1. Exploratory Data Analysis

Load the data and print first 10 rows:

```
df = pd.read_csv('/content/Basket_dataset.csv')
```

```
df.head(10)
```

	Member_number	Date	itemDescription
0	1808	21/07/2015	tropical fruit
1	2552	05/01/2015	whole milk
2	2300	19/09/2015	pip fruit
3	1187	12/12/2015	other vegetables
4	3037	01/02/2015	whole milk
5	4941	14/02/2015	rolls/buns
6	4501	08/05/2015	other vegetables
7	3803	23/12/2015	pot plants
8	2762	20/03/2015	whole milk
9	4119	12/02/2015	tropical fruit

The info() function shows that dataset compromises 38,765 transactions from grocery stores with complete data across all fields: Member_number (numeric), Date and itemDescription (both nominal). The Date field, listed as an object type, will require conversion to datetime for proper temporal analysis. 38,765 non-null entries indicate no missing values in all three fields.

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 38765 entries, 0 to 38764
Data columns (total 3 columns):
 #   Column           Non-Null Count  Dtype 
---  ------           --------------  ----- 
 0   Member_number    38765 non-null  int64 
 1   Date             38765 non-null  object
 2   itemDescription  38765 non-null  object
dtypes: int64(1), object(2)
memory usage: 908.7+ KB
```

```
df.isnull().sum()
```

```
Member_number      0
Date               0
itemDescription    0
dtype: int64
```

This is_null function and info() function given above indicate that there are no missing values in any of the columns.

Transformation of Date field into Date data type:

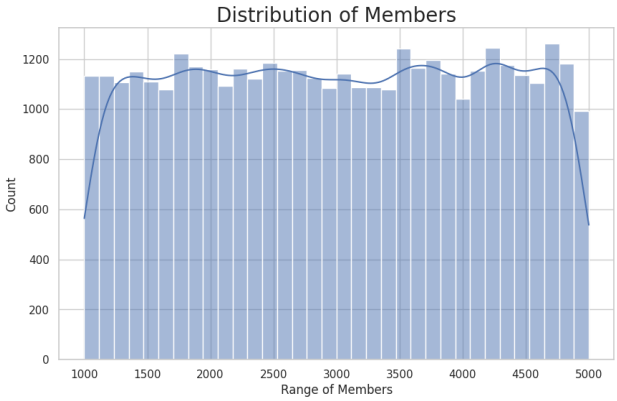
Date field is converted to Date data type and run info() function again to check the result:

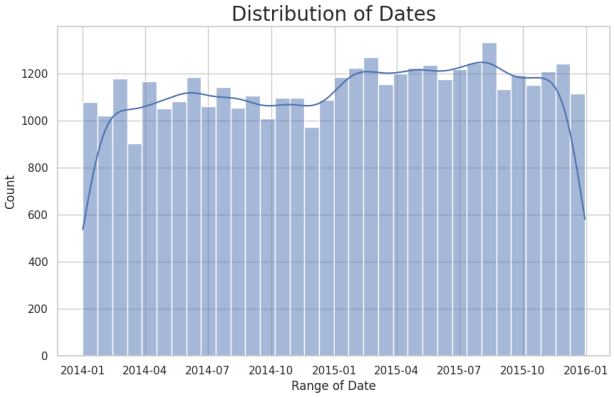
```
data_dist['Date'] = pd.to_datetime(data_dist['Date'], dayfirst=True)
```

```
data_dist.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 38765 entries, 0 to 38764
Data columns (total 3 columns):
 #   Column           Non-Null Count  Dtype         
---  ------           --------------  -----         
 0   Member_number    38765 non-null  int64         
 1   Date             38765 non-null  datetime64[ns]
 2   itemDescription  38765 non-null  object        
dtypes: datetime64[ns](1), int64(1), object(1)
memory usage: 908.7+ KB
```

1.1. Distribution analysis





This histogram on the right shows the frequency distribution of transactions per member, indicating a relatively uniform spread with a consistent number of transactions across different members.

The histogram on the left, complemented by a density trend line, shows that the transaction frequency from early 2014 to the onset of 2016, showcasing consistent distribution with notable peaks. These peaks likely signify seasonal purchase trends or the effects of marketing efforts. Such insights are instrumental in pinpointing critical periods of consumer activity, serving as a roadmap for strategic business planning.

1.2. Statistical Exploration

	Member_number	Date	itemDescription
count	38765.000000	38765	38765
unique	NaN	728	167
top	NaN	21/01/2015	whole milk
freq	NaN	96	2502
mean	3003.641868	NaN	NaN
std	1153.611031	NaN	NaN
min	1000.000000	NaN	NaN
25%	2002.000000	NaN	NaN
50%	3005.000000	NaN	NaN
75%	4007.000000	NaN	NaN
max	5000.000000	NaN	NaN

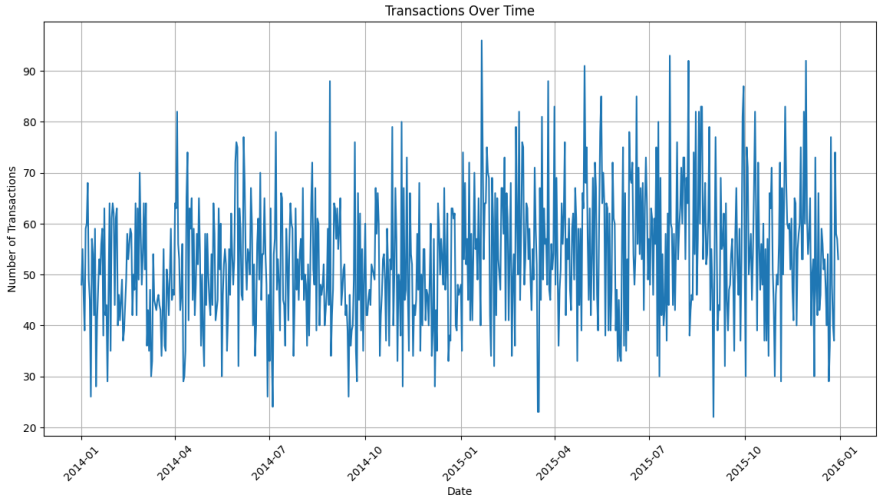
Summary Statistics

This dataset records 38,765 grocery store transactions, with member numbers ranging from 1,000 to 5,000 and an average of 3,003.64, reflecting a wide customer base. Transaction data spans 728 unique dates, highlighting continuous engagement, with the 21st of January, 2015, as the most active shopping day. Among 167 unique items, 'whole milk' is the most frequently purchased, appearing 2,502 times, indicating its staple status among customers. These insights suggest focal points for targeted marketing and sales optimization strategies.

Let’s explore Date distribution:

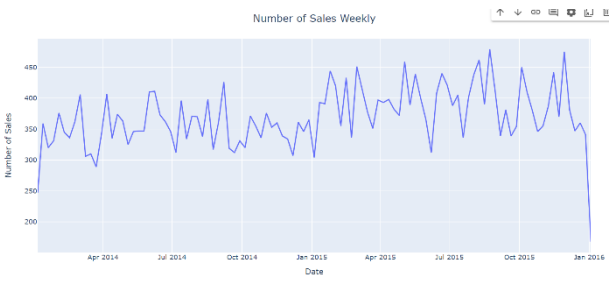
```
Earliest Date: 2014-01-01 00:00:00
Latest Date: 2015-12-30 00:00:00
Number of Unique Days: 728
```

Transactions Over Time

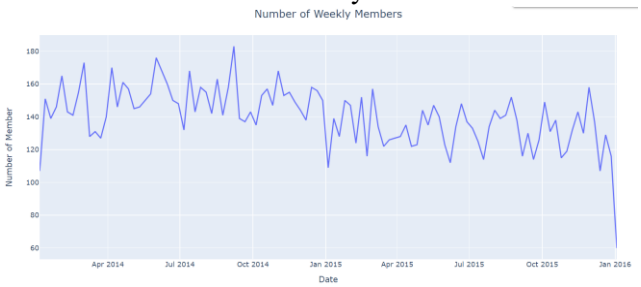


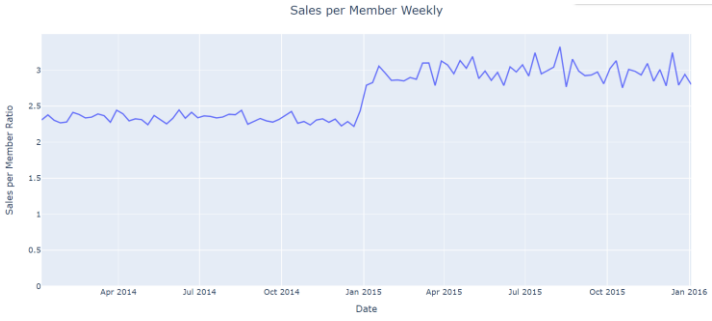
The screenshot above shows a time series plot titled "Transactions Over Time" that depicts the number of transactions recorded at a grocery store from early 2014 to the end of 2015. From this visual, we observe that the transaction volume exhibits fluctuations over time with no clear long-term upward or downward trend. There are spikes in transactions on certain dates, suggesting potential seasonal peaks or special promotion days that draw more customers. The variability in the data could indicate diverse customer purchasing behavior and might be influenced by various factors such as holidays, marketing campaigns, or other external events.

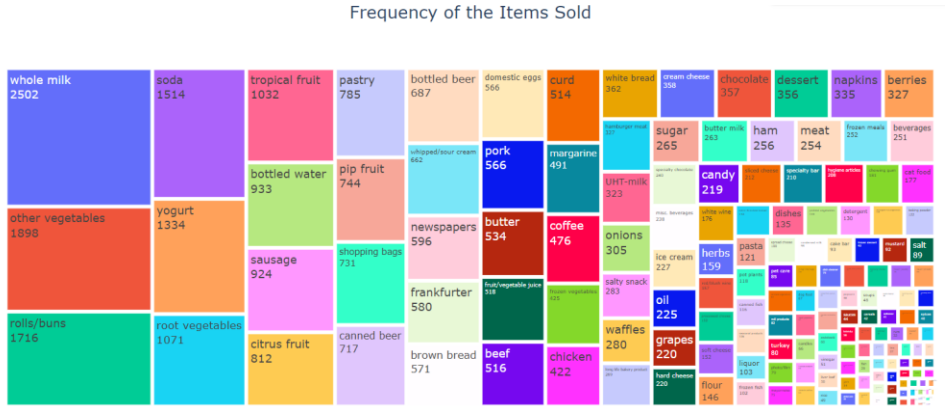
Number of weekly sales:



Number of weekly members:







Top 10 most frequently dates:

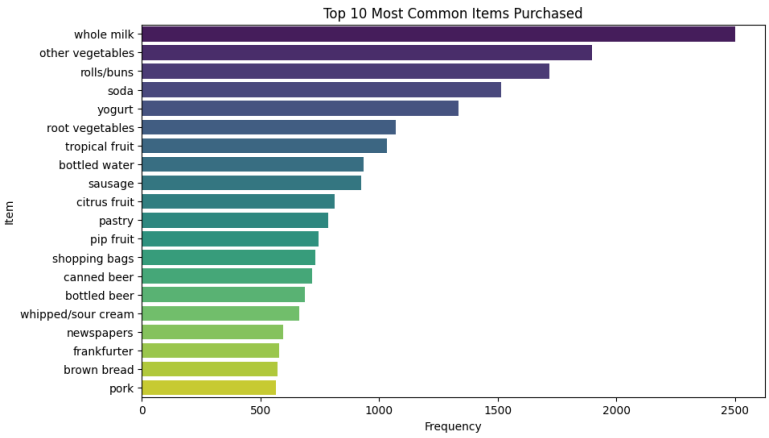
```
df['Date'].value_counts().head(10)
```

