**#Step 1 | Importing The Libraries**

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

import datetime as dt

from sklearn.cluster import KMeans

import random

**#Step 2 | Loading and Reading the Dataset**

df = pd.read\_excel('/kaggle/input/online-retail-xlsx/Online Retail.xlsx')

df.head()

#Output:

|  | *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* |
| *3* | *536365* | *84029G* | *KNITTED UNION FLAG HOT WATER BOTTLE* | *6* | *2010-12-01 08:26:00* | *3.39* | *17850.0* | *United Kingdom* |
| *4* | *536365* | *84029E* | *RED WOOLLY HOTTIE WHITE HEART.* | *6* | *2010-12-01 08:26:00* | *3.39* | *17850.0* | *United Kingdom* |

#--Exploring Dataset Information:

df.info()

#Output:

*<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 datetime64[ns]*

*5 UnitPrice 541909 non-null float64*

*6 CustomerID 406829 non-null float64*

*7 Country 541909 non-null object*

*dtypes: datetime64[ns](1), float64(2), int64(1), object(4)*

*memory usage: 33.1+ MB*

df = df[df['CustomerID'].notnull()]

df.info()

*<class 'pandas.core.frame.DataFrame'>*

*Index: 406829 entries, 0 to 541908*

*Data columns (total 8 columns):*

*# Column Non-Null Count Dtype*

*--- ------ -------------- -----*

*0 InvoiceNo 406829 non-null object*

*1 StockCode 406829 non-null object*

*2 Description 406829 non-null object*

*3 Quantity 406829 non-null int64*

*4 InvoiceDate 406829 non-null datetime64[ns]*

*5 UnitPrice 406829 non-null float64*

*6 CustomerID 406829 non-null float64*

*7 Country 406829 non-null object*

*dtypes: datetime64[ns](1), float64(2), int64(1), object(4)*

*memory usage: 27.9+ MB*

**#Step 3 | Optimizing Data for Enhanced Clustering**

#--"Creating 'InvoiceDay' Column for Date-Based Analysis":

df['InvoiceDay'] = df['InvoiceDate'].apply(lambda x: dt.datetime(x.year, x.month, x.day))

df.head()

Output:

|  | *InvoiceNo* | *StockCode* | *Description* | *Quantity* | *InvoiceDate* | *UnitPrice* | *CustomerID* | *Country* | *InvoiceDay* |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *0* | *536365* | *85123A* | *WHITE HANGING HEART T-LIGHT HOLDER* | *6* | *2010-12-01 08:26:00* | *2.55* | *17850.0* | *United Kingdom* | *2010-12-01* |
| *1* | *536365* | *71053* | *WHITE METAL LANTERN* | *6* | *2010-12-01 08:26:00* | *3.39* | *17850.0* | *United Kingdom* | *2010-12-01* |
| *2* | *536365* | *84406B* | *CREAM CUPID HEARTS COAT HANGER* | *8* | *2010-12-01 08:26:00* | *2.75* | *17850.0* | *United Kingdom* | *2010-12-01* |
| *3* | *536365* | *84029G* | *KNITTED UNION FLAG HOT WATER BOTTLE* | *6* | *2010-12-01 08:26:00* | *3.39* | *17850.0* | *United Kingdom* | *2010-12-01* |
| *4* | *536365* | *84029E* | *RED WOOLLY HOTTIE WHITE HEART.* | *6* | *2010-12-01 08:26:00* | *3.39* | *17850.0* | *United Kingdom* | *2010-12-01* |

#--"Finding Last Purchase Date for Customer Management":

dt.timedelta(1)

Output:

*datetime.timedelta(days=1)*

pin\_date = max(df['InvoiceDay']) + dt.timedelta(1)

pin\_date

Output:

*Timestamp(‘2011-12-10 00:00:00’)*

#--"Creating 'TotalSum' Variable for Financial Analysis":

df['TotalSum'] = df['Quantity'] \* df['UnitPrice']

df.head()

