```
import pandas as pd
from datetime import datetime as dt, timedelta
import plotly.express as px
import plotly.graph_objects as go
import plotly.colors
```

```
df = pd.read_csv('online_retail.csv')
```

```
df.head()
```

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	C
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 08:26:00	2.55	17850.0	K
1	536365	71053	WHITE METAL	6	2010-12-01 08:26:00	3.39	17850.0	K

```
df.tail()
```

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	Custome
541904	581587	22613	PACK OF 20 SPACEBOY NAPKINS	12	2011-12-09 12:50:00	0.85	1268
541905	581587	22899	CHILDREN'S APRON DOLLY GIRL	6	2011-12-09 12:50:00	2.10	1268

```
df.dropna(subset = ['CustomerID'], inplace = True)
```

```
df['InvoiceDate'] = pd.to_datetime(df['InvoiceDate'])
df['Total Amount'] = df['Quantity'] * df['UnitPrice']
```

```
df.head()
```

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

```
reference_date = pd.Timestamp(dt.now().date())
```

```
reference_date
```

```
Timestamp('2024-08-26 00:00:00')
```

```
reference_date = df['InvoiceDate'].max() + timedelta(days=1)
```

```
reference_date
```

```
Timestamp('2011-12-10 12:50:00')
```

```
rfm = df.groupby('CustomerID').agg({
    'InvoiceDate': lambda x: (reference_date - x.max()).days,
    'InvoiceNo': 'count',
    'Total Amount': 'sum'
})
```

```
rfm.rename(columns = {
    'InvoiceDate': 'Recency',
    'InvoiceNo': 'Frequency',
    'Total Amount': 'Monetary'
}, inplace = True)
```

```
rfm.head()
```

CustomerID	Recency	Frequency	Monetary
12346.0	326	2	0.00
12347.0	2	182	4310.00
12348.0	75	31	1797.24
12349.0	19	73	1757.55
12350.0	310	17	334.40

Next steps: Generate code with rfm | View recommended plots | New interactive sheet

```
# Define Quantiles

quantiles = rfm.quantile(q=[0.25, 0.5, 0.75])

# Assign RFM Scores
def RScore(x,p,d):
  if p == 'Recency':
    if x <= d[p][0.25]:
      return 4
    elif x <= d[p][0.5]:
      return 3
    elif x <= d[p][0.75]:
      return 2
    else:
      return 1
  else:
    if x <= d[p][0.25]:
      return 1
    elif x <= d[p][0.5]:
      return 2
    elif x <= d[p][0.75]:
      return 3
    else:
      return 4


rfm['R'] = rfm['Recency'].apply(RScore, args=('Recency', quantiles))
rfm['F'] = rfm['Frequency'].apply(RScore, args=('Frequency', quantiles))
rfm['M'] = rfm['Monetary'].apply(RScore, args=('Monetary', quantiles))


rfm.head()
```

CustomerID	Recency	Frequency	Monetary	R	F	M
12346.0	326	2	0.00	1	1	1
12347.0	2	182	4310.00	4	4	4
12348.0	75	31	1797.24	2	2	4
12349.0	19	73	1757.55	3	3	4
12350.0	310	17	334.40	1	1	2

Next steps: Generate code with rfm | View recommended plots | New interactive sheet

```
rfm['RFM_Segment'] = rfm['R'].astype(str) + rfm['F'].astype(str) + rfm['M'].astype(str)
rfm['RFM_Score']= rfm[['R','F','M']].sum(axis=1)
```

```
rfm.head()
```

CustomerID	Recency	Frequency	Monetary	R	F	M	RFM_Segment	RFM_Score
12346.0	326	2	0.00	1	1	1	111	3
12347.0	2	182	4310.00	4	4	4	444	12
12348.0	75	31	1797.24	2	2	4	224	8
12349.0	19	73	1757.55	3	3	4	334	10
12350.0	310	17	334.40	1	1	2	112	4

Next steps: Generate code with rfm | View recommended plots | New interactive sheet

