In [1]:
```
# Step 1: Install Required Packages (if not already installed)
!pip install pandas matplotlib seaborn boto3 --quiet
!pip install s3fs
```

```
Requirement already satisfied: s3fs in /opt/conda/lib/python3.12/site-packages (202
4.10.0)
Requirement already satisfied: aiobotocore<3.0.0,>=2.5.4 in /opt/conda/lib/python3.1
2/site-packages (from s3fs) (2.21.1)
Requirement already satisfied: fsspec==2024.10.0.* in /opt/conda/lib/python3.12/site
-packages (from s3fs) (2024.10.0)
Requirement already satisfied: aiohttp!=4.0.0a0,!=4.0.0a1 in /opt/conda/lib/python3.
12/site-packages (from s3fs) (3.9.5)
Requirement already satisfied: aioitertools<1.0.0,>=0.5.1 in /opt/conda/lib/python3.
12/site-packages (from aiobotocore<3.0.0,>=2.5.4->s3fs) (0.12.0)
Requirement already satisfied: botocore<1.37.2,>=1.37.0 in /opt/conda/lib/python3.1
2/site-packages (from aiobotocore<3.0.0,>=2.5.4->s3fs) (1.37.1)
Requirement already satisfied: python-dateutil<3.0.0,>=2.1 in /opt/conda/lib/python
3.12/site-packages (from aiobotocore<3.0.0,>=2.5.4->s3fs) (2.9.0.post0)
Requirement already satisfied: jmespath<2.0.0,>=0.7.1 in /opt/conda/lib/python3.12/s
ite-packages (from aiobotocore<3.0.0,>=2.5.4->s3fs) (1.0.1)
Requirement already satisfied: multidict<7.0.0,>=6.0.0 in /opt/conda/lib/python3.12/
site-packages (from aiobotocore<3.0.0,>=2.5.4->s3fs) (6.2.0)
Requirement already satisfied: wrapt<2.0.0,>=1.10.10 in /opt/conda/lib/python3.12/si
te-packages (from aiobotocore<3.0.0,>=2.5.4->s3fs) (1.17.2)
Requirement already satisfied: aiosignal>=1.1.2 in /opt/conda/lib/python3.12/site-pa
ckages (from aiohttp!=4.0.0a0,!=4.0.0a1->s3fs) (1.3.2)
Requirement already satisfied: attrs>=17.3.0 in /opt/conda/lib/python3.12/site-packa
ges (from aiohttp!=4.0.0a0,!=4.0.0a1->s3fs) (23.2.0)
Requirement already satisfied: frozenlist>=1.1.1 in /opt/conda/lib/python3.12/site-p
ackages (from aiohttp!=4.0.0a0,!=4.0.0a1->s3fs) (1.5.0)
Requirement already satisfied: yarl<2.0,>=1.0 in /opt/conda/lib/python3.12/site-pack
ages (from aiohttp!=4.0.0a0,!=4.0.0a1->s3fs) (1.18.3)
Requirement already satisfied: urllib3!=2.2.0,<3,>=1.25.4 in /opt/conda/lib/python3.
12/site-packages (from botocore<1.37.2,>=1.37.0->aiobotocore<3.0.0,>=2.5.4->s3fs)
(2.3.0)
Requirement already satisfied: six>=1.5 in /opt/conda/lib/python3.12/site-packages
(from python-dateutil<3.0.0,>=2.1->aiobotocore<3.0.0,>=2.5.4->s3fs) (1.17.0)
Requirement already satisfied: idna>=2.0 in /opt/conda/lib/python3.12/site-packages
(from yarl<2.0,>=1.0->aiohttp!=4.0.0a0,!=4.0.0a1->s3fs) (3.10)
Requirement already satisfied: propcache>=0.2.0 in /opt/conda/lib/python3.12/site-pa
ckages (from yarl<2.0,>=1.0->aiohttp!=4.0.0a0,!=4.0.0a1->s3fs) (0.2.1)
```

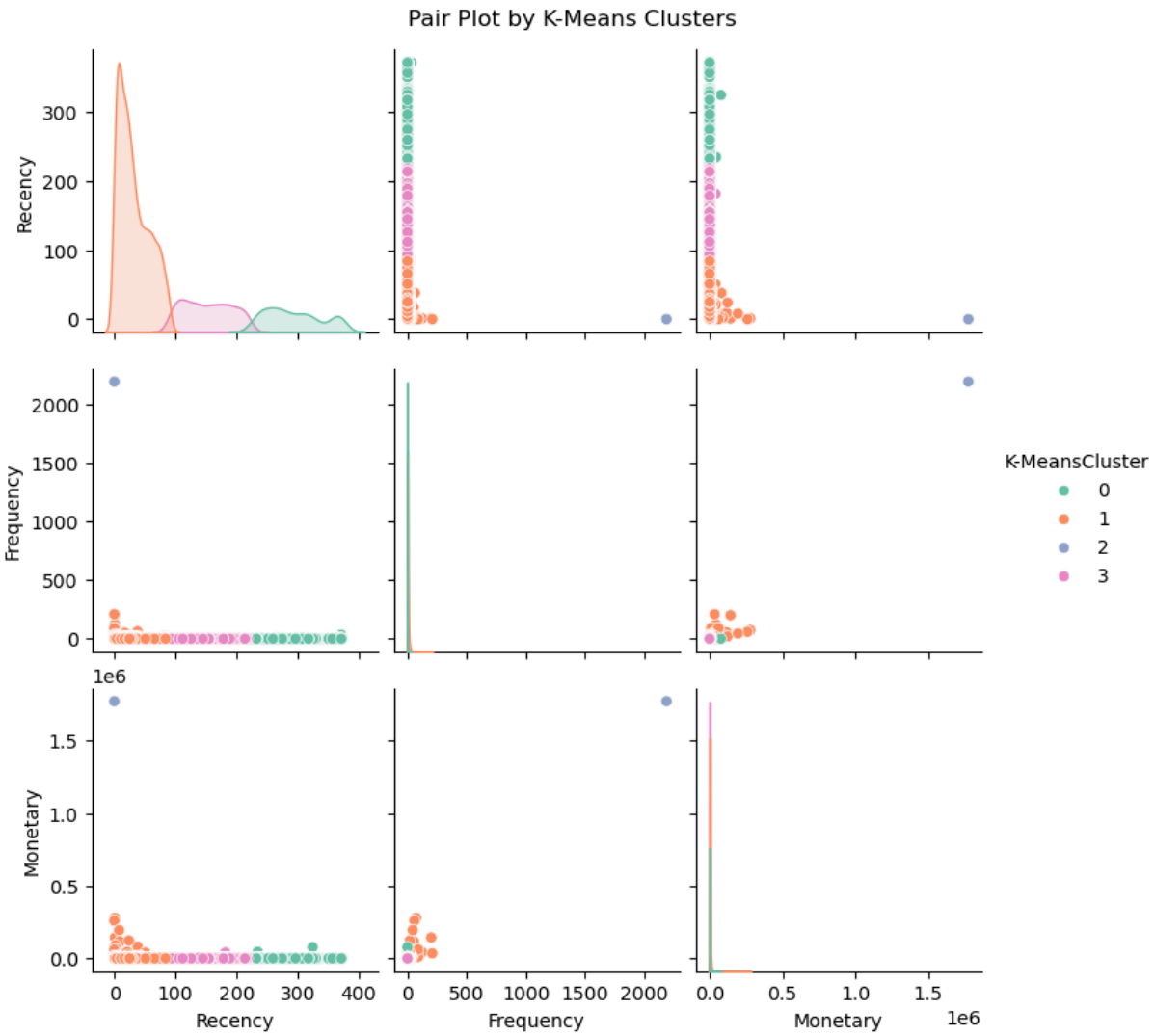
In [2]:
```
# Step 2: Import Libraries
import pandas as pd
import boto3
import io
import matplotlib.pyplot as plt
import seaborn as sns
from io import BytesIO
```

In [3]:
```
df = pd.read_csv("s3://cse-projectdataviz/Clustering.csv")
df.head()
```

Out[3]:

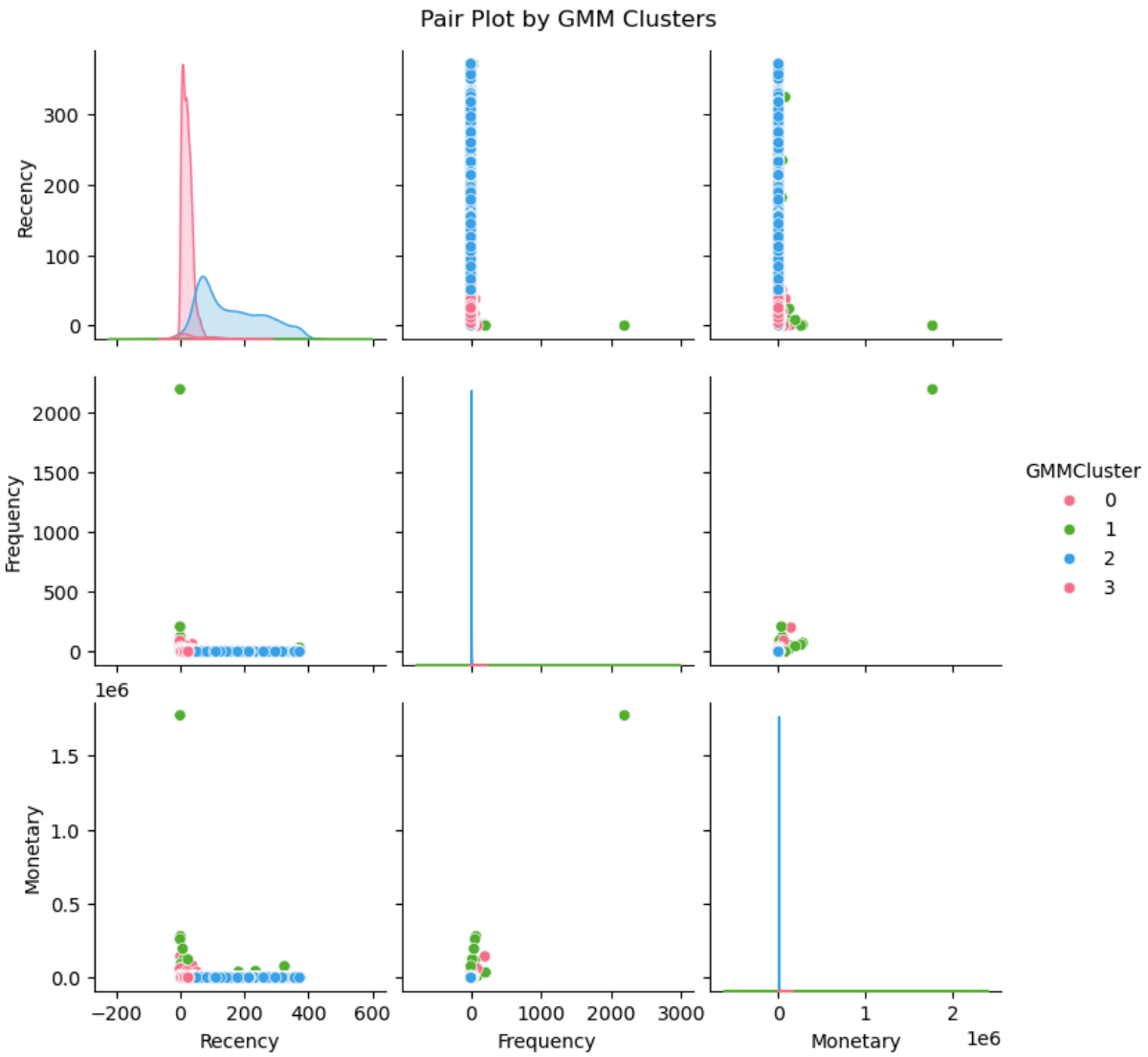
	Recency	Frequency	Monetary	K-MeansCluster	GMMCluster
0	23	9	6620.48	1	3
1	16	12	5162.91	1	3
2	89	4	1649.50	1	2
3	65	1	140.34	1	2
4	0	5	1857.46	1	3

In [4]:
```
# Drop GMM cluster column if exists
df_kmeans = df.drop(columns=['GMMCluster'])

# Pair Plot for K-Means
sns.pairplot(df_kmeans, hue="K-MeansCluster", palette="Set2")
plt.suptitle("Pair Plot by K-Means Clusters", y=1.02)
plt.show()

# Save and upload
buffer = BytesIO()
plt.savefig(buffer, format='png')
buffer.seek(0)

bucket_name = 'cse-projectdataviz'
key = 'pairplotKMeans.png'
s3 = boto3.client('s3')
s3.upload_fileobj(buffer, bucket_name, key)
plt.close()

print(f"Saved K-Means Pairplot to S3: s3://{bucket_name}/{key}")
```



