Import Laibrary

In [79]:
```
import pandas as pd
import plotly.io as pio
pio.renderers.default = 'notebook'  # For Jupyter Notebook
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import plotly.express as px
```

Load or Read CSV file

In [80]:
```
df = pd.read_csv("RETAIL.csv",encoding="latin1")
```

Sample Data

In [81]:
```
df.head()
```

Out[81]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Cou
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	12/1/2010 8:26	2.55	17850.0	Un Kingc
1	536365	71053	WHITE METAL LANTERN	6	12/1/2010 8:26	3.39	17850.0	Un Kingc
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	12/1/2010 8:26	2.75	17850.0	Un Kingc
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	12/1/2010 8:26	3.39	17850.0	Un Kingc
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	12/1/2010 8:26	3.39	17850.0	Un Kingc

Basic information

In [82]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 541909 entries, 0 to 541908
Data columns (total 8 columns):
 #   Column       Non-Null Count   Dtype  
---  ------       --------------   -----  
 0   InvoiceNo    541909 non-null  object 
 1   StockCode    541909 non-null  object 
 2   Description  540455 non-null  object 
 3   Quantity     541909 non-null  int64  
 4   InvoiceDate  541909 non-null  object 
 5   UnitPrice    541909 non-null  float64
 6   CustomerID   406829 non-null  float64
 7   Country      541909 non-null  object 
dtypes: float64(2), int64(1), object(5)
memory usage: 33.1+ MB
```

Summary Statistics

In [83]:
```
df.describe()
```

Out[83]:

	Quantity	UnitPrice	CustomerID
count	541909.000000	541909.000000	406829.000000
mean	9.552250	4.611114	15287.690570
std	218.081158	96.759853	1713.600303
min	-80995.000000	-11062.060000	12346.000000
25%	1.000000	1.250000	13953.000000
50%	3.000000	2.080000	15152.000000
75%	10.000000	4.130000	16791.000000
max	80995.000000	38970.000000	18287.000000

Rows and columns

In [84]:
```
df.shape
```

Out[84]:
```
(541909, 8)
```

In [85]:
```
# Create TotalPrice column
df["TotalPrice"] = df["Quantity"] * df["UnitPrice"]
```

Revenue , Loss, Profit

In [86]:
```
# Revenue (only positive sales)
revenue = df[df['Quantity'] > 0]['TotalPrice'].sum()

# Losses (returns/cancellations)
losses = df[df['Quantity'] < 0]['TotalPrice'].sum()  # this will be negative

# Profit
profit = revenue + losses   # since losses is negative
print("Revenue:", revenue)
print("Losses:", losses)
print("Profit:", profit)
```

```
Revenue: 10644560.424
Losses: -896812.49
Profit: 9747747.934
```

Check null values

In [87]:
```
df.isnull().sum()
```

Out[87]:
```
InvoiceNo          0
StockCode          0
Description     1454
Quantity           0
InvoiceDate        0
UnitPrice          0
CustomerID     135080
Country            0
TotalPrice         0
dtype: int64
```

Handle missing Values

In [88]:
```
# Option: Fill missing product names
df['Description'] = df['Description'].fillna("Unknown")
```

In [89]:
```
df = df.dropna(subset=['CustomerID'])
```

In [90]:
```
df.isnull().sum()
```

Out[90]:
```
InvoiceNo      0
StockCode      0
Description    0
Quantity       0
InvoiceDate    0
UnitPrice      0
CustomerID     0
Country        0
TotalPrice     0
dtype: int64
```

In [91]:
```
df.head()
```

Out[91]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Cou
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	12/1/2010 8:26	2.55	17850.0	Un Kingc
1	536365	71053	WHITE METAL LANTERN	6	12/1/2010 8:26	3.39	17850.0	Un Kingc
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	12/1/2010 8:26	2.75	17850.0	Un Kingc
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	12/1/2010 8:26	3.39	17850.0	Un Kingc
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	12/1/2010 8:26	3.39	17850.0	Un Kingc

Separate Data and time columns

In [92]:

```
# Convert InvoiceDate to datetime
df["InvoiceDate"] = pd.to_datetime(df["InvoiceDate"])

# Create new columns
df["InvoiceDate_Date"] = df["InvoiceDate"].dt.date
df["InvoiceDate_Time"] = df["InvoiceDate"].dt.time

# Drop original InvoiceDate column (optional)
df = df.drop(columns=["InvoiceDate"])

# Show first few rows
print(df.head())
```

```
  InvoiceNo StockCode                          Description  Quantity  \
0    536365    85123A   WHITE HANGING HEART T-LIGHT HOLDER         6
1    536365     71053                  WHITE METAL LANTERN         6
2    536365    84406B       CREAM CUPID HEARTS COAT HANGER         8
3    536365    84029G  KNITTED UNION FLAG HOT WATER BOTTLE         6
4    536365    84029E       RED WOOLLY HOTTIE WHITE HEART.         6

   UnitPrice  CustomerID         Country  TotalPrice InvoiceDate_Date  \
0       2.55     17850.0  United Kingdom       15.30       2010-12-01
1       3.39     17850.0  United Kingdom       20.34       2010-12-01
2       2.75     17850.0  United Kingdom       22.00       2010-12-01
3       3.39     17850.0  United Kingdom       20.34       2010-12-01
4       3.39     17850.0  United Kingdom       20.34       2010-12-01

  InvoiceDate_Time
0         08:26:00
1         08:26:00
2         08:26:00
3         08:26:00
4         08:26:00
```

RFM ANALYSIS

In [93]:

```python
import datetime as dt

# Reference date = 1 day after last purchase
reference_date = df["InvoiceDate_Date"].max() + dt.timedelta(days=1)

# RFM calculation
rfm = df.groupby("CustomerID").agg({
    "InvoiceDate_Date": lambda x: (reference_date - x.max()).days,  # Recency
    "InvoiceNo": "nunique",                                         # Frequency
    "TotalPrice": "sum"                                             # Monetary
})

# Rename columns
rfm.rename(columns={
    "InvoiceDate_Date": "Recency",
    "InvoiceNo": "Frequency",
    "TotalPrice": "Monetary"
}, inplace=True)

print(rfm)
```

```
            Recency  Frequency  Monetary
CustomerID
12346.0         326          2      0.00
12347.0           3          7   4310.00
12348.0          76          4   1797.24
12349.0          19          1   1757.55
12350.0         311          1    334.40
...             ...        ...       ...
18280.0         278          1    180.60
18281.0         181          1     80.82
18282.0           8          3    176.60
18283.0           4         16   2094.88
18287.0          43          3   1837.28

