RFM Analysis Using Python

In [1]:
```
import pandas as pd
import plotly.express as px
import plotly.io as pio
import plotly.graph_objects as go
%matplotlib inline
import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)

pio.templates.default = "plotly_white"

data = pd.read_csv(r"C:\Users\maddh\Downloads\rfm_data.csv")
print(data.head())
```

```
   CustomerID PurchaseDate  TransactionAmount ProductInformation  OrderID  \
0        8814   2023-04-11             943.31          Product C   890075
1        2188   2023-04-11             463.70          Product A   176819
2        4608   2023-04-11              80.28          Product A   340062
3        2559   2023-04-11             221.29          Product A   239145
4        9482   2023-04-11             739.56          Product A   194545

   Location
0     Tokyo
1    London
2  New York
3    London
4     Paris
```

In [2]:
```
from datetime import datetime


# Ensure 'PurchaseDate' is in datetime format
data['PurchaseDate'] = pd.to_datetime(data['PurchaseDate'])

# Calculate Recency (days since last purchase)
today_date = datetime.now()
data['Recency'] = (today_date - data['PurchaseDate']).dt.days  # Apply .dt.days cor

# Calculate Frequency (Number of orders per Customer)
frequency_data = data.groupby('CustomerID')['OrderID'].count().reset_index()
frequency_data.rename(columns={'OrderID': 'Frequency'}, inplace=True)

# Merge Frequency Data
data = data.merge(frequency_data, on='CustomerID', how='left')

# Calculate Monetary Value (Total spending per Customer)
monetary_data = data.groupby('CustomerID')['TransactionAmount'].sum().reset_index()
monetary_data.rename(columns={'TransactionAmount': 'MonetaryValue'}, inplace=True)

# Merge Monetary Value Data
data = data.merge(monetary_data, on='CustomerID', how='left')
```

In [3]:
```
print(data.head())
```

```
   CustomerID PurchaseDate  TransactionAmount ProductInformation  OrderID  \
0        8814   2023-04-11             943.31          Product C   890075
1        2188   2023-04-11             463.70          Product A   176819
2        4608   2023-04-11              80.28          Product A   340062
3        2559   2023-04-11             221.29          Product A   239145
4        9482   2023-04-11             739.56          Product A   194545

   Location  Recency  Frequency  MonetaryValue
0     Tokyo      666          1         943.31
1    London      666          1         463.70
2  New York      666          1          80.28
3    London      666          1         221.29
4     Paris      666          1         739.56
```

In [4]:
```
# Define scoring criteria for each RFM value
recency_scores = [5, 4, 3, 2, 1]  # Higher score for lower recency (more recent)
frequency_scores = [1, 2, 3, 4, 5]  # Higher score for higher frequency
monetary_scores = [1, 2, 3, 4, 5]  # Higher score for higher monetary value

# Calculate RFM scores
data['RecencyScore'] = pd.cut(data['Recency'], bins=5, labels=recency_scores)
data['FrequencyScore'] = pd.cut(data['Frequency'], bins=5, labels=frequency_scores)
data['MonetaryScore'] = pd.cut(data['MonetaryValue'], bins=5, labels=monetary_score
```

In [5]:
```
# Convert RFM scores to numeric type
data['RecencyScore'] = data['RecencyScore'].astype(int)
data['FrequencyScore'] = data['FrequencyScore'].astype(int)
data['MonetaryScore'] = data['MonetaryScore'].astype(int)
```

In [6]:
```
# Calculate RFM score by combining the individual scores
data['RFM_Score'] = data['RecencyScore'] + data['FrequencyScore'] + data['MonetaryS

# Create RFM segments based on the RFM score
segment_labels = ['Low-Value', 'Mid-Value', 'High-Value']
data['Value Segment'] = pd.qcut(data['RFM_Score'], q=3, labels=segment_labels)
```

In [7]:
```
print(data.head())
```

```
   CustomerID PurchaseDate  TransactionAmount ProductInformation  OrderID  \
0        8814   2023-04-11             943.31          Product C   890075
1        2188   2023-04-11             463.70          Product A   176819
2        4608   2023-04-11              80.28          Product A   340062
3        2559   2023-04-11             221.29          Product A   239145
4        9482   2023-04-11             739.56          Product A   194545

   Location  Recency  Frequency  MonetaryValue  RecencyScore  FrequencyScore  \
0     Tokyo      666          1         943.31             1               1
1    London      666          1         463.70             1               1
2  New York      666          1          80.28             1               1
3    London      666          1         221.29             1               1
4     Paris      666          1         739.56             1               1

   MonetaryScore  RFM_Score Value Segment
0              2          4     Low-Value
1              1          3     Low-Value
2              1          3     Low-Value
3              1          3     Low-Value
4              2          4     Low-Value
```

In [26]:
```python
# RFM Segment Distribution
segment_counts = data['Value Segment'].value_counts().reset_index()
segment_counts.columns = ['Value Segment', 'Count']

pastel_colors = px.colors.qualitative.Pastel

# Create the bar chart
fig_segment_dist = px.bar(segment_counts, x='Value Segment', y='Count',
                          color='Value Segment', color_discrete_sequence=pastel_col
                          title='RFM Value Segment Distribution')

# Update the layout
fig_segment_dist.update_layout(xaxis_title='RFM Value Segment',
                               yaxis_title='Count',
                               showlegend=False,
                                width=800,  # Adjust the width of the figure
                                height=400)

# Show the figure
fig_segment_dist.show()
fig_segment_dist.write_html("segment_distribution.html")
```

In [17]:
```
# Create a new column for RFM Customer Segments
data['RFM Customer Segments'] = ''

# Assign RFM segments based on the RFM score
data.loc[data['RFM_Score'] >= 9, 'RFM Customer Segments'] = 'Champions'
data.loc[(data['RFM_Score'] >= 6) & (data['RFM_Score'] < 9), 'RFM Customer Segments
data.loc[(data['RFM_Score'] >= 5) & (data['RFM_Score'] < 6), 'RFM Customer Segments
data.loc[(data['RFM_Score'] >= 4) & (data['RFM_Score'] < 5), 'RFM Customer Segments
data.loc[(data['RFM_Score'] >= 3) & (data['RFM_Score'] < 4), 'RFM Customer Segments

# Print the updated data with RFM segments
print(data[['CustomerID', 'RFM Customer Segments']].head(10))
```

```
   CustomerID RFM Customer Segments
0        8814            Can't Lose
1        2188                  Lost
2        4608                  Lost
3        2559                  Lost
4        9482            Can't Lose
5        8483                  Lost
6        8317   Potential Loyalists
7        6911                  Lost
8        8993                  Lost
9        3519                  Lost
```

In [27]:
```
segment_product_counts = data.groupby(['Value Segment', 'RFM Customer Segments']).s

segment_product_counts = segment_product_counts.sort_values('Count', ascending=Fals

fig_treemap_segment_product = px.treemap(segment_product_counts,
```

```
                                        path=['Value Segment', 'RFM Customer Segme
                                        values='Count',
                                        color='Value Segment', color_discrete_sequ
                                        title='RFM Customer Segments by Value')

# Update layout to set figure size
fig_treemap_segment_product.update_layout(
    width=800,  # Adjust the width of the figure
    height=500  # Adjust the height of the figure
)

fig_treemap_segment_product.show()
fig_treemap_segment_product.write_html("treemap_segment_product.html")
```

In [29]:
```
# Filter the data to include only the customers in the Champions segment
champions_segment = data[data['RFM Customer Segments'] == 'Champions']

fig = go.Figure()
fig.add_trace(go.Box(y=champions_segment['RecencyScore'], name='Recency'))
fig.add_trace(go.Box(y=champions_segment['FrequencyScore'], name='Frequency'))
fig.add_trace(go.Box(y=champions_segment['MonetaryScore'], name='Monetary'))

fig.update_layout(title='Distribution of RFM Values within Champions Segment',
                  yaxis_title='RFM Value',
```

```
                  showlegend=True,
                 width=800,  # Adjust the width of the figure
    height=400 )

fig.show()
fig.write_html("champions_segment_rfm_distribution.html")
```

In [30]:
```
correlation_matrix = champions_segment[['RecencyScore', 'FrequencyScore', 'Monetary

# Visualize the correlation matrix using a heatmap
fig_heatmap = go.Figure(data=go.Heatmap(
                   z=correlation_matrix.values,
                   x=correlation_matrix.columns,
                   y=correlation_matrix.columns,
                   colorscale='RdBu',
                   colorbar=dict(title='Correlation')))


# Update layout to set figure size
fig_heatmap.update_layout(
    title='Correlation Matrix of RFM Values within Champions Segment',
    width=800,  # Adjust the width of the figure
    height=500 ) # Adjust the height of the figure

fig_heatmap.show()
fig_heatmap.write_html("champions_segment_rfm_heatmap.html")
```