Saved K-Means Pairplot to S3: s3://cse-projectdataviz/pairplotKMeans.png

In [5]:
```
# Drop K-Means cluster column if exists
df_gmm = df.drop(columns=['K-MeansCluster'])

# Pair Plot for GMM
sns.pairplot(df_gmm, hue="GMMCluster", palette="husl")
plt.suptitle("Pair Plot by GMM Clusters", y=1.02)
plt.show()

# Save and upload
buffer = BytesIO()
plt.savefig(buffer, format='png')
buffer.seek(0)

key = 'pairplotGMM.png'
s3.upload_fileobj(buffer, bucket_name, key)
plt.close()

print(f"Saved GMM Pairplot to S3: s3://{bucket_name}/{key}")
```

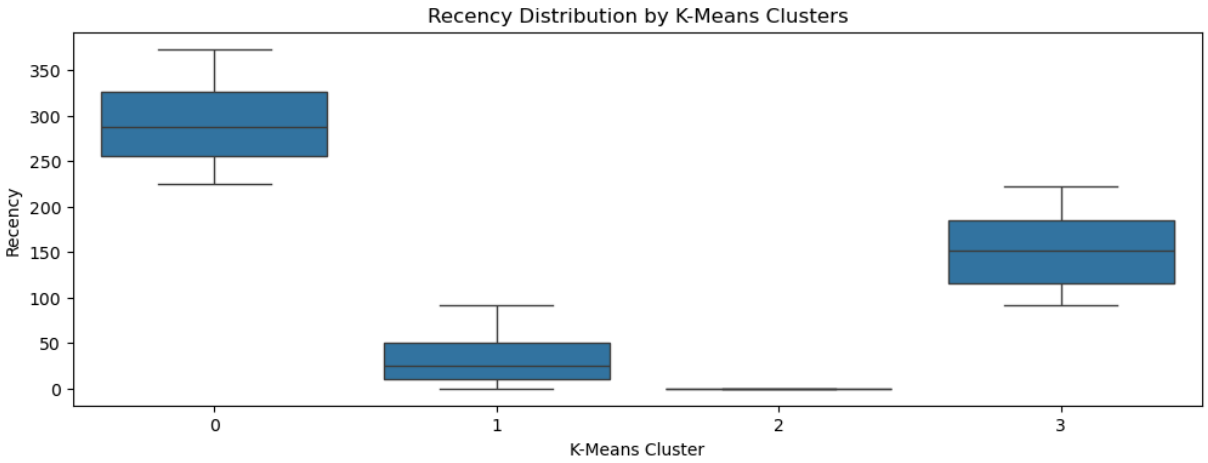


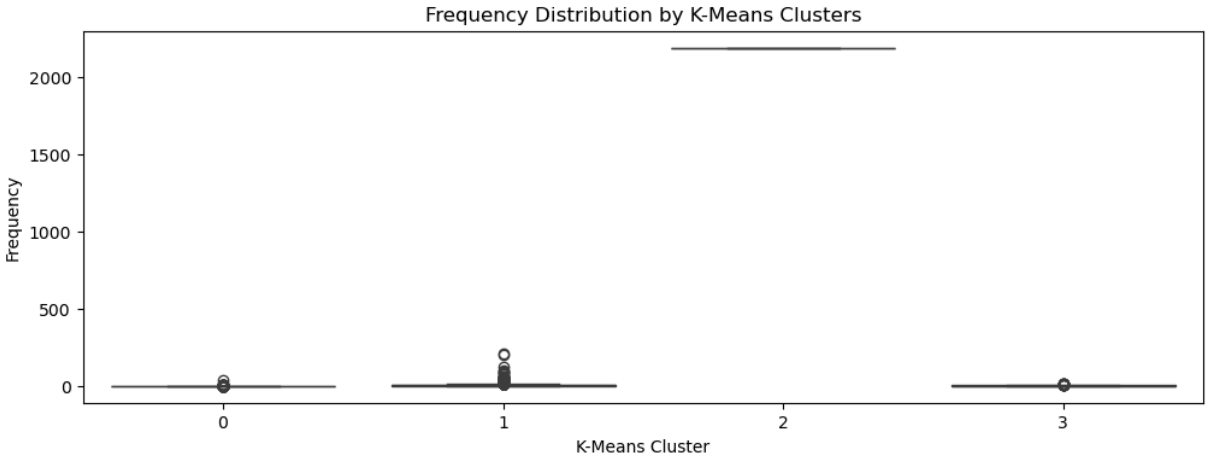

```
Saved GMM Pairplot to S3: s3://cse-projectdataviz/pairplotGMM.png
```

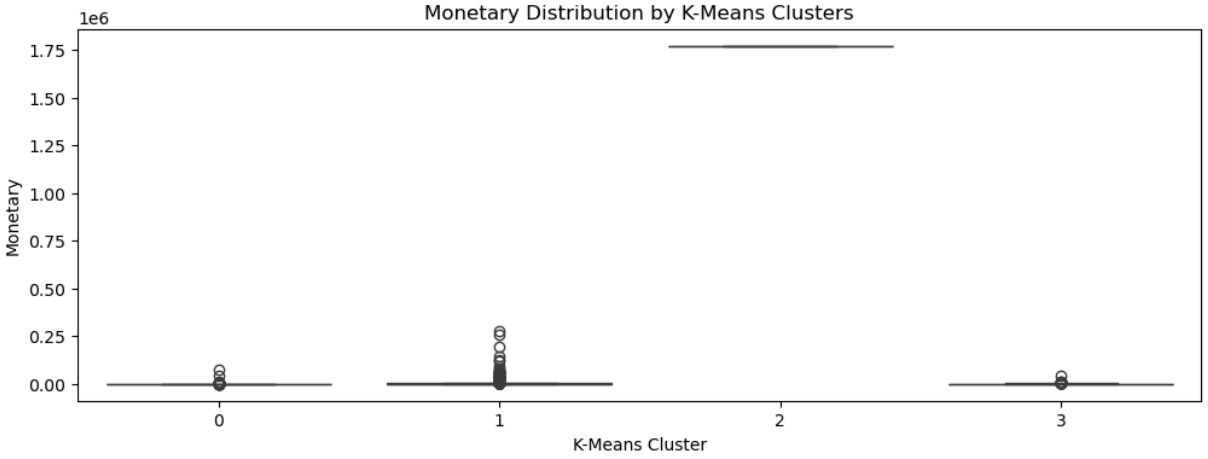
Boxplots for Each Cluster (KMeans & GMM) Goal: Show distribution of features per cluster.

In [6]:
```
features = ['Recency', 'Frequency', 'Monetary']

for col in features:
    plt.figure(figsize=(12, 4))
    sns.boxplot(x='K-MeansCluster', y=col, data=df)
    plt.title(f'{col} Distribution by K-Means Clusters')
    plt.xlabel('K-Means Cluster')
    plt.ylabel(col)
    plt.show()

# Save and upload
buffer = BytesIO()
plt.savefig(buffer, format='png')
buffer.seek(0)

key = 'boxplotkmeans.png'
s3.upload_fileobj(buffer, bucket_name, key)
plt.close()
```

```
print(f"Saved boxplot to S3: s3://{bucket_name}/{key}")
```



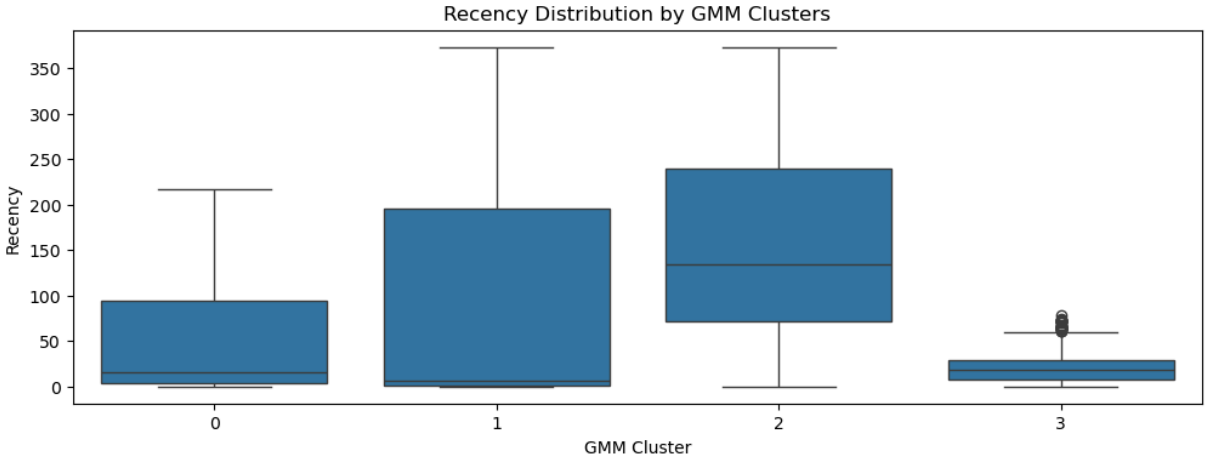


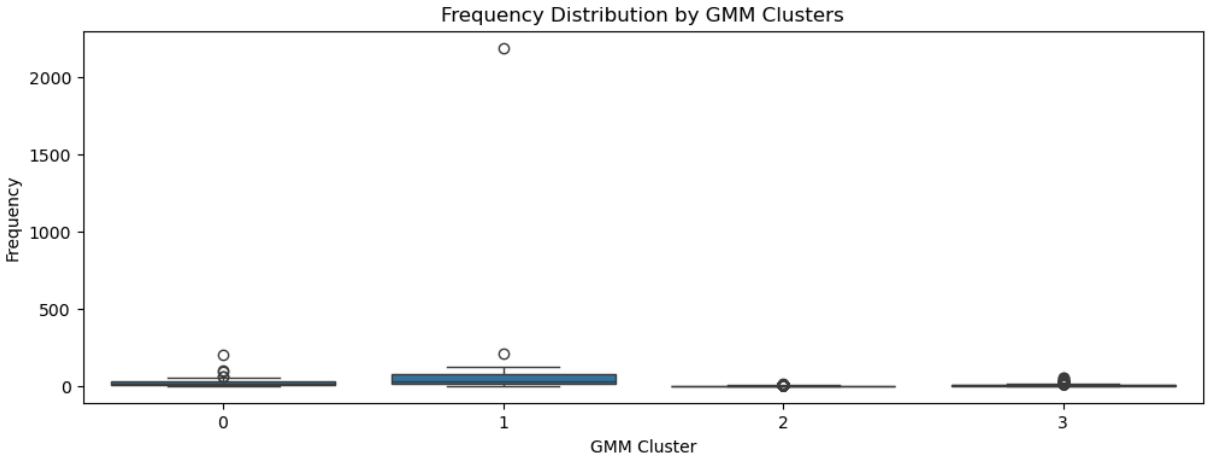


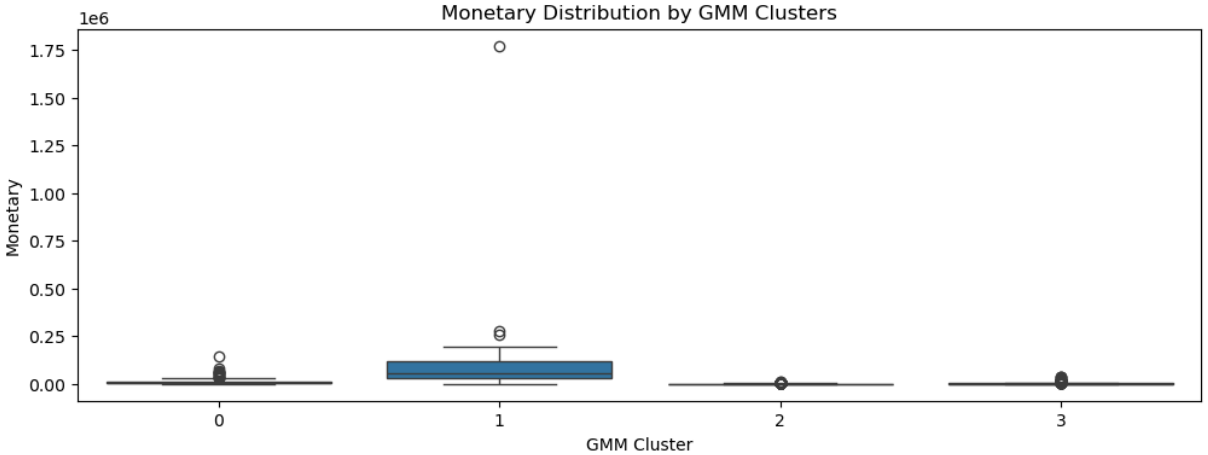

```
Saved boxplot to S3: s3://cse-projectdataviz/boxplotkmeans.png
```

