RFM Customer Segmentation

by Louis Bailey

Summary

This project uses transaction data to understand customer behavior and classify customers into segments. These segments will help in creating a roadmap for the marketing strategy.

In [1]:
```
import pandas as pd
import numpy as np

%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sns
plt.rcParams['figure.figsize'] = [12, 6]
```

Data

In [2]:
```
df = pd.read_excel(r'C:\Users\baile\Downloads\Online Retail.xlsx')
```

In [3]:
```
df.head(3)
```

Out[3]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 08:26:00	2.55	17850.0	United Kingdom
1	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	2010-12-01 08:26:00	2.75	17850.0	United Kingdom

In [4]:
```
df.shape
```

Out[4]:
```
(541909, 8)
```

Clean & Prepare

In [5]:
```
df.dropna(inplace=True)
df.drop_duplicates(inplace=True)
```

In [6]:
```
df = df[df.Quantity>0]
df = df[df.UnitPrice>0]
```

In [7]:
```
df['Total'] = df.Quantity*df.UnitPrice
df['Date'] = df.InvoiceDate.dt.date

latest_purchase = df.groupby('CustomerID')['Date'].max()
df['LatestPurchaseDate'] = df['CustomerID'].map(latest_purchase)
```

Basic Info on Customers

In [8]:
```
print(f'Number of Unique Customers: {len(df.CustomerID.unique())}')
```

```
Number of Unique Customers: 4338
```

In [9]:
```
#
customers_by_country = df.groupby("Country")["CustomerID"].nunique()
customers_by_country = customers_by_country.reset_index()
customers_by_country.columns = ["Country", "UniqueCustomers"]
customers_by_country = customers_by_country[customers_by_country.UniqueCustomers>10].sort_values(by='UniqueCustomers
                                                                                                   ascending=False)

customers_by_country
```

Out[9]:

	Country	UniqueCustomers
35	United Kingdom	3920
14	Germany	94
13	France	87
30	Spain	30
3	Belgium	25
32	Switzerland	21
26	Portugal	19
18	Italy	14
12	Finland	12
1	Austria	11

