In [25]:
```
# --- IMPORT LIBRARIES ---
import pandas as pd
import numpy as np
from datetime import timedelta
from sklearn.preprocessing import StandardScaler
from sklearn.mixture import GaussianMixture
from sklearn.cluster import AgglomerativeClustering
from scipy.spatial.distance import pdist
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
```

In [2]:
```
# --- LOAD & CLEAN DATA ---
df = pd.read_excel(r"C:\Users\mukki\OneDrive\Desktop\Online Retail.xlsx", sheet_nam

df
```

Out[2]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	C
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 08:26:00	2.55	17850.0	K
1	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00	3.39	17850.0	K
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	2010-12-01 08:26:00	2.75	17850.0	K
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	2010-12-01 08:26:00	3.39	17850.0	K
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	2010-12-01 08:26:00	3.39	17850.0	K
...	...	...	...	...	...	...	...	
4995	536836	21843	RED RETROSPOT CAKE STAND	2	2010-12-02 18:08:00	10.95	18168.0	K
4996	536836	21531	RED RETROSPOT SUGAR JAM BOWL	2	2010-12-02 18:08:00	2.55	18168.0	K
4997	536836	21539	RED RETROSPOT BUTTER DISH	3	2010-12-02 18:08:00	4.95	18168.0	K
4998	536836	22198	LARGE POPCORN HOLDER	2	2010-12-02 18:08:00	1.65	18168.0	K
4999	536836	22197	SMALL POPCORN HOLDER	2	2010-12-02 18:08:00	0.85	18168.0	K

5000 rows × 8 columns

In [9]:
```
df=df.dropna(subset=['CustomerID'])
df = df[(df['Quantity'] > 0) & (df['UnitPrice'] > 0)].copy()
df['TotalPrice'] = df['Quantity'] * df['UnitPrice']
df['InvoiceDate'] = pd.to_datetime(df['InvoiceDate'])
df['InvoiceDate']
```

Out[9]:
```
0      2010-12-01 08:26:00
1      2010-12-01 08:26:00
2      2010-12-01 08:26:00
3      2010-12-01 08:26:00
4      2010-12-01 08:26:00
               ...
4995   2010-12-02 18:08:00
4996   2010-12-02 18:08:00
4997   2010-12-02 18:08:00
4998   2010-12-02 18:08:00
4999   2010-12-02 18:08:00
Name: InvoiceDate, Length: 3725, dtype: datetime64[ns]
```

In [11]:
```
# --- RFM ANALYSIS ---
snapshot_date = df['InvoiceDate'].max() + timedelta(days=1)
rfm = df.groupby('CustomerID').agg({
    'InvoiceDate': lambda x: (snapshot_date - x.max()).days,
    'InvoiceNo': 'nunique',
    'TotalPrice': 'sum'
})
rfm
rfm.columns = ['Recency', 'Frequency', 'Monetary']
```

In [13]:
```
# --- ADDITIONAL FEATURES ---

# Loyalty Score: normalized Frequency
rfm['LoyaltyScore'] = (rfm['Frequency'] - rfm['Frequency'].min()) / (rfm['Frequency
rfm['LoyaltyScore']
```

Out[13]:
```
CustomerID
12431.0    0.000000
12433.0    0.000000
12583.0    0.000000
12662.0    0.000000
12748.0    0.030303
             ...
18085.0    0.000000
18144.0    0.000000
18168.0    0.000000
18229.0    0.000000
18239.0    0.000000
Name: LoyaltyScore, Length: 179, dtype: float64
```

In [15]:
```
# Discount Utilization Rate: estimated by detecting purchases with low price per it
product_avg_price = df.groupby('StockCode')['UnitPrice'].mean()
df = df.join(product_avg_price, on='StockCode', rsuffix='_Avg')
df['DiscountUsed'] = np.where(df['UnitPrice'] < 0.8 * df['UnitPrice_Avg'], 1, 0)
```

```
df['DiscountUsed']
```