Output:

|  | *InvoiceNo* | *StockCode* | *Description* | *Quantity* | *InvoiceDate* | *UnitPrice* | *CustomerID* | *Country* | *InvoiceDay* | *TotalSum* |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *0* | *536365* | *85123A* | *WHITE HANGING HEART T-LIGHT HOLDER* | *6* | *2010-12-01 08:26:00* | *2.55* | *17850.0* | *United Kingdom* | *2010-12-01* | *15.30* |
| *1* | *536365* | *71053* | *WHITE METAL LANTERN* | *6* | *2010-12-01 08:26:00* | *3.39* | *17850.0* | *United Kingdom* | *2010-12-01* | *20.34* |
| *2* | *536365* | *84406B* | *CREAM CUPID HEARTS COAT HANGER* | *8* | *2010-12-01 08:26:00* | *2.75* | *17850.0* | *United Kingdom* | *2010-12-01* | *22.00* |
| *3* | *536365* | *84029G* | *KNITTED UNION FLAG HOT WATER BOTTLE* | *6* | *2010-12-01 08:26:00* | *3.39* | *17850.0* | *United Kingdom* | *2010-12-01* | *20.34* |
| *4* | *536365* | *84029E* | *RED WOOLLY HOTTIE WHITE HEART.* | *6* | *2010-12-01 08:26:00* | *3.39* | *17850.0* | *United Kingdom* | *2010-12-01* | *20.34* |

#--Creating RFM Variables for Customer Analysis and Marketing Strategies:

rfm = df.groupby('CustomerID').agg({

'InvoiceDay': lambda x: (pin\_date - x.max()).days,

'InvoiceNo': 'count',

'TotalSum': 'sum'

})

rfm

Output:

|  | *InvoiceDay* | *InvoiceNo* | *TotalSum* |
| --- | --- | --- | --- |
| *CustomerID* |  |  |  |
| *12346.0* | *326* | *2* | *0.00* |
| *12347.0* | *3* | *182* | *4310.00* |
| *12348.0* | *76* | *31* | *1797.24* |
| *12349.0* | *19* | *73* | *1757.55* |
| *12350.0* | *311* | *17* | *334.40* |
| *...* | *...* | *...* | *...* |
| *18280.0* | *278* | *10* | *180.60* |
| *18281.0* | *181* | *7* | *80.82* |
| *18282.0* | *8* | *13* | *176.60* |
| *18283.0* | *4* | *756* | *2094.88* |
| *18287.0* | *43* | *70* | *1837.28* |

*4372 rows × 3 columns*

rfm.rename(columns= {

'InvoiceDay': 'Recency',

'InvoiceNo': 'Frequency',

'TotalSum': 'Monetary'

}, inplace=True)

rfm

Output:

|  | *Recency* | *Frequency* | *Monetary* |
| --- | --- | --- | --- |
| *CustomerID* |  |  |  |
| *12346.0* | *326* | *2* | *0.00* |
| *12347.0* | *3* | *182* | *4310.00* |
| *12348.0* | *76* | *31* | *1797.24* |
| *12349.0* | *19* | *73* | *1757.55* |
| *12350.0* | *311* | *17* | *334.40* |
| *...* | *...* | *...* | *...* |
| *18280.0* | *278* | *10* | *180.60* |
| *18281.0* | *181* | *7* | *80.82* |
| *18282.0* | *8* | *13* | *176.60* |
| *18283.0* | *4* | *756* | *2094.88* |
| *18287.0* | *43* | *70* | *1837.28* |

*4372 rows × 3 columns*

**#Step 4 | Data Preprocessing**

r\_labels = range(4, 0, -1) *#[4, 3, 2, 1]*

r\_groups = pd.qcut(rfm['Recency'], q=4, labels=r\_labels)

f\_labels = range(1, 5) *# [1, 2, 3, 4]*

f\_groups = pd.qcut(rfm['Frequency'], q=4, labels=f\_labels)

m\_labels = range(1, 5)

m\_groups = pd.qcut(rfm['Monetary'], q=4, labels=m\_labels)

rfm['R'] = r\_groups.values

rfm['F'] = f\_groups.values

rfm['M'] = m\_groups.values

rfm

Output:

|  | *Recency* | *Frequency* | *Monetary* | *R* | *F* | *M* |
| --- | --- | --- | --- | --- | --- | --- |
| *CustomerID* |  |  |  |  |  |  |
| *12346.0* | *326* | *2* | *0.00* | *1* | *1* | *1* |
| *12347.0* | *3* | *182* | *4310.00* | *4* | *4* | *4* |
| *12348.0* | *76* | *31* | *1797.24* | *2* | *2* | *4* |
| *12349.0* | *19* | *73* | *1757.55* | *3* | *3* | *4* |
| *12350.0* | *311* | *17* | *334.40* | *1* | *1* | *2* |
| *...* | *...* | *...* | *...* | *...* | *...* | *...* |
| *18280.0* | *278* | *10* | *180.60* | *1* | *1* | *1* |
| *18281.0* | *181* | *7* | *80.82* | *1* | *1* | *1* |
| *18282.0* | *8* | *13* | *176.60* | *4* | *1* | *1* |
| *18283.0* | *4* | *756* | *2094.88* | *4* | *4* | *4* |
| *18287.0* | *43* | *70* | *1837.28* | *3* | *3* | *4* |