[4372 rows x 3 columns]
```

Add RFM SCORE

In [94]:
```
# R Score (lower recency = better, so we reverse it)
rfm["R_Score"] = pd.qcut(rfm["Recency"], 5, labels=[5,4,3,2,1])

# F Score (higher frequency = better)
rfm["F_Score"] = pd.qcut(rfm["Frequency"].rank(method="first"), 5, labels=[1,2,3,4,

# M Score (higher monetary = better)
rfm["M_Score"] = pd.qcut(rfm["Monetary"], 5, labels=[1,2,3,4,5])

# Combine scores into RFM_Segment
rfm["RFM_Segment"] = rfm["R_Score"].astype(str) + rfm["F_Score"].astype(str) + rfm[

# Calculate RFM Score (average or sum)
rfm["RFM_Score"] = rfm[["R_Score","F_Score","M_Score"]].astype(int).sum(axis=1)

rfm.head()
```

Out[94]:

CustomerID	Recency	Frequency	Monetary	R_Score	F_Score	M_Score	RFM_Segment	RI
12346.0	326	2	0.00	1	2	1	121	
12347.0	3	7	4310.00	5	4	5	545	
12348.0	76	4	1797.24	2	3	4	234	
12349.0	19	1	1757.55	4	1	4	414	
12350.0	311	1	334.40	1	1	2	112	

Add segment of RFM_SCORE

In [95]:
```python
def segment_customer(score):
    if score >= 13:
        return "Champions"
    elif score >= 10:
        return "Loyal Customers"
    elif score >= 7:
        return "Potential Loyalist"
    elif score >= 5:
        return "At Risk"
    else:
        return "Lost Customers"

rfm["Segment"] = rfm["RFM_Score"].apply(segment_customer)

# View first 20 rows
print(rfm.head(20))
```

CustomerID	Recency	Frequency	Monetary	R_Score	F_Score	M_Score	RFM_Segment	RFM_Score	Segment
12346.0	326	2	0.00	1	2	1	121	4	Lost Customers
12347.0	3	7	4310.00	5	4	5	545	14	Champions
12348.0	76	4	1797.24	2	3	4	234	9	Potential Loyalist
12349.0	19	1	1757.55	4	1	4	414	9	Potential Loyalist
12350.0	311	1	334.40	1	1	2	112	4	Lost Customers
12352.0	37	11	1545.41	3	5	4	354	12	Loyal Customers
12353.0	205	1	89.00	1	1	1	111	3	Lost Customers
12354.0	233	1	1079.40	1	1	4	114	6	At Risk
12355.0	215	1	459.40	1	1	2	112	4	Lost Customers
12356.0	23	3	2811.43	4	3	5	435	12	Loyal Customers
12357.0	34	1	6207.67	3	1	5	315	9	Potential Loyalist
12358.0	2	2	1168.06	5	2	4	524	11	Loyal Customers
12359.0	8	6	6245.53	5	4	5	545	14	Champions
12360.0	53	3	2662.06	3	3	5	335	11	Loyal Customers
12361.0	288	1	189.90	1	1	1	111	3	Lost Customers
12362.0	4	13	5154.58	5	5	5	555	15	Champions
12363.0	110	2	552.00	2	2	3	223	7	Potential Loyalist
12364.0	8	4	1313.10	5	3	4	534	12	Loyal Customers
12365.0	292	3	320.69	1	3	2	132	6	At Risk
12367.0	5	1	168.90	5	1	1	511	7	Potential Loyalist

In [96]:
```
rfm.head()
```

Out[96]:

CustomerID	Recency	Frequency	Monetary	R_Score	F_Score	M_Score	RFM_Segment	R
12346.0	326	2	0.00	1	2	1	121	
12347.0	3	7	4310.00	5	4	5	545	
12348.0	76	4	1797.24	2	3	4	234	
12349.0	19	1	1757.55	4	1	4	414	
12350.0	311	1	334.40	1	1	2	112	

Recommend personalized marketing strategies for each customer segment.

In [97]:

```
def marketing_recommendation(segment):
    if segment == "Champions":
        return "Reward loyalty, VIP offers, ask for referrals"
    elif segment == "Loyal Customers":
        return "Upsell, cross-sell, loyalty tiers, personalized offers"
    elif segment == "Potential Loyalist":
        return "Special discounts, product bundles, personalized emails"
    elif segment == "At Risk":
        return "Win-back campaigns, limited-time discounts, product updates"
    elif segment == "Lost Customers":
        return "Re-engagement emails, deep discounts, or remove from campaigns"
    else:
        return "General marketing campaign"