Out[15]:
```
0       0
1       0
2       0
3       0
4       0
       ..
4995    0
4996    0
4997    0
4998    0
4999    0
Name: DiscountUsed, Length: 3725, dtype: int32
```

In [17]:
```
discount_rate = df.groupby('CustomerID')['DiscountUsed'].mean()
rfm['DiscountRate'] = rfm.index.map(discount_rate)

rfm['DiscountRate']
```

Out[17]:
```
CustomerID
12431.0    0.0
12433.0    0.0
12583.0    0.0
12662.0    0.0
12748.0    0.0
          ...
18085.0    0.0
18144.0    0.0
18168.0    0.0
18229.0    0.0
18239.0    0.0
Name: DiscountRate, Length: 179, dtype: float64
```

In [39]:
```
# Fill missing discount rates (if customer only purchased items at avg or above pri
rfm['DiscountRate'] = rfm['DiscountRate'].fillna(0)

rfm['DiscountRate']
```

Out[39]:
```
CustomerID
12431.0    0.0
12433.0    0.0
12583.0    0.0
12662.0    0.0
12748.0    0.0
          ...
18085.0    0.0
18144.0    0.0
18168.0    0.0
18229.0    0.0
18239.0    0.0
Name: DiscountRate, Length: 179, dtype: float64
```

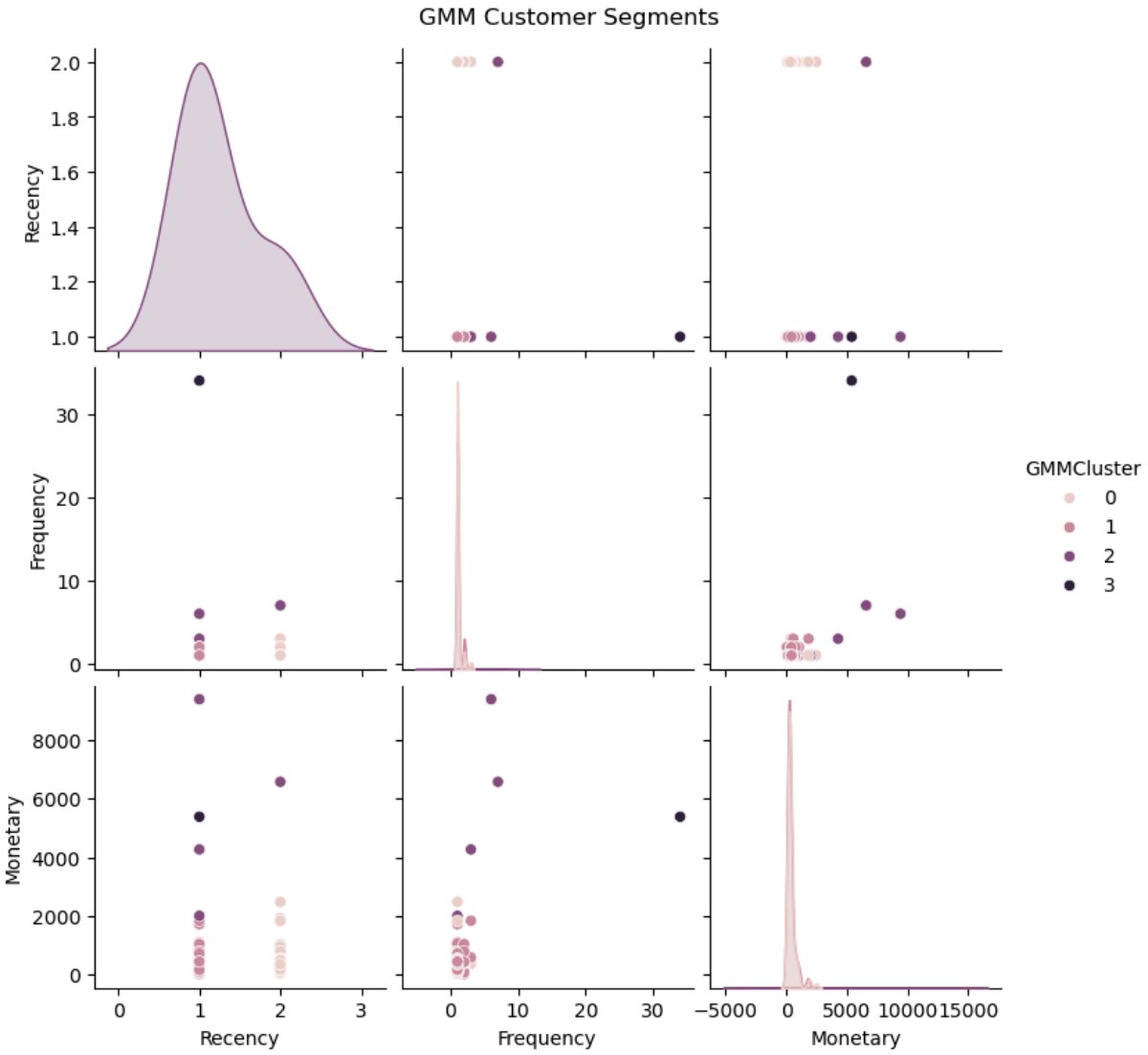
In [21]:
```
# Feature matrix
features = ['Recency', 'Frequency', 'Monetary', 'LoyaltyScore', 'DiscountRate']
```

```
X = rfm[features].copy()
X_scaled = StandardScaler().fit_transform(X)
```