*4372 rows × 6 columns*

**#Step 5 | Customer Clustering for Targeted Marketing**

X = rfm[['R', 'F', 'M']]

kmeans = KMeans(n\_clusters=10, init='k-means++', max\_iter=300)

kmeans.fit(X)

Output:

*KMeans*

*KMeans(n\_clusters=10)*

kmeans.labels\_

Output:

*array([9, 3, 8, ..., 5, 3, 6], dtype=int32)*

rfm['kmeans\_cluster'] = kmeans.labels\_

rfm

Output:

|  | *Recency* | *Frequency* | *Monetary* | *R* | *F* | *M* | *kmeans\_cluster* |
| --- | --- | --- | --- | --- | --- | --- | --- |
| *CustomerID* |  |  |  |  |  |  |  |
| *12346.0* | *326* | *2* | *0.00* | *1* | *1* | *1* | *9* |
| *12347.0* | *3* | *182* | *4310.00* | *4* | *4* | *4* | *3* |
| *12348.0* | *76* | *31* | *1797.24* | *2* | *2* | *4* | *8* |
| *12349.0* | *19* | *73* | *1757.55* | *3* | *3* | *4* | *6* |
| *12350.0* | *311* | *17* | *334.40* | *1* | *1* | *2* | *9* |
| *...* | *...* | *...* | *...* | *...* | *...* | *...* | *...* |
| *18280.0* | *278* | *10* | *180.60* | *1* | *1* | *1* | *9* |
| *18281.0* | *181* | *7* | *80.82* | *1* | *1* | *1* | *9* |
| *18282.0* | *8* | *13* | *176.60* | *4* | *1* | *1* | *5* |
| *18283.0* | *4* | *756* | *2094.88* | *4* | *4* | *4* | *3* |
| *18287.0* | *43* | *70* | *1837.28* | *3* | *3* | *4* | *6* |

*4372 rows × 7 columns*

rfm[rfm['kmeans\_cluster'] == 0]

Output:

|  | *Recency* | *Frequency* | *Monetary* | *R* | *F* | *M* | *kmeans\_cluster* |
| --- | --- | --- | --- | --- | --- | --- | --- |
| *CustomerID* |  |  |  |  |  |  |  |
| *12363.0* | *110* | *23* | *552.00* | *2* | *2* | *2* | *0* |
| *12365.0* | *292* | *23* | *320.69* | *1* | *2* | *2* | *0* |
| *12390.0* | *80* | *32* | *549.84* | *2* | *2* | *2* | *0* |
| *12414.0* | *218* | *18* | *562.41* | *1* | *2* | *2* | *0* |
| *12420.0* | *64* | *29* | *600.39* | *2* | *2* | *2* | *0* |
| *...* | *...* | *...* | *...* | *...* | *...* | *...* | *...* |
| *18212.0* | *327* | *43* | *248.42* | *1* | *3* | *1* | *0* |
| *18218.0* | *207* | *24* | *626.38* | *1* | *2* | *2* | *0* |
| *18222.0* | *93* | *19* | *443.00* | *2* | *2* | *2* | *0* |
| *18232.0* | *82* | *40* | *582.47* | *2* | *2* | *2* | *0* |
| *18250.0* | *302* | *22* | *342.92* | *1* | *2* | *2* | *0* |

*591 rows × 7 columns*

**#Step 6 | Customer Clustering Visualization**

*# Number of clusters*

num\_clusters = 10

*# Create subplots with two clusters in each row*

fig, axes = plt.subplots(num\_clusters // 2, 2, figsize=(12, 20))

*# Flatten the axes array to iterate through subplots*

axes = axes.ravel()

*# Loop through each cluster and plot it*

for cluster\_id **in** range(num\_clusters):

*# Filter data for the current cluster*

cluster\_data = rfm[rfm['kmeans\_cluster'] == cluster\_id]

*# Plot the data with a distinct color*

sns.scatterplot(data=cluster\_data, x='Recency', y='Frequency', hue='Monetary', palette='viridis', ax=axes[cluster\_id])

*# Set the title for the subplot*

axes[cluster\_id].set\_title(f'Cluster **{**cluster\_id**}**')

*# Customize axes labels, if needed*

*# axes[cluster\_id].set\_xlabel('X-axis Label')*

*# axes[cluster\_id].set\_ylabel('Y-axis Label')*

*# Add a common legend*

handles, labels = axes[0].get\_legend\_handles\_labels()

fig.legend(handles, labels, loc='center right')

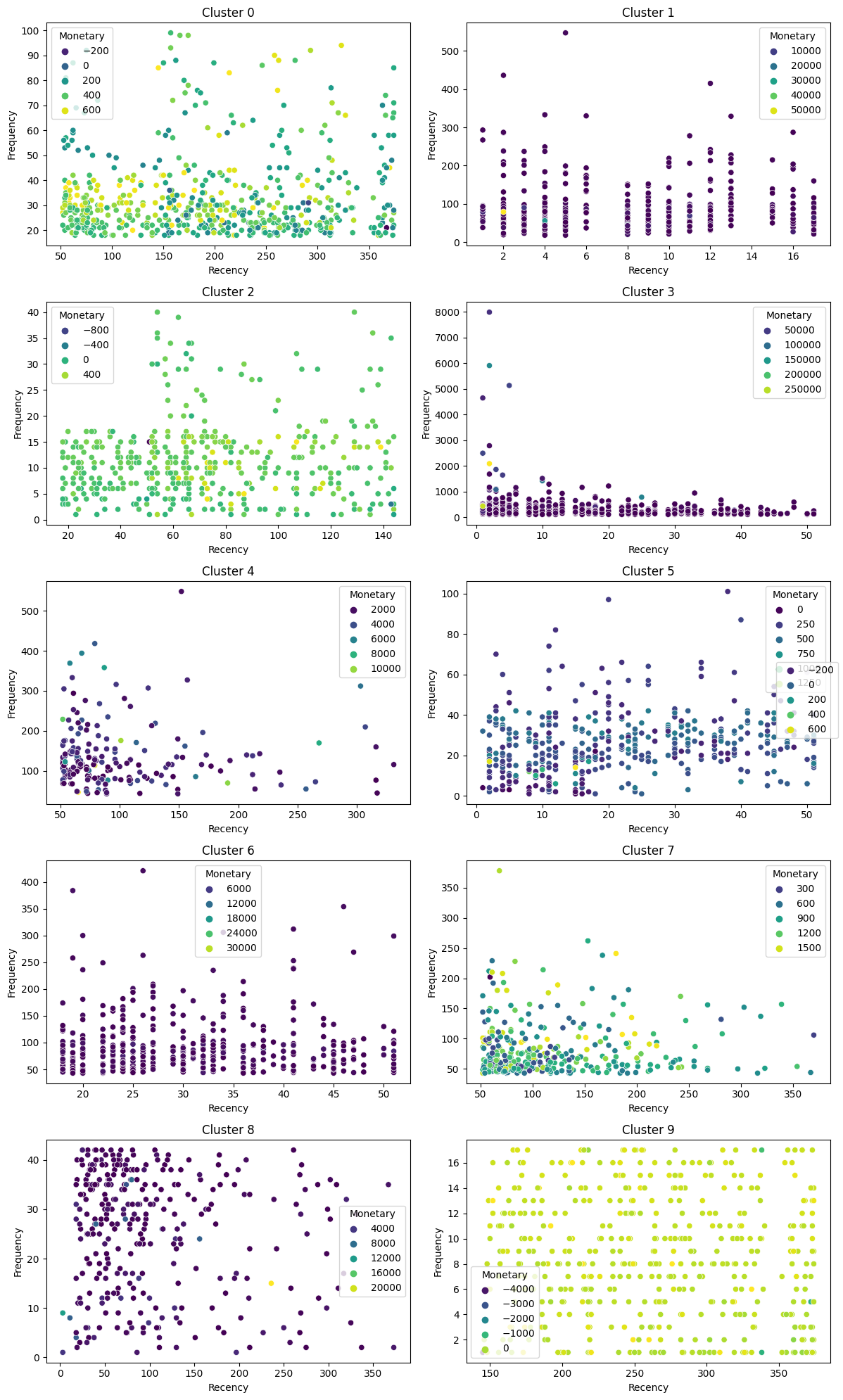
*# Adjust subplot spacing*

plt.tight\_layout()