# Add new column
rfm["Recommendation"] = rfm["Segment"].apply(marketing_recommendation)

# Preview
print(rfm.head(15))
```

CustomerID	Recency	Frequency	Monetary	R_Score	F_Score	M_Score	RFM_Segment	RFM_Score	Segment	Recommendation
12346.0	326	2	0.00	1	2	1	121	4	Lost Customers	Re-engagement emails, deep discounts, or remov...
12347.0	3	7	4310.00	5	4	5	545	14	Champions	Reward loyalty, VIP offers, ask for referrals
12348.0	76	4	1797.24	2	3	4	234	9	Potential Loyalist	Special discounts, product bundles, personaliz...
12349.0	19	1	1757.55	4	1	4	414	9	Potential Loyalist	Special discounts, product bundles, personaliz...
12350.0	311	1	334.40	1	1	2	112	4	Lost Customers	Re-engagement emails, deep discounts, or remov...
12352.0	37	11	1545.41	3	5	4	354	12	Loyal Customers	Upsell, cross-sell, loyalty tiers, personalize...
12353.0	205	1	89.00	1	1	1	111	3	Lost Customers	Re-engagement emails, deep discounts, or remov...
12354.0	233	1	1079.40	1	1	4	114	6	At Risk	Win-back campaigns, limited-time discounts, pr...
12355.0	215	1	459.40	1	1	2	112	4	Lost Customers	Re-engagement emails, deep discounts, or remov...
12356.0	23	3	2811.43	4	3	5	435	12	Loyal Customers	Upsell, cross-sell, loyalty tiers, personalize...
12357.0	34	1	6207.67	3	1	5	315	9	Potential Loyalist	Special discounts, product bundles, personaliz...
12358.0	2	2	1168.06	5	2	4	524	11	Loyal Customers	Upsell, cross-sell, loyalty tiers, personalize...
12359.0	8	6	6245.53	5	4	5	545	14	Champions	Reward loyalty, VIP offers, ask for referrals
12360.0	53	3	2662.06	3	3	5	335	11	Loyal Customers	Upsell, cross-sell, loyalty tiers, personalize...
12361.0	288	1	189.90	1	1	1	111	3	Lost Customers	Re-engagement emails, deep discounts, or remov...

In [98]:
```
rfm.head()
```

Out[98]:

CustomerID	Recency	Frequency	Monetary	R_Score	F_Score	M_Score	RFM_Segment	RI
12346.0	326	2	0.00	1	2	1	121	
12347.0	3	7	4310.00	5	4	5	545	
12348.0	76	4	1797.24	2	3	4	234	
12349.0	19	1	1757.55	4	1	4	414	
12350.0	311	1	334.40	1	1	2	112	

- # Heatmaps for Recency vs Frequency vs Monetary values.

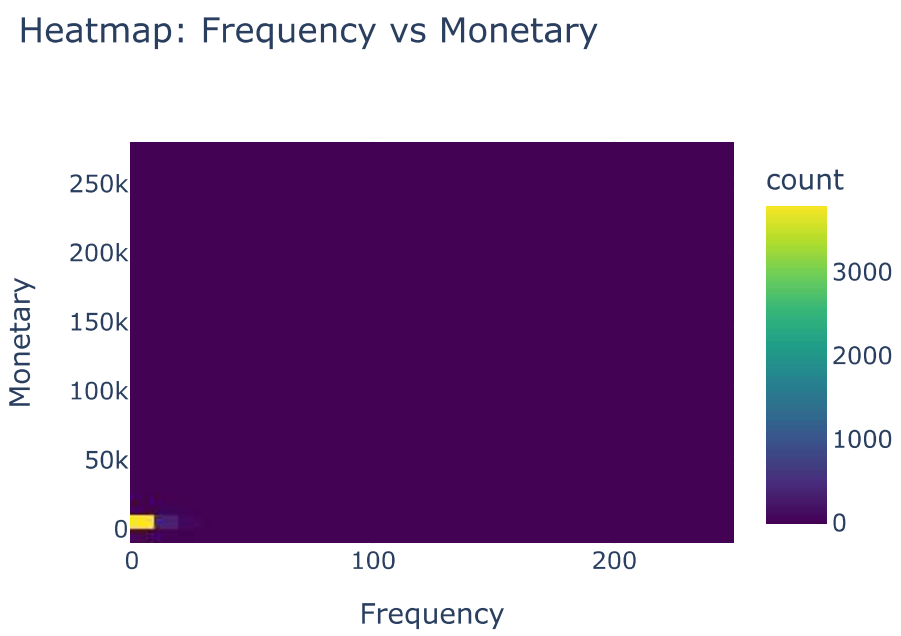
Recency vs Frequency

In [114…

```
fig = px.density_heatmap(
    rfm,
    x="Frequency",
    y="Recency",
    color_continuous_scale="Viridis",
    nbinsx=30,
    nbinsy=30,
    title="Heatmap: Recency vs Frequency"
)
fig.update_layout(
    width=480,   # Set the width in pixels (must be >= 10)
    height=380  # Set the height in pixels (must be >= 10)
)

fig.show()
fig.write_image("Recency vs Frequency.png")
```

Frequency VS Monetary

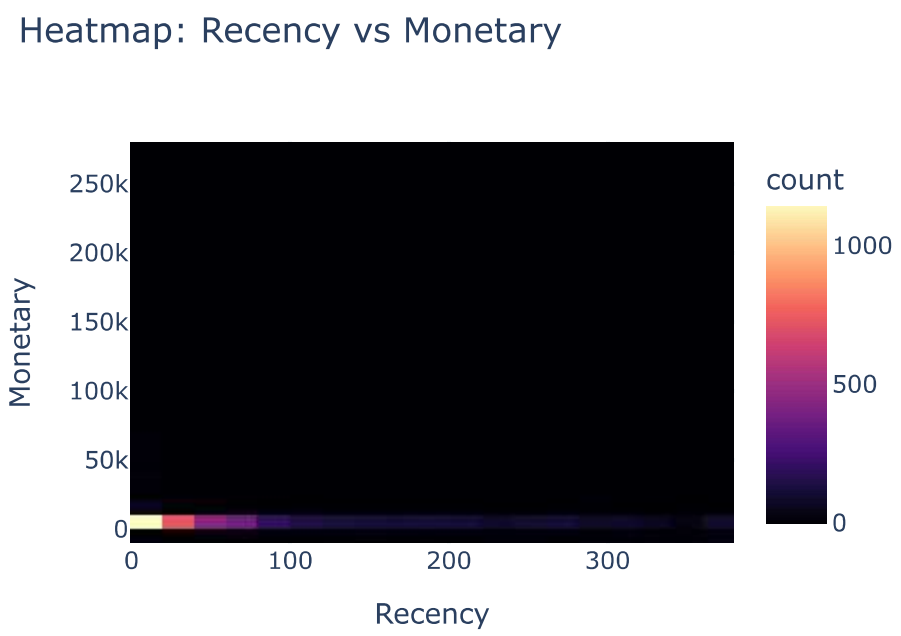
In [99]:
```
fig = px.density_heatmap(
    rfm,
    x="Frequency",
    y="Monetary",
    color_continuous_scale="Viridis",
    nbinsx=30,
    nbinsy=30,
    title="Heatmap: Frequency vs Monetary"
)
fig.update_layout(
    width=480,   # Set the width in pixels (must be >= 10)
    height=380  # Set the height in pixels (must be >= 10)
)