In [29]:
```
# --- CLUSTERING ---
import warnings
warnings.filterwarnings("ignore",category=UserWarning)
# GMM Clustering
gmm = GaussianMixture(n_components=4, random_state=42)
rfm['GMMCluster'] = gmm.fit_predict(X_scaled)
```

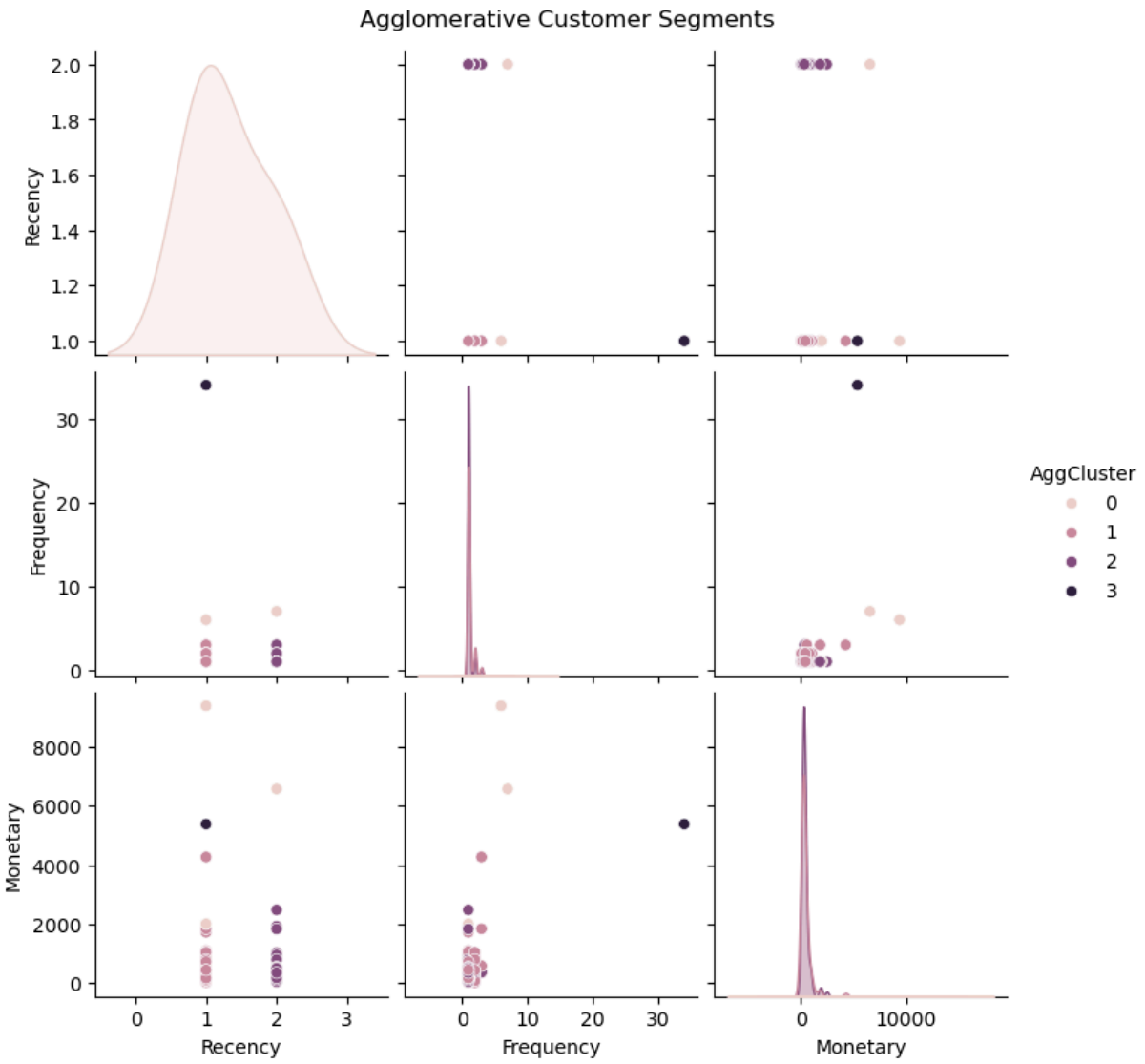
In [31]:
```
# Agglomerative Clustering
agg = AgglomerativeClustering(n_clusters=4, linkage='ward')
rfm['AggCluster'] = agg.fit_predict(X_scaled)
```