```
Date
2015-01-21    96
2015-07-21    93
2015-11-29    92
2015-08-08    92
2015-04-30    91
2014-08-28    88
2015-03-26    88
2015-09-30    87
2015-06-19    85
2015-05-16    85
Name: count, dtype: int64
```

Now let's explore **item description frequency**:

Frequency of Top Item Purchases

```
Number of Unique Items: 167
Average Purchases per Item: 232.125748502994
SD of Purchases per Item: 363.4420982644306
```

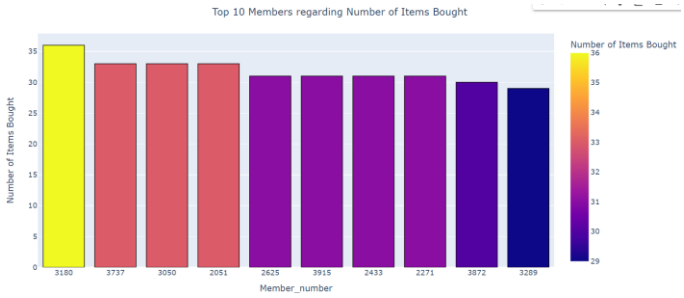


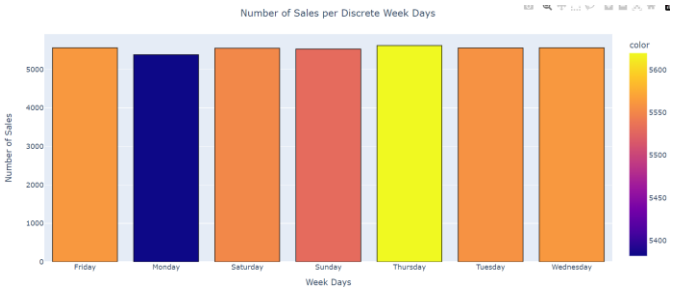
The screenshot displays top 10 most common purchased items which ranks items by their purchase frequency. "Whole milk" is the most frequently bought item, with a count close to 2500, indicating its essential role in the shopping habits of customers. Other vegetables and rolls/buns also show high purchase frequencies, suggesting they are staple items. Items like soda, yogurt, and root vegetables have moderate purchase frequencies, while the rest, including tropical fruit, bottled water, and sausage, are less frequent but still among the top 10.

Now let’s explore Transactions Per Member

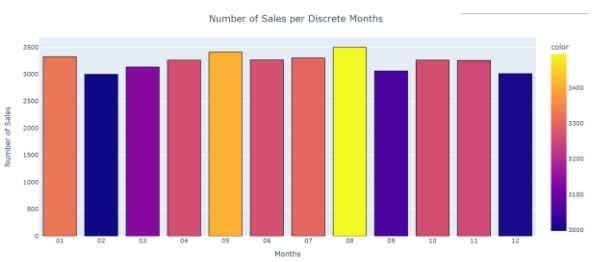
```
Average Number of Transactions per Member: 9.944843509492047
Median Number of Transactions per Member: 9.0
Standard Deviation of Transactions per Member: 5.310795850646273
```

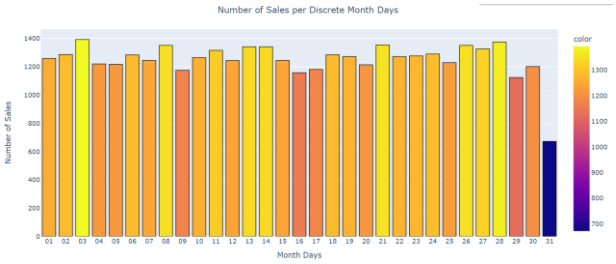
The following graph shows top 10 members regarding purchased no of items





The graph above shows that sales peak on Thursdays, with Sunday and Wednesday being the next highest days for sales volume.



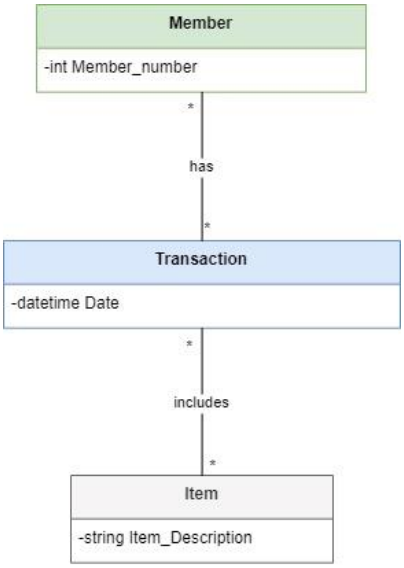


The graphs above on the right show that sales peak on August and May months and 3rd, 21st and 28th of days in a month.

Finally I checked unique values in the categorical feature – item description variable and the result confirms that the feature is consistent and does not contain any unexpected categories

```
Unique values in the categorical feature: ['tropical fruit' 'whole milk' 'pip fruit' 'other vegetables' 'rolls/
 'pot plants' 'citrus fruit' 'beef' 'frankfurter' 'chicken' 'butter'
 'fruit/vegetable juice' 'packaged fruit/vegetables' 'chocolate'
 'specialty bar' 'butter milk' 'bottled water' 'yogurt' 'sausage'
 'brown bread' 'hamburger meat' 'root vegetables' 'pork' 'pastry'
 'canned beer' 'berries' 'coffee' 'misc. beverages' 'ham' 'turkey'
 'curd cheese' 'red/blush wine' 'frozen potato products' 'flour' 'sugar'
 'frozen meals' 'herbs' 'soda' 'detergent' 'grapes' 'processed cheese'
 'fish' 'sparkling wine' 'newspapers' 'curd' 'pasta' 'popcorn'
 'finished products' 'beverages' 'bottled beer' 'dessert' 'dog food'
 'specialty chocolate' 'condensed milk' 'cleaner' 'white wine' 'meat'
 'ice cream' 'hard cheese' 'cream cheese ' 'liquor' 'pickled vegetables'
 'liquor (appetizer)' 'UHT-milk' 'candy' 'onions' 'hair spray'
 'photo/film' 'domestic eggs' 'margarine' 'shopping bags' 'salt' 'oil'
 'whipped/sour cream' 'frozen vegetables' 'sliced cheese' 'dish cleaner'
 'baking powder' 'specialty cheese' 'salty snack' 'Instant food products'
 'pet care' 'white bread' 'female sanitary products' 'cling film/bags'
 'soap' 'frozen chicken' 'house keeping products' 'spread cheese'
 'decalcifier' 'frozen dessert' 'vinegar' 'nuts/prunes' 'potato products'
 'frozen fish' 'hygiene articles' 'artif. sweetener' 'light bulbs'
 'canned vegetables' 'chewing gum' 'canned fish' 'cookware'
 'semi-finished bread' 'cat food' 'bathroom cleaner' 'prosecco'
 'liver loaf' 'zwieback' 'canned fruit' 'frozen fruits' 'brandy'
 'baby cosmetics' 'spices' 'napkins' 'waffles' 'sauces' 'rum'
```

1.3. UML design



2. Perform RFM Segmentation in SQL

2.1. Definition of RFM Metrics

Recency (R): Measures how recently a customer made their last purchase. A lower Recency value is better because it means the customer has purchased more recently.

Frequency (F): Measures how often a customer makes a purchase within a specified time frame. A higher Frequency value is better, indicating a customer makes purchases more frequently.

Monetary (M): Measures how much money a customer has spent in total over a specified period. A higher Monetary value is better, indicating a customer spends more.

2.2. Implementation of RFM Metrics

Initially, I created the table and imported the pro-processed data into Oracle database.

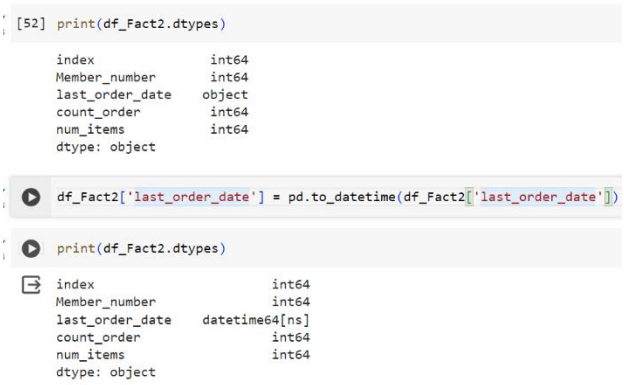
```
# load CRM data into the RFMdb database
df_Fact.to_sql("data", conn, if_exists='replace', index=False)
```

Then the data is grouped into segments by member number

```
# clean data and group transactions by Customerid
cleandata= pd.read_sql('''
SELECT Member_number,
       MAX(Date) AS last_order_date,
       COUNT(DISTINCT Date) AS count_order,
       COUNT(DISTINCT itemDescription) AS num_items
FROM data
GROUP BY Member_number ''', conn)
# Write clean CRM data into the database
cleandata.to_sql("cleandata", conn)
```

This is how the cleandata table looks like:

	Member_number	last_order_date	count_order	num_items
0	1000	2015-11-25 00:00:00	5	11
1	1001	2015-05-02 00:00:00	5	9
2	1002	2015-08-30 00:00:00	4	8
3	1003	2015-02-10 00:00:00	4	6
4	1004	2015-12-02 00:00:00	8	16
5	1005	2014-01-23 00:00:00	2	3
6	1006	2015-06-14 00:00:00	4	12
7	1008	2015-10-03 00:00:00	2	11
8	1009	2015-10-05 00:00:00	4	8
9	1010	2015-07-31 00:00:00	5	10



As it was shown in the screenshot on the left, the transformation was carried out using the pd.to_datetime() function from the pandas library which converts a series of date-like objects into a datetime64 series.

Then the last_order_date column underwent an additional transformation where only the date part was retained. This was done using the "**.dt.date"**, removing the time component if any existed.

```
# get only date part from datetime feature column
df_Fact2["last_order_date"] = df_Fact2["last_order_date"].dt.date
```

```
# Create snapshot date
snapshot_date = df_Fact2['last_order_date'].max() + datetime.timedelta(days=1)
snapshot_date
```

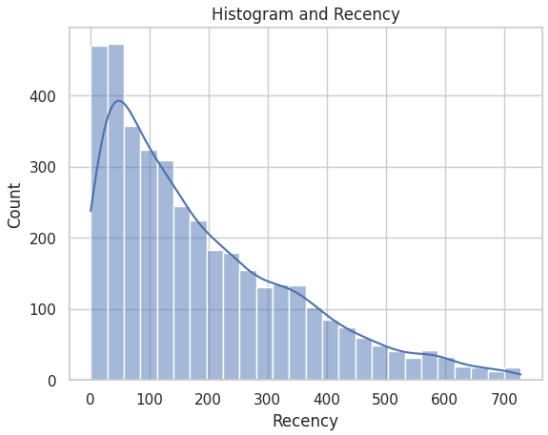
```
datetime.date(2015, 12, 31)
```

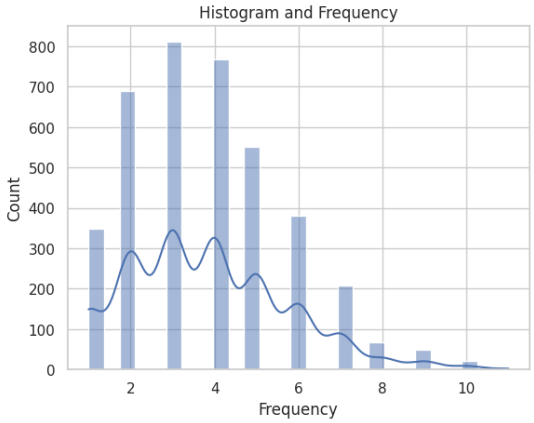
```
# Aggregate data by each customer
members = df_Fact2.groupby(['Member_number']).agg({
   'last_order_date': lambda x: (snapshot_date - x.max()).days,
   'count_order':'sum' ,
   'num_items': 'sum'})
```

A snapshot date was created to serve as a reference point for the analysis. This date is calculated as one day (2015-12-31) after the latest date (2015-12-30) found in the last_order_date column. This snapshot date is used for calculating recency in customer behavior analysis, where the recency metric is a measure of how recent the last interaction with a customer was.

The DataFrame was then grouped by the Member_number column to aggregate data at the customer level. The following aggregations were applied:

- The recency of the last order is calculated as the number of days between the snapshot date and the most recent order for each customer.
- The count_order column is summed up to provide a total count of orders per customer.
- The num_items column is summed up to provide a total number of items ordered per customer.

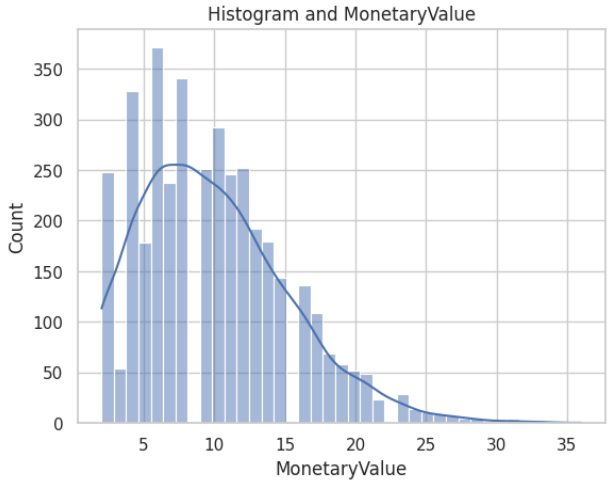




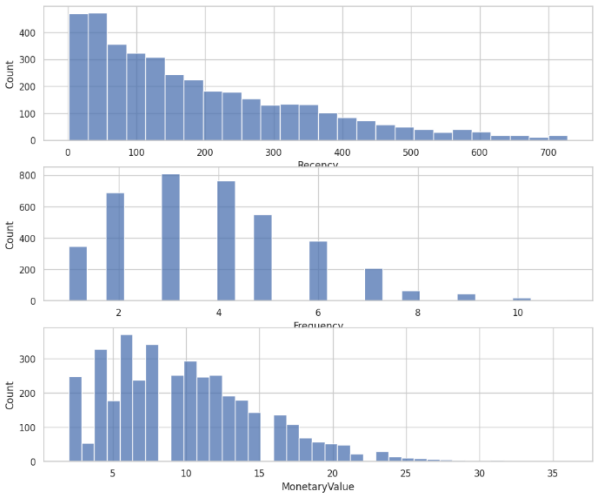
The histogram on the left shows the distribution of Recency values for customers, with most customers having made a purchase recently (skewed towards the left), and fewer customers going longer without purchasing (tail extending to the right). The curve indicates the density estimation of these values.

The histogram on the right shows the frequency of customer transactions with the majority of customers having low transaction counts, indicating a larger number of infrequent buyers, and fewer customers making more frequent purchases, as shown by the declining count as the frequency number increases.

Monetary: Given that our dataset lacks information on product prices, I will use the number of items purchased per user as a proxy for the Monetary value.



The histogram on the left depicts the Monetary Value distribution among customers, showing most customers spend in the lower monetary range with a right-skewed distribution, indicating that fewer customers have very high spend amounts.



The combined histograms for Recency, Frequency, and MonetaryValue indicate a customer profile with the following traits: most recent transactions are quite recent, transaction frequency has a varied distribution with a fall-off for higher frequencies, and most customers spend in lower monetary ranges with fewer high-spenders, displaying a right-skewed distribution in MonetaryValue.

The calculated RFM table:

Member_number	Recency	Frequency	MonetaryValue
1000	36	5	13
1001	243	5	12
1002	123	4	8
1003	324	4	8
1004	29	8	21
1005	707	2	4
1006	200	4	15
1008	89	2	12
1009	87	4	9
1010	153	5	12

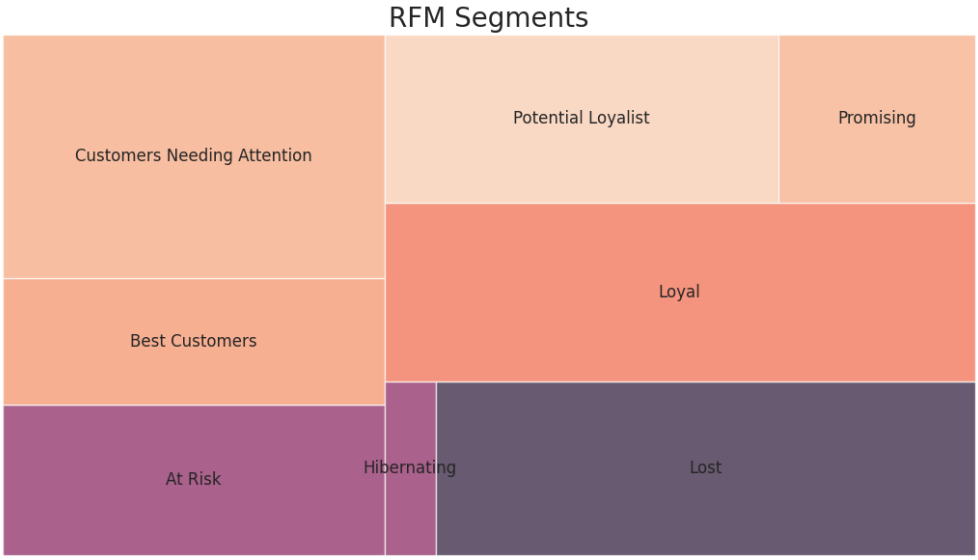
2.3. RFM segmentation

In RFM segmentation, customers are categorized into eight groups according to their buying habits, as shown in the accompanying table and chart below. This allows marketers to focus on customers in the 'at risk' and 'needing attention' segments, which encompass more than 1000 customers. By improving engagement strategies, the goal is to increase sales and profits. Moreover, the significant count of lost customers indicates a need for the marketing team to explore why interest in the store or products is decreasing. Conducting this analysis could reveal critical opportunities for business enhancement.

	Member_number	rfm_recency	rfm_frequency	rfm_monetary	rfm_combined	rfm_level
3848	3209	4	3	3	433	Loyal
3849	2879	4	4	3	443	Loyal
3850	1086	4	4	4	444	Best Customers
3851	2764	4	4	3	443	Loyal
3852	4091	4	4	4	444	Best Customers
3853	1997	4	4	4	444	Best Customers
3854	4719	4	4	4	444	Best Customers
3855	2222	4	4	4	444	Best Customers
3856	2447	4	4	4	444	Best Customers
3857	4010	4	1	2	412	Potential Loyalist
3858	4095	4	2	2	422	Potential Loyalist
3859	3118	4	2	2	422	Potential Loyalist
3860	4285	4	3	2	432	Potential Loyalist

rfm_level	Member_number
At Risk	444
Best Customers	371
Customers Needing Attention	716
Hibernating	69
Lost	721
Loyal	812
Potential Loyalist	511
Promising	254

Visualization of RFM segments

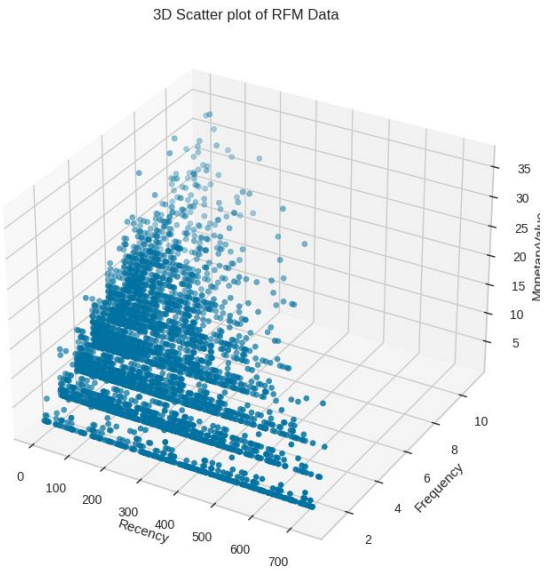


3. Customer segmentation with DBSCAN

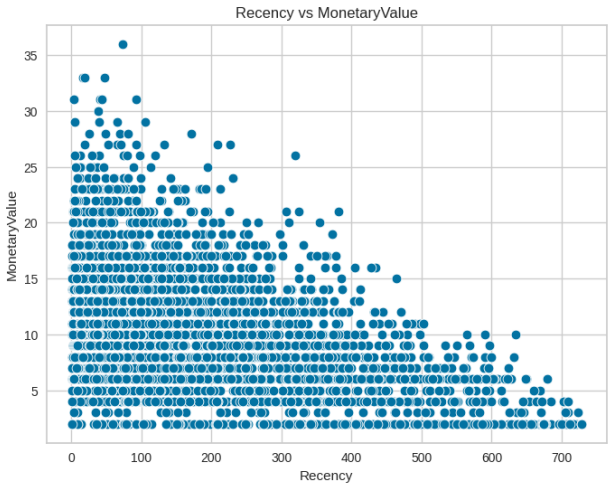
Now let’s try to segment customers with clustering algorithms. DBSCAN algorithm is particularly robust in the presence of outliers, as one of its main features is the ability to identify and exclude noise or outlier points from the clusters it forms. Therefore, it's not necessary to delete outliers before running DBSCAN, as the algorithm itself is designed to handle them by categorically assigning them to a noise category (labelled as -1).

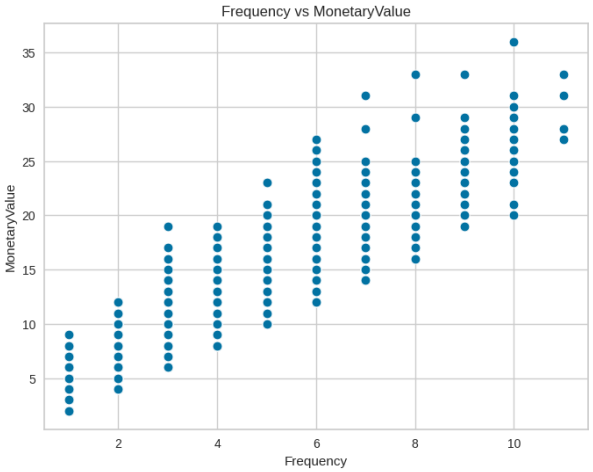
Before applying the DBSCAN algorithm, a preliminary visualization of the RFM dataset was conducted to understand the data's distribution and inherent clustering tendencies.

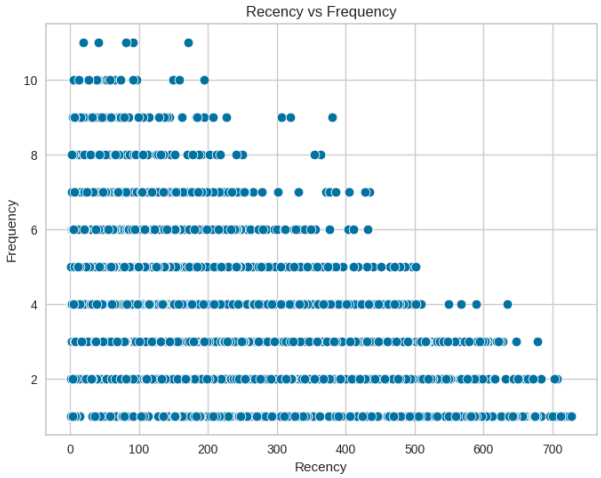
3D Visualization Pre-DBSCAN Analysis:



A 3D scatter plot of the RFM data displays a prominent cluster with low Recency and Frequency but varied MonetaryValue, indicating a core segment of engaged customers. The distribution provides guidance for selecting DBSCAN's eps and minPts, crucial for differentiating between dense customer groupings and potential outliers.







In the three 2D scatter plots depicting relationships within the RFM data:

1. **Recency vs. MonetaryValue**: Shows that newer engagements (lower Recency) are not necessarily linked to higher spend (MonetaryValue). The data points appear widely spread, suggesting no strong direct relationship between how recently a customer has purchased and the amount spent.

2. **Frequency vs. MonetaryValue**: Indicates clusters at lower frequencies, with most customers making a moderate number of transactions, but a few engage more frequently. The spread is even, revealing no significant increase in spend with higher transaction frequency.

3. **Recency vs. Frequency**: Reveals distinct horizontal bands of Frequency, showing that most customers make purchases at consistent intervals, with only a few deviating from this pattern. No clear correlation with Recency suggests the regularity of purchases does not directly influence how recently customers have engaged.

3.1. Building DBSCAN model:

```
# Testing DBSCAN with initial random eps and min_samples values
dbscan_model = DBSCAN(eps=11, min_samples=4)

dbscan_fit = dbscan_model.fit(RFM)

y_db = dbscan_fit.labels_

# Get the cluster labels
print(y_db)

# Get unique cluster labels
print(np.unique(y_db))

# Calculate silhouette score
print(silhouette_score(RFM, y_db))

RFM_1 = RFM.copy()
RFM_1["Labels"] = y_db
RFM_1["Labels"].value_counts()
```

```
[0 0 0 ... 0 0 0]
[0 1]
0.6027389466891632
```

Initial clustering with DBSCAN using eps=11 and min_samples=4 identified two distinct clusters and achieved a silhouette score of 0.60, indicating moderate separation. However, clustering result, with one large cluster and a much smaller one, suggests a disparity in data density.

Labels	DBSCAN_size
0	3877
1	21

```
# Attempt to improve the model
dbscan_model = DBSCAN(eps=7.68, min_samples=4).fit(RFM)

y_db = dbscan_model.labels_

# Get the cluster labels
print(y_db)

# Get unique cluster labels
print(np.unique(y_db))

# Calculate silhouette score
print(silhouette_score(RFM, y_db))

RFM_DB = RFM.copy()
RFM_DB["Labels"] = y_db
RFM_DB["Labels"].value_counts()
```

To refine eps, the k-distance graph was employed, identifying an optimal eps at the knee point of approximately 7.68. This method, coupled with a rule for min_samples as twice the data dimensionality, suggests a min_samples value of 4 for 2D data, aligning with standard practice and literature. These calculated parameters aim to balance cluster separation with the inclusion of core points, potentially enhancing cluster validity for subsequent analyses.