*# Show the plot*

plt.show()

Output:



*# Create a histogram for Recency in each cluster*

plt.figure(figsize=(12, 6))

for cluster\_id **in** range(num\_clusters):

plt.subplot(2, 5, cluster\_id + 1)

sns.histplot(rfm[rfm['kmeans\_cluster'] == cluster\_id]['Recency'], bins=20, kde=True)

plt.title(f'Cluster **{**cluster\_id**}**')

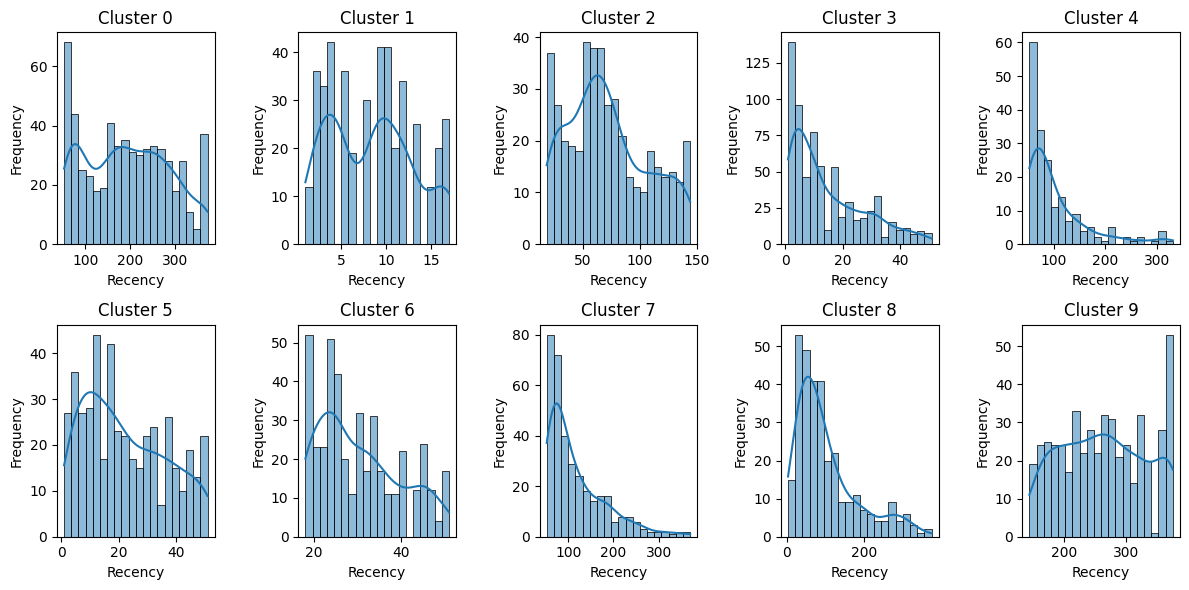
plt.xlabel('Recency')

plt.ylabel('Frequency')

plt.tight\_layout()

plt.show()

Output:



**#Step 7 | Creating a Recommendation System**

Generating Top Product Recommendations for Each Cluster

# Number of clusters (groups)

num\_clusters = 10

# Create an empty dictionary to store recommendations for each cluster

cluster\_recommendations = {}

# Loop through each cluster

for cluster\_id **in** range(num\_clusters):

# Find customers in the current cluster

customers\_in\_cluster = rfm[rfm['kmeans\_cluster'] == cluster\_id].index

# Find top products for customers in the current cluster

top\_products\_for\_cluster = df[df['CustomerID'].isin(customers\_in\_cluster)].groupby(['StockCode'])['InvoiceNo'].count().sort\_values(ascending=False).head(10)

# Store the top products for the current cluster in the dictionary

cluster\_recommendations[f'Cluster **{**cluster\_id**}**'] = top\_products\_for\_cluster.index.tolist()

# Display the recommendations for each cluster

for cluster, recommended\_products **in** cluster\_recommendations.items():

print(f"**{**cluster**}** -> Recommended Products: **{**recommended\_products**}**")

Output:

*Cluster 0 -> Recommended Products: ['85123A', 22423, 47566, 21034, 22960, 22457, 84879, 22720, 22138, 22470]*

*Cluster 1 -> Recommended Products: [23084, 22423, '85123A', 21034, 22086, 'POST', 22469, 22197, 23355, 22178]*

*Cluster 2 -> Recommended Products: ['85123A', 84946, 84879, 'POST', 22423, '85099B', 22138, 'M', 22487, 22577]*

*Cluster 3 -> Recommended Products: ['85099B', '85123A', 22423, 20725, 47566, 23203, 20727, 23209, 22383, 84879]*

*Cluster 4 -> Recommended Products: [22423, '85123A', 47566, '85099B', 22720, 84879, 22960, 23298, 21212, 20725]*

*Cluster 5 -> Recommended Products: [84879, 22086, 21034, 23084, 22423, 'POST', 22138, '85123A', 23355, 21790]*

*Cluster 6 -> Recommended Products: ['85123A', 22423, 21034, 84879, '85099B', 22197, 22469, 22086, 22138, 20727]*

*Cluster 7 -> Recommended Products: ['85123A', 84879, 22423, 47566, 22720, 22960, 21212, 22470, 21034, 22457]*

*Cluster 8 -> Recommended Products: [47566, 'POST', 22423, '85123A', 84879, '85099B', 22138, 22086, 22699, 22720]*

*Cluster 9 -> Recommended Products: ['85123A', 22423, 47566, 'POST', 22178, 22699, 22469, 22457, 22427, 22969]*

#--Cluster Analysis: Product Recommendations

def generate\_cluster\_recommendations(num\_clusters, num\_customers\_to\_display, rfm, df):

# Create an empty dictionary to store recommendations for each cluster

cluster\_recommendations = {}

# Loop through each cluster

for cluster\_id **in** range(num\_clusters):

# Find customers in the current cluster

customers\_in\_cluster = rfm[rfm['kmeans\_cluster'] == cluster\_id].index

# Find top products for customers in the current cluster

top\_products\_for\_cluster = df[df['CustomerID'].isin(customers\_in\_cluster)].groupby(['StockCode'])['InvoiceNo'].count().sort\_values(ascending=False).head(10)

# Find customers who haven't purchased any of the top products in the current cluster

non\_buyers = [customer for customer **in** customers\_in\_cluster if **not** (df[(df['CustomerID'] == customer) & (df['StockCode'].isin(top\_products\_for\_cluster.index.tolist()))]).empty]

# Limit the number of non-buyers to the specified number

num\_customers\_to\_display = min(num\_customers\_to\_display, len(non\_buyers))

# Select non-buyer customers for the current cluster

selected\_customers = non\_buyers[:num\_customers\_to\_display]

# Store the top products and selected non-buyer customers for the current cluster in the dictionary

cluster\_recommendations[f'Cluster **{**cluster\_id**}**'] = {

'Recommended Products': top\_products\_for\_cluster.index.tolist(),

'Selected Non-Buyer Customers': selected\_customers

}

return cluster\_recommendations

# Example usage:

num\_clusters = 10

num\_customers\_to\_display = 5

# Assuming you already have 'rfm' and 'df' dataframes

cluster\_recommendations = generate\_cluster\_recommendations(num\_clusters, num\_customers\_to\_display, rfm, df)

# Display the recommendations and selected non-buyer customers for each cluster

for cluster, recommendations\_and\_customers **in** cluster\_recommendations.items():

print(f"**{**cluster**}** ->")

print("Recommended Products:")

for customer\_id **in** recommendations\_and\_customers['Selected Non-Buyer Customers']:

print(f"Customer: **{**customer\_id**}** =====>>>> Recommended Products: **{**recommendations\_and\_customers['Recommended Products']**}**")

print()

Output:

*Cluster 0 ->*

*Recommended Products:*

*Customer: 12365.0 =====>>>> Recommended Products: ['85123A', 22423, 47566, 21034, 22960, 22457, 84879, 22720, 22138, 22470]*

*Customer: 12414.0 =====>>>> Recommended Products: ['85123A', 22423, 47566, 21034, 22960, 22457, 84879, 22720, 22138, 22470]*

*Customer: 12420.0 =====>>>> Recommended Products: ['85123A', 22423, 47566, 21034, 22960, 22457, 84879, 22720, 22138, 22470]*

*Customer: 12426.0 =====>>>> Recommended Products: ['85123A', 22423, 47566, 21034, 22960, 22457, 84879, 22720, 22138, 22470]*

*Customer: 12506.0 =====>>>> Recommended Products: ['85123A', 22423, 47566, 21034, 22960, 22457, 84879, 22720, 22138, 22470]*

*Cluster 1 ->*

*Recommended Products:*

*Customer: 12358.0 =====>>>> Recommended Products: [23084, 22423, '85123A', 21034, 22086, 'POST', 22469, 22197, 23355, 22178]*

*Customer: 12364.0 =====>>>> Recommended Products: [23084, 22423, '85123A', 21034, 22086, 'POST', 22469, 22197, 23355, 22178]*

*Customer: 12381.0 =====>>>> Recommended Products: [23084, 22423, '85123A', 21034, 22086, 'POST', 22469, 22197, 23355, 22178]*

*Customer: 12388.0 =====>>>> Recommended Products: [23084, 22423, '85123A', 21034, 22086, 'POST', 22469, 22197, 23355, 22178]*

*Customer: 12421.0 =====>>>> Recommended Products: [23084, 22423, '85123A', 21034, 22086, 'POST', 22469, 22197, 23355, 22178]*

*Cluster 2 ->*

*Recommended Products:*

*Customer: 12430.0 =====>>>> Recommended Products: ['85123A', 84946, 84879, 'POST', 22423, '85099B', 22138, 'M', 22487, 22577]*

*Customer: 12436.0 =====>>>> Recommended Products: ['85123A', 84946, 84879, 'POST', 22423, '85099B', 22138, 'M', 22487, 22577]*

*Customer: 12445.0 =====>>>> Recommended Products: ['85123A', 84946, 84879, 'POST', 22423, '85099B', 22138, 'M', 22487, 22577]*

*Customer: 12454.0 =====>>>> Recommended Products: ['85123A', 84946, 84879, 'POST', 22423, '85099B', 22138, 'M', 22487, 22577]*

*Customer: 12492.0 =====>>>> Recommended Products: ['85123A', 84946, 84879, 'POST', 22423, '85099B', 22138, 'M', 22487, 22577]*

*Cluster 3 ->*

*Recommended Products:*

*Customer: 12347.0 =====>>>> Recommended Products: ['85099B', '85123A', 22423, 20725, 47566, 23203, 20727, 23209, 22383, 84879]*

*Customer: 12357.0 =====>>>> Recommended Products: ['85099B', '85123A', 22423, 20725, 47566, 23203, 20727, 23209, 22383, 84879]*

*Customer: 12359.0 =====>>>> Recommended Products: ['85099B', '85123A', 22423, 20725, 47566, 23203, 20727, 23209, 22383, 84879]*

*Customer: 12362.0 =====>>>> Recommended Products: ['85099B', '85123A', 22423, 20725, 47566, 23203, 20727, 23209, 22383, 84879]*

*Customer: 12380.0 =====>>>> Recommended Products: ['85099B', '85123A', 22423, 20725, 47566, 23203, 20727, 23209, 22383, 84879]*

*Cluster 4 ->*

*Recommended Products:*

*Customer: 12370.0 =====>>>> Recommended Products: [22423, '85123A', 47566, '85099B', 22720, 84879, 22960, 23298, 21212, 20725]*

*Customer: 12378.0 =====>>>> Recommended Products: [22423, '85123A', 47566, '85099B', 22720, 84879, 22960, 23298, 21212, 20725]*

*Customer: 12383.0 =====>>>> Recommended Products: [22423, '85123A', 47566, '85099B', 22720, 84879, 22960, 23298, 21212, 20725]*

*Customer: 12405.0 =====>>>> Recommended Products: [22423, '85123A', 47566, '85099B', 22720, 84879, 22960, 23298, 21212, 20725]*

*Customer: 12409.0 =====>>>> Recommended Products: [22423, '85123A', 47566, '85099B', 22720, 84879, 22960, 23298, 21212, 20725]*

*Cluster 5 ->*

*Recommended Products:*

*Customer: 12367.0 =====>>>> Recommended Products: [84879, 22086, 21034, 23084, 22423, 'POST', 22138, '85123A', 23355, 21790]*

*Customer: 12375.0 =====>>>> Recommended Products: [84879, 22086, 21034, 23084, 22423, 'POST', 22138, '85123A', 23355, 21790]*

*Customer: 12384.0 =====>>>> Recommended Products: [84879, 22086, 21034, 23084, 22423, 'POST', 22138, '85123A', 23355, 21790]*

*Customer: 12403.0 =====>>>> Recommended Products: [84879, 22086, 21034, 23084, 22423, 'POST', 22138, '85123A', 23355, 21790]*

*Customer: 12442.0 =====>>>> Recommended Products: [84879, 22086, 21034, 23084, 22423, 'POST', 22138, '85123A', 23355, 21790]*

*Cluster 6 ->*

*Recommended Products:*

*Customer: 12349.0 =====>>>> Recommended Products: ['85123A', 22423, 21034, 84879, '85099B', 22197, 22469, 22086, 22138, 20727]*

*Customer: 12352.0 =====>>>> Recommended Products: ['85123A', 22423, 21034, 84879, '85099B', 22197, 22469, 22086, 22138, 20727]*

*Customer: 12356.0 =====>>>> Recommended Products: ['85123A', 22423, 21034, 84879, '85099B', 22197, 22469, 22086, 22138, 20727]*

*Customer: 12371.0 =====>>>> Recommended Products: ['85123A', 22423, 21034, 84879, '85099B', 22197, 22469, 22086, 22138, 20727]*

*Customer: 12391.0 =====>>>> Recommended Products: ['85123A', 22423, 21034, 84879, '85099B', 22197, 22469, 22086, 22138, 20727]*

*Cluster 7 ->*

*Recommended Products:*

*Customer: 12399.0 =====>>>> Recommended Products: ['85123A', 84879, 22423, 47566, 22720, 22960, 21212, 22470, 21034, 22457]*

*Customer: 12446.0 =====>>>> Recommended Products: ['85123A', 84879, 22423, 47566, 22720, 22960, 21212, 22470, 21034, 22457]*

*Customer: 12534.0 =====>>>> Recommended Products: ['85123A', 84879, 22423, 47566, 22720, 22960, 21212, 22470, 21034, 22457]*

*Customer: 12545.0 =====>>>> Recommended Products: ['85123A', 84879, 22423, 47566, 22720, 22960, 21212, 22470, 21034, 22457]*

*Customer: 12550.0 =====>>>> Recommended Products: ['85123A', 84879, 22423, 47566, 22720, 22960, 21212, 22470, 21034, 22457]*

*Cluster 8 ->*

*Recommended Products:*

*Customer: 12348.0 =====>>>> Recommended Products: [47566, 'POST', 22423, '85123A', 84879, '85099B', 22138, 22086, 22699, 22720]*

*Customer: 12374.0 =====>>>> Recommended Products: [47566, 'POST', 22423, '85123A', 84879, '85099B', 22138, 22086, 22699, 22720]*

*Customer: 12379.0 =====>>>> Recommended Products: [47566, 'POST', 22423, '85123A', 84879, '85099B', 22138, 22086, 22699, 22720]*

*Customer: 12394.0 =====>>>> Recommended Products: [47566, 'POST', 22423, '85123A', 84879, '85099B', 22138, 22086, 22699, 22720]*

*Customer: 12410.0 =====>>>> Recommended Products: [47566, 'POST', 22423, '85123A', 84879, '85099B', 22138, 22086, 22699, 22720]*

*Cluster 9 ->*

*Recommended Products:*

*Customer: 12350.0 =====>>>> Recommended Products: ['85123A', 22423, 47566, 'POST', 22178, 22699, 22469, 22457, 22427, 22969]*

*Customer: 12355.0 =====>>>> Recommended Products: ['85123A', 22423, 47566, 'POST', 22178, 22699, 22469, 22457, 22427, 22969]*

*Customer: 12361.0 =====>>>> Recommended Products: ['85123A', 22423, 47566, 'POST', 22178, 22699, 22469, 22457, 22427, 22969]*

*Customer: 12373.0 =====>>>> Recommended Products: ['85123A', 22423, 47566, 'POST', 22178, 22699, 22469, 22457, 22427, 22969]*

*Customer: 12401.0 =====>>>> Recommended Products: ['85123A', 22423, 47566, 'POST', 22178, 22699, 22469, 22457, 22427, 22969]*