In [33]:
```
# --- VISUALIZATION ---
sns.pairplot(rfm, vars=['Recency', 'Frequency', 'Monetary'], hue='GMMCluster')
plt.suptitle("GMM Customer Segments", y=1.02)
plt.show()
```



In [35]:
```
sns.pairplot(rfm, vars=['Recency', 'Frequency', 'Monetary'], hue='AggCluster')
plt.suptitle("Agglomerative Customer Segments", y=1.02)
```

```
plt.show()
```



In [37]:
```
# --- SEGMENT PROFILES ---
profiles = rfm.groupby('GMMCluster')[features].mean()
print("GMM Segment Profiles:\n", profiles)
```

```
GMM Segment Profiles:
              Recency  Frequency     Monetary  LoyaltyScore  DiscountRate
GMMCluster
0                2.00   1.102273   380.701818      0.003099      0.001117
1                1.00   1.162791   347.442209      0.004933      0.003522
2                1.25   4.250000  5566.605000      0.098485      0.191366
3                1.00  34.000000  5391.210000      1.000000      0.000000
```

In [41]:
```
# --- RECOMMENDATIONS BASED ON CLUSTERS ---
recommendations = {
    0: "Engage high loyalty spenders with VIP offers.",
    1: "Re-engage dormant customers with time-sensitive discounts.",
    2: "Reward frequent discounters with flash sales.",
    3: "Upsell to medium-frequency buyers with bundles."
}
```

In [43]:
```
print("\nTargeted Marketing Strategies:")
for cluster_id, message in recommendations.items():
    print(f"Segment {cluster_id}: {message}")
```

```
Targeted Marketing Strategies:
Segment 0: Engage high loyalty spenders with VIP offers.
Segment 1: Re-engage dormant customers with time-sensitive discounts.
Segment 2: Reward frequent discounters with flash sales.
Segment 3: Upsell to medium-frequency buyers with bundles.
```

In [ ]: