RFM Analysis

Unleashing Customer Insights with RFM Analysis

Welcome to my RFM analysis project, a powerful tool for understanding customer behavior. We'll explore Recency, Frequency, and Monetary (RFM) analysis to segment customers and drive growth. Whether you're a marketer or a business owner, this project will equip you with data-driven strategies to enhance customer engagement and business success. Let's unlock the potential of RFM analysis together.

In [1]:
```
import pandas as pd
import plotly.express as px
import plotly.io as pio
import plotly.graph_objects as go
pio.templates.default = "plotly_white"

data = pd.read_csv("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
```

Calculating RFM Values

- The code calculates Recency, Frequency, and Monetary Value for RFM analysis.
- Recency is determined by finding the number of days between the current date and the 'PurchaseDate' column.
- Frequency is calculated by counting the number of unique 'OrderID' values per customer.
- Monetary Value is computed by summing the 'TransactionAmount' for each customer.

This data preparation is essential for subsequent RFM analysis and customer segmentation.

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

# Convert 'PurchaseDate' to datetime
data['PurchaseDate'] = pd.to_datetime(data['PurchaseDate'])

# Calculate Recency
data['Recency'] = (datetime.now().date() - data['PurchaseDate'].dt.date).dt.days

# Calculate Frequency
frequency_data = data.groupby('CustomerID')['OrderID'].count().reset_index()
frequency_data.rename(columns={'OrderID': 'Frequency'}, inplace=True)
data = data.merge(frequency_data, on='CustomerID', how='left')

# Calculate Monetary Value
monetary_data = data.groupby('CustomerID')['TransactionAmount'].sum().reset_index()
monetary_data.rename(columns={'TransactionAmount': 'MonetaryValue'}, inplace=True)
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      308          1         943.31
1    London      308          1         463.70
2  New York      308          1          80.28
3    London      308          1         221.29
4     Paris      308          1         739.56
```

Calculating RFM Scores

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_scores)
```

- Scoring criteria for Recency, Frequency, and Monetary Value are established with lists: Higher recency receives lower scores, while higher frequency and monetary value receive higher scores.
- The code then calculates RFM scores:
  - `RecencyScore` is determined by dividing the 'Recency' values into five equal bins and assigning the corresponding recency score based on the established criteria.
  - `FrequencyScore` is calculated in a similar manner, dividing 'Frequency' values into bins and assigning scores.
  - `MonetaryScore` is computed by dividing 'MonetaryValue' into bins and applying scores.

These scores are essential for segmenting and analyzing customers based on RFM values.

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)
```

- The code computes the RFM score for each customer by adding their individual Recency, Frequency, and Monetary scores.
- RFM scores can range from low to high, where higher scores indicate more valuable customers.
- The code then creates RFM segments based on the RFM scores. These segments categorize customers into three groups:
  - 'Low-Value': Customers with lower RFM scores.
  - 'Mid-Value': Customers with intermediate RFM scores.
  - 'High-Value': Customers with higher RFM scores.
- The `pd.qcut` function is used to distribute customers into these segments based on quantiles, ensuring an approximately equal number of customers in each segment.

This process enables the categorization of customers into different value segments, allowing for targeted marketing and analysis based on their behavior and value to the business.

RFM Value Segmentation

In [6]:

```
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      308          1         943.31             1               1
1    London      308          1         463.70             1               1
2  New York      308          1          80.28             1               1
3    London      308          1         221.29             1               1
4     Paris      308          1         739.56             1               1

