PCA Vs RFM Project

PART 1: Perform RFM Analysis

In [213…

```
import pandas as pd
import datetime as dt
import numpy as np
import matplotlib.pyplot as plt
```

In [214…

```
# Loading the dataset
flo = pd.read_csv("flo_data_20k.csv")
flo.head()
```

Out[214…

	master_id	order_channel	last_order_channel	first_order_date	last_order_date	last_order_date_online	last_order_date_offline	order_num_
0	cc294636-19f0-11eb-8d74-000d3a38a36f	Android App	Offline	2020-10-30	2021-02-26	2021-02-21	2021-02-26	
1	f431bd5a-ab7b-11e9-a2fc-000d3a38a36f	Android App	Mobile	2017-02-08	2021-02-16	2021-02-16	2020-01-10	
2	69b69676-1a40-11ea-941b-000d3a38a36f	Android App	Android App	2019-11-27	2020-11-27	2020-11-27	2019-12-01	
3	1854e56c-491f-11eb-806e-000d3a38a36f	Android App	Android App	2021-01-06	2021-01-17	2021-01-17	2021-01-06	
4	d6ea1074-f1f5-11e9-9346-000d3a38a36f	Desktop	Desktop	2019-08-03	2021-03-07	2021-03-07	2019-08-03	

In [215…
```
# Checking the information of the dataset
flo.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 19945 entries, 0 to 19944
Data columns (total 12 columns):
 #   Column                             Non-Null Count  Dtype  
---  ------                             --------------  -----  
 0   master_id                          19945 non-null  object 
 1   order_channel                      19945 non-null  object 
 2   last_order_channel                 19945 non-null  object 
 3   first_order_date                   19945 non-null  object 
 4   last_order_date                    19945 non-null  object 
 5   last_order_date_online             19945 non-null  object 
 6   last_order_date_offline            19945 non-null  object 
 7   order_num_total_ever_online        19945 non-null  float64
 8   order_num_total_ever_offline       19945 non-null  float64
 9   customer_value_total_ever_offline  19945 non-null  float64
 10  customer_value_total_ever_online   19945 non-null  float64
 11  interested_in_categories_12        19945 non-null  object 
dtypes: float64(4), object(8)
memory usage: 1.8+ MB
```

There is no null in this dataset

In [217…
```
# Explore the data: average, min, max, statistics number
flo.describe()
```

Out[217…

	order_num_total_ever_online	order_num_total_ever_offline	customer_value_total_ever_offline	customer_value_total_ever_online
count	19945.000000	19945.000000	19945.000000	19945.000000
mean	3.110855	1.913913	253.922597	497.321690
std	4.225647	2.062880	301.532853	832.601886
min	1.000000	1.000000	10.000000	12.990000
25%	1.000000	1.000000	99.990000	149.980000
50%	2.000000	1.000000	179.980000	286.460000
75%	4.000000	2.000000	319.970000	578.440000
max	200.000000	109.000000	18119.140000	45220.130000

In [218…
```
# Examining the variable types and changing the type of variables that express date to date
flo['first_order_date'] = pd.to_datetime(flo['first_order_date'])
```

```python
flo['last_order_date'] = pd.to_datetime(flo['last_order_date'])
```

In [219…

```python
#reference date
ref_date = flo["last_order_date"].max() + pd.Timedelta(days=1)
```

In [220…

```python
#Calculating the Recency: Total number of days since last order Using the most recent order date (either online or offline)
flo["Recency"] = (ref_date - flo["last_order_date"]).dt.days

# Calculating the Frequency (sum of total orders)
flo['Frequency'] = flo['order_num_total_ever_online'] + flo['order_num_total_ever_offline']

# Calculating the Monetary (Total spenderutes of each customer)
flo['Monetary'] = flo['customer_value_total_ever_online'] + flo['customer_value_total_ever_offline']
```

In [221…

```python
flo[["master_id", "Recency", "Frequency", "Monetary"]].head()
```

Out[221…

	master_id	Recency	Frequency	Monetary
0	cc294636-19f0-11eb-8d74-000d3a38a36f	94	5.0	939.37
1	f431bd5a-ab7b-11e9-a2fc-000d3a38a36f	104	21.0	2013.55
2	69b69676-1a40-11ea-941b-000d3a38a36f	185	5.0	585.32
3	1854e56c-491f-11eb-806e-000d3a38a36f	134	2.0	121.97
4	d6ea1074-f1f5-11e9-9346-000d3a38a36f	85	2.0	209.98