fig.show()
fig.write_html("mygraph.html", include_plotlyjs="inline")
```



Recency vs Monetary

In [100…

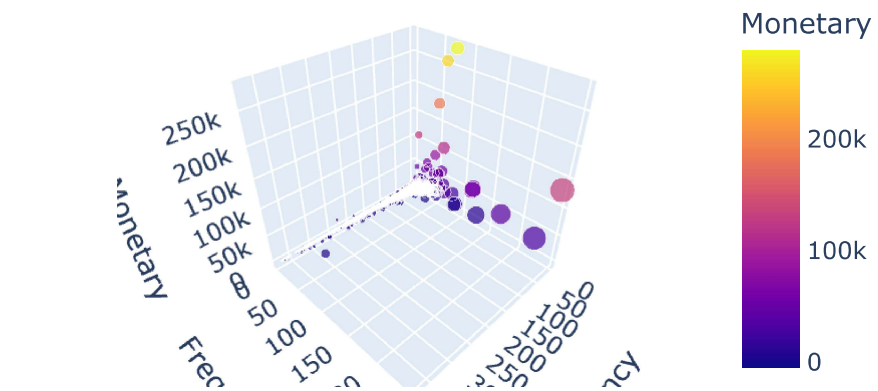
```
fig = px.density_heatmap(
    rfm,
    x="Recency",
    y="Monetary",
    color_continuous_scale="Magma",
    nbinsx=30,
    nbinsy=30,
    title="Heatmap: Recency vs Monetary"
)
fig.update_layout(
    width=480,   # Set the width in pixels (must be >= 10)
    height=380   # Set the height in pixels (must be >= 10)
)
fig.show()
fig.write_image("Recency vs Monetary.png")
```



Scatter of Recency , Frequency and Monetary

In [101…

```
import plotly.express as px

fig = px.scatter_3d(
    rfm,
    x="Recency",
    y="Frequency",
    z="Monetary",
    color="Monetary",        # color by Monetary
    size="Frequency",        # bubble size by Frequency
    hover_data=["Recency", "Frequency", "Monetary"],
    title="3D Scatter of RFM"
)
fig.update_layout(
    width=480,   # Set the width in pixels (must be >= 10)
    height=380   # Set the height in pixels (must be >= 10)
)
fig.show()

fig.write_image("Recency vs Monetary vs Frequency.png")
```

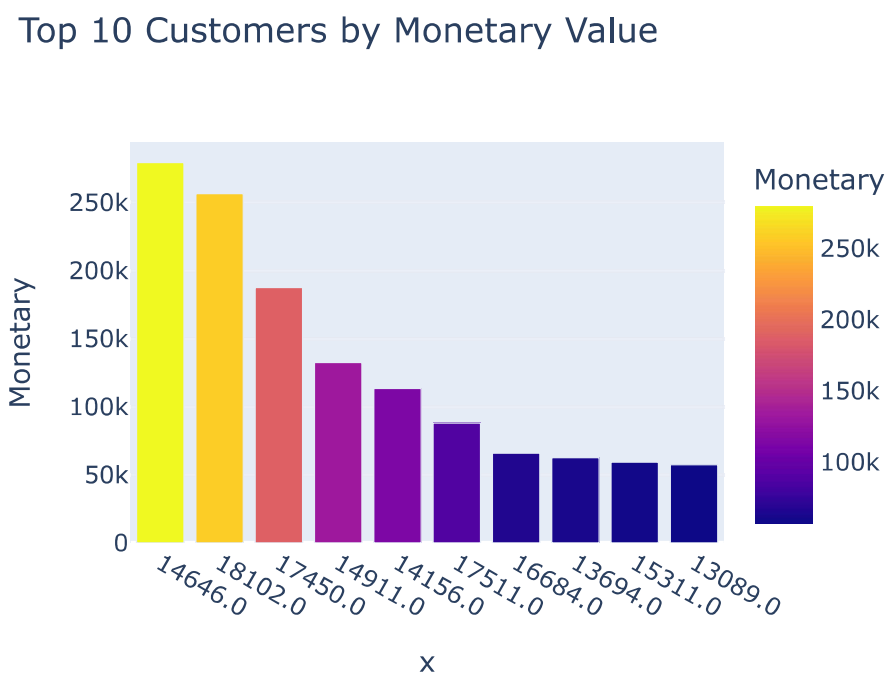


Top 10 Customers by Monetary Value

In [102…

```
# Sort customers: most recent, highest frequency, highest monetary
top_customers = rfm.sort_values(
    by=["RFM_Score", "Monetary"],
    ascending=[False, False]
).head(10)

print(top_customers)

