Phase 1: EDA Objective: The goal is to understand the data and how strong certain columns are tied to each other to ultimately help us determine how to drive higher retention.

Column Name	Description
id	Unique identifier for each customer
age	Age of the customer
gender	Gender of the customer (Male, Female, Other)
income	Annual income of the customer (in USD)
spending_score	Spending score (1-100), indicating spending behavior and loyalty
membership_years	Number of years the customer has been a member
purchase_frequency	Number of purchases made in the last year
preferred_category	Preferred shopping category (Electronics, Clothing, Groceries, etc.)
last_purchase_amount	Amount spent on the last purchase (in USD)

In [1]:
```
pip install pandas scikit-learn openpyxl
```

```
Requirement already satisfied: pandas in c:\users\user\anaconda3\lib\site-packages (2.2.2)
Requirement already satisfied: scikit-learn in c:\users\user\anaconda3\lib\site-packages (1.5.1)
Requirement already satisfied: openpyxl in c:\users\user\anaconda3\lib\site-packages (3.1.5)
Requirement already satisfied: numpy>=1.26.0 in c:\users\user\anaconda3\lib\site-packages (from pandas) (1.26.4)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\user\anaconda3\lib\site-packages (from pandas)
(2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in c:\users\user\anaconda3\lib\site-packages (from pandas) (2024.1)
Requirement already satisfied: tzdata>=2022.7 in c:\users\user\anaconda3\lib\site-packages (from pandas) (2023.3
)
Requirement already satisfied: scipy>=1.6.0 in c:\users\user\anaconda3\lib\site-packages (from scikit-learn) (1.
13.1)
Requirement already satisfied: joblib>=1.2.0 in c:\users\user\anaconda3\lib\site-packages (from scikit-learn) (1
.4.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in c:\users\user\anaconda3\lib\site-packages (from scikit-le
arn) (3.5.0)
Requirement already satisfied: et-xmlfile in c:\users\user\anaconda3\lib\site-packages (from openpyxl) (1.1.0)
Requirement already satisfied: six>=1.5 in c:\users\user\anaconda3\lib\site-packages (from python-dateutil>=2.8.
2->pandas) (1.16.0)
Note: you may need to restart the kernel to use updated packages.
```

In [3]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
```

In [7]:
```
# Basic Summary Statitics


# Load the CSV file
customer = pd.read_csv('D:/MS Data Science/customer_segmentation_data.csv')

# Display basic summary statistics
display(customer.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 9 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   id                    1000 non-null   int64  
 1   age                   1000 non-null   int64  
 2   gender                1000 non-null   object 
 3   income                1000 non-null   int64  
 4   spending_score        1000 non-null   int64  
 5   membership_years      1000 non-null   int64  
 6   purchase_frequency    1000 non-null   int64  
 7   preferred_category    1000 non-null   object 
 8   last_purchase_amount  1000 non-null   float64
dtypes: float64(1), int64(6), object(2)
memory usage: 70.4+ KB
None
```

In [28]:
```
customer = pd.read_csv('customer_segmentation_data.csv')

display(customer.head())
```

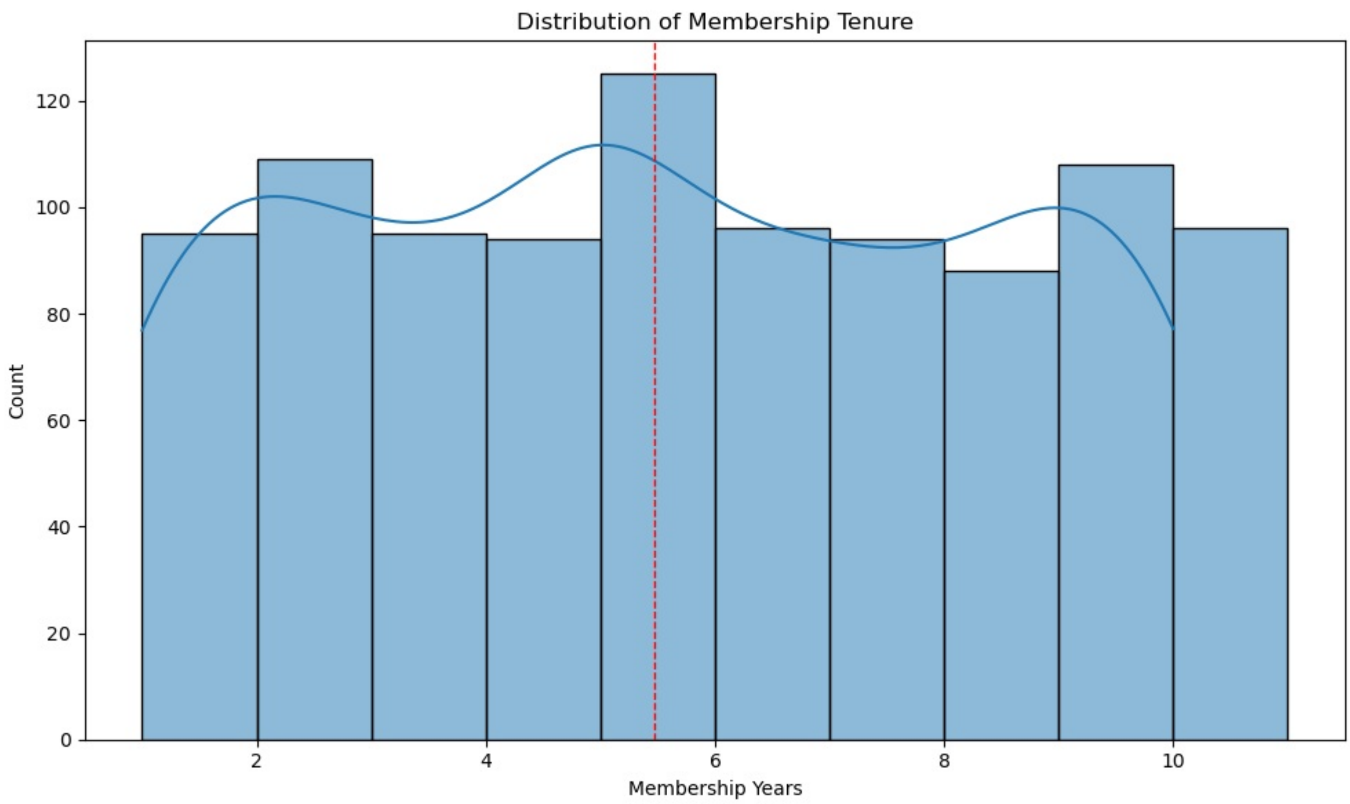
	id	age	gender	income	spending_score	membership_years	purchase_frequency	preferred_category	last_purchase_amount
0	1	38	Female	99342	90	3	24	Groceries	113.53
1	2	21	Female	78852	60	2	42	Sports	41.93
2	3	60	Female	126573	30	2	28	Clothing	424.36
3	4	40	Other	47099	74	9	5	Home & Garden	991.93
4	5	65	Female	140621	21	3	25	Electronics	347.08

In [9]:
```
display(customer.describe())
```

	id	age	income	spending_score	membership_years	purchase_frequency	last_purchase_amount
count	1000.000000	1000.000000	1000.000000	1000.000000	1000.00000	1000.000000	1000.000000
mean	500.500000	43.783000	88500.800000	50.685000	5.46900	26.596000	492.348670
std	288.819436	15.042213	34230.771122	28.955175	2.85573	14.243654	295.744253
min	1.000000	18.000000	30004.000000	1.000000	1.00000	1.000000	10.400000
25%	250.750000	30.000000	57911.750000	26.000000	3.00000	15.000000	218.762500
50%	500.500000	45.000000	87845.500000	50.000000	5.00000	27.000000	491.595000
75%	750.250000	57.000000	116110.250000	76.000000	8.00000	39.000000	747.170000
max	1000.000000	69.000000	149973.000000	100.000000	10.00000	50.000000	999.740000

EDA: In this section I will be focused on identifying the relationships between these variables to identify what makes up the best customers

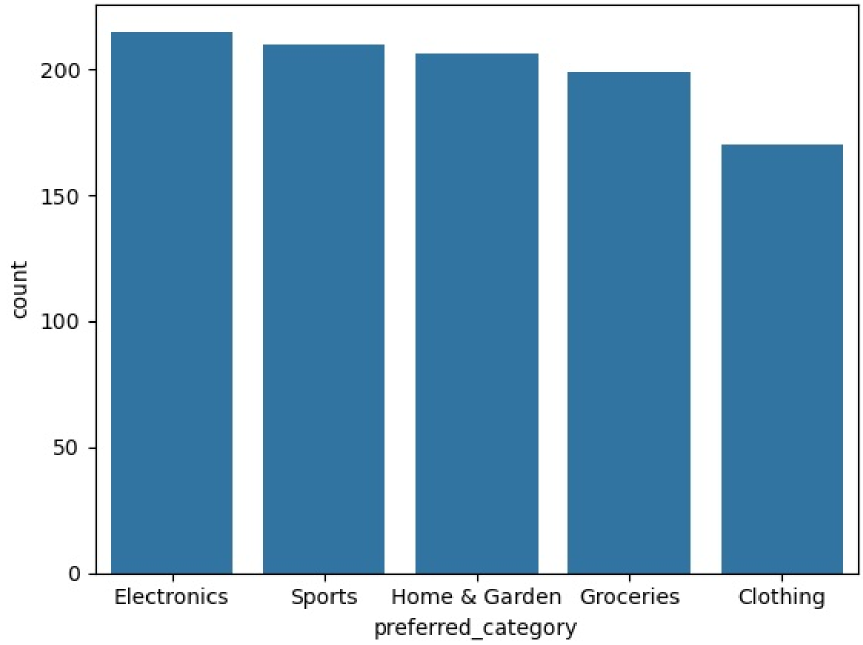
In [11]:
```
#See the distribution of what the average membership looks like
#Histogram — Membership Distribution
plt.figure(figsize=(10, 6))
sns.histplot(customer['membership_years'], kde=True, bins=range(1, 12), edgecolor='black')
plt.axvline(customer['membership_years'].mean(), color='red', linestyle='--', linewidth=1)
plt.title('Distribution of Membership Tenure')
plt.xlabel('Membership Years')
plt.ylabel('Count')
plt.tight_layout()
plt.show()
```



Membership tenure is relatively balanced across the range, but the highest concentration of customers has been active for about six years. This indicates a strong mid-tenure base, suggesting that retention strategies should focus on reinforcing value around the five to seven-year mark to maintain long-term loyalty and reduce drop-off.

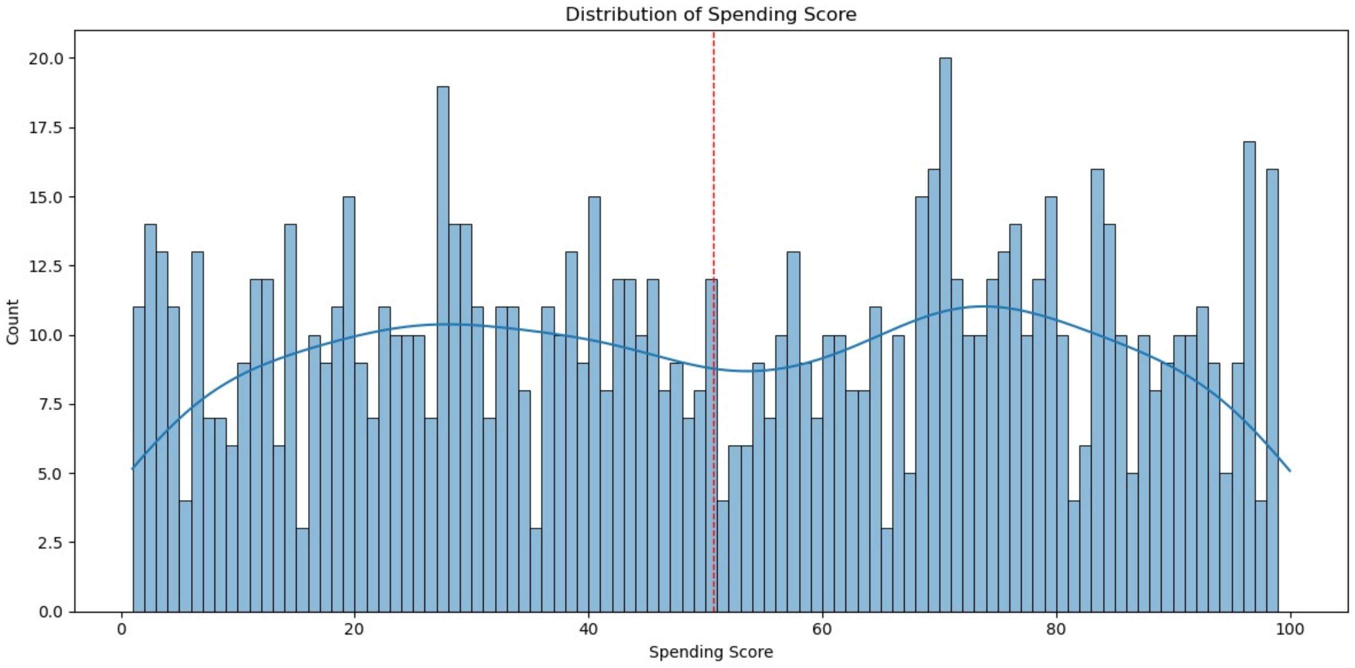
In [44]:
```
#Provide a breakdown on the quantity of orders by category
sns.countplot(data=customer, x='preferred_category', order=customer['preferred_category'].value_counts().index)
```

Out[44]: <Axes: xlabel='preferred_category', ylabel='count'>



Electronics, Sports, and Home & Garden are the top three preferred categories among customers, suggesting these are key product areas driving engagement. Loyalty retention efforts should prioritize perks, promotions, or exclusive benefits tied to these categories to maximize impact.

In [17]:
```
#Distribution of the Spending Score
plt.figure(figsize=(12, 6))
sns.histplot(customer['spending_score'], kde=True, bins=range(1, 100), edgecolor='black')
plt.axvline(customer['spending_score'].mean(), color='red', linestyle='--', linewidth=1)
plt.title('Distribution of Spending Score')
plt.xlabel('Spending Score')
plt.ylabel('Count')
plt.tight_layout()
plt.show()
```

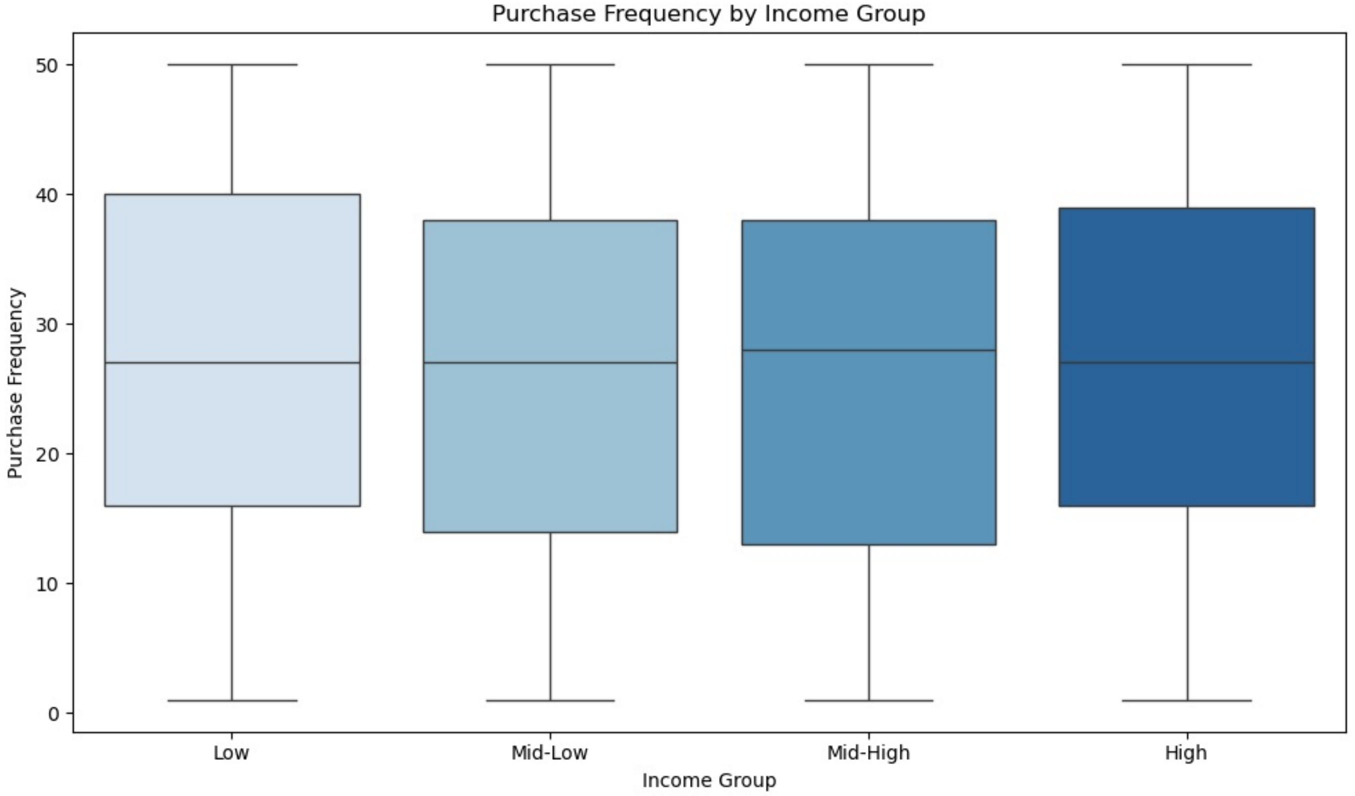


Spending scores are spread widely across the customer base with noticeable spikes at both low and high ends. This bimodal pattern suggests we may have two distinct customer segments—one frugal and one high-spending—which could inform tiered loyalty incentives to improve retention.

In [25]:
```
#Group customers into buckets by income level
customer['income_group'] = pd.qcut(customer['income'], q=4, labels=['Low', 'Mid-Low', 'Mid-High', 'High'])

