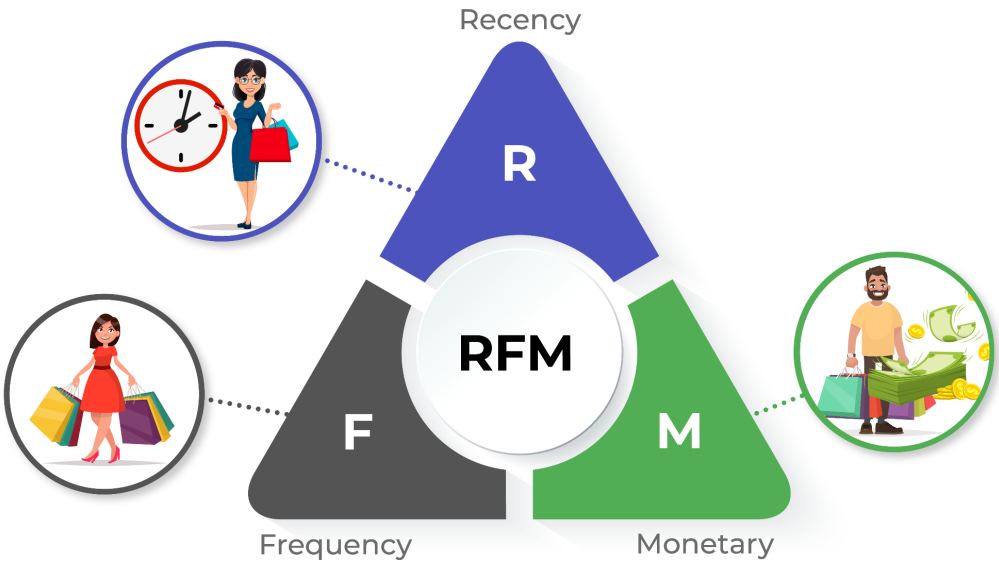
Customer segmentation using

RFM Analysis



IE6400 – FOUNDATIONS OF DATA ANALYTICS

Project Report 2

Group Number 17

Janani Karthikeyan (002830003)

Milan Gurumurthy (002833029)

Prathyusha Adhikam (002835277)

Saathvika Kethineni (002893814)

Shreyas Sreenivas (002825934)

Submitted to: Sivarit Sultornsanee

Submitted Date: December 1st, 2023

Contents:

Part 1:

Introduction:

In this project, we delve into the vast world of customer behavior in the hopes of discovering valuable insights that can transform marketing and customer engagement strategies. Our focal point is the rigorous application of the RFM (Recency, Frequency, Monetary) analysis method – an established framework empowering businesses to discern and categorize customers based on their recent purchasing patterns, transaction frequency, and monetary contributions.

Objective:

Our goal is simple: run RFM analysis on the dataset to generate a Customer Segmentation model. Recency, Frequency, and Monetary (RFM) are three key pillars that show various aspects of customer behavior. Recency is concerned with how recently a customer made a purchase, Frequency is concerned with the pattern of purchases, and Monetary is concerned with the monetary value of these transactions.

We intend to create distinct customer segments by leveraging RFM scores. These segments will not only provide a thorough understanding of customer purchasing patterns, but will also serve as a starting point for customized marketing and customer retention strategies. The ultimate goal is to provide businesses with the knowledge they need to better engage customers and drive long-term growth in the ever-changing eCommerce landscape.

Data Sources:

The dataset we are working with provides a snapshot of the eCommerce landscape, capturing the essence of customer transactions, preferences, and interactions.

Link to the dataset: https://www.kaggle.com/datasets/carrie1/ecommerce-data

Tasks performed:

Data Preprocessing:

- Import and Cleaning:

  Imported the dataset and executed essential data preprocessing steps, encompassing data cleaning, handling missing values, and converting data types as necessary.

In [17]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import datetime as dt
```

In [18]:
```
df = pd.read_csv(r"C:\Users\janan\Downloads\dataproj2.csv\data.csv",encoding = 'unicod
```

1. Data Overview

In [19]:
```
df.head()
```

Out[19]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	12/1/2010 8:26	2.55	17850.0	United Kingdom
1	536365	71053	WHITE METAL LANTERN	6	12/1/2010 8:26	3.39	17850.0	United Kingdom
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	12/1/2010 8:26	2.75	17850.0	United Kingdom
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	12/1/2010 8:26	3.39	17850.0	United Kingdom
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	12/1/2010 8:26	3.39	17850.0	United Kingdom

In [20]:
```
df.dropna(inplace=True)
df.isnull().sum()
```

Out[20]:
```
InvoiceNo      0
StockCode      0
Description    0
Quantity       0
InvoiceDate    0
UnitPrice      0
CustomerID     0
Country        0
dtype: int64
```

1. What is the size of the dataset in terms of the number of rows and columns?

In [21]:
```
print('The number of rows are:',df.shape[0])
print('The number of columns are:',df.shape[1])
```

```
The number of rows are: 406829
The number of columns are: 8
```

1. Can you provide a brief description of each column in the dataset?

In [22]:
```
df.dtypes
```

Out[22]:
```
InvoiceNo       object
StockCode       object
Description     object
Quantity         int64
InvoiceDate     object
UnitPrice      float64
CustomerID     float64
Country         object
dtype: object
```

In [23]:
```
df['InvoiceNo'].unique()
```

Out[23]:
```
array(['536365', '536366', '536367', ..., '581585', '581586', '581587'],
      dtype=object)
```

1. InvoiceNo: A six-digit number storing the details of the transaction. A cancellation if it begins with the letter "c".
2. StockCode: A code which defines the product which has been sold.
3. Description: Product name
4. Quantity: The quantities of each product per transaction
5. InvoiceDate: Shows the time and day that each transaction was created.
6. UnitPrice: Product price per unit.
7. CustomerID: A unique number designated to each customer.
8. Country: Name of the country where each customer resides.

1. What is the time period covered by this dataset?

In [24]:
```
df['DATE'] = pd.to_datetime(df['InvoiceDate'])
```

In [25]:
```
min_date = min(df['InvoiceDate'])
max_date = max(df['InvoiceDate'])
print("The dataset contains data starting from", min_date,'to',max_date)
```

```
The dataset contains data starting from 1/10/2011 10:32 to 9/9/2011 9:52
```

RFM Calculation:

- Recency, Frequency, Monetary Metrics:

  Calculated RFM metrics for each customer:

➢ Recency (R): Determined the days since the customer's last purchase.
➢ Frequency (F): Computed the total number of orders for each customer.
➢ Monetary (M): Summed up the total monetary value of a customer's purchases.

RFM Calculation:

```
# Step 2: Calculating Recency
current_date = max(df['DATE'])
recency = df.groupby('CustomerID')['DATE'].max().reset_index()
recency['Recency'] = (current_date - recency['DATE']).dt.days

# Step 3: Calculating Frequency
frequency = df.groupby('CustomerID')['DATE'].count().reset_index()
frequency.columns = ['CustomerID', 'Frequency']

# Step 4: Calculating Monetary
monetary = df.groupby('CustomerID')['Cost'].sum().reset_index()
monetary.columns = ['CustomerID', 'Monetary']