- The vast majority of customers are in the UK

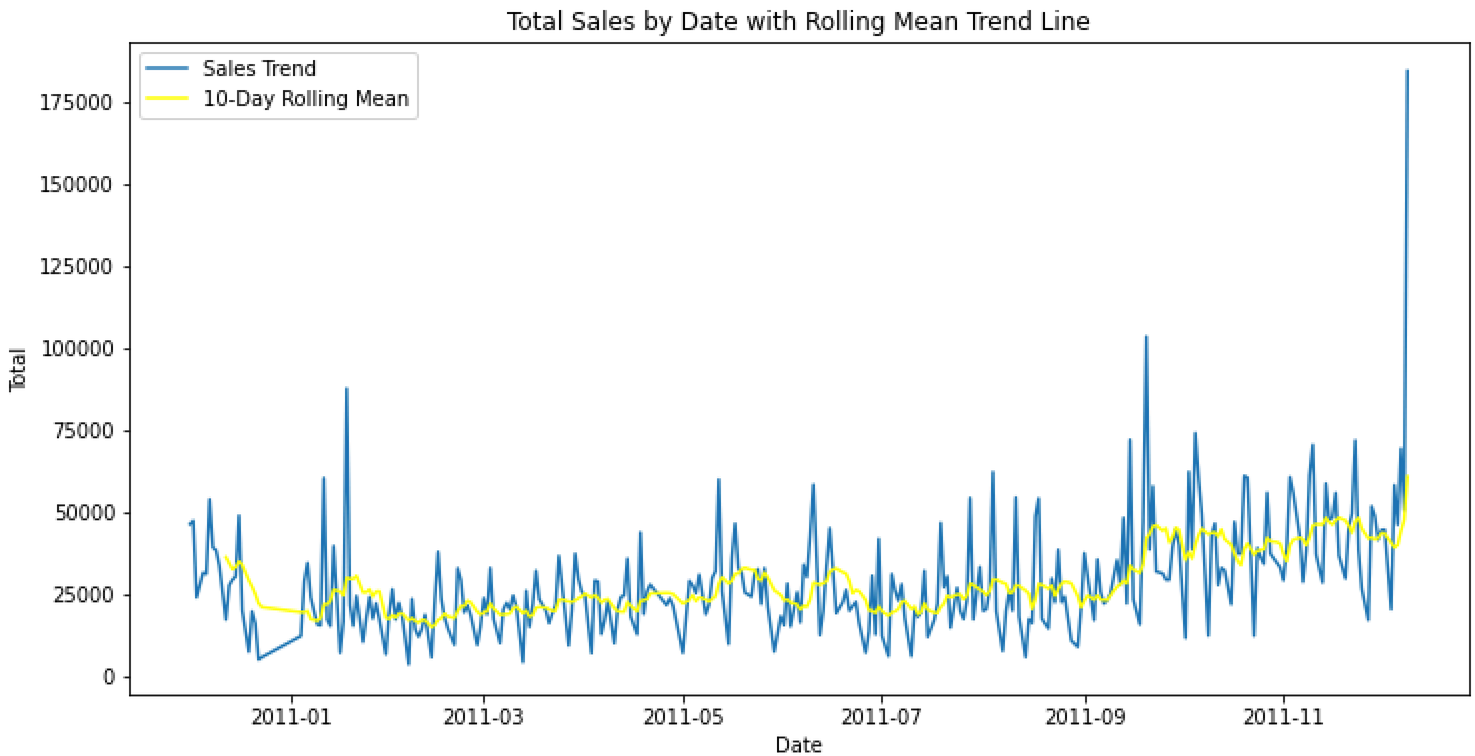
EDA

product sales over time

In [10]:
```
grouped = df.groupby('Date')['Total'].sum().reset_index()
total_by_date = grouped.sort_values(by='Date', ascending=True)
```

In [11]:
```
grouped['RollingMean'] = grouped['Total'].rolling(window=10, center=False).mean()

sns.lineplot(data=grouped, x='Date', y='Total', label="Sales Trend")
sns.lineplot(data=grouped, x='Date', y='RollingMean', color='yellow',label="10-Day Rolling Mean")
plt.title('Total Sales by Date with Rolling Mean Trend Line')
plt.legend();
```



- There are three significant spikes in the sales trend, with a particularly prominent one occurring at the end.

---

best selling products by total revenue

In [12]:
```
product_stats = df.groupby('Description').agg(
    TotalQuantity=('Quantity', 'sum'),
    UniquePurchasers=('CustomerID', 'nunique'),
    TotalRevenue=('Total', 'sum')
).reset_index()

top_revenue = product_stats.sort_values(by='TotalRevenue', ascending=False).head(5)
```

In [13]:
```
top_revenue
```

Out[13]:

	Description	TotalQuantity	UniquePurchasers	TotalRevenue
2319	PAPER CRAFT , LITTLE BIRDIE	80995	1	168469.60
2767	REGENCY CAKESTAND 3 TIER	12374	881	142264.75
3698	WHITE HANGING HEART T-LIGHT HOLDER	36706	856	100392.10
1762	JUMBO BAG RED RETROSPOT	46078	635	85040.54
1992	MEDIUM CERAMIC TOP STORAGE JAR	77916	138	81416.73

best selling products by total quantity

In [14]:
```
product_stats = df.groupby('Description').agg(
    TotalQuantity=('Quantity', 'sum'),
    UniquePurchasers=('CustomerID', 'nunique'),
    TotalRevenue=('Total', 'sum')
).reset_index()

top_quantity = product_stats.sort_values(by='TotalQuantity', ascending=False).head(5)
```

In [15]:
```
top_quantity
```

Out[15]:

	Description	TotalQuantity	UniquePurchasers	TotalRevenue
2319	PAPER CRAFT , LITTLE BIRDIE	80995	1	168469.60
1992	MEDIUM CERAMIC TOP STORAGE JAR	77916	138	81416.73
3786	WORLD WAR 2 GLIDERS ASSTD DESIGNS	54319	307	13558.41
1762	JUMBO BAG RED RETROSPOT	46078	635	85040.54
3698	WHITE HANGING HEART T-LIGHT HOLDER	36706	856	100392.10

- The huge spike in the sales trend plot was from a single customer buying lots of 'PAPER CRAFT , LITTLE BIRDIE'.