# Boxplot: frequency per income group
plt.figure(figsize=(10, 6))
sns.boxplot(data=customer, x='income_group', y='purchase_frequency', palette='Blues')
plt.title('Purchase Frequency by Income Group')
plt.xlabel('Income Group')
plt.ylabel('Purchase Frequency')
```

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

```
/var/folders/66/5v7r_k0d7dq4xv5h9vk2t8nc0000gn/T/ipykernel_14900/4111551766.py:6: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable
to `hue` and set `legend=False` for the same effect.

  sns.boxplot(data=customer, x='income_group', y='purchase_frequency', palette='Blues')
```

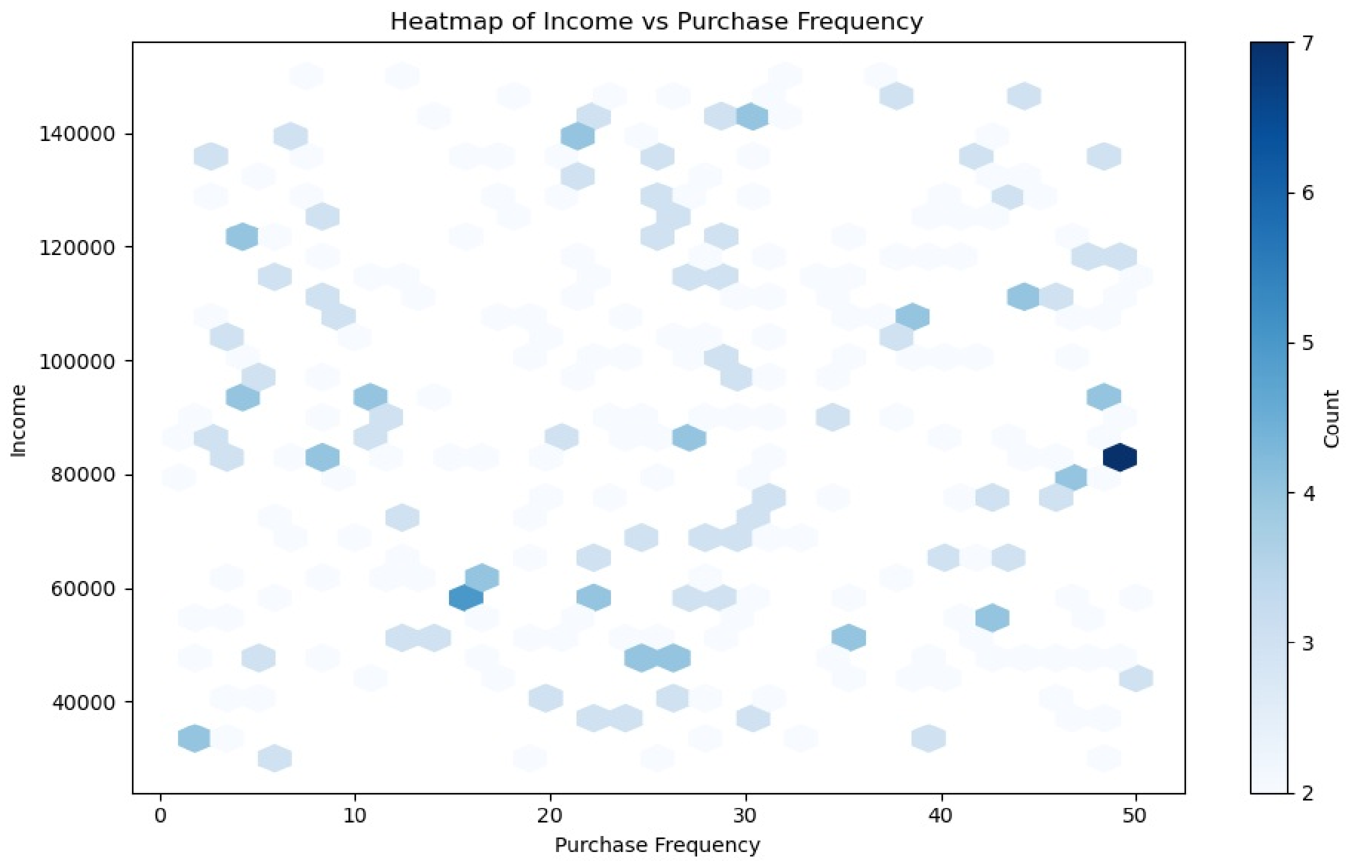


Customers across all income levels show similar median purchase frequency, but the variability is higher in lower and higher income groups. This suggests that income alone doesn't strongly predict how often someone buys—other behavioral or demographic factors may better explain loyalty.

In [29]:
```
customer.groupby('preferred_category')['id'].nunique()
```

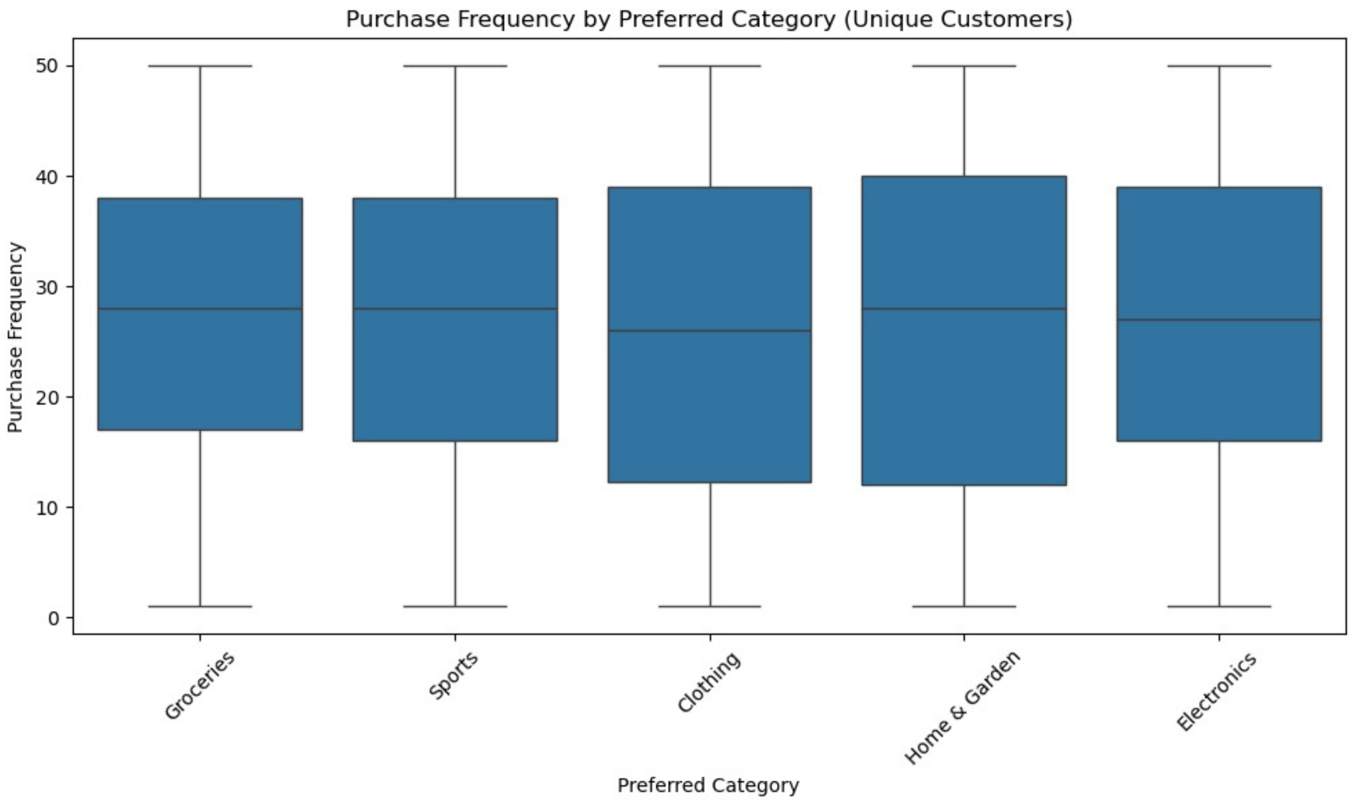
Out[29]:
```
preferred_category
Clothing         170
Electronics      215
Groceries        199
Home & Garden    206
Sports           210
Name: id, dtype: int64
```

In [27]:
```
plt.figure(figsize=(10, 6))
plt.hexbin(customer['purchase_frequency'], customer['income'], gridsize=30, cmap='Blues', mincnt=2)
plt.colorbar(label='Count')
plt.xlabel('Purchase Frequency')
plt.ylabel('Income')
plt.title('Heatmap of Income vs Purchase Frequency')
plt.tight_layout()
plt.show()
```



The heatmap shows that there’s no strong linear relationship between income and purchase frequency. While purchases occur across all income levels, the highest cluster density appears in the $50k-$ 80k income range with moderate to high purchase frequency. This suggests that mid-income customers might be key drivers of activity within the loyalty program.

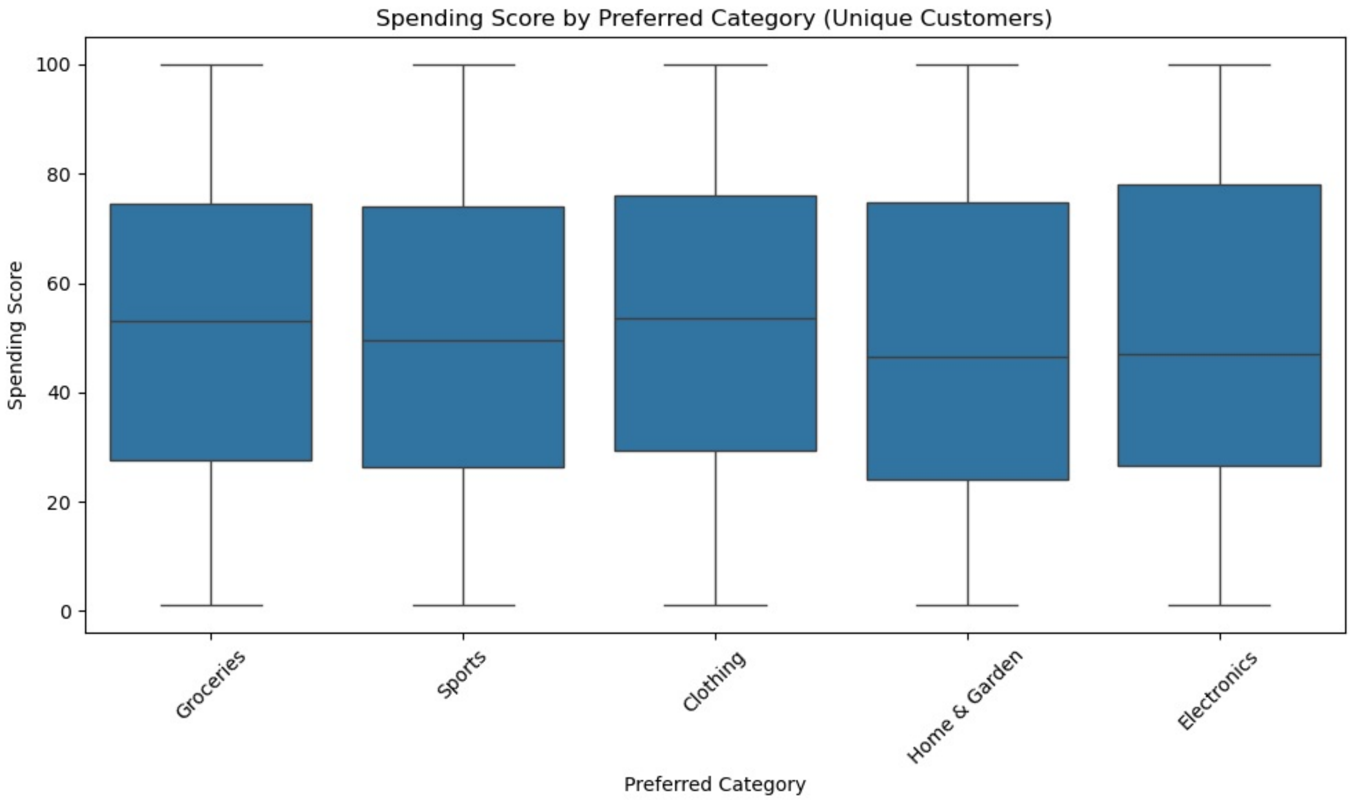
In [31]:
```
# Optional: drop duplicates just in case
customer_unique = customer.drop_duplicates(subset='id')

plt.figure(figsize=(10, 6))
sns.boxplot(data=customer_unique, x='preferred_category', y='purchase_frequency')
plt.title('Purchase Frequency by Preferred Category (Unique Customers)')
plt.xlabel('Preferred Category')
plt.ylabel('Purchase Frequency')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



Purchase frequency is fairly consistent across preferred categories, but Home & Garden customers show slightly higher median and upper-range frequencies. These shoppers might represent a more engaged segment and could be worth targeting with loyalty perks or exclusive offers.

In [32]:

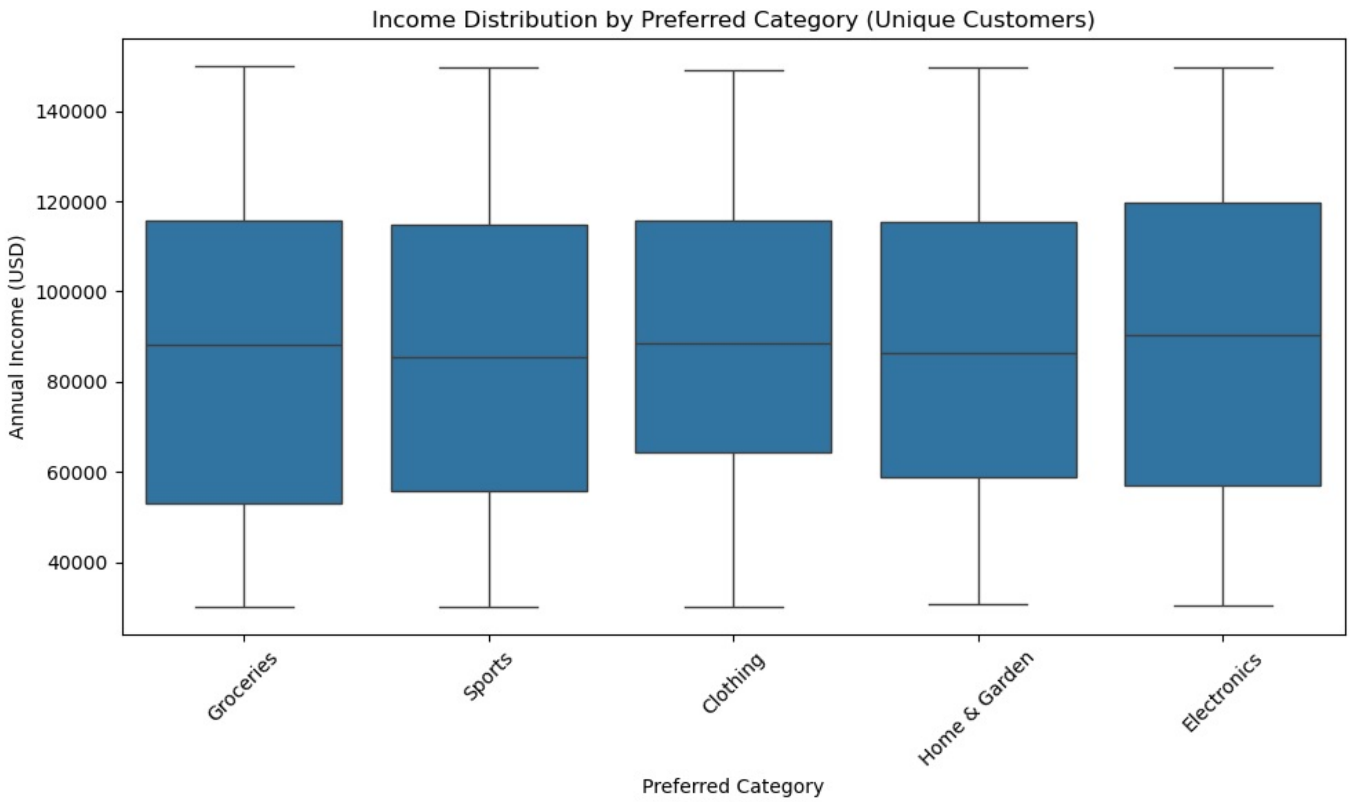
```
#spending spread by the categories
plt.figure(figsize=(10, 6))
sns.boxplot(data=customer_unique, x='preferred_category', y='spending_score')
plt.title('Spending Score by Preferred Category (Unique Customers)')
plt.xlabel('Preferred Category')
plt.ylabel('Spending Score')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



This boxplot compares how much unique customers tend to spend across different product categories. While all categories have a wide range, electronics and clothing have slightly higher median spending scores. This suggests these categories might attract customers with greater willingness to spend, which could be helpful when planning promotions or bundling strategies.

In [34]:

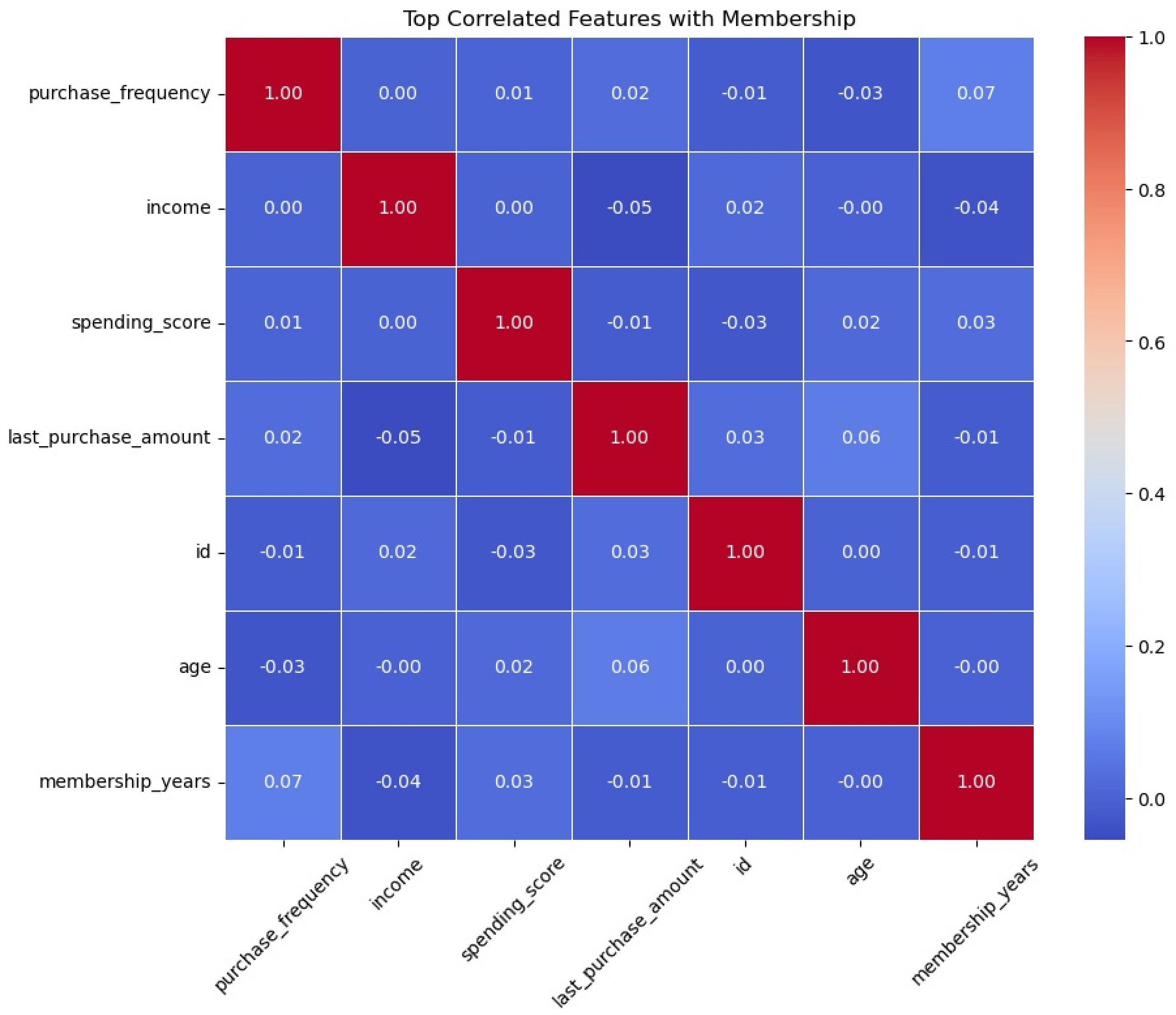
```
#Identify if there are any trends between income and the spend in categories
plt.figure(figsize=(10, 6))
sns.boxplot(data=customer_unique, x='preferred_category', y='income')
plt.title('Income Distribution by Preferred Category (Unique Customers)')
plt.xlabel('Preferred Category')
plt.ylabel('Annual Income (USD)')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