In [7]:
```
features = ['Recency', 'Frequency', 'Monetary']

for col in features:
    plt.figure(figsize=(12, 4))
    sns.boxplot(x='GMMCluster', y=col, data=df)
    plt.title(f'{col} Distribution by GMM Clusters')
    plt.xlabel('GMM Cluster')
    plt.ylabel(col)
    plt.show()
```

```
# Save and upload
buffer = BytesIO()
plt.savefig(buffer, format='png')
buffer.seek(0)

key = 'boxplotGMM.png'
s3.upload_fileobj(buffer, bucket_name, key)
plt.close()

print(f"Saved boxplot to S3: s3://{bucket_name}/{key}")
```



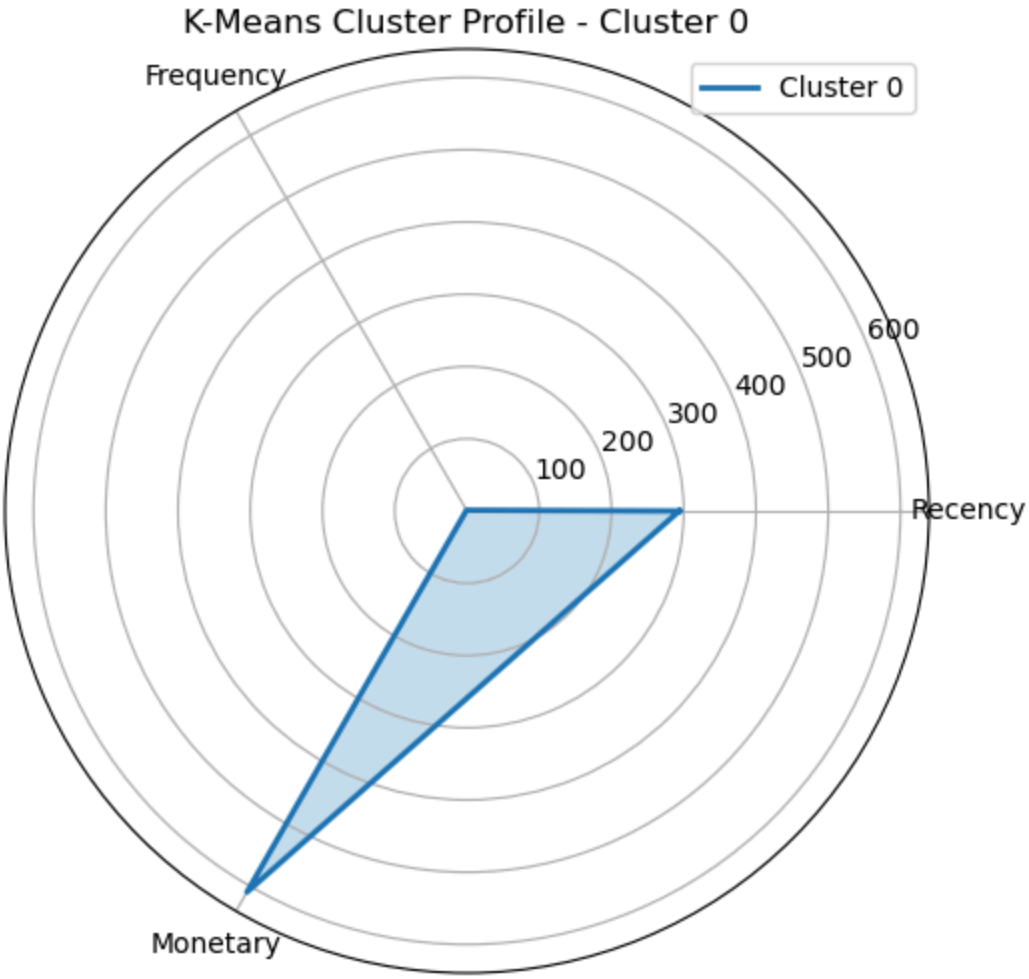


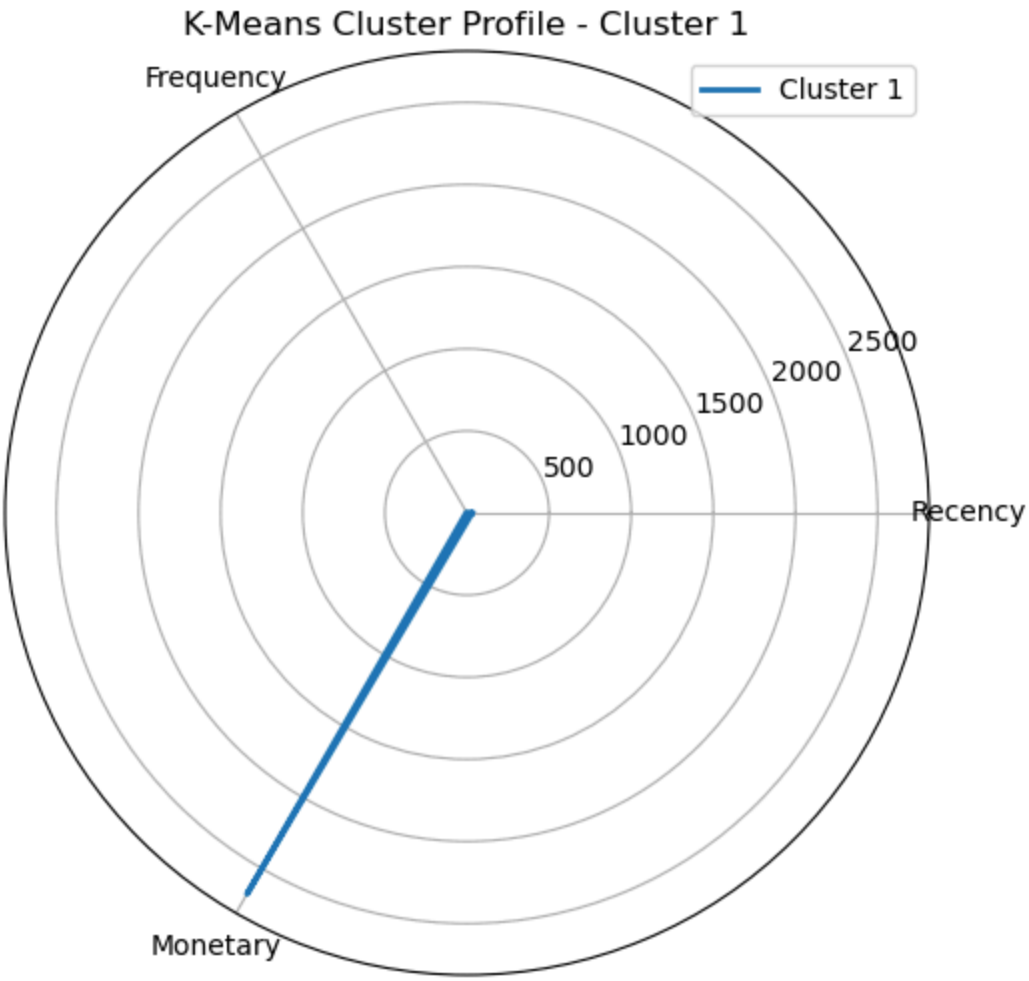



```
Saved boxplot to S3: s3://cse-projectdataviz/boxplotGMM.png
```

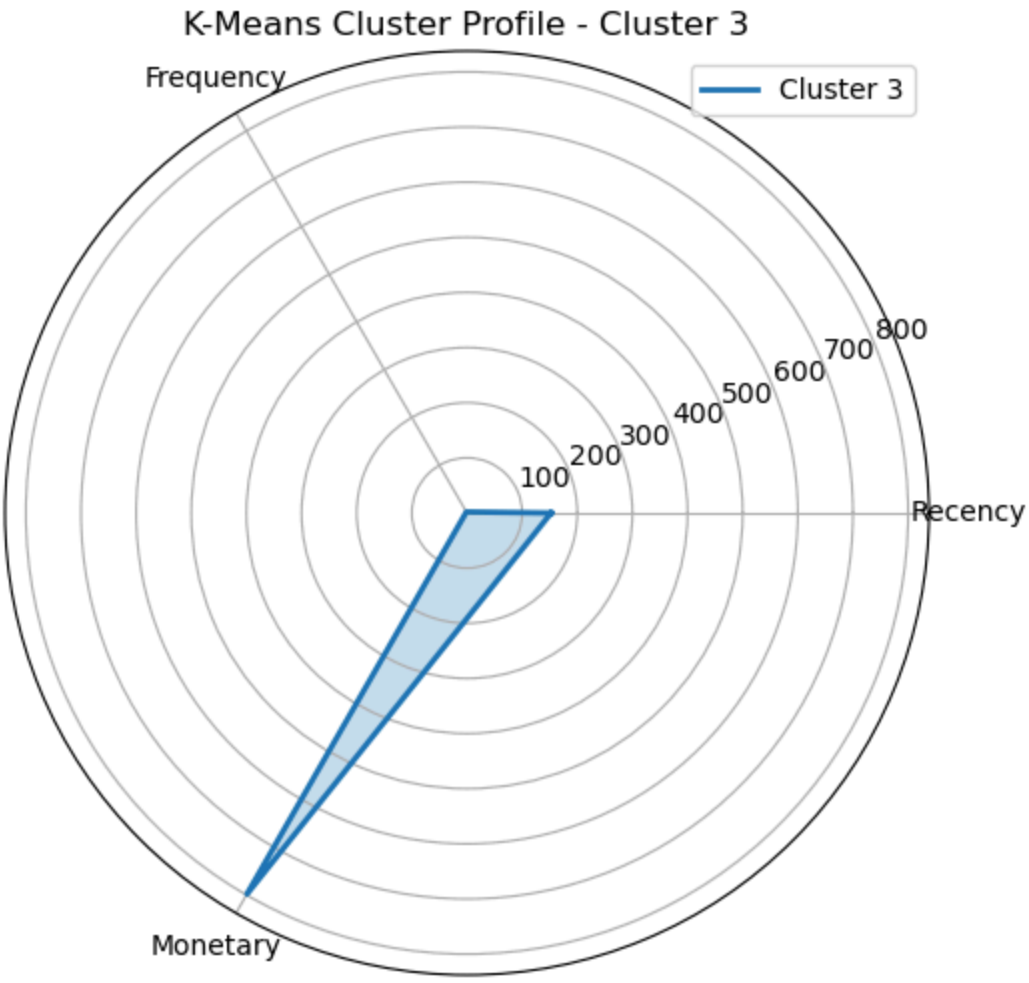
Centroid Radar Chart for Cluster Profiles Goal: Understand average feature values per cluster.