```
[0 0 0 ... 0 0 0]
[-1  0  1]
```

Labels	DBSCAN_size
-1	4
0	3873
1	21

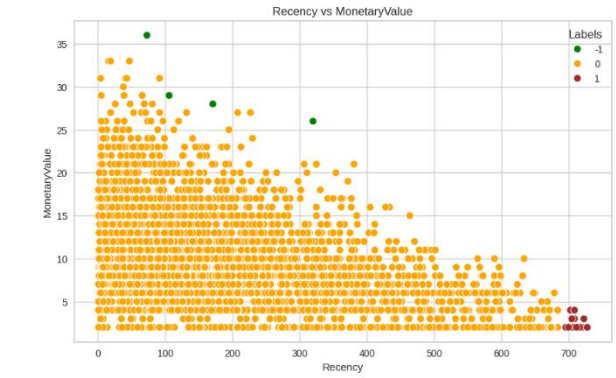
With the updated eps=7.68 and min_samples=4, the DBSCAN algorithm identified three groups: a primary cluster with 3873 points, a smaller cluster with 21 points, and 4 points categorized as noise (label -1). The negative silhouette score of -0.217 indicates overlapping clusters, suggesting that these parameter settings may be suboptimal, as they do not provide clear separation between clusters.

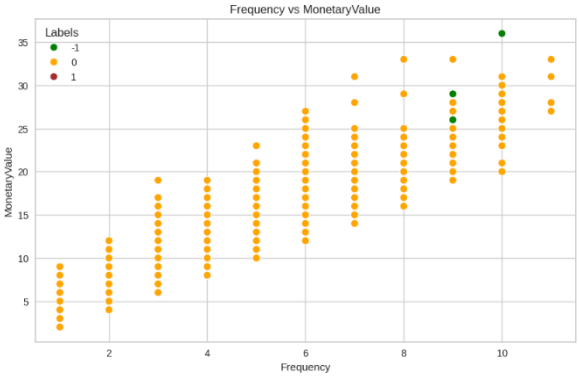
3.2. Testing the DBSCAN Model

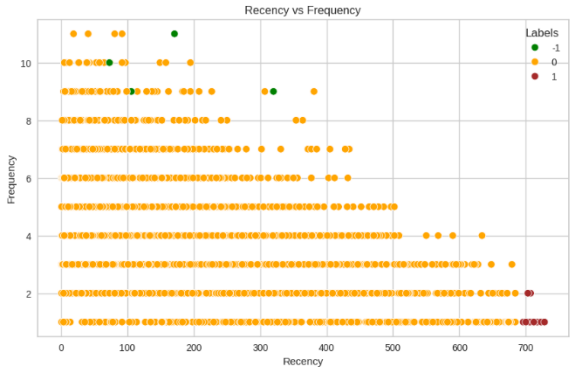
The visual assessment of DBSCAN's clustering on the RFM dataset after adjusted to eps=7.68 and min_samples=4, reveals distinct data groupings and outliers. The scatter plots demonstrate that DBSCAN can identify outliers (green points) and segregate the majority of data into a primary cluster (orange

points), with a small number of points forming a separate cluster (red points). This segregation is particularly visible in the Recency vs. MonetaryValue plot.

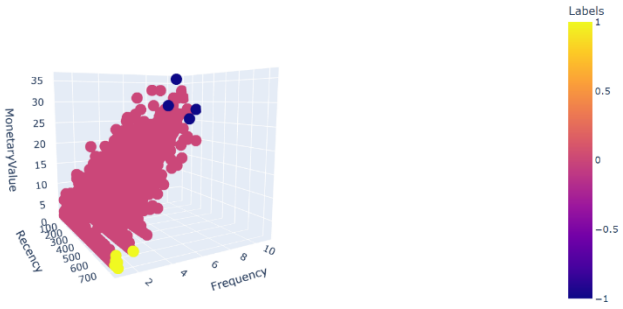
However, the silhouette score of -0.217 suggests poor cluster definition, with a considerable overlap between clusters, potentially undermining the algorithm's practicality for actionable segmentation in its current state.







The 3D plot further corroborates this by showing extensive overlap and the dominance of one main cluster, which may impede the extraction of meaningful customer segments.



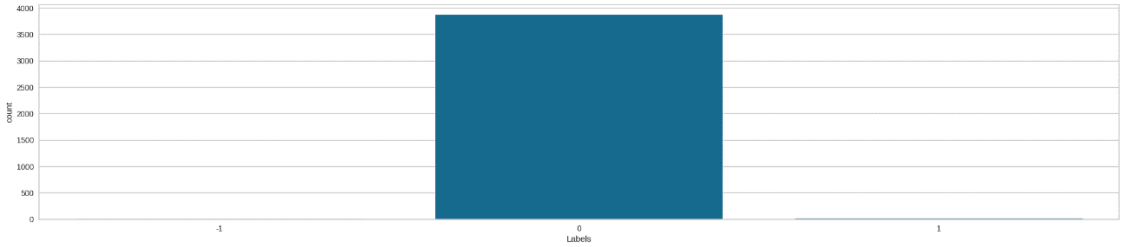
Conclusion: This implies the need for further parameter tuning or reconsideration of the clustering approach, taking into account the density variations within the dataset to achieve meaningful segmentation.

4. Review of Results in terms of business value

4.1. Results when using DBSCAN

Cluster Characterization and Business Value Identification:

Analysis of the DBSCAN-generated RFM clusters presents a clear disparity in the distribution of customers. The majority (3,873) fall within one primary cluster, with a small segment (21) forming a secondary cluster, and a nominal group (4) classified as noise, as seen in the count visualization.




```
RFM_DB.groupby(['Labels']).count()
```

Labels	Recency	Frequency	MonetaryValue
-1	4	4	4
0	3873	3873	3873
1	21	21	21