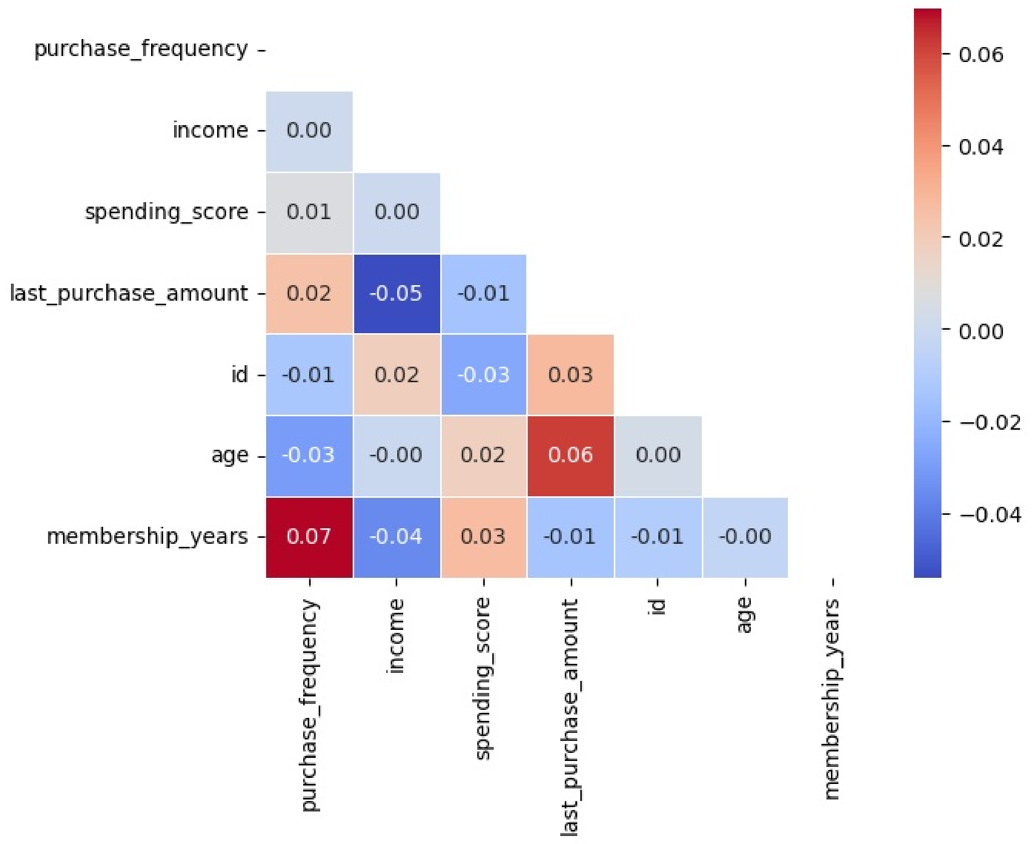
This boxplot shows how annual income varies across preferred shopping categories. Customers who prefer electronics appear to have slightly higher median incomes, while other categories like groceries and sports are more evenly spread. The wide range in every category suggests income alone doesn't dictate category preference, but electronics might appeal more to higher-earning customers.

In [35]:
```
#Compute the top corrrelated features with Membership
corr_matrix = customer.corr(numeric_only=True)
top_corr = corr_matrix['membership_years'].abs().sort_values(ascending=False)[1:11]
top_features = top_corr.index.tolist()
top_corr_matrix = customer[top_features + ['membership_years']].corr()
```

In [38]:
```
plt.figure(figsize=(10, 8))
sns.heatmap(top_corr_matrix, annot=True, fmt=".2f", cmap="coolwarm", linewidths=0.5)
plt.title("Top Correlated Features with Membership")
plt.xticks(rotation=45)
plt.yticks(rotation=0)
plt.show()

mask = np.triu(np.ones_like(top_corr_matrix, dtype=bool))
sns.heatmap(top_corr_matrix, mask=mask, annot=True, fmt=".2f", cmap="coolwarm", linewidths=0.5)
```

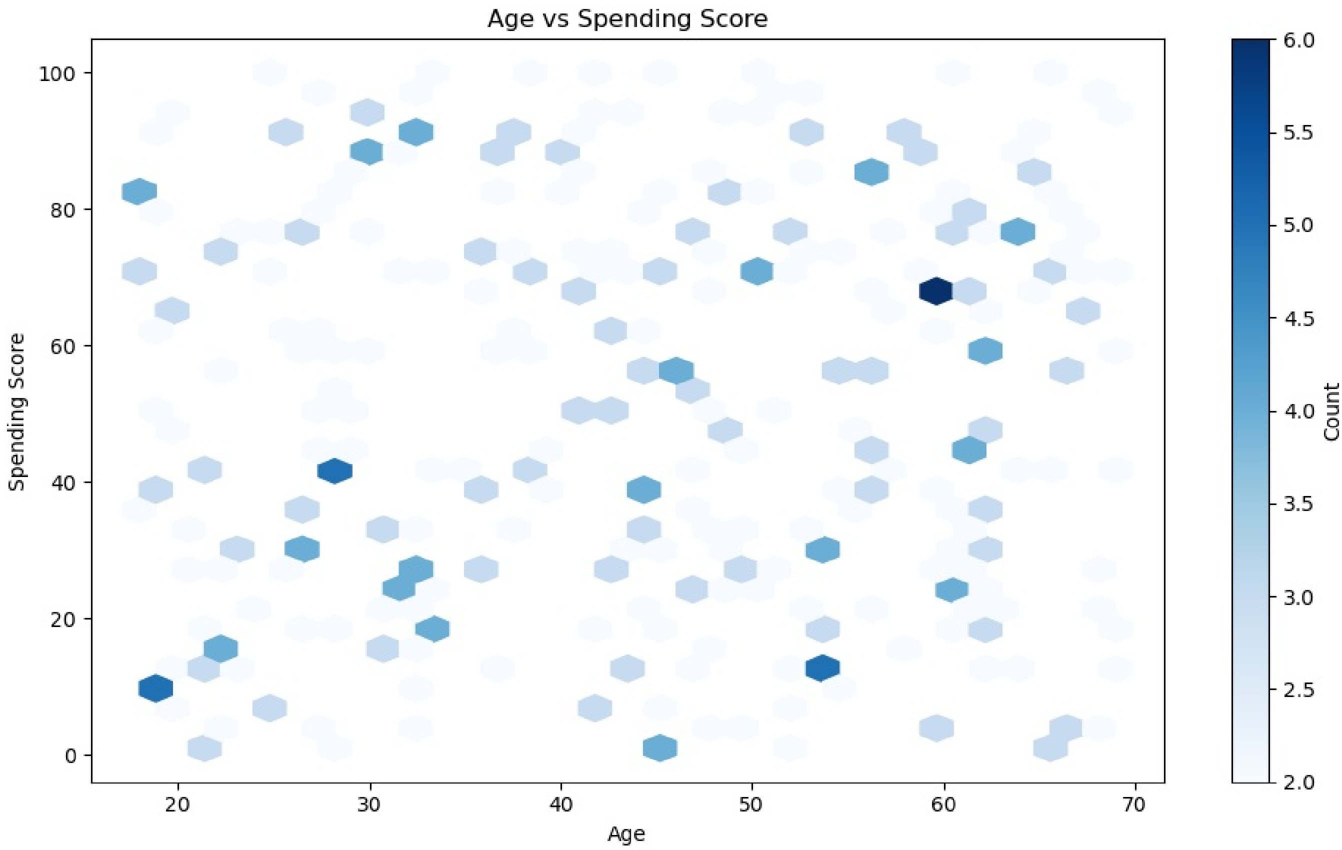


Out[38]: <Axes: >



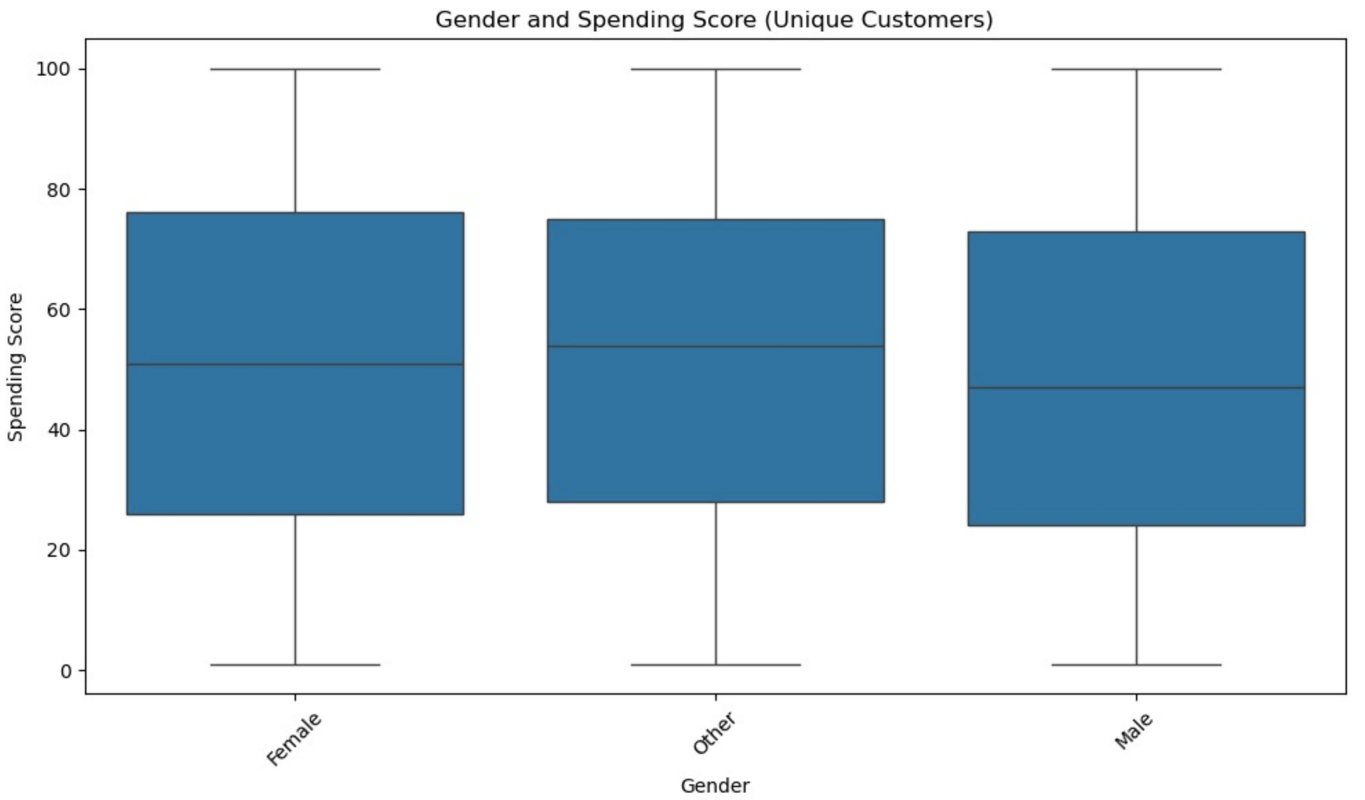
None of the features show strong linear correlations with each other. The highest positive relationship is between membership years and purchase frequency, but even that is very weak. This suggests that most variables contribute independently when predicting membership behavior.

