Online Retail Dataset

RFM Analysis

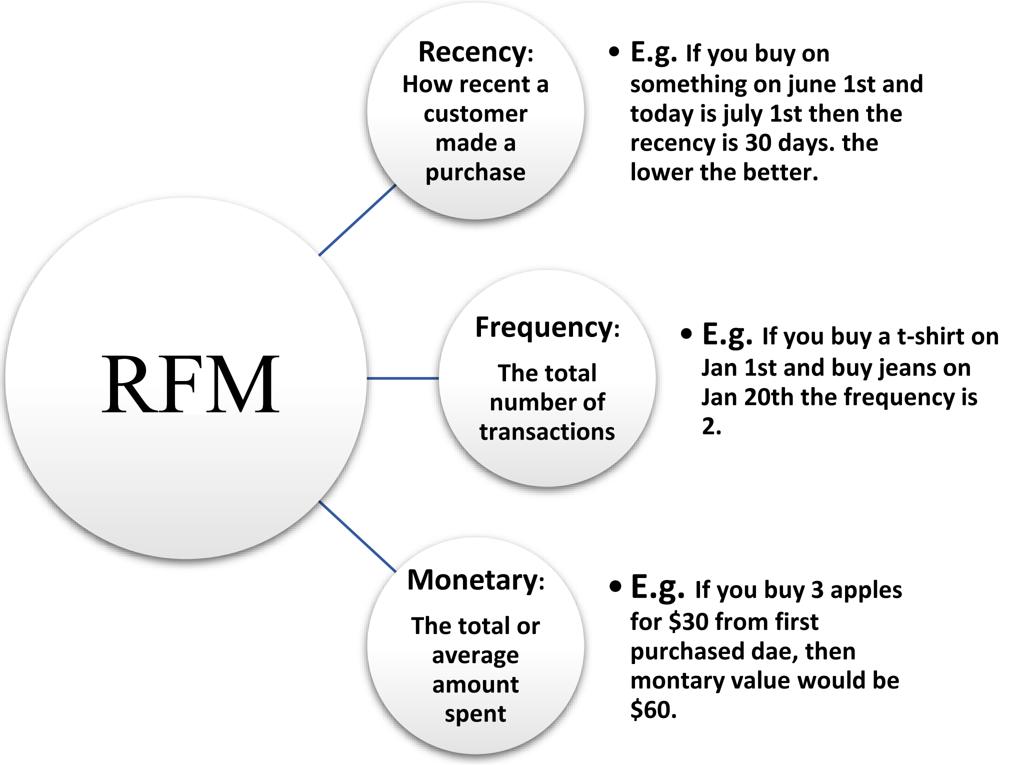
Report

Mubeen Arshad
10-31-2023

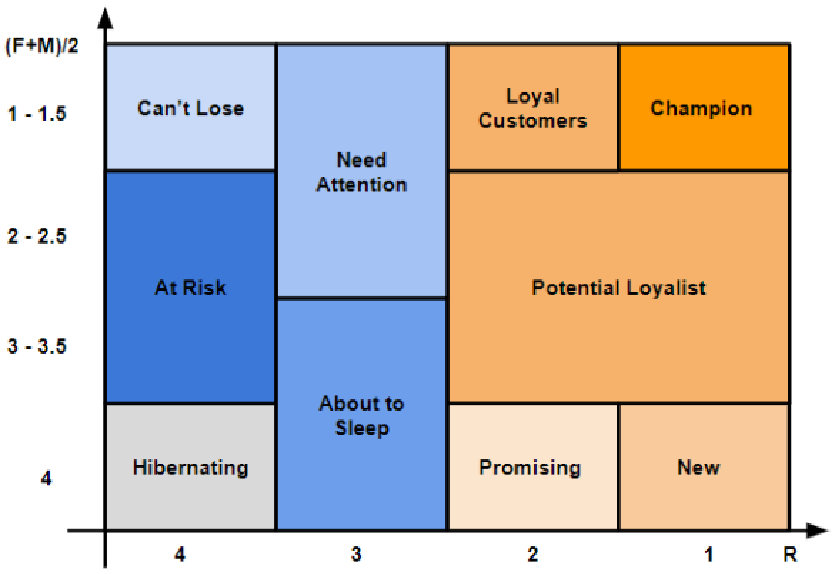
Throughout this report I will discuss and deep dive into RFM analysis. In the end, I will share my insights and recommendations.

RFM Analysis:

The RFM analysis is a customer segmentation technique that categorizes the customers based on three factors which are, Recency Frequency, and Monetary. Categorizing the customer on an RFM basis helps the company to better understand customer behavior and tailor marketing strategies accordingly.



RFM Distribution:



The above chart shows that the x-axis has a recency score, and the y-axis has a frequency and monetary score consecutively. The customers' recency, frequency, and monetary are ranked from 1 to 4, the top 25% of customers are considered as 1, the 50% are considered as 2, and so on.

About Data:

This is a transnational data set that contains all the transactions occurring between 01/12/2010 and 09/12/2011 for a UK-based and registered non-store online retail. The company mainly sells unique all-occasion gifts. Many customers of the company are wholesalers.

Has Missing Values? Yes, 0.26% in the Description column.

Instances: 541909

Features: 8

Data Dictionary:

Variable Name	Role	Type	Description	Units	Missing values
InvoiceNo	ID	Categorical	a 6-digit integral number uniquely assigned to each transaction. If this code starts with the letter 'c', it indicates a cancellation		no
StockCode	ID	Categorical	a 5-digit integral number uniquely assigned to each distinct product		no
Description	Feature	Categorical	product name		no
Quantity	Feature	Integer	the quantities of each product (item) per transaction		no
InvoiceDate	Feature	Date	the day and time when each transaction was generated		no
UnitPrice	Feature	Continuous	product price per unit	sterling	no
CustomerID	Feature	Categorical	a 5-digit integral number uniquely assigned to each customer		no
Country	Feature	Categorical	the name of the country where each customer resides		no

Data Analysis Approach/Process:

Data Gathering:

Online Retail data was gathered from an online resource.

Data Importing:

Imported the data in a web-based interactive computing environment "Jupyter Notebook" launched through Anaconda for further analysis.