   MonetaryScore
0              2
1              1
2              1
3              1
4              2
```

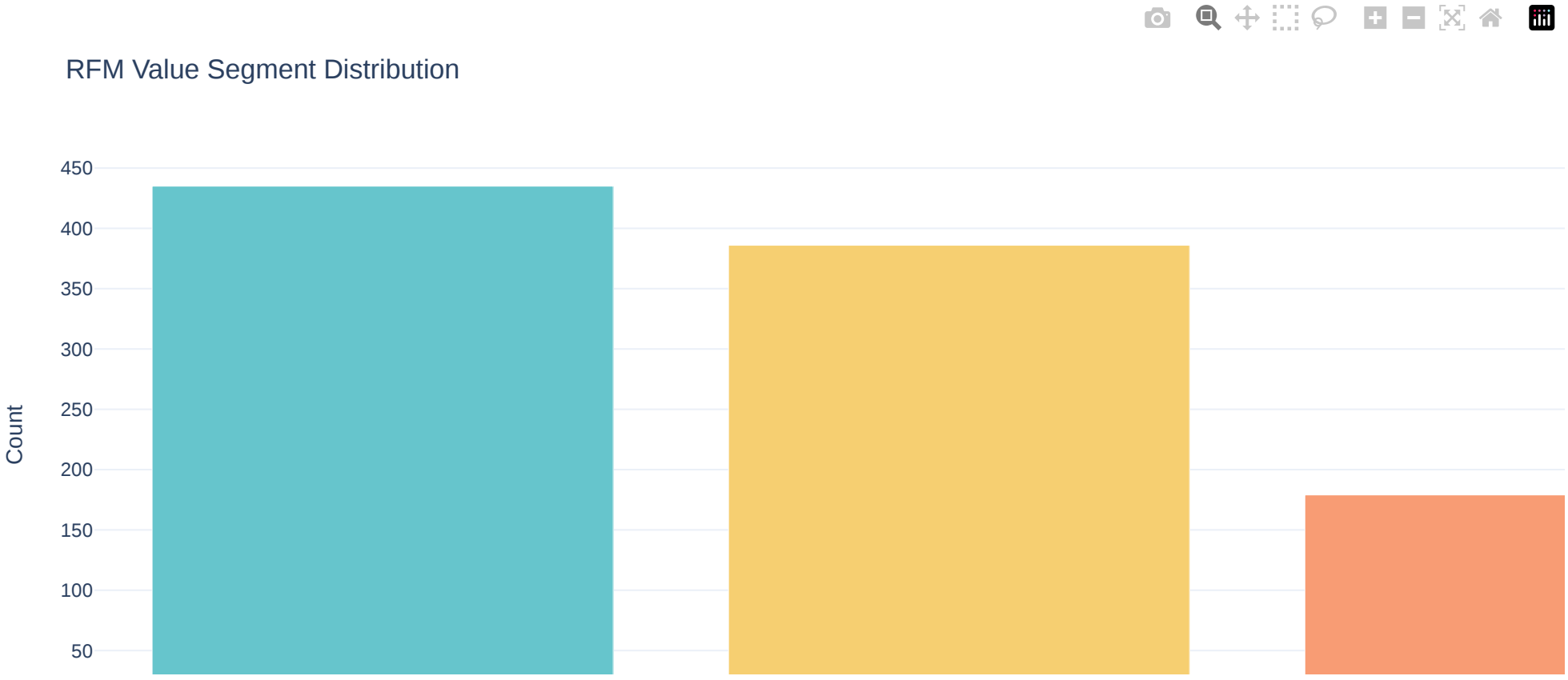
In [7]:

```
# Calculate RFM score by combining the individual scores
data['RFM_Score'] = data['RecencyScore'] + data['FrequencyScore'] + data['MonetaryScore']
# 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 [8]:

```
# 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_colors,
                          title='RFM Value Segment Distribution')

# Update the layout
fig_segment_dist.update_layout(xaxis_title='RFM Value Segment',
                               yaxis_title='Count',
                               showlegend=False)
```

```
# Show the figure
fig_segment_dist.show()
```



- The code begins by counting the number of customers in each RFM value segment ('Low-Value,' 'Mid-Value,' 'High-Value') using `data['Value Segment'].value_counts()`.
- It then organizes this count data into a DataFrame with columns 'Value Segment' and 'Count' for further visualization.
- A color palette, 'pastel_colors,' is defined for visualizing the segments in a pleasing color scheme.
- The code creates a bar chart using Plotly Express (`px.bar`) to visualize the distribution of customers across the RFM value segments.

- The chart's title is set as 'RFM Value Segment Distribution,' and the x and y-axis titles are also defined.
- The legend is disabled with `showlegend=False` as the colors alone represent the segments.
- Finally, the bar chart is displayed with `fig_segment_dist.show()`.

This visualization helps you understand how your customer base is distributed across different RFM value segments, which can inform your marketing and engagement strategies.

RFM Customer Segments

In [9]:
```
# 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'] = 'Potential Loyalists'
data.loc[(data['RFM_Score'] >= 5) & (data['RFM_Score'] < 6), 'RFM Customer Segments'] = 'At Risk Customers'
data.loc[(data['RFM_Score'] >= 4) & (data['RFM_Score'] < 5), 'RFM Customer Segments'] = "Can't Lose"
data.loc[(data['RFM_Score'] >= 3) & (data['RFM_Score'] < 4), 'RFM Customer Segments'] = "Lost"

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

```
     CustomerID RFM Customer Segments
0          8814            Can't Lose
1          2188                  Lost
2          4608                  Lost
3          2559                  Lost
4          9482            Can't Lose
..          ...                   ...
995        2970   Potential Loyalists
996        6669   Potential Loyalists
997        8836   Potential Loyalists
998        1440   Potential Loyalists
999        4759   Potential Loyalists

[1000 rows x 2 columns]
```

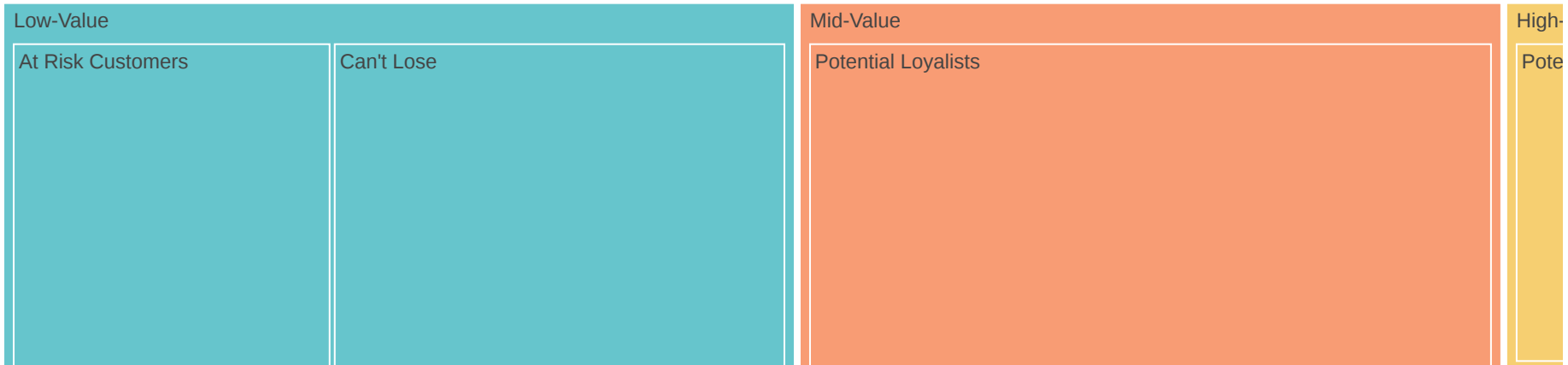
- A new column called 'RFM Customer Segments' is created to store the RFM segments for each customer.
- The code assigns RFM segments based on the RFM score using conditional statements. Here are the segment assignments:
  - 'Champions': Customers with an RFM score of 9 or higher.

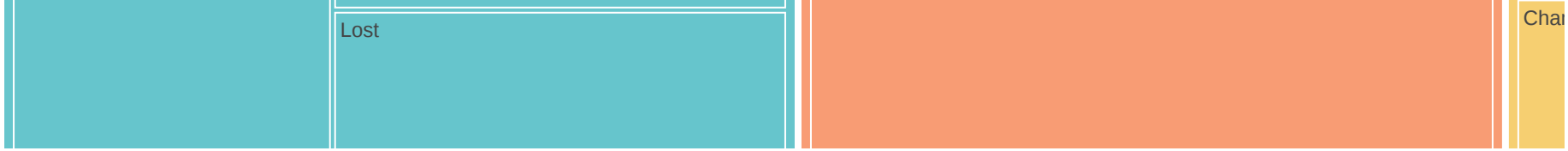
    - 'Potential Loyalists': Customers with an RFM score between 6 and 8.
    - 'At Risk Customers': Customers with an RFM score between 5 and 5.99.
    - "Can't Lose": Customers with an RFM score between 4 and 4.99.
    - "Lost": Customers with an RFM score between 3 and 3.99.
- The code then prints the updated data, showing the 'CustomerID' and their corresponding 'RFM Customer Segments' to provide insights into the customer segmentation.

This assignment of RFM Customer Segments helps categorize customers based on their recency, frequency, and monetary value scores, enabling more targeted marketing and retention strategies.

RFM Analysis

In [10]:
```
segment_product_counts = data.groupby(['Value Segment', 'RFM Customer Segments']).size().reset_index(name='Count')