In [9]:
```
plt.figure(figsize=(10, 6))
plt.hexbin(customer['age'], customer['spending_score'], gridsize=30, cmap='Blues', mincnt=2)
plt.colorbar(label='Count')
plt.xlabel('Age')
plt.ylabel('Spending Score')
plt.title('Age vs Spending Score')
plt.tight_layout()
plt.show()
```



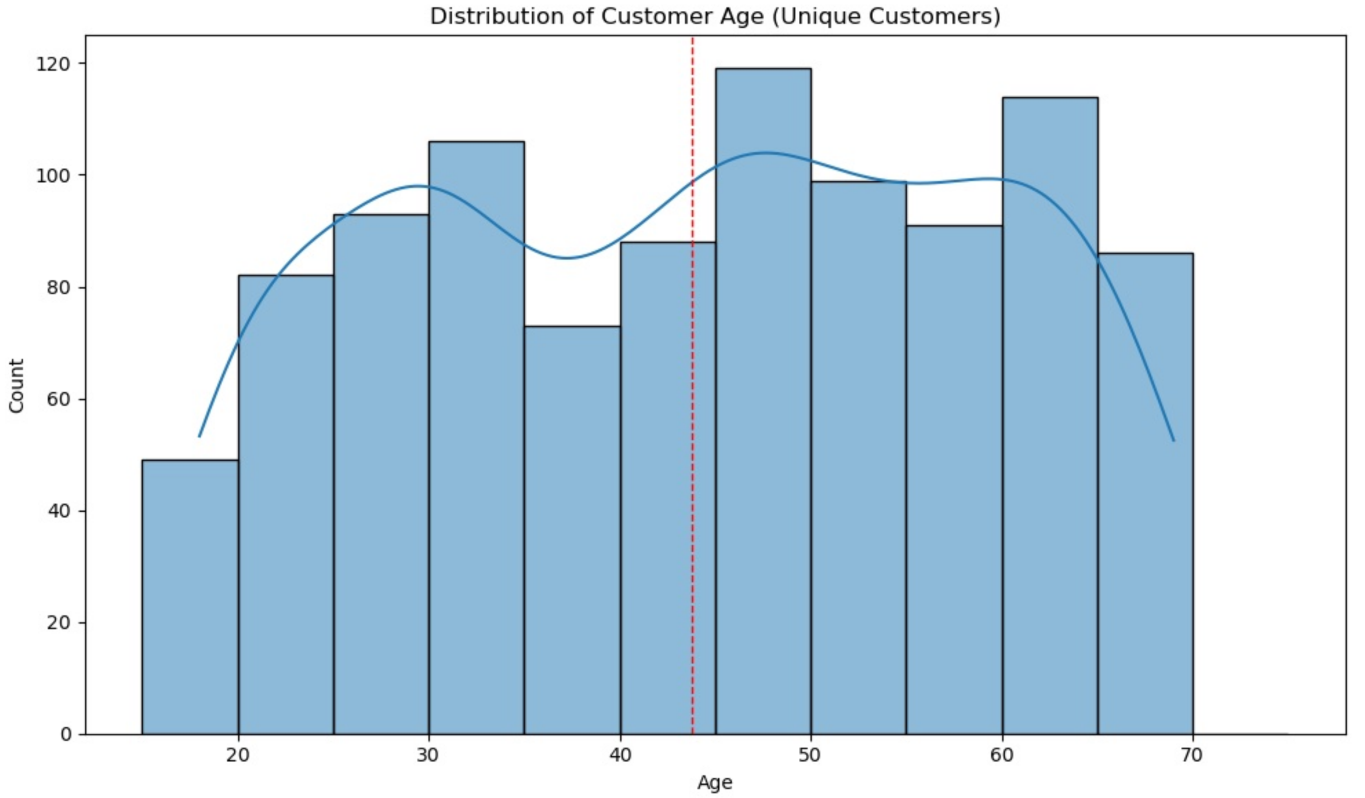
This hexbin plot shows that spending behavior is distributed across all age groups without a strong trend. However, there are slightly denser pockets of higher spending around ages 30, 50, and 60, suggesting these groups may include more active spenders.

In [11]:
```
customer_unique = customer.drop_duplicates(subset='id')
#Box Plots of Gender
plt.figure(figsize=(10, 6))
sns.boxplot(data=customer_unique, x='gender', y='spending_score')
plt.title('Gender and Spending Score (Unique Customers)')
plt.xlabel('Gender')
plt.ylabel('Spending Score')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



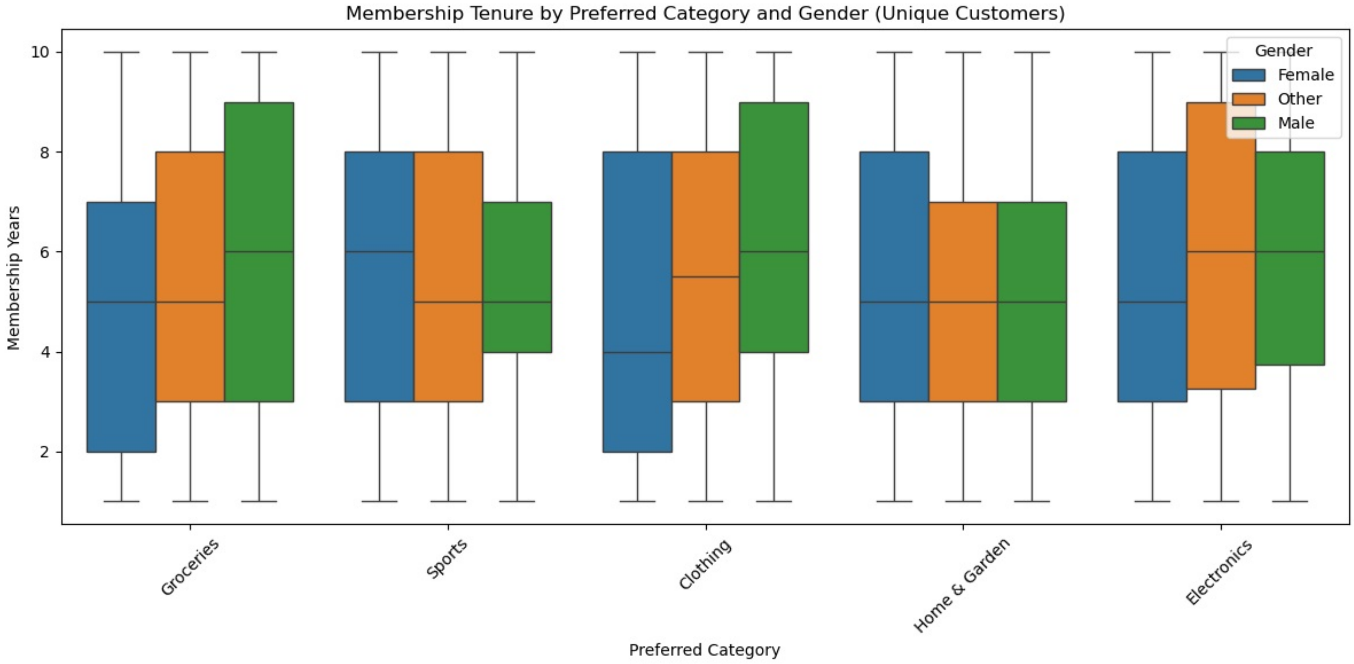
After filtering to unique customers, we observed similar spending score distributions across gender groups. While minor differences exist, this suggests that loyalty behaviors are not strongly skewed by gender alone.

In [14]:
```
customer_unique = customer.drop_duplicates(subset='id')
#See the distribution of what the average membership looks like
plt.figure(figsize=(10, 6))
sns.histplot(customer_unique['age'], kde=True, bins=range(15, 80, 5), edgecolor='black')
plt.axvline(customer_unique['age'].mean(), color='red', linestyle='--', linewidth=1)
plt.title('Distribution of Customer Age (Unique Customers)')
plt.xlabel('Age')
plt.ylabel('Count')
plt.tight_layout()
plt.show()
```



In [16]:
```
# Make sure you're using unique customers, Membership by Gender
customer_unique = customer.drop_duplicates(subset='id')

# Plot membership tenure by category and gender
plt.figure(figsize=(12, 6))
```

```
sns.boxplot(
    data=customer_unique,
    x='preferred_category',
    y='membership_years',
    hue='gender'
)
plt.title('Membership Tenure by Preferred Category and Gender (Unique Customers)')
plt.xlabel('Preferred Category')
plt.ylabel('Membership Years')
plt.xticks(rotation=45)
plt.legend(title='Gender', loc='upper right')
plt.tight_layout()
plt.show()
```



In [ ]:

In [17]:
```
# Filter to unique customers
customer_unique = customer.drop_duplicates(subset='id')

# Group and aggregate
summary_table = (
    customer_unique
    .groupby(['preferred_category', 'gender'])['membership_years']
    .agg(['count', 'mean', 'median', 'std'])
    .reset_index()
    .rename(columns={
        'count': 'Customer Count',
        'mean': 'Avg Tenure (yrs)',
        'median': 'Median Tenure',
        'std': 'Std Dev'
    })
)

# Round for readability
summary_table = summary_table.round(2)

# Display table
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np

summary_table
```

Out[17]:

	preferred_category	gender	Customer Count	Avg Tenure (yrs)	Median Tenure	Std Dev
0	Clothing	Female	56	4.75	4.0	3.04
1	Clothing	Male	56	6.12	6.0	2.92
2	Clothing	Other	58	5.62	5.5	2.92
3	Electronics	Female	65	5.63	5.0	2.67
4	Electronics	Male	76	5.87	6.0	2.80
5	Electronics	Other	74	5.89	6.0	2.99
6	Groceries	Female	66	4.92	5.0	2.85
7	Groceries	Male	71	5.72	6.0	3.09
8	Groceries	Other	62	5.29	5.0	2.92
9	Home & Garden	Female	68	5.54	5.0	2.83
10	Home & Garden	Male	77	4.90	5.0	2.63
11	Home & Garden	Other	61	5.28	5.0	2.87
12	Sports	Female	61	5.67	6.0	2.79
13	Sports	Male	77	5.39	5.0	2.68
14	Sports	Other	72	5.42	5.0	2.81

In [ ]:

In [ ]:

In [ ]:

In [ ]:

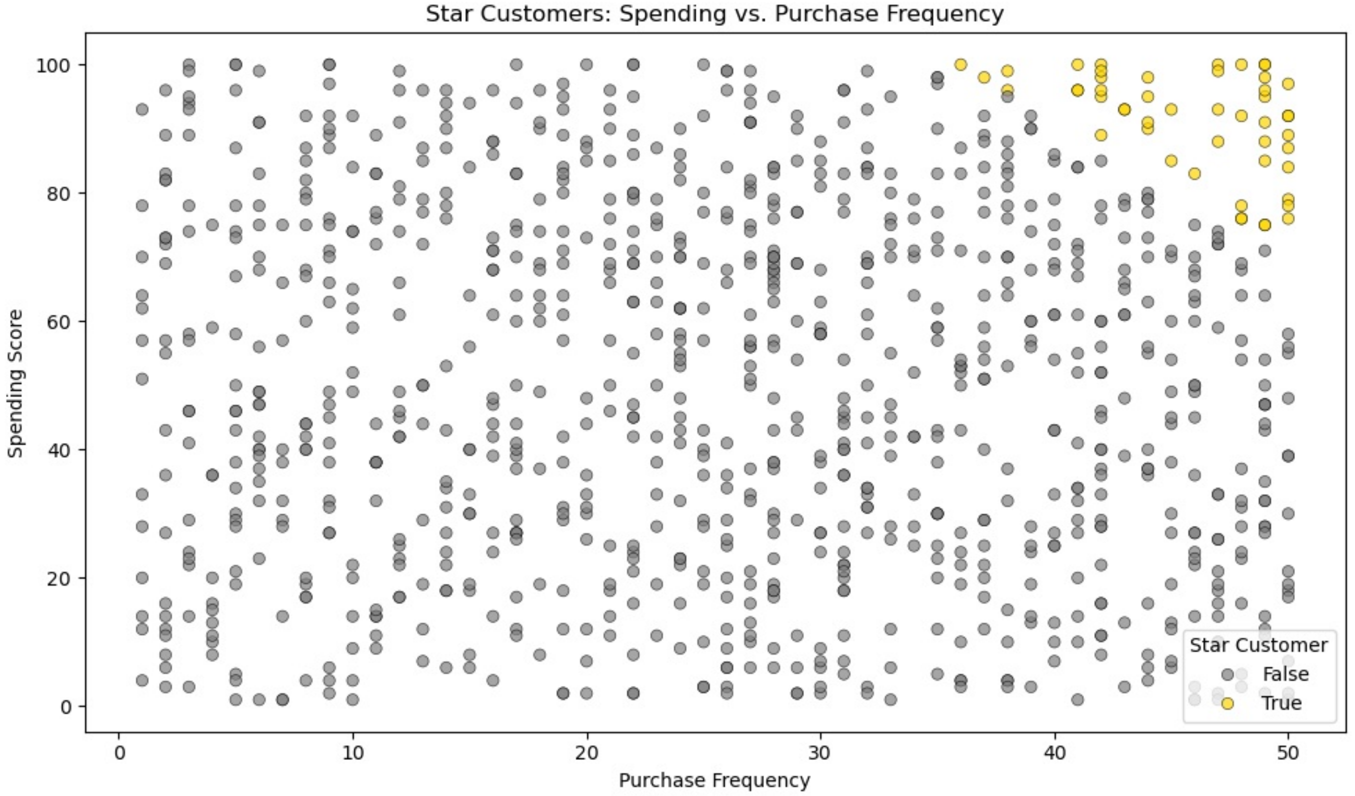
In [18]:
```
customer_unique = customer.drop_duplicates(subset='id')