In [222…

```python
rfm = flo[["master_id", "Recency", "Frequency", "Monetary"]].copy()
rfm.set_index("master_id", inplace=True)
```

In [223…

```python
rfm["R_score"] = pd.qcut(rfm["Recency"], 5, labels=[5, 4, 3, 2, 1]).astype(int) #lowset recency get highest R score
rfm["F_score"] = pd.qcut(rfm["Frequency"].rank(method="first"), 5, labels=[1, 2, 3, 4, 5]).astype(int)
rfm["M_score"] = pd.qcut(rfm["Monetary"], 5, labels=[1, 2, 3, 4, 5]).astype(int)

rfm['RFM_Score']= rfm[["R_score", "F_score", "M_score"]].astype(int).sum(axis=1)
rfm["RFM_Segment"] = rfm["R_score"].astype(str) + rfm["F_score"].astype(str) + rfm["M_score"].astype(str)

rfm.head()
```

Out[223...

master_id	Recency	Frequency	Monetary	R_score	F_score	M_score	RFM_Score	RFM_Segment
cc294636-19f0-11eb-8d74-000d3a38a36f	94	5.0	939.37	3	4	4	11	344
f431bd5a-ab7b-11e9-a2fc-000d3a38a36f	104	21.0	2013.55	3	5	5	13	355
69b69676-1a40-11ea-941b-000d3a38a36f	185	5.0	585.32	2	4	3	9	243
1854e56c-491f-11eb-806e-000d3a38a36f	134	2.0	121.97	3	1	1	5	311
d6ea1074-f1f5-11e9-9346-000d3a38a36f	85	2.0	209.98	3	1	1	5	311

In [224...

```
# Classifying the customers based on RFM scores

rfm["ValueSegment"] = pd.cut(
    rfm["RFM_Score"],
    bins=[0, 6, 11, 15],
    labels=["Low Value", "Mid Value", "High Value"]
)

# Showing the results
rfm.head()
```

Out[224...

master_id	Recency	Frequency	Monetary	R_score	F_score	M_score	RFM_Score	RFM_Segment	ValueSegment
cc294636-19f0-11eb-8d74-000d3a38a36f	94	5.0	939.37	3	4	4	11	344	Mid Value
f431bd5a-ab7b-11e9-a2fc-000d3a38a36f	104	21.0	2013.55	3	5	5	13	355	High Value
69b69676-1a40-11ea-941b-000d3a38a36f	185	5.0	585.32	2	4	3	9	243	Mid Value
1854e56c-491f-11eb-806e-000d3a38a36f	134	2.0	121.97	3	1	1	5	311	Low Value
d6ea1074-f1f5-11e9-9346-000d3a38a36f	85	2.0	209.98	3	1	1	5	311	Low Value

In [225...

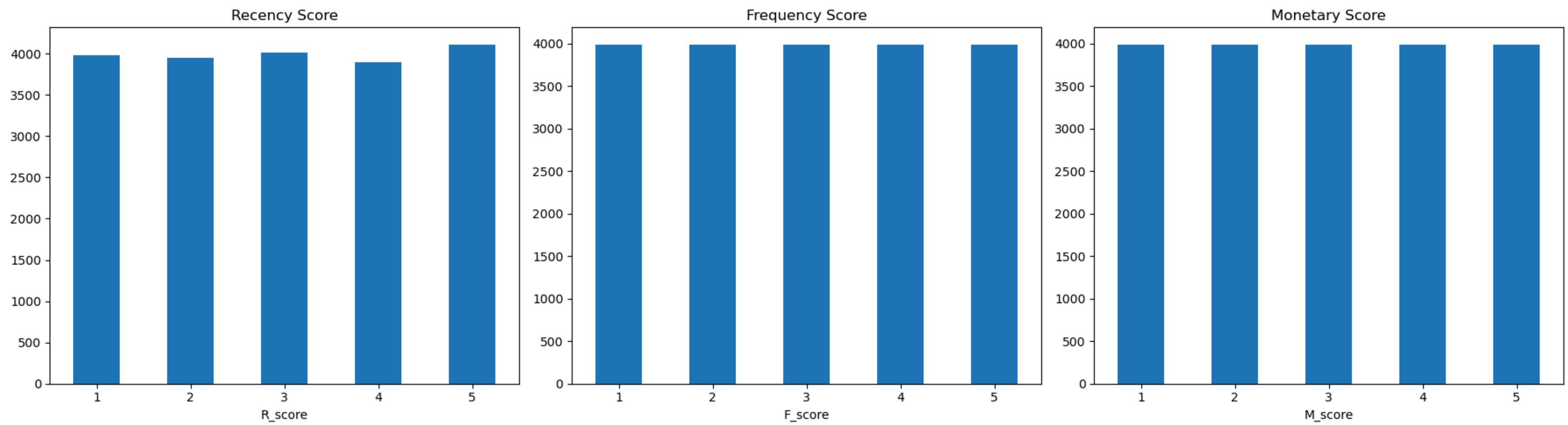
```
#Counting the customers in each segment
segment_counts = rfm["ValueSegment"].value_counts().sort_index()
segment_counts
```