```
RFM_DB.groupby('Labels').agg({
    'Recency':'mean',
    'Frequency':'mean',
    'MonetaryValue':['mean', 'count']}).round(2)
```

Labels	Recency	Frequency	MonetaryValue	
	mean	mean	mean	count
-1	167.50	9.75	29.75	4
0	185.89	3.85	9.97	3873
1	710.90	1.10	2.43	21

	ID	Cluster	Attribute	Value
0	1000	0	Recency	36
1	1001	0	Recency	243
2	1002	0	Recency	123
3	1003	0	Recency	324
4	1004	0	Recency	29

- The dominant cluster (Label 0), encompassing most of the dataset, likely represents the general customer base with average engagement and spending. For marketers, this segment may be the primary target for standard marketing campaigns, with the potential for improved customer retention programs.
- The minor cluster (Label 1) with high Recency could represent at-risk customers who have not engaged recently. This group offers a critical business opportunity for re-engagement strategies, possibly through personalized offers or loyalty incentives to increase their lifetime value.
- Outliers (Label -1), although few, display high Frequency and MonetaryValue, marking them as high-value, possibly lapsed loyal customers. They could be targeted for win-back campaigns, leveraging their past high engagement to maximize customer lifetime value.

Business Value Justification:

The business value of each cluster is justified by its RFM characteristics:

- For the primary cluster, maintaining the base through loyalty programs could increase overall lifetime value and stabilize revenue.
- The minor cluster's high Recency underscores an urgent need for re-engagement tactics to prevent further attrition.

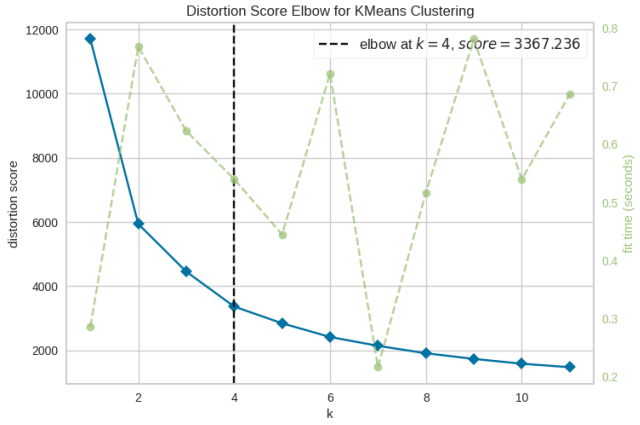
- The outliers present a unique opportunity; despite their small number, their high engagement levels in the past suggest a potential for high returns on re-engagement efforts.

Conclusion:

DBSCAN segmentation identified three main customer groups: a primary cluster for ongoing engagement, a smaller segment for reactivation efforts, and high-value outliers for targeted recovery. Despite its insights for tailoring loyalty initiatives, DBSCAN's poor differentiation between groups—evidenced by a large primary cluster and a negative silhouette score—suggests it may not be ideal for this dataset. Testing with varied eps and min_samples confirmed its limitations, as adjustments did not significantly improve segmentation. The predominance of one cluster and a negative silhouette score of -0.217 indicate that DBSCAN is unsuitable for effective marketing segmentation here. Exploring alternative clustering methods could better differentiate customer segments and enhance marketing strategies.

4.2. Testing with another clustering algorithm – Agglomerative Clustering with Optimal Cluster Analysis

What is the optimal number of clusters for this dataset? Lets check this with Elbow graph.

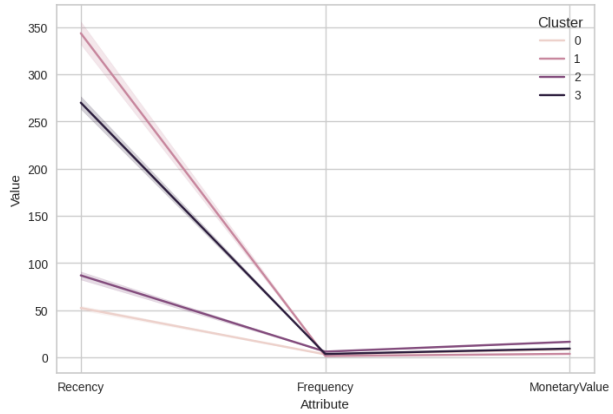


The elbow method graphically determines the optimal number of clusters. By plotting the sum of squared distances (distortion score) from each point to its assigned cluster center against different k values, we observe a decrease in distortion as k increases. The "elbow" point, where the rate of decrease sharply changes, indicates the optimal k. In this dataset, the elbow is identified at k=4 with a distortion score of 3367.236, suggesting that four clusters provide a good balance between compactness and separation without unnecessary complexity.

```
data1 = members.copy()
from sklearn.cluster import AgglomerativeClustering
hc = AgglomerativeClustering(n_clusters = 4, metric = 'euclidean', linkage = 'ward')
y_hc = hc.fit_predict(data1)
df2=data1[["Recency","Frequency","MonetaryValue"]]
df2["label"] = y_hc
df2
```

Member_number	Recency	Frequency	MonetaryValue	label
1000	36	5	13	2
1001	243	5	12	0
1002	123	4	8	0
1003	324	4	8	1
1004	29	8	21	2
...	...	...	...	...
4996	37	3	10	2
4997	4	2	6	2
4998	78	1	2	2

We now know the optimal number of clusters for this data. Then we will build the Agglomerative



Clustering algorithm and test it.

The graph represents the average values of Recency, Frequency, and MonetaryValue for four clusters determined by Agglomerative Clustering, based on the optimal number of clusters identified via the elbow method. Each line corresponds to one cluster, showing distinct behavior patterns:

Cluster 0 shows high Recency and lower Frequency and MonetaryValue, indicating these are infrequent and recent customers with lower spending.

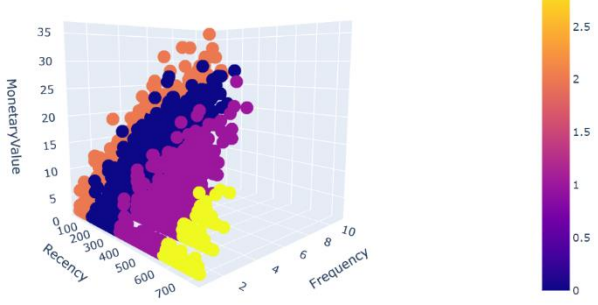
Cluster 1 exhibits moderate Recency and MonetaryValue but lower Frequency, representing occasionally purchasing customers.

Cluster 2 maintains very low Recency, Frequency, and MonetaryValue, suggesting these customers are the least engaged.

Cluster 3 indicates customers with moderately low Recency and higher MonetaryValue, likely representing loyal and valuable customers.

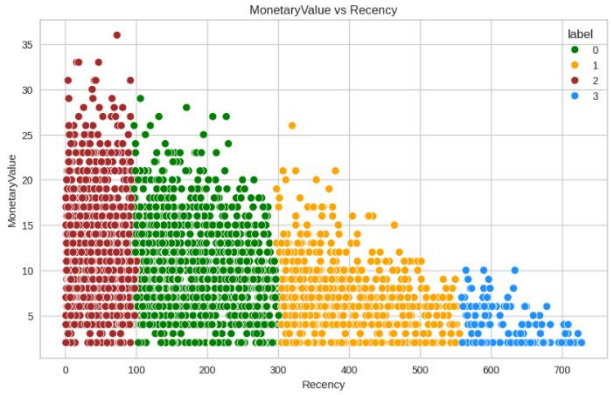
The silhouette score of 0.54 for this clustering suggests a reasonably good fit, where clusters are well apart from each other and cohesive internally. This score indicates effective segmentation of the customer base into meaningful groups that could be targeted differently according to their purchasing behaviors.

3D Plot Visualization:



The 3D scatter plot effectively displays the segmentation of customers based on Recency, Frequency, and MonetaryValue. Each color represents a distinct cluster, visually capturing the difference in customer behavior dimensions. The plot provides a clear separation between the clusters, with distinct areas occupied by each group, reflecting varied customer engagement levels.

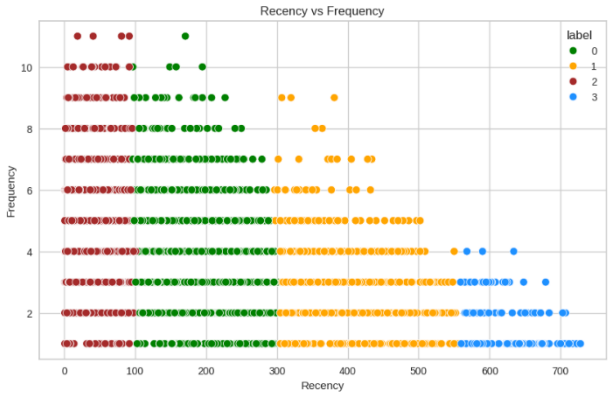
Recency vs MonetaryValue Plot:



This 2D scatter plot shows each customer's Monetary Value against their Recency. The cluster colors differentiate customer groups with unique behavioral patterns, such as those who recently made high-value purchases versus those with older, less valuable interactions.

Recency vs Frequency Plot

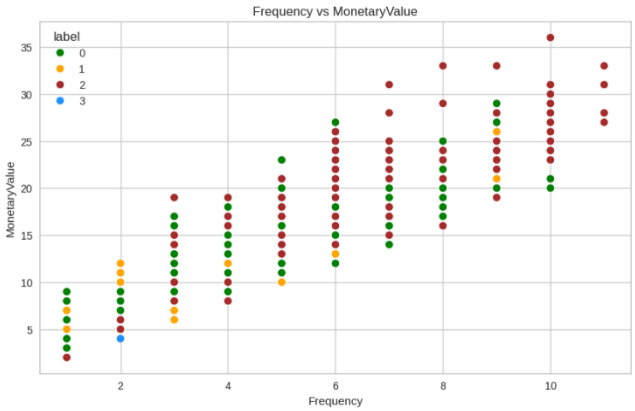
This visualization categorizes customers by their purchase frequency against the recency of their last purchase. Clusters show distinct patterns, for example, frequent recent purchasers versus occasional long-time customers, offering insights into customer retention and engagement strategies



Frequency vs MonetaryValue Plot:

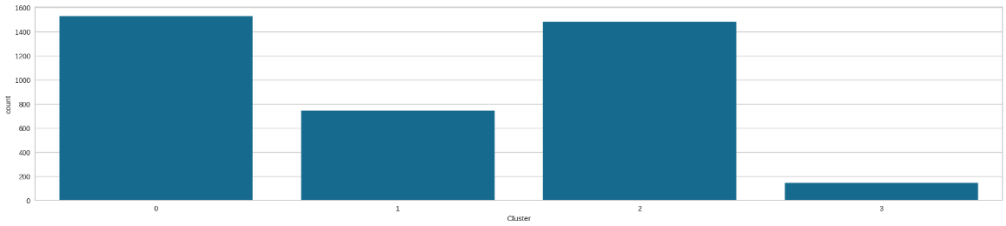
The scatter plot on the right correlates customer purchase frequency with spending, highlighting valuable segments like high-frequency, high-spending customers, which are crucial for targeted marketing efforts to enhance customer value.

Conclusion and Suitability:



Agglomerative Clustering demonstrates clear and meaningful segmentation in this dataset, overcoming the limitations observed with DBSCAN, which struggled with parameter sensitivity and often resulted in overgeneralized clusters. The silhouette score of 0.54 in Agglomerative Clustering suggests a moderately good separation and cohesion within clusters, making it a more suitable choice for actionable customer segmentation. This method's ability to form well-differentiated clusters based on RFM attributes is crucial for designing targeted marketing strategies and enhancing customer relationship management.

4.3. Review of Results in terms of business value for Agglomerative Clustering algorithm



Cluster	Recency	Frequency	MonetaryValue
0	1527	1527	1527
1	743	743	743
2	1483	1483	1483
3	145	145	145

	Recency	Frequency	MonetaryValue	
	mean	mean	mean	count
Cluster				
0	52.44	3.31	8.53	874
1	343.46	1.61	3.90	845
2	86.98	6.22	16.66	1031
3	269.89	3.74	9.44	1148

Cluster 0 (Loyal Regulars): Characterized by the lowest recency (52.44 days on average), higher frequency (3.31), and moderate spending (8.53). This group frequently shops and engages recently, indicating high customer loyalty. Strategies could focus on reward programs to further increase their lifetime value.

Cluster 1 (Churning Low Spenders): High recency (343.46 days), low frequency (1.61), and low spending (3.90). These are at-risk customers who shop infrequently and spend the least, suggesting a disengagement risk. Targeted re-engagement campaigns with special offers might prevent churn.

Cluster 2 (High-value Customers): Moderately recent engagement (86.98 days), highest frequency (6.22), and highest monetary contributions (16.66). This group represents the most valuable customers who should be the focus for up-sell and cross-sell opportunities through exclusive promotions and loyalty incentives.

Cluster 3 (Infrequent Recent Spenders): Recent engagement (269.89 days), moderate frequency (3.74), and higher spend (9.44) than Clusters 0 and 1. Strategies for this segment might include personalized marketing to increase visit frequency, leveraging their willingness to spend.

Each cluster's distinct behavior presents opportunities to tailor marketing efforts that cater to their specific needs and buying patterns, thus potentially increasing overall customer loyalty and lifetime value.

5. Data Mart Design

For designing a data mart that aligns with the insights gained from RFM analysis and Agglomerative Clustering, we focus on structuring data around key dimensions that reflect the aspects of customer behavior and interactions. Here's a suggested framework:

Identification of Dimensions

1. **Customer ID**: Unique identifier for each customer (based on Member numbers).
2. **Transaction Date**: Time dimension to capture the Date of transactions.
3. **Product Category**: Categorization of each item (derived from Item Description).

Justification of Selected Dimensions

- **Customer ID**: This dimension allows for individual-level analysis of shopping patterns, loyalty, and value, essential for personalized marketing.
- **Transaction Date**: The temporal aspect is vital for identifying trends, seasonality, and the efficacy of marketing campaigns over time.
- **Product Category**: Understanding the performance of different product categories helps tailor product-specific strategies and inventory management.

Identification of Measures

1. **Recency Score**: Reflects the time since the last transaction.
2. **Frequency Score**: Indicates the regularity of transactions.
3. **Monetary Value Score**: Represents the total spend by the customer.

Justification of Selected Measures

- **Recency Score**: Offers insight into the likelihood of a customer returning, crucial for retention strategies.
- **Frequency Score**: Helps in recognizing and rewarding loyalty, crucial for formulating frequency-based incentives.
- **Monetary Value Score**: Identifies high-value customers who contribute significantly to revenue, informing strategies for upselling and cross-selling.

The data mart structured around these dimensions and measures will equip the marketing department with the tools necessary to drive customer-centric decision-making. It will support targeted marketing initiatives, foster customer loyalty, and potentially increase customer lifetime value by enabling a deep dive into customer transaction behaviors and value segmentation. This data mart is designed not only to capture a snapshot of the current customer base but also to monitor and respond to changes in customer behavior over time.

Reference List

1. scikit-learn 2023, *AgglomerativeClustering*, viewed 10 April 2024, https://scikit-learn.org/stable/modules/generated/sklearn.cluster.AgglomerativeClustering.html.

2. scikit-learn 2023, *StandardScaler*, viewed 10 April 2024, https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html.

3. Data Science Stack Exchange 2023, *How to handle outliers for clustering algorithms*, viewed 15 April 2024, https://datascience.stackexchange.com/questions/63695/how-to-handle-outliers-for-clustering-algorithms#:~:text=If%20you%20have%20outliers%2C%20the,approach%20is%20sensitive%20to%20outliers.

4. Voleti, D. 2023, *DBSCAN algorithm for fraud detection & outlier detection in a data set*, Medium, viewed 15 April 2024, https://medium.com/@dilip.voleti/dbscan-algorithm-for-fraud-detection-outlier-detection-in-a-data-set-60a10ad06ea8.

5. Pierian Training 2023, *DBSCAN for outlier detection in Python and scikit-learn: machine learning in Python*, viewed 16 April 2024, https://pieriantraining.com/dbscan-for-outlier-detection-in-python-and-scikit-learn-machine-learning-in-python/#:~:text=In%20summary%2C%20DBSCAN%20is%20a,to%20any%20cluster%20as%20outliers.

6. scikit-learn 2023, *DBSCAN clustering example*, viewed 20 April 2024, https://scikit-learn.org/stable/auto_examples/cluster/plot_dbscan.html.

7. scikit-learn 2023, *DBSCAN*, viewed 20 April 2024, https://scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.html#.

8. Codex 2023, *K-means clustering explained*, Medium, viewed 30 April 2024, https://medium.com/codex/k-means-c5763e50898e.

9. Yellowbrick 2023, *KElbowVisualizer*, viewed 30 April 2024, https://www.scikit-yb.org/en/latest/api/cluster/elbow.html.

10. Mullin, T. 2023, *DBSCAN parameter estimation*, Medium, viewed 30 April 2024, https://medium.com/@tarammullin/dbscan-parameter-estimation-ff8330e3a3bd.

11. JetBrains Academy 2023, *Machine learning with scikit-learn*, viewed 30 April 2024, https://hyperskill.org/learn/step/31529.