# Convert index to string for display
fig = px.bar(top_customers, x=top_customers.index.astype(str), y="Monetary",color="
fig.update_layout(
    width=480,   # Set the width in pixels (must be >= 10)
    height=380 ) # Set the height in pixels (must be >= 10)
fig.show()
fig.write_image("Top 10 Customers vs Monetary.png")
```

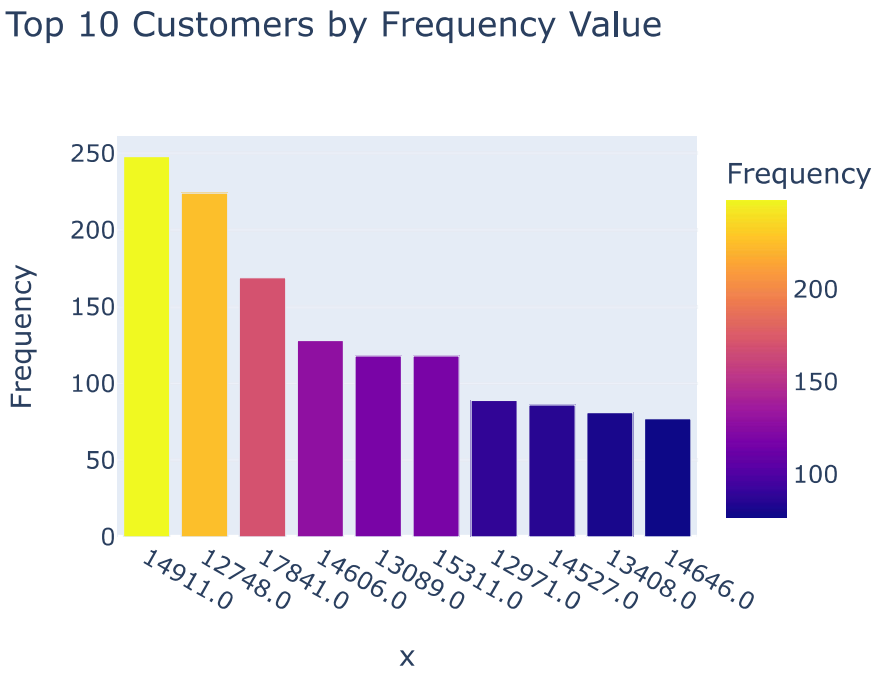
CustomerID	Recency	Frequency	Monetary	R_Score	F_Score	M_Score	RFM_Segment	RFM_Score	Segment	Recommendation
14646.0	2	77	279489.02	5	5	5	555	15	Champions	Reward loyalty, VIP offers, ask for referrals
18102.0	1	62	256438.49	5	5	5	555	15	Champions	Reward loyalty, VIP offers, ask for referrals
17450.0	9	55	187482.17	5	5	5	555	15	Champions	Reward loyalty, VIP offers, ask for referrals
14911.0	2	248	132572.62	5	5	5	555	15	Champions	Reward loyalty, VIP offers, ask for referrals
14156.0	10	66	113384.14	5	5	5	555	15	Champions	Reward loyalty, VIP offers, ask for referrals
17511.0	3	46	88125.38	5	5	5	555	15	Champions	Reward loyalty, VIP offers, ask for referrals
16684.0	5	31	65892.08	5	5	5	555	15	Champions	Reward loyalty, VIP offers, ask for referrals
13694.0	4	60	62653.10	5	5	5	555	15	Champions	Reward loyalty, VIP offers, ask for referrals
15311.0	1	118	59419.34	5	5	5	555	15	Champions	Reward loyalty, VIP offers, ask for referrals
13089.0	3	118	57385.88	5	5	5	555	15	Champions	Reward loyalty, VIP offers, ask for referrals



Top 10 Customers by Frequency Value

In [103…

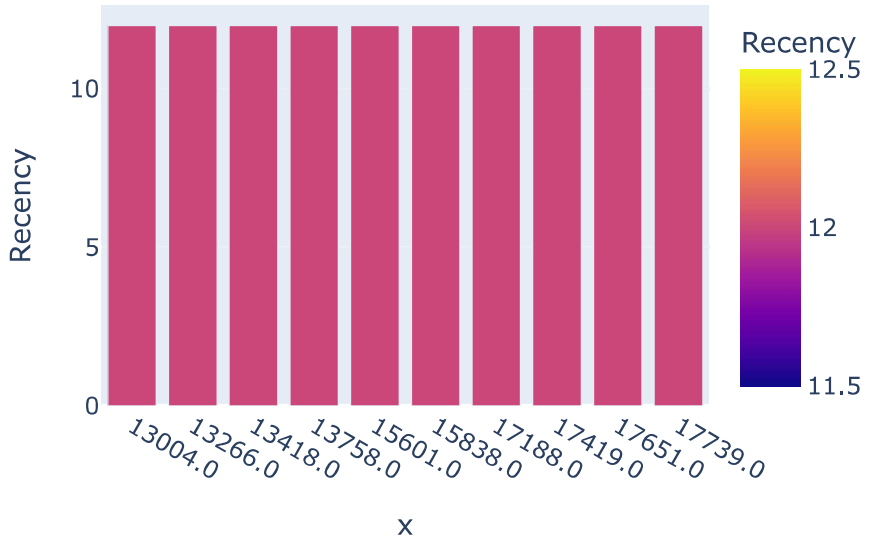
```
# Sort customers: most recent, highest frequency, highest monetary
top_customers = rfm.sort_values(
    by=["RFM_Score", "Frequency"],
    ascending=[False, False]
).head(10)



# Convert index to string for display
fig =px.bar(top_customers, x=top_customers.index.astype(str), y="Frequency",color="
            title="Top 10 Customers by Frequency Value")

fig.update_layout(
    width=480,   # Set the width in pixels (must be >= 10)
    height=380 )
fig.show() # Set the height in pixels (must be >= 10)
fig.write_image("Top 10  Customers vs Frequency.png")
```



Top 10 Customers by Recency Value

In [104…

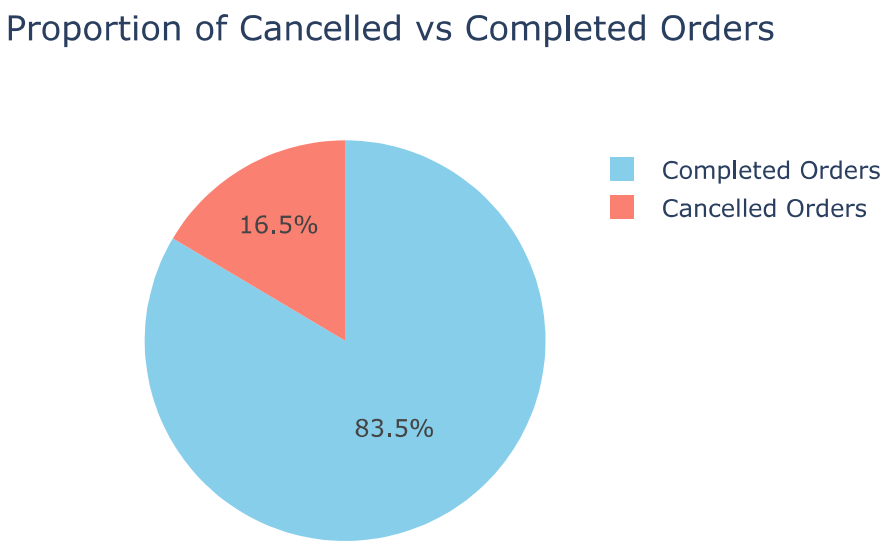
```
# Sort customers: most recent, highest frequency, highest monetary
top_customers = rfm.sort_values(
    by=["RFM_Score", "Recency"],
    ascending=[False, False]
).head(10)



# Convert index to string for display
fig = px.bar(top_customers, x=top_customers.index.astype(str), y="Recency",
             color="Recency" ,title="Top 10 Customers by Recency Value")
fig.update_layout(
    width=480,   # Set the width in pixels (must be >= 10)
    height=380 )
fig.show() # Set the height in pixels (must be >= 10)
fig.write_image("Top 10 Customers vs Recency.png")
```



Total Orders vs Cancelled Orders

In [105…

```
total_orders = df["InvoiceNo"].nunique()
cancelled_orders = df[df["Quantity"] < 0]["InvoiceNo"].nunique()
completed_orders = total_orders - cancelled_orders

# Prepare data
orders_summary = {
    "Order Status": ["Completed Orders", "Cancelled Orders"],
    "Count": [completed_orders, cancelled_orders]
}

# Create pie chart
fig = px.pie(
    orders_summary,
    names="Order Status",
    values="Count",
    color="Order Status",
    color_discrete_map={"Completed Orders": "skyblue", "Cancelled Orders": "salmon"
    title="Proportion of Cancelled vs Completed Orders"
)
fig.update_layout(
    width=480,   # Set the width in pixels (must be >= 10)
    height=380 ) # Set the height in pixels (must be >= 10)

fig.show()
fig.write_image("Proportion of Cancelled vs Completed Orders.png")
```



Total Order vs Cancelled Order per customer id

In [106…

```
import plotly.express as px

# Calculate total and cancelled orders per customer
customer_orders = df.groupby("CustomerID").agg(
    Total_Orders=("InvoiceNo", "nunique"),
    Cancelled_Orders=("InvoiceNo", lambda x: df.loc[x.index, "Quantity"].lt(0).grou
)

# Reset index to make CustomerID a column
customer_orders = customer_orders.reset_index()

# Convert CustomerID to string
customer_orders["CustomerID"] = customer_orders["CustomerID"].astype(str)

# Take top 10 customers by total orders
top10_customers = customer_orders.sort_values(
    ["Total_Orders", "Cancelled_Orders"], ascending=[False, False]
).head(10)

# Convert to long format for grouped bar plot
top10_melted = top10_customers.melt(
    id_vars="CustomerID",
    value_vars=["Total_Orders", "Cancelled_Orders"],
    var_name="Order_Type",
    value_name="Count"
)
```

```
# Bar plot
fig = px.bar(
    top10_melted,
    x="CustomerID",
    y="Count",
    color="Order_Type",
    barmode="group",
    text="Count",
    title="Top 10 Customers: Total Orders vs Cancelled Orders"
)
fig.update_layout(
    width=700,   # Set the width in pixels (must be >= 10)
    height=500) # Set the height in pixels (must be >= 10)

fig.show()
fig.write_image("Top 10 Customers: Total Orders vs Cancelled Orders.png")
```

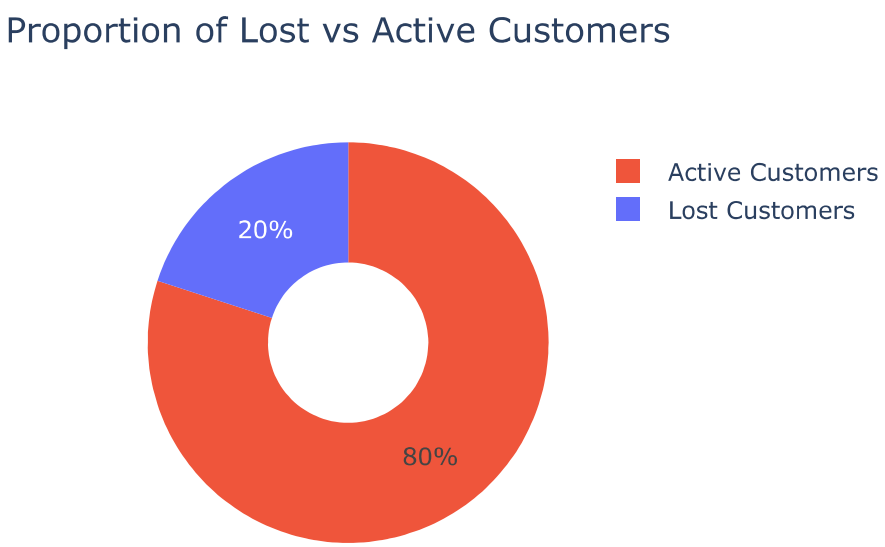
Total Customers Vs Lost Customers

In [107…

```
# Count customers per R_Score
customer_recency_groups = rfm["R_Score"].value_counts().sort_index()

# Total customers
```

```
total_customers = rfm.shape[0]

# Lost customers (R_Score = 1)
lost_customers = customer_recency_groups.loc[1]
lost_percent = (lost_customers / total_customers) * 100

print("Total Customers:", total_customers)
print("Lost Customers:", lost_customers)
print("Proportion Lost: {:.2f}%".format(lost_percent))


# Active customers = total - lost
active_customers = total_customers - lost_customers

# Data for pie chart
pie_data = {
    "Customer Status": ["Lost Customers", "Active Customers"],
    "Count": [lost_customers, active_customers]
}

# Create pie chart
fig = px.pie(
    pie_data,
    names="Customer Status",
    values="Count",
    hole=0.4,  # donut style (optional)
    color="Customer Status",
    title="Proportion of Lost vs Active Customers"
)
fig.update_layout(
    width=480,   # Set the width in pixels (must be >= 10)
    height=380 )

fig.show()
fig.write_image("Proportion of Lost vs Active Customers.png", width=794, height=112
```

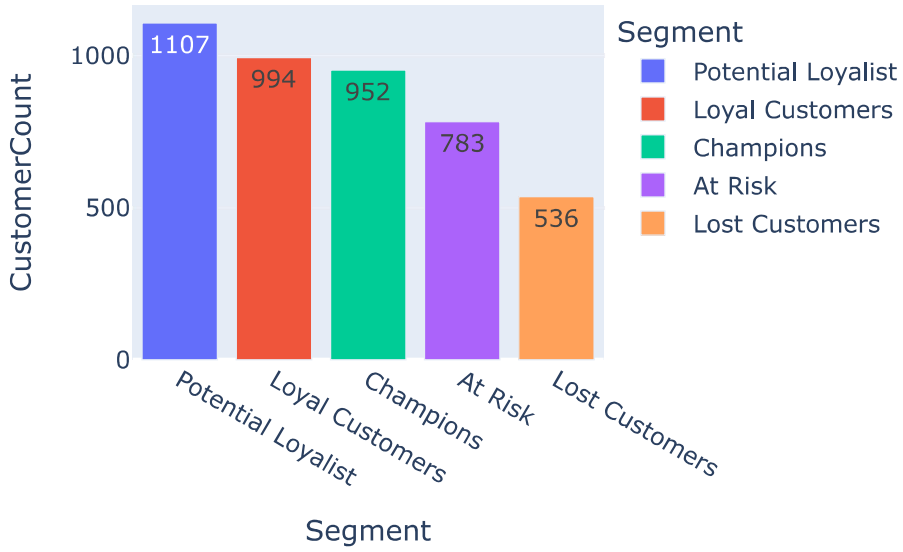
```
Total Customers: 4372
Lost Customers: 874
Proportion Lost: 19.99%
```



Total Customers across Segment

In [108…

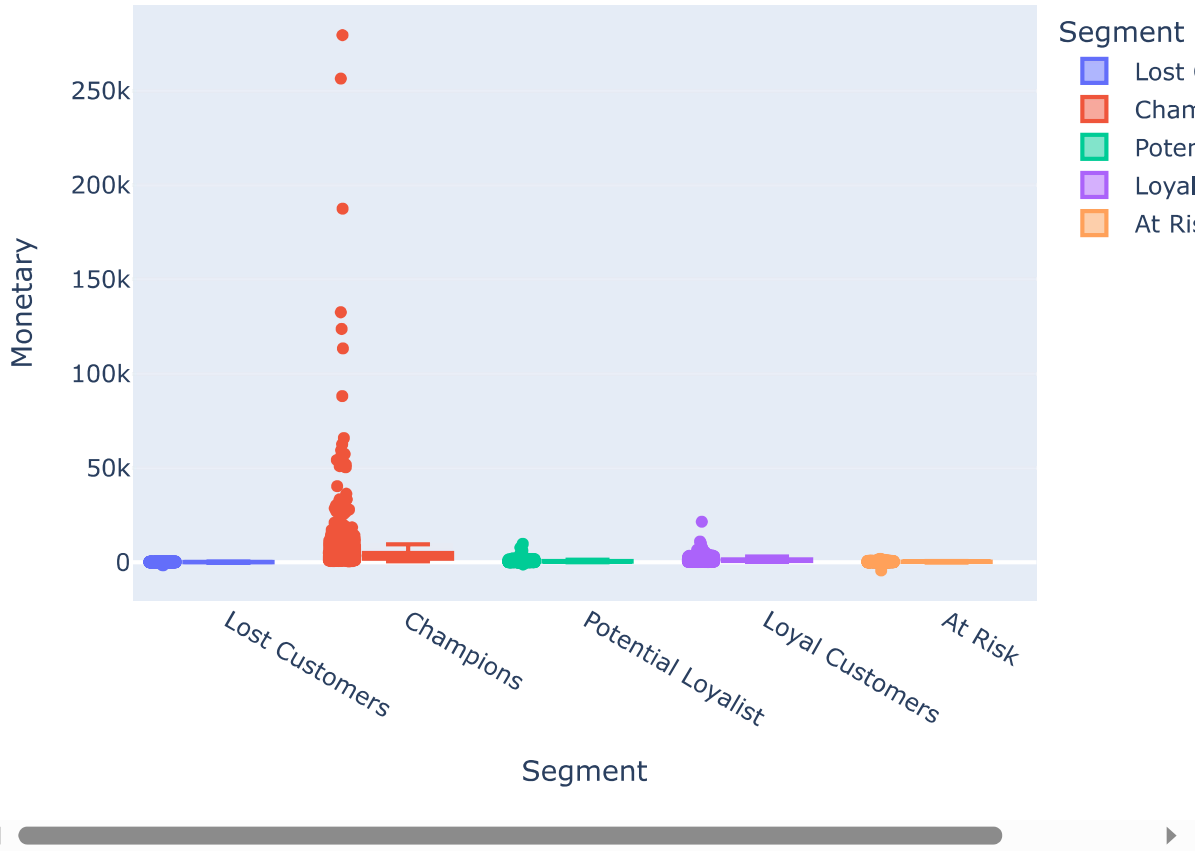
```
import plotly.express as px

# Count customers per segment
segment_counts = rfm["Segment"].value_counts().reset_index()
segment_counts.columns = ["Segment", "CustomerCount"]

# Plot interactive bar chart
fig = px.bar(
    segment_counts,
    x="Segment",
    y="CustomerCount",
    color="Segment",
    text="CustomerCount",
    title="Customer Count per RFM Segment",
)
fig.update_layout(
    width=480,   # Set the width in pixels (must be >= 10)
    height=380 )

fig.show()
fig.write_image("Customer Count per RFM Segment.png")
```



Boxplots for spending patterns across Monetary.

In [109…

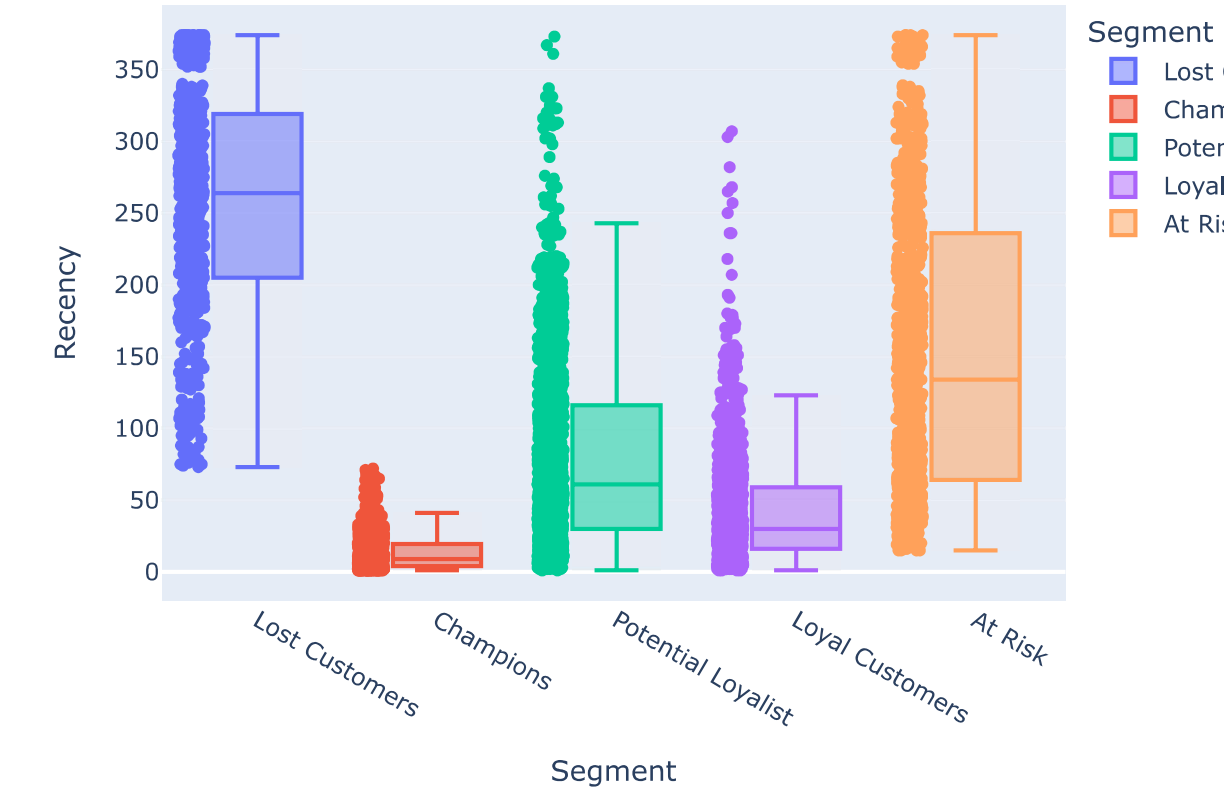
```
fig = px.box(
    rfm,
    x="Segment",          # or R_Score / F_Score if you don't have full Segment
    y="Monetary",
    color="Segment",      # color by group
    points="all",         # show all points (jittered)
    title="Spending Patterns (Monetary) Across RFM Segments"
)
fig.update_layout(
    width=700,   # Set the width in pixels (must be >= 10)
    height=500 )

fig.show()
fig.write_image("Spending Patterns (Monetary) Across RFM Segments.png")
```



Boxplots for spending patterns across Recency.

In [110…

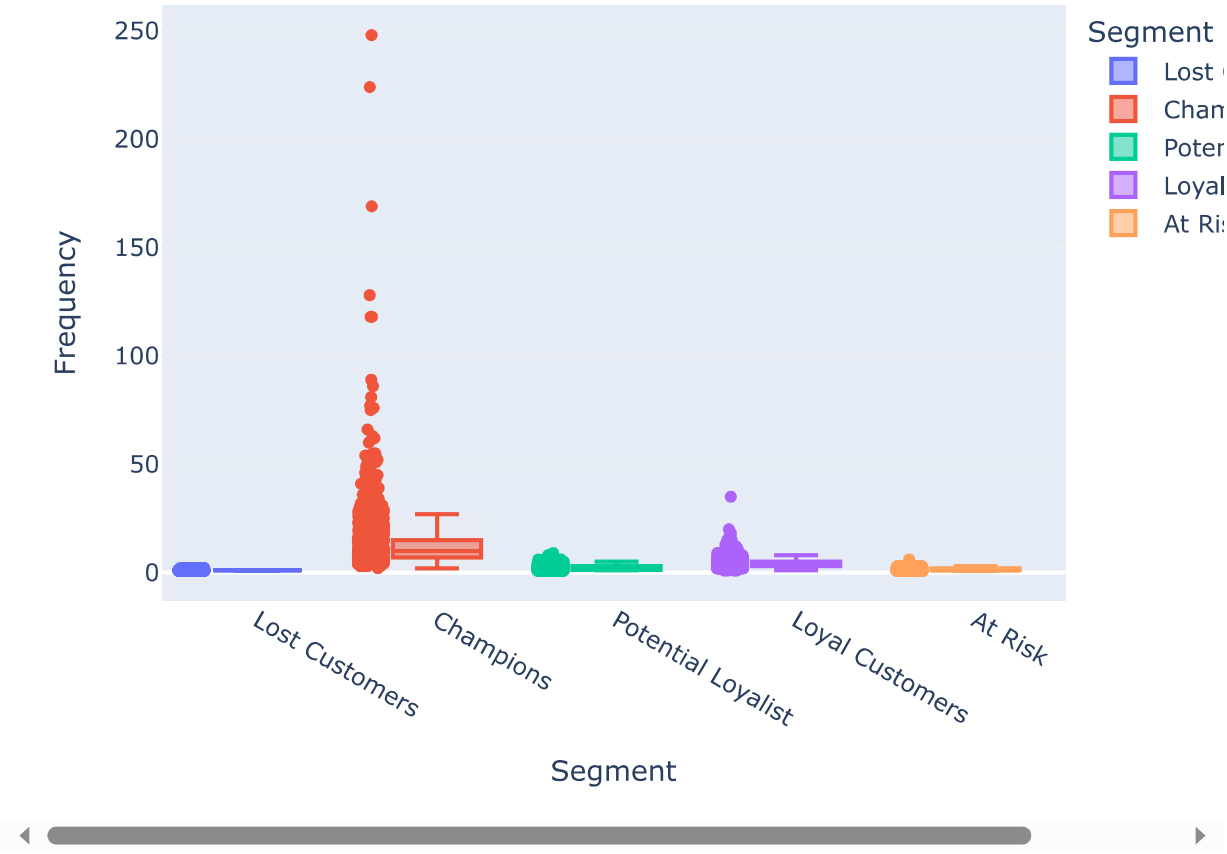
```
fig = px.box(
    rfm,
    x="Segment",          # or R_Score / F_Score if you don't have full Segment
    y="Recency",
    color="Segment",      # color by group
    points="all",         # show all points (jittered)
    title="Spending Patterns (Recency) Across RFM Segments"
)
fig.update_layout(
    width=700,    # Set the width in pixels (must be >= 10)
    height=500 )
fig.show()
fig.write_image("Spending Patterns (Recency) Across RFM Segments.png")
```



Boxplots for spending patterns across Frequency.

In [111…

```
fig = px.box(
    rfm,
    x="Segment",          # or R_Score / F_Score if you don't have full Segment
    y="Frequency",
    color="Segment",      # color by group
    points="all",         # show all points (jittered)
    title="Spending Patterns (Frequency) Across RFM Segments"
)
fig.update_layout(
    width=700,    # Set the width in pixels (must be >= 10)
    height=500 )


fig.show()
fig.write_image("Spending Patterns (Frequency) Across RFM Segments.png")
```


Spending Patterns (Frequency) Across RFM Segments
Frequency
250
200
150
100
50
0
Lost Customers
Champions
Potential Loyalist
Loyal Customers
At Risk
Segment
Segment
Lost
Chan
Poter
Loyal
At Ri