Out[225…

```
ValueSegment
Low Value       4943
Mid Value      10043
High Value      4959
Name: count, dtype: int64
```

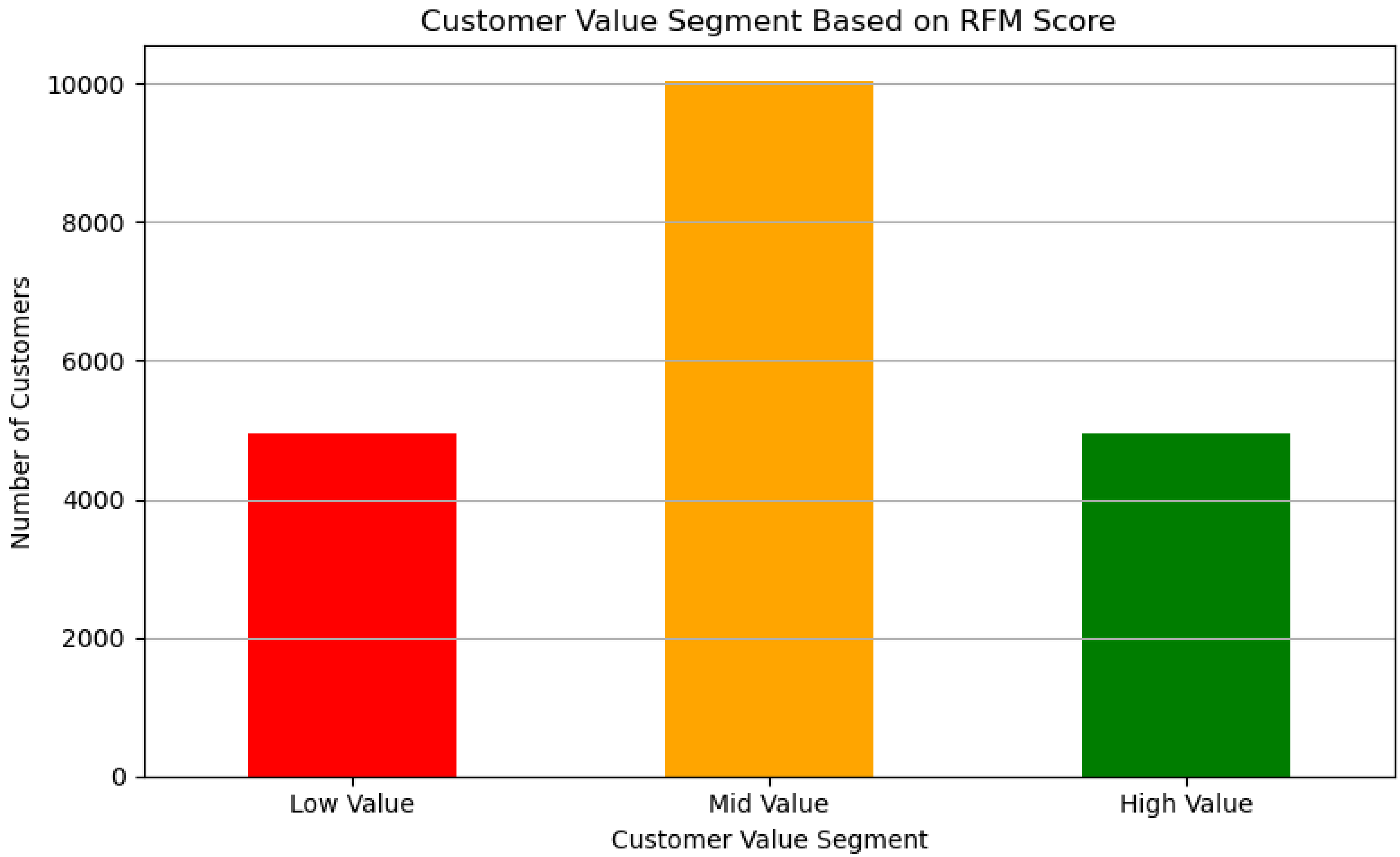
In [226…

```python
# RFM Score Bar Chart
fig, axs = plt.subplots(1, 3, figsize=(18,5))
rfm["R_score"].value_counts().sort_index().plot(kind="bar", ax=axs[0], title="Recency Score")
axs[0].set_xticklabels(axs[0].get_xticklabels(), rotation=0)
rfm["F_score"].value_counts().sort_index().plot(kind="bar", ax=axs[1], title="Frequency Score")
axs[1].set_xticklabels(axs[1].get_xticklabels(), rotation=0)
rfm["M_score"].value_counts().sort_index().plot(kind="bar", ax=axs[2], title="Monetary Score")
axs[2].set_xticklabels(axs[2].get_xticklabels(), rotation=0)
plt.tight_layout()
plt.show()
```