# Rank-based score between 0–1
customer_unique['loyalty_score'] = (
    customer_unique['purchase_frequency'].rank(pct=True) +
    customer_unique['spending_score'].rank(pct=True)
) / 2
```

In [19]:
```
star_cutoff = customer_unique['loyalty_score'].quantile(0.95)
star_customers = customer_unique[customer_unique['loyalty_score'] >= star_cutoff]
```

In [20]:
```
star_profile = (
    star_customers
    .groupby(['preferred_category', 'gender'])[['income', 'age', 'membership_years']]
    .agg(['mean', 'median', 'count'])
    .round(2)
)
```

In [24]:
```
import matplotlib.pyplot as plt
import seaborn as sns

# First, make sure 'loyalty_score' and 'star_customers' are calculated
customer_unique = customer.drop_duplicates(subset='id')
customer_unique['loyalty_score'] = (
    customer_unique['purchase_frequency'].rank(pct=True) +
    customer_unique['spending_score'].rank(pct=True)
) / 2

# Define star customers (top 5%)
star_cutoff = customer_unique['loyalty_score'].quantile(0.95)
customer_unique['is_star'] = customer_unique['loyalty_score'] >= star_cutoff

# Plot
plt.figure(figsize=(10, 6))
sns.scatterplot(
    data=customer_unique,
    x='purchase_frequency',
    y='spending_score',
    hue='is_star',
    palette={True: 'gold', False: 'gray'},
    alpha=0.7,
    edgecolor='black'
)
plt.title('Star Customers: Spending vs. Purchase Frequency')
plt.xlabel('Purchase Frequency')
```

```python
plt.ylabel('Spending Score')
plt.legend(title='Star Customer', loc='lower right')
plt.tight_layout()
plt.show()
```



This scatterplot shows that star customers, those in the top 5 percent of the loyalty score, are concentrated in the upper right corner. They consistently purchase more often and spend more, making them the most valuable segment. These customers should be prioritized for exclusive offers, loyalty rewards, or early product access to increase retention and lifetime value.

This scatterplot highlights that star customers, those in the top 5% of loyalty score—cluster in the upper-right quadrant, combining high purchase frequency with high spending scores. These customers represent the most valuable segment and should be prioritized for exclusive offers, loyalty rewards, or early product access to maximize retention and lifetime value.

In [26]:
```python
star_summary = (
    star_customers
    .groupby(['preferred_category', 'gender'])[['income', 'age', 'membership_years']]
    .agg(['mean', 'median', 'count'])
    .round(2)
)
display(star_summary.describe())
```

	income			age			membership_years		
	mean	median	count	mean	median	count	mean	median	count
count	14.000000	14.000000	14.000000	14.000000	14.000000	14.000000	14.00000	14.000000	14.000000
mean	91105.483571	92749.035714	3.571429	44.271429	44.178571	3.571429	5.90500	6.321429	3.571429
std	18431.593766	21157.149614	1.283881	6.682582	10.915069	1.283881	1.73749	2.358350	1.283881
min	57698.000000	49020.000000	2.000000	33.500000	28.000000	2.000000	3.67000	2.000000	2.000000
25%	78130.512500	79777.750000	3.000000	38.372500	36.625000	3.000000	4.49750	5.000000	3.000000
50%	91879.875000	93848.750000	3.000000	45.335000	45.500000	3.000000	6.00000	6.750000	3.000000
75%	100017.585000	109079.875000	4.000000	50.670000	50.625000	4.000000	7.35500	8.500000	4.000000
max	121518.670000	125014.000000	6.000000	53.500000	61.000000	6.000000	9.00000	9.500000	6.000000

In [30]:
```python
# Create a customer segmentation by income
customer_unique['income_group'] = pd.cut(
    customer_unique['income'],
    bins=[0, 40000, 60000, 80000, 100000, float('inf')],
    labels=['<40K', '40K–60K', '60K–80K', '80K–100K', '100K+']
)

# Membership Tenure Bins
customer_unique['tenure_group'] = pd.cut(
    customer_unique['membership_years'],
    bins=[0, 2, 5, 8, float('inf')],
```

```
    labels=['<2 yrs', '2–5 yrs', '5–8 yrs', '8+ yrs']
)
```

In [40]:

```
#Re acclimate to the star customer portion
star_cutoff = customer_unique['loyalty_score'].quantile(0.95)
star_customers = customer_unique[customer_unique['loyalty_score'] >= star_cutoff]

# Group by gender, income group, and tenure group
star_breakdown = (
    star_customers
    .groupby(['gender', 'income_group', 'tenure_group'], observed = True)
    .agg(
        customer_count=('id', 'count'),
        avg_purchase_freq=('purchase_frequency', 'mean'),
        avg_spending_score=('spending_score', 'mean')
    )
    .reset_index()
    .sort_values(by='customer_count', ascending=False)
    .round(2)
)

star_breakdown = star_breakdown[['gender', 'income_group', 'tenure_group',
                                 'customer_count', 'avg_purchase_freq', 'avg_spending_score']]

# Display the breakdown
import pandas as pd
import IPython.display as display

display.display(star_breakdown.sort_values(by='customer_count', ascending=False))
```

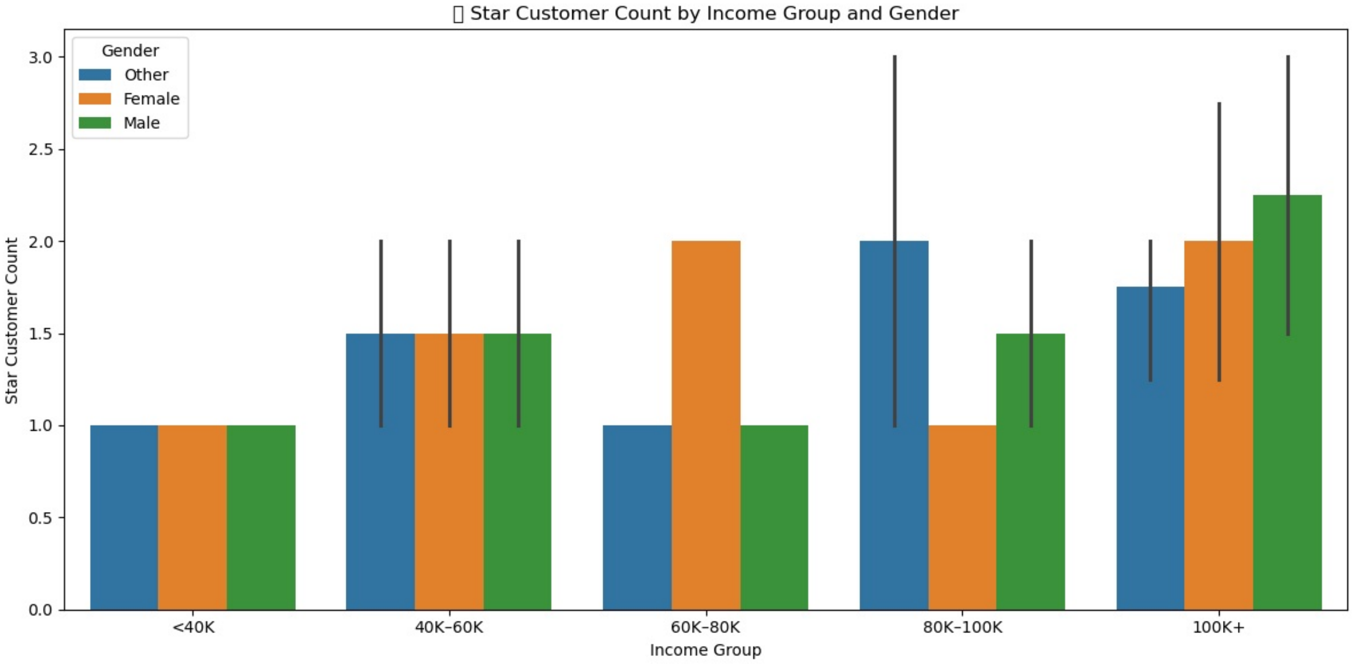
	gender	income_group	tenure_group	customer_count	avg_purchase_freq	avg_spending_score
26	Other	80K–100K	5–8 yrs	3	49.33	83.00
18	Male	100K+	5–8 yrs	3	45.33	87.33
17	Male	100K+	2–5 yrs	3	43.33	94.67
8	Female	100K+	8+ yrs	3	46.67	96.67
28	Other	100K+	2–5 yrs	2	46.00	96.00
22	Other	40K–60K	5–8 yrs	2	45.50	92.50
2	Female	40K–60K	8+ yrs	2	49.50	83.50
3	Female	60K–80K	8+ yrs	2	45.00	93.00
5	Female	100K+	<2 yrs	2	49.00	81.50
29	Other	100K+	5–8 yrs	2	45.00	97.00
7	Female	100K+	5–8 yrs	2	38.00	97.50
30	Other	100K+	8+ yrs	2	45.50	99.00
11	Male	40K–60K	8+ yrs	2	45.00	98.00
14	Male	80K–100K	5–8 yrs	2	47.50	91.50
16	Male	100K+	<2 yrs	2	45.00	88.50
1	Female	40K–60K	<2 yrs	1	50.00	76.00
4	Female	80K–100K	5–8 yrs	1	44.00	90.00
6	Female	100K+	2–5 yrs	1	37.00	98.00
9	Male	<40K	5–8 yrs	1	49.00	88.00
10	Male	40K–60K	<2 yrs	1	45.00	85.00
12	Male	60K–80K	<2 yrs	1	36.00	100.00
13	Male	60K–80K	8+ yrs	1	42.00	95.00
27	Other	100K+	<2 yrs	1	44.00	91.00
19	Male	100K+	8+ yrs	1	48.00	76.00
20	Other	<40K	<2 yrs	1	49.00	91.00
0	Female	<40K	2–5 yrs	1	49.00	95.00
23	Other	60K–80K	<2 yrs	1	50.00	84.00
24	Other	60K–80K	2–5 yrs	1	50.00	79.00
25	Other	80K–100K	2–5 yrs	1	50.00	97.00
21	Other	40K–60K	<2 yrs	1	42.00	100.00
15	Male	80K–100K	8+ yrs	1	49.00	96.00

The highest star customer counts are concentrated in the 80K–100K and 100K+ income groups, typically with 5–8 years of tenure. Interestingly, customers with longer tenure also show higher average purchase frequency and spending scores—notably, some with spending scores above 95. This suggests a strong relationship between income, loyalty, and value, and presents a high-value segment worth prioritizing in retention and upsell strategies.

In [ ]:

In [41]:
```
plt.figure(figsize=(12, 6))
sns.barplot(
    data=star_breakdown,
    x='income_group',
    y='customer_count',
    hue='gender'
)
plt.title('⭐ Star Customer Count by Income Group and Gender')
plt.xlabel('Income Group')
plt.ylabel('Star Customer Count')
plt.legend(title='Gender')
plt.tight_layout()
plt.show()
```

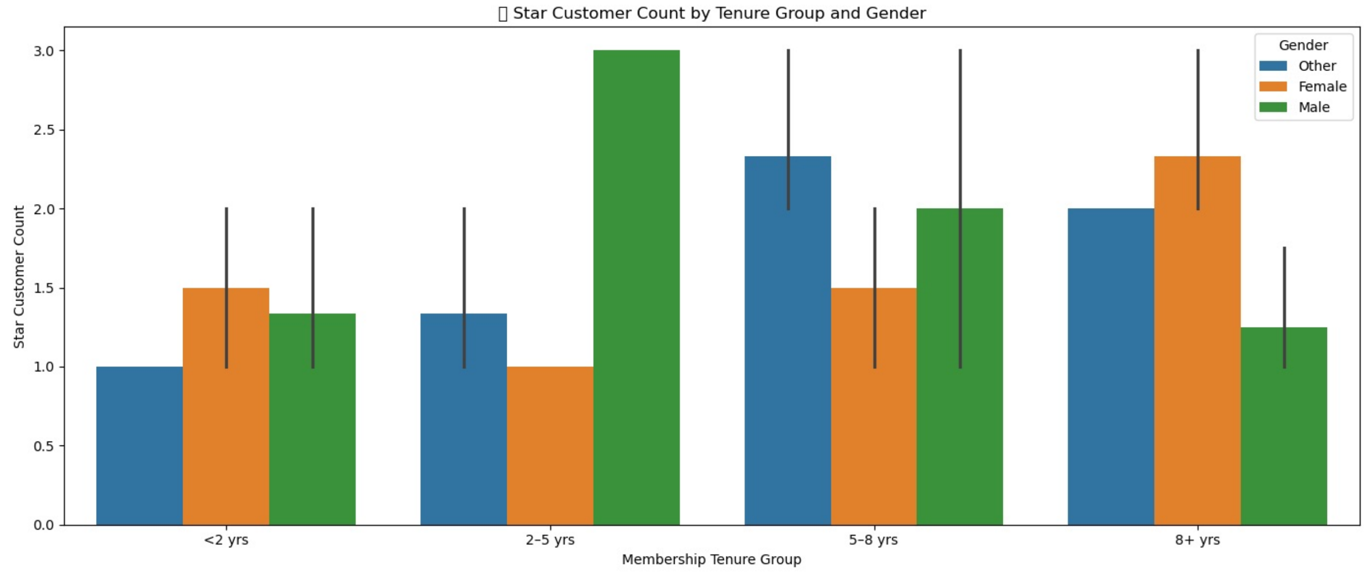
```
/var/folders/66/5v7r_k0d7dq4xv5h9vk2t8nc0000gn/T/ipykernel_20644/3879070706.py:12: UserWarning: Glyph 11088 (\N{
WHITE MEDIUM STAR}) missing from font(s) DejaVu Sans.
  plt.tight_layout()
/opt/anaconda3/lib/python3.13/site-packages/IPython/core/pylabtools.py:170: UserWarning: Glyph 11088 (\N{WHITE M
EDIUM STAR}) missing from font(s) DejaVu Sans.
  fig.canvas.print_figure(bytes_io, **kw)
```



Star customers in the highest income bracket, 100K and above, are the most represented across all genders, with male customers leading slightly. This suggests that higher income is a strong indicator of customer loyalty, regardless of gender. Income groups below 80K show more balanced but lower star customer counts, indicating that long-term loyalty may be more concentrated in top-earning segments.

In [42]:
```
plt.figure(figsize=(14, 6))
sns.barplot(
    data=star_breakdown,
    x='tenure_group',
    y='customer_count',
    hue='gender',
    dodge=True
)
plt.title('⭐ Star Customer Count by Tenure Group and Gender')
plt.xlabel('Membership Tenure Group')
plt.ylabel('Star Customer Count')
plt.legend(title='Gender')
plt.tight_layout()
plt.show()
```

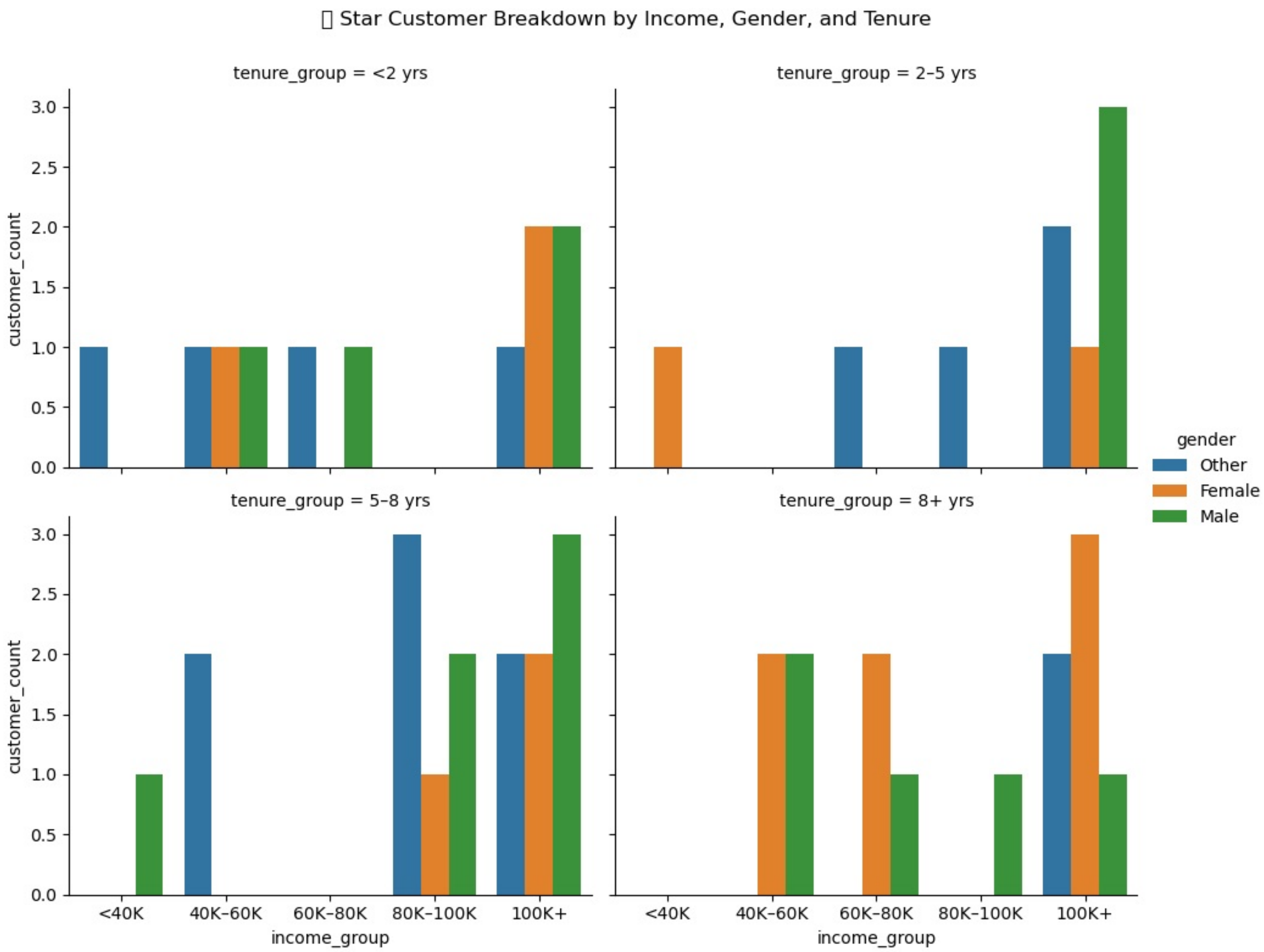
```
/var/folders/66/5v7r_k0d7dq4xv5h9vk2t8nc0000gn/T/ipykernel_20644/666399140.py:13: UserWarning: Glyph 11088 (\N{W
HITE MEDIUM STAR}) missing from font(s) DejaVu Sans.
  plt.tight_layout()
/opt/anaconda3/lib/python3.13/site-packages/IPython/core/pylabtools.py:170: UserWarning: Glyph 11088 (\N{WHITE M
EDIUM STAR}) missing from font(s) DejaVu Sans.
  fig.canvas.print_figure(bytes_io, **kw)
```



In [ ]:

In [43]:
```
sns.catplot(
    data=star_breakdown,
    x='income_group',
    y='customer_count',
    hue='gender',
    col='tenure_group',
    kind='bar',
    col_wrap=2,
    height=4,
    aspect=1.2
)
plt.subplots_adjust(top=0.9)
plt.suptitle('⭑ Star Customer Breakdown by Income, Gender, and Tenure')
plt.show()
```

```
/opt/anaconda3/lib/python3.13/site-packages/IPython/core/pylabtools.py:170: UserWarning: Glyph 11088 (\N{WHITE MEDIUM STAR}) missing from font(s) DejaVu Sans.
  fig.canvas.print_figure(bytes_io, **kw)
```



This chart shows that long-tenured star customers, with eight or more years of membership, are primarily female and earn over 100K. These individuals are strong candidates for high-tier loyalty programs and retention efforts. In contrast, newer star customers with less than two years of tenure are spread more evenly across income groups and genders, highlighting an opportunity to build loyalty early through targeted engagement.

In [17]:

```
print("Available columns:\n", customer.columns.tolist())

# Create 'Total_Spend' if the relevant columns exist
if 'Spend_Category1' in customer.columns and 'Spend_Category2' in customer.columns:
    customer['Total_Spend'] = customer['Spend_Category1'] + customer['Spend_Category2']
    print("\n 'Total_Spend' feature created.")
else:
    print("\n Spend_Category1 or Spend_Category2 not found.")

# Create 'Age_Group' if 'Age' exists
if 'Age' in customer.columns:
    customer['Age_Group'] = pd.cut(customer['Age'],
                                   bins=[0, 25, 35, 50, 70, 100],
                                   labels=['<25', '25-35', '35-50', '50-70', '70+'])
    print("'Age_Group' feature created.")
else:
    print("'Age' column not found.")

# Display the updated DataFrame with new features
display(customer[['Age', 'Age_Group']] if 'Age' in customer.columns else customer.head())
if 'Total_Spend' in customer.columns:
    display(customer[['Total_Spend']].head())
```

```
Available columns:
 ['id', 'age', 'gender', 'income', 'spending_score', 'membership_years', 'purchase_frequency', 'preferred_catego
ry', 'last_purchase_amount']

 Spend_Category1 or Spend_Category2 not found.
'Age' column not found.
```

	id	age	gender	income	spending_score	membership_years	purchase_frequency	preferred_category	last_purchase_amount
0	1	38	Female	99342	90	3	24	Groceries	113.53
1	2	21	Female	78852	60	2	42	Sports	41.93
2	3	60	Female	126573	30	2	28	Clothing	424.36
3	4	40	Other	47099	74	9	5	Home & Garden	991.93
4	5	65	Female	140621	21	3	25	Electronics	347.08

In [43]:

```
customer = df.copy()

# 1. Age Group
customer['age_group'] = pd.cut(customer['age'],
                               bins=[0, 25, 35, 50, 70, 100],
                               labels=['<25', '25-35', '35-50', '50-70', '70+'])

# 2. Income Group
customer['income_group'] = pd.cut(customer['income'],
                                  bins=[0, 40000, 80000, 120000, 160000, float('inf')],
                                  labels=['Low', 'Lower-Mid', 'Mid', 'Upper-Mid', 'High'])

# 3. Spending Score Category
customer['spending_category'] = pd.cut(customer['spending_score'],
                                       bins=[0, 30, 60, 100],
                                       labels=['Low', 'Medium', 'High'])

# 4. High Spender Flag (based on last_purchase_amount)
threshold = customer['last_purchase_amount'].quantile(0.75)
customer['high_value_customer'] = (customer['last_purchase_amount'] > threshold).astype(int)

# 5. Gender Binary Flag
customer['is_female'] = customer['gender'].apply(lambda x: 1 if str(x).strip().lower() == 'female' else 0)

# 6. Interaction: Income × Spending
customer['income_spend_interaction'] = customer['income'] * customer['spending_score']

# 7. Loyalty Duration Bucket
customer['loyalty_level'] = pd.cut(customer['membership_years'],
                                  bins=[0, 2, 5, 10, float('inf')],
                                  labels=['New', 'Established', 'Loyal', 'Veteran'])

# Final feature list preview
engineered_cols = ['age_group', 'income_group', 'spending_category', 'high_value_customer',
                   'is_female', 'income_spend_interaction', 'loyalty_level']
```