segment_product_counts = segment_product_counts.sort_values('Count', ascending=False)

fig_treemap_segment_product = px.treemap(segment_product_counts,
                                         path=['Value Segment', 'RFM Customer Segments'],
                                         values='Count',
                                         color='Value Segment', color_discrete_sequence=px.colors.qualitative.Pastel,
                                         title='RFM Customer Segments by Value')
fig_treemap_segment_product.show()
```





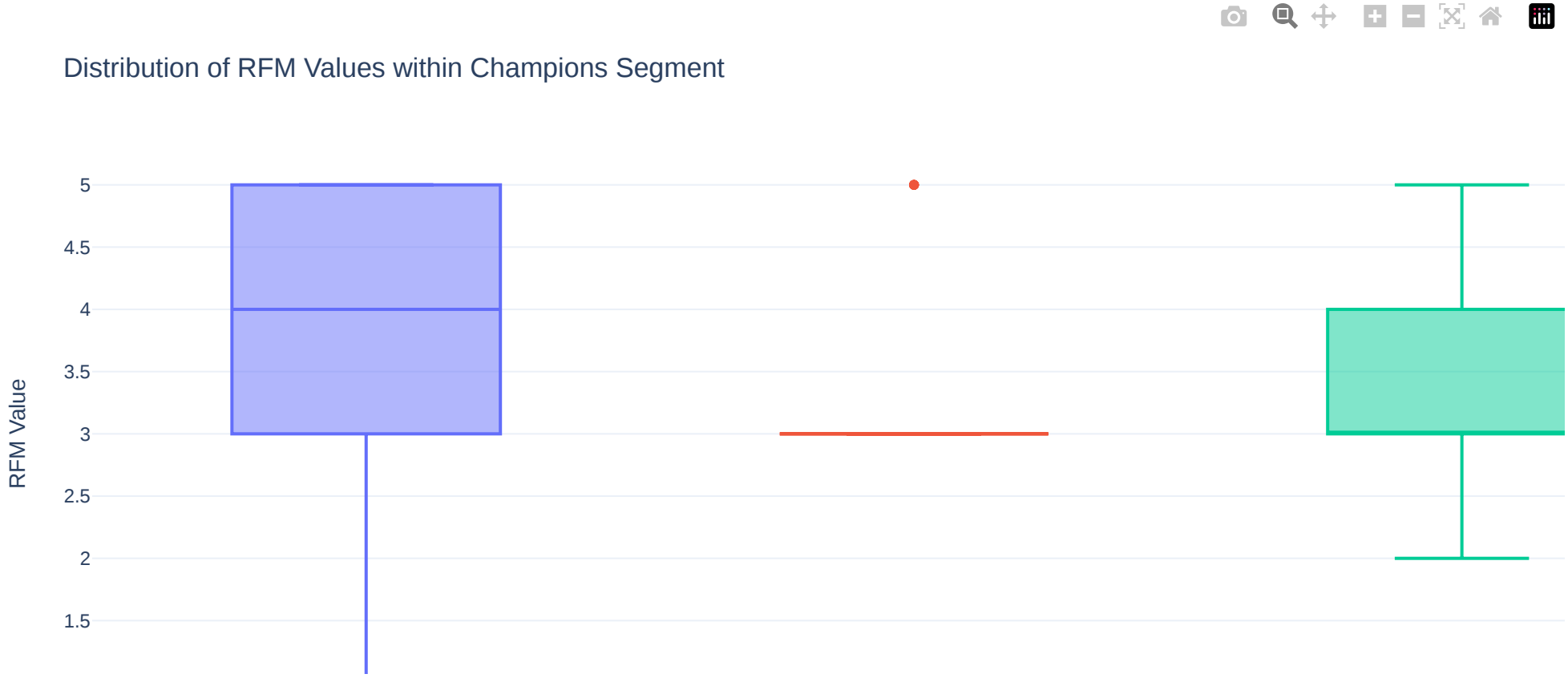
- The code groups the data by two columns, 'Value Segment' and 'RFM Customer Segments,' and calculates the count of customers in each combination using `.groupby(['Value Segment', 'RFM Customer Segments']).size()`.
- It organizes this count data into a DataFrame with columns 'Value Segment,' 'RFM Customer Segments,' and 'Count.'
- The DataFrame is sorted in descending order based on the 'Count' column to show the most significant segments first.
- A Treemap chart is created using Plotly Express (`px.treemap`). This chart visualizes the hierarchy of segments where 'Value Segment' is the top-level category, 'RFM Customer Segments' is the subcategory, and 'Count' is represented by the size of each block.
- The color of each block in the Treemap is based on the 'Value Segment' to distinguish the top-level segments.
- The chart is titled 'RFM Customer Segments by Value.'
- Finally, the Treemap chart is displayed with `fig_treemap_segment_product.show()`.

This visualization provides a clear representation of how different RFM Customer Segments are distributed across Value Segments, offering insights into customer behavior and potential marketing opportunities.

In [11]:

```
# 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)
```

```
fig.show()
```



- The code filters the data to include only customers in the 'Champions' segment by using `data[data['RFM Customer Segments'] == 'Champions']`. This new DataFrame, 'champions_segment,' contains data specifically for these high-value customers.
- A box plot is created using Plotly Express (`go.Figure()`) to display the distribution of RFM values within the 'Champions' segment. Three box plots are added to the figure, each representing the distribution of Recency, Frequency, and Monetary scores.
- The title of the box plot is set as 'Distribution of RFM Values within Champions Segment.'
- The y-axis is labeled as 'RFM Value,' and the legend is enabled to distinguish the three box plots.

- Finally, the box plot is displayed with `fig.show()`.

This visualization allows you to explore the spread and central tendencies of Recency, Frequency, and Monetary values within the 'Champions' segment, helping to better understand the behavior of these high-value customers.

In [12]:

```
correlation_matrix = champions_segment[['RecencyScore', 'FrequencyScore', 'MonetaryScore']].corr()

# 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')))

fig_heatmap.update_layout(title='Correlation Matrix of RFM Values within Champions Segment')

fig_heatmap.show()
```

- The code calculates the correlation matrix for the 'Champions' segment, specifically focusing on the 'RecencyScore,' 'FrequencyScore,' and 'MonetaryScore' columns. The `corr()` function is used to compute the correlation coefficients between these three variables.
- A heatmap is created to visualize the correlation matrix. In this heatmap:
  - The `z` parameter represents the correlation values from the correlation matrix.
  - The `x` and `y` parameters specify the labels for the x-axis and y-axis, which correspond to the RFM values.
  - The 'colorscale' is set to 'RdBu,' indicating the color spectrum used for representing correlations.
  - The colorbar title is defined as 'Correlation.'
- The title of the heatmap is set as 'Correlation Matrix of RFM Values within Champions Segment.'
- Finally, the heatmap is displayed with `fig_heatmap.show()`.

This visualization helps you understand the relationships between Recency, Frequency, and Monetary values within the 'Champions' segment and assess whether there are any significant correlations between these RFM components.

In [13]:

```
import plotly.colors

pastel_colors = plotly.colors.qualitative.Pastel

segment_counts = data['RFM Customer Segments'].value_counts()

# Create a bar chart to compare segment counts
fig = go.Figure(data=[go.Bar(x=segment_counts.index, y=segment_counts.values,
                             marker=dict(color=pastel_colors))])

# Set the color of the Champions segment as a different color
champions_color = 'rgb(158, 202, 225)'
fig.update_traces(marker_color=[champions_color if segment == 'Champions' else pastel_colors[i]
                                for i, segment in enumerate(segment_counts.index)],
                  marker_line_color='rgb(8, 48, 107)',
                  marker_line_width=1.5, opacity=0.6)

# Update the layout
fig.update_layout(title='Comparison of RFM Segments',
                  xaxis_title='RFM Segments',
                  yaxis_title='Number of Customers',
                  showlegend=False)
```

```
fig.show()
```

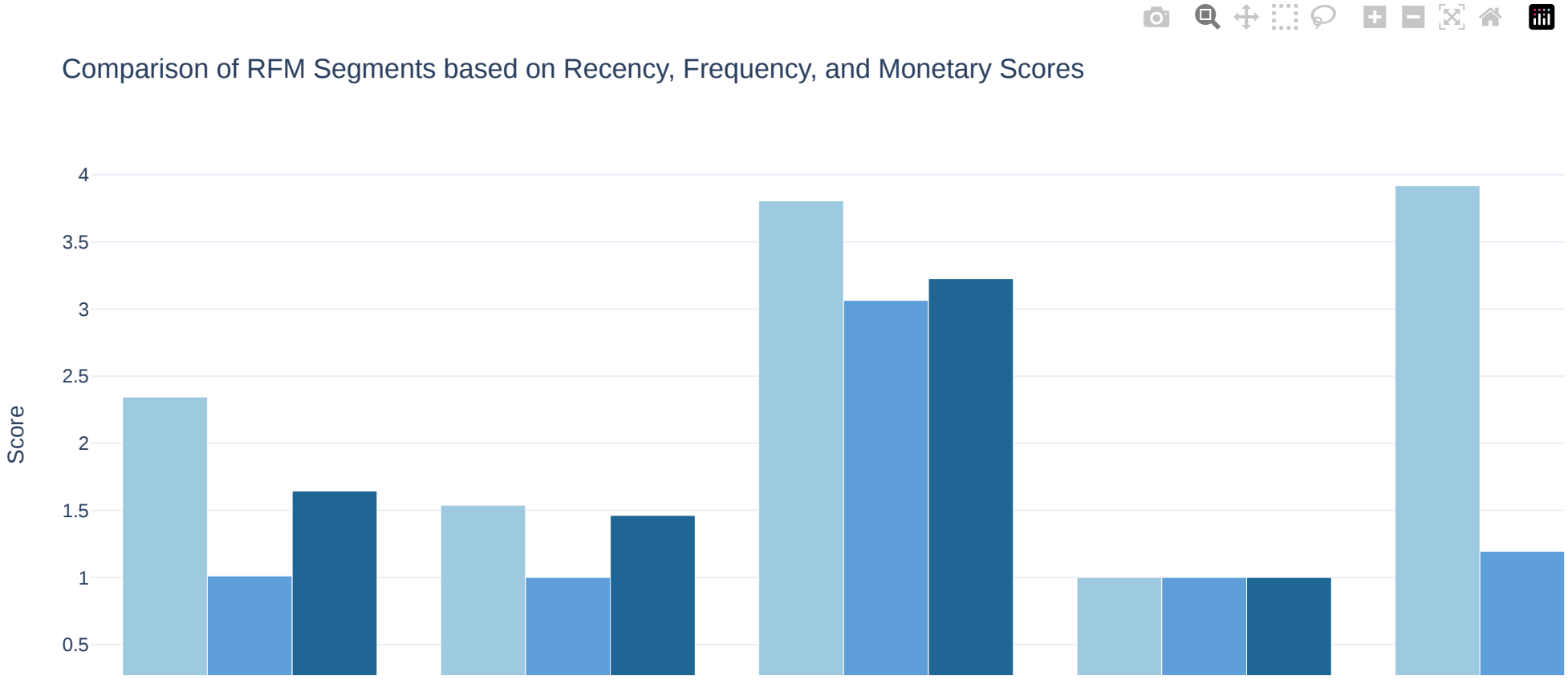
- The code defines a list of pastel colors using Plotly's `plotly.colors.qualitative.Pastel` to set the color scheme for the chart.
- It calculates the number of customers in each RFM segment using `data['RFM Customer Segments'].value_counts()`. This gives you the count of customers in each segment.
- A bar chart is created using Plotly Express (`go.Figure`) to visualize the distribution of customers across the RFM segments. The x-axis represents the RFM segments, and the y-axis represents the number of customers.
- The color of the bars is customized to differentiate the 'Champions' segment, which is set to a different color ('rgb(158, 202, 225)').

- The code updates the layout of the chart with a title, x-axis title, and y-axis title, and it disables the legend to keep the chart clean.
- Finally, the bar chart is displayed with `fig.show()`.

This visualization allows you to compare the distribution of customers across different RFM segments, providing insights into the customer segmentation and distribution within your dataset.

In [14]:

```
# Calculate the average Recency, Frequency, and Monetary scores for each segment
segment_scores = data.groupby('RFM Customer Segments')[['RecencyScore', 'FrequencyScore', 'MonetaryScore']].mean().reset_index()

# Create a grouped bar chart to compare segment scores
fig = go.Figure()

# Add bars for Recency score
fig.add_trace(go.Bar(
    x=segment_scores['RFM Customer Segments'],
    y=segment_scores['RecencyScore'],
    name='Recency Score',
    marker_color='rgb(158,202,225)'
))

# Add bars for Frequency score
fig.add_trace(go.Bar(
    x=segment_scores['RFM Customer Segments'],
    y=segment_scores['FrequencyScore'],
    name='Frequency Score',
    marker_color='rgb(94,158,217)'
))

# Add bars for Monetary score
fig.add_trace(go.Bar(
    x=segment_scores['RFM Customer Segments'],
    y=segment_scores['MonetaryScore'],
    name='Monetary Score',
    marker_color='rgb(32,102,148)'
))

# Update the layout
fig.update_layout(
    title='Comparison of RFM Segments based on Recency, Frequency, and Monetary Scores',
    xaxis_title='RFM Segments',
    yaxis_title='Score',
    barmode='group',
    showlegend=True
)

fig.show()
```



- The code groups the data by 'RFM Customer Segments' and calculates the mean scores for 'RecencyScore,' 'FrequencyScore,' and 'MonetaryScore' within each segment. This is done using `data.groupby('RFM Customer Segments')[['RecencyScore', 'FrequencyScore', 'MonetaryScore']].mean().reset_index()`.
- A grouped bar chart is created using Plotly Express (`go.Figure()`) to visually compare the average scores for each segment.
- Three sets of bars are added to the chart, each representing the Recency, Frequency, and Monetary scores. These bars are named 'Recency Score,' 'Frequency Score,' and 'Monetary Score,' respectively.
- Each set of bars is color-coded for visual distinction, with 'Recency Score' in 'rgb(158,202,225),' 'Frequency Score' in 'rgb(94,158,217),' and 'Monetary Score' in 'rgb(32,102,148)'.

- The chart's layout is updated to include a title, x-axis title ('RFM Segments'), and y-axis title ('Score'). The 'barmode' is set to 'group' to display bars for each score category side by side, and the legend is enabled to differentiate the score categories.

- Finally, the grouped bar chart is displayed with `fig.show()`.

This visualization allows you to compare the average Recency, Frequency, and Monetary scores across different RFM segments, providing insights into the behavior and value of customers in each segment.