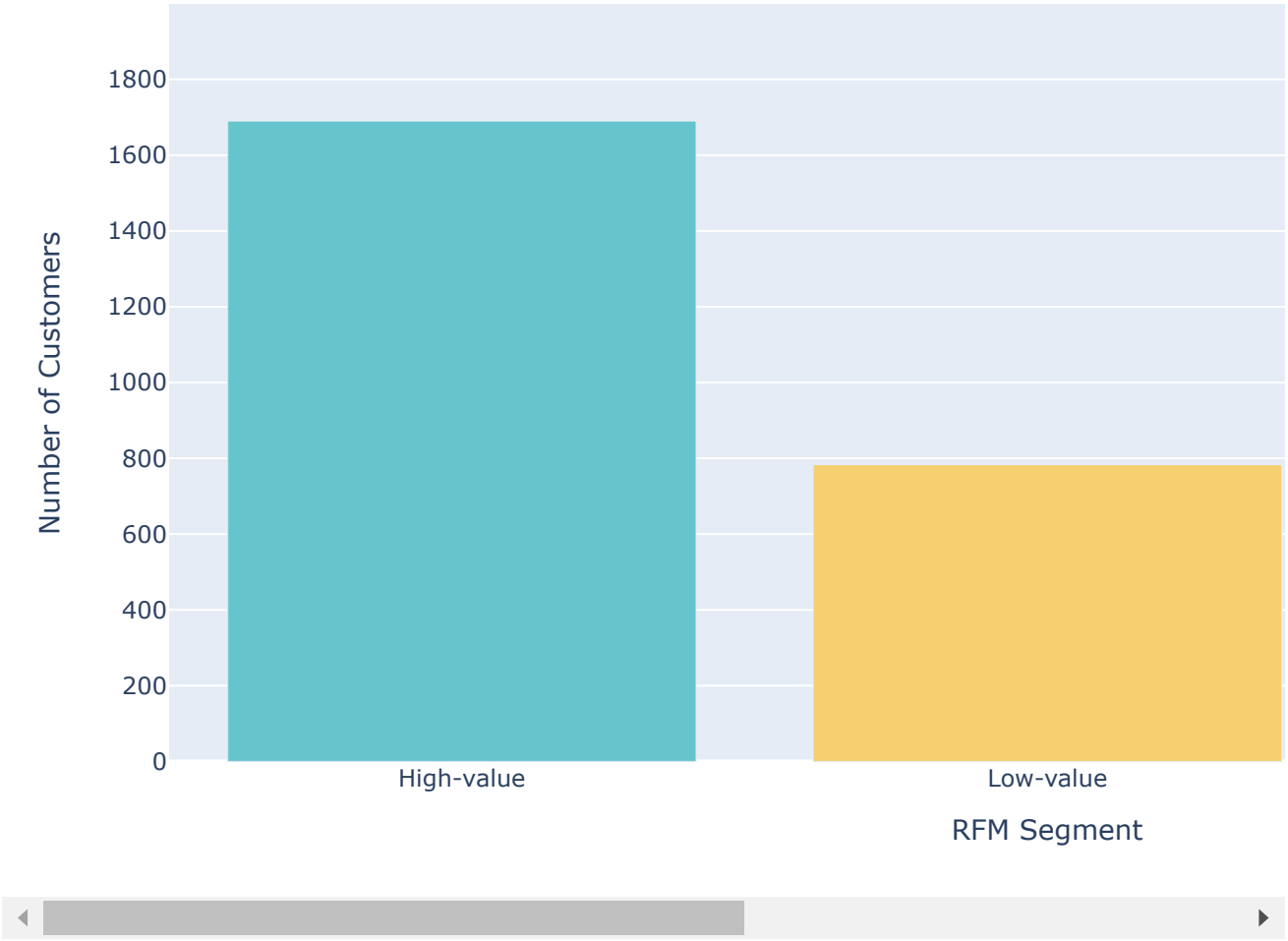
```
segment_labels = ['Low-value', 'Mid-value', 'High-value']

def assign_segment(score):
  if score < 5:
    return 'Low-value'
  elif score < 9:
    return 'Mid-value'
  else:
    return 'High-value'

rfm['RFM_Segment_Label'] = rfm['RFM_Score'].apply(assign_segment)
```

```
rfm.head()
```