In [227…

```python
# Segment Bar Chart
plt.figure(figsize= (8, 5))
segment_counts.plot(kind="bar", color=["red", "orange", "green"])
plt.title("Customer Value Segment Based on RFM Score")
plt.xlabel("Customer Value Segment")
plt.ylabel("Number of Customers")
plt.xticks(rotation=0)
plt.grid(axis='y')
plt.tight_layout()
plt.show()
```



Step 1: RFM Analysis

1. Calculate Recency, Frequency, and Monetary Values

For each customer, we calculated the following metrics:

Recency: Number of days since the customer's last purchase (compared to a reference date). Frequency: Total number of transactions made by the customer (both online and offline). Monetary: Total amount of money spent by the customer (online and offline combined).

2. Assign RFM Scores Using Quintile-Based Scoring (1–5)

Each of the RFM metrics was scored based on quintiles:

Recency: Lower recency (more recent purchase) received a higher score (5 = most recent). Frequency & Monetary: Higher values received higher scores (5 = most frequent or highest spending). These scores were then concatenated into a 3-digit string called RFM_Segment (e.g., "355", "421").

3. Segment Customers Based on Total RFM Score

We summed the individual RFM scores to get a total RFM score per customer. Based on this total score, customers were segmented as follows:

High Value: Customers with high RFM scores (12–15). They buy frequently, recently, and spend more. Mid Value: Customers with moderate RFM scores (7–11). Good buyers but not top-tier. Low Value: Customers with low RFM scores (≤6). They are either inactive, low spenders, or infrequent buyers. Final customer distribution by segment:

High Value: 4,959 customers Mid Value: 10,043 customers Low Value: 4,943 customers

Part 2: Apply Principal Component Analysis (PCA)

In [230…

```
# import libraries
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
#to standardize (normalize) data so that: Each feature (e.g., Recency, Frequency, Monetary) has a mean of 0 And a standard deviati
from sklearn.decomposition import PCA
```

the dataset has already been loaded

In [232…

```
features = [
    "order_num_total_ever_online",
    "order_num_total_ever_offline",
    "customer_value_total_ever_online",
    "customer_value_total_ever_offline",
    "Recency",
]

X = flo[features].copy()
# Clusters capture interactions among selected features —not just the sum of RFM score.
```

In [233…

```
# Standardize
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

In [234…

```
# Apply PCA
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_scaled)

# Create DataFrame for plotting
pca_df = pd.DataFrame(X_pca, columns=["PC1", "PC2"])
```

In [235…

```
# KMeans Clustering
from sklearn.cluster import KMeans

kmeans = KMeans(n_clusters=3, random_state=42)
pca_df["Cluster"] = kmeans.fit_predict(X_pca)
```

In [236…

```
# To understand the meaning of cluster 0,1,2, calculate NEW RFM score:
# Join to have a dataframe with PCA
X_with_pca = X.join(pca_df)

# Calculating the Frequency
X_with_pca['Frequency'] = X_with_pca['order_num_total_ever_online'] + X_with_pca['order_num_total_ever_offline']

# Calculating the Monetary value
X_with_pca['Monetary'] = X_with_pca['customer_value_total_ever_online'] + X_with_pca['customer_value_total_ever_offline']

X_with_pca["R_score"] = pd.qcut(X_with_pca["Recency"], 5, labels=[5, 4, 3, 2, 1]).astype(int)
X_with_pca["F_score"] = pd.qcut(X_with_pca["Frequency"].rank(method="first"), 5, labels=[1, 2, 3, 4, 5]).astype(int)
X_with_pca["M_score"] = pd.qcut(X_with_pca["Monetary"], 5, labels=[1, 2, 3, 4, 5]).astype(int)

X_with_pca['RFM_Score']= X_with_pca[["R_score", "F_score", "M_score"]].astype(int).sum(axis=1)
X_with_pca
```

Out[236…

	order_num_total_ever_online	order_num_total_ever_offline	customer_value_total_ever_online	customer_value_total_ever_offline	Recency
0	4.0	1.0	799.38	139.99	94
1	19.0	2.0	1853.58	159.97	104
2	3.0	2.0	395.35	189.97	185
3	1.0	1.0	81.98	39.99	134
4	1.0	1.0	159.99	49.99	85
...	...	...	...	...	...
19940	1.0	2.0	111.98	289.98	330
19941	1.0	1.0	239.99	150.48	160
19942	2.0	1.0	492.96	139.98	7
19943	1.0	5.0	297.98	711.79	107
19944	1.0	1.0	221.98	39.99	359

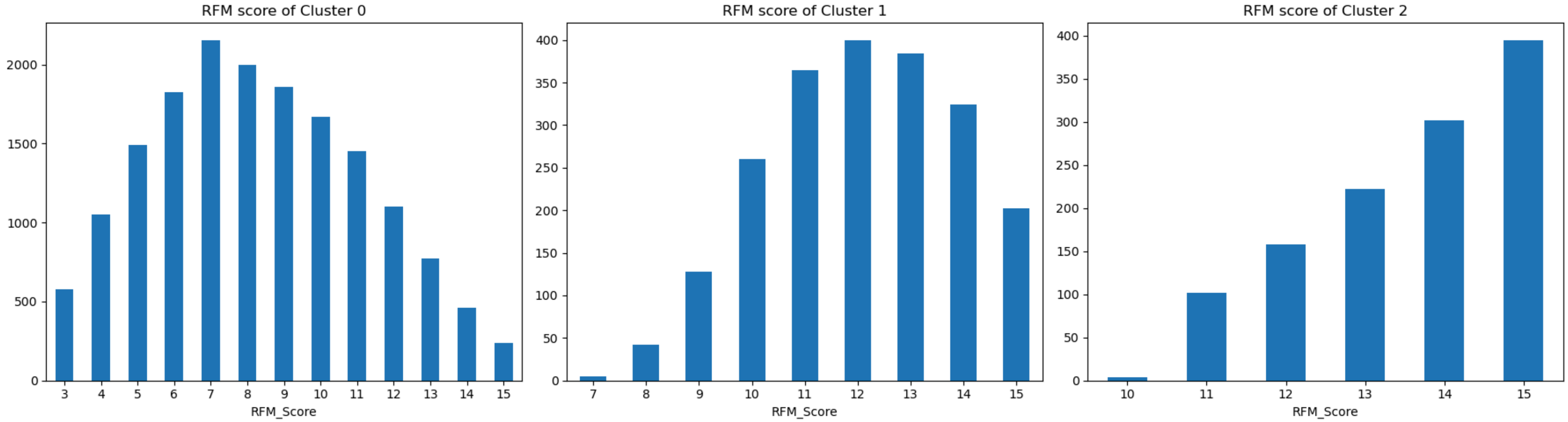
19945 rows × 14 columns

In [237…

```
# Select each cluster DataFrame
cluster_0_df = X_with_pca[X_with_pca["Cluster"] == 0]
cluster_1_df = X_with_pca[X_with_pca["Cluster"] == 1]
cluster_2_df = X_with_pca[X_with_pca["Cluster"] == 2]
```

In [238…

```
# Plot RFM distribution of each cluster
fig, axs = plt.subplots(1, 3, figsize=(18,5))
cluster_0_df["RFM_Score"].value_counts().sort_index().plot(kind="bar", ax=axs[0], title="RFM score of Cluster 0")
axs[0].set_xticklabels(axs[0].get_xticklabels(), rotation=0)
cluster_1_df["RFM_Score"].value_counts().sort_index().plot(kind="bar", ax=axs[1], title="RFM score of Cluster 1")
axs[1].set_xticklabels(axs[1].get_xticklabels(), rotation=0)
cluster_2_df["RFM_Score"].value_counts().sort_index().plot(kind="bar", ax=axs[2], title="RFM score of Cluster 2")
axs[2].set_xticklabels(axs[2].get_xticklabels(), rotation=0)
plt.tight_layout()
plt.show()
```



Based on RFM distribution, there are some insights of each cluster which is divided quite similar to the idea of RFM segmentation (high/mid/low):

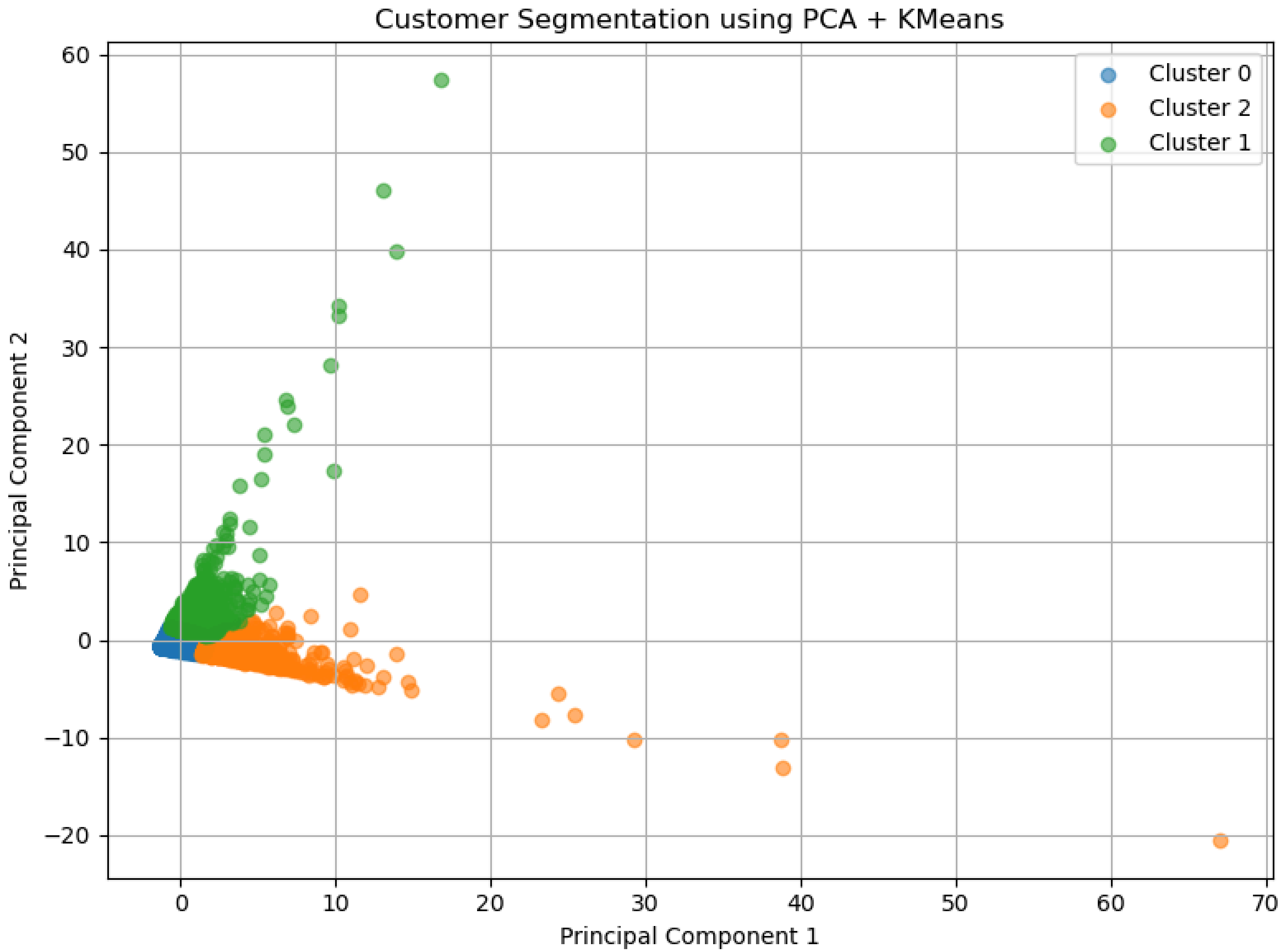
Cluster 0 has many mid-range scores, but may include a mix of behavior types (e.g., frequent but low monetary).

Cluster 1 spans roughly from 7 to 15, peaking between 11 and 13.It shows a concentration in the upper-mid to high range of RFM scores. Compared to Cluster 2 (which is elite/high-end only) and Cluster 0 (which has a broader lower-to-mid distribution), Cluster 1 is solidly high-performing but not top-tier.

Cluster 2, peaking sharply at RFM score 15, likely captures elite customers where all R, F, and M are high, not just in total score, but jointly optimal.

In [240…

```
# Visualize
plt.figure(figsize=(8, 6))

for label in pca_df["Cluster"].unique():
    subset = pca_df[pca_df["Cluster"] == label]
    plt.scatter(subset["PC1"], subset["PC2"], label=f"Cluster {label}", alpha=0.6)

plt.xlabel("Principal Component 1")
plt.ylabel("Principal Component 2")
plt.title("Customer Segmentation using PCA + KMeans")
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```



Part 3: Compare

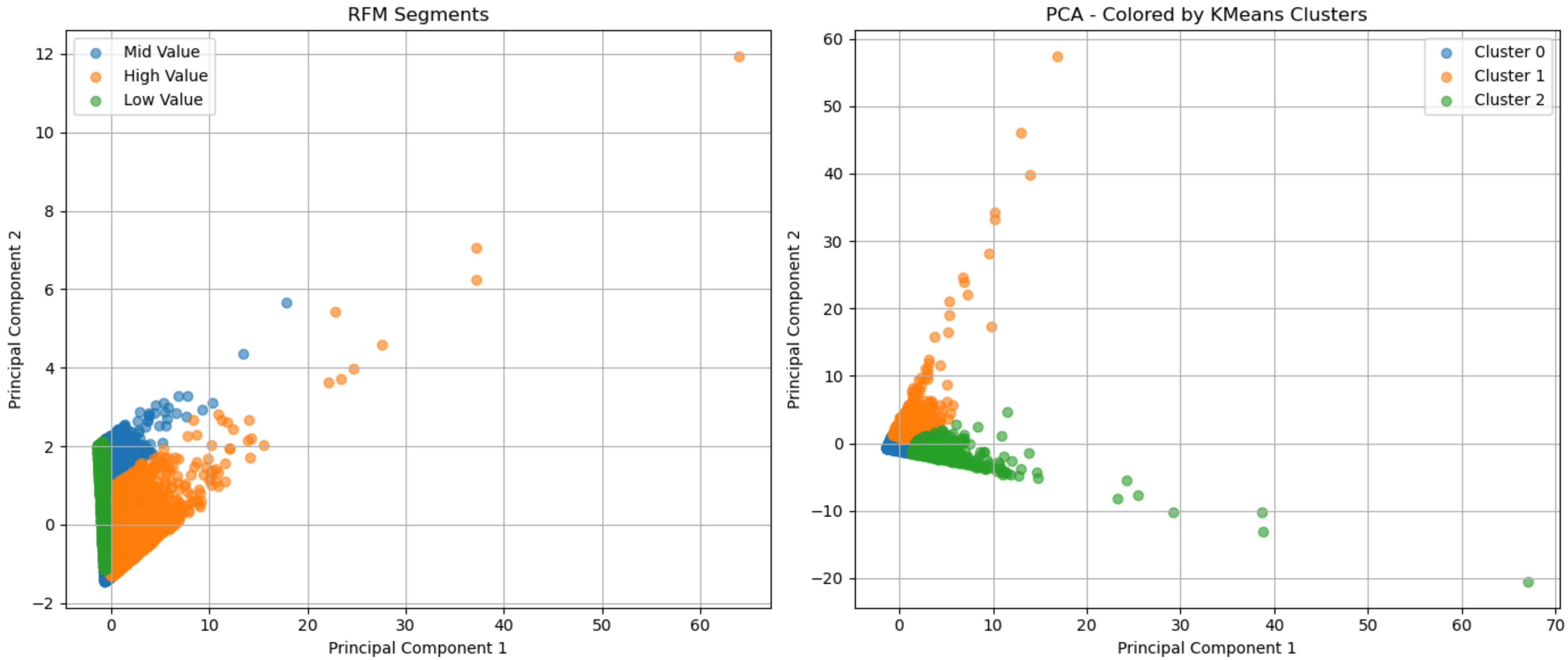
In [242…

```
# Create PCA visulization for RFM Method to easier comparision
rfm_features = flo[["Recency", "Frequency", "Monetary"]].copy()
scaler1 = StandardScaler()
rfm_scaled = scaler1.fit_transform(rfm_features)
pca1 = PCA(n_components=2)
rfm_pca = pca.fit_transform(rfm_scaled)

# Add to a new DataFrame
```

```
pca_df1 = pd.DataFrame(rfm_pca, columns=["PC1", "PC2"])
pca_df1["Segment"] = rfm["ValueSegment"].values
```