# Combining RFM Scores
rfm = recency.merge(frequency, on='CustomerID').merge(monetary, on='CustomerID')

# Displaying the RFM table
rfm.head()
```


IE6400

	CustomerID	DATE	Recency	Frequency	Monetary
0	12346.0	2011-01-18 10:17:00	325	2	0.00
1	12347.0	2011-12-07 15:52:00	1	182	4310.00
2	12348.0	2011-09-25 13:13:00	74	31	1797.24
3	12349.0	2011-11-21 09:51:00	18	73	1757.55
4	12350.0	2011-02-02 16:01:00	309	17	334.40

RFM segmentation:

- Assigning RFM Scores:

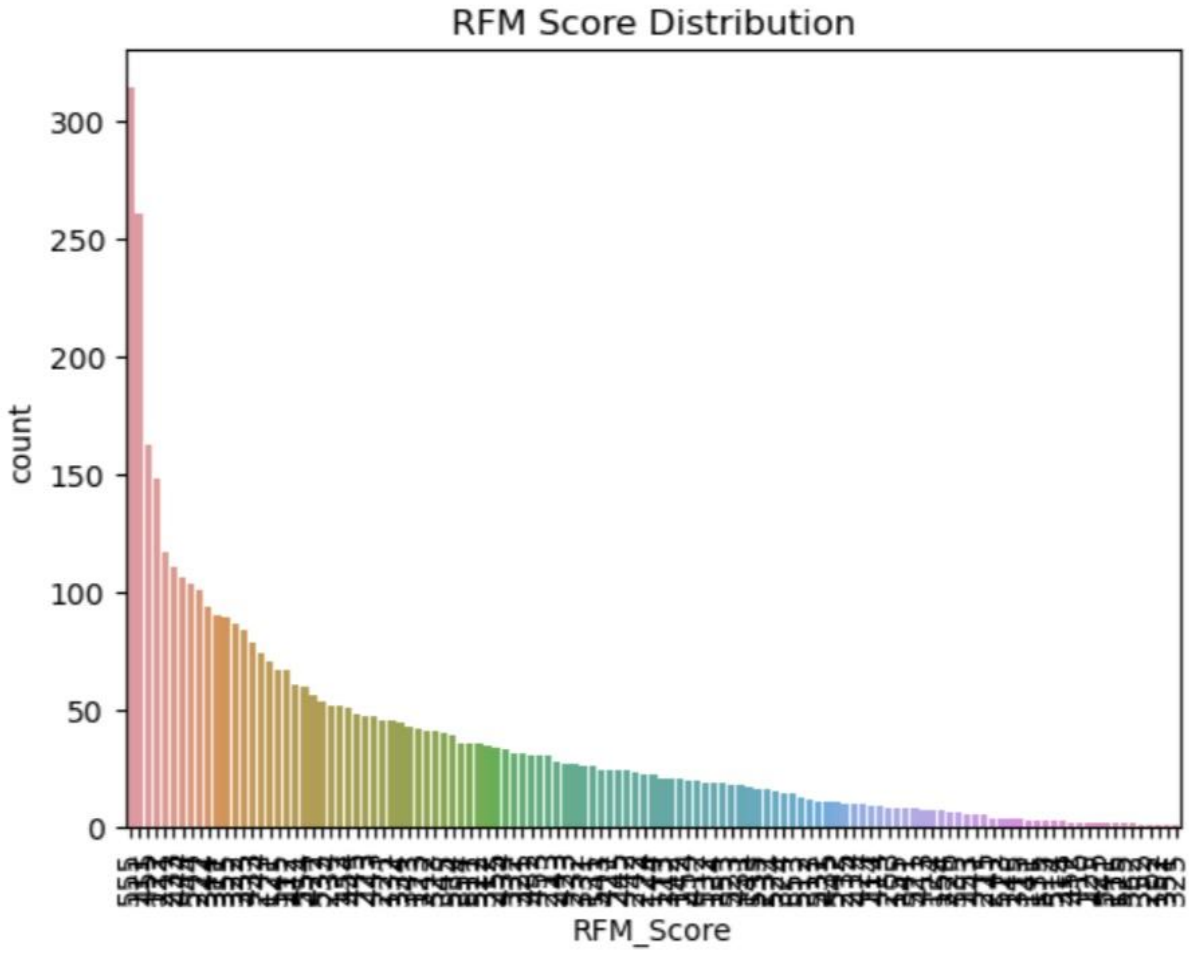
  Assigned RFM scores based on quartiles or custom-defined bins, facilitating the creation of a single RFM score for each customer.

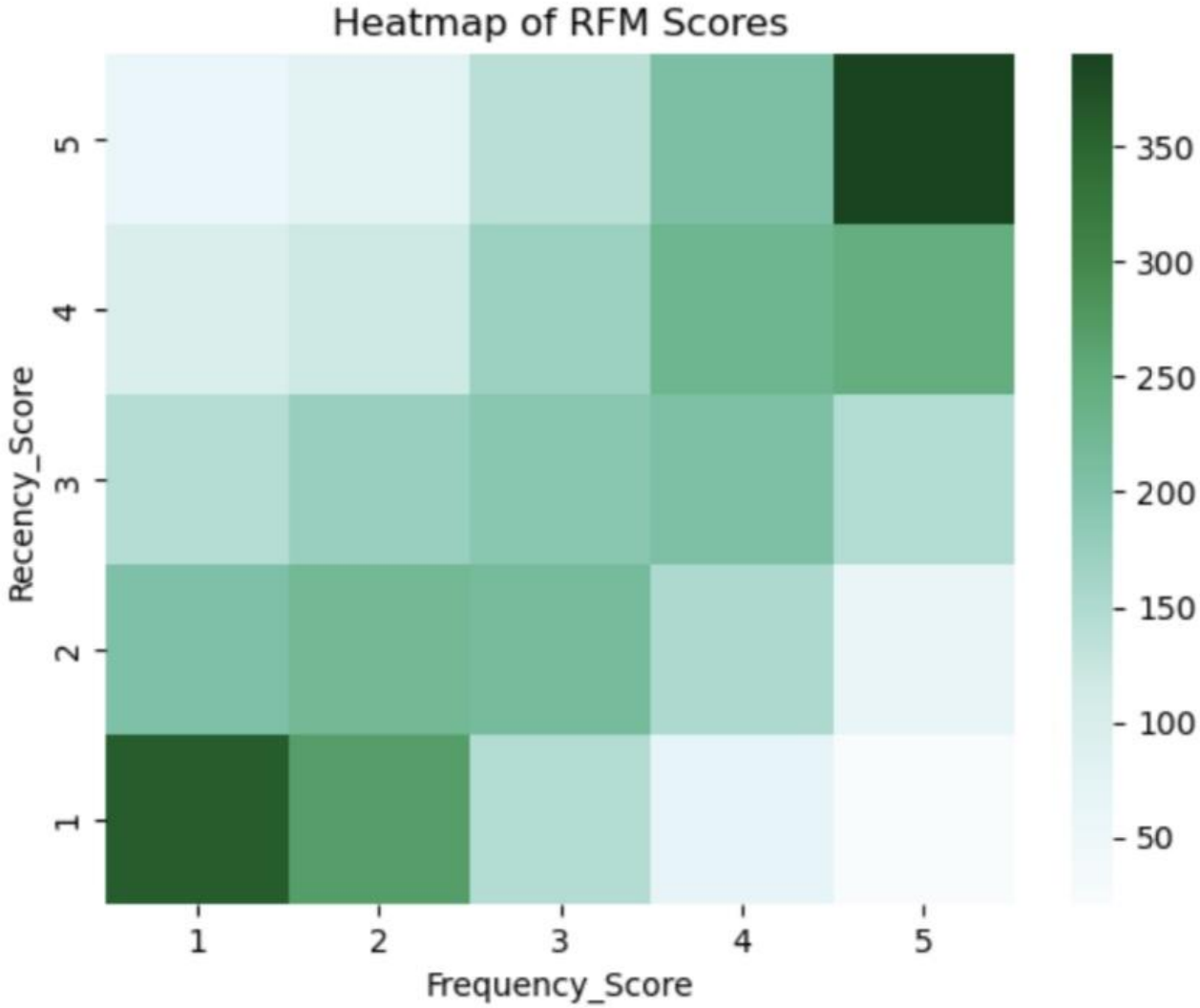
RFM Segmentation:

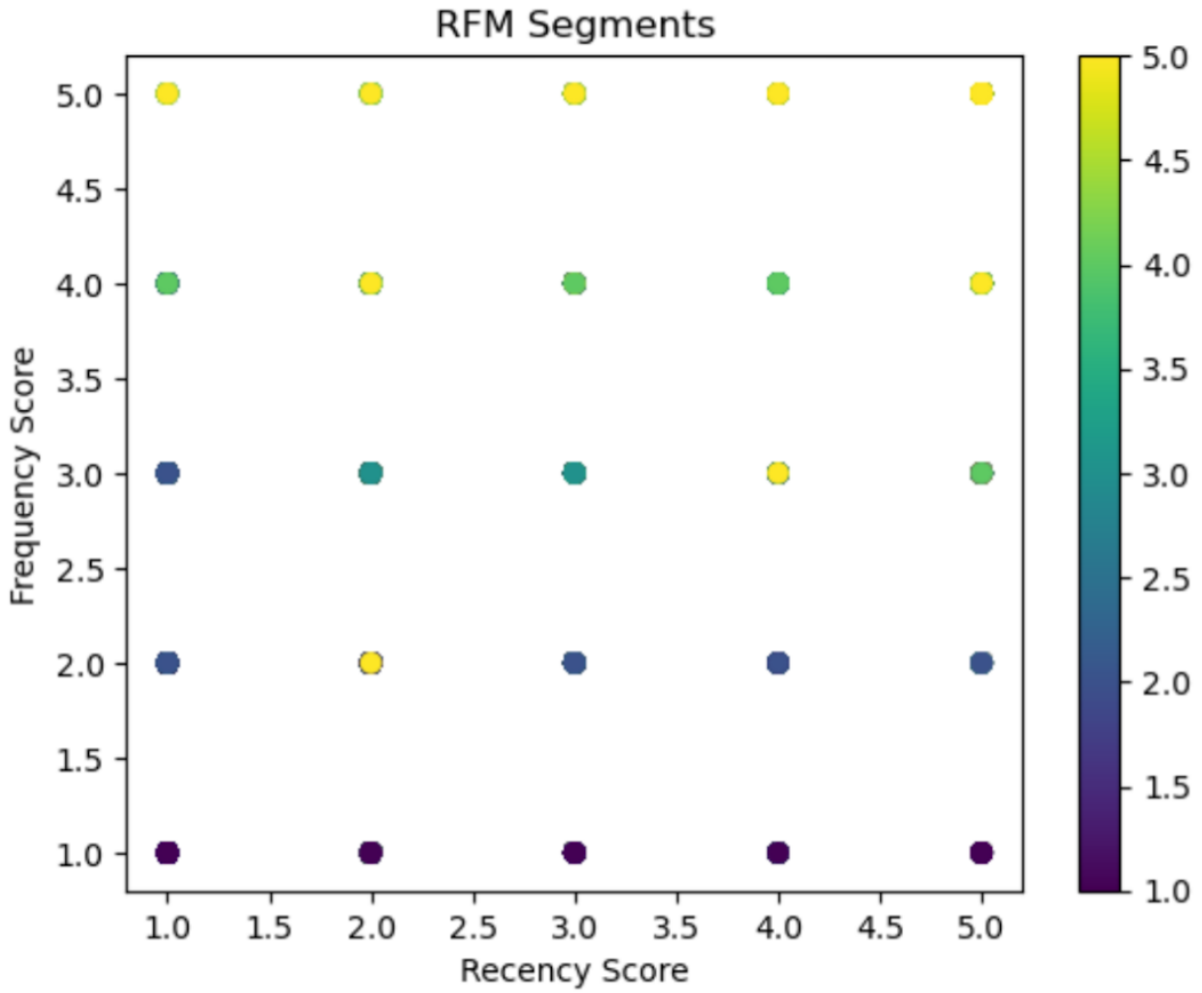
```
# recency score and monetary score  : 1=bad , 5 = good
# freq score : 5=bad , 1 = good

rfm["Recency_Score"]=pd.qcut(rfm["Recency"],5,labels=[5,4,3,2,1,])
rfm["Frequency_Score"]=pd.qcut(rfm["Frequency"].rank(method="first"),5,labels=[1,2,3,4
rfm["Monetary_Score"]=pd.qcut(rfm["Monetary"],5,labels=[1,2,3,4,5])
rfm["RFM_Score"]=(rfm["Recency_Score"].astype(str)+rfm["Frequency_Score"].astype(str)+
rfm=rfm.sort_values("CustomerID",ascending=True)
rfm.head()
```

	CustomerID	DATE	Recency	Frequency	Monetary	Recency_Score	Frequency_Score	Monetary_S
1	12347.0	2011-12-07 15:52:00	1	182	4310.00	5	5	
2	12348.0	2011-09-25 13:13:00	74	31	1797.24	2	3	
3	12349.0	2011-11-21 09:51:00	18	73	1757.55	4	4	
4	12350.0	2011-02-02 16:01:00	309	17	334.40	1	2	
5	12352.0	2011-11-03 14:37:00	35	95	1545.41	3	4	

In [46]:
```
import seaborn as sns
import matplotlib.pyplot as plt

# Assuming rfm is your precomputed DataFrame with 'RFM_Score' column
sns.countplot(x='RFM_Score', data=rfm, order=rfm['RFM_Score'].value_counts().index)
plt.title('RFM Score Distribution')
plt.xticks(rotation=90)
plt.show()
```


Heatmap of RFM Scores
Recency_Score
5
4
3
2
1
1
2
3
4
5
Frequency_Score
350
300
250
200
150
100
50



Customer Segmentation:

- Utilizing Clustering Techniques:

  Applied clustering techniques, such as K-Means clustering, to segment customers based on their RFM scores. Experimented with different cluster numbers to identify the optimal configuration for meaningful segmentation.

Segment Profiling:

- Analyzing Customer Segments:

  Analyzed and profiled each customer segment, elucidating the characteristics of customers within each segment, including RFM scores and other relevant attributes.

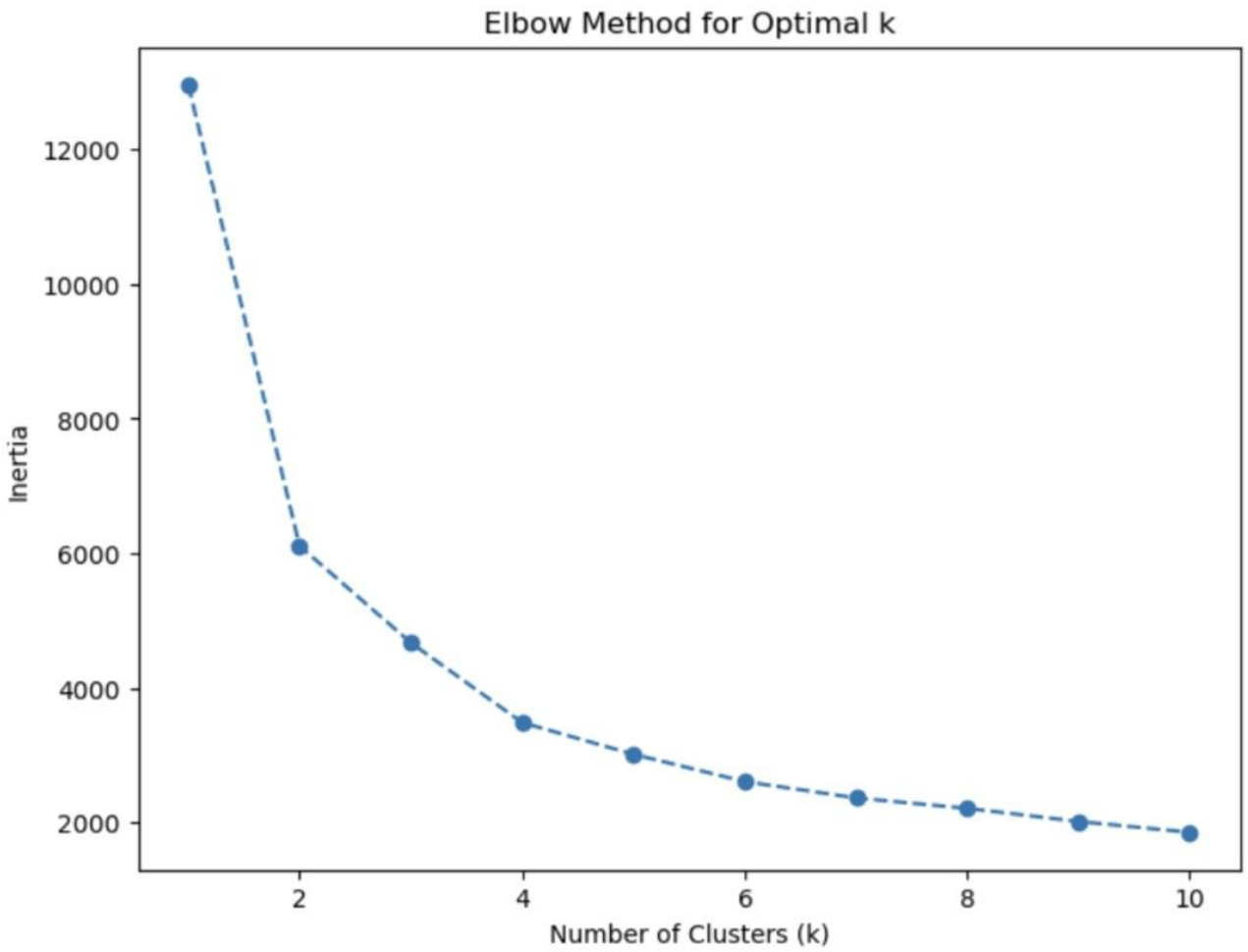
Visualization:

- RFM Data Visualization:

  Developed visualizations such as bar charts, scatter plots, and heat maps to illustrate the distribution of RFM scores and visualize the formed clusters.

4. Customer Segmentation:

```
import warnings
warnings.filterwarnings('ignore')

from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
rfm_data = rfm[['Recency_Score', 'Frequency_Score', 'Monetary_Score']]

# Standardize the data (important for K-Means)
scaler = StandardScaler()
rfm_scaled = scaler.fit_transform(rfm_data)

# Determine the optimal number of clusters using the Elbow Method
inertia = []
for k in range(1, 11):
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(rfm_scaled)
    inertia.append(kmeans.inertia_)

# Plot the Elbow Method to find the optimal number of clusters
plt.figure(figsize=(8, 6))
plt.plot(range(1, 11), inertia, marker='o', linestyle='--')
plt.title('Elbow Method for Optimal k')
plt.xlabel('Number of Clusters (k)')
plt.ylabel('Inertia')
plt.show()
```



30]:
```
#optimal number of cluster is 4 according to the elbow method
```

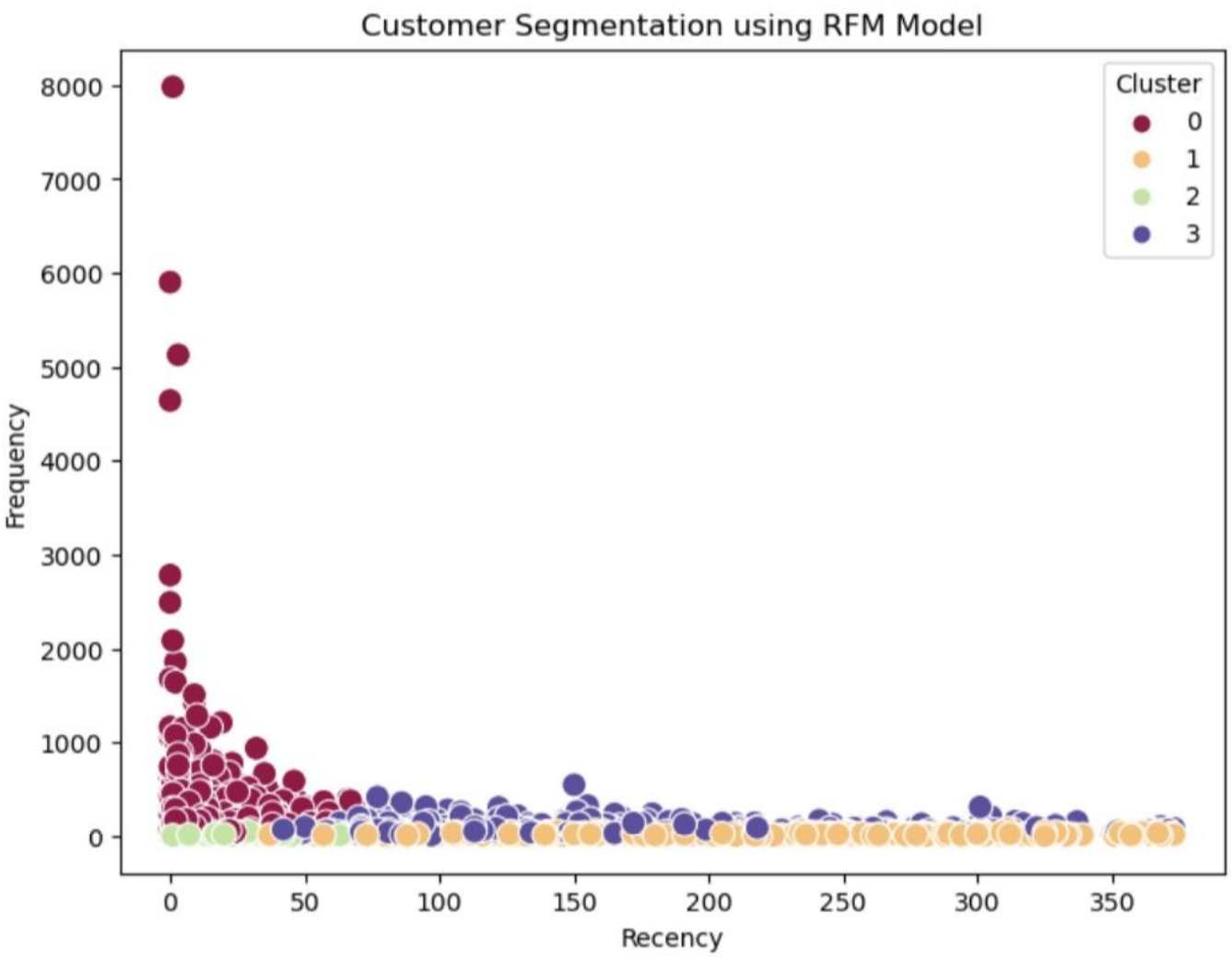
32]:
```
# Applying K-means with the optimal number of clusters
optimal_k = 4
kmeans = KMeans(n_clusters=optimal_k, init='k-means++',n_init=10, random_state=42)
rfm['Cluster'] = kmeans.fit_predict(rfm_scaled)

# Displaying the RFM table with clusters
rfm.head()
```

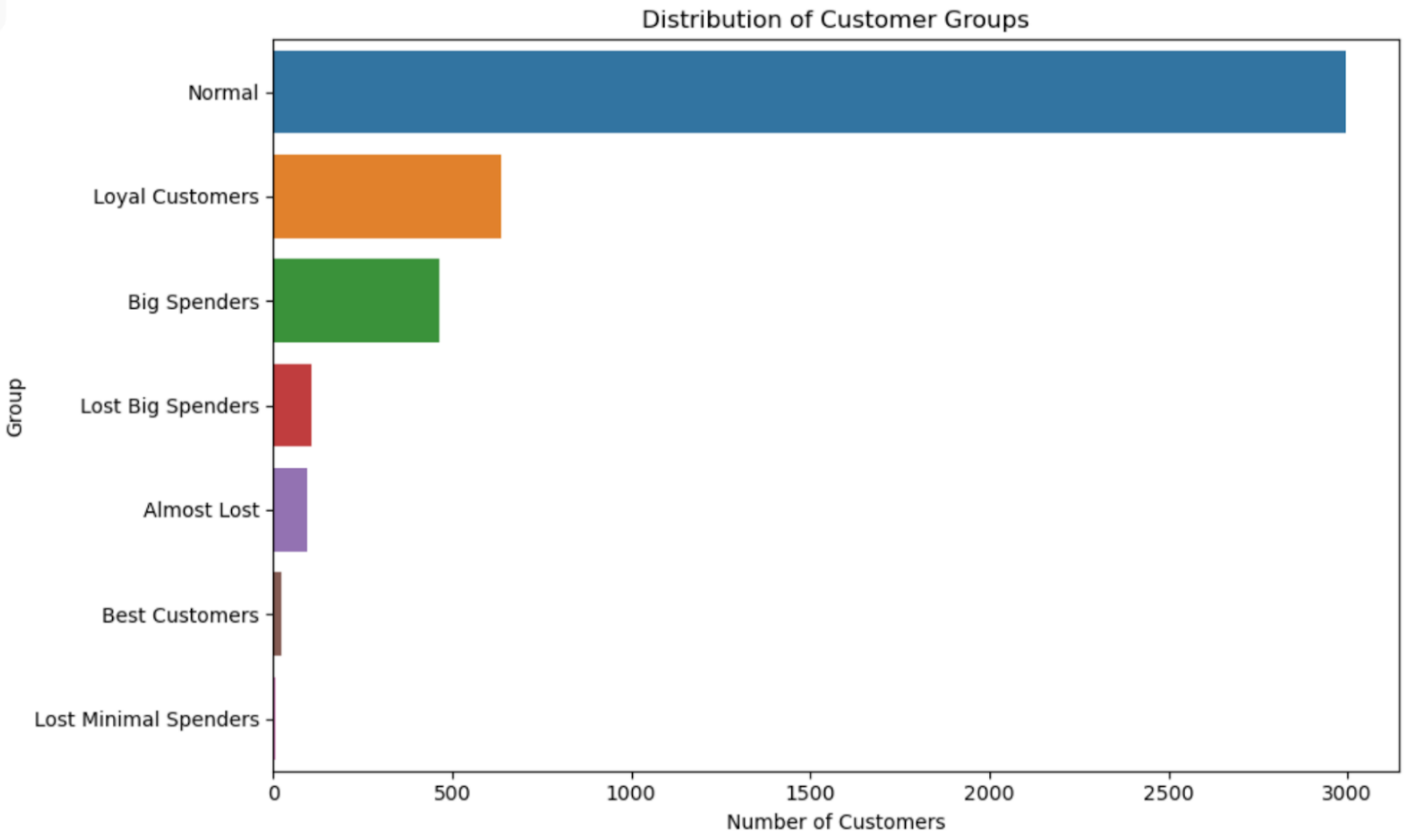
]:

	CustomerID	DATE	Recency	Frequency	Monetary	Recency_Score	Frequency_Score	Monetary_S
1	12347.0	2011-12-07 15:52:00	1	182	4310.00	5	5	
2	12348.0	2011-09-25 13:13:00	74	31	1797.24	2	3	
3	12349.0	2011-11-21 09:51:00	18	73	1757.55	4	4	
4	12350.0	2011-02-02 16:01:00	309	17	334.40	1	2	
5	12352.0	2011-11-03 14:37:00	35	95	1545.41	3	4	

33]:

```
import seaborn as sns
plt.figure(figsize=(8, 6))
sns.scatterplot(x='Recency', y='Frequency', hue='Cluster', data=rfm, palette='Spectral
plt.title('Customer Segmentation using RFM Model')
plt.xlabel('Recency')
plt.ylabel('Frequency')
plt.legend(title='Cluster')
plt.show()
```


Customer Segmentation using RFM Model
Cluster
0
1
2
3
Frequency
8000
7000
6000
5000
4000
3000
2000
1000
0
0
50
100
150
200
250
300
350
Recency



Type of customer	No. of Customers
Normal	2994
Loyal	637
Big spenders	464
Lost big spenders	106
Almost lost	94
Best	22
Lost minima spenders	5

Marketing Recommendations:

- **Tailored Marketing Strategies:**

  Based on the visualizations generated, that outline the distribution of customer segments, their RFM (Recency, Frequency, Monetary) scores, and the clustering of these segments, here are actionable marketing recommendations for each customer segment to improve customer retention and maximize revenue:

  Normal (Largest Segment):

  Strategy: Convert 'Normal' customers into more engaged segments through targeted communications and introductory offers. Actionable Steps: Implement a customer education campaign about the benefits of more frequent purchases, offer incentives for frequent shopping, and personalized product recommendations based on browsing behavior.

  Loyal Customers:

  Strategy: Maintain their high purchase frequency with rewards and recognition. Actionable Steps: Offer a loyalty program with points or discounts, exclusive access to sales, and engage them with brand storytelling to deepen their emotional connection to the brand.

  Big Spenders:

  Strategy: Encourage larger basket sizes and premium product purchases. Actionable Steps: Upsell and cross-sell premium products or services,

provide bundle offers, and create high-value tailored content that resonates with their willingness to spend more.

Lost Big Spenders:

Strategy: Re-engage to understand their disengagement and reintroduce them to the brand. Actionable Steps: Send out personalized re-engagement campaigns, survey to understand why they left, and offer a we-miss-you discount or gift with purchase to incentivize their return.

Almost Lost:

Strategy: Win back these customers before they lapse with urgency-triggering campaigns. Actionable Steps: Implement a win-back email series with time-sensitive offers, highlight product improvements or new arrivals, and offer a limited-time welcome back discount.

Best Customers:

Strategy: Foster exclusivity and premium experiences to retain their high-value status. Actionable Steps: Provide VIP service, exclusive events or content, early access to new products, and high-engagement experiences that exceed expectations.

Lost Minimal Spenders:

Strategy: Do not invest heavily but keep the door open for their return.

Actionable Steps: Include them in mass-market campaigns, offer self-service tools for re-engagement, and maintain a brand presence in their lives without significant marketing spend. The bubble chart of RFM segments suggests varying levels of engagement across different RFM scores.

The heatmap indicates the density of customers within specific RFM score combinations, which can be used to tailor the intensity of marketing efforts.

Lastly, the RFM score distribution shows us the spread of customer engagement. We can leverage this information to focus our efforts on moving customers from lower RFM scores to higher ones through personalized marketing strategies that consider their unique behaviors and value to the company.

All strategies should be continuously monitored and optimized based on customer response and feedback to ensure the best return on investment and customer satisfaction.

Summary of results:

1.Data Overview:

- The size of the dataset in terms of rows and columns are concluded to be as following:
  The number of rows are: 406829
  The number of columns are: 8
- Brief description of each column:
    - InvoiceNo: A six-digit number storing the details of the transaction. A cancellation if it begins with the letter "c".
    - StockCode: A code which defines the product which has been sold.
    - Description: Product name
    - Quantity: The quantities of each product per transaction
    - InvoiceDate: Shows the time and day that each transaction was created.
    - UnitPrice: Product price per unit.
    - CustomerID: A unique number designated to each customer.
    - Country: Name of the country where each customer resides.

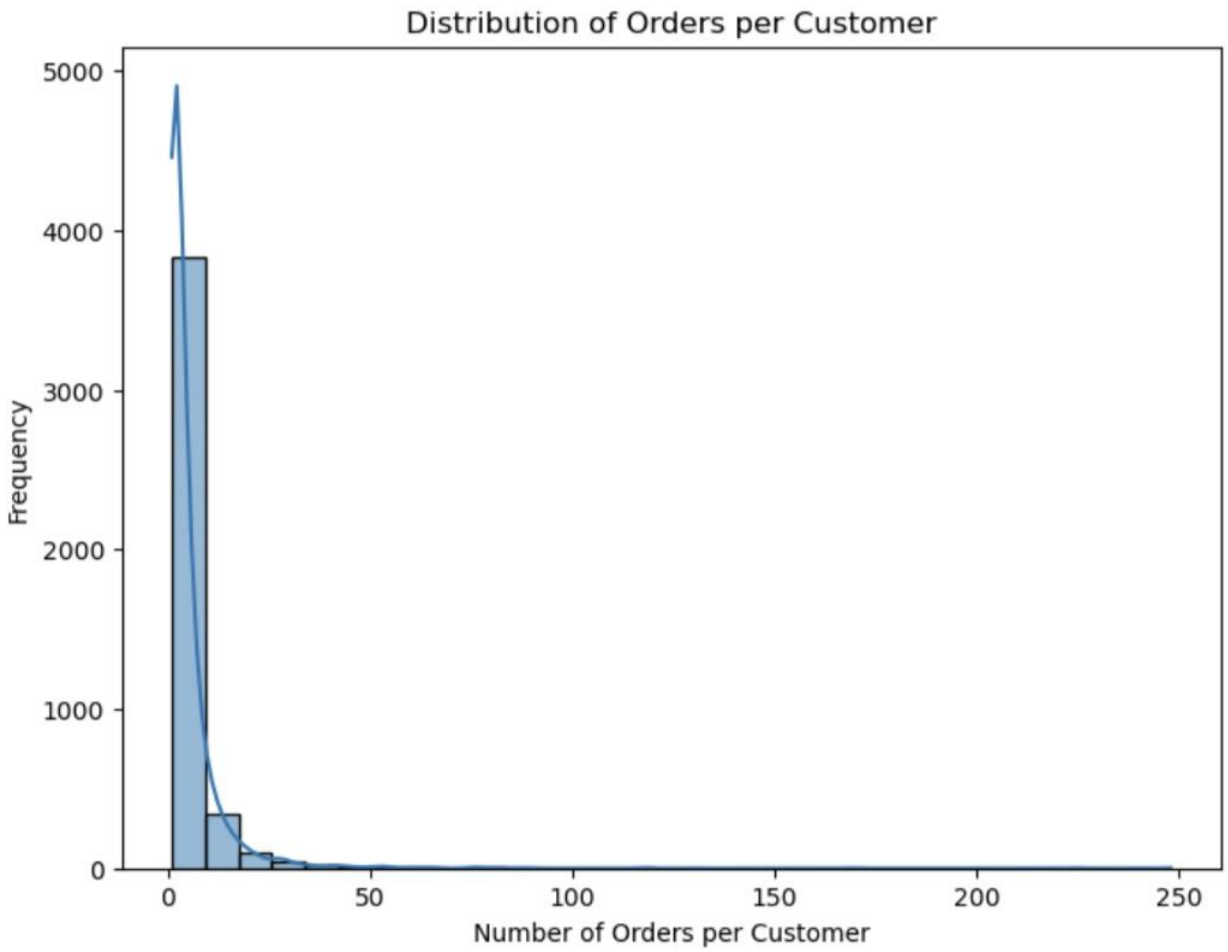
- Time period covered by this dataset:

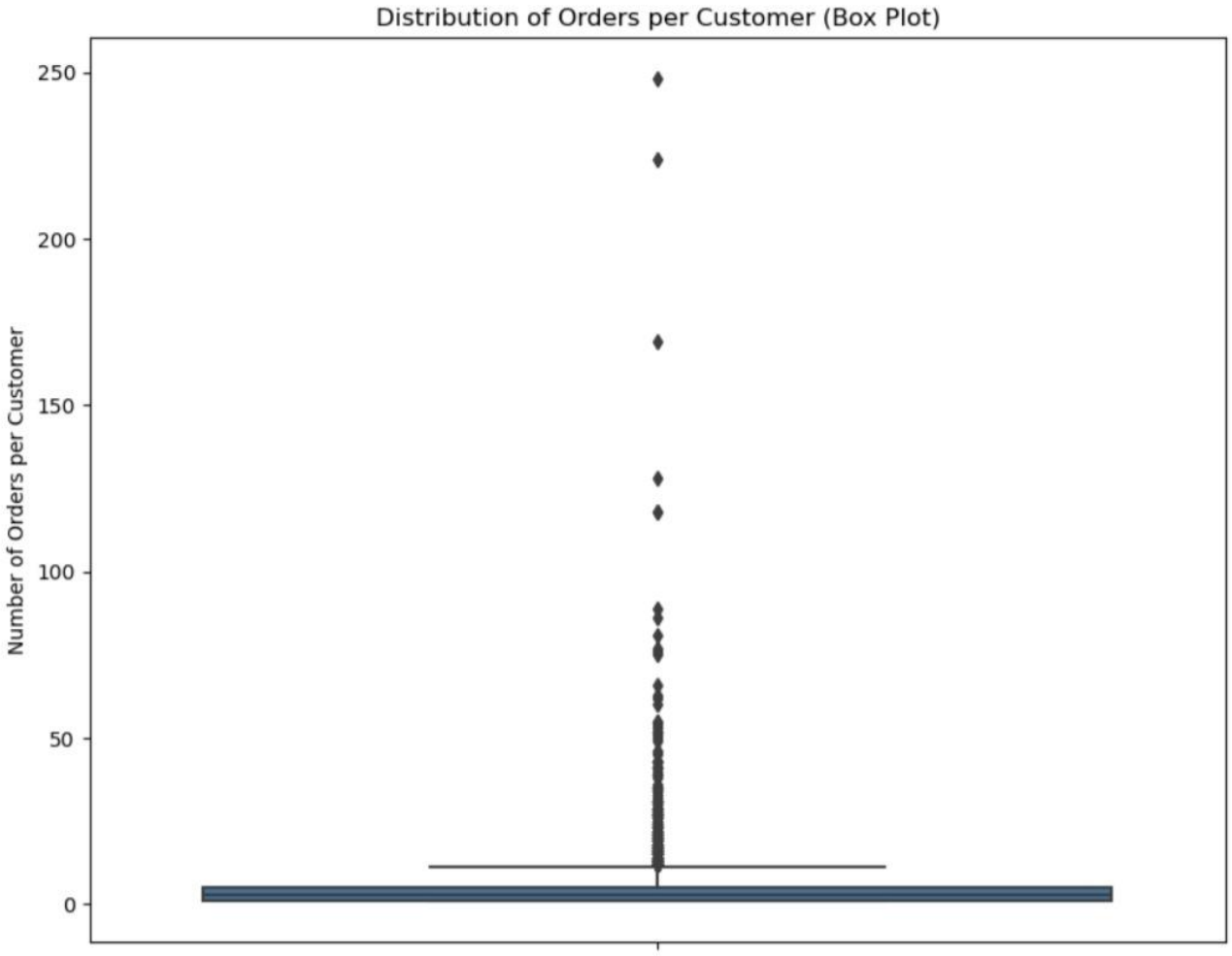
  The dataset contains data starting from 1/10/2011, 10:32 - 9/9/2011, 9:52.

2. Customer Analysis:

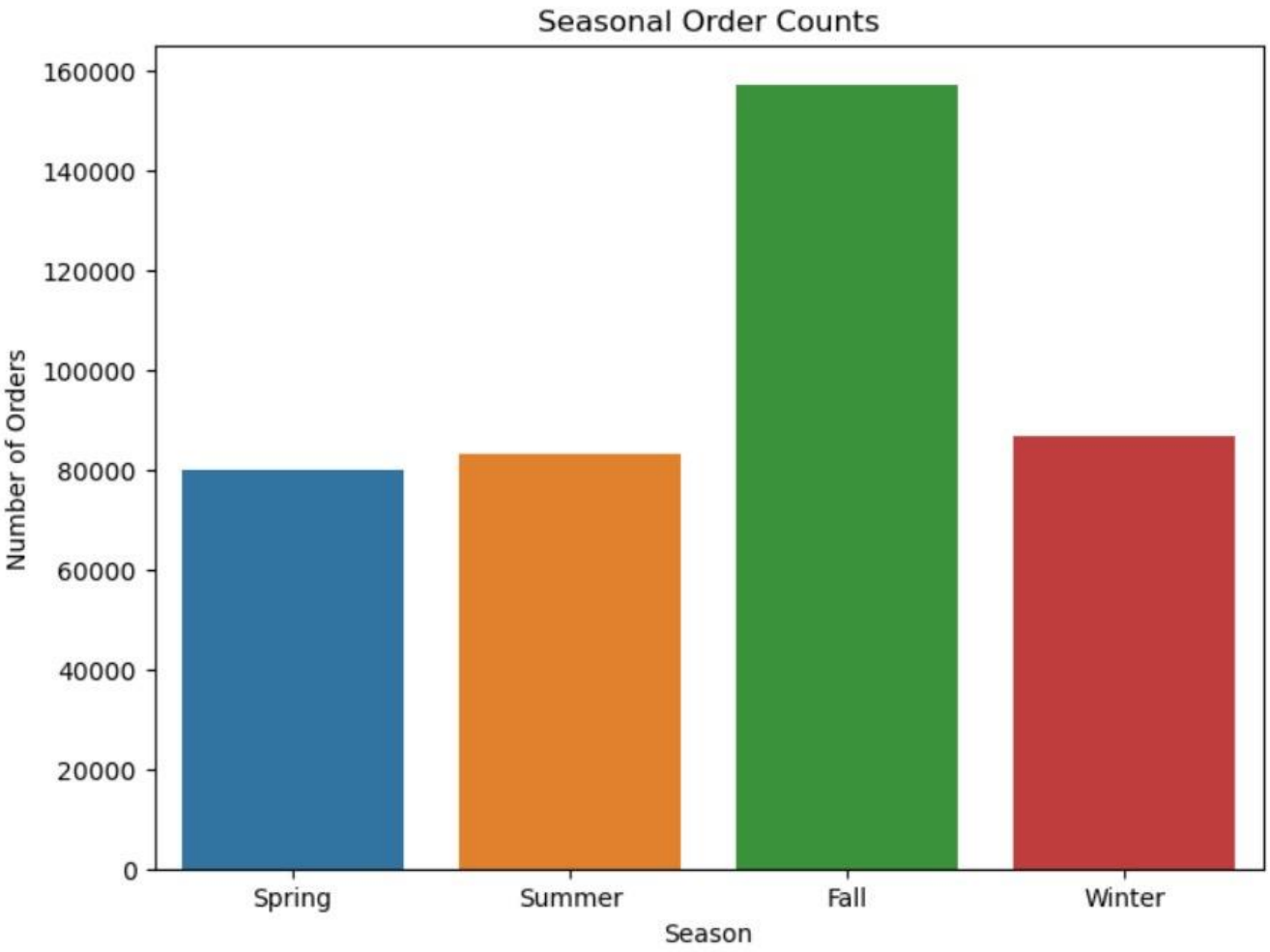
- By calculating the number of unique customers using the CustomerID column we could conclude the number of unique customers present in the dataset to be 4372.
- From the total number of unique transactions/orders in the dataset and the average number of orders per customer, it can be indicated that the distribution of the number of orders per customer is 5.07548032936871.
- By grouping the DataFrame by 'CustomerID' and calculating the number of unique invoices (orders) for each customer, the top 5 customers are identified as

Customer ID	No. of Orders
14911.0	248
12748.0	224
17841.0	169
14606.0	128
13089.0	118


Distribution of Orders per Customer
Frequency
Number of Orders per Customer
0
1000
2000
3000
4000
5000
0
50
100
150
200
250


Distribution of Orders per Customer (Box Plot)
Number of Orders per Customer
250
200
150
100
50
0

- We can also take a look at the seasonal behavior of customer order counts. The order counts peak in the fall season and are lowest in spring.



3. Product Analysis:

- Counting the occurrences of each product description has resulted in identifying the top 10 products to be:

WHITE HANGING HEART T-LIGHT HOLDER - 2070

REGENCY CAKESTAND 3 TIER - 1905

JUMBO BAG RED RETROSPOT - 1662

ASSORTED COLOUR BIRD ORNAMENT - 1418

PARTY BUNTING - 1416

LUNCH BAG RED RETROSPOT - 1358

SET OF 3 CAKE TINS PANTRY DESIGN - 1232

POSTAGE - 1196

LUNCH BAG BLACK SKULL - 1126

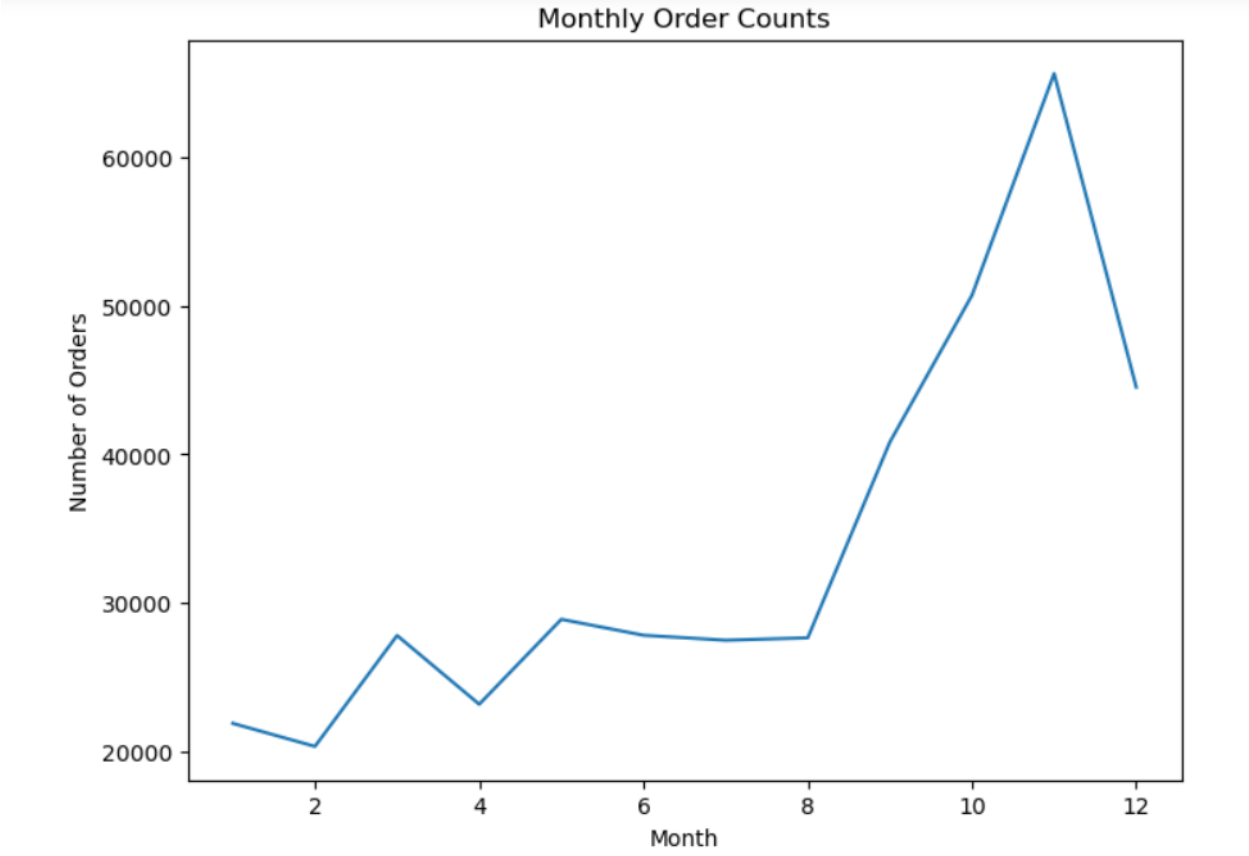
PACK OF 72 RETROSPOT CAKE CASES - 1080

- Further we calculated the total price of all products in the DataFrame and the total number of products in the DataFrame and used the result to find the average price of a product to be 3.460471018536043
- The highest selling product is: PAPER CRAFT , LITTLE BIRDIE. This has been found by calculating the cost for each product and add a new 'Cost' column to the DataFrame and then Identifying the highest selling product by finding the row where 'Cost' is equal to the maximum 'Cost'.

4. Time Analysis:

- By grouping the DataFrame by 'Day of Week' and 'Hour of Day', aggregating the sum of 'Cost' and 'Quantity', calculating the average order amount and add a new column 'Average Order Amount' to the DataFrame and identifying the day and hour with the highest average order amount, we were able to conclude that the day of the week with the highest average order amount is Monday and the hour of the day with the highest average order amount is 19.
- By creating a line plot to visualize the monthly order counts, we were able to identify the presence of seasonal trends in the dataset.

  Using the graph below we can conclude that there is a seasonal fluctuation in the dataset as we can see that the number of orders is as low as 20000 in the starting months 2 and 4 while it hits a peak of above 60000 between the months 10 and 12.



5. Geographical Analysis:

- By grouping the DataFrame by 'Country' and calculate the total quantity of orders for each country we identified the top 5 countries with the highest number of orders:

Country	**No. of Orders**
United Kingdom	4008533
Netherlands	200128
Ireland	136329
Germany	117448
France	109848

6. Customer Behavior:

- From converting the 'InvoiceDate' column to datetime format and calculating the time difference between the latest and earliest invoice dates for each customer we were able to derive that the average duration of customer activity is 133 days 17:25:29.204025618.

- We then converted the 'InvoiceDate' column to datetime format and grouped by 'CustomerID' and calculated the first and last purchase dates to find the mean of customers remaining active (between their first and last purchase) to be 133.38586459286367.

**There is no sufficient data in the given data set for analyzing Returns and refunds, payment analysis, profitability and customer satisfaction.