Data Cleaning and transformation:

After checking the statistical summary of the dataset, it was observed that both the Quantity and UnitPrice columns contain negative values, while the Description Columns consist of null values. The negative and null values have been effectively removed from the dataset. Notably, the eliminated rows collectively constituted only **2.07%** of the entire dataset.

```
df.describe().T
```

	count	mean	std	min	25%	50%	75%	max
Quantity	541909.000	9.552	218.081	-80995.000	1.000	3.000	10.000	80995.000
UnitPrice	541909.000	4.611	96.760	-11062.060	1.250	2.080	4.130	38970.000
CustomerID	541909.000	15287.518	1484.746	12346.000	14367.000	15287.000	16255.000	18287.000

Changing Data Types:

The **'InvoiceDate'** column, initially classified as an object data type, has been efficiently transformed into a datetime64 data type, with a new format of **'%Y-%m-%d %H:%M:%S'** for enhanced readability and consistency. Additionally, the **'CustomerID'** column, previously recorded as a float data type, has been appropriately converted into an object data type, aligning it with the data representation requirements.

```
#changing invoice date data type

df1['InvoiceDate'] = pd.to_datetime(df1['InvoiceDate'], format='%Y-%m-%d %H:%M:%S')

# changing customer id data type
df1.loc[:, "CustomerID"] = df1["CustomerID"].astype(np.int64).astype(object)

```

Calculated TotalCost:

Calculated the 'TotalCost' from each transaction, that will be needed for RFM calculations.

```
# finding total cost

df1["TotalCost"] = df1["Quantity"] * df1["UnitPrice"]
```

Main Objective:

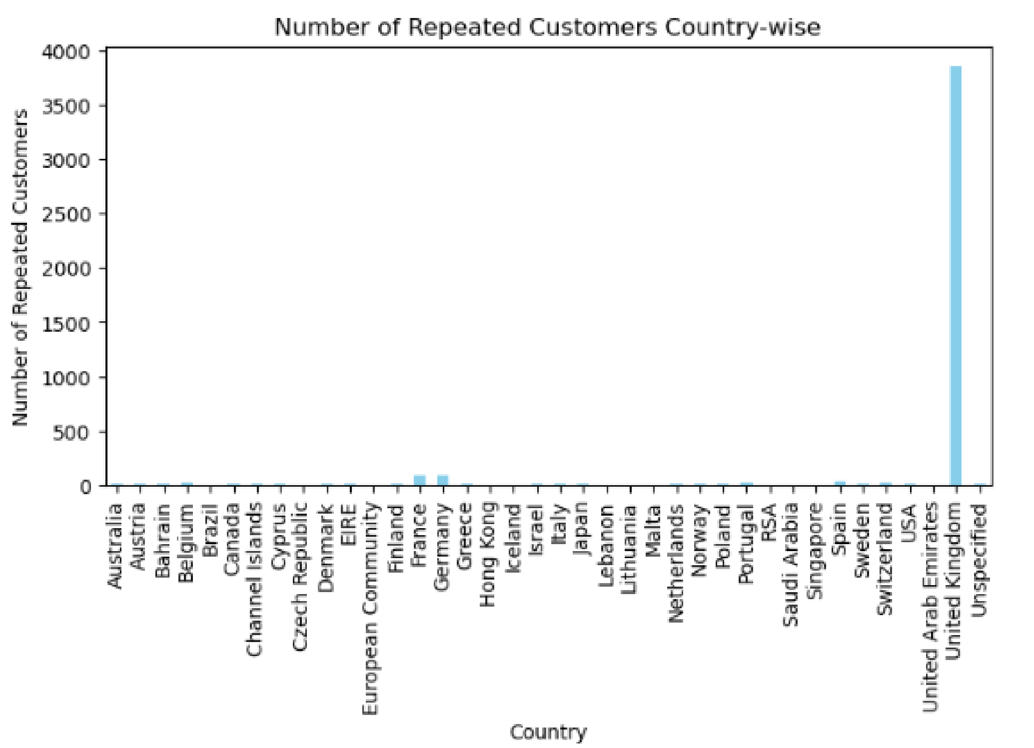
Initially, we will compute the RFM (Recency, Frequency, and Monetary) values and corresponding ranges for individual customers. After that, we'll dive into the RFM analysis to get a clearer picture of the general behavior exhibited by customers during their online shopping experiences.

Analysis:

Country Breakdown by Repeated Customers:

Firstly, I have checked which country has the most repeat customers, certainly, it is performed to gain insights into the specific behavioral trends that contribute to customer loyalty within that market.

```
# check which country has more repeated customers
repeated_customers = df1[df1.duplicated(subset = ['CustomerID'], keep = False)]
repeated_customers_country_wise = repeated_customers.groupby('Country')['CustomerID'].nunique()
# Plot the results
plt.figure(figsize = (8, 4))
repeated_customers_country_wise.plot(kind='bar', color='skyblue')
plt.title('Number of Repeated Customers Country-wise')
plt.xlabel('Country')
plt.ylabel('Number of Repeated Customers')
plt.show()
```



I will further analyze the RFM on 'United Kingdom'.

Total Time Frame for Country UK:

I have checked the total time frame for the country UK.

```
# total timeframe of dataset
print('----Time Frame----')
print('Start Date:', df1['InvoiceDate'].min())
print('End Date:', df1['InvoiceDate'].max())
print('Number of customers within the time frame in the UK:', df1['CustomerID'].nunique())
```

```
----Time Frame----
Start Date: 2010-12-01 08:26:00
End Date: 2011-12-09 12:49:00
Number of customers within the time frame in the UK: 3921
```

Setting the Time Frame:

First, need to set the time frame for RFM further analysis. And for that, I chose the last 6 months period of June 2011 to Dec 2011 from the whole and checked the total number of customers within that time frame. For example, if the customer transaction was before June 2011, it was considered as churned.

```
# Given start and end dates
start_date = dt.datetime(2010, 12, 1)
end_date = dt.datetime(2011, 12, 9)
from dateutil.relativedelta import relativedelta
# Calculate the start date of the 6-month window
start_date_window = end_date - relativedelta(months = 6)
# Filter the DataFrame based on the date range
rfm_df = df1[(df1['InvoiceDate'] >= start_date_window) & (df1['InvoiceDate'] <= end_date)]
# Get the count of unique customers in the filtered DataFrame
num_customers = rfm_df['CustomerID'].nunique()
print("Start date of the last 6 months:", start_date_window)
print("End date:", end_date)
print("Number of customers within the timeframe: ", num_customers)
```

```
Start date of the last 6 months: 2011-06-09 00:00:00
End date: 2011-12-09 00:00:00
Number of customers within the timeframe:  3159
```

RFM calculations:

In RFM calculations, to calculate the recency we need the latest purchase date for each customer, for frequency, we need the count of unique transactions, and in monetary we need the total cost/ sales of each customer. So for that purpose, rfm() function was created and computed the recency, frequency and monetary.

```
# function for RFM calculation
def rfm(df1):
    recency = df1.groupby('CustomerID').agg(Latest_Pur_Date=('InvoiceDate', np.max)).reset_index()
    frequency = df1.groupby('CustomerID')['InvoiceDate'].nunique().reset_index()
```

```
    monetary = df1.groupby('CustomerID')['TotalCost'].sum().reset_index()
    # merging r,f,m df and renaming the columns
    df1 = pd.merge(recency, frequency, on = ['CustomerID'])
    df1 = pd.merge(df1, monetary, on = ['CustomerID'])
    df1 = df1.rename(columns = {'Latest_Pur_Date': 'LatestPurchaseDate', 'InvoiceDate': 'TransactionCount',
'TotalCost':'TotalSpending'})
    df1['AvgSpending'] = df1['TotalSpending'] / df1['TransactionCount']
    df1['LatestPurchaseDays'] = (df1['LatestPurchaseDate'].max() - df1['LatestPurchaseDate']).dt.days
    return df1
```

Testing the function

```
rfm_test = rfm(rfm_df)
rfm_test.sample(5)
```

	CustomerID	LatestPurchaseDate	TransactionCount	TotalSpending	AvgSpending	LatestPurchaseDays
2531	17190	2011-10-12 11:33:00	1	249.74	249.740000	57
2071	16401	2011-12-08 18:15:00	7	2436.43	348.061429	0
504	13692	2011-11-15 11:17:00	1	616.74	616.740000	23
2944	17904	2011-11-18 12:57:00	1	191.67	191.670000	20
2797	17653	2011-11-03 12:46:00	4	1207.56	301.890000	35

Calculating Quantiles:

Now we need to calculate the quantiles after computing the RFM values. In this section, we will rank the customers from 1 to 4. The first 25% will be ranked as 1, the next 25% will be ranked as 2, and so on. And for that, we need to computer percentiles of 25th, 50th, 75th, and 100th. We will also check how to identify the percentile for customer ranking.

```
# calculaiting quantiles
rfm_quan =
rfm_test['TransactionCount'].value_counts().rename_axis('TransactionCount').reset_index(name='counts')
rfm_quan['cumulative_sum'] = rfm_quan['counts'].cumsum()
rfm_quan['cumulative_perc'] = 100 * rfm_quan['cumulative_sum'] / rfm_quan['counts'].sum()

# making categories from quantiles
def rfm_quantile(df1):
    # getting the cumulative precentile so customers can be ranked accordingly
    rfm_freq_cum =
df1['TransactionCount'].value_counts().rename_axis('TransactionCount').reset_index(name='counts')
    rfm_freq_cum['cumulative_sum'] = rfm_freq_cum['counts'].cumsum()
    rfm_freq_cum['cumulative_perc'] = rfm_freq_cum['cumulative_sum'] / rfm_freq_cum['counts'].sum()
    #using qcut to categorize the customers from 1 to 4.
    df1["Recency"] = pd.qcut(df1["LatestPurchaseDays"], q=[0,.25,.5,.75,1], labels=['1','2','3','4'])
    df1["Frequency"] = pd.qcut(df1["TransactionCount"],
q=[0,rfm_freq_cum['cumulative_perc'][0],rfm_freq_cum['cumulative_perc'][1],rfm_freq_cum['cumulative_perc'][
2],1], labels=['4','3','2','1'])
```

```
    df1["Monetary"] = pd.qcut(df1["AvgSpending"], q=[0,.25,.5,.75,1], labels=['4','3','2','1'])
    df1['RFMLabel'] = df1[['Recency', 'Frequency', 'Monetary']].agg(''.join, axis=1)
    #this is not neccessary but just to illustrate the raw range values based on the qcut
    df1["RecencyRange"] = pd.qcut(df1["LatestPurchaseDays"], q=[0,.25,.5,.75,1])
    df1["FrequencyRange"] = pd.qcut(df1["TransactionCount"],
q=[0,rfm_freq_cum['cumulative_perc'][0],rfm_freq_cum['cumulative_perc'][1],rfm_freq_cum['cumulative_perc'][2],1])
    df1["MonetaryRange"] = pd.qcut(df1["AvgSpending"], q=[0,.25,.5,.75,1])
    df1['FxMLabel'] = (pd.to_numeric(df1['Frequency']) + pd.to_numeric(df1['Monetary'])) / 2
    df1['FxMLabel'] = df1['FxMLabel']
    df1['Recency'] = df1['Recency'].astype(int)
    return df1
```

Testing the function:

```
rfm_test = rfm_quantile(rfm_test)
rfm_test.sample(5)
```

Count	TotalSpending	AvgSpending	LatestPurchaseDays	Recency	Frequency	Monetary	RFMLabel	RecencyRange	FrequencyRange	MonetaryRange	FxMLabel
2	2292.20	1146.100000	51	3	3	1	331	(32.0, 71.0]	(1.58, 2.369]	(431.907, 13305.5]	2.0
1	230.51	230.510000	32	2	4	3	243	(13.0, 32.0]	(0.999, 1.58]	(180.088, 296.213]	3.5
7	2896.31	413.758571	8	1	1	2	112	(-0.001, 13.0]	(3.243, 959.0]	(296.213, 431.907]	1.5
2	778.54	389.270000	97	4	3	2	432	(71.0, 182.0]	(1.58, 2.369]	(296.213, 431.907]	2.5
1	100.80	100.800000	70	3	4	4	344	(32.0, 71.0]	(0.999, 1.58]	(-0.001, 180.088]	4.0

Creating Categories:

First, there are main categories of 'Hot' and 'Cold'. And there are other RFM categories of 'Champion, Loyal Customer, Potential Loyalist, Promising, New, Sleep, Hibernating, At Risk, Cant Lose, Need Attention'.

The Hot leads include Champions, Loyal Customer, Potential Loyalist, Promising, and New.

The Cold leads include Sleep, At Risk, Can't Loose, Hibernating, and Need Attention.

```
# categories function
def rfm_categories(df1):
    # assigning rfm labels
    df1['RFMCategory'] = np.select(
        [
            (df1['Recency'] == 1) & (df1['FxMLabel'] >= 1) & (df1['FxMLabel'] <= 1.5),
            (df1['Recency'] == 2) & (df1['FxMLabel'] >= 1) & (df1['FxMLabel'] <= 1.5),
            (df1['Recency'].between(1, 2)) & (df1['FxMLabel'].between(2, 2.5)),
            (df1['Recency'] == 1) & (df1['FxMLabel'] >= 3) & (df1['FxMLabel'] <= 4),
            (df1['Recency'] == 2) & (df1['FxMLabel'] >= 3) & (df1['FxMLabel'] <= 4),
            (df1['Recency'] == 3) & (df1['FxMLabel'] >= 1.5) & (df1['FxMLabel'] <= 2.5),
            (df1['Recency'] == 3) & (df1['FxMLabel'] >= 3) & (df1['FxMLabel'] <= 4),
            (df1['Recency'].between(3, 4)) & (df1['FxMLabel'] >= 1) & (df1['FxMLabel'] <= 1.5),
```

```
        (df1['Recency'] == 4) & (df1['FxMLabel'] == 4),
        (df1['Recency'] == 4) & (df1['FxMLabel'] >= 2) & (df1['FxMLabel'] <= 3.5)
     ],
     [
        'Champion',
        'Loyal Customer',
        'Potential Loyalist',
        'New',
        'Promising',
        'Need Attention',
        'Sleep',
        'Can\'t Loose',
        'Hibernating',
        'At Risk'
     ],
     default = 'Unknown'
  )
  # main categories
  hot = ['Champion', 'Loyal Customer', 'Potential Loyalist', 'New', 'Promising']
  cold = ['Need Attention', 'Sleep', 'Can\'t Loose', 'Hibernating', 'At Risk'
  df1['RFMMainCategory'] = np.where(df1['RFMCategory'].isin(hot), 'Hot', 'Cold')
  return df1
```

Testing the function:

```
rfm_categories(rfm_test)
rfm_test.sample(5)
```

LatestPurchaseDays	Recency	Frequency	Monetary	RFMLabel	RecencyRange	FrequencyRange	MonetaryRange	FxMLabel	RFMCategory	RFMMainCategory
18	2	4	3	243	(13.0, 32.0]	(0.999, 1.58]	(180.088, 296.213]	3.5	Promising	Hot
25	2	2	4	224	(13.0, 32.0]	(2.369, 3.243]	(-0.001, 180.088]	3.0	Promising	Hot
21	2	1	2	212	(13.0, 32.0]	(3.243, 959.0]	(296.213, 431.907]	1.5	Loyal Customer	Hot
3	1	1	4	114	(-0.001, 13.0]	(3.243, 959.0]	(-0.001, 180.088]	2.5	Potential Loyalist	Hot
44	3	4	1	341	(32.0, 71.0]	(0.999, 1.58]	(431.907, 13305.5]	2.5	Need Attention	Cold

EDA of RFM:

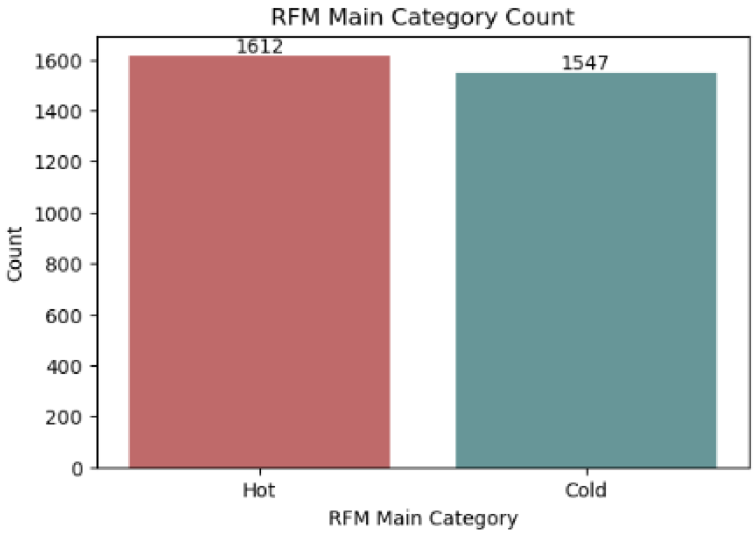
In this section, we will explore customers' general behavior while online shopping.

Count of customers in Main categories:

First, check the number of customers in the main categories of hot and cold.

```
# RFM visualization
plt.figure(figsize=(6, 4))
rfm_main = rfm_test['RFMMainCategory'].value_counts().rename_axis('RFM_Main').reset_index(name='count')
# custome rcolor palette
```

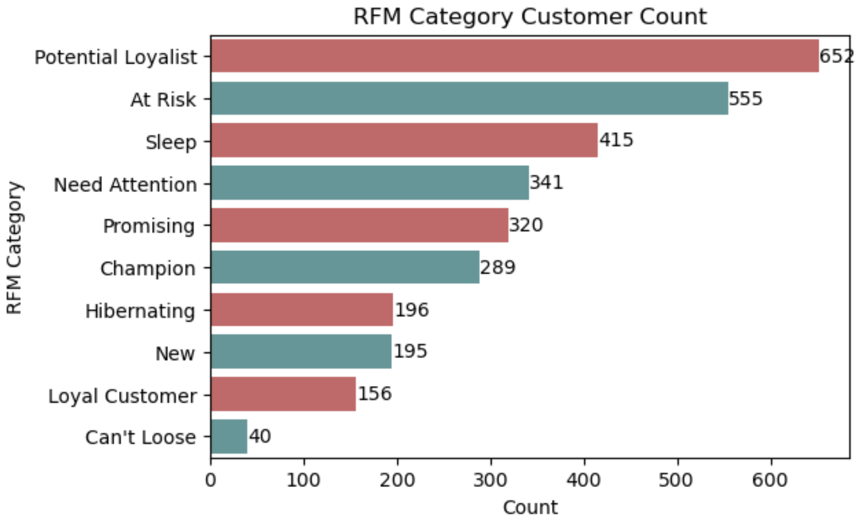
```
custom_palette = ["indianred" , "cadetblue"]  # Example colors, customize as needed
ax = sns.barplot(data = rfm_main, x = "RFM_Main", y = "count", palette = custom_palette)
plt.title('RFM Main Category Count')
plt.xlabel('RFM Main Category')
plt.ylabel('Count')
# Adding text labels on each bar
for container in ax.containers:
    ax.bar_label(container, label_type = 'edge', fontsize = 10)
plt.show()
```



RFM Other Categories:

Now, visualize the other categories to get the proper insights.

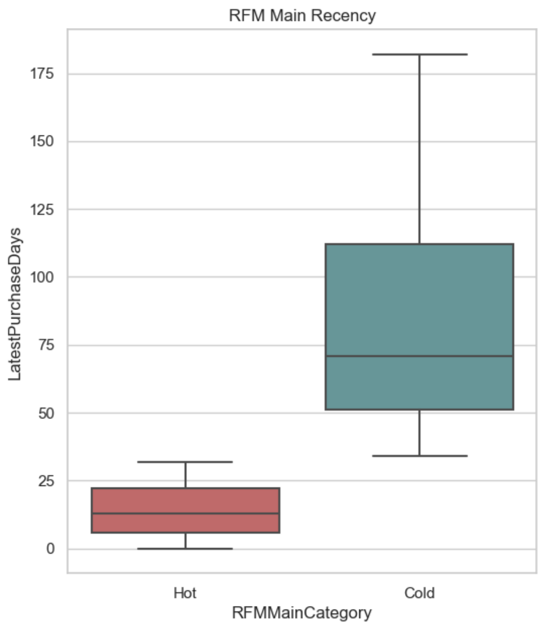
```
 # RFM categories
plt.figure(figsize=(6, 4))
rfm_cat = rfm_test['RFMCategory'].value_counts().rename_axis('RFM_Cat').reset_index(name='count')
# Custom color palette
custom_palette = ["indianred" , "cadetblue"]  # Example colors, customize as needed
ax = sns.barplot(data=rfm_cat, x="count", y="RFM_Cat", palette=custom_palette)
plt.title('RFM Category Customer Count')
plt.xlabel('Count')
plt.ylabel('RFM Category')
# Adding text labels to each bar
for i in range(len(rfm_cat)):
    plt.text(rfm_cat['count'][i], i, str(rfm_cat['count'][i]), va='center', fontsize = 10)
plt.show()
```



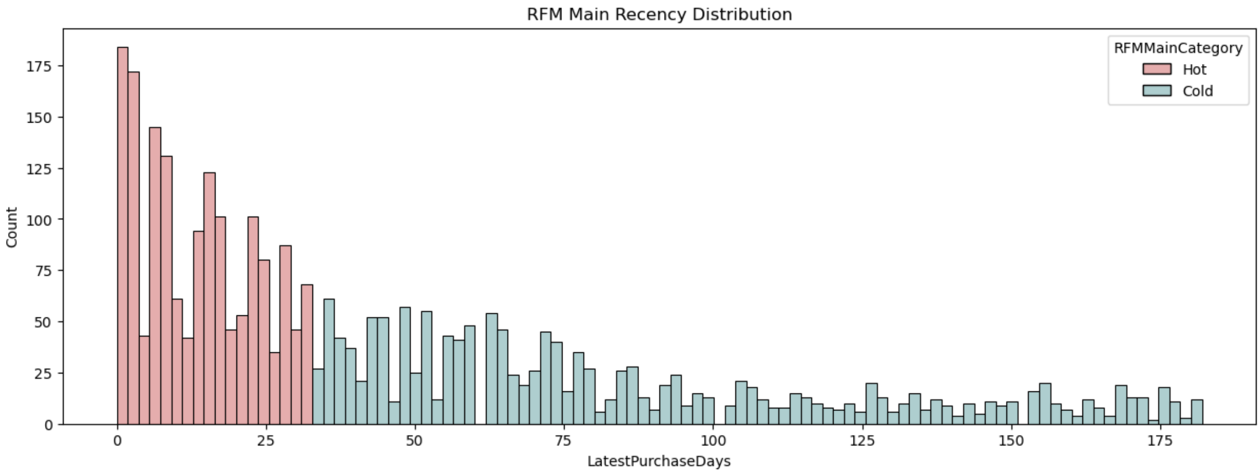
Insights:
The first graph shows there is not a huge difference between hot and cold customers as hot customers are 1612 and cold customers have a customer count of 1547. On the other side, the significant count of customers lies in the potential loyalist followed by the 'At Risk' category.

RFM Main Recency:
Here I visualize the recency of main categories using a box plot and then use the histogram to count the recency distribution for customer count.

```
custom_palette = ["indianred" , "cadetblue"]  # Example colors, customize as needed
# Box plot
plt.figure(figsize = (6, 7))
ax = sns.boxplot(data = rfm_test, y = "LatestPurchaseDays", x = "RFMMainCategory", palette = custom_palette)
plt.title("RFM Main Recency")
plt.show()
```



```
# histogram
custom_palette = ["indianred" , "cadetblue"]  # Example colors, customize as needed
plt.figure(figsize=(15, 5))
sns.histplot(data = rfm_test, x = "LatestPurchaseDays", hue = "RFMMainCategory", bins = 100, palette =
custom_palette)
plt.title("RFM Main Recency Distribution")
plt.show()
# Grouping by RFM Main Category
rfm_recency = rfm_test.groupby('RFMMainCategory')[['LatestPurchaseDays']].describe().reset_index()
```



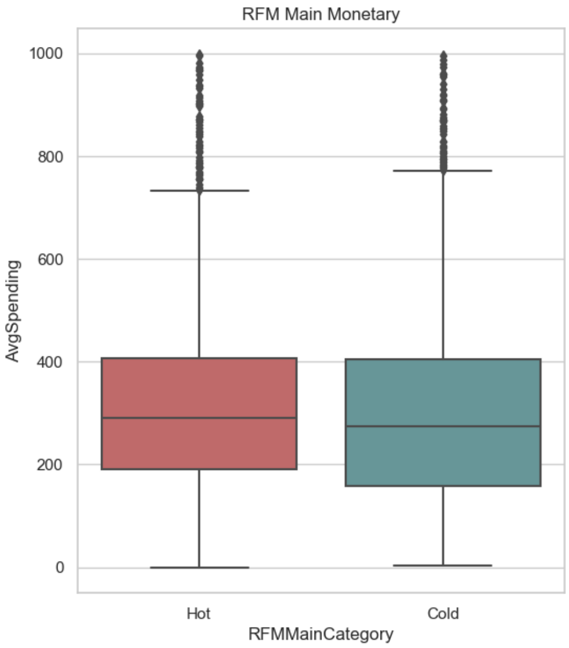
Insights And Recommendations:

After approximately 35 days, customers show a shift towards the 'Cold' category, signifying a decline in their engagement. Additionally, the mean recency for 'Hot' users is 13.0 days, indicating frequent interactions within a relatively short timeframe.

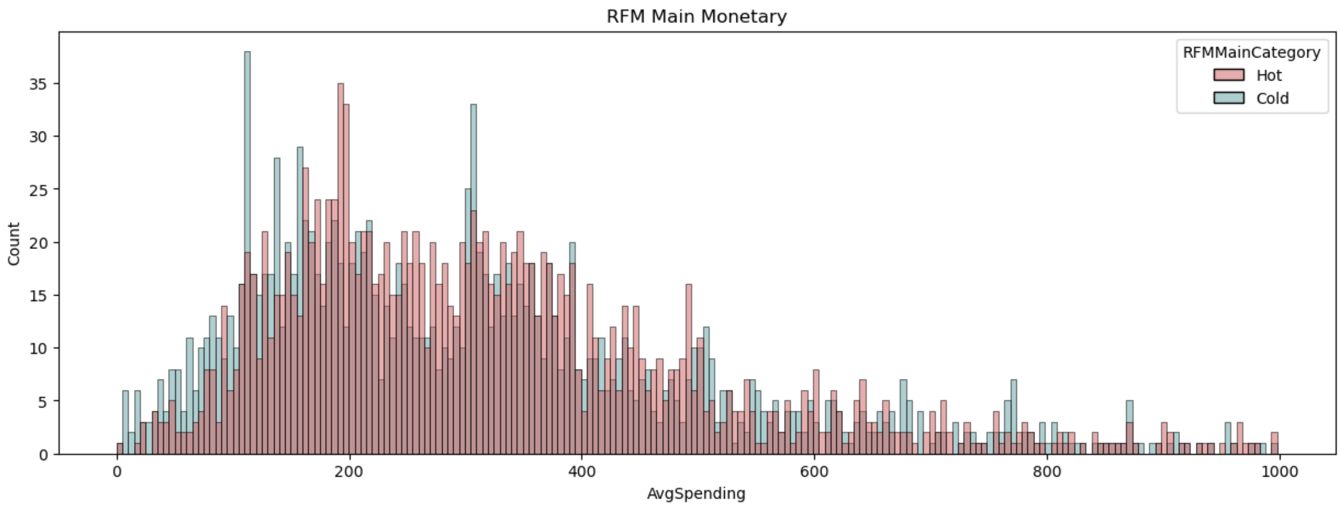
This can serve as a benchmark or criteria to concentrate engagements during these periods, aimed at retaining customers.

RFM Main Monetary:

```
# main frequency category
custom_palette = ["indianred" , "cadetblue"]  # Example colors, customize as needed
# Box plot with data labels
plt.figure(figsize = (6, 7))
ax = sns.boxplot(data = rfm_test[rfm_test['AvgSpending'] < 1000], y = "AvgSpending", x = "RFMMainCategory",
palette = custom_palette)
plt.title("RFM Main Monetary")
plt.show()
```




```
# Histogram plot
plt.figure(figsize = (15, 5))
sns.histplot(data = rfm_test[rfm_test['AvgSpending'] < 1000], x = "AvgSpending", hue = "RFMMainCategory", bins = 200, palette = custom_palette)
plt.title("RFM Main Monetary")
plt.show()
```



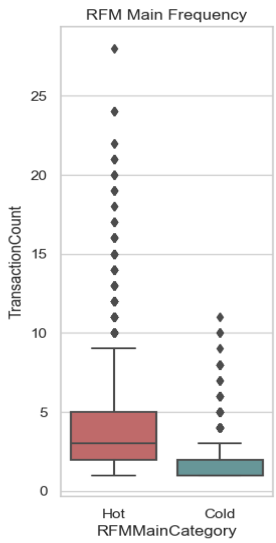
Insights And Recommendations:

From the distribution, it can be observed that customers in the 'Cold' category tend to have lower average spending, typically ranging from $0 to $100. It is notable that higher average spending correlates with increased customer retention, particularly among those who have spent around $500.

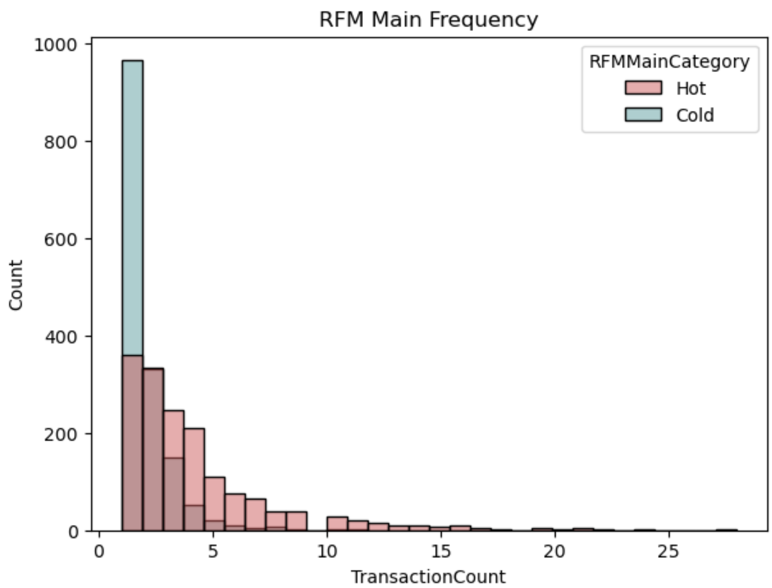
Greater emphasis can be attributed to re-engaging customers exhibiting consistently low average spending.

RFM Main Frequency:

```
# main monetary category
#box plot
plt.figure(figsize=(6, 7))
ax1 = plt.subplot(1, 2, 1)
sns.boxplot(data=rfm_test[rfm_test['TransactionCount'] < 30], y = "TransactionCount", x = "RFMMainCategory",
palette = custom_palette)
ax1.set_title("RFM Main Frequency")
plt.show()
```




```
# Histogram plot
plt.figure(figsize = (15, 5))
ax2 = plt.subplot(1, 2, 2)
sns.histplot(data = rfm_test[rfm_test['TransactionCount'] < 30], x = "TransactionCount", hue =
"RFMMainCategory", bins = 30, palette = custom_palette)
ax2.set_title("RFM Main Frequency")
plt.show()
```



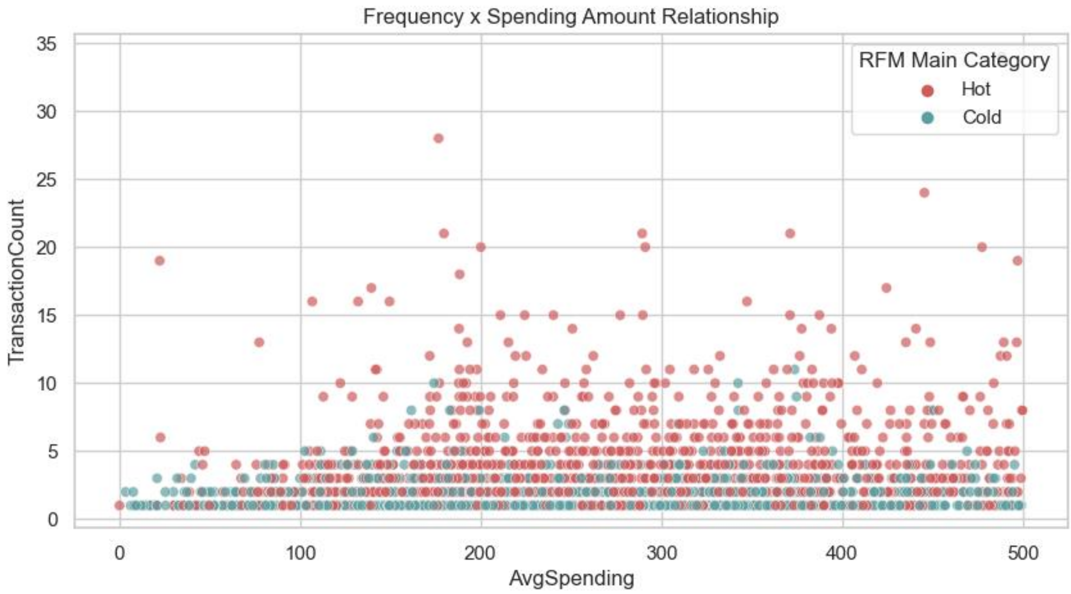
Insights And Recommendations:

From the analysis, we can observe that 'cold' customers consist of those who have made only one transaction. This pattern potentially signals that these customers are likely to churn or remain inactive after just one purchase.

Consequently, it may be imperative to prompt customers to make a second purchase in order to enhance the likelihood of retention.

RFM Relationship:

```
sns.set(rc={'figure.figsize':(10,5)})
sns.set_style('whitegrid')
ax = sns.scatterplot(data=rfm_test[(rfm_test['AvgSpending'] < 500) & (rfm_test['TransactionCount'] < 40)],
                y="TransactionCount", x="AvgSpending", hue="RFMMainCategory", alpha=0.7,
palette=custom_palette)
ax.legend(loc='upper right', title='RFM Main Category')  # Set the legend position and title
ax.set_title("Frequency x Spending Amount Relationship")
```



Insights And Recommendations:

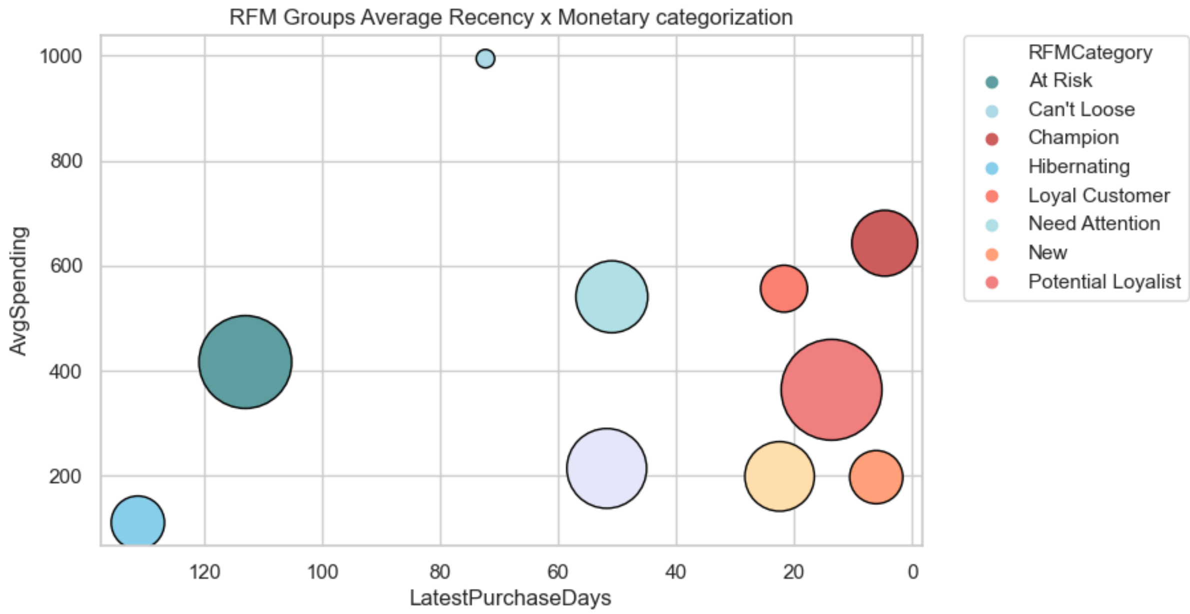
An observation reveals a minimal correlation between transaction frequency and the average spending per transaction. Notably, particularly for users who engage in five or more transactions, they consistently demonstrate continued activity and are categorized as 'hot' users.

This may be established as a target or benchmark to encourage users to conduct five transactions.

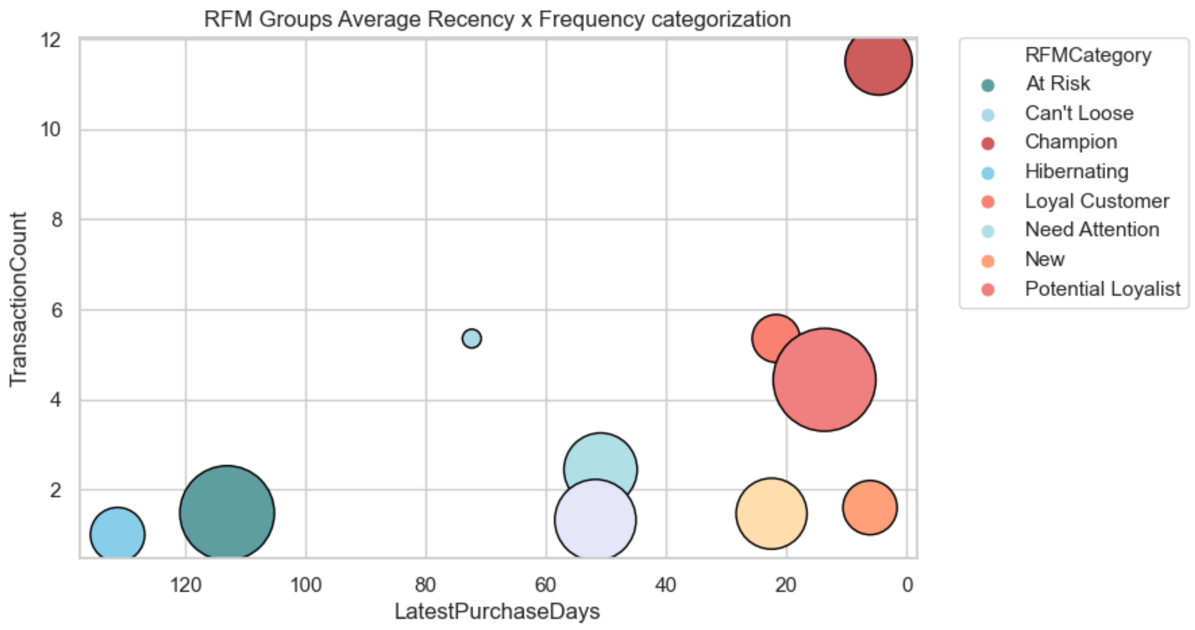
RFM Grouping Categories:

The RFM categories are illustrated using bubble charts. The RFM categories are assigned based on the average calculations of each RFM group. This provides a comprehensive overview of the distinctiveness among the RFM groups.

```
# bubble plotting
sns.set(rc={'figure.figsize': (8, 5)})
sns.set_style('whitegrid')
rfm_mean = rfm_test.groupby('RFMCategory')[['LatestPurchaseDays', 'AvgSpending']].mean().reset_index()
rfm_stats_count = rfm_test['RFMCategory'].value_counts().reset_index()
rfm_stats_count.columns = ['RFMCategory', 'CustomerID']
rfm_stats = rfm_mean.merge(rfm_stats_count, on='RFMCategory')
g = sns.scatterplot(data=rfm_stats, x='LatestPurchaseDays', y='AvgSpending', size='CustomerID',
            hue='RFMCategory', alpha=1, sizes=(100, 3000),
            edgecolor='black', linewidth=1, palette=cus_pal)
h, l = g.get_legend_handles_labels()
plt.legend(h[0:9], l[0:9], bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0., fontsize=11)
g.invert_xaxis()
g.set_title('RFM Groups Average Recency x Monetary categorization')
plt.show(g)
```




```
sns.set(rc={'figure.figsize':(8,5)})
sns.set_style('whitegrid')
rfm_stats_mean = rfm_test.groupby('RFMCategory')[['LatestPurchaseDays',
'TransactionCount']].mean().reset_index()
rfm_stats_count = rfm_test['RFMCategory'].value_counts().reset_index()
rfm_stats_count.columns = ['RFMCategory', 'CustomerID']
rfm_stats = rfm_stats_mean.merge(rfm_stats_count, on='RFMCategory')
g = sns.scatterplot(data=rfm_stats, x="LatestPurchaseDays", y="TransactionCount", size="CustomerID",
            hue="RFMCategory", alpha=1, sizes=(100, 3000), edgecolor='black', linewidth=1, palette=cus_pal)
h, l = g.get_legend_handles_labels()
plt.legend(h[0:9], l[0:9], bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0., fontsize=11)
g.invert_xaxis()
g.set_title('RFM Groups Average Recency x Frequency categorization')
plt.show(g)
```



Insights And Recommendation:

It's evident that 'Champion' customers have the lowest Latest Purchase Days and relatively high Average Spending, indicating their consistent loyalty and significant value. Conversely, the 'Hibernating' and 'At Risk' segments exhibit high Latest Purchase Days and comparably lower Average Spending, suggesting the need for re-engagement strategies.