CustomerID	Recency	Frequency	Monetary	R	F	M	RFM_Segment	RFM_Score	RFM_Segmen
12346.0	326	2	0.00	1	1	1	111	3	L
12347.0	2	182	4310.00	4	4	4	444	12	H
12348.0	75	31	1797.24	2	2	4	224	8	M
12349.0	19	73	1757.55	3	3	4	334	10	H
12350.0	310	17	334.40	1	1	2	112	4	L

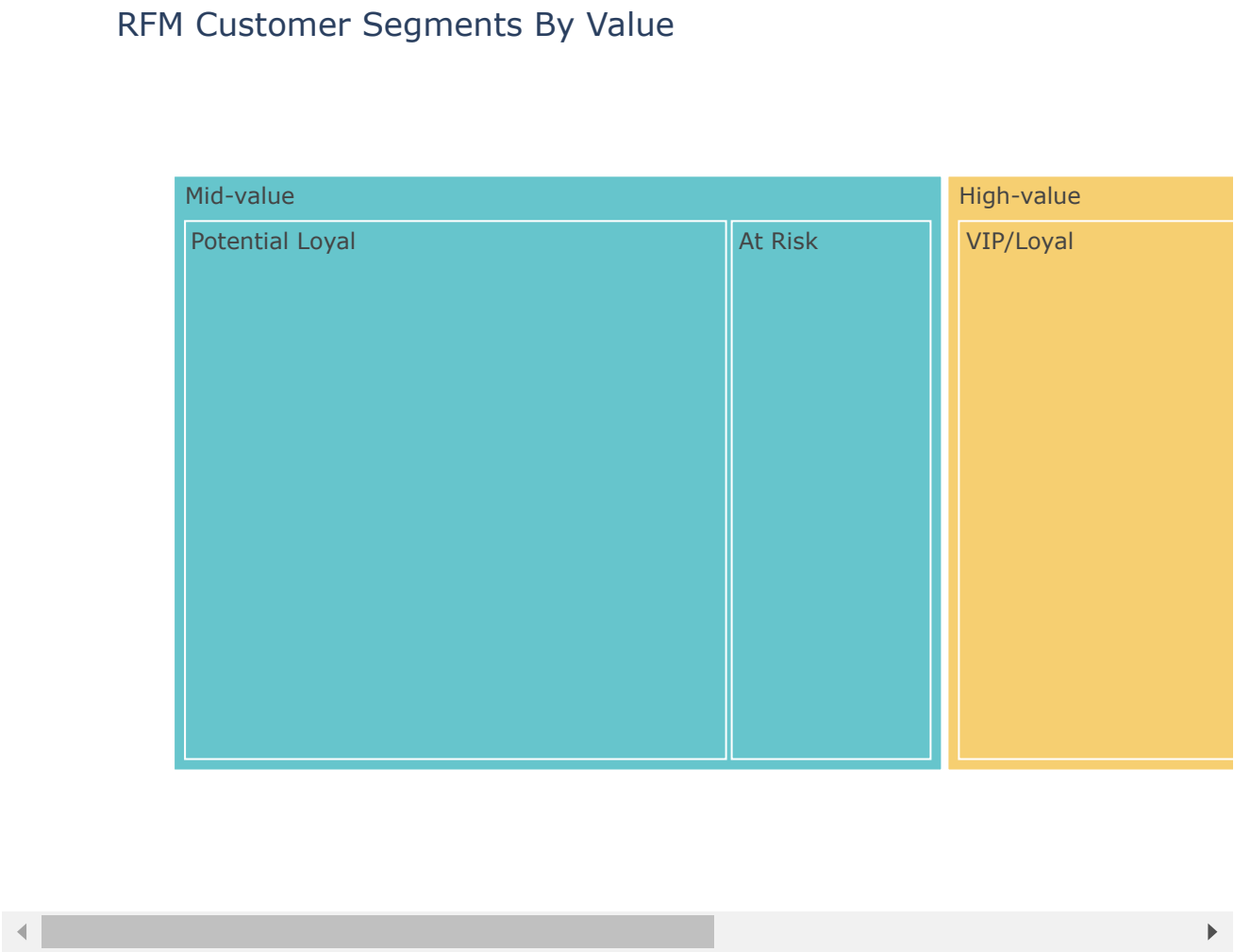
Next steps: Generate code with rfm | View recommended plots | New interactive sheet

```
segment_counts = rfm['RFM_Segment_Label'].value_counts().reset_index()
segment_counts.columns = ['RFM_Segment', 'Count']
segment_counts = segment_counts.sort_values('RFM_Segment')


# Create the bar char using plotly

fig = px.bar(segment_counts,
             x='RFM_Segment',
             y='Count',
             labels={'RFM_Segment': 'RFM Segment', 'Count': 'Number of Customers'},
             color = 'RFM_Segment',
             color_discrete_sequence=px.colors.qualitative.Pastel)
fig.show()
```