In [243…

```
# Visualize
fig, axes = plt.subplots(1, 2, figsize=(14, 6))

# Left: RFM Segments
for label in pca_df1["Segment"].dropna().unique():
    subset = pca_df1[pca_df1["Segment"] == label]
    axes[0].scatter(subset["PC1"], subset["PC2"], label=label, alpha=0.6)

axes[0].set_title("RFM Segments")
axes[0].set_xlabel("Principal Component 1")
axes[0].set_ylabel("Principal Component 2")
axes[0].legend()
axes[0].grid(True)

# Right: PCA Clusters from KMeans
for cluster in sorted(pca_df["Cluster"].dropna().unique()):
    subset = pca_df[pca_df["Cluster"] == cluster]
    axes[1].scatter(subset["PC1"], subset["PC2"], label=f"Cluster {cluster}", alpha=0.6)

axes[1].set_title("PCA - Colored by KMeans Clusters")
axes[1].set_xlabel("Principal Component 1")
axes[1].set_ylabel("Principal Component 2")
axes[1].legend()
axes[1].grid(True)

plt.tight_layout()
plt.show()
```



Two clustering approaches for customer segmentation: **RFM Segmentation** and **PCA with KMeans Clustering**.

RFM Segmentation

Strengths:

- Straightforward and intuitive because of using Recency, Frequency, and Monetary value, which are familiar metrics for most businesses.
- Offers clear insights for targeting campaigns (e.g., rewarding high-value or re-engaging lapsed customers).
- Easy to implement with minimal technical complexity.

Weaknesses:

- Thresholds for segmentation are often arbitrary and may lack nuance. These thresholds are set manually or based on intuition, not data. Thresholds don't account for industry norms, customer lifecycle stages, or seasonality.
- Does not adapt well to complex patterns in customer behavior. Lack of personalization: Customers with very different behaviors might be lumped into the same segment because of lack of personalization. Slight changes in behavior could move a customer across segments, despite little real change in value, leading to misclassification risk.
- May underperform with very large, multidimensional datasets.

PCA with KMeans Clustering

Strengths:

- Handles high-dimensional, complex customer data well.
- PCA (Principal Component Analysis) reduces noise and enhances interpretability for visualization.
- KMeans groups customers based on underlying patterns, which can uncover non-obvious insights.

Weaknesses:

- Choosing the right number of clusters (K) can be tricky and requires experimentation.
- PCA can oversimplify or obscure important details when reducing dimensions.
- The resulting clusters might not have immediately intuitive business meanings and may need additional interpretation.

Recommendation for Business Use Cases

For most businesses, especially those with **larger and more complex customer datasets**—**PCA with KMeans** is the better choice. It goes beyond static rules and learns from patterns in the data, enabling **more tailored and scalable customer strategies**.

However, RFM still has its place. It's great for quick wins or for companies just beginning to segment their audience. But for deeper personalization and growth-focused insights, PCA + KMeans offers a more powerful toolkit. Because its segment boundaries derived from multidimensional relationships, shape of distribution reflects natural density and grouping in data, and captures more nuance than just score sums.