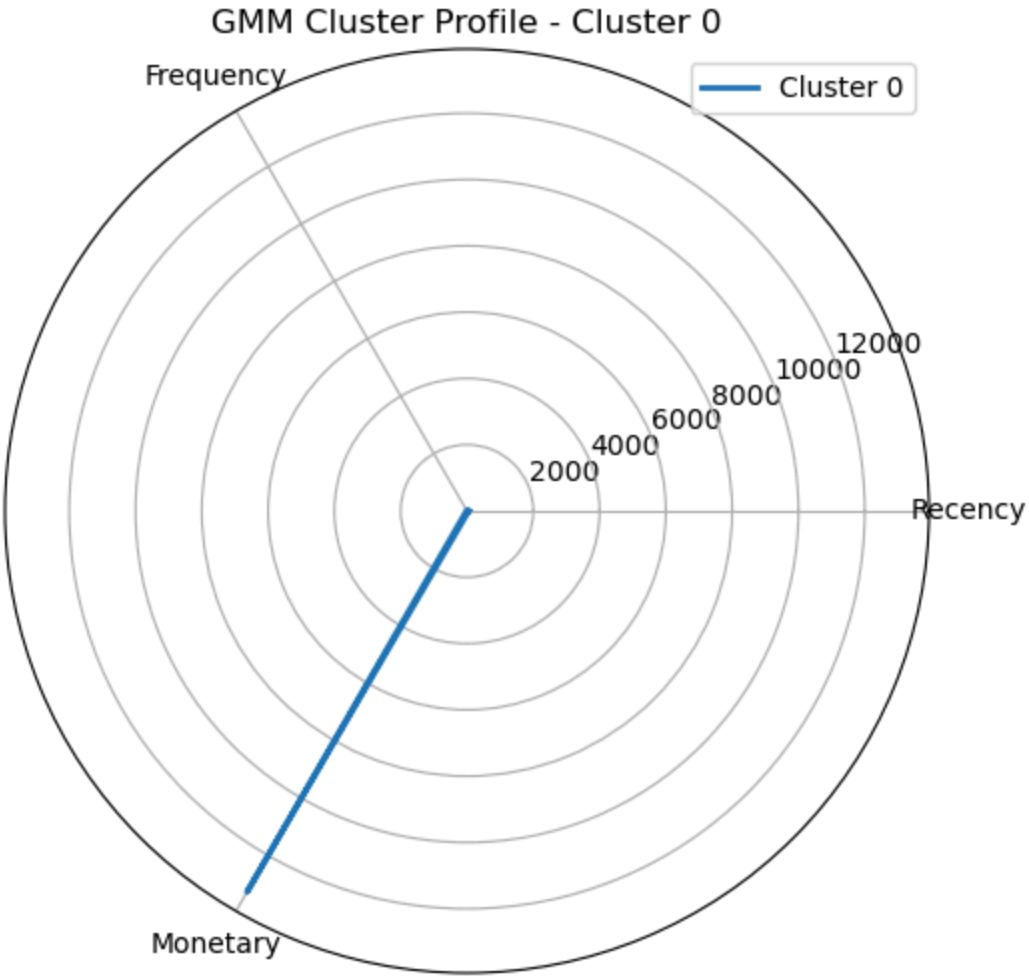
In [8]:
```
import numpy as np

def plot_radar_chart(cluster_data, title):
    categories = list(cluster_data.columns)
    N = len(categories)

    for idx, row in cluster_data.iterrows():
        values = row.tolist()
        values += values[:1]  # repeat first value to close the circle
        angles = [n / float(N) * 2 * np.pi for n in range(N)]
        angles += angles[:1]

        plt.figure(figsize=(6, 6))
        ax = plt.subplot(111, polar=True)
        plt.xticks(angles[:-1], categories)
        ax.plot(angles, values, linewidth=2, linestyle='solid', label=f'Cluster {id
        ax.fill(angles, values, alpha=0.25)
        plt.title(f'{title} - Cluster {idx}')
        plt.legend()
        plt.show()

# Group means
kmeans_profile = df.groupby('K-MeansCluster')[['Recency', 'Frequency', 'Monetary']]
gmm_profile = df.groupby('GMMCluster')[['Recency', 'Frequency', 'Monetary']].mean()

plot_radar_chart(kmeans_profile, "K-Means Cluster Profile")
plot_radar_chart(gmm_profile, "GMM Cluster Profile")

# Save and upload
buffer = BytesIO()
plt.savefig(buffer, format='png')
buffer.seek(0)

key = 'radar.png'
s3.upload_fileobj(buffer, bucket_name, key)
plt.close()

print(f"Saved Centroid Radar Chart  to S3: s3://{bucket_name}/{key}")
```

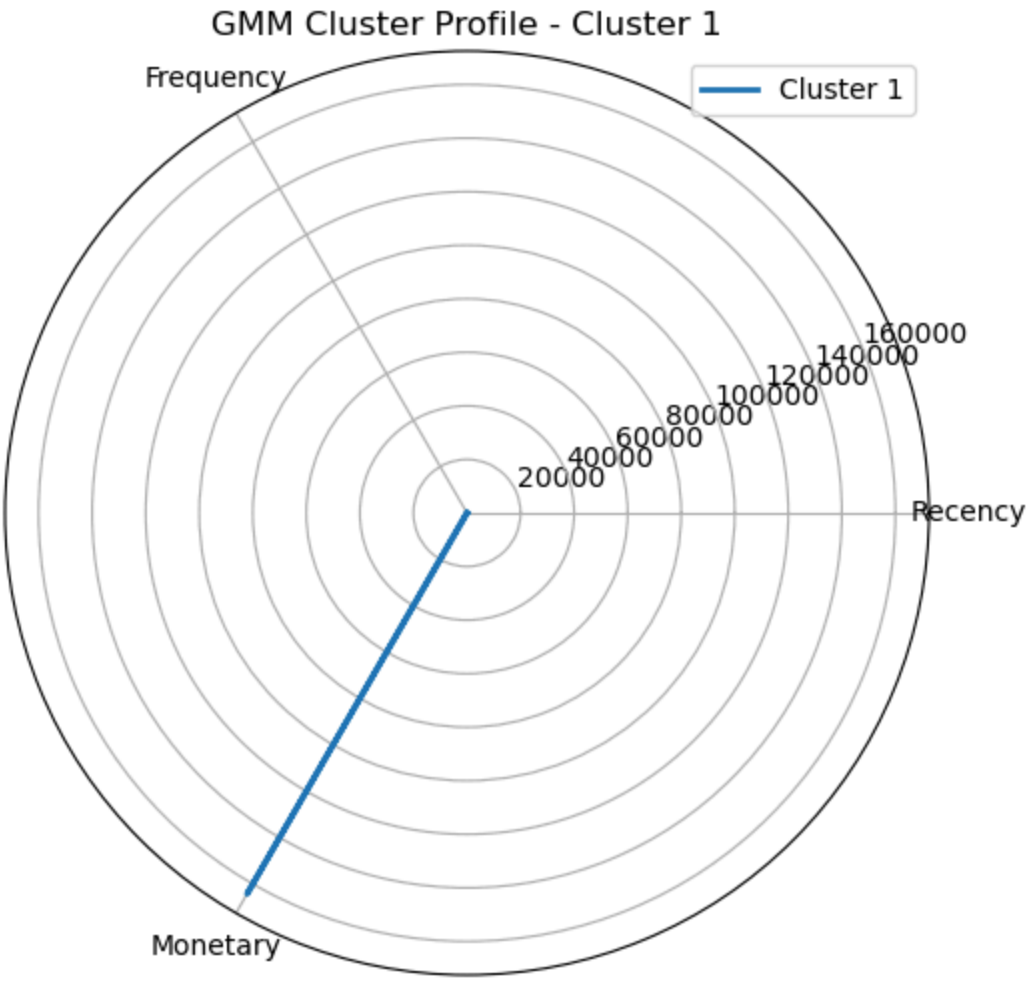

K-Means Cluster Profile - Cluster 0
Cluster 0
Frequency
Recency
Monetary
100
200
300
400
500
600

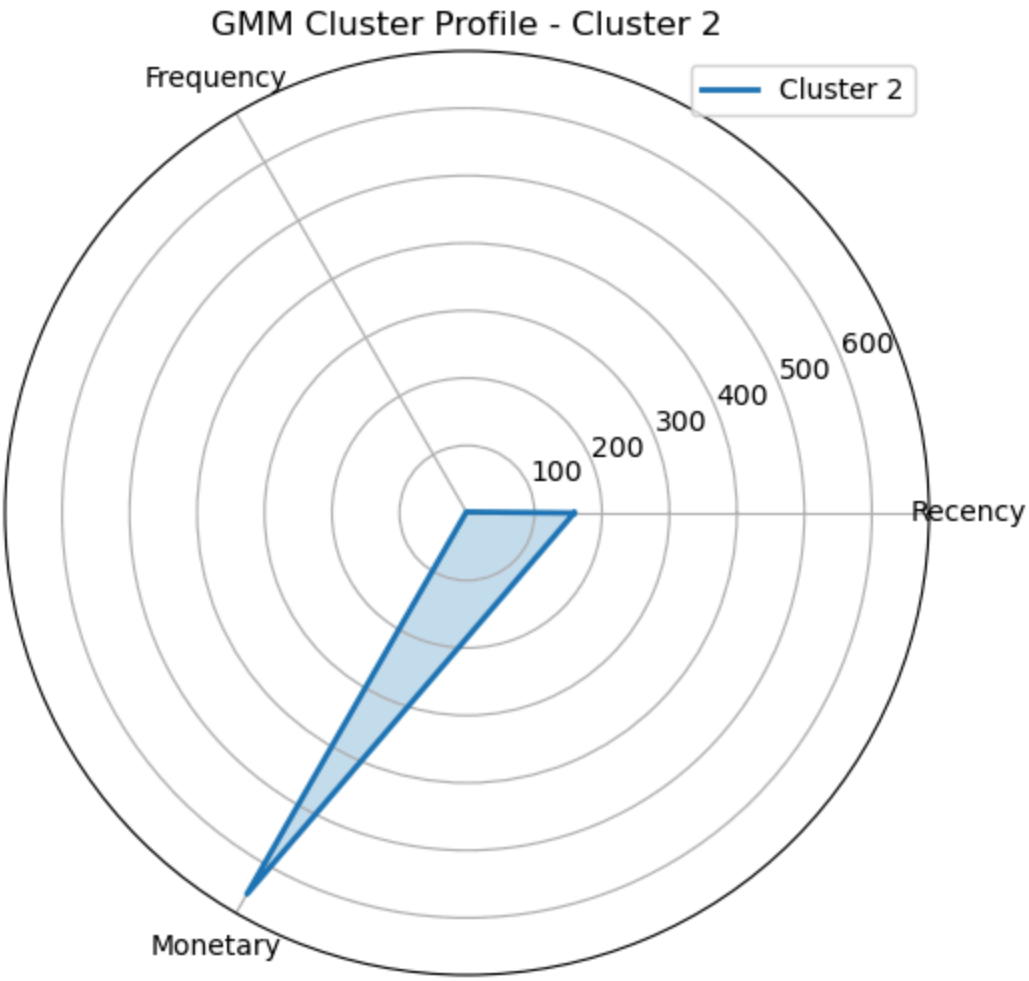

K-Means Cluster Profile - Cluster 1
Cluster 1
Frequency
Recency
Monetary
500
1000
1500
2000
2500

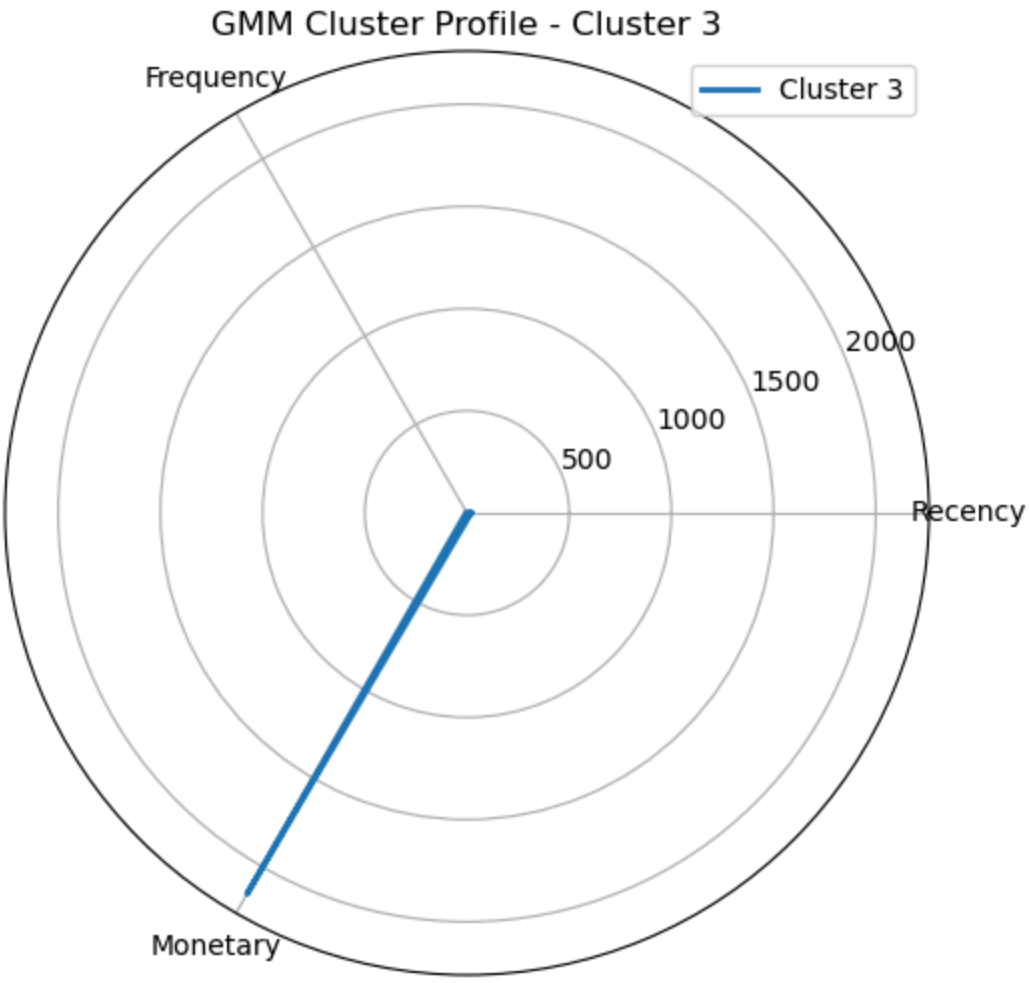

K-Means Cluster Profile - Cluster 2
1e6
Frequency
Cluster 2
0.25
0.50
0.75
1.00
1.25
1.50
1.75
Recency
Monetary


K-Means Cluster Profile - Cluster 3
Cluster 3
Frequency
Recency
Monetary
100
200
300
400
500
600
700
800


GMM Cluster Profile - Cluster 0
Cluster 0
Frequency
Recency
Monetary
2000
4000
6000
8000
10000
12000


GMM Cluster Profile - Cluster 1
Cluster 1
Frequency
Recency
Monetary
20000
40000
60000
80000
100000
120000
140000
160000

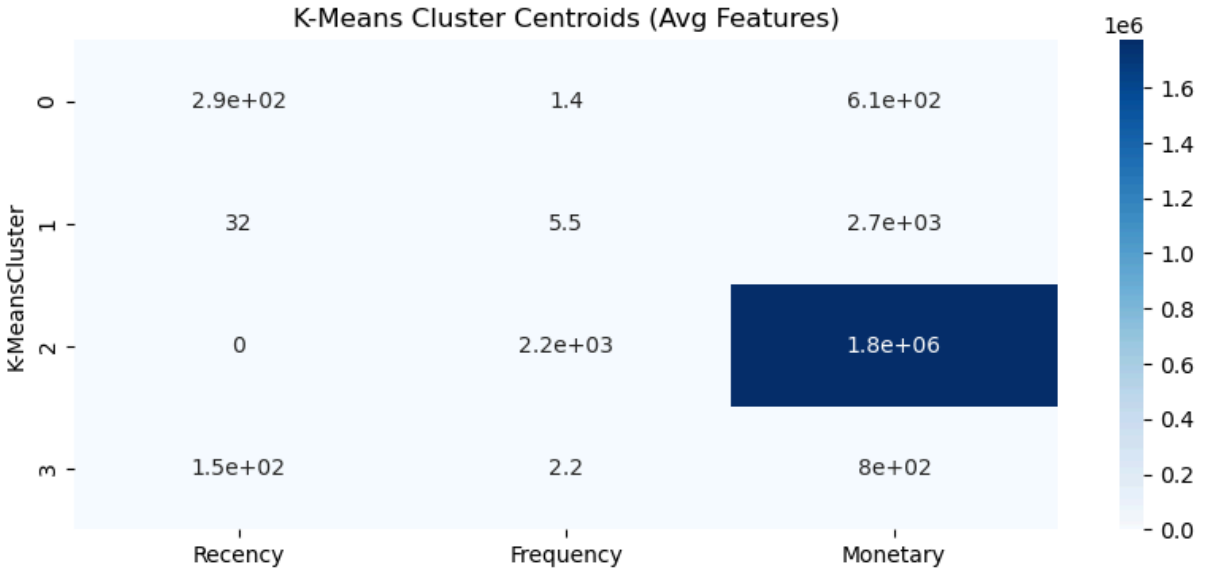

GMM Cluster Profile - Cluster 2
Cluster 2
Frequency
Recency
Monetary
100
200
300
400
500
600




```
Saved Centroid Radar Chart  to S3: s3://cse-projectdataviz/radar.png
```

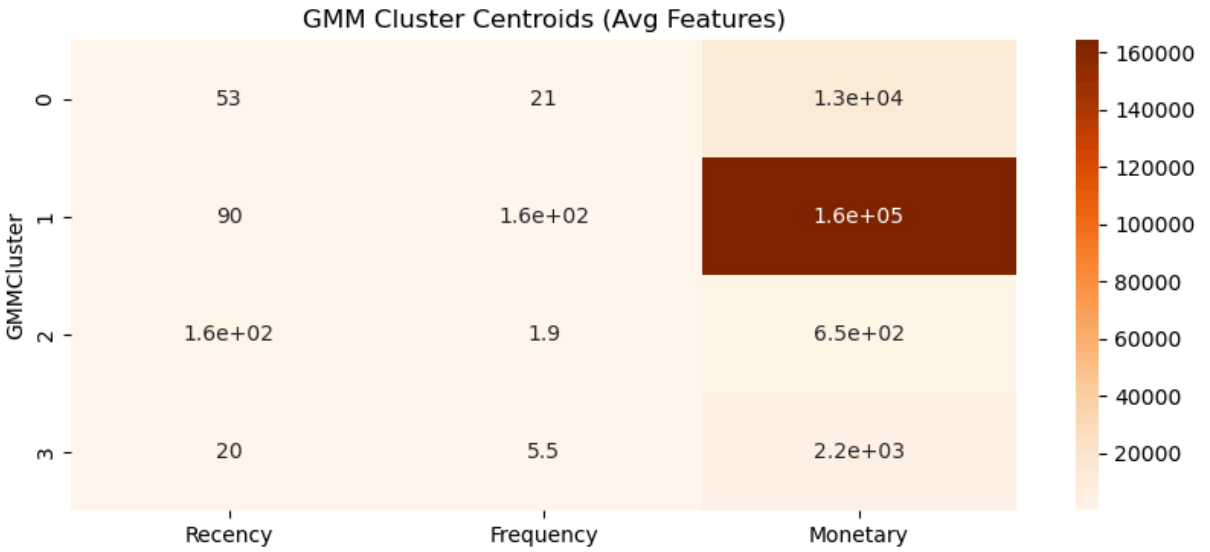
Heatmap of Cluster Centroids Goal: Compare how features differ across clusters.

In [9]:
```
plt.figure(figsize=(10, 4))
sns.heatmap(kmeans_profile, annot=True, cmap="Blues")
plt.title("K-Means Cluster Centroids (Avg Features)")
plt.show()

# Save and upload
buffer = BytesIO()
plt.savefig(buffer, format='png')
buffer.seek(0)

key = 'heatmapclustercentroidsKMeans.png'
s3.upload_fileobj(buffer, bucket_name, key)
plt.close()

print(f"Saved Heatmap of Cluster Centroids for K-Means to S3: s3://{bucket_name}/{k
```




```
Saved Heatmap of Cluster Centroids for K-Means to S3: s3://cse-projectdataviz/heatma
pclustercentroidsKMeans.png
```

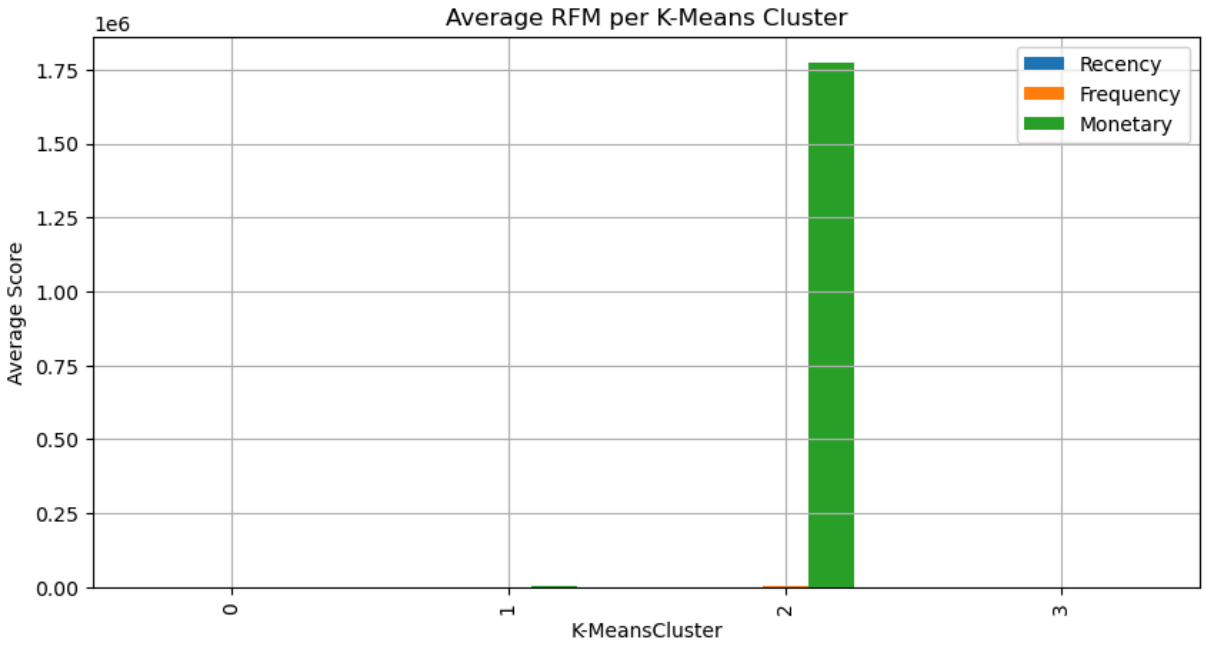
In [10]:
```
plt.figure(figsize=(10, 4))
sns.heatmap(gmm_profile, annot=True, cmap="Oranges")
plt.title("GMM Cluster Centroids (Avg Features)")
plt.show()

# Save and upload
buffer = BytesIO()
plt.savefig(buffer, format='png')
buffer.seek(0)

key = 'heatmapclustercentroidsGMM.png'
s3.upload_fileobj(buffer, bucket_name, key)
plt.close()

print(f"Saved Heatmap of Cluster Centroids for GMM to S3: s3://{bucket_name}/{key}"
```




```
Saved Heatmap of Cluster Centroids for GMM to S3: s3://cse-projectdataviz/heatmapclu
stercentroidsGMM.png
```

Average RFM Scores per Segment

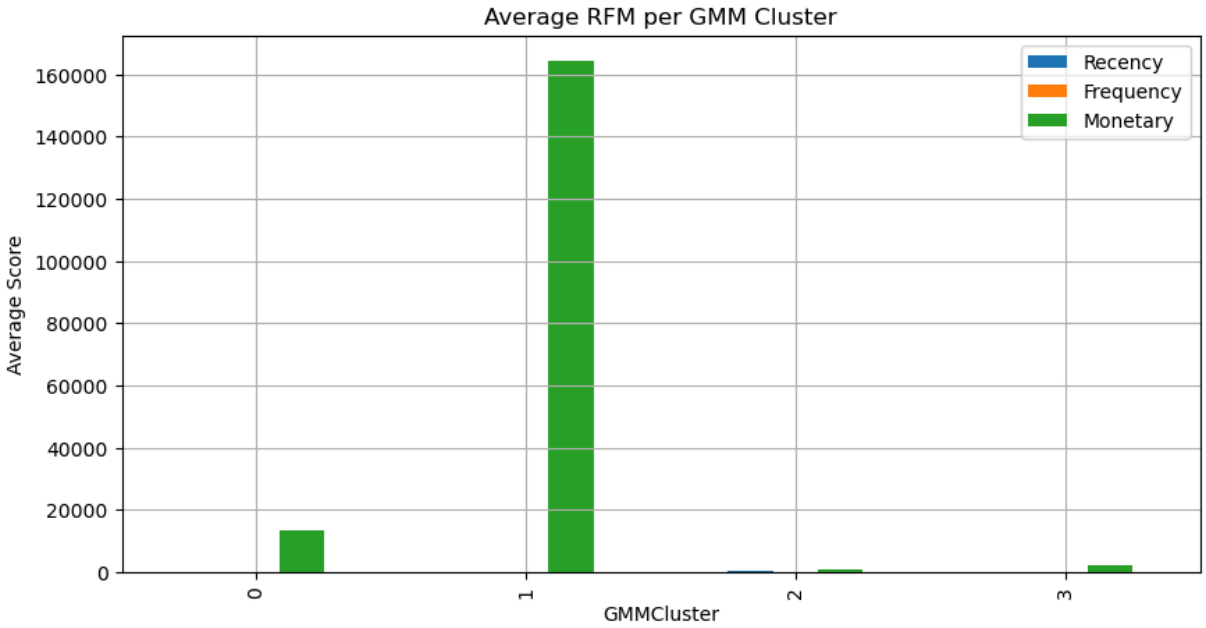
In [11]:
```
# Average RFM by K-Means
kmeans_avg_rfm = df.groupby("K-MeansCluster")[["Recency", "Frequency", "Monetary"]]
kmeans_avg_rfm.plot(kind='bar', figsize=(10, 5), title='Average RFM per K-Means Clu
plt.ylabel("Average Score")
plt.grid(True)
plt.show()

# Save and upload
buffer = BytesIO()
plt.savefig(buffer, format='png')
buffer.seek(0)

key = 'AvgRFMKMeans.png'
s3.upload_fileobj(buffer, bucket_name, key)
plt.close()

print(f"Saved Average RFM Scores of K-Means to S3: s3://{bucket_name}/{key}")
```




```
Saved Average RFM Scores of K-Means to S3: s3://cse-projectdataviz/AvgRFMKMeans.png
```