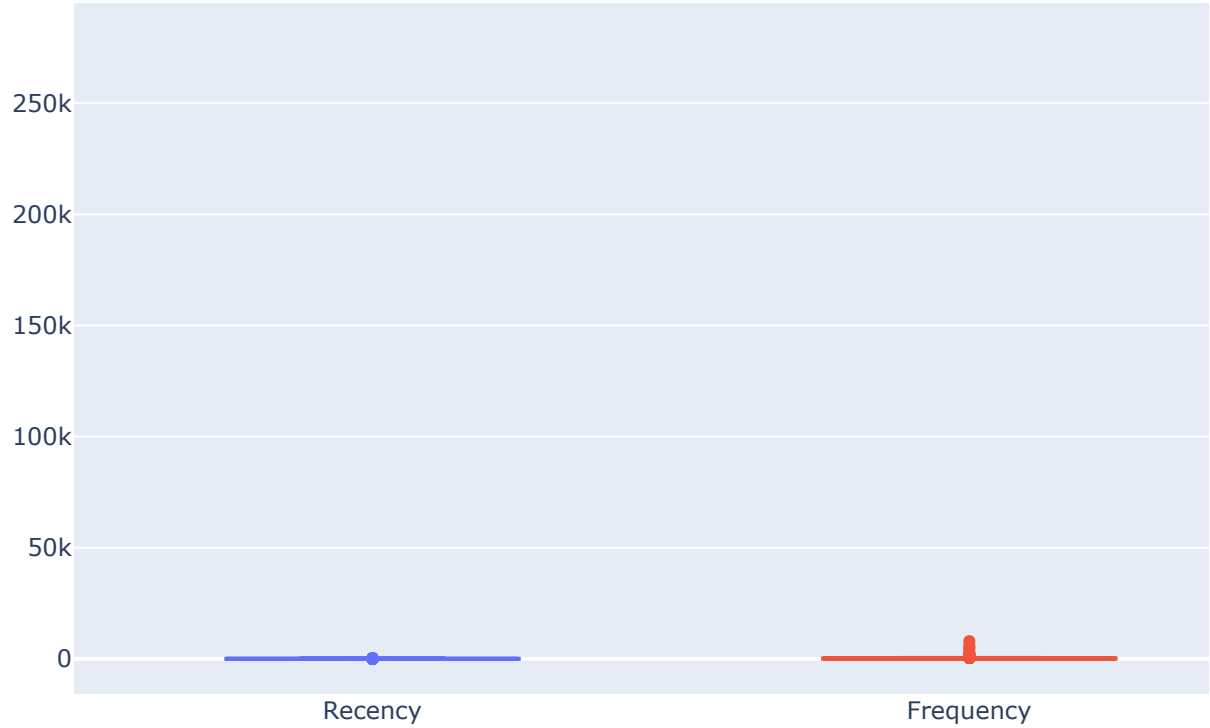



```
rfm['RFM_Customer_Segments']=''

rfm.loc[rfm['RFM_Score'] >= 9, 'RFM_Customer_Segments'] = 'VIP/Loyal'
rfm.loc[(rfm['RFM_Score'] >= 6) & (rfm['RFM_Score'] < 9), 'RFM_Customer_Segments'] = 'Po
rfm.loc[(rfm['RFM_Score'] >= 5) & (rfm['RFM_Score'] < 6), 'RFM_Customer_Segments'] = 'At
rfm.loc[(rfm['RFM_Score'] >= 4) & (rfm['RFM_Score'] < 5), 'RFM_Customer_Segments'] = 'Ca
rfm.loc[(rfm['RFM_Score'] >= 3) & (rfm['RFM_Score'] < 4), 'RFM_Customer_Segments'] = 'Lo
segment_counts= rfm['RFM_Customer_Segments'].value_counts().reset_index()



segment_product_counts = rfm.groupby(['RFM_Segment_Label', 'RFM_Customer_Segments']).siz
segment_product_counts = segment_product_counts.sort_values('Count', ascending=False)
```

```
# Create the treemap

fig_treemap_segment_product = px.treemap(segment_product_counts,
                                         path=['RFM_Segment_Label', 'RFM_Customer_Segmen
                                         values='Count',
                                         color='RFM_Segment_Label',
                                         color_discrete_sequence=px.colors.qualitative.P
                                         title='RFM Customer Segments By Value')
# Display the treemap
fig_treemap_segment_product.show()
```