```
customer[engineered_cols].head()
```

Out[43]:

	age_group	income_group	spending_category	high_value_customer	is_female	income_spend_interaction	loyalty_level
0	35-50	Mid	High	0	1	8940780	Established
1	<25	Lower-Mid	Medium	0	1	4731120	New
2	50-70	Upper-Mid	Low	0	1	3797190	New
3	35-50	Lower-Mid	High	1	0	3485326	Loyal
4	50-70	Upper-Mid	Low	0	1	2953041	Established

In [47]:
```
# Assuming recent = frequent purchases
customer['recency_level'] = pd.cut(customer['purchase_frequency'],
                                   bins=[0, 5, 15, 30, 50, float('inf')],
                                   labels=['Very Low', 'Low', 'Moderate', 'High', 'Very High'])
```

In [53]:
```
# Top categories by count
top_cats = customer['preferred_category'].value_counts().nlargest(3).index.tolist()
customer['top_category_loyal'] = customer['preferred_category'].apply(lambda x: 1 if x in top_cats else 0)
```

In [55]:
```
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()
customer['value_score'] = scaler.fit_transform(
    customer[['last_purchase_amount']]) + scaler.fit_transform(customer[['purchase_frequency']])
```

In [57]:
```
# Flag potential churners (low tenure + low spend)
customer['churn_risk'] = ((customer['membership_years'] < 2) &
                          (customer['spending_score'] < 30)).astype(int)
```

In [59]:
```
# Normalize all and sum
to_scale = ['income', 'spending_score', 'membership_years', 'purchase_frequency']
customer_scaled = scaler.fit_transform(customer[to_scale])
customer['engagement_index'] = customer_scaled.sum(axis=1)
```

In [65]:
```
from sklearn.preprocessing import MinMaxScaler

# Step 1: Recency Bucket from purchase frequency
print("Step 1: Creating 'recency_level'...")
customer['recency_level'] = pd.cut(customer['purchase_frequency'],
                                   bins=[0, 5, 15, 30, 50, float('inf')],
                                   labels=['Very Low', 'Low', 'Moderate', 'High', 'Very High'])
print("'recency_level' created.")
display(customer[['purchase_frequency', 'recency_level']].head())

# Step 2: Top Category Loyalty Flag
print("\nStep 2: Creating 'top_category_loyal'...")
top_cats = customer['preferred_category'].value_counts().nlargest(3).index.tolist()
customer['top_category_loyal'] = customer['preferred_category'].apply(lambda x: 1 if x in top_cats else 0)
print("'top_category_loyal' flag created.")
display(customer[['preferred_category', 'top_category_loyal']].head())

# Step 3: Value Score (normalized last purchase + frequency)
print("\nStep 3: Creating 'value_score'...")
scaler = MinMaxScaler()
customer['value_score'] = (
    scaler.fit_transform(customer[['last_purchase_amount']]) +
    scaler.fit_transform(customer[['purchase_frequency']])
)
print("'value_score' created.")
display(customer[['last_purchase_amount', 'purchase_frequency', 'value_score']].head())

# Step 4: Churn Risk Flag
print("\nStep 4: Creating 'churn_risk'...")
customer['churn_risk'] = ((customer['membership_years'] < 2) &
                          (customer['spending_score'] < 30)).astype(int)
print("'churn_risk' flag created.")
display(customer[['membership_years', 'spending_score', 'churn_risk']].head())

# Step 5: Engagement Index
print("\nStep 5: Creating 'engagement_index'...")
to_scale = ['income', 'spending_score', 'membership_years', 'purchase_frequency']
engagement_scaled = scaler.fit_transform(customer[to_scale])
customer['engagement_index'] = engagement_scaled.sum(axis=1)
print(" 'engagement_index' created.")
display(customer[to_scale + ['engagement_index']].head())
```

```
Step 1: Creating 'recency_level'...
'recency_level' created.
```

	purchase_frequency	recency_level
0	24	Moderate
1	42	High
2	28	Moderate
3	5	Very Low
4	25	Moderate

```
Step 2: Creating 'top_category_loyal'...
'top_category_loyal' flag created.
```

	preferred_category	top_category_loyal
0	Groceries	0
1	Sports	1
2	Clothing	0
3	Home & Garden	1
4	Electronics	1

```
Step 3: Creating 'value_score'...
'value_score' created.
```

	last_purchase_amount	purchase_frequency	value_score
0	113.53	24	0.573629
1	41.93	42	0.868604
2	424.36	28	0.969441
3	991.93	5	1.073739
4	347.08	25	0.830104

```
Step 4: Creating 'churn_risk'...
'churn_risk' flag created.
```

	membership_years	spending_score	churn_risk
0	3	90	0
1	2	60	0
2	2	30	0
3	9	74	0
4	3	21	0

```
Step 5: Creating 'engagement_index'...
 'engagement_index' created.
```

	income	spending_score	membership_years	purchase_frequency	engagement_index
0	99342	90	3	24	2.168566
1	78852	60	2	42	1.950977
2	126573	30	2	28	1.760010
3	47099	74	9	5	1.850390
4	140621	21	3	25	1.836085

In [69]:

```python
# Select numeric columns (excluding ID if still present)
numerical_cols = customer.select_dtypes(include=['int64', 'float64']).columns.tolist()

# Optionally drop identifier columns
numerical_cols = [col for col in numerical_cols if col != 'id']

print("Numerical columns to transform and scale:")
print(numerical_cols)
```

```
Numerical columns to transform and scale:
['age', 'income', 'spending_score', 'membership_years', 'purchase_frequency', 'last_purchase_amount', 'is_female
', 'income_spend_interaction', 'top_category_loyal', 'value_score', 'engagement_index']
```

In [71]:

```python
import numpy as np

# Calculate skewness
skewed = customer[numerical_cols].skew().sort_values(ascending=False)
print("Skewness:\n", skewed)

# Apply log1p to highly skewed columns (e.g., skewness > 1)
for col in skewed.index:
```

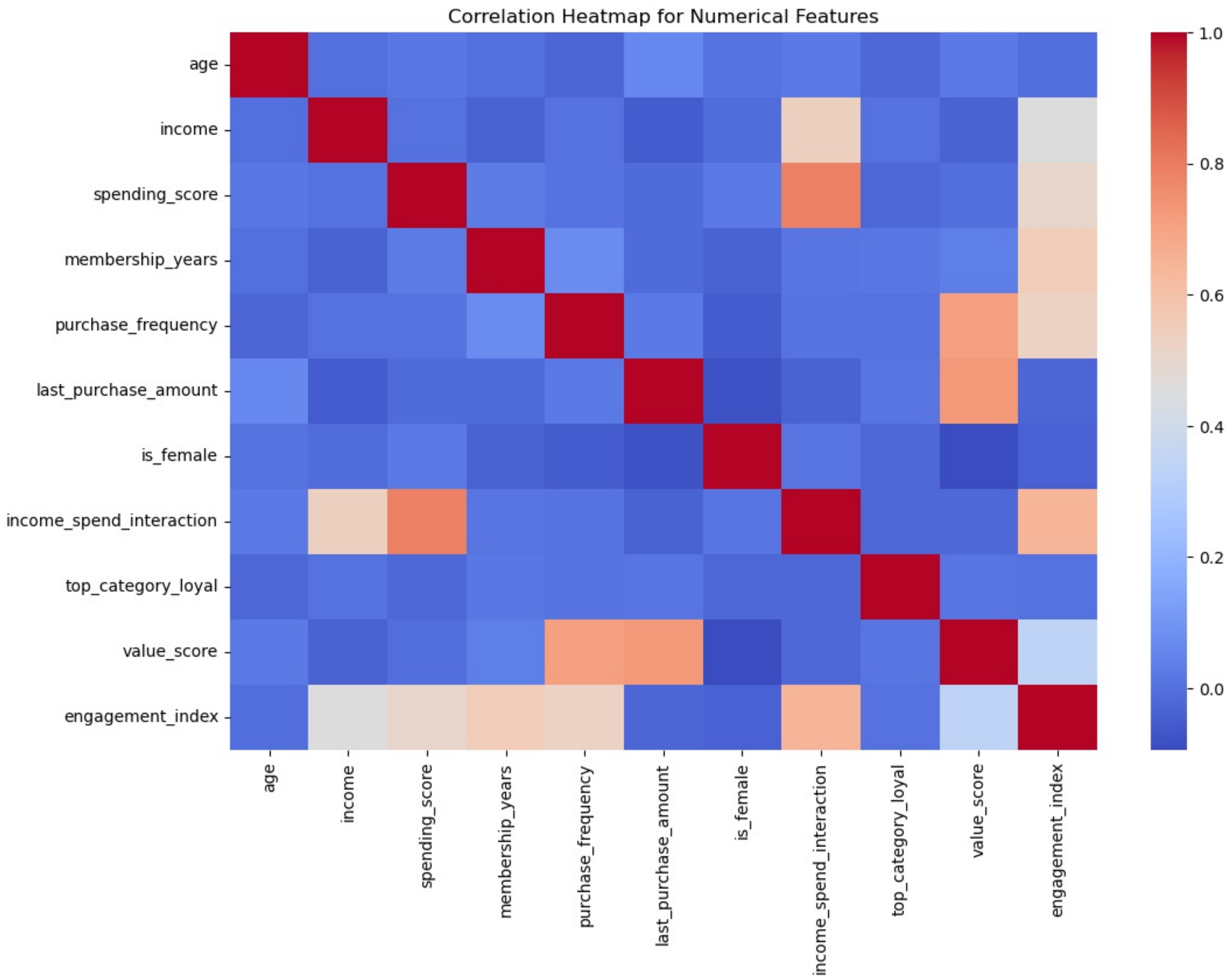
```python
    if skewed[col] > 1:
        customer[f'log_{col}'] = np.log1p(customer[col])
        print(f"Applied log1p transform to: {col}")
```

```
Skewness:
 income_spend_interaction    0.815022
is_female                   0.792736
engagement_index            0.119160
income                      0.051065
membership_years            0.029844
last_purchase_amount        0.017554
spending_score             -0.016577
value_score                -0.044177
age                        -0.046000
purchase_frequency         -0.083966
top_category_loyal         -0.543783
dtype: float64
```

In [73]:
```python
from sklearn.preprocessing import StandardScaler

# Choose original + transformed columns
to_scale = [col for col in customer.columns if col in numerical_cols or col.startswith('log_')]

# Scale them
scaler = StandardScaler()
customer_scaled = customer.copy()
customer_scaled[to_scale] = scaler.fit_transform(customer_scaled[to_scale])

print("StandardScaler applied.")
```

```
StandardScaler applied.
```

In [75]:
```python
import seaborn as sns
import matplotlib.pyplot as plt

plt.figure(figsize=(12,8))
sns.heatmap(customer_scaled[to_scale].corr(), annot=False, cmap='coolwarm')
plt.title("Correlation Heatmap for Numerical Features")
plt.show()
```



In [77]:
```python
from scipy.stats import shapiro
```

```
stat, p = shapiro(customer['income'])
print(f"Shapiro-Wilk test for 'income': p-value = {p:.4f}")
```

```
Shapiro-Wilk test for 'income': p-value = 0.0000
```

In [79]:
```
import numpy as np
customer['log_income'] = np.log1p(customer['income'])
```

In [81]:
```
from sklearn.preprocessing import StandardScaler

# Select numerical features (including transformed income)
to_scale = ['log_income', 'spending_score', 'membership_years',
            'purchase_frequency', 'last_purchase_amount']

# Apply StandardScaler
scaler = StandardScaler()
customer_scaled = customer.copy()
customer_scaled[to_scale] = scaler.fit_transform(customer_scaled[to_scale])

print("Scaled features:")
display(customer_scaled[to_scale].head())
```

```
Scaled features:
```

	log_income	spending_score	membership_years	purchase_frequency	last_purchase_amount
0	0.465366	1.358468	-0.865010	-0.182348	-1.281540
1	-0.066938	0.321865	-1.215358	1.082005	-1.523763
2	1.023607	-0.714738	-1.215358	0.098620	-0.230005
3	-1.254432	0.805613	1.237080	-1.516943	1.690080
4	1.266143	-1.025718	-0.865010	-0.112106	-0.491443

In [83]:
```
customer_scaled[to_scale].describe()
```

Out[83]:

	log_income	spending_score	membership_years	purchase_frequency	last_purchase_amount
count	1.000000e+03	1.000000e+03	1.000000e+03	1.000000e+03	1.000000e+03
mean	4.174439e-17	-7.815970e-17	-1.145750e-16	-1.065814e-17	-9.681145e-17
std	1.000500e+00	1.000500e+00	1.000500e+00	1.000500e+00	1.000500e+00
min	-2.293514e+00	-1.716787e+00	-1.565707e+00	-1.797910e+00	-1.630428e+00
25%	-7.781918e-01	-8.529512e-01	-8.650101e-01	-8.145243e-01	-9.255398e-01
50%	1.819511e-01	-2.366909e-02	-1.643134e-01	2.837770e-02	-2.549659e-03
75%	8.247862e-01	8.747199e-01	8.867317e-01	8.712797e-01	8.620585e-01
max	1.414517e+00	1.704002e+00	1.587428e+00	1.643940e+00	1.716501e+00