In [12]:
```
# Average RFM by GMM
gmm_avg_rfm = df.groupby("GMMCluster")[["Recency", "Frequency", "Monetary"]].mean()
gmm_avg_rfm.plot(kind='bar', figsize=(10, 5), title='Average RFM per GMM Cluster')
plt.ylabel("Average Score")
plt.grid(True)
plt.show()

# Save and upload
buffer = BytesIO()
plt.savefig(buffer, format='png')
buffer.seek(0)

key = 'AvgRFMGMM.png'
s3.upload_fileobj(buffer, bucket_name, key)
```

```
plt.close()

print(f"Saved Average RFM Scores of GMM to S3: s3://{bucket_name}/{key}")
```

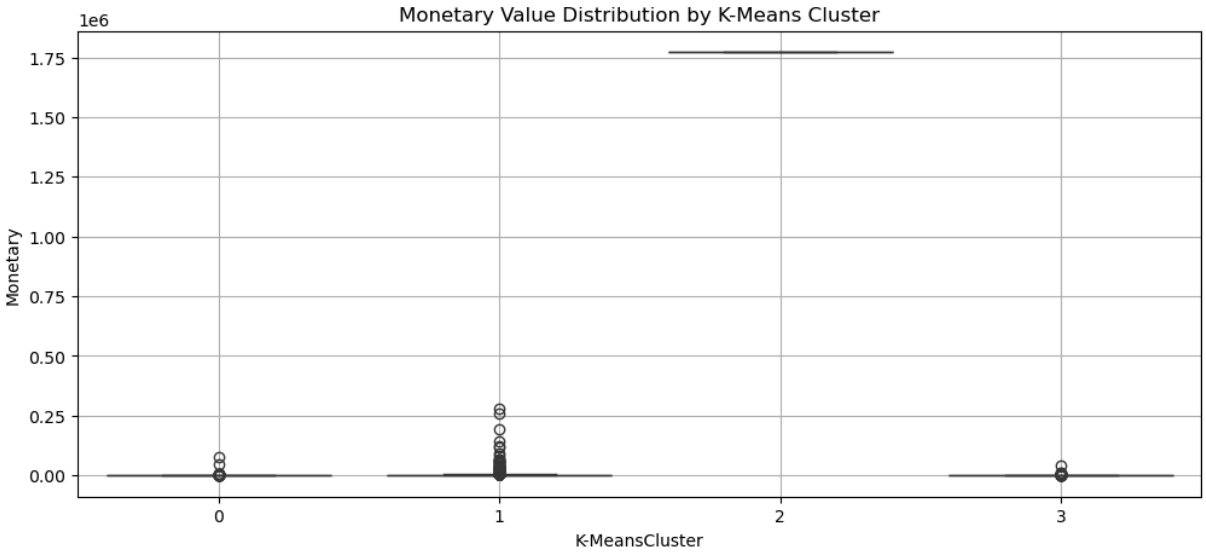



```
Saved Average RFM Scores of GMM to S3: s3://cse-projectdataviz/AvgRFMGMM.png
```

Distribution of Monetary Value by Segment

In [13]:
```
plt.figure(figsize=(12, 5))
sns.boxplot(data=df, x='K-MeansCluster', y='Monetary', palette='Blues')
plt.title("Monetary Value Distribution by K-Means Cluster")
plt.grid(True)
plt.show()

# Save and upload
buffer = BytesIO()
plt.savefig(buffer, format='png')
buffer.seek(0)

key = 'MonetaryKMeans.png'
s3.upload_fileobj(buffer, bucket_name, key)
plt.close()

print(f"Saved Distribution of Monetary Value by Segment of K-Means to S3: s3://{buc
```

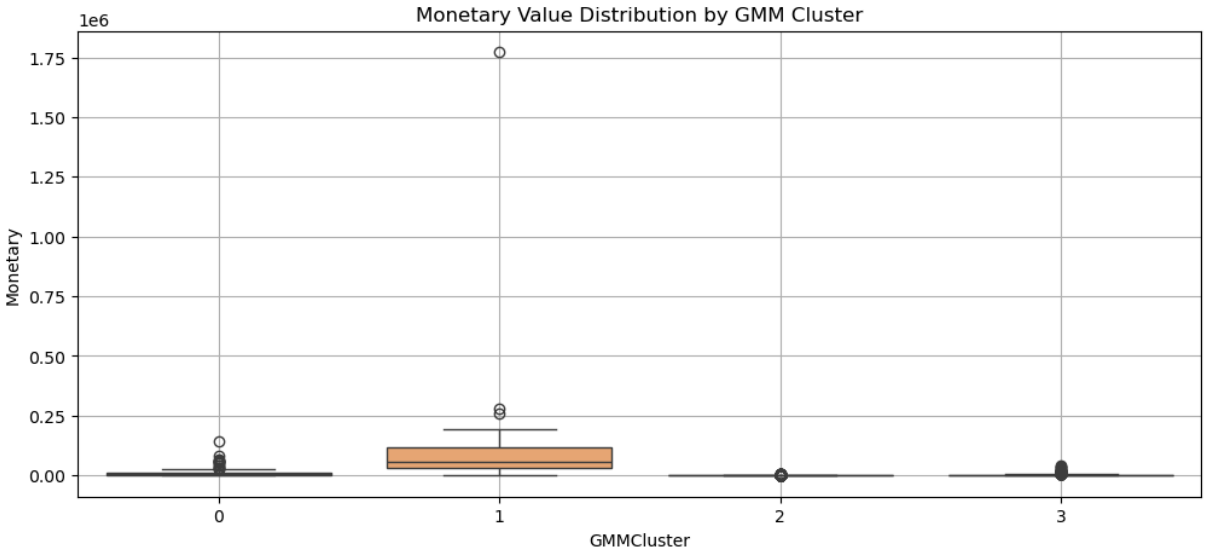
```
/tmp/ipykernel_116/361480608.py:2: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.1
4.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.boxplot(data=df, x='K-MeansCluster', y='Monetary', palette='Blues')
```




```
Saved Distribution of Monetary Value by Segment of K-Means to S3: s3://cse-projectda
taviz/MonetaryKMeans.png
```

In [14]:
```
plt.figure(figsize=(12, 5))
sns.boxplot(data=df, x='GMMCluster', y='Monetary', palette='Oranges')
plt.title("Monetary Value Distribution by GMM Cluster")
plt.grid(True)
plt.show()

# Save and upload
buffer = BytesIO()
plt.savefig(buffer, format='png')
buffer.seek(0)

key = 'MonetaryGMM.png'
s3.upload_fileobj(buffer, bucket_name, key)
plt.close()

print(f"Saved Distribution of Monetary Value by Segment of GMM to S3: s3://{bucket_
```

```
/tmp/ipykernel_116/3308720253.py:2: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.1
4.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.boxplot(data=df, x='GMMCluster', y='Monetary', palette='Oranges')
```

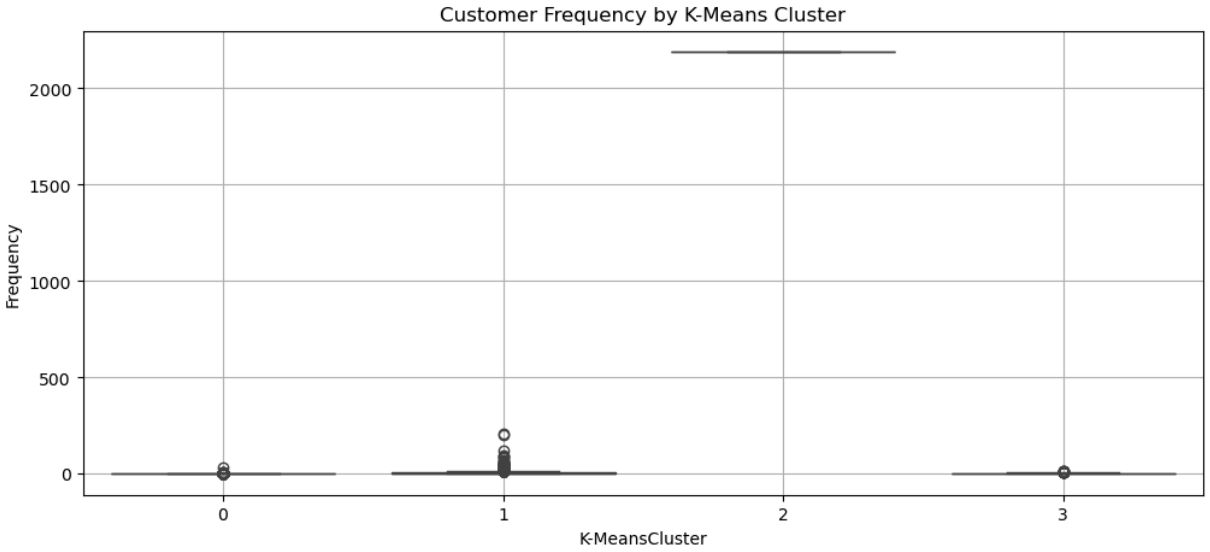



```
Saved Distribution of Monetary Value by Segment of GMM to S3: s3://cse-projectdatavi
z/MonetaryGMM.png
```

Most Frequent Customers per Segment

In [15]:
```
plt.figure(figsize=(12, 5))
sns.boxplot(data=df, x='K-MeansCluster', y='Frequency', palette='Purples')
plt.title("Customer Frequency by K-Means Cluster")
plt.grid(True)
plt.show()

# Save and upload
buffer = BytesIO()
plt.savefig(buffer, format='png')
buffer.seek(0)

key = 'FrequentCustomersKMeans.png'
s3.upload_fileobj(buffer, bucket_name, key)
plt.close()

print(f"Saved Most Frequent Customers per Segment of K-Means to S3: s3://{bucket_na
```

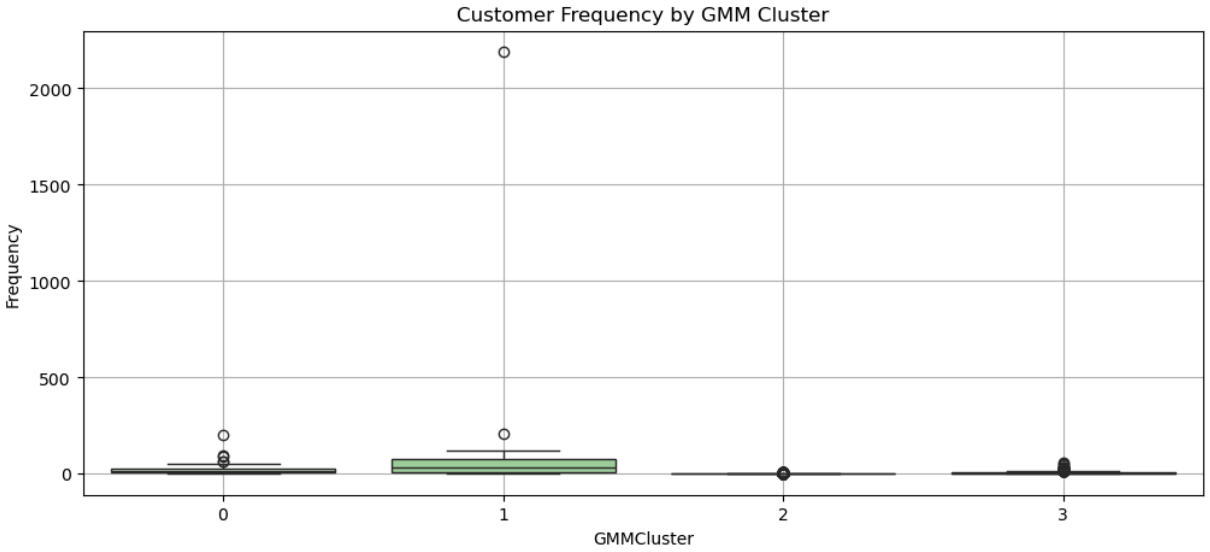
```
/tmp/ipykernel_116/1012618580.py:2: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.1
4.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.boxplot(data=df, x='K-MeansCluster', y='Frequency', palette='Purples')
```




```
Saved Most Frequent Customers per Segment of K-Means to S3: s3://cse-projectdataviz/
FrequentCustomersKMeans.png
```

In [16]:
```
plt.figure(figsize=(12, 5))
sns.boxplot(data=df, x='GMMCluster', y='Frequency', palette='Greens')
plt.title("Customer Frequency by GMM Cluster")
plt.grid(True)
plt.show()

# Save and upload
buffer = BytesIO()
plt.savefig(buffer, format='png')
buffer.seek(0)

key = 'FrequentCustomersGMM.png'
s3.upload_fileobj(buffer, bucket_name, key)
plt.close()

print(f"Saved Most Frequent Customers per Segment of GMM to S3: s3://{bucket_name}/
```

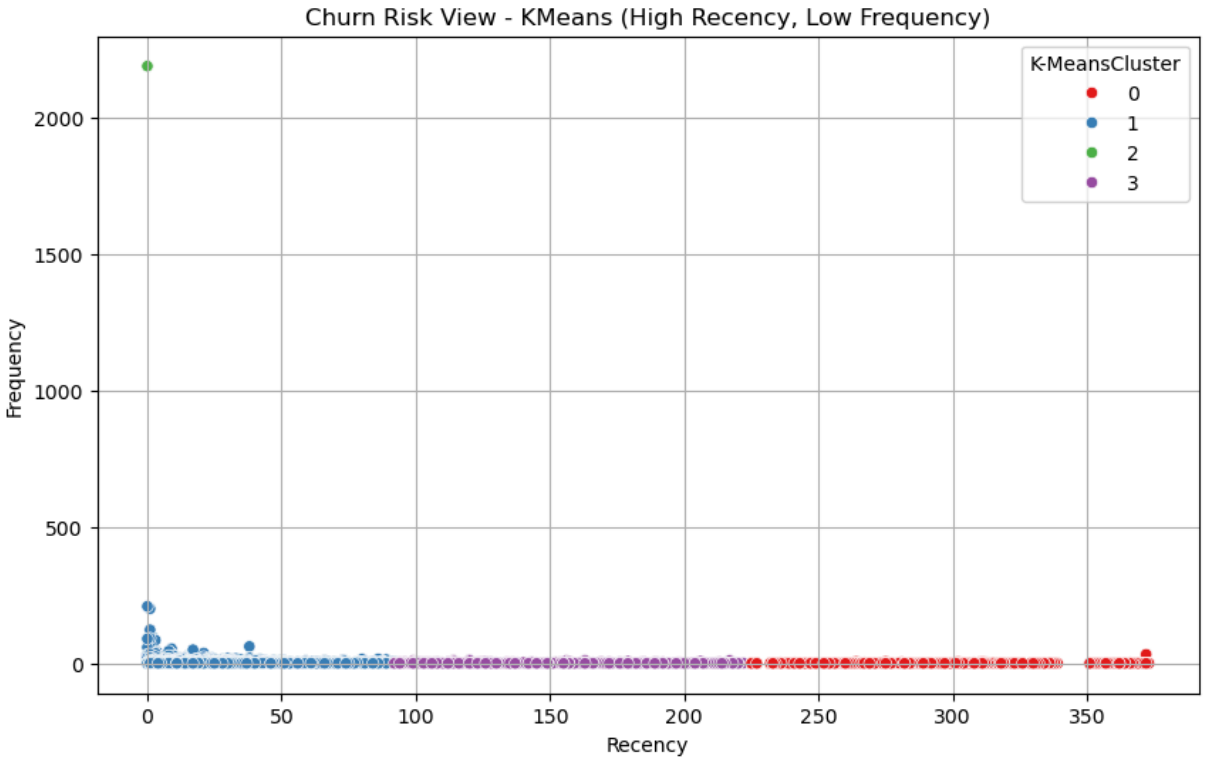
```
/tmp/ipykernel_116/4101572334.py:2: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.1
4.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.boxplot(data=df, x='GMMCluster', y='Frequency', palette='Greens')
```




```
Saved Most Frequent Customers per Segment of GMM to S3: s3://cse-projectdataviz/Freq
uentCustomersGMM.png
```

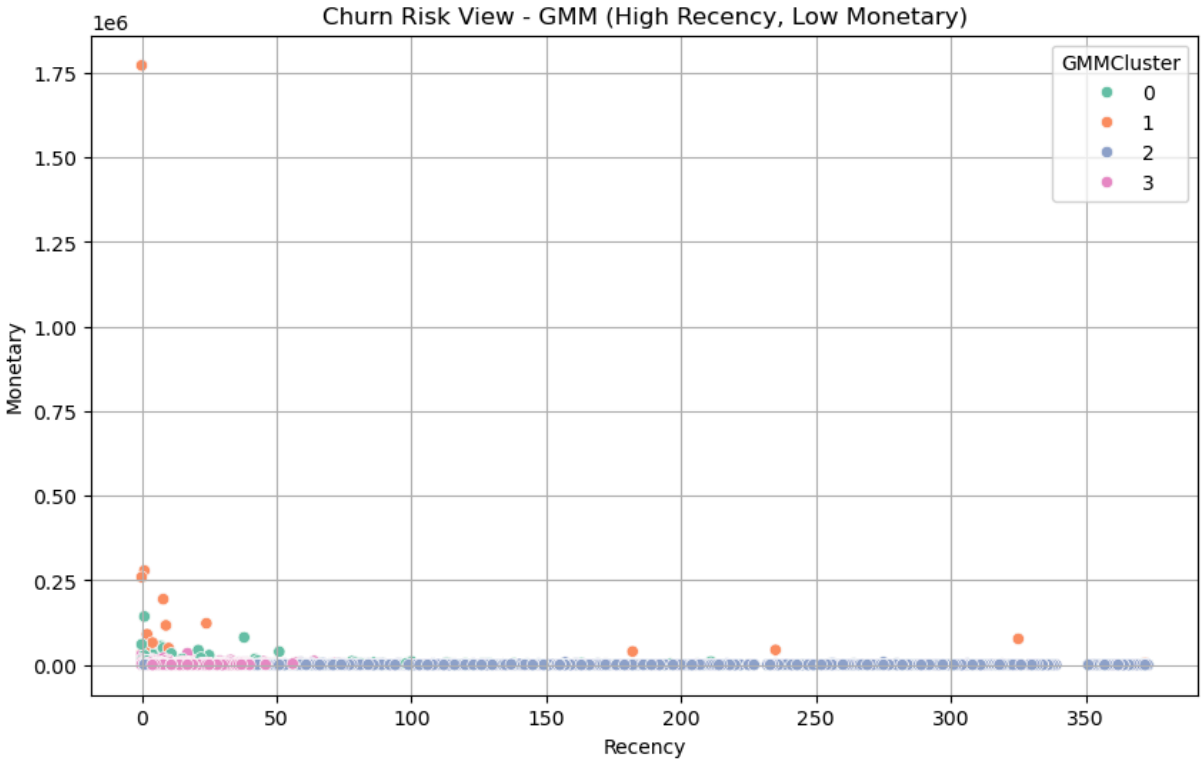
Churn Risk: High Recency, Low Frequency & Monetary

In [17]:
```
plt.figure(figsize=(10, 6))
sns.scatterplot(data=df, x='Recency', y='Frequency', hue='K-MeansCluster', palette=
plt.title("Churn Risk View - KMeans (High Recency, Low Frequency)")
plt.xlabel("Recency")
plt.ylabel("Frequency")
plt.grid(True)
plt.show()

# Save and upload
buffer = BytesIO()
plt.savefig(buffer, format='png')
buffer.seek(0)

key = 'ChurnRiskKMeans.png'
s3.upload_fileobj(buffer, bucket_name, key)
plt.close()

print(f"Saved Churn Risk of K-Means to S3: s3://{bucket_name}/{key}")
```




```
Saved Churn Risk of K-Means to S3: s3://cse-projectdataviz/ChurnRiskKMeans.png
```

In [18]:
```
plt.figure(figsize=(10, 6))
sns.scatterplot(data=df, x='Recency', y='Monetary', hue='GMMCluster', palette='Set2
plt.title("Churn Risk View - GMM (High Recency, Low Monetary)")
plt.xlabel("Recency")
plt.ylabel("Monetary")
plt.grid(True)
plt.show()

# Save and upload
buffer = BytesIO()
plt.savefig(buffer, format='png')
buffer.seek(0)

key = 'ChurnRiskGMM.png'
s3.upload_fileobj(buffer, bucket_name, key)
plt.close()

print(f"Saved Churn Risk of GMM to S3: s3://{bucket_name}/{key}")
```



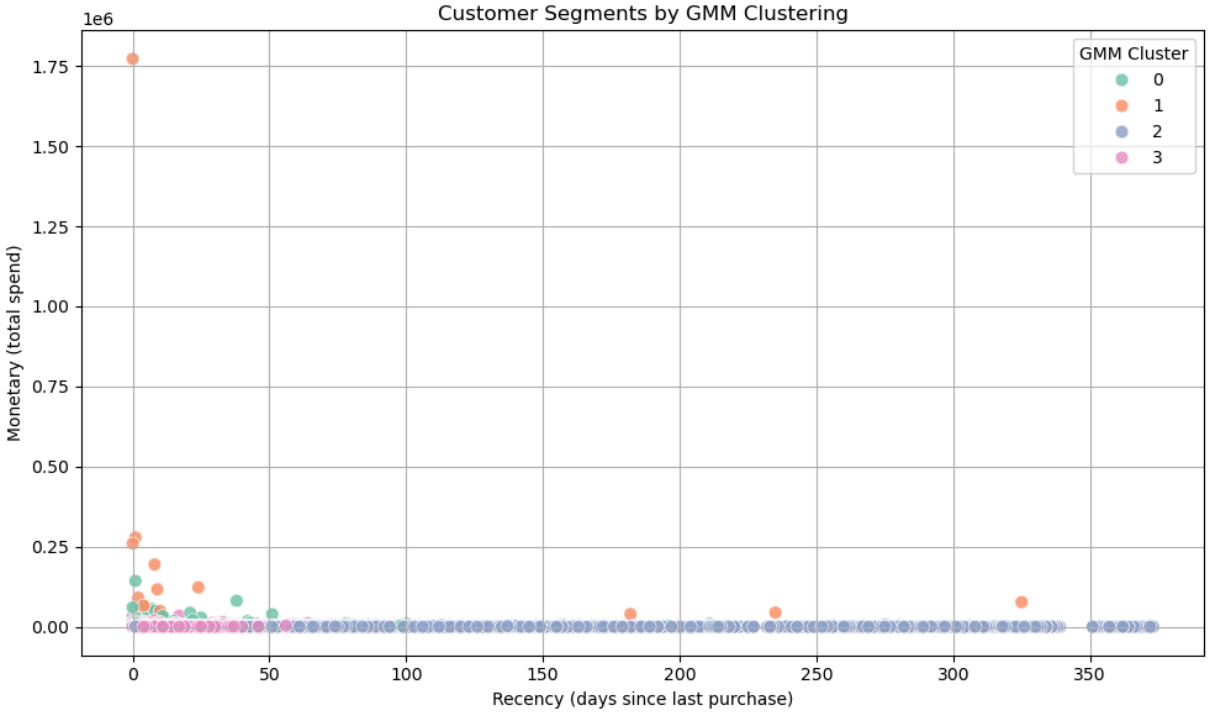

```
Saved Churn Risk of GMM to S3: s3://cse-projectdataviz/ChurnRiskGMM.png
```

Scatter Plot – K-Means Clusters

In [19]:
```
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(10, 6))
sns.scatterplot(
    data=df,
    x="Recency",
    y="Monetary",
    hue="K-MeansCluster",
    palette="Set1",
    s=60,
    alpha=0.8
)
plt.title("Customer Segments by K-Means Clustering")
plt.xlabel("Recency (days since last purchase)")
plt.ylabel("Monetary (total spend)")
plt.legend(title="K-Means Cluster")
plt.grid(True)
plt.tight_layout()
plt.show()
```