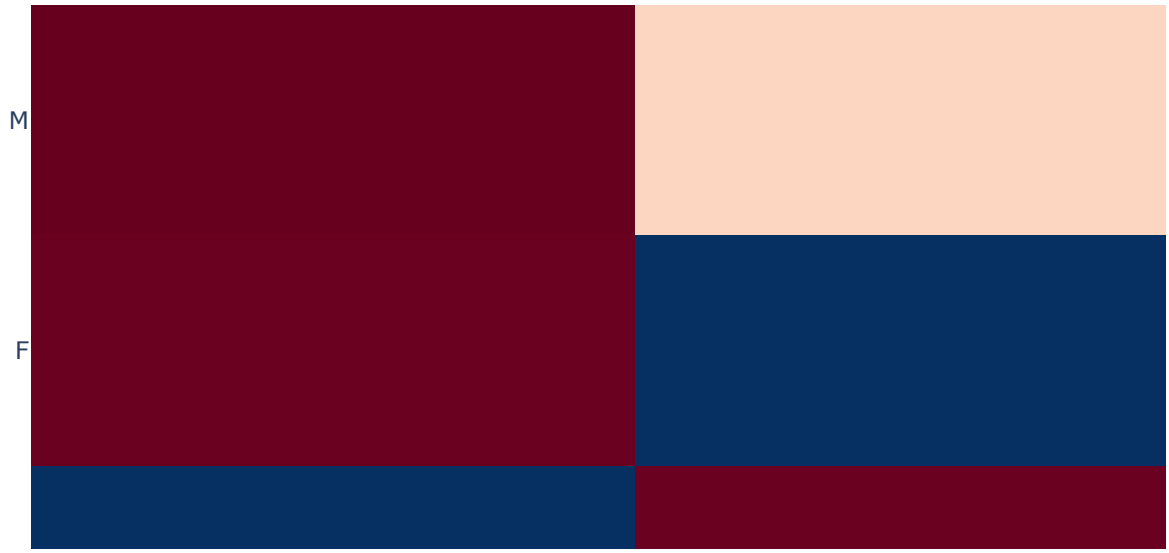



```
vip_segment = rfm[rfm['RFM_Customer_Segments'] == 'VIP/Loyal']


fig = go.Figure()
fig.add_trace(go.Box(y=vip_segment['Recency'], name='Recency'))
fig.add_trace(go.Box(y=vip_segment['Frequency'], name='Frequency'))
fig.add_trace(go.Box(y=vip_segment['Monetary'], name='Monetary'))
```




```
correlation_matrix = vip_segment[['R', 'F', 'M']].corr()


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 Heatmap')

# Display the heatmap
fig_heatmap.show()
```

Correlation Heatmap




```
segment_scores = rfm.groupby('RFM_Customer_Segments')[['R','F','M']].mean().reset_index(

fig=go.Figure()

# Add bars for Recency Score
fig.add_trace(go.Bar(
    x=segment_scores['RFM_Customer_Segments'],
    y=segment_scores['R'],
    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['F'],
    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['M'],
    name = 'Monetary Score'
```