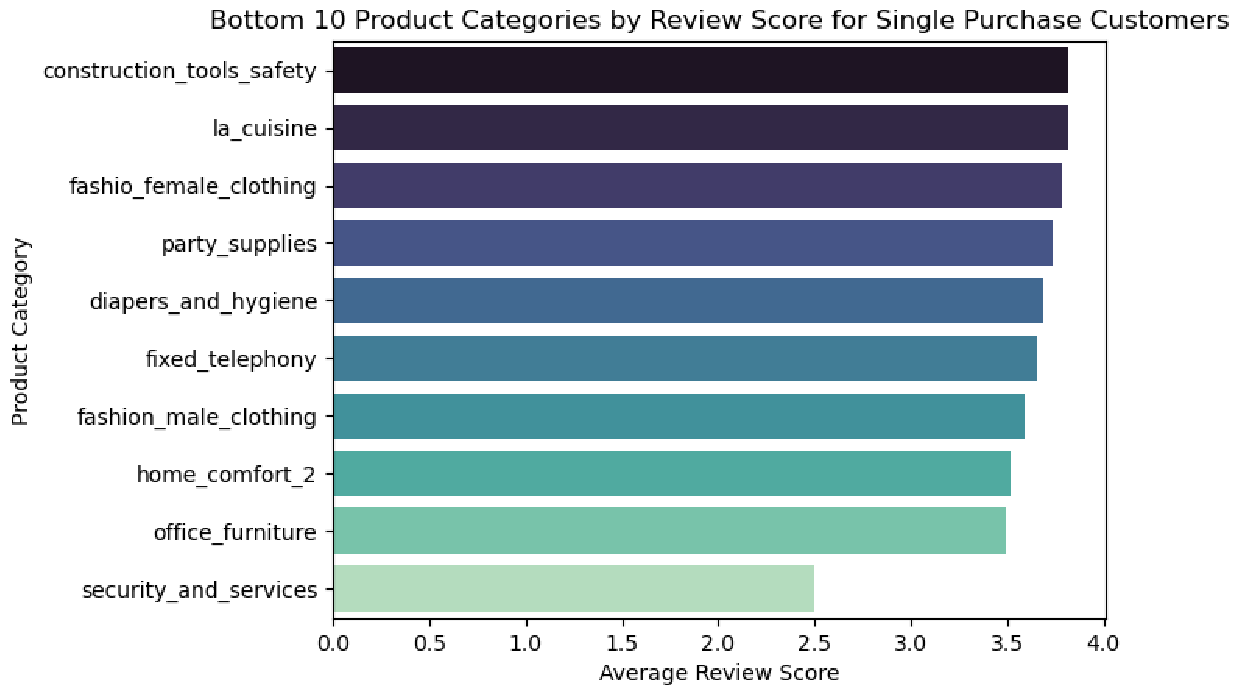
The company can conduct campaigns at these locations to encourage customers to buy again.

Analysing the reviews made by single purchase customers

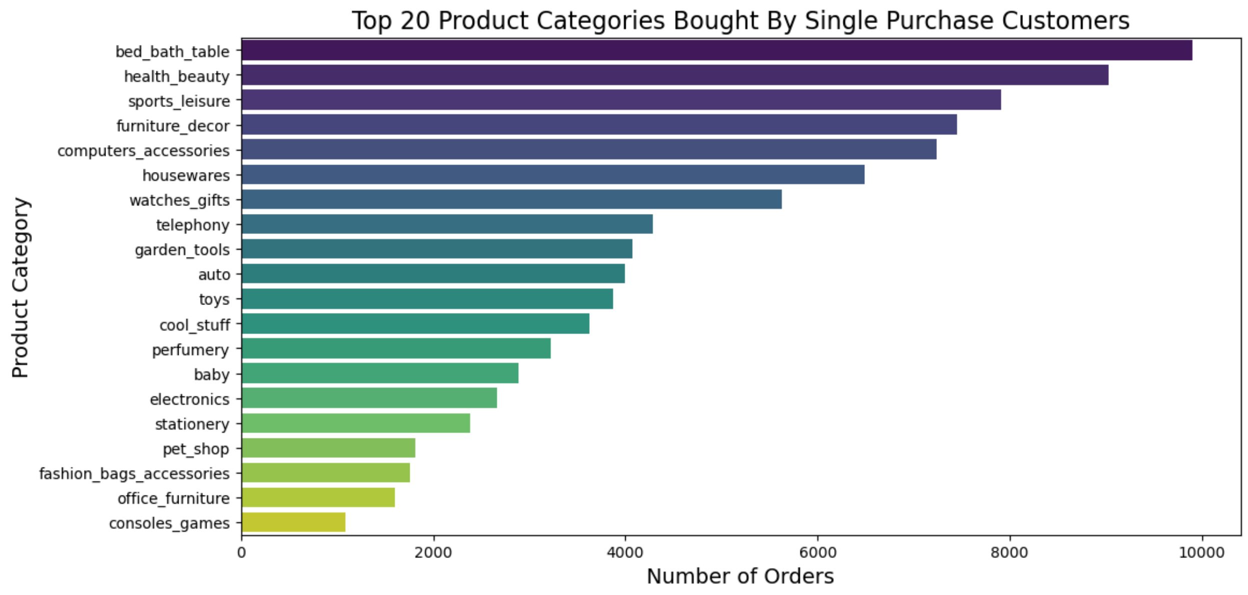


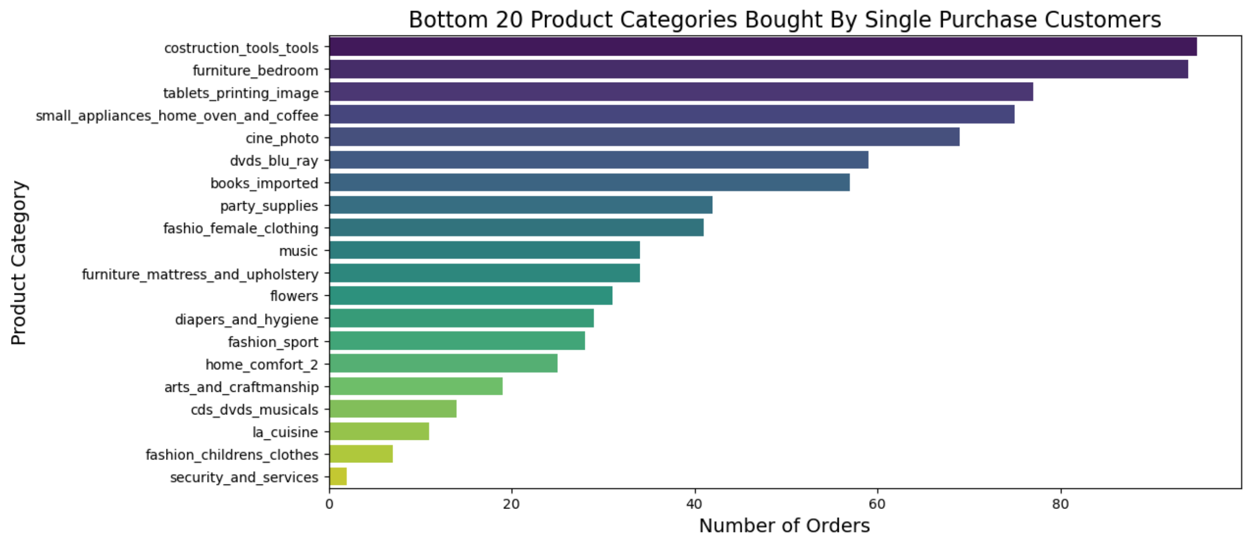
Top and bottom categories by review score



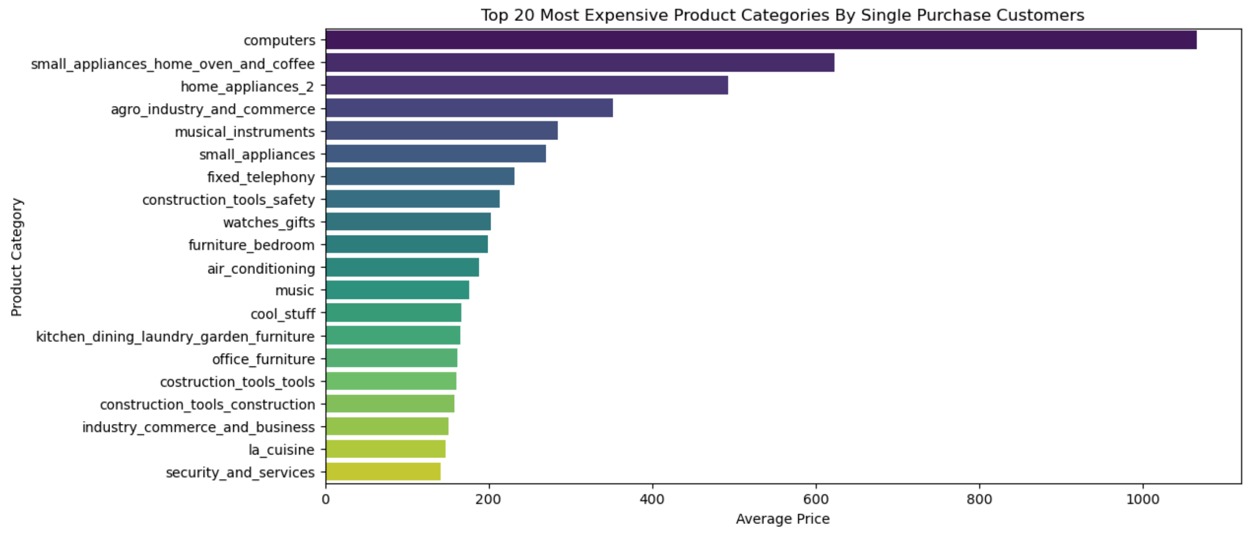


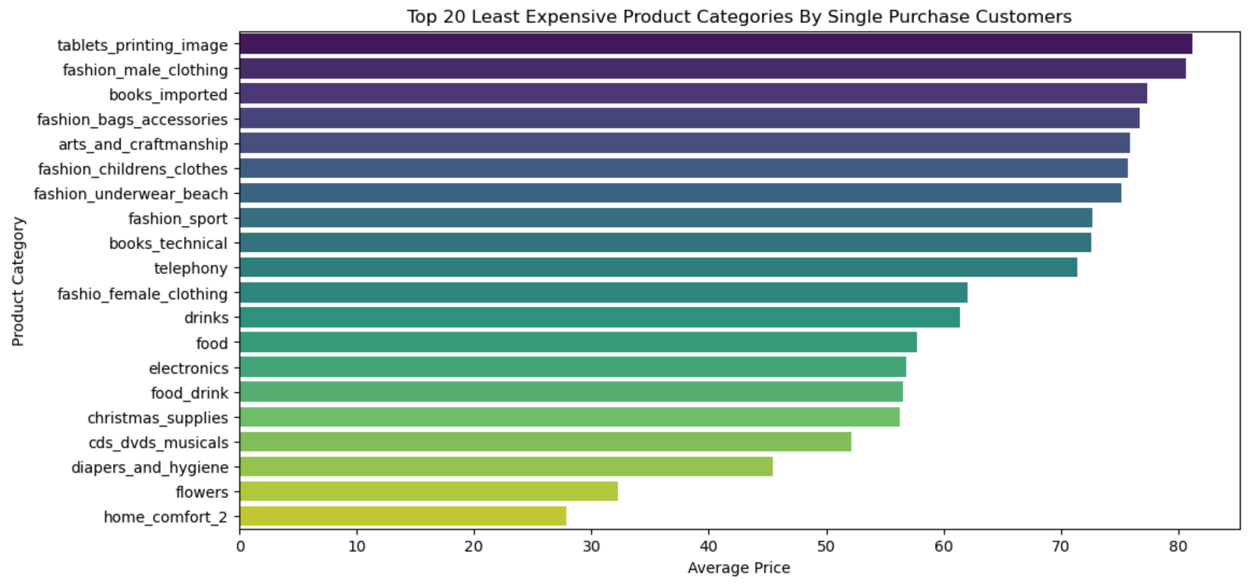
Finding the top and bottom product categories bought by single purchase customers



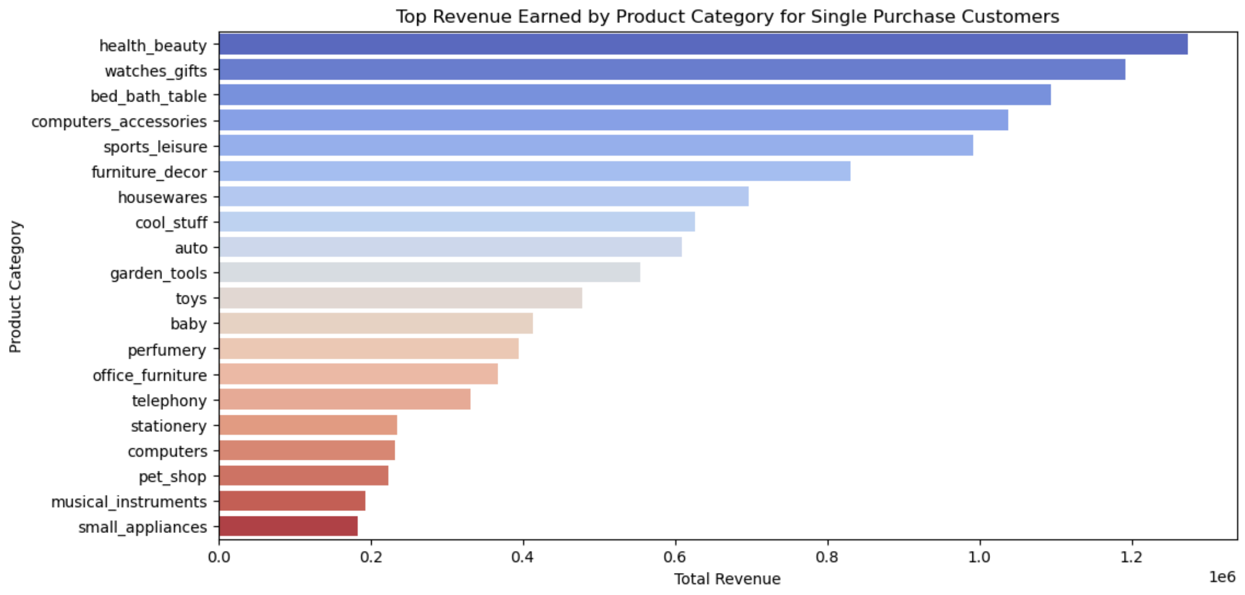


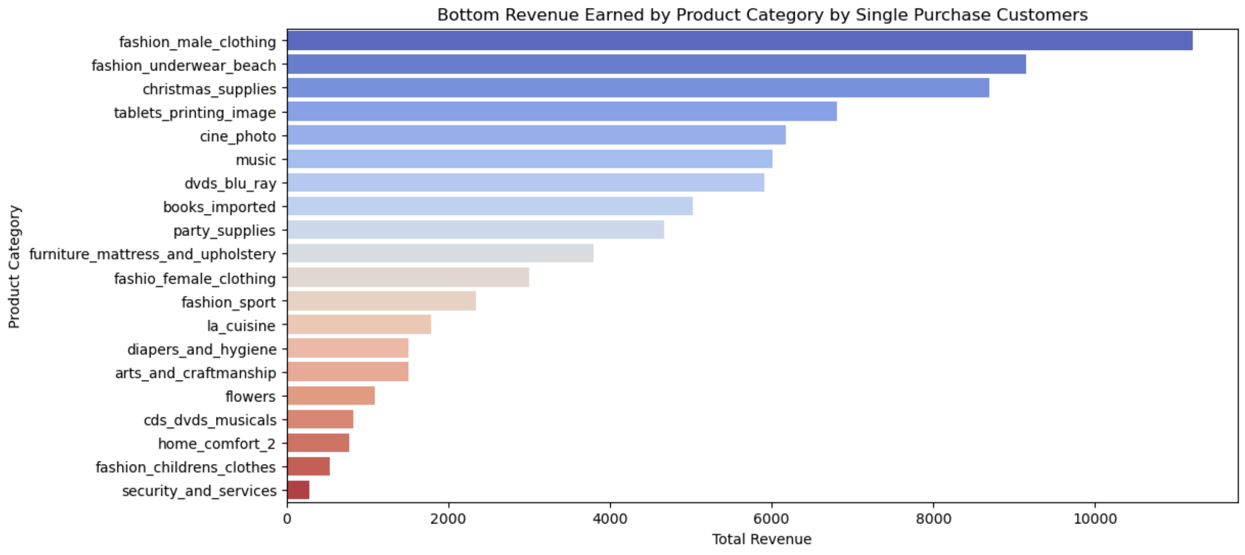
Finding the most and least expensive product categories bought by single purchase customers





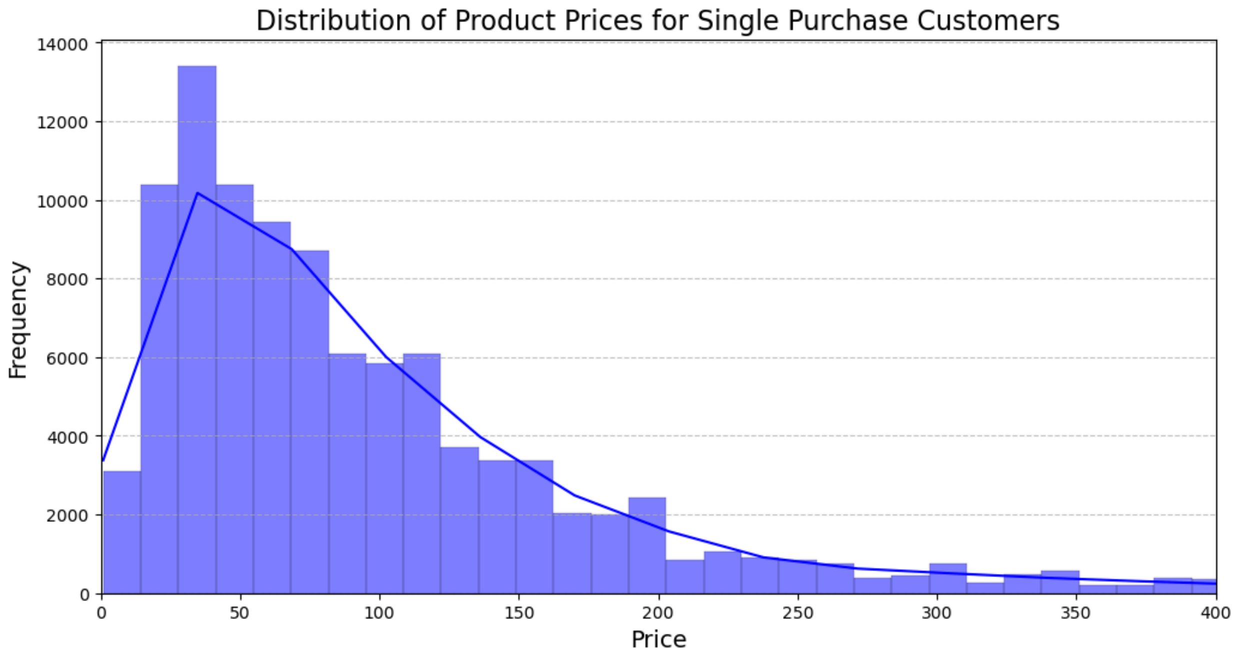
Finding the top and bottom product category by revenue made by single purchase customers

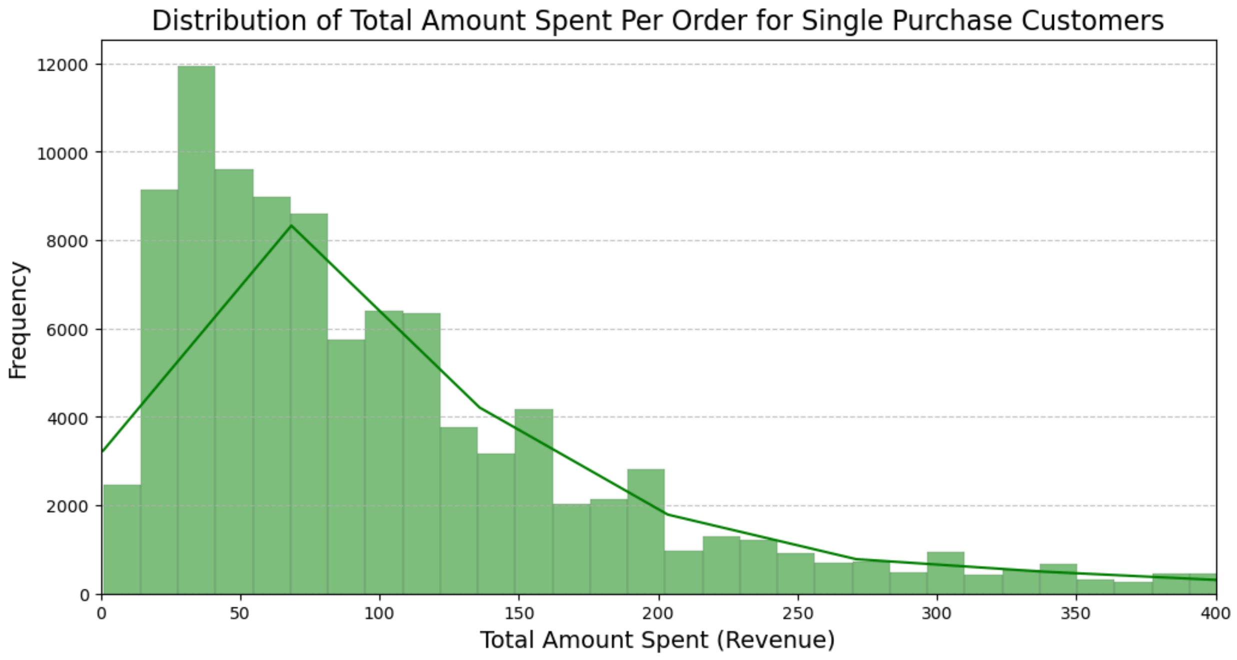




These plots are very useful because they tell the company which product categories are more likely to increase customer retention. By cross checking with the top 20 categories bought by the repeating customers, the company is able to know which categories to focus on. For example, ‘bed bath table’, ‘health beauty’ and ‘sports leisure’ are the top categories bought by the repeating customers and single purchase customers.

Distribution of the prices of items bought and total amount spent per order by single purchase customers





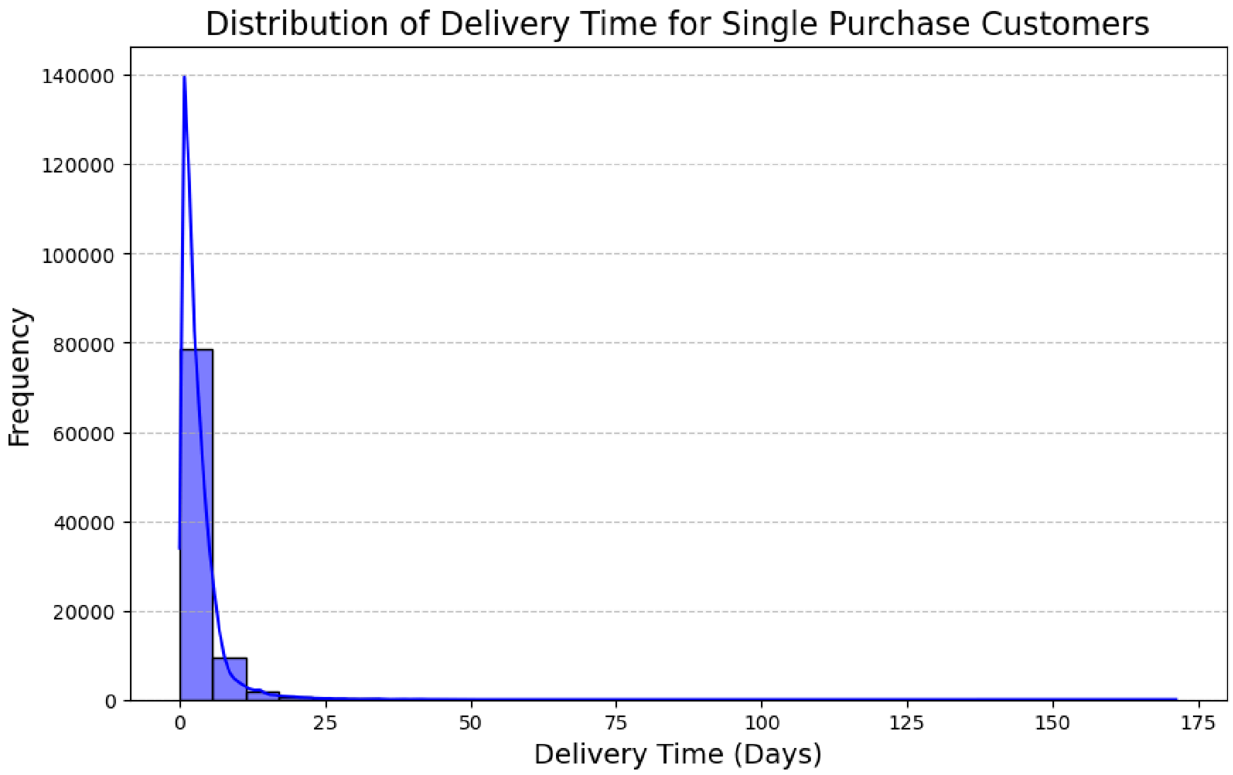
We see that these customers do not have the habit of buying expensive items or spending a lot in a single order.

Analysing delivery times for single purchase customers

Statistics:

```
count    90889.000000
mean         3.224111
```

```
std        3.595652
min        0.000278
25%        1.124444
50%        2.196285
75%        4.059097
max      171.212419
```

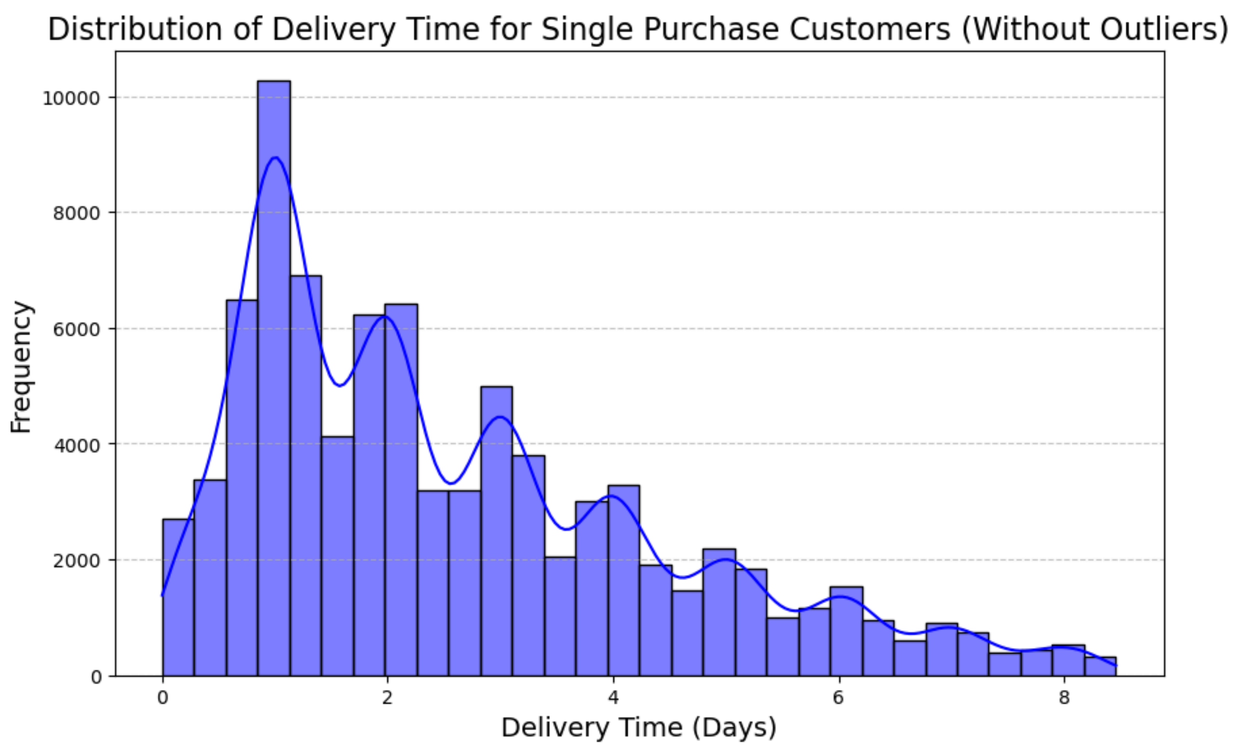


The delivery time of 171 days is very out of place. I will remove the outliers and plot again.

Remove outliers

Statistics:

```
count    86021.000000
mean         2.603553
std          1.860277
min          0.000278
25%          1.092049
50%          2.089525
75%          3.751655
max          8.459178
```



The delivery time is very good with most orders delivered within 4 days and having a median of 2. The company should keep it up in order to retain customers, as this is probably what the customers will expect from now on.

Identifying inactive customers

I categorize inactive customers as those who did not buy anything 6 months before the last date of this dataset.

```
# Find the latest order_purchase_timestamp
latest_order =
customers_orders_orderItems_orderPayments['order_purchase_timestamp'].max()

print(f"Latest order_purchase_timestamp: {latest_order}")
```

The latest order_purchase_timestamp is: 2018-09-03 09:06:57

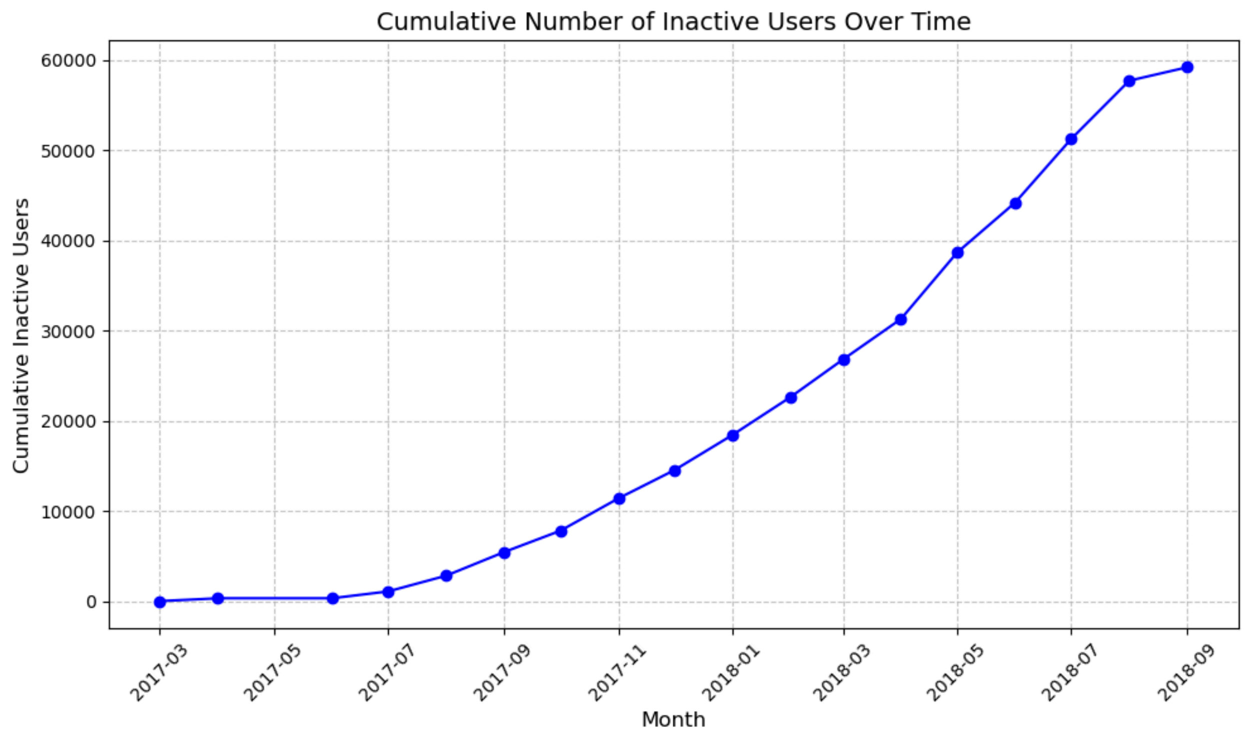
```
customers_orders['order_purchase_timestamp'] =
pd.to_datetime(customers_orders['order_purchase_timestamp'])

# Get the latest order for each customer
latest_orders_per_customer =
customers_orders.groupby('customer_unique_id')['order_purchase_timestamp'].max().
reset_index()
```

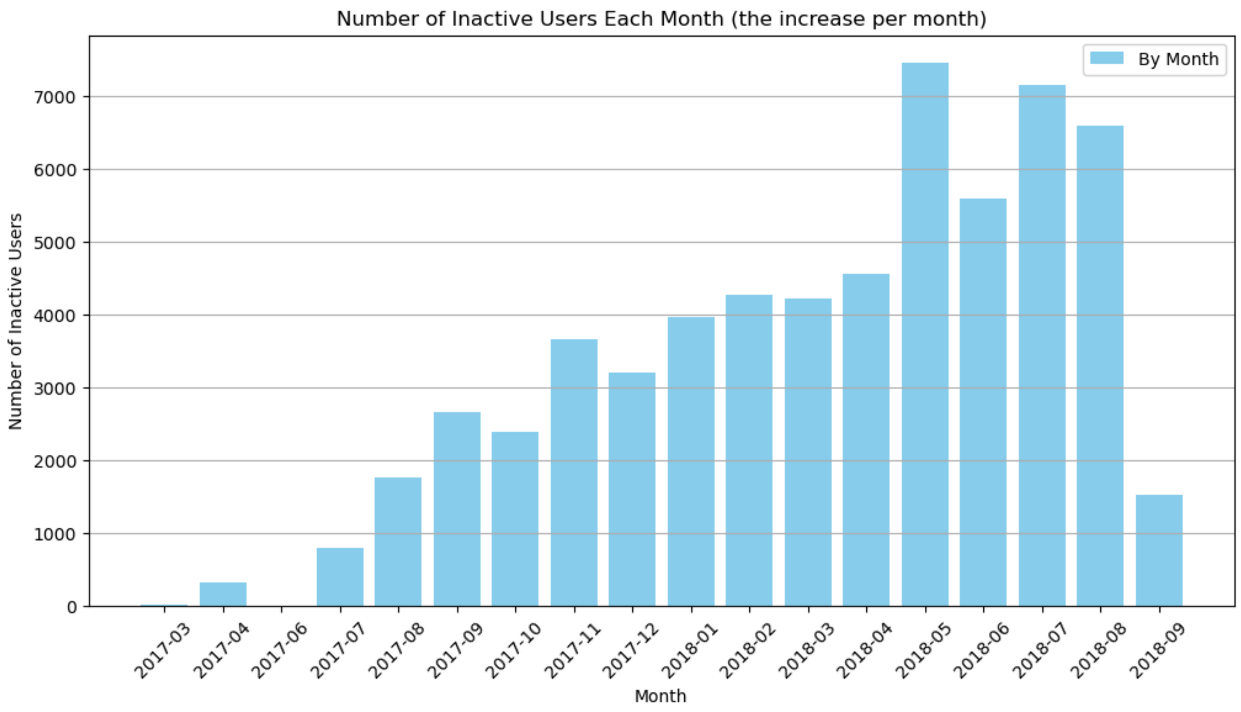
```
# 6 months inactivity period
inactivity_period = pd.Timedelta(days=6*30)  # 6 months = 180 days

# Find the cutoff date for inactivity
cutoff_date = pd.to_datetime('2018-09-03 09:06:57') - inactivity_period

# Classify customers as inactive if their latest order is older than the cutoff
date
inactive_customers =
latest_orders_per_customer[latest_orders_per_customer['order_purchase_timestamp']
< cutoff_date]
```

The code above is to identify the inactive customers

Analysis of inactive customers

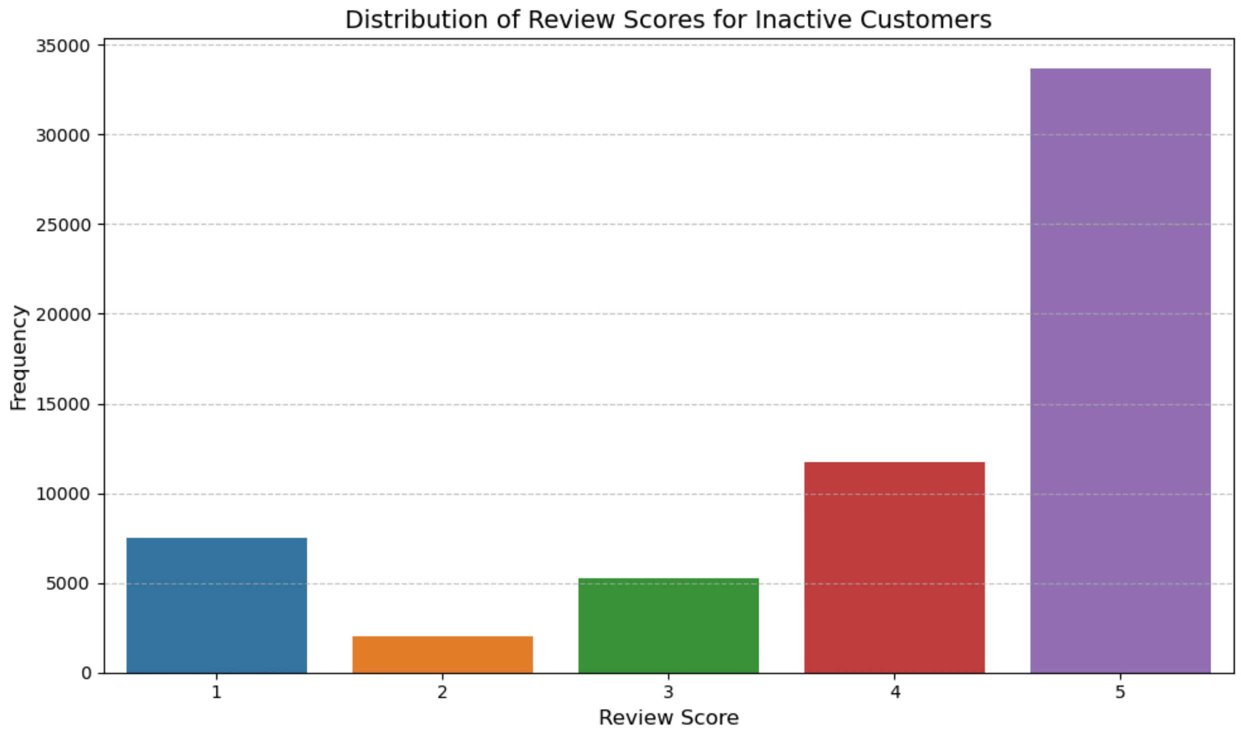
Plotting the number of inactive customers over time



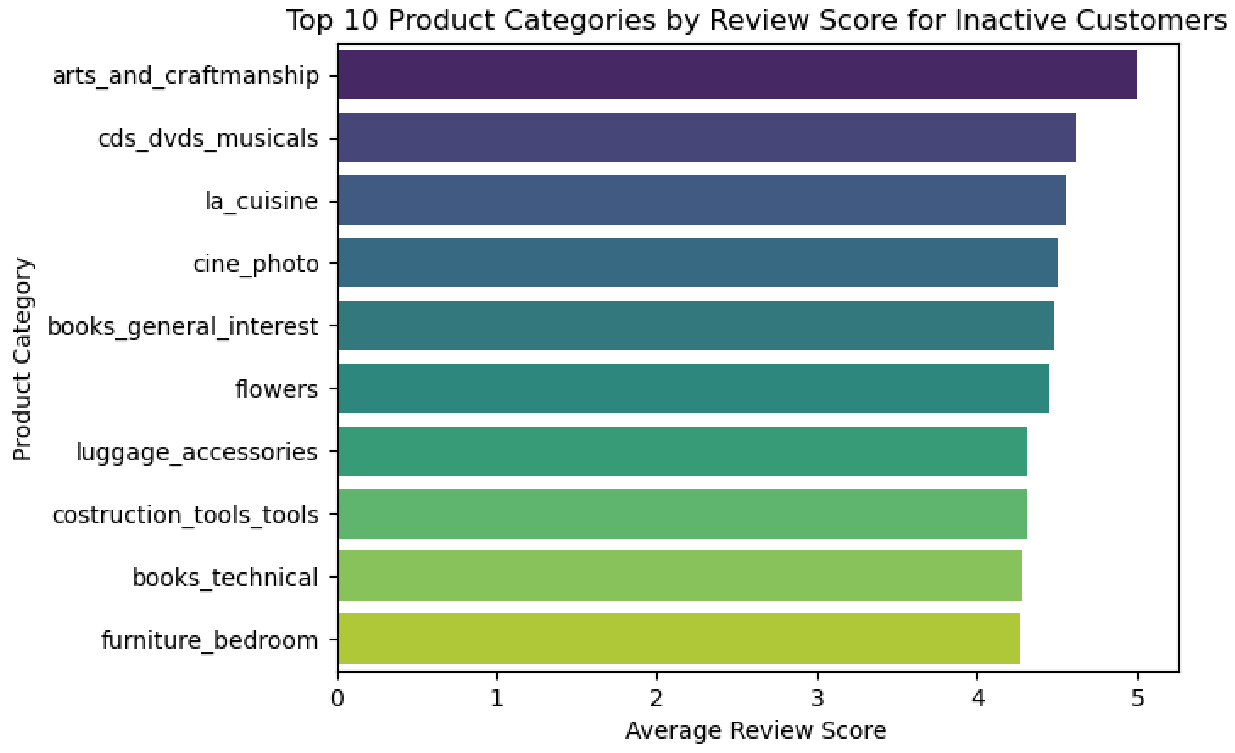


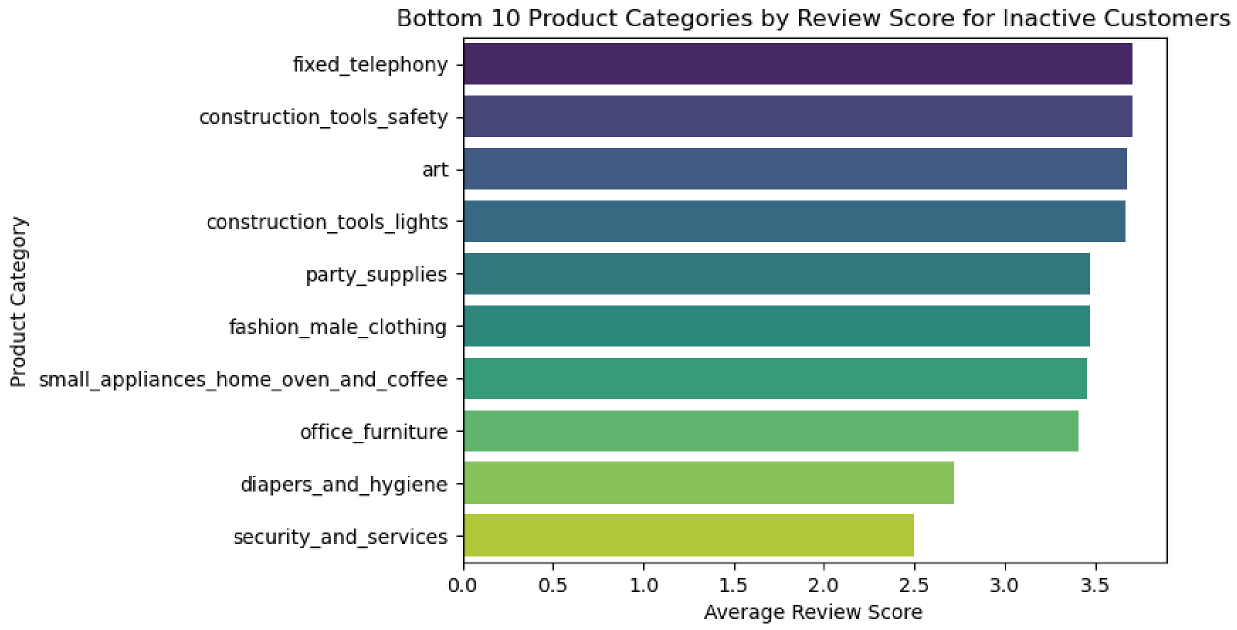
Analysing the reviews of inactive customers

Distribution of review scores

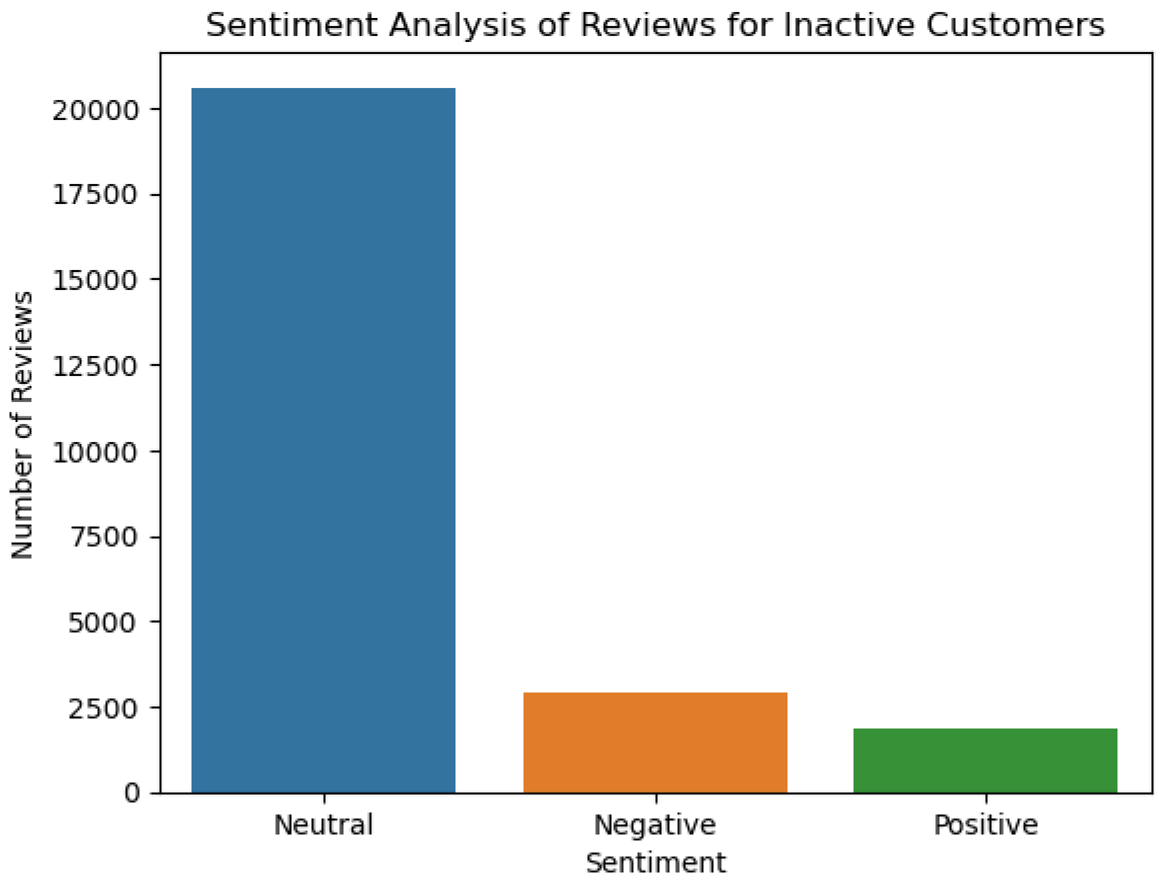


Finding the top and bottom product categories by review score

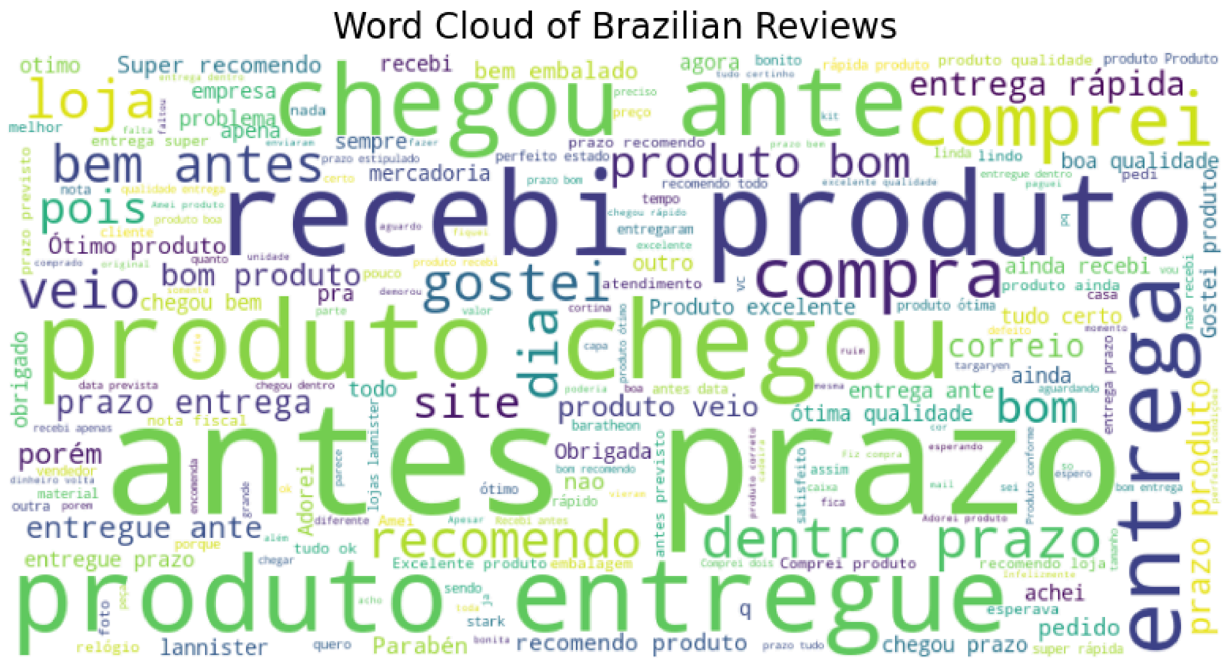




Review sentiment analysis



Wordcloud

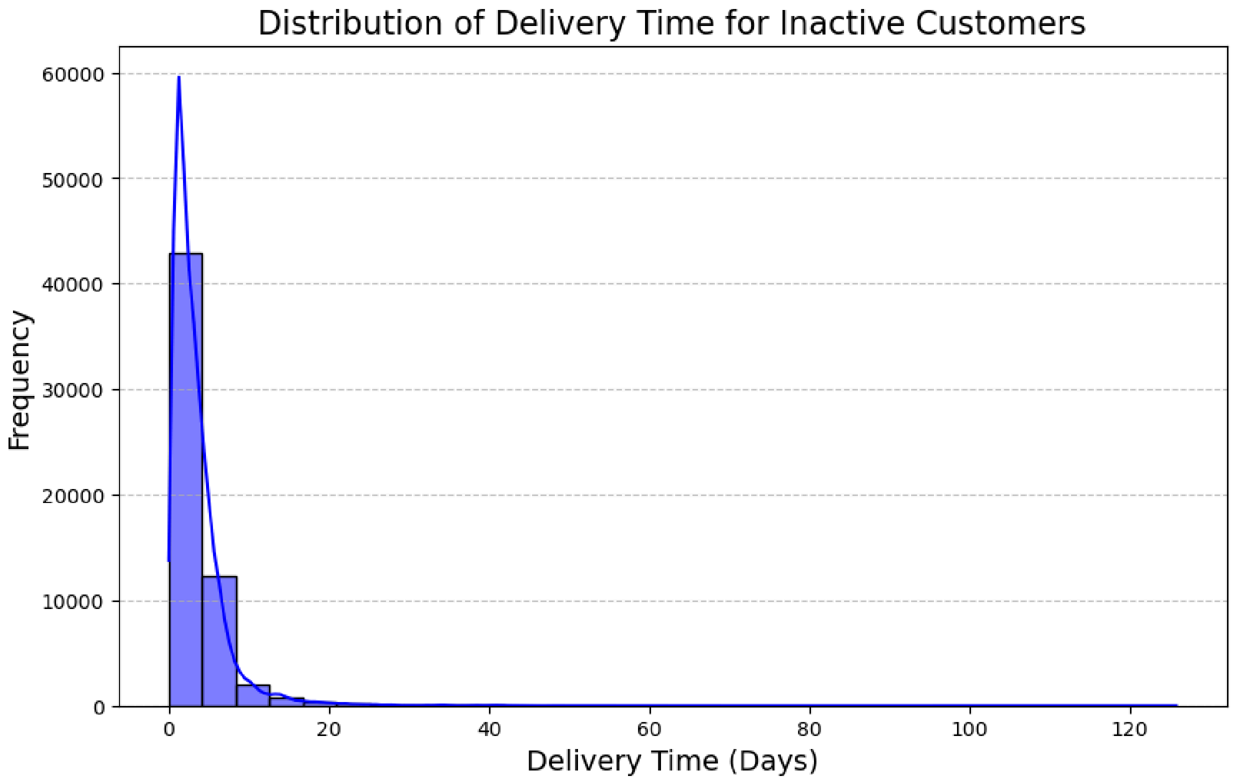


Insights from review analysis of inactive customers

The sentiment analysis is similar to that of repeating customers. The distribution of review scores is surprisingly good, as I expected there to be more negative reviews. It is interesting to note that the average review score of each product has generally fallen, as the top product categories have a lower review score than repeating customers.
This also shows that customers may not have left due to receiving bad products.

Analysing delivery times for inactive customers

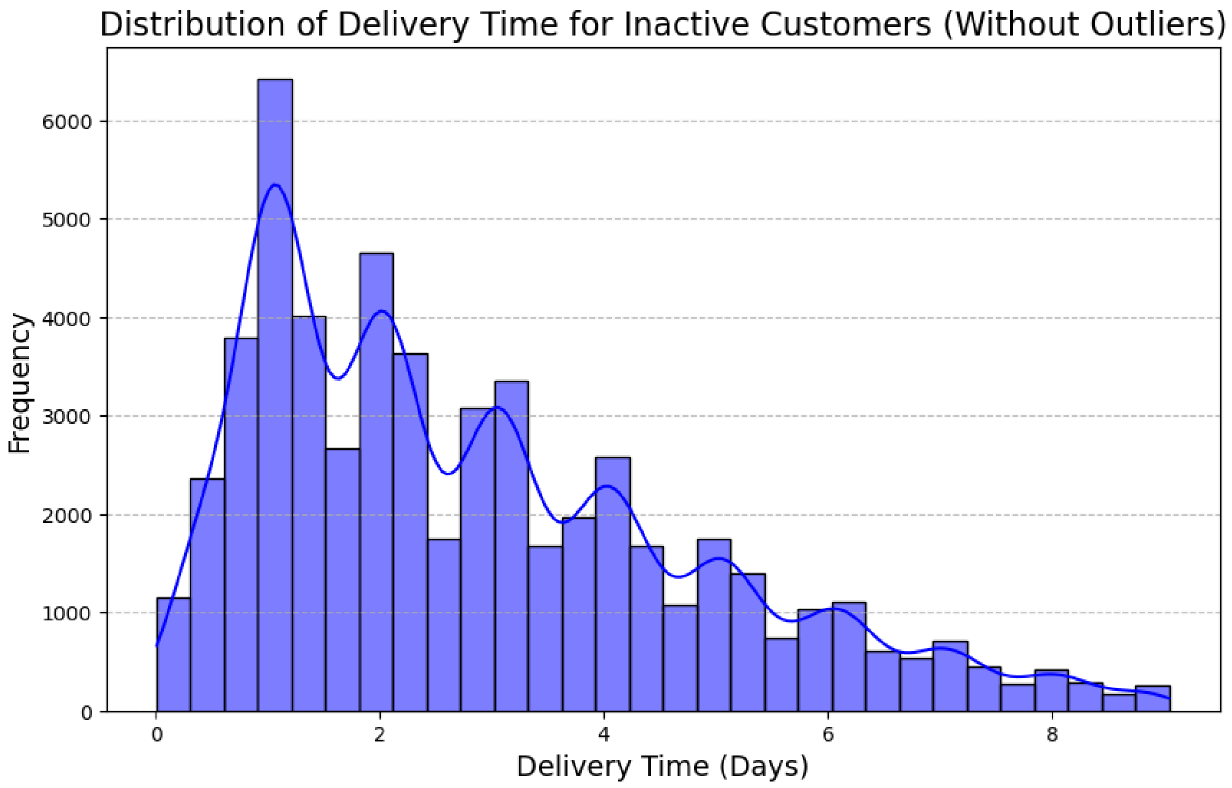
```
count    58767.000000
mean         3.523622
std          3.811586
min          0.010799
25%          1.267946
50%          2.533056
75%          4.379358
max        125.775521
```



Removing outliers

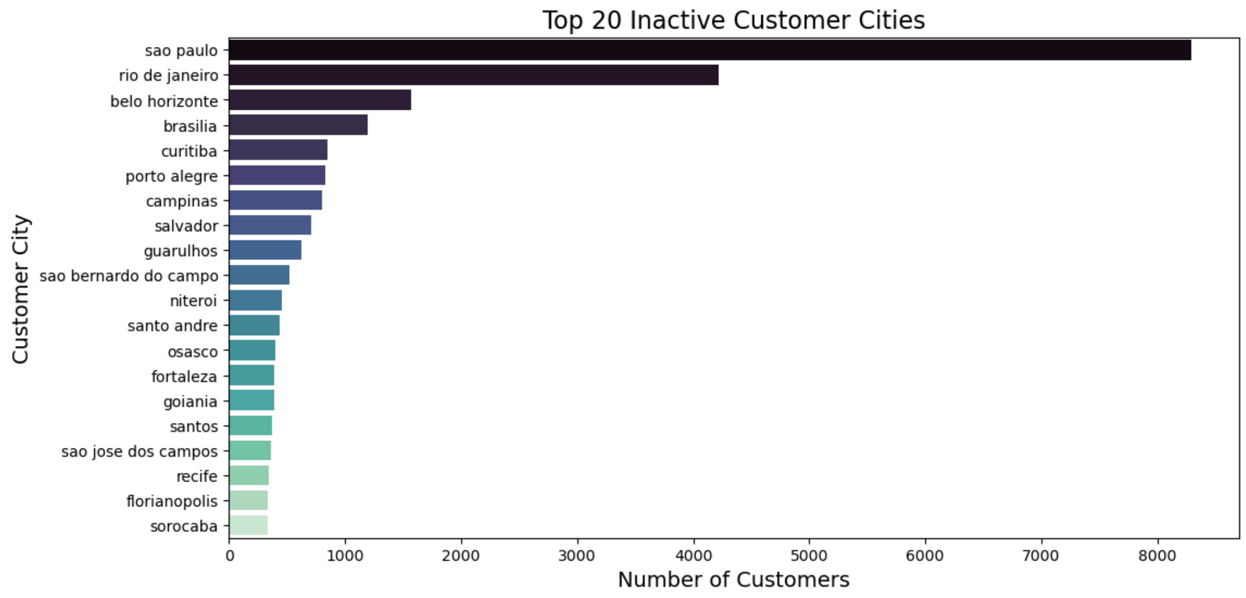
```
count    55607.000000
mean         2.865096
std          1.962515
min          0.010799
25%          1.224711
50%          2.319722
```

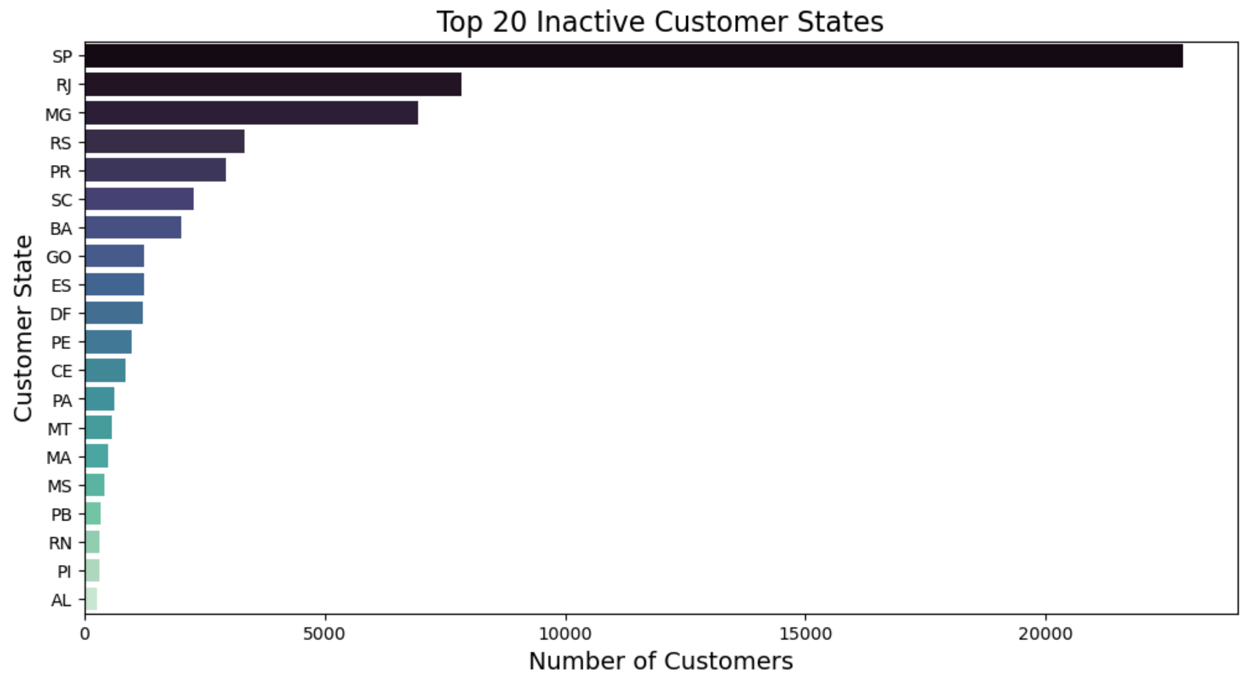
```
75%        4.062616
max        9.045671
```



Again, the delivery time is quite good, with most orders delivered within 4 days and a median of 2 days. However, there is a max value of 125 days, which may be why the customer decided not to order anymore.

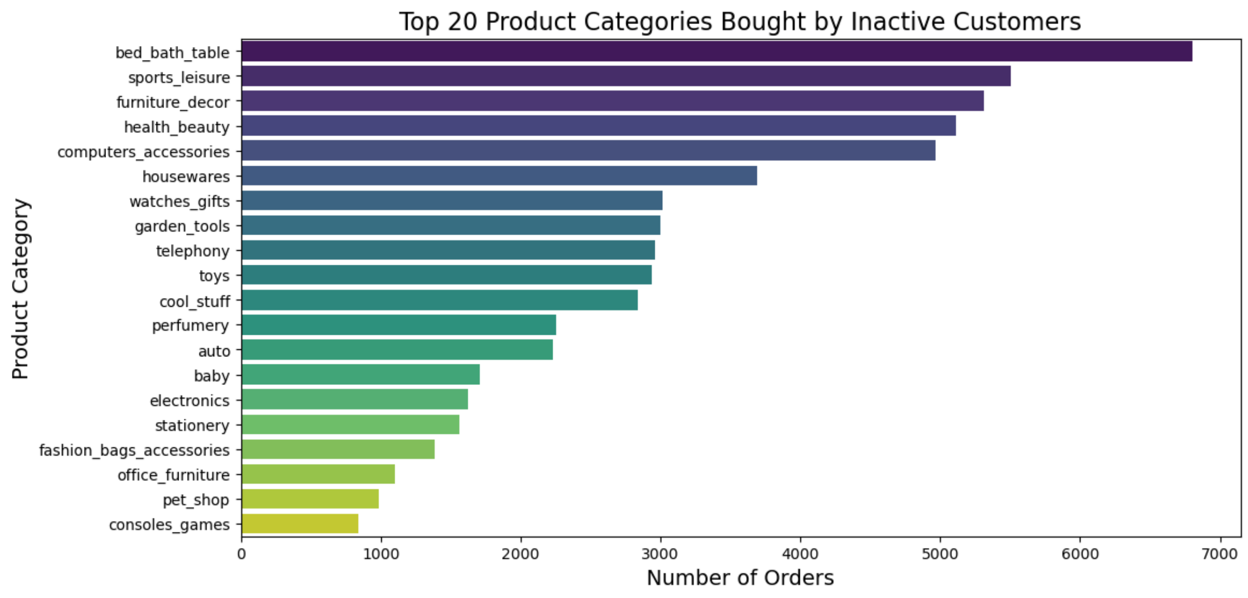
Finding the location of where inactive customers stay

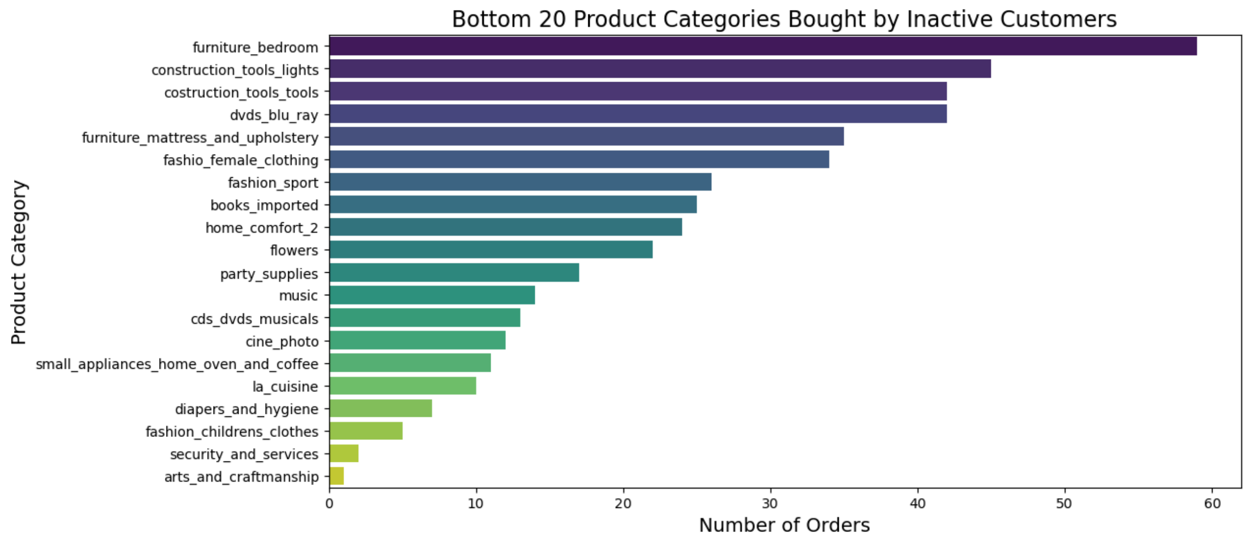




This plot shows the company where most of the inactive customers stay. This means if the company wants to get the customers back and improve customer retention, they can conduct marketing campaigns there.

Finding the top products purchased by inactive customers

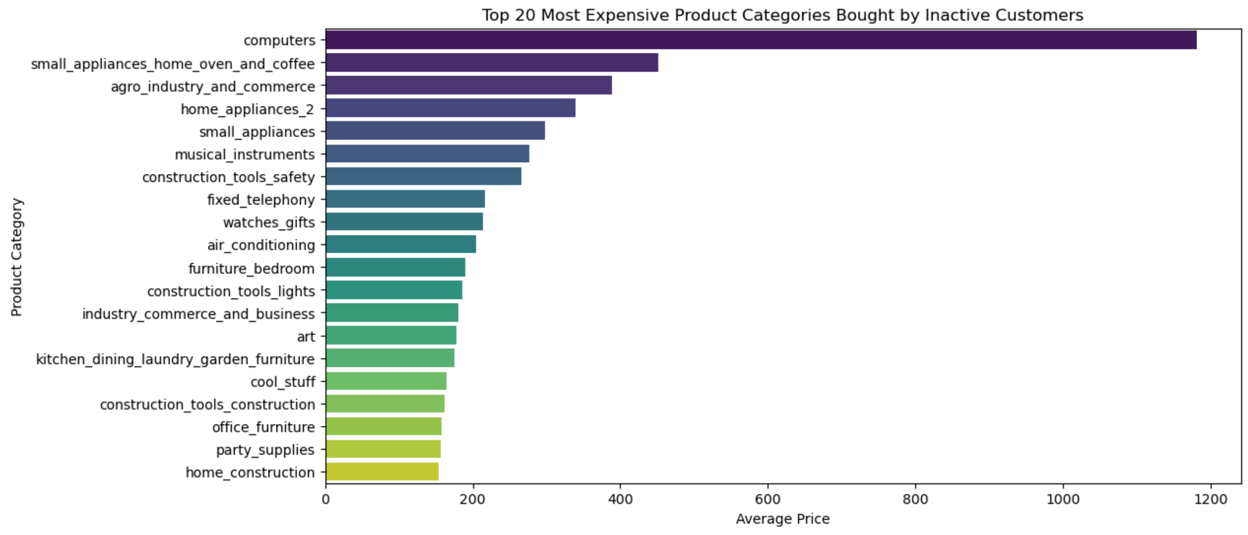


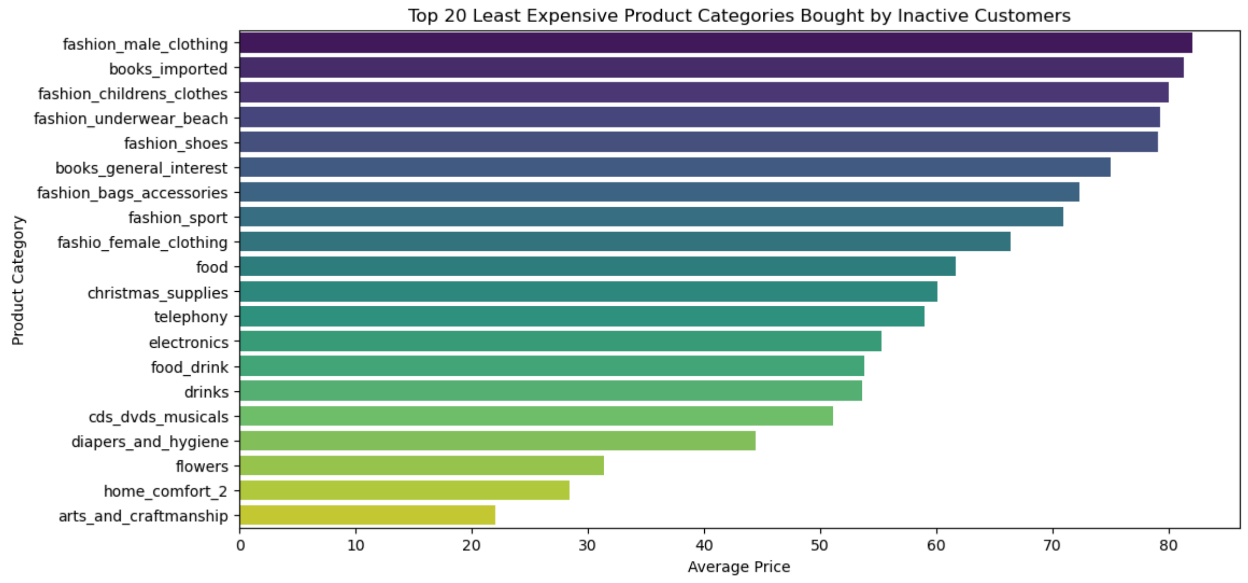


The top product categories plot tells the company which products they should promote in order to increase the chance of getting the inactive customers to buy again.
The bottom product categories plot tells the company which products they should avoid as these products are not well liked.

Ironically, some of the top rated product categories are in the bottom categories sold. Thus I investigated the price of the product categories as well

Finding the most and least expensive product categories purchased by inactive customers

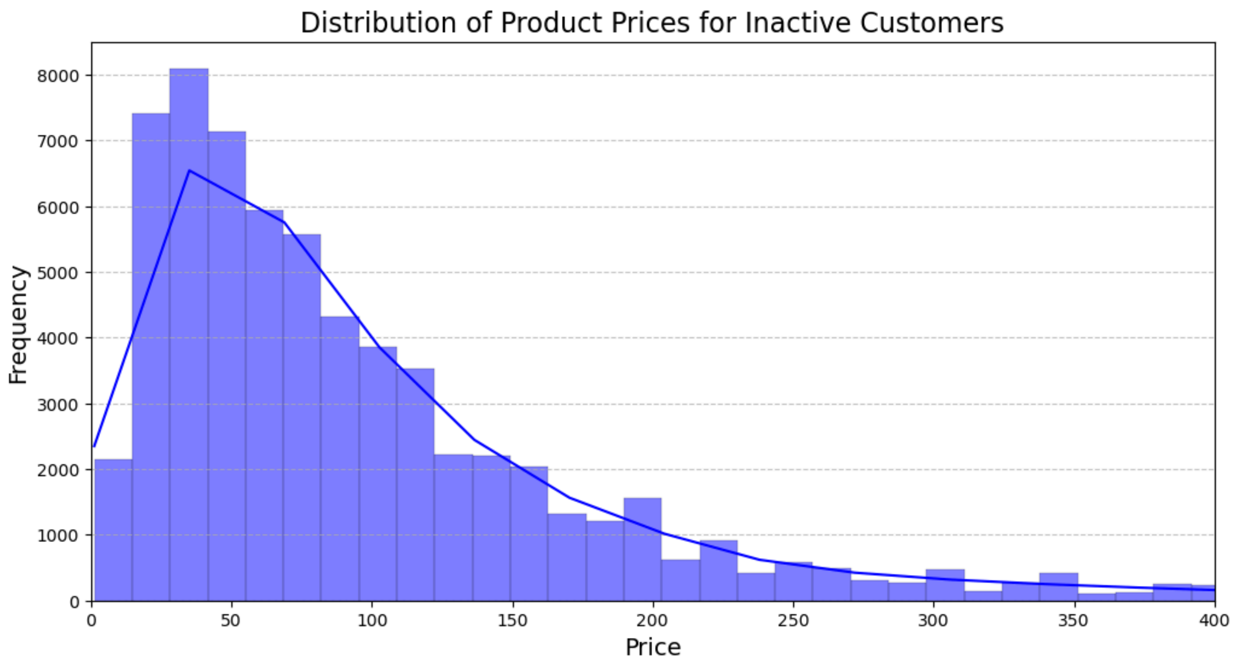




This is giving mixed observations as some of the expensive items appear in the bottom categories sold chart (which is expected as people are less willing to buy expensive items repeatedly) but some of the cheapest items also appear in the bottom categories sold chart.

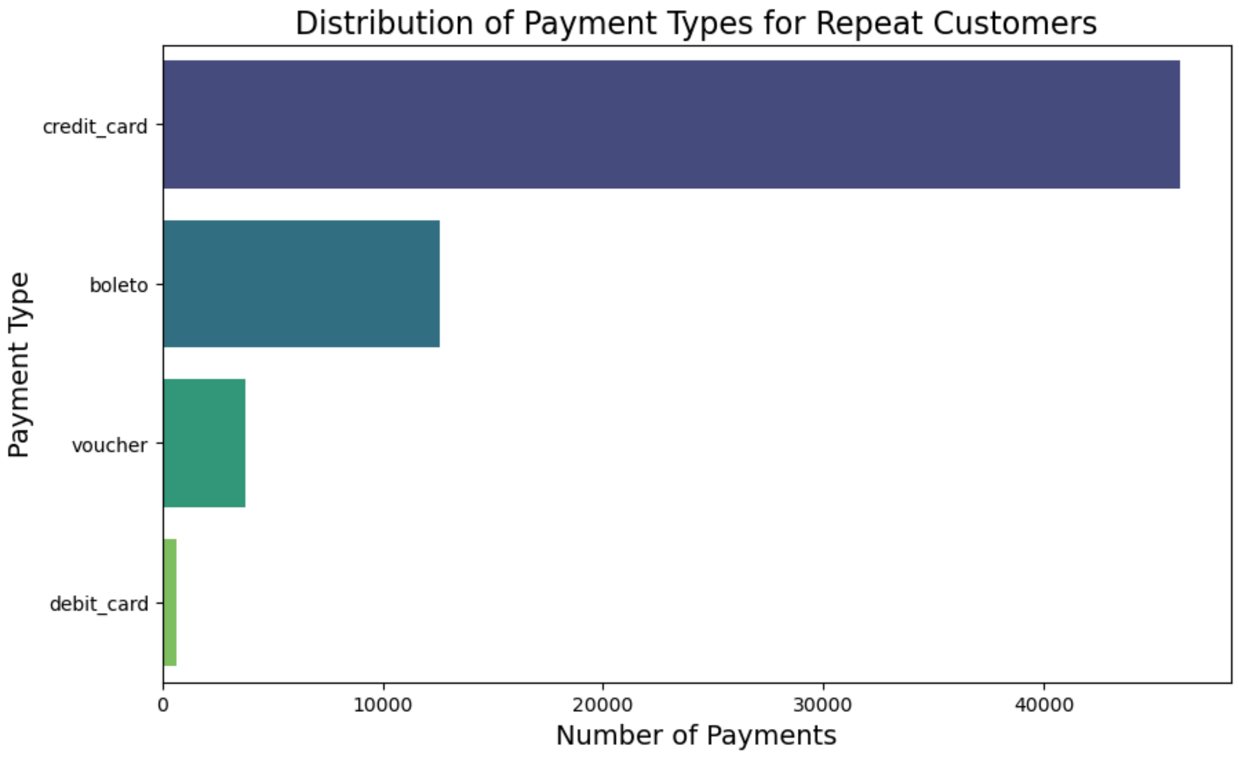
It is interesting to note that only movies_escritorio is found in both the most expensive items chart and the top categories sold chart. There are more categories that are found in both the least expensive chart and the top categories sold chart. This indicates that price can actually be a factor in how the inactive customers spend their money, but not explain why they did not buy anything again.

Distribution of the prices of items bought by single purchase customers



The purpose of this plot is to tell the company the price range of the items they should promote. Since most of the inactive customers do not spend much as well, it would be ineffective to promote expensive items to them, for example.

Finding the payment types of inactive customers



The distribution is roughly the same as that for repeat customers. Thus, payment type may not be the factor causing people to leave.

Market Analysis

I will now conduct market analysis. This will be done on the whole dataset and not a subset of customers.

Investigating the revenue earned per week

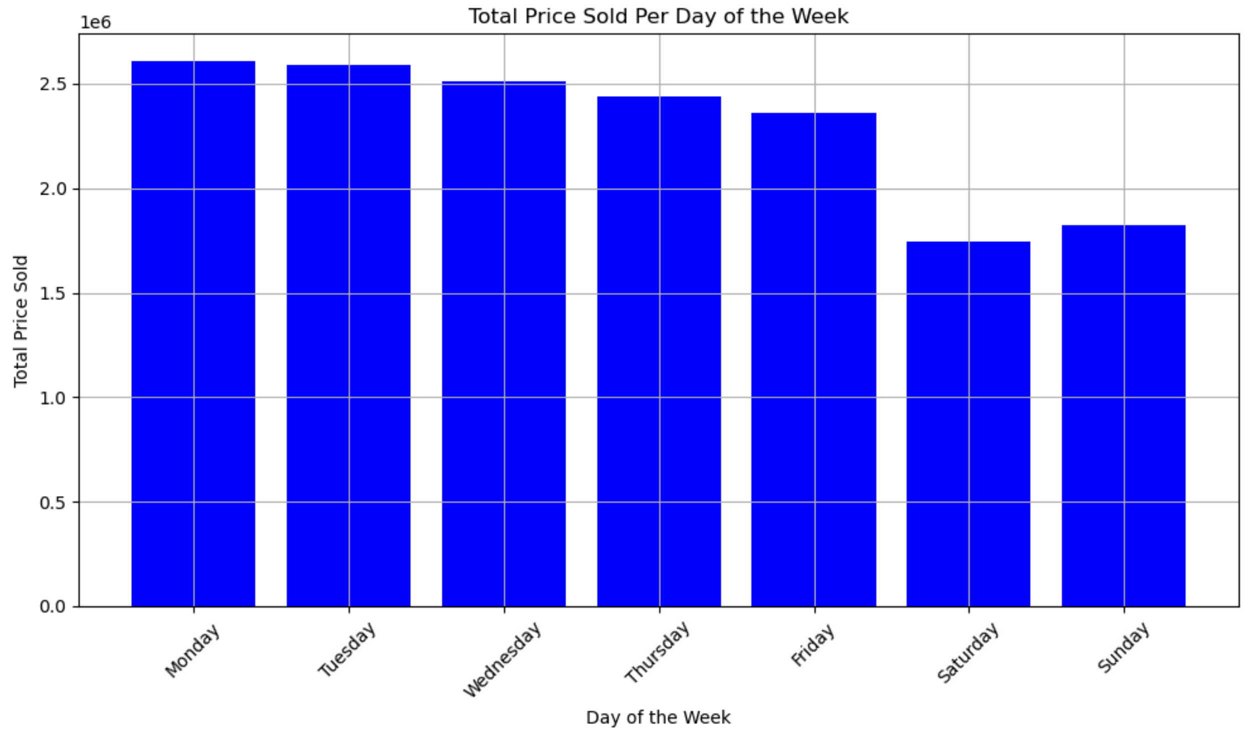
I am interested in finding out how much the company earns per day. The tables I will be using are customers, orders, order_items and order_payments. I also created a new column called revenue which is the price x order_item_id. I also added a new column to store the day of the week the order is placed

```
customers_orders_orderItems_orderPayments['revenue'] =
customers_orders_orderItems_orderPayments['price'] *
customers_orders_orderItems_orderPayments['order_item_id']
customers_orders_orderItems_orderPayments['order_purchase_timestamp']=
pd.to_datetime(customers_orders_orderItems_orderPayments['order_purchase_timestam
p'])

# Add a new column with the day of the week
customers_orders_orderItems_orderPayments['day_of_week'] =
customers_orders_orderItems_orderPayments['order_purchase_timestamp'].dt.day_name
()
```

Finding the distribution of revenue across the days of the week

```
# Ordering the days of the week (otherwise the plot will be confusing because the
days do not come out in order)
day_order = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday',
'Sunday']

# Convert day_of_week to a categorical column with a specific order
customers_orders_orderItems_orderPayments['day_of_week'] =
pd.Categorical(customers_orders_orderItems_orderPayments['day_of_week'],
categories=day_order, ordered=True)

# Groupby day_of_week and sum the revenue
total_sales_per_day =
customers_orders_orderItems_orderPayments.groupby('day_of_week')['revenue'].sum()
.reset_index()
```

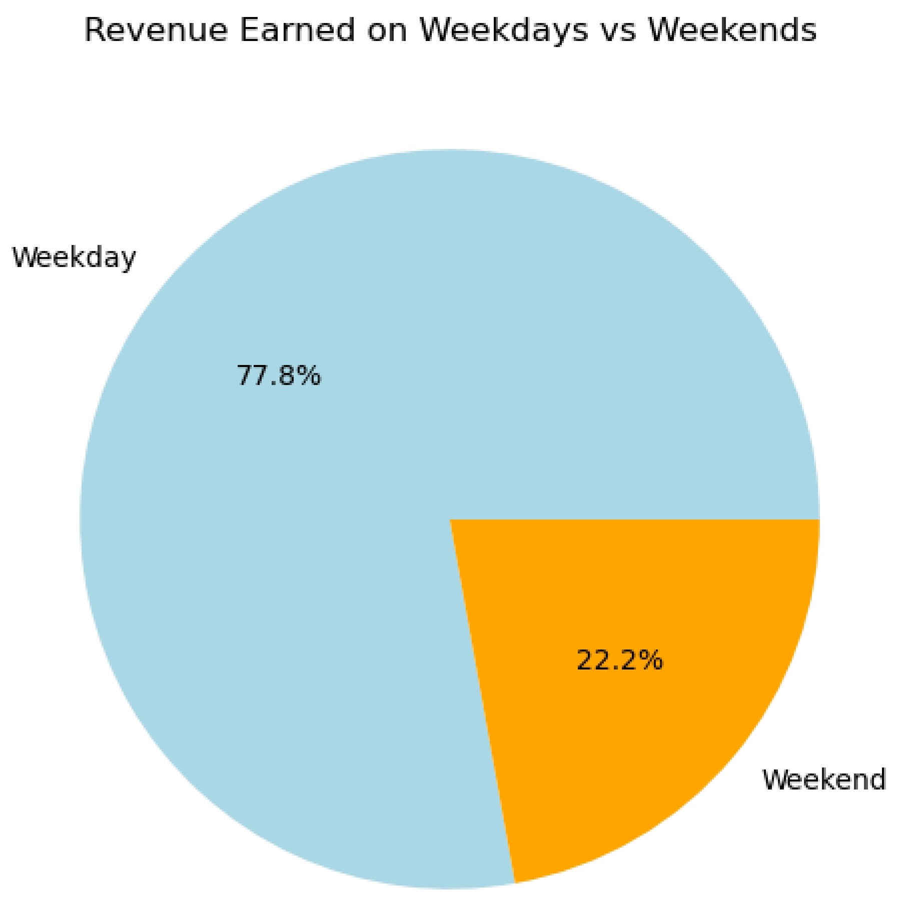
```
# Sort the dataframe
total_sales_per_day = total_sales_per_day.sort_values('day_of_week')

# Plotting
plt.figure(figsize=(10, 6))
plt.bar(total_sales_per_day['day_of_week'], total_sales_per_day['revenue'],
color='b')
plt.title('Total Price Sold Per Day of the Week')
plt.xlabel('Day of the Week')
plt.ylabel('Total Price Sold')
plt.xticks(rotation=45)
plt.grid(True)
plt.tight_layout()
plt.show()
```



Weekdays vs Weekends

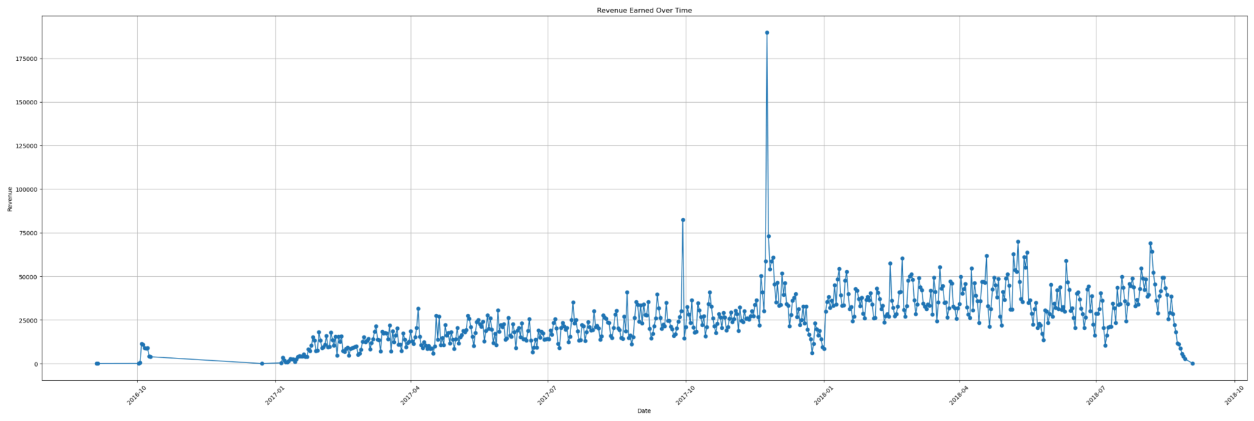
```
# Create a new column to classify weekend vs weekday
customers_orders_orderItems_orderPayments['weekend_or_weekday'] =
customers_orders_orderItems_orderPayments['day_of_week'].apply(lambda x:
'Weekend' if x in ["Saturday", 'Sunday'] else 'Weekday')

# Group by 'weekend_or_weekday' and sum the 'revenue'
```

```
revenue_by_day_type =
customers_orders_orderItems_orderPayments.groupby('weekend_or_weekday')['revenue'
].sum()

# Plot the pie chart
plt.figure(figsize=(6, 6))
plt.pie(revenue_by_day_type, labels=revenue_by_day_type.index, autopct='%1.1f%%',
colors=['lightblue', 'orange'])
plt.title('Revenue Earned on Weekdays vs Weekends')
plt.show()
```

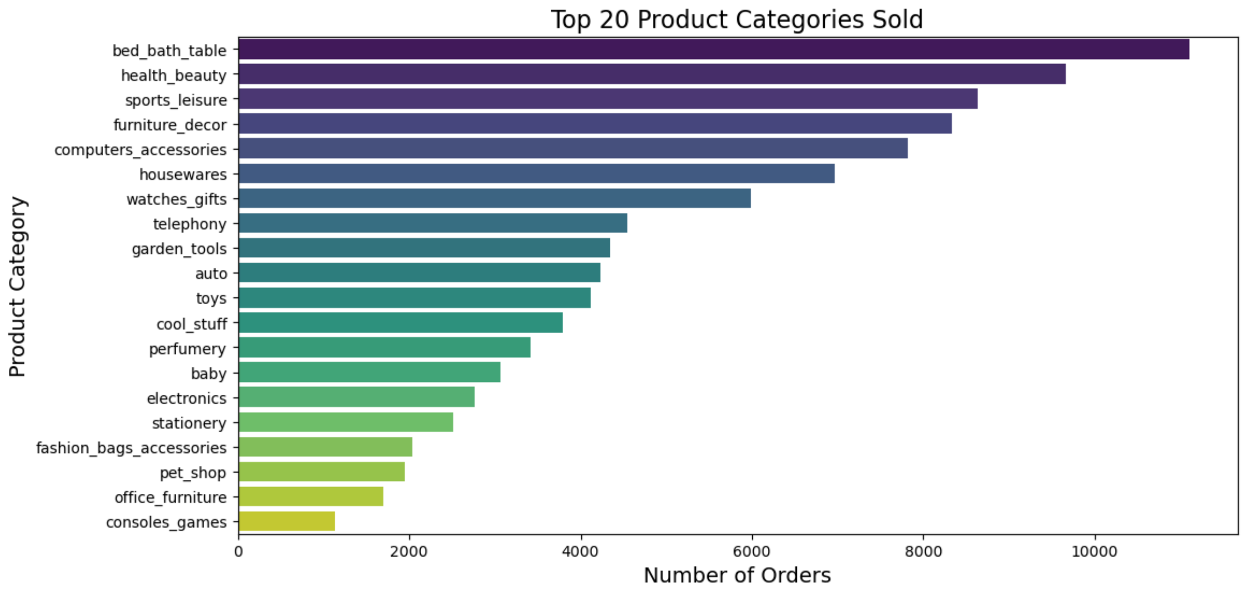


Most of the sales are conducted on weekdays. Thus the company can host more marketing campaigns on the weekdays. The company should ensure that the effects of the campaigns land on Mondays, as it is the day with the most revenue.

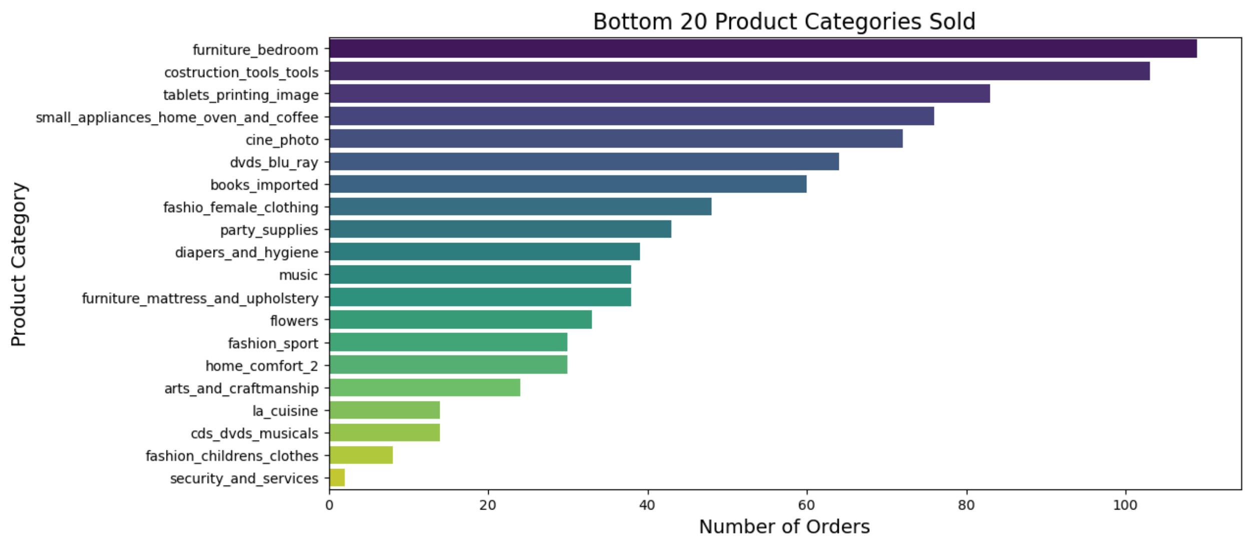
Revenue over time



Finding the top and bottom product categories sold



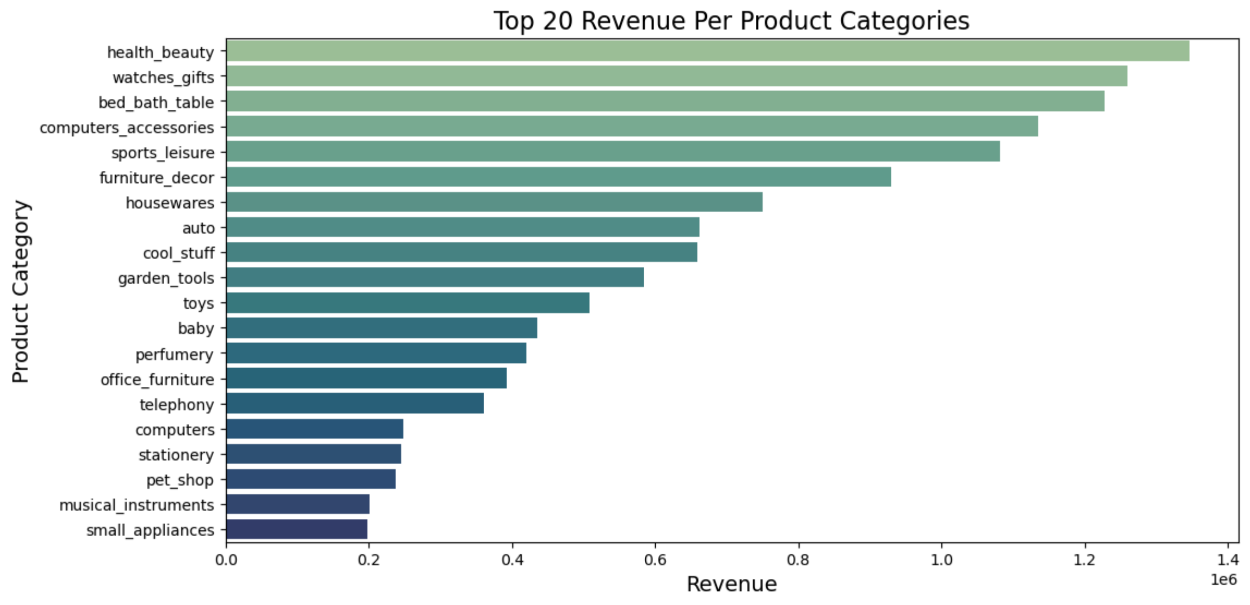
The most sold product category is 'cama_mesa_banho' with 11115 products sold.

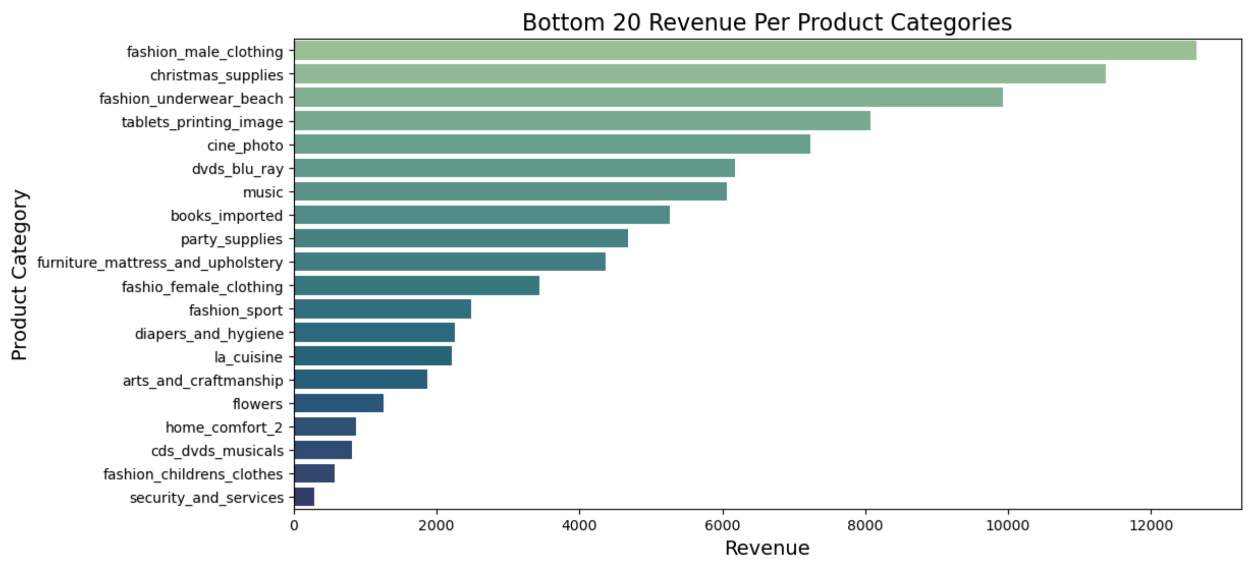


The least sold product category is 'seguros_e_servicos' with 2 products sold.

This will show the company how to allocate their resources towards the different product categories. However, the total number sold does not mean that the category produces the highest revenue. Hence, I also plotted the revenue earned by product category.

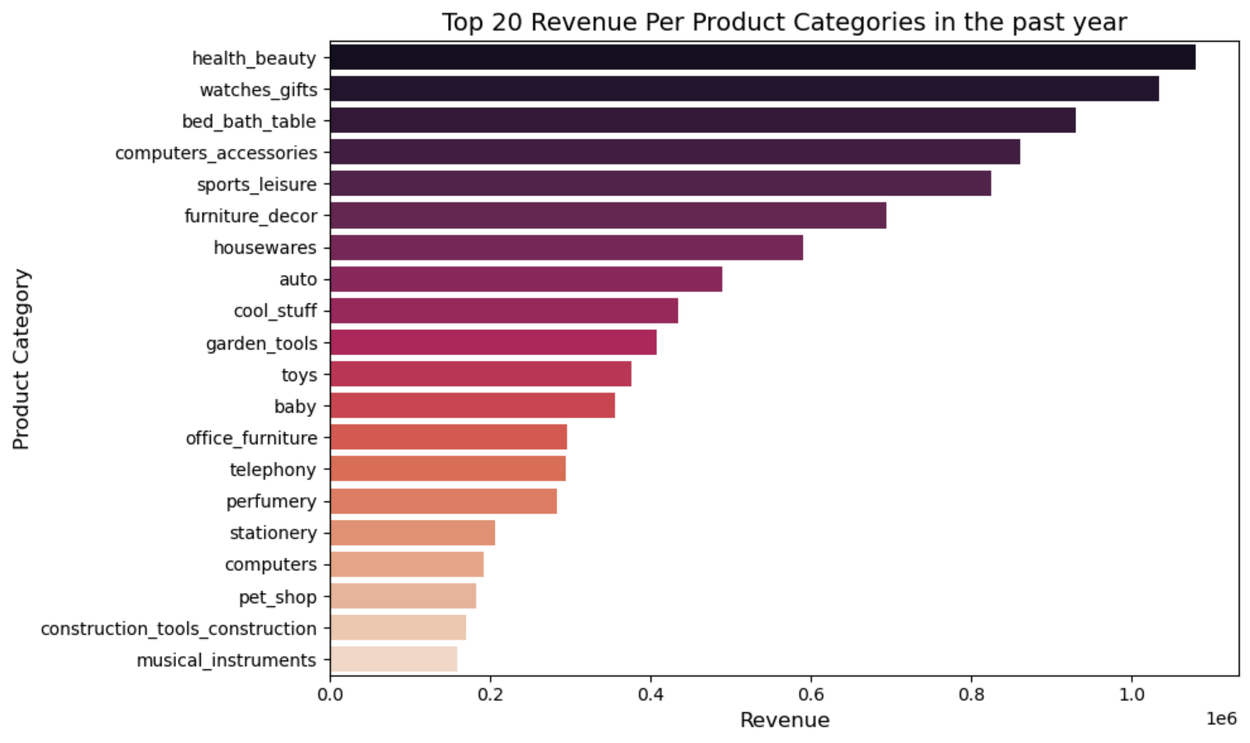
Finding the revenue earned by product category of all time

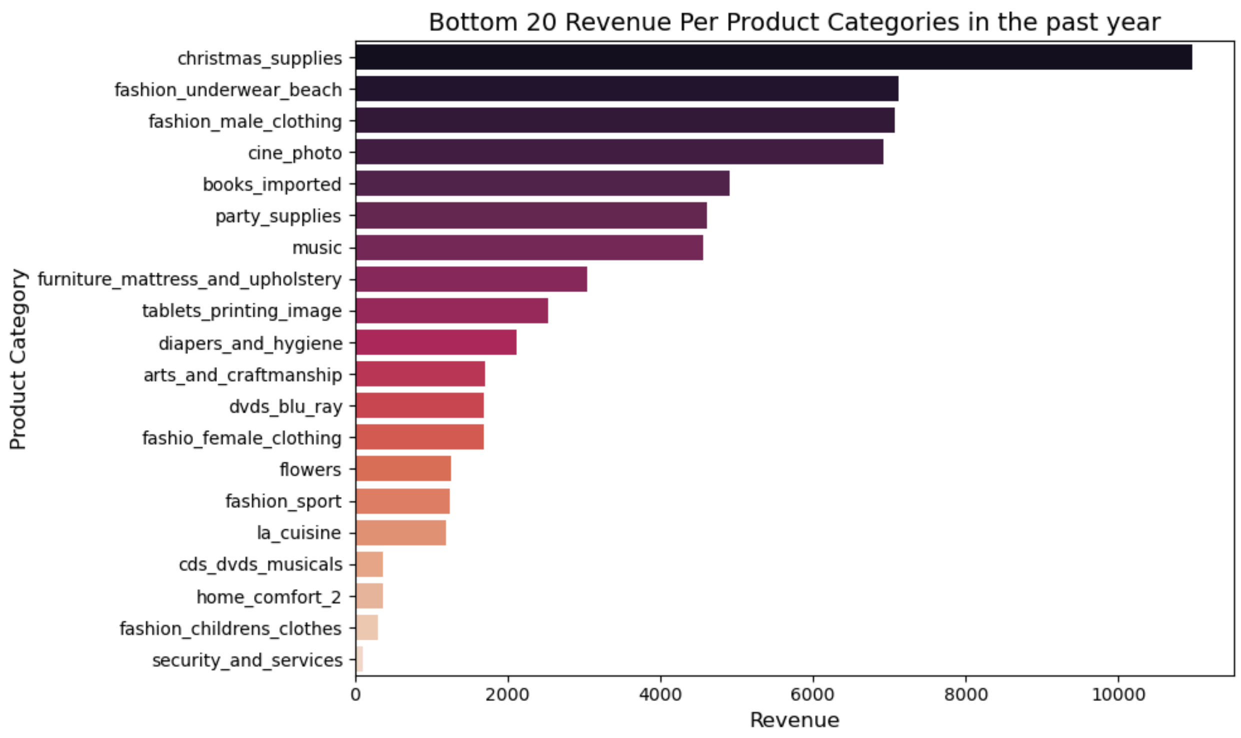




Finding the revenue per product category for the past year

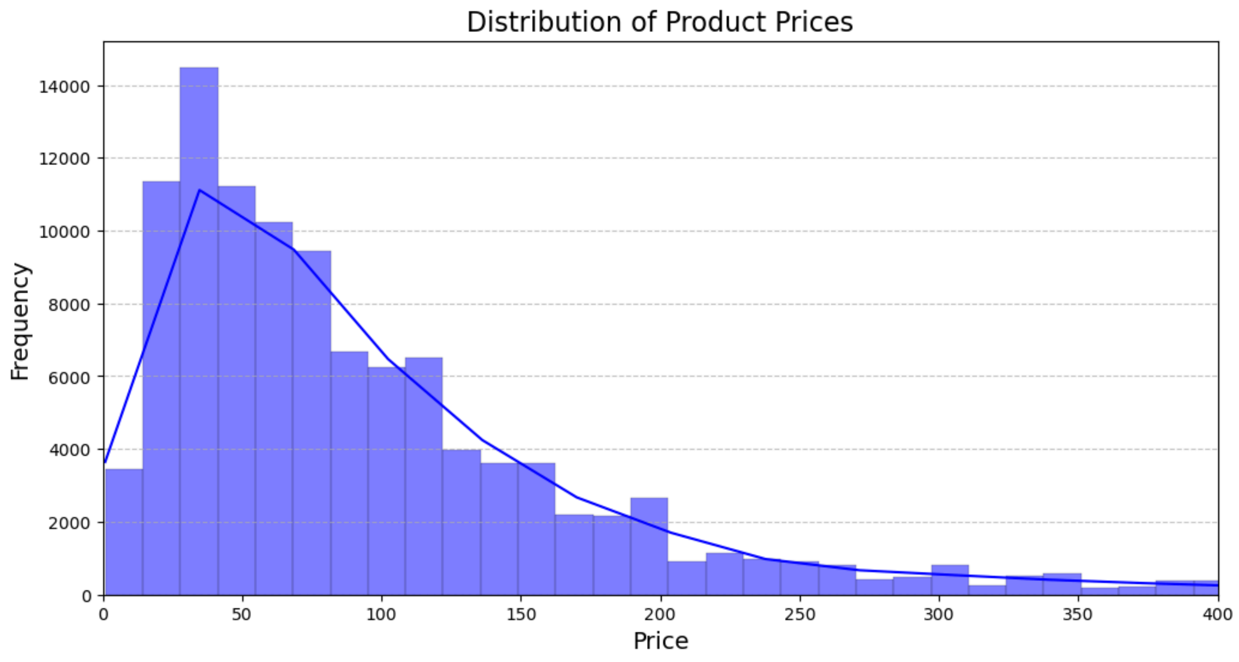
To do this, I first found the orders which were at most one year before the latest order placed in the dataset. Recall the latest order_purchase_timestamp is: 2018-09-03 09:06:57.

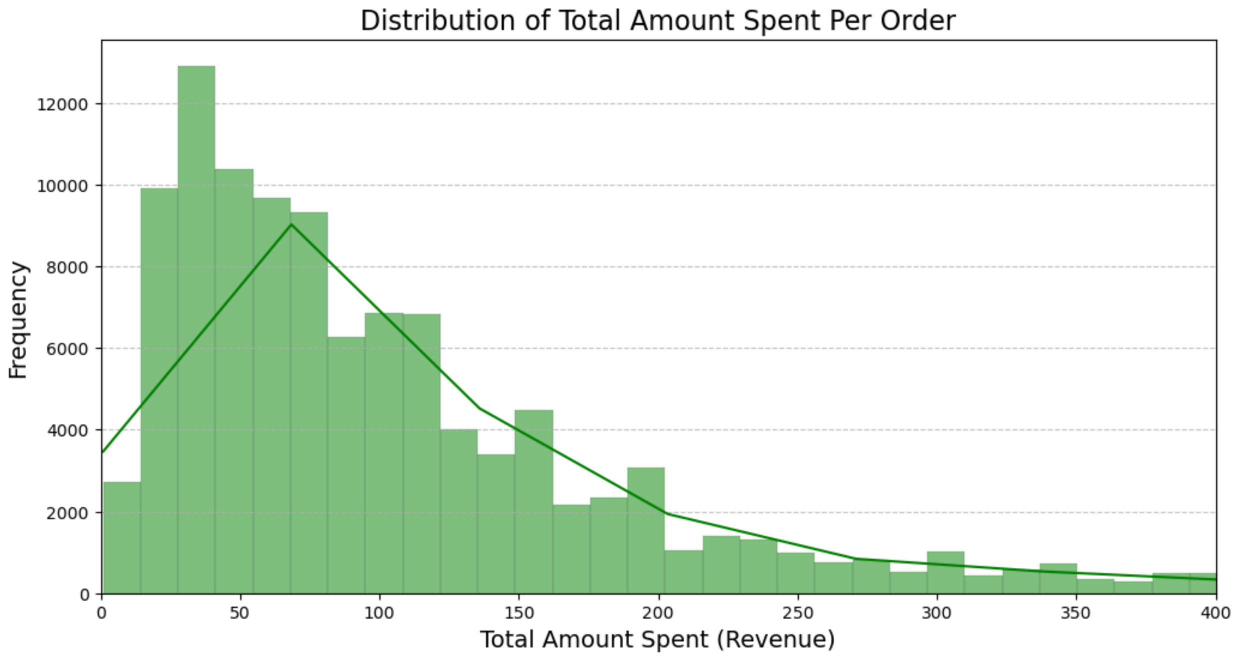




Again, we can see that the top categories sold usually bring in the highest revenue, be it of all time or over the past year. However, not necessarily in the same order. This could be due to the different prices of the products. This means the company can focus more on these products in order to bring in the most revenue.

Distribution of the prices of items bought and total amount spent per order by customers

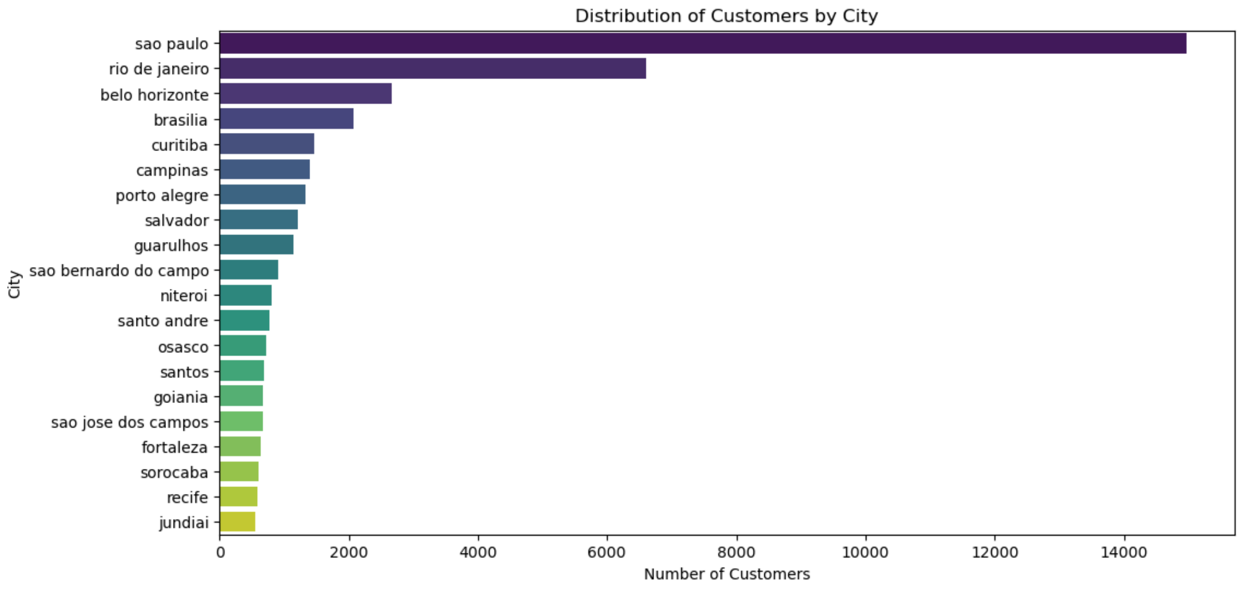




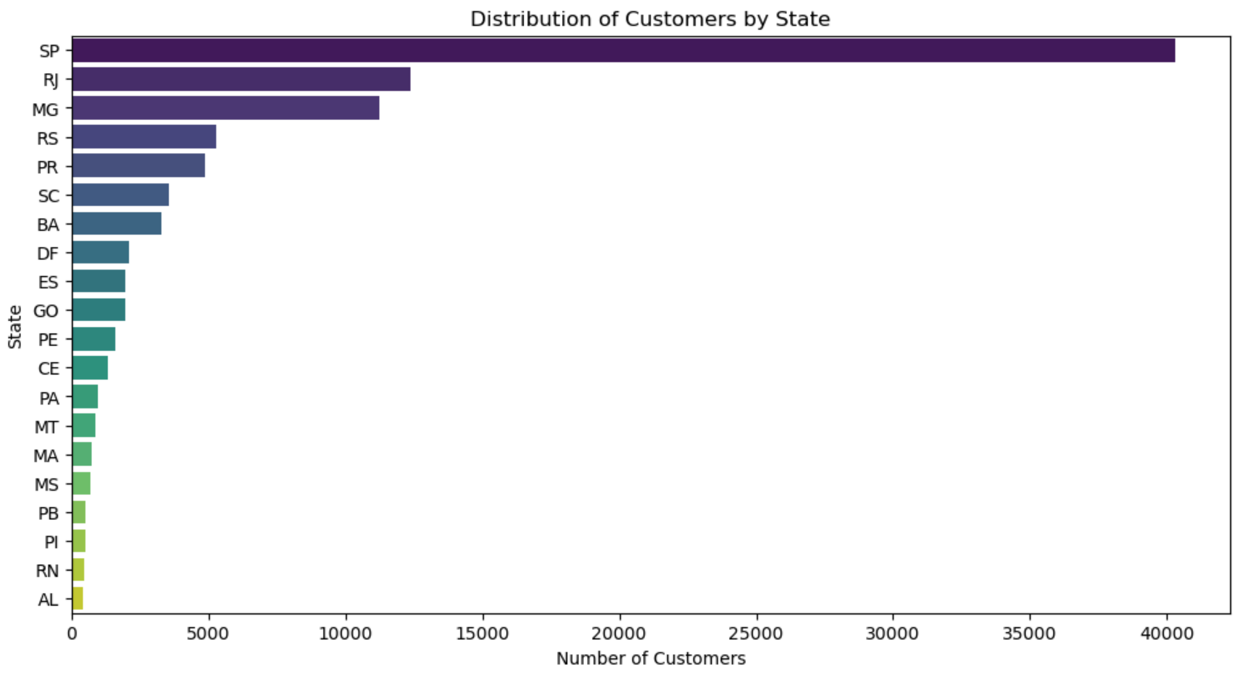
The shape of these plots are expected. However, this gives the company the price range of the products they should be selling and promoting.

Location of where most of the customers stay

Cities

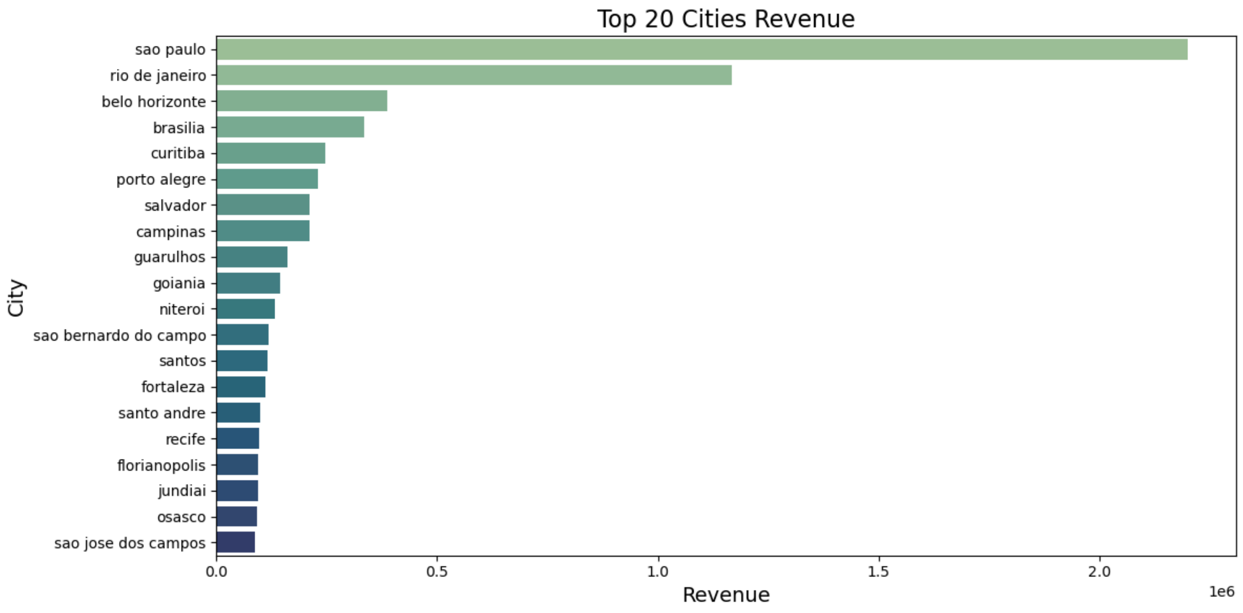


States

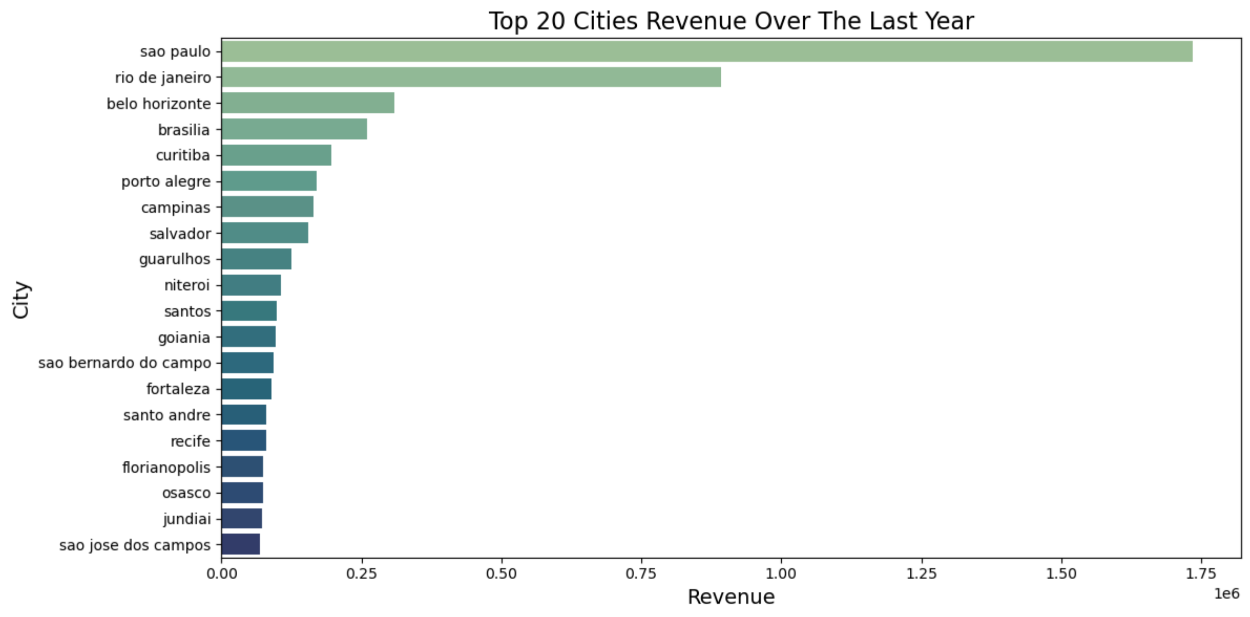


Finding the revenue earned by location

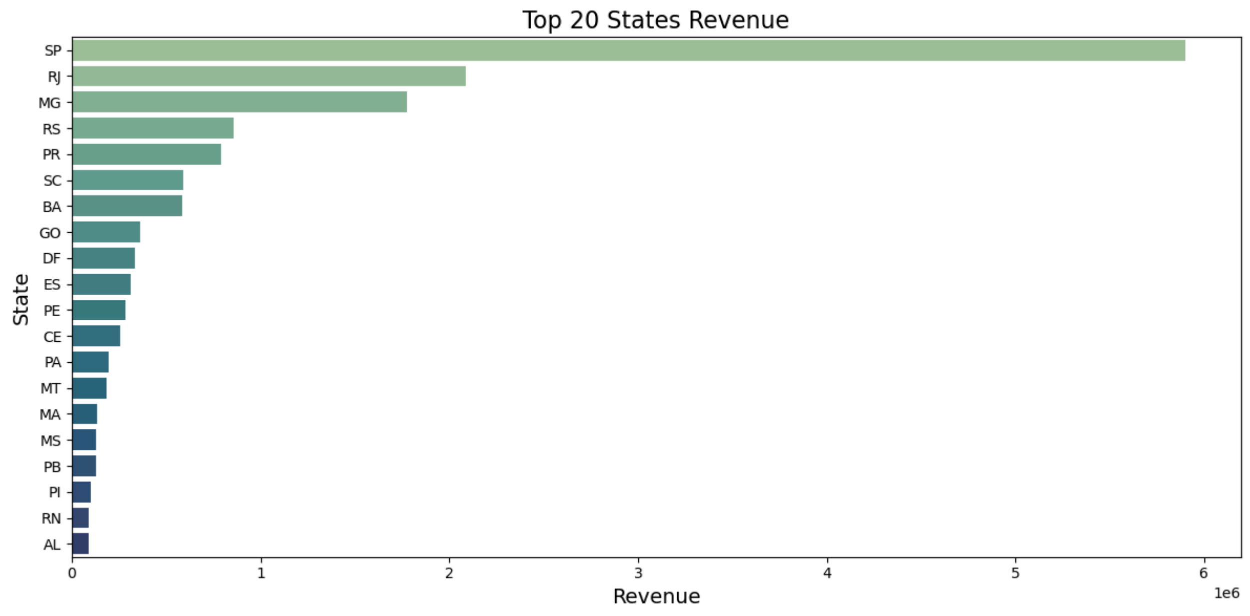
Cities



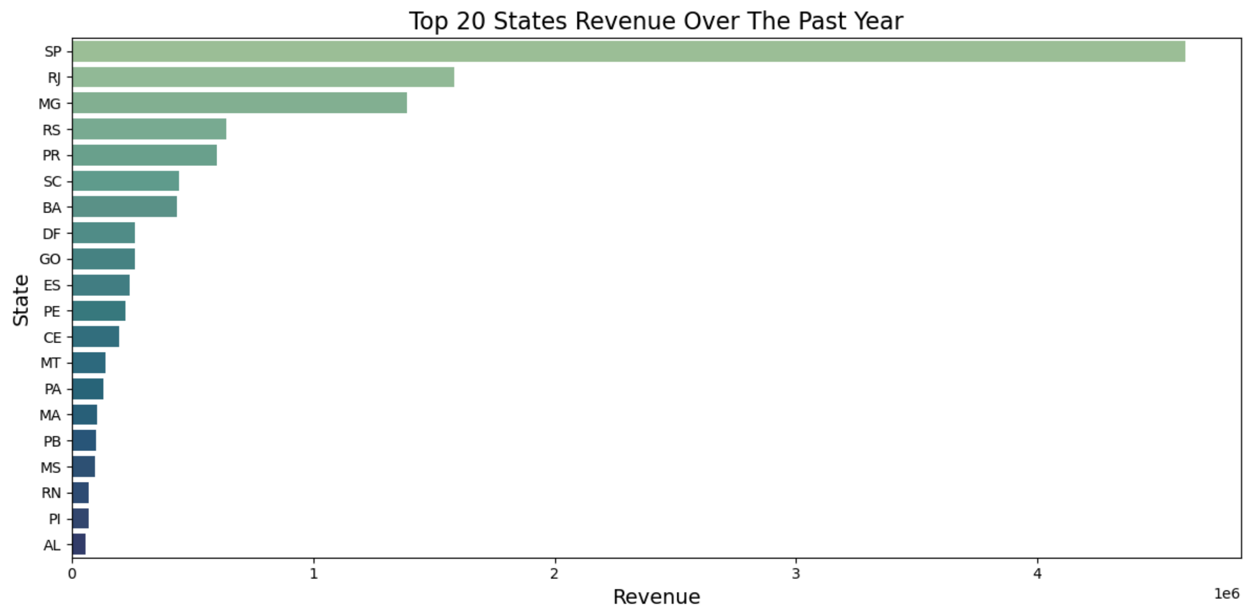
Cities over the last year



States

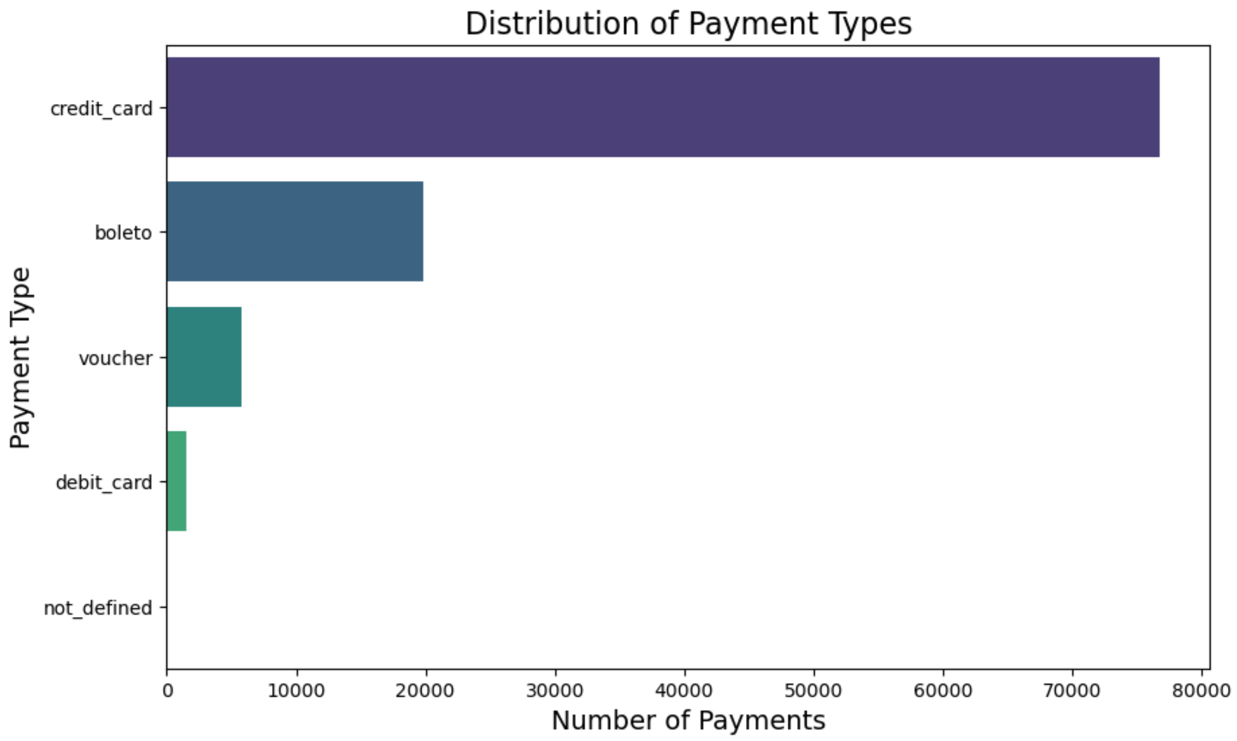


States over the past year



These carts tell the company where to focus their resources on. For the more updated data, the company should look at the past year charts.

Analysing the payment types

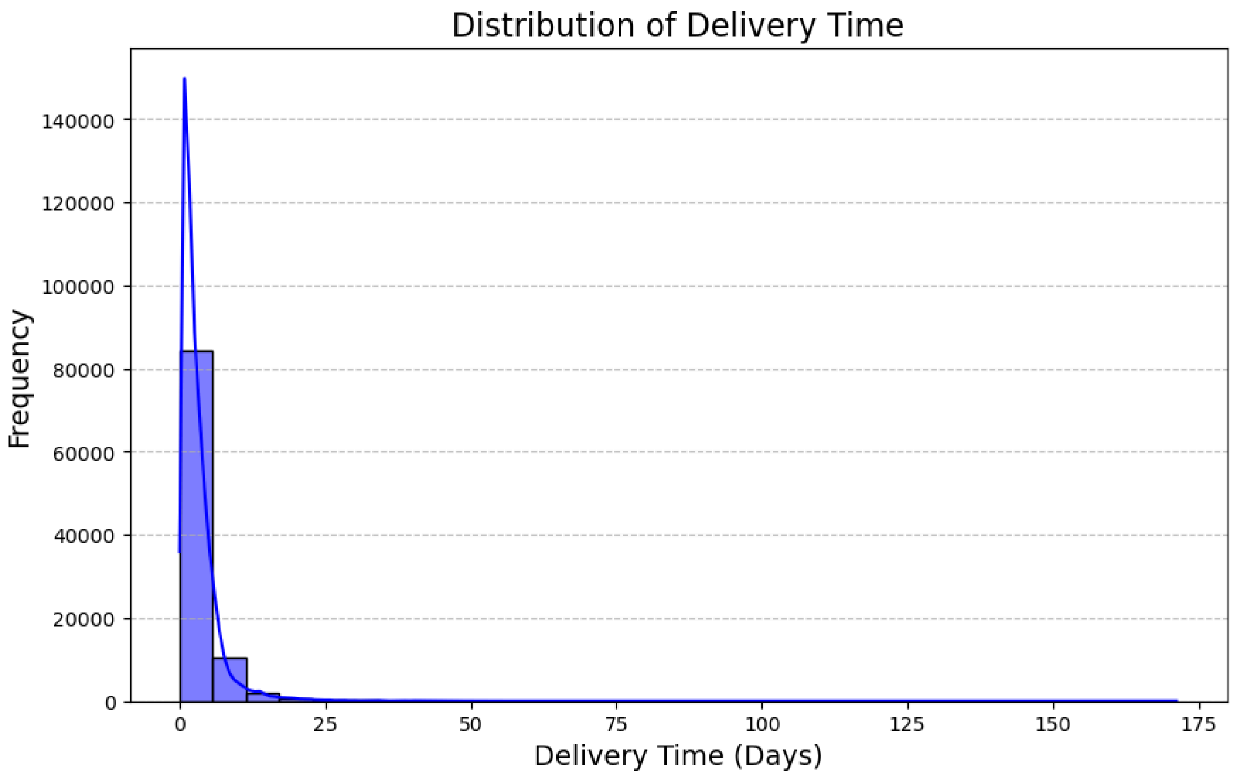


Since most customers use credit cards, the company could conduct some promotions where when customers purchase using credit cards, the customer could get a discount. This can spur spending.

Analysing delivery times

Statistics calculated:

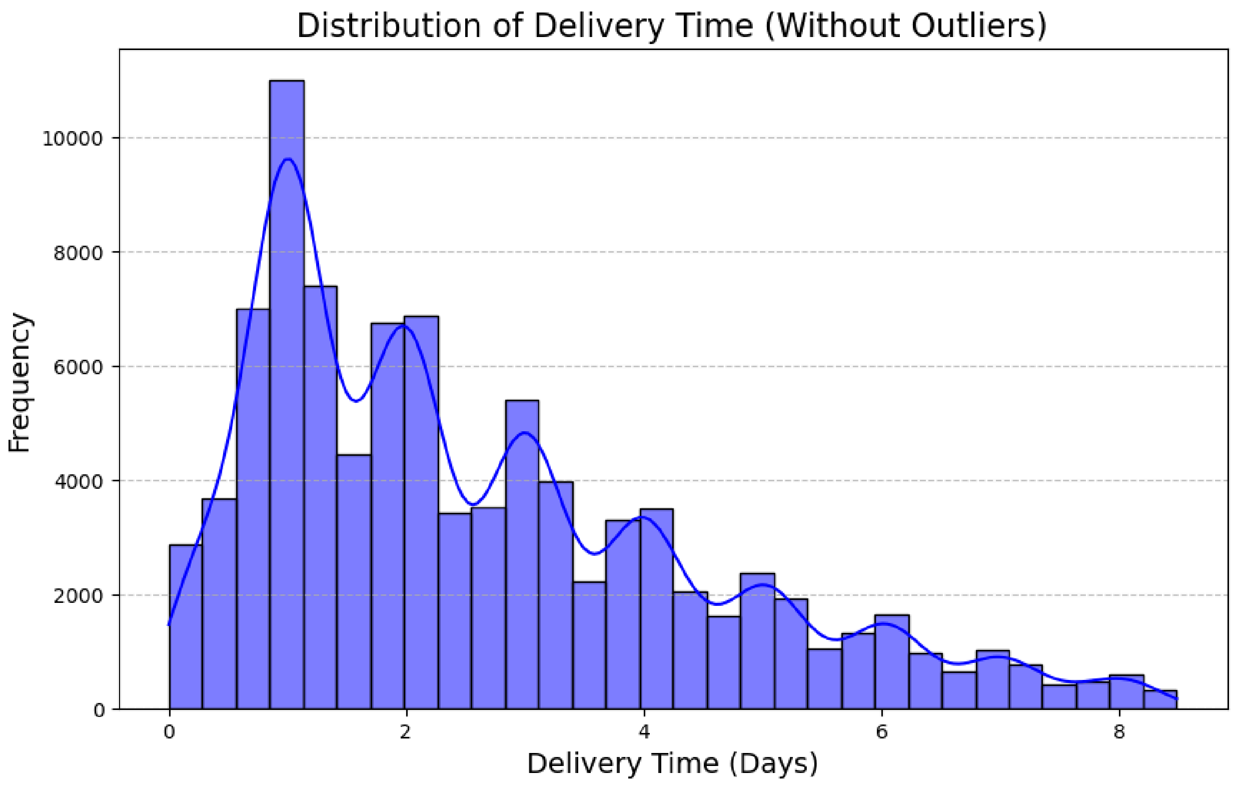
```
count    97658.000000
mean         3.237739
std          3.608690
min          0.000278
25%          1.129094
50%          2.204757
75%          4.071832
max        171.212419
```



Delivery times without outliers

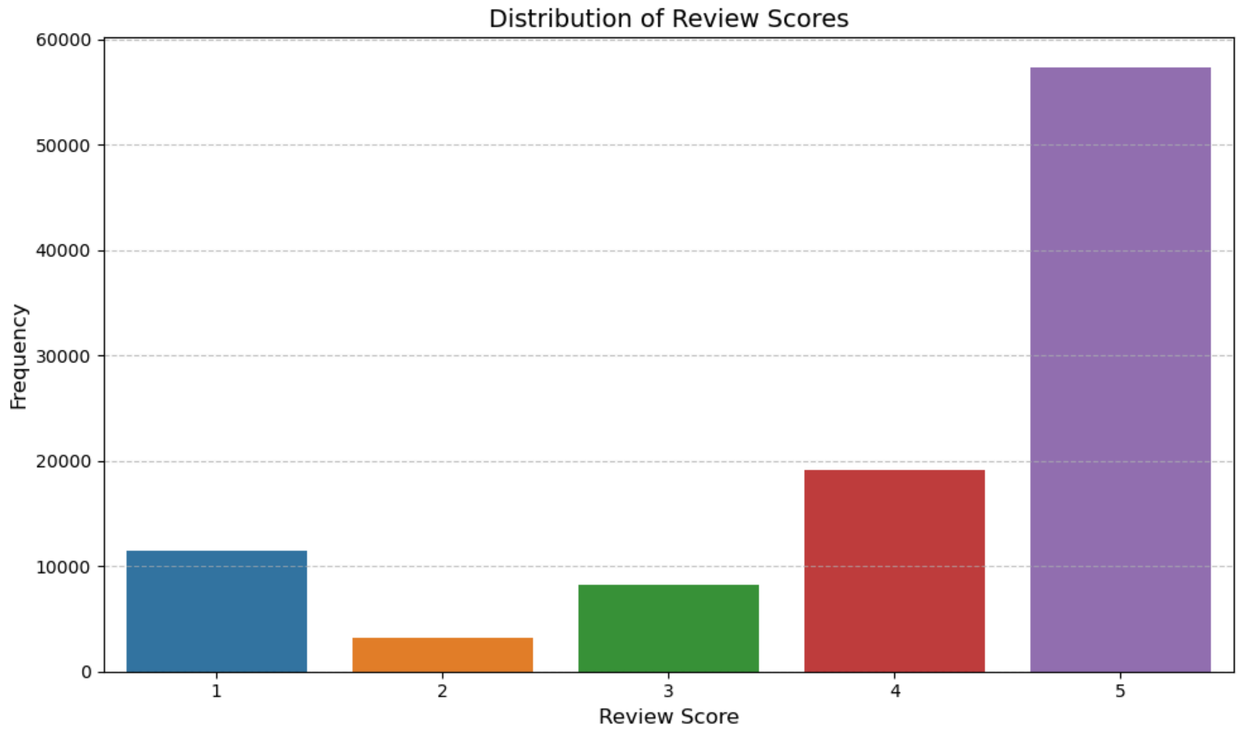
Statistics:

```
count    92377.000000
mean         2.611355
std          1.864104
min          0.000278
25%          1.095856
50%          2.094271
75%          3.762037
max          8.485440
```

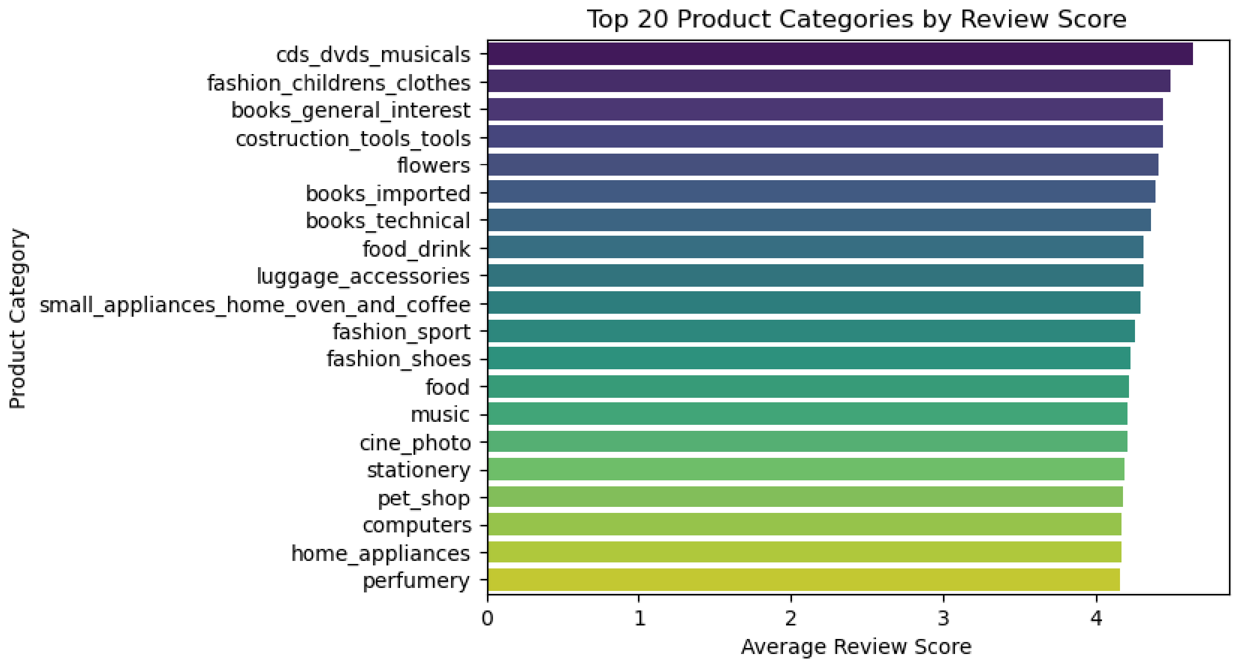


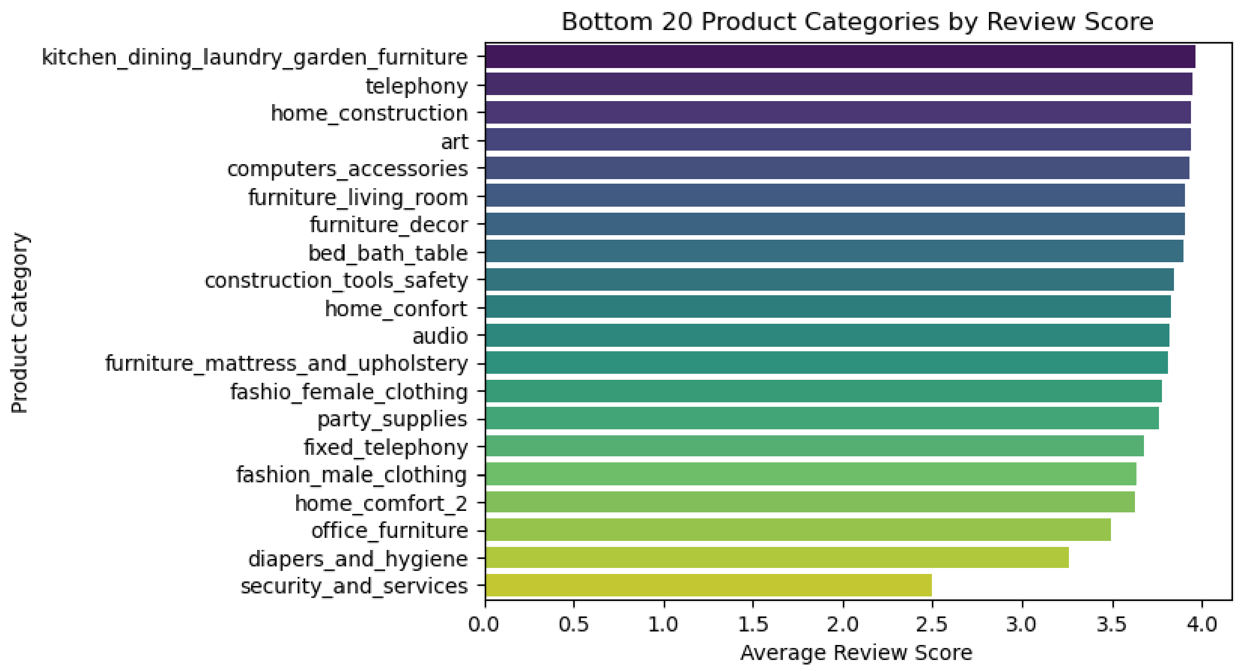
With a median of around 2 days, the company's delivery is very efficient. They should keep this up to retain the customers

Analysing reviews



Top and bottom categories by review score





Conclusions

There are certainly areas where the company has done well. The delivery times are generally very short, showing that they have an efficient delivery system. Their reviews are generally good as well, showing that the products sold are quite good and the customers are satisfied. Perhaps the company can consider dropping the products which did not sell much, as these products generally also come with average or bad reviews. An example would be 'security and services'. On the other hand, the company can focus more on the categories that sell well and have high reviews, like 'pet shop' and 'stationery'. The company should also promote products that are within the price range of the distribution of product prices plot. It is unlikely that customers will buy expensive items as well.

The company can also conduct more marketing campaigns at locations where most of their customers stay and the locations where the most revenue was generated.
That being said, the company should plan their campaigns such that the estimated time when the campaigns take effect happens on a weekday, especially on Monday. This is because they earn the most revenue on Mondays.
The company should conduct promotions for customers using credit cards. Since most customers use credit cards to pay, the company could provide incentives like discounts for customers who pay by credit cards, and this could spur increased spending.

To increase customer retention, the company can carry out more promotions for the product categories the repeating customers like. These campaigns can take place in the cities where most of the repeating customers stay as well. One important note the company should investigate is the outliers in the delivery times. It is unacceptable to have a delivery time of over 171 days where most of the time it takes only 4 days. Perhaps the data was an error made by the carrier. The company should cross check the top categories which generated the most revenue and the top categories sold by both repeating customers and single purchase customers. Identify which categories are present in both charts and promote them. This will increase the chances of the single purchase customers to buy again. For example, ‘bed bath table’, ‘health beauty’ and ‘sports leisure’ are categories that are worth a look.

Possible Improvements

Create function definitions to streamline the plotting process as some of the analysis were repeated.

Extra Resources Used

Cohort Analysis

https://www.youtube.com/watch?v=WWUG7T9ixTs

Sentiment Analysis

https://www.youtube.com/watch?v=tXuvh5_Xyrw
https://www.youtube.com/watch?v=szczpgOEdXs
https://www.youtube.com/watch?v=szczpgOEdXs

Word Cloud

https://www.youtube.com/watch?v=HcKUU5nNmrs
https://www.youtube.com/watch?v=szczpgOEdXs