In [16]:
```
df[df.Quantity==80995]
```

Out[16]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	Total	Date	LatestPu
540421	581483	23843	PAPER CRAFT , LITTLE BIRDIE	80995	2011-12-09 09:15:00	2.08	16446.0	United Kingdom	168469.6	2011-12-09	

- And it was a single purchase

---

notable customer

In [17]:
```
df[df.CustomerID==16446.0]
```

Out[17]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	Total	Date	LatestPu
194354	553573	22980	PANTRY SCRUBBING BRUSH	1	2011-05-18 09:52:00	1.65	16446.0	United Kingdom	1.65	2011-05-18	
194355	553573	22982	PANTRY PASTRY BRUSH	1	2011-05-18 09:52:00	1.25	16446.0	United Kingdom	1.25	2011-05-18	
540421	581483	23843	PAPER CRAFT , LITTLE BIRDIE	80995	2011-12-09 09:15:00	2.08	16446.0	United Kingdom	168469.60	2011-12-09	

- The customer who made the large purchase has only made three purchases this year. The other two were in May and were much smaller.

- Despite making only three purchases, the customer has spent a lot and made their purchases recently. They currently seem to be a high-value customer.

- Grouping all of the customers by value could provide useful insights. RFM segmentation will be used to achieve this.

RFM Segmentation

- RFM segmentation is a marketing technique used to segment customers based on their purchasing behavior. It stands for Recency, Frequency, and Monetary. By scoring customers on these three metrics we can identify and prioritize customer segments.

In [18]:
```
#calculating recency
df_recency = df.groupby(by='CustomerID', as_index=False)['Date'].max()
df_recency.columns = ['CustomerID', 'LastPurchaseDate']
recent_date = df_recency['LastPurchaseDate'].max()
df_recency['Recency'] = df_recency['LastPurchaseDate'].apply(lambda x: (recent_date - x).days)

#calculating frequency
frequency_df = df.drop_duplicates().groupby(
    by=['CustomerID'], as_index=False)['Date'].count()
frequency_df.columns = ['CustomerID', 'Frequency']

#calculating monetary value
monetary_df = df.groupby(by='CustomerID', as_index=False)['Total'].sum()
monetary_df.columns = ['CustomerID', 'Monetary']

#merging all 3 columns into one dataframe
rf_df = df_recency.merge(frequency_df, on='CustomerID')
rfm_df = rf_df.merge(monetary_df, on='CustomerID').drop(columns='LastPurchaseDate')

#ranking based on r,f and m
rfm_df['R_rank'] = rfm_df['Recency'].rank(ascending=False)
rfm_df['F_rank'] = rfm_df['Frequency'].rank(ascending=True)
rfm_df['M_rank'] = rfm_df['Monetary'].rank(ascending=True)

# normalizing the rank of the customers
rfm_df['R_rank_norm'] = (rfm_df['R_rank']/rfm_df['R_rank'].max())*100
rfm_df['F_rank_norm'] = (rfm_df['F_rank']/rfm_df['F_rank'].max())*100
rfm_df['M_rank_norm'] = (rfm_df['F_rank']/rfm_df['M_rank'].max())*100


rfm_df.drop(columns=['Recency','Frequency','Monetary','R_rank','F_rank','M_rank'], inplace=True)
```

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

Out[19]:

	CustomerID	R_rank_norm	F_rank_norm	M_rank_norm
0	12346.0	3.760704	0.829876	0.829876
1	12347.0	96.169868	88.231904	88.231904
2	12348.0	38.093034	42.346704	42.346704
3	12349.0	74.276788	67.093130	67.093130
4	12350.0	5.264985	24.953896	24.953896

In [20]:

```
rfm_df.shape
```

Out[20]:

```
(4338, 4)
```

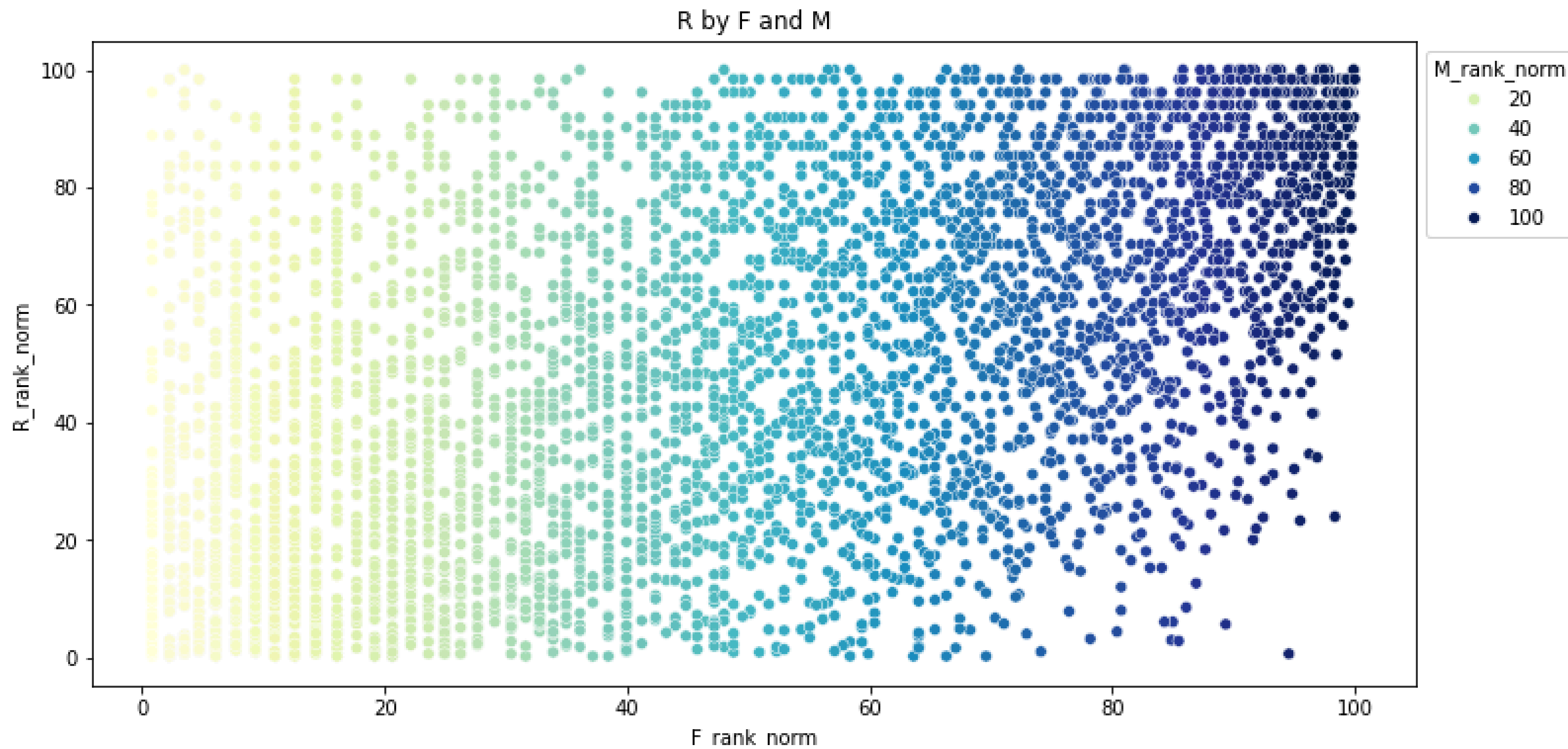
- The normalized rank scores are from 0-100.

Visualizing the RFM Scores

scatter plot

In [21]:

```
sns.scatterplot(data=rfm_df,x='F_rank_norm',y='R_rank_norm', palette="YlGnBu",hue='M_rank_norm')
plt.title('R by F and M')
plt.legend(title="M_rank_norm",loc='best', bbox_to_anchor=(1, 1));
```



- Those having higher frequency and recency are generally associated with higher monetary values. This pattern suggests that distinct customer segments exist within the dataset.

- Initially I thought k-means clustering would be the best way to go here, but actually it seems like segmenting with quantiles will allow for easy interpretation and analysis of customer behavior within discrete categories. Using quantiles to create Low, Medium, and High categories for Recency, Frequency, and Monetary should simplify the segmentation process and provid actionable insights.

---

segmenting using quantiles

In [22]:
```python
rfm_df['R_category'] = pd.qcut(rfm_df['R_rank_norm'], q=3, labels=['Low', 'Medium', 'High'])
rfm_df['F_category'] = pd.qcut(rfm_df['F_rank_norm'], q=3, labels=['Low', 'Medium', 'High'])
rfm_df['M_category'] = pd.qcut(rfm_df['M_rank_norm'], q=3, labels=['Low', 'Medium', 'High'])
```

In [23]:
```python
rfm_df.sample(5)
```

Out[23]:

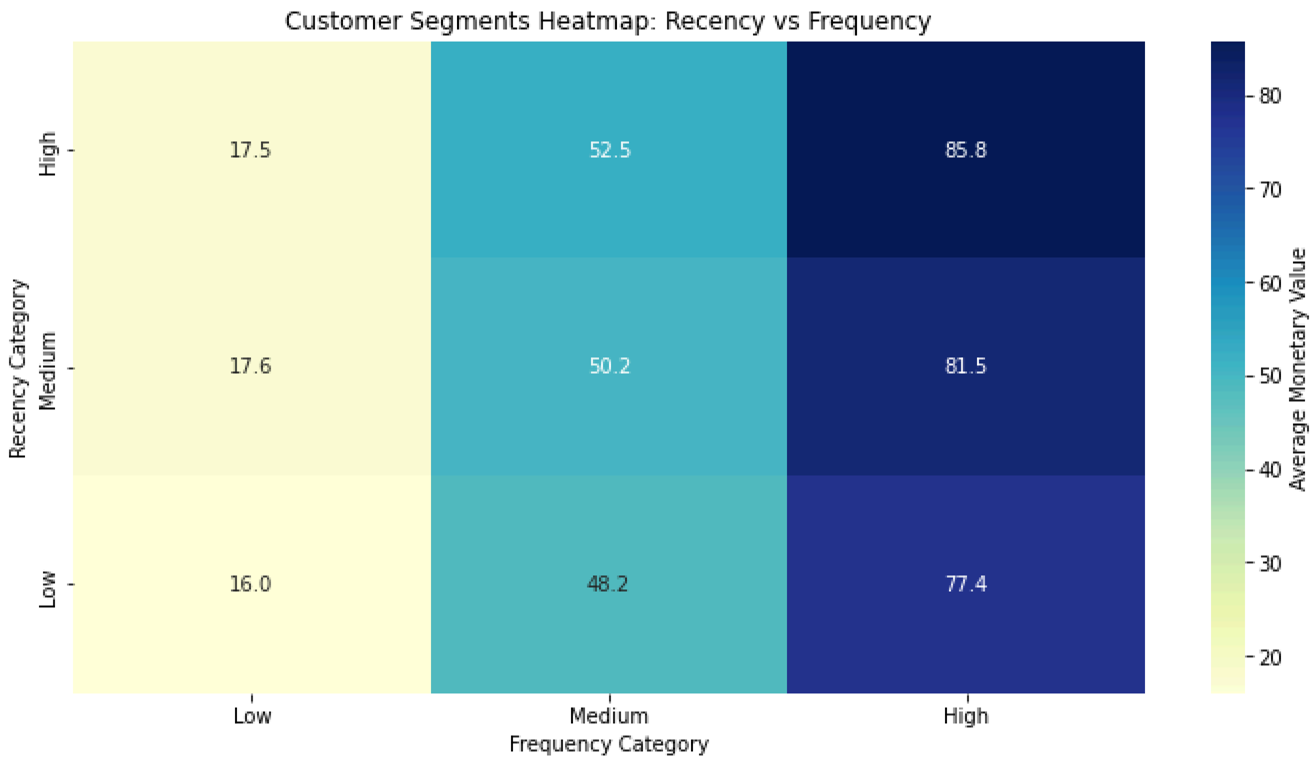
	CustomerID	R_rank_norm	F_rank_norm	M_rank_norm	R_category	F_category	M_category
3565	17223.0	5.264985	55.048409	55.048409	Low	Medium	Medium
2084	15192.0	28.373062	56.558322	56.558322	Low	Medium	Medium
2439	15664.0	50.520713	24.953896	24.953896	Medium	Low	Low
3243	16768.0	51.562138	58.379438	58.379438	Medium	Medium	Medium
1474	14354.0	90.222171	12.597971	12.597971	High	Low	Low

heatmap

In [24]:
```python
heatmap_data = rfm_df.pivot_table(
    index='R_category',
    columns='F_category',
    values='M_rank_norm',
    aggfunc='mean',
    observed=False
)

heatmap_data = heatmap_data[::-1]

sns.heatmap(heatmap_data, annot=True, fmt=".1f", cmap="YlGnBu", cbar_kws={'label': 'Average Monetary Value'})
plt.title("Customer Segments Heatmap: Recency vs Frequency")
plt.xlabel("Frequency Category")
plt.ylabel("Recency Category")
plt.show()
```



Creating the Groups

In [25]:
```
rfm_df = rfm_df[['CustomerID','R_category','F_category', 'M_category']]
rfm_df.sample(5)
```

Out[25]:

	CustomerID	R_category	F_category	M_category
2528	15786.0	Medium	High	High
2918	16327.0	High	High	High
877	13517.0	Medium	High	High
2387	15597.0	Medium	Low	Low
319	12736.0	Low	Low	Low

mappings

In [26]:
```
def classify_customer(row):
    r = row['R_category']
    f = row['F_category']
    m = row['M_category']

    if m == 'High' and not (r == 'Low' and f == 'Low'):
        return 'High Value'
    elif f == 'High' and m != 'High':
        return 'Loyal'
    elif r == 'Low' and f == 'Low':
        return 'At Risk or Churned'
    else:
        return 'Growth Opportunities'

rfm_df['Customer_Group'] = rfm_df.apply(classify_customer, axis=1)
```

In [27]:
```
rfm_df_groups = rfm_df[['CustomerID','Customer_Group']]
rfm_df_groups.head()
```

Out[27]:

	CustomerID	Customer_Group
0	12346.0	At Risk or Churned
1	12347.0	High Value
2	12348.0	Growth Opportunities
3	12349.0	High Value
4	12350.0	At Risk or Churned

The logic for the mappings are:

- High value: high M (and R and F are not both low)
- Loyal: high F (but not high M)
- At risk or churned: R and F are both low
- Growth opportunities: (everyone else)

Conclusion

Summary of Overall Strategy:

High Value: Focus on keeping these top customers happy by offering rewards and recommending higher-quality or complementary products they might like.

Loyal: Encourage these frequent shoppers to keep coming back and try to get them to spend more by showing them new or related products.

Growth Opportunities: Help these customers shop more often by sending personalized suggestions and offering deals that motivate them to stay engaged.

At Risk or Churned: Try to bring these inactive customers back with special offers or discounts, and figure out why they stopped buying to prevent it in the future.

```
rfm_df_groups.to_csv('customer_categories.csv')
```