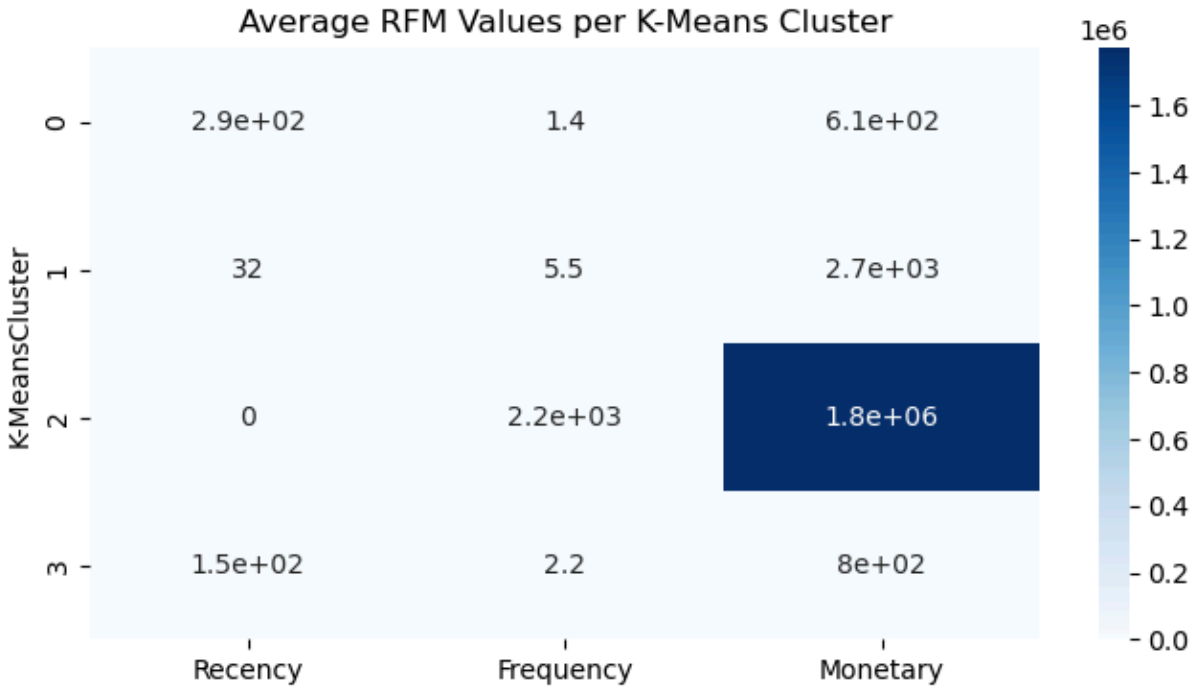
Scatter Plot – GMM Clusters

In [20]:
```
plt.figure(figsize=(10, 6))
sns.scatterplot(
    data=df,
    x="Recency",
    y="Monetary",
    hue="GMMCluster",
    palette="Set2",
    s=60,
    alpha=0.8
)
plt.title("Customer Segments by GMM Clustering")
plt.xlabel("Recency (days since last purchase)")
plt.ylabel("Monetary (total spend)")
plt.legend(title="GMM Cluster")
plt.grid(True)
plt.tight_layout()
plt.show()
```



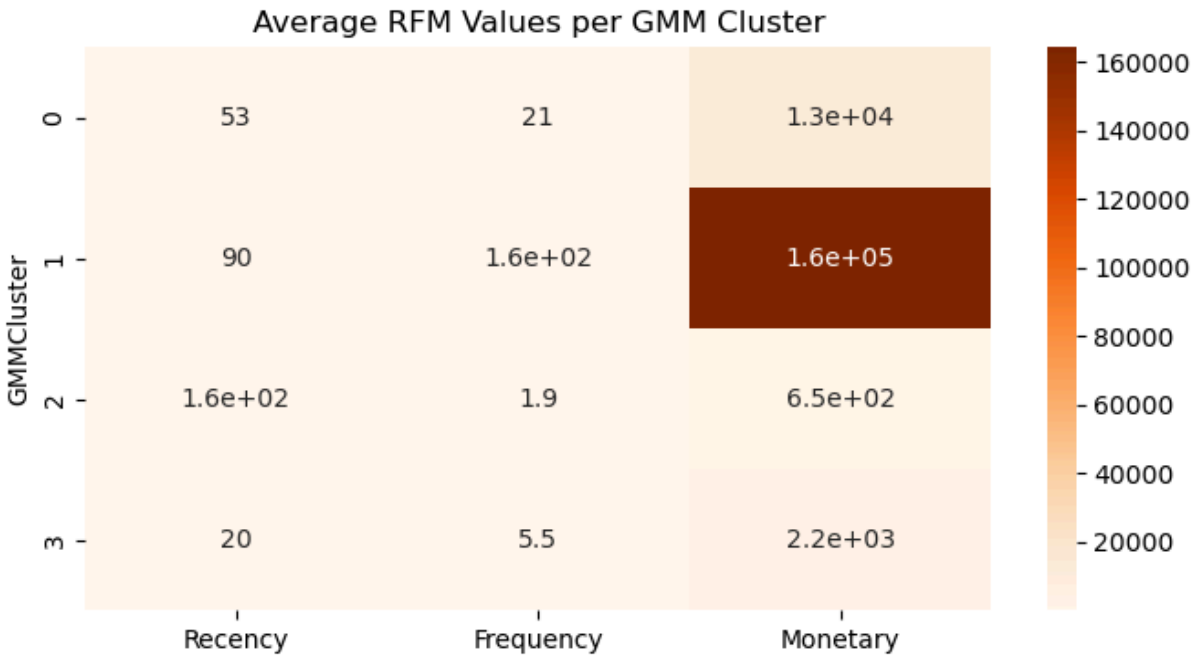
Cluster-wise Average Feature Heatmap Goal: Compare average RFM scores across clusters.

In [21]:
```
# K-Means heatmap
plt.figure(figsize=(8, 4))
sns.heatmap(df.groupby("K-MeansCluster")[["Recency", "Frequency", "Monetary"]].mean
plt.title("Average RFM Values per K-Means Cluster")
plt.show()
```



In [22]:
```
# GMM heatmap
plt.figure(figsize=(8, 4))
sns.heatmap(df.groupby("GMMCluster")[["Recency", "Frequency", "Monetary"]].mean(),
```

```
plt.title("Average RFM Values per GMM Cluster")
plt.show()
```



Clustering Plot for KMeans and GMM

Apply Log Transformation Before PCA

In [23]:
```
from sklearn.preprocessing import FunctionTransformer
from sklearn.pipeline import Pipeline

# Apply log(1 + x) to handle skewness (RFM usually has long tail)
log_transform = FunctionTransformer(func=lambda x: np.log1p(x), validate=True)
rfm_log = log_transform.fit_transform(df[['Recency', 'Frequency', 'Monetary']])
```

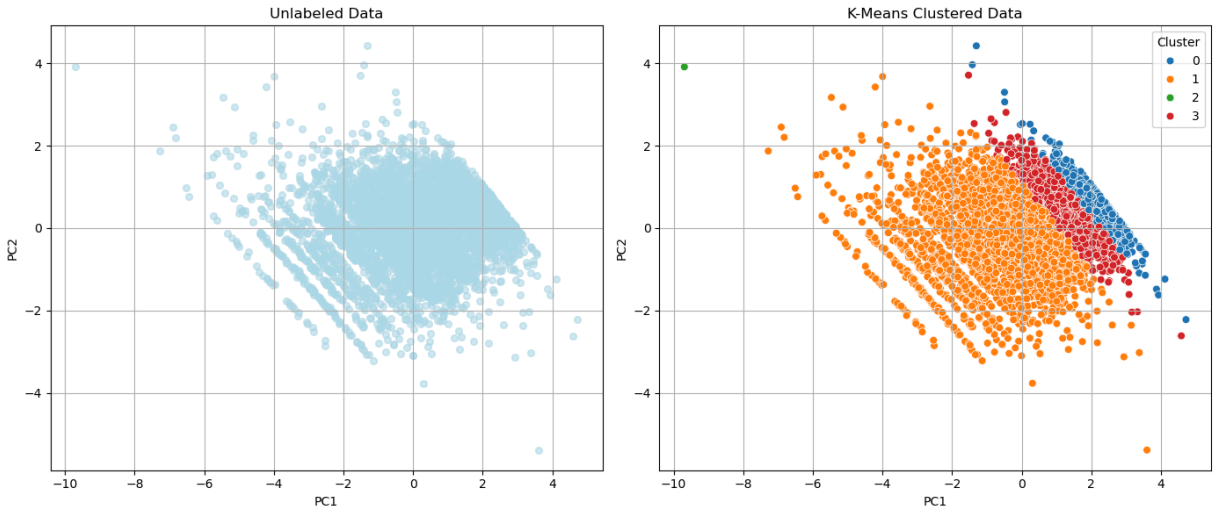
Re-run PCA on log-transformed data

In [24]:
```
from sklearn.decomposition import PCA

# PCA after log transformation
pca = PCA(n_components=2)
reduced = pca.fit_transform(rfm_log)

df['PC1'] = reduced[:, 0]
df['PC2'] = reduced[:, 1]
```

Visualize

In [25]:
```
import matplotlib.pyplot as plt
import seaborn as sns
# K-Means cluster plot
plt.figure(figsize=(14, 6))
# Unlabeled Data
plt.subplot(1, 2, 1)
```

```
plt.scatter(df['PC1'], df['PC2'], s=30, alpha=0.6, color='lightblue')
plt.title("Unlabeled Data")
plt.xlabel("PC1")
plt.ylabel("PC2")
plt.grid(True)
# K-Means Clustered
plt.subplot(1, 2, 2)
sns.scatterplot(x='PC1', y='PC2', hue='K-MeansCluster', data=df, palette='tab10', s
plt.title("K-Means Clustered Data")
plt.xlabel("PC1")
plt.ylabel("PC2")
plt.grid(True)
plt.legend(title="Cluster")
plt.tight_layout()
plt.show()
# Save and upload
buffer = BytesIO()
plt.savefig(buffer, format='png')
buffer.seek(0)
key = 'ClusteringKMeans.png'
s3.upload_fileobj(buffer, bucket_name, key)
plt.close()
print(f"Saved Clustering of K-Means to S3: s3://{bucket_name}/{key}")
```

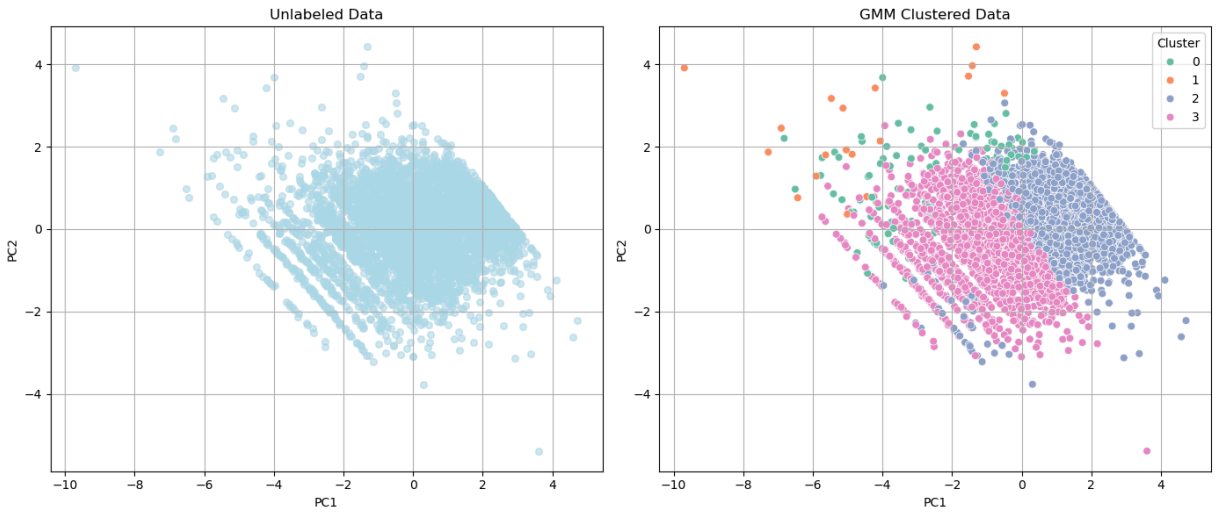



```
Saved Clustering of K-Means to S3: s3://cse-projectdataviz/ClusteringKMeans.png
```

In [26]:

```
plt.figure(figsize=(14, 6))

# Unlabeled Data
plt.subplot(1, 2, 1)
plt.scatter(df['PC1'], df['PC2'], s=30, alpha=0.6, color='lightblue')
plt.title("Unlabeled Data")
plt.xlabel("PC1")
plt.ylabel("PC2")
plt.grid(True)

# GMM Clustered
plt.subplot(1, 2, 2)
sns.scatterplot(x='PC1', y='PC2', hue='GMMCluster', data=df, palette='Set2', s=40)
plt.title("GMM Clustered Data")
plt.xlabel("PC1")
```

```
plt.ylabel("PC2")
plt.grid(True)
plt.legend(title="Cluster")

plt.tight_layout()
plt.show()

# Save and upload
buffer = BytesIO()
plt.savefig(buffer, format='png')
buffer.seek(0)

key = 'ClusteringGMM.png'
s3.upload_fileobj(buffer, bucket_name, key)
plt.close()

print(f"Saved clustering of GMM to S3: s3://{bucket_name}/{key}")
```




```
Saved clustering of GMM to S3: s3://cse-projectdataviz/ClusteringGMM.png
```

In [ ]: