Customer Segmentation with K - Means Clustering

Project Objective

```
The goal of this project is to perform customer segmentation
using a dataset of customer demographics, spending patterns, and
purchase behavior.
By applying clustering techniques, we aim to identify distinct
behavioral personas that can help businesses better understand
their customers and design targeted marketing strategies.
```

Import Libraries

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
```

In [2]:
```
df = pd.read_csv("marketing_campaign.csv", sep="\t")
df
```

Out[2]:

	ID	Year_Birth	Education	Marital_Status	Income	Kidhome	Teenhome	Dt_C
0	5524	1957	Graduation	Single	58138.0	0	0	04
1	2174	1954	Graduation	Single	46344.0	1	1	08
2	4141	1965	Graduation	Together	71613.0	0	0	21
3	6182	1984	Graduation	Together	26646.0	1	0	10
4	5324	1981	PhD	Married	58293.0	1	0	19
...	...	...	...	...	...	...	...	...
2235	10870	1967	Graduation	Married	61223.0	0	1	13
2236	4001	1946	PhD	Together	64014.0	2	1	10
2237	7270	1981	Graduation	Divorced	56981.0	0	0	25
2238	8235	1956	Master	Together	69245.0	0	1	24
2239	9405	1954	PhD	Married	52869.0	1	1	15

2240 rows × 29 columns

Data Preprocessing

In [3]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2240 entries, 0 to 2239
Data columns (total 29 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   ID                   2240 non-null   int64  
 1   Year_Birth           2240 non-null   int64  
 2   Education            2240 non-null   object 
 3   Marital_Status       2240 non-null   object 
 4   Income               2216 non-null   float64
 5   Kidhome              2240 non-null   int64  
 6   Teenhome             2240 non-null   int64  
 7   Dt_Customer          2240 non-null   object 
 8   Recency              2240 non-null   int64  
 9   MntWines             2240 non-null   int64  
 10  MntFruits            2240 non-null   int64  
 11  MntMeatProducts      2240 non-null   int64  
 12  MntFishProducts      2240 non-null   int64  
 13  MntSweetProducts     2240 non-null   int64  
 14  MntGoldProds         2240 non-null   int64  
 15  NumDealsPurchases    2240 non-null   int64  
 16  NumWebPurchases      2240 non-null   int64  
 17  NumCatalogPurchases  2240 non-null   int64  
 18  NumStorePurchases    2240 non-null   int64  
 19  NumWebVisitsMonth    2240 non-null   int64  
 20  AcceptedCmp3         2240 non-null   int64  
 21  AcceptedCmp4         2240 non-null   int64  
 22  AcceptedCmp5         2240 non-null   int64  
 23  AcceptedCmp1         2240 non-null   int64  
 24  AcceptedCmp2         2240 non-null   int64  
 25  Complain             2240 non-null   int64  
 26  Z_CostContact        2240 non-null   int64  
 27  Z_Revenue            2240 non-null   int64  
 28  Response             2240 non-null   int64  
dtypes: float64(1), int64(25), object(3)
memory usage: 507.6+ KB
```

In [4]:
```
df.isnull().sum()
```

Out[4]:
```
ID                      0
Year_Birth              0
Education               0
Marital_Status          0
Income                 24
Kidhome                 0
Teenhome                0
Dt_Customer             0
Recency                 0
MntWines                0
MntFruits               0
MntMeatProducts         0
MntFishProducts         0
MntSweetProducts        0
MntGoldProds            0
NumDealsPurchases       0
NumWebPurchases         0
NumCatalogPurchases     0
NumStorePurchases       0
NumWebVisitsMonth       0
AcceptedCmp3            0
AcceptedCmp4            0
AcceptedCmp5            0
AcceptedCmp1            0
AcceptedCmp2            0
Complain                0
Z_CostContact           0
Z_Revenue               0
Response                0
dtype: int64
```

In [5]:
```
df.duplicated().sum()
```

Out[5]:
```
np.int64(0)
```

```
The missing values in the income column represent about 1% of
the dataset, so dropping them
will not significantly reduce the data size.
```

In [6]:
```
df = df.dropna()
```

In [7]:
```
df.nunique()
```

Out[7]:
```
ID                     2216
Year_Birth               59
Education                 5
Marital_Status            8
Income                 1974
Kidhome                   3
Teenhome                  3
Dt_Customer             662
Recency                 100
MntWines                776
MntFruits               158
MntMeatProducts         554
MntFishProducts         182
MntSweetProducts        176
MntGoldProds            212
NumDealsPurchases        15
NumWebPurchases          15
NumCatalogPurchases      14
NumStorePurchases        14
NumWebVisitsMonth        16
AcceptedCmp3              2
AcceptedCmp4              2
AcceptedCmp5              2
AcceptedCmp1              2
AcceptedCmp2              2
Complain                  2
Z_CostContact             1
Z_Revenue                 1
Response                  2
dtype: int64
```

The columns Z_CostContact and Z_Revenue have only 1 unique value, meaning they do not provide any useful information for analysis or modeling. Therefore, we can safely remove them from the dataset.

In [8]:
```
df = df.drop(columns=['Z_CostContact', 'Z_Revenue'])
```

The Dt_Customer column was converted to datetime using .loc to avoid pandas warnings, ensuring proper handling of date operations.

In [9]:
```
df['Dt_Customer'] = pd.to_datetime(df['Dt_Customer'].astype(str), errors='coerce
```

In [10]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 2216 entries, 0 to 2239
Data columns (total 27 columns):
 #   Column               Non-Null Count  Dtype         
---  ------               --------------  -----         
 0   ID                   2216 non-null   int64         
 1   Year_Birth           2216 non-null   int64         
 2   Education            2216 non-null   object        
 3   Marital_Status       2216 non-null   object        
 4   Income               2216 non-null   float64       
 5   Kidhome              2216 non-null   int64         
 6   Teenhome             2216 non-null   int64         
 7   Dt_Customer          2216 non-null   datetime64[ns]
 8   Recency              2216 non-null   int64         
 9   MntWines             2216 non-null   int64         
 10  MntFruits            2216 non-null   int64         
 11  MntMeatProducts      2216 non-null   int64         
 12  MntFishProducts      2216 non-null   int64         
 13  MntSweetProducts     2216 non-null   int64         
 14  MntGoldProds         2216 non-null   int64         
 15  NumDealsPurchases    2216 non-null   int64         
 16  NumWebPurchases      2216 non-null   int64         
 17  NumCatalogPurchases  2216 non-null   int64         
 18  NumStorePurchases    2216 non-null   int64         
 19  NumWebVisitsMonth    2216 non-null   int64         
 20  AcceptedCmp3         2216 non-null   int64         
 21  AcceptedCmp4         2216 non-null   int64         
 22  AcceptedCmp5         2216 non-null   int64         
 23  AcceptedCmp1         2216 non-null   int64         
 24  AcceptedCmp2         2216 non-null   int64         
 25  Complain             2216 non-null   int64         
 26  Response             2216 non-null   int64         
dtypes: datetime64[ns](1), float64(1), int64(23), object(2)
memory usage: 484.8+ KB
```

In [11]:
```python
newest_date = df['Dt_Customer'].max()
oldest_date = df['Dt_Customer'].min()

print(f"The newest customer's enrolment date in the records: {newest_date.date()
print(f"The oldest customer's enrolment date in the records: {oldest_date.date()
```

```
The newest customer's enrolment date in the records: 2014-06-29
The oldest customer's enrolment date in the records: 2012-07-30
```

The Customer_Tenure column was created by calculating the number of days between each customer's enrolment date and the newest enrolment date in the records.

In [12]:
```python
reference_date = df['Dt_Customer'].max()
df.loc[:, 'Customer_Tenure'] = (reference_date - df['Dt_Customer']).dt.days
```

In [13]:
```python
df['Marital_Status'].value_counts()
```

Out[13]:
```
Marital_Status
Married     857
Together    573
Single      471
Divorced    232
Widow        76
Alone         3
Absurd        2
YOLO          2
Name: count, dtype: int64
```

The 'Alone' category in Marital_Status will be replaced with 'Single' to consolidate similar categories and reduce noise in the clustering process.

In [14]:
```
df.loc[:,'Marital_Status'] = df['Marital_Status'].replace({'Alone': 'Single'})
```

Rows with rare and inconsistent categories 'Absurd' and 'YOLO' in Marital_Status will be removed, as they represent a negligible portion of the dataset

In [15]:
```
df=df[~df['Marital_Status'].isin(['Absurd', 'YOLO'])]
```

In [16]:
```
df['Marital_Status'].value_counts()
```

Out[16]:
```
Marital_Status
Married     857
Together    573
Single      474
Divorced    232
Widow        76
Name: count, dtype: int64
```

In [17]:
```
df['Education'].value_counts()
```

Out[17]:
```
Education
Graduation    1115
PhD            479
Master         364
2n Cycle       200
Basic           54
Name: count, dtype: int64
```

The Age column will be created by subtracting Year_Birth from 2025, providing the age of each customer for further analysis and clustering.

In [18]:
```
df.loc[:, 'Age'] = 2025 - df['Year_Birth']
```

In [19]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 2212 entries, 0 to 2239
Data columns (total 29 columns):
 #   Column               Non-Null Count  Dtype         
---  ------               --------------  -----         
 0   ID                   2212 non-null   int64         
 1   Year_Birth           2212 non-null   int64         
 2   Education            2212 non-null   object        
 3   Marital_Status       2212 non-null   object        
 4   Income               2212 non-null   float64       
 5   Kidhome              2212 non-null   int64         
 6   Teenhome             2212 non-null   int64         
 7   Dt_Customer          2212 non-null   datetime64[ns]
 8   Recency              2212 non-null   int64         
 9   MntWines             2212 non-null   int64         
 10  MntFruits            2212 non-null   int64         
 11  MntMeatProducts      2212 non-null   int64         
 12  MntFishProducts      2212 non-null   int64         
 13  MntSweetProducts     2212 non-null   int64         
 14  MntGoldProds         2212 non-null   int64         
 15  NumDealsPurchases    2212 non-null   int64         
 16  NumWebPurchases      2212 non-null   int64         
 17  NumCatalogPurchases  2212 non-null   int64         
 18  NumStorePurchases    2212 non-null   int64         
 19  NumWebVisitsMonth    2212 non-null   int64         
 20  AcceptedCmp3         2212 non-null   int64         
 21  AcceptedCmp4         2212 non-null   int64         
 22  AcceptedCmp5         2212 non-null   int64         
 23  AcceptedCmp1         2212 non-null   int64         
 24  AcceptedCmp2         2212 non-null   int64         
 25  Complain             2212 non-null   int64         
 26  Response             2212 non-null   int64         
 27  Customer_Tenure      2212 non-null   int64         
 28  Age                  2212 non-null   int64         
dtypes: datetime64[ns](1), float64(1), int64(25), object(2)
memory usage: 518.4+ KB
```

In [20]:
```
df['Kidhome'].value_counts()
```

Out[20]:
```
Kidhome
0    1279
1     887
2      46
Name: count, dtype: int64
```

In [21]:
```
df['Teenhome'].value_counts()
```

Out[21]:
```
Teenhome
0    1145
1    1016
2      51
Name: count, dtype: int64
```

The Children column will be created by summing Kidhome and Teenhome, providing the total number of children in each customer's household.

In [22]:
```
df.loc[:, 'Children'] = df['Kidhome'] + df['Teenhome']
```

In [23]:
```
df['Children'].value_counts()
```

Out[23]:
```
Children
1    1115
0     631
2     416
3      50
Name: count, dtype: int64
```

The Children column shows that most customers have 0 or 1 child/teen, while a smaller portion has 2 or 3, representing the total children per household.

In [24]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 2212 entries, 0 to 2239
Data columns (total 30 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   ID                   2212 non-null   int64
 1   Year_Birth           2212 non-null   int64
 2   Education            2212 non-null   object
 3   Marital_Status       2212 non-null   object
 4   Income               2212 non-null   float64
 5   Kidhome              2212 non-null   int64
 6   Teenhome             2212 non-null   int64
 7   Dt_Customer          2212 non-null   datetime64[ns]
 8   Recency              2212 non-null   int64
 9   MntWines             2212 non-null   int64
 10  MntFruits            2212 non-null   int64
 11  MntMeatProducts      2212 non-null   int64
 12  MntFishProducts      2212 non-null   int64
 13  MntSweetProducts     2212 non-null   int64
 14  MntGoldProds         2212 non-null   int64
 15  NumDealsPurchases    2212 non-null   int64
 16  NumWebPurchases      2212 non-null   int64
 17  NumCatalogPurchases  2212 non-null   int64
 18  NumStorePurchases    2212 non-null   int64
 19  NumWebVisitsMonth    2212 non-null   int64
 20  AcceptedCmp3         2212 non-null   int64
 21  AcceptedCmp4         2212 non-null   int64
 22  AcceptedCmp5         2212 non-null   int64
 23  AcceptedCmp1         2212 non-null   int64
 24  AcceptedCmp2         2212 non-null   int64
 25  Complain             2212 non-null   int64
 26  Response             2212 non-null   int64
 27  Customer_Tenure      2212 non-null   int64
 28  Age                  2212 non-null   int64
 29  Children             2212 non-null   int64
dtypes: datetime64[ns](1), float64(1), int64(26), object(2)
memory usage: 535.7+ KB
```

In [25]:
```
df.describe()
```

Out[25]:

	ID	Year_Birth	Income	Kidhome	Teenhome	Dt_Cu
count	2212.000000	2212.000000	2212.000000	2212.000000	2212.000000	
mean	5587.731917	1968.811031	52232.510850	0.442586	0.505425	2013 17:28:06.075
min	0.000000	1893.000000	1730.000000	0.000000	0.000000	2012 0
25%	2814.750000	1959.000000	35233.500000	0.000000	0.000000	2013 0
50%	5458.500000	1970.000000	51381.500000	0.000000	0.000000	2013 0
75%	8421.750000	1977.000000	68522.000000	1.000000	1.000000	2013 0
max	11191.000000	1996.000000	666666.000000	2.000000	2.000000	2014 0
std	3247.944128	11.982065	25187.455359	0.537052	0.544258	

8 rows × 28 columns

In [26]:

```
df.nunique()
```

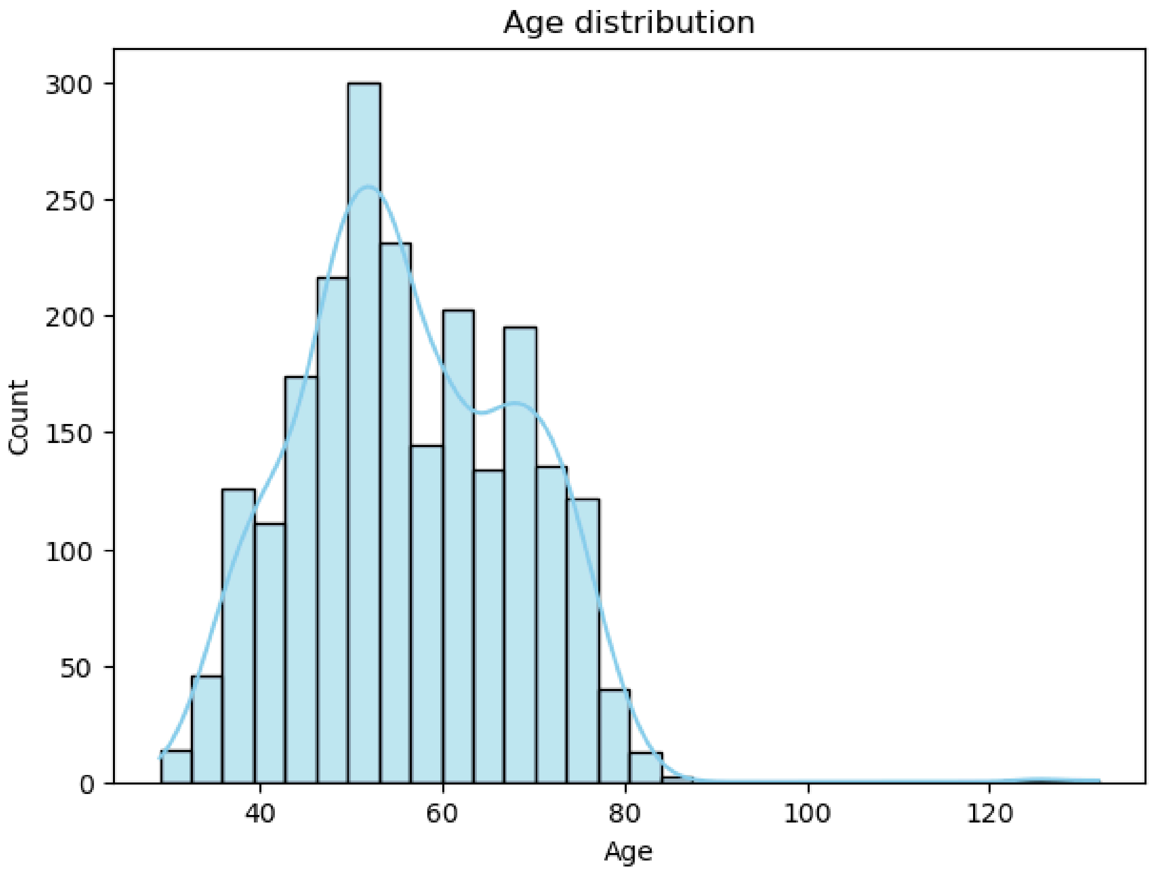
Out[26]:

```
ID                     2212
Year_Birth               59
Education                 5
Marital_Status            5
Income                 1973
Kidhome                   3
Teenhome                  3
Dt_Customer             662
Recency                 100
MntWines                776
MntFruits               158
MntMeatProducts         554
MntFishProducts         182
MntSweetProducts        176
MntGoldProds            211
NumDealsPurchases        15
NumWebPurchases          15
NumCatalogPurchases      14
NumStorePurchases        14
NumWebVisitsMonth        16
AcceptedCmp3              2
AcceptedCmp4              2
AcceptedCmp5              2
AcceptedCmp1              2
AcceptedCmp2              2
Complain                  2
Response                  2
Customer_Tenure         662
Age                      59
Children                  4
dtype: int64
```

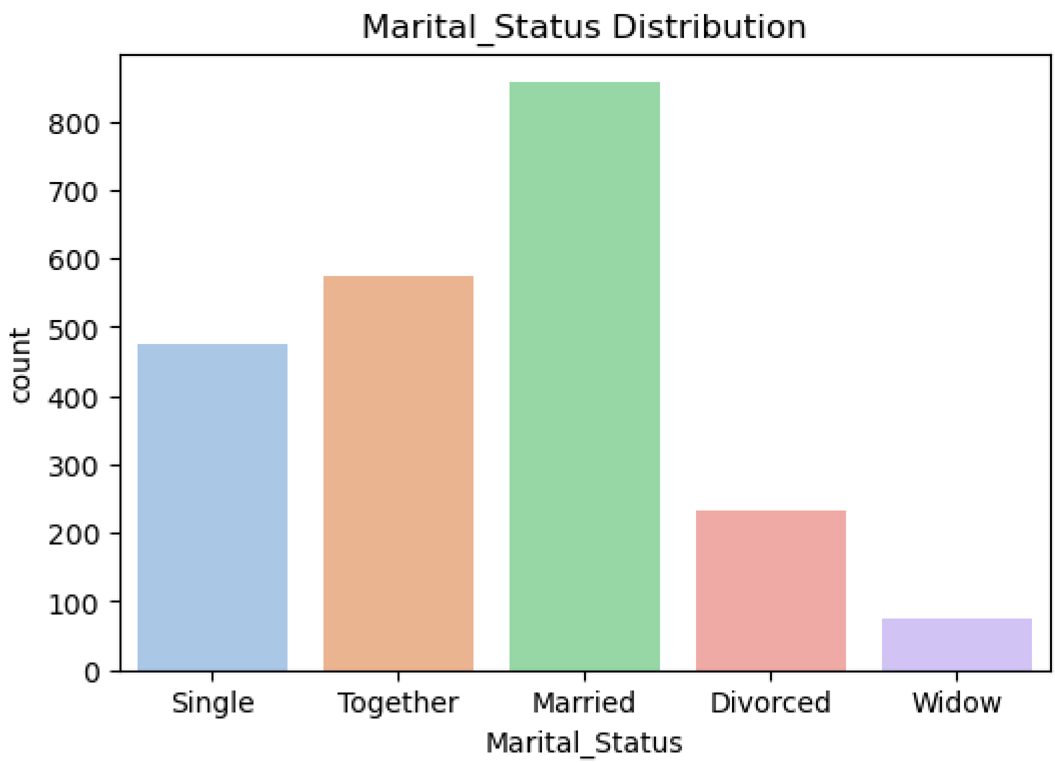
In [27]:
```
df['TotalSpent'] = df[['MntWines','MntFruits','MntMeatProducts',
                       'MntFishProducts','MntSweetProducts','MntGoldProds']].sum
```

Exploratory Data Analysis (EDA)

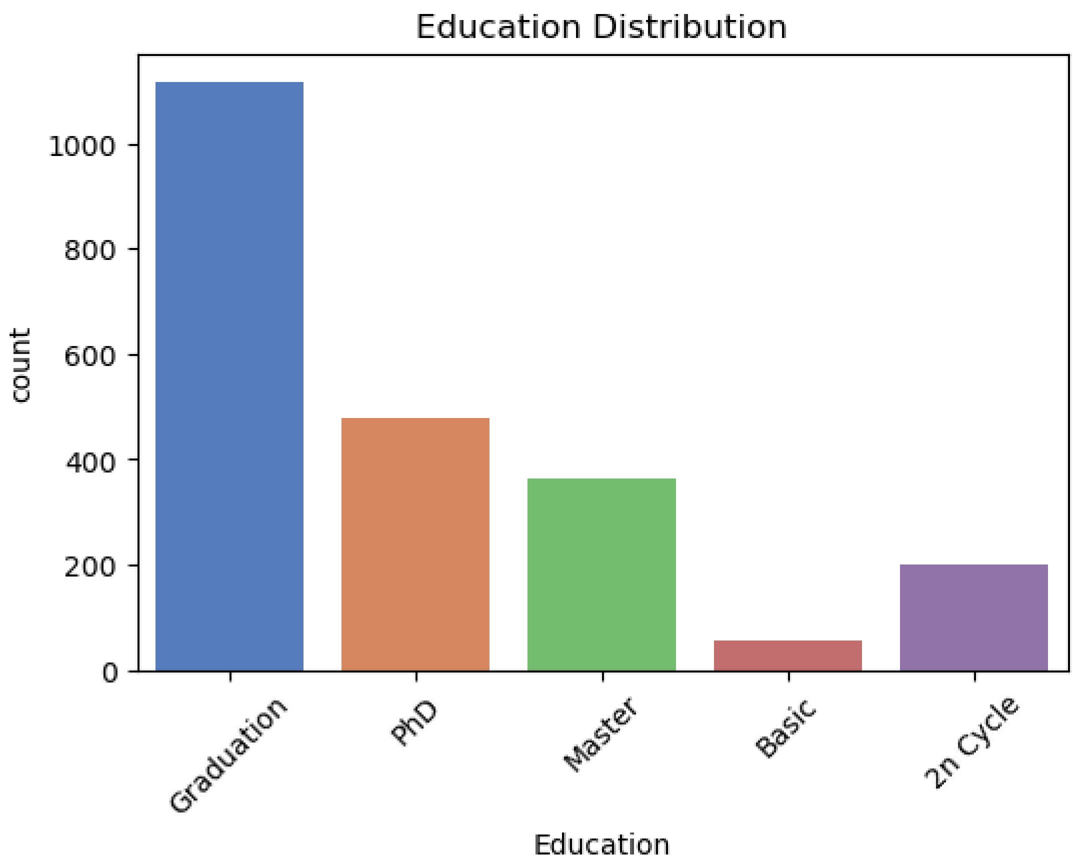
In [28]:
```
plt.figure(figsize=(7,5))
sns.histplot(df['Age'],bins=30,kde=True, color='skyblue')
plt.title('Age distribution')
plt.show()
```



In [29]:
```
plt.figure(figsize=(6,4))
sns.countplot(x='Marital_Status', data=df, palette='pastel')
plt.title('Marital_Status Distribution')
plt.show()
```

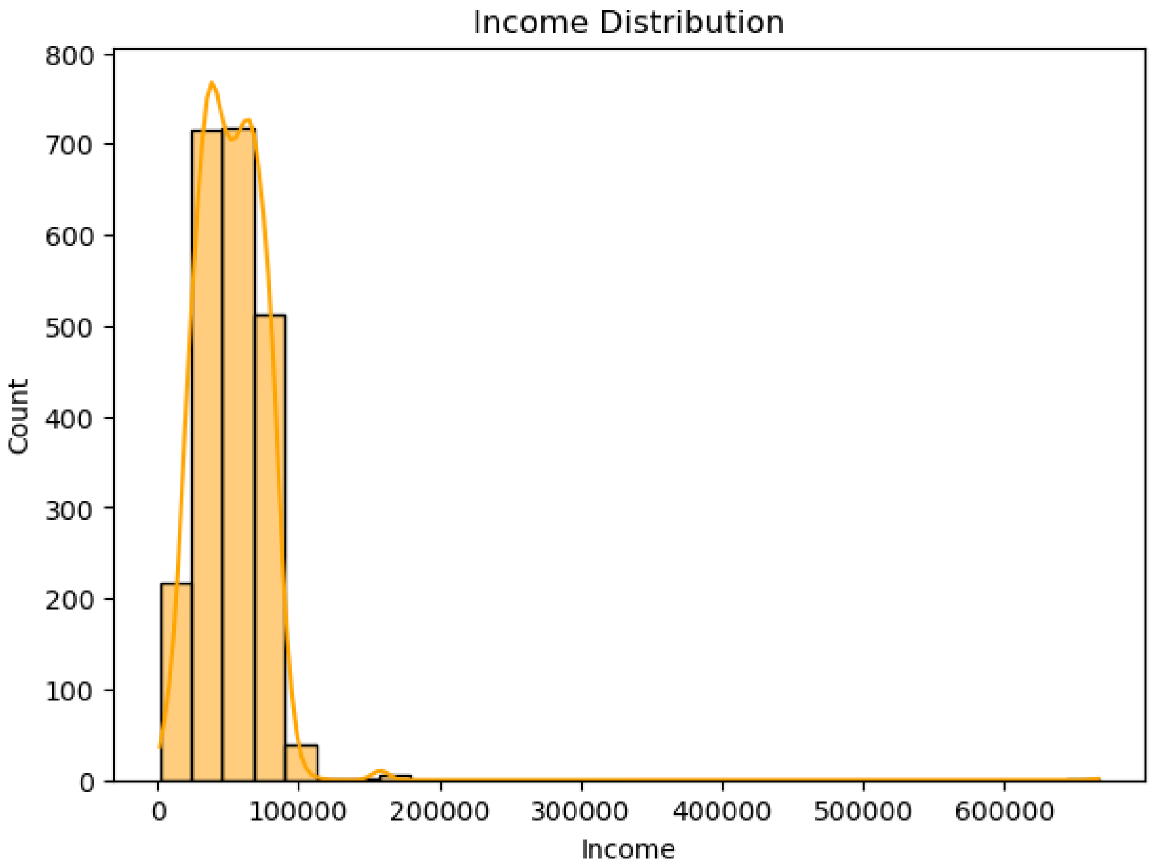


In [30]:
```
plt.figure(figsize=(6,4))
sns.countplot(x='Education', data=df, palette='muted')
plt.title('Education Distribution')
plt.xticks(rotation=45)
plt.show()
```



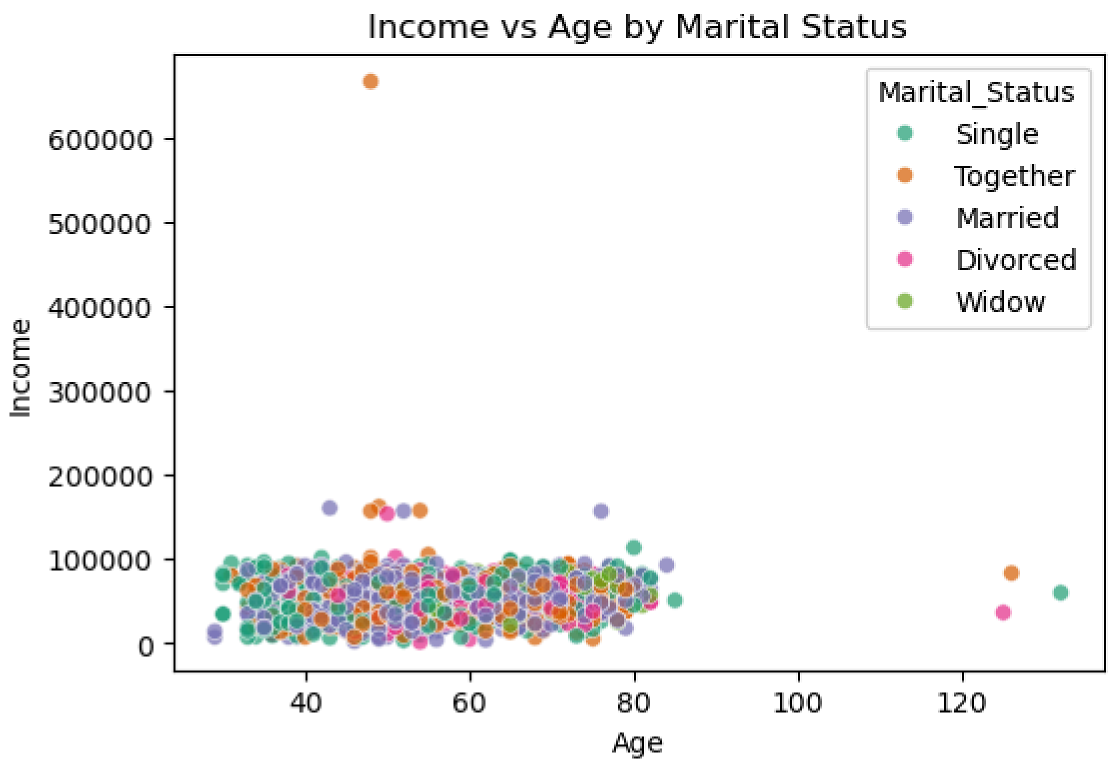
In [31]:
```
plt.figure(figsize=(7,5))
sns.histplot(df['Income'], bins=30, kde=True, color='orange')
```

```
plt.title('Income Distribution')
plt.show()
```



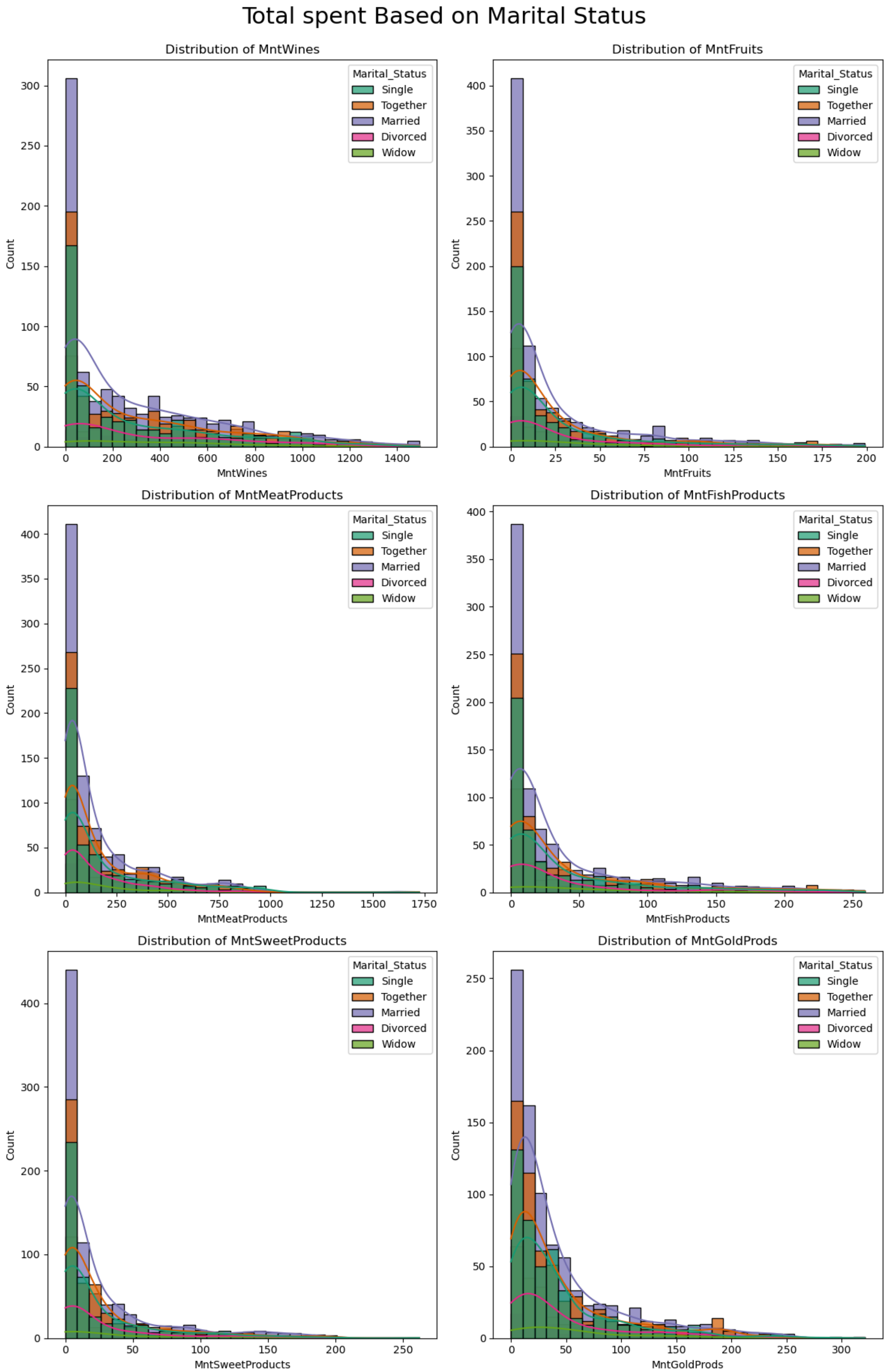
The income distribution is right-skewed, with most customers earning between 20,000 and 80,000. There are a few customers with very high incomes (over $120,000), but these are outliers. This skew suggests that while the majority of customers have moderate income, targeting high-income individuals could focus on a small, high-value segment.

In [32]:
```
plt.figure(figsize=(6,4))
sns.scatterplot(data=df, x='Age', y='Income', hue='Marital_Status', alpha=0.7, p
plt.title('Income vs Age by Marital Status')
plt.show()
```



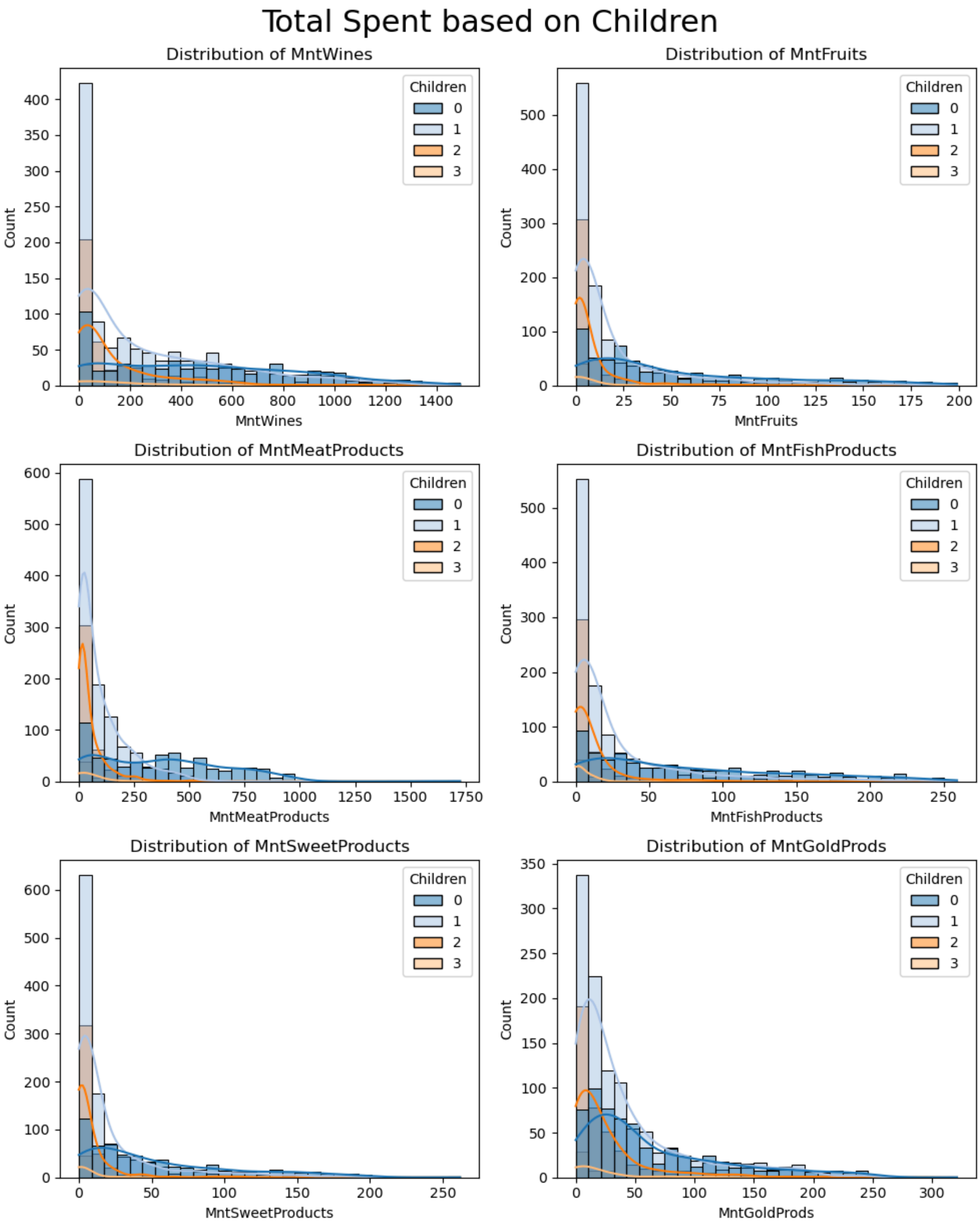
In [33]:

```
total_spent=['MntWines','MntFruits','MntMeatProducts','MntFishProducts','MntSwee

n_cols = 2
n_rows = -(-len(total_spent) // n_cols)

fig, axes = plt.subplots(n_rows, n_cols, figsize=(6 * n_cols, 6 * n_rows))
axes = axes.flatten()

for i, feature in enumerate(total_spent):
    sns.histplot(data=df, x=feature, hue='Marital_Status', kde=True, bins=30, ax
    axes[i].set_title(f"Distribution of {feature}")

plt.tight_layout()
plt.suptitle("Total spent Based on Marital Status", fontsize=22, y=1.02)
plt.show()
```



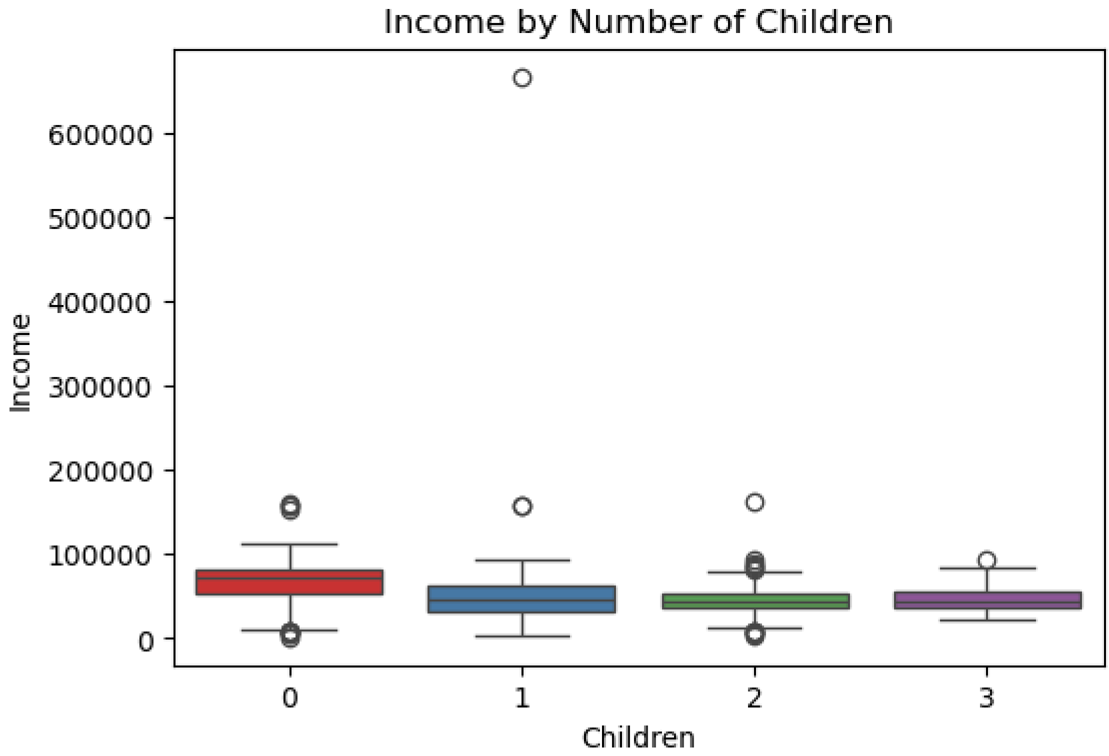
The histograms show the distribution of spending on different product categories segmented by Marital Status. Customers who are married or together tend to spend more on most categories, especially on wines and meat products. Single, divorced, or widowed customers generally spend less, with fewer high spenders. This indicates that marital status has a noticeable influence on purchasing behavior.

In [34]:
```
n_cols = 2
n_rows = -(-len(total_spent) // n_cols)

fig, axes = plt.subplots(n_rows, n_cols, figsize=(5 * n_cols, 4 * n_rows))
axes = axes.flatten()

for i, feature in enumerate(total_spent):
    sns.histplot(data=df, x=feature, hue='Children', kde=True, bins=30, ax=axes[
    axes[i].set_title(f"Distribution of {feature}")

plt.tight_layout()
plt.suptitle("Total Spent based on Children", fontsize=22, y=1.02)
plt.show()
```

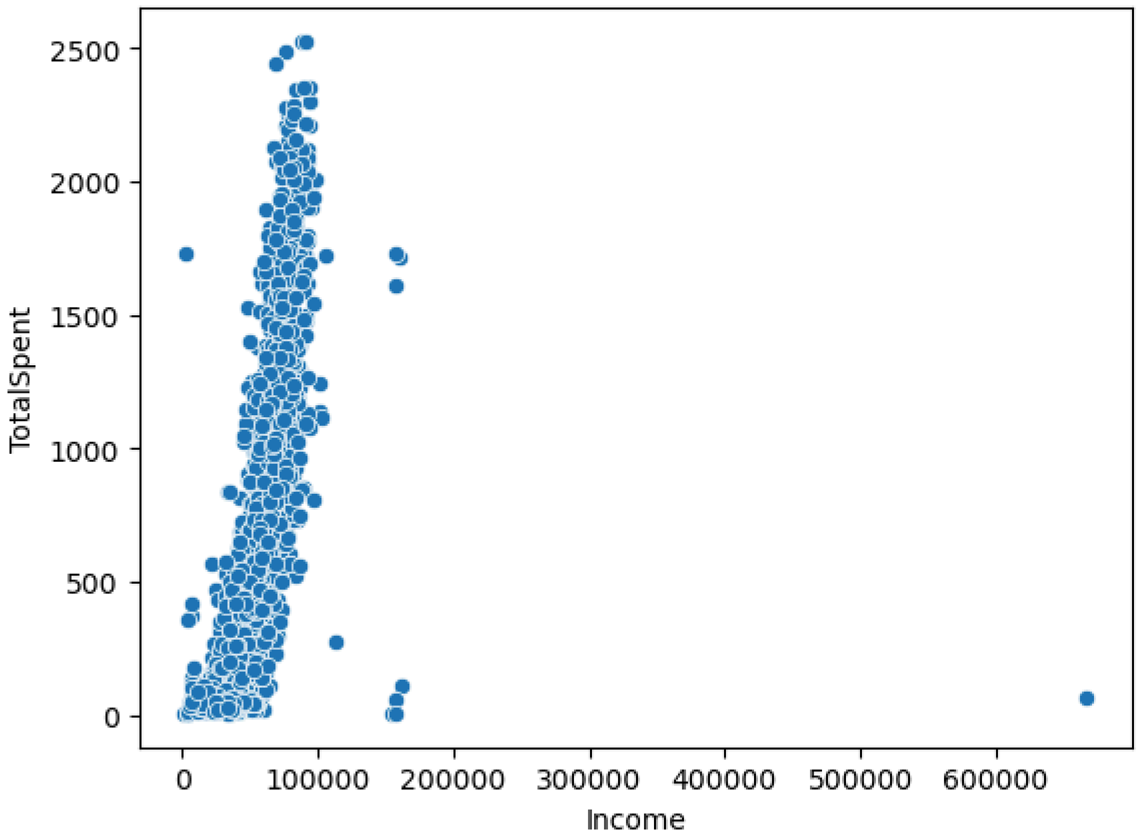


The histograms display spending across product categories according to the number of children in the household. Households with no children show a wider range of spending, including some very high spenders, while families with one or more children spend moderately, particularly on discretionary items like sweets and wine. This suggests that household size influences customer spending patterns.

In [35]:
```
plt.figure(figsize=(6,4))
sns.boxplot(data=df, x='Children', y='Income', palette='Set1')
plt.title('Income by Number of Children')
plt.show()
```

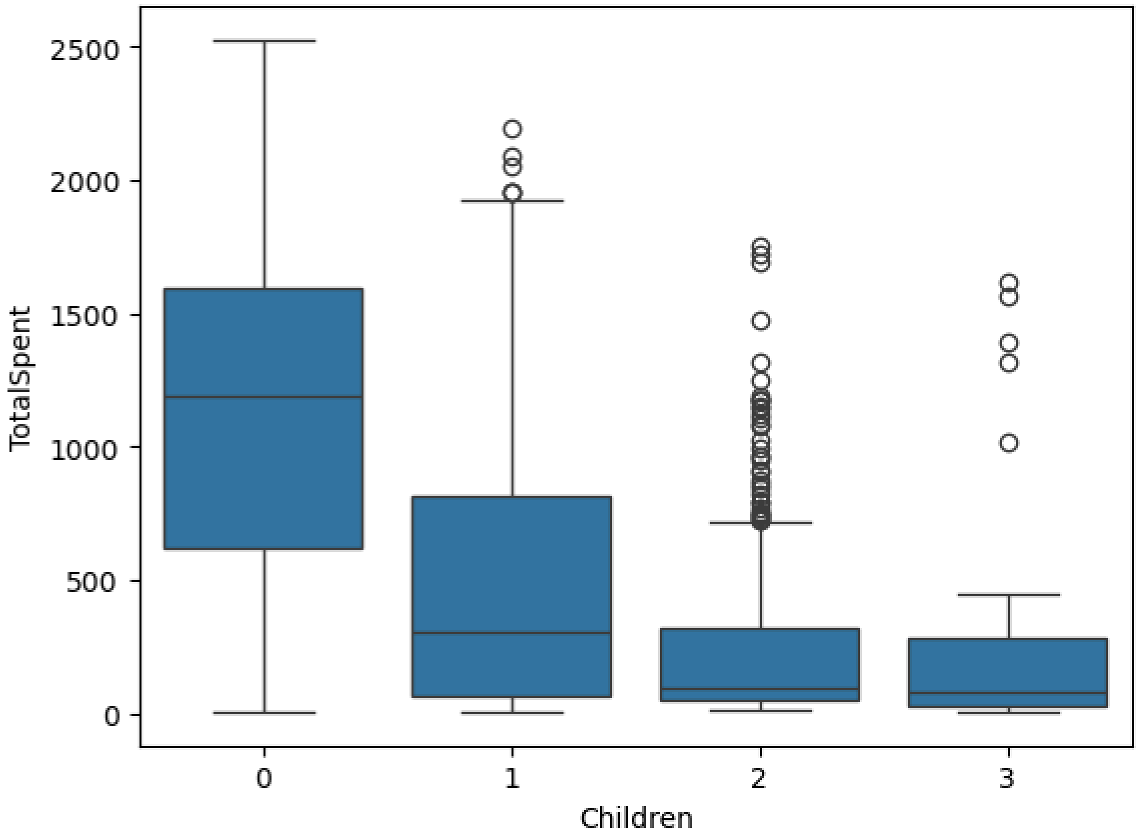


In [36]:
```
sns.scatterplot(x='Income', y='TotalSpent', data=df)
plt.show()
```



This scatter plot displays the relationship between customers'income and their total spending. A positive trend can be observed: clients with higher income tend to spend more overall, although the relationship is not perfectly linear. Some high-income customers still show moderate spending, which may indicate different spending habits or levels of engagement. The spread of points suggests significant variability in spending behavior across income levels, highlighting the need for segmentation to better understand these patterns.

In [37]:
```
sns.boxplot(x='Children', y='TotalSpent', data=df)
plt.show()
```



The boxplot shows how total spending varies with the number of children in the household.

Clustering Process

1. Scaling

In [38]:
```
features = ['Age', 'Income', 'Children', 'Recency', 'TotalSpent']
```

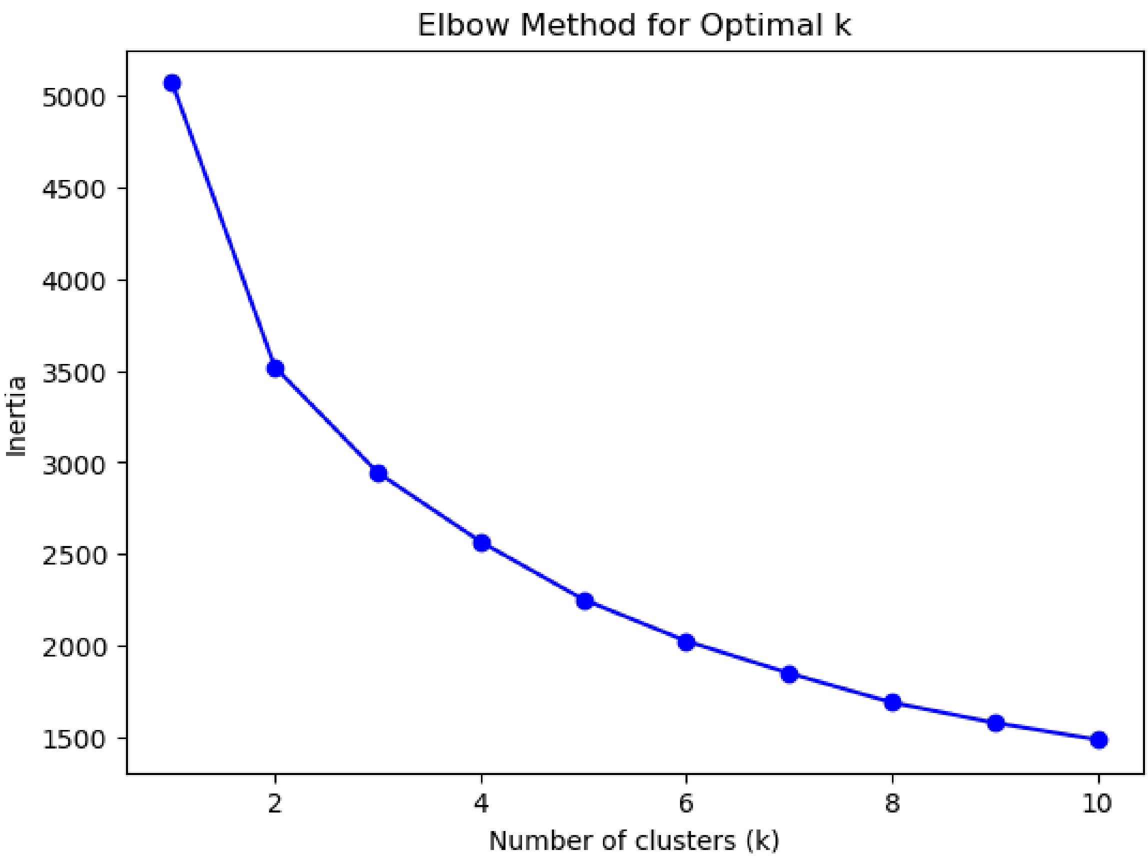
The features were scaled using RobustScaler, this makes it more resistant to outliers observed in the income and spending distributions.

As a result, even though a few customers have unusually high incomes or spending levels, these outliers do not distort the clustering results, ensuring more reliable and stable cluster formation.

In [39]:
```
from sklearn.preprocessing import RobustScaler
scaler = RobustScaler()
X_scaled = scaler.fit_transform(df[features])
```

2. Elbow Method

In [40]:
```
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt

list = []
K = range(1, 11)

for k in K:
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(X_scaled)
    list.append(kmeans.inertia_)


plt.figure(figsize=(7,5))
plt.plot(K, list, 'bo-')
plt.xlabel('Number of clusters (k)')
plt.ylabel('Inertia')
plt.title('Elbow Method for Optimal k')
plt.show()
```



```
Although the Elbow method did not show a clear inflection point,
four clusters were retained as the most meaningful and
actionable segmentation. This allows us to capture meaningful
customer segments without creating too many small or overlapping
clusters.
```

3. K-means

In [41]:
```
best_k = 4
kmeans = KMeans(n_clusters=best_k, init='k-means++', random_state=42)
df['Cluster'] = kmeans.fit_predict(X_scaled)
```

Cluster Analysis and Interpretation

In [42]:
```
df['Cluster'].value_counts()
```

Out[42]:
```
Cluster
0    677
1    600
2    488
3    447
Name: count, dtype: int64
```

In [43]:
```
df.groupby('Cluster')[features].mean().round(2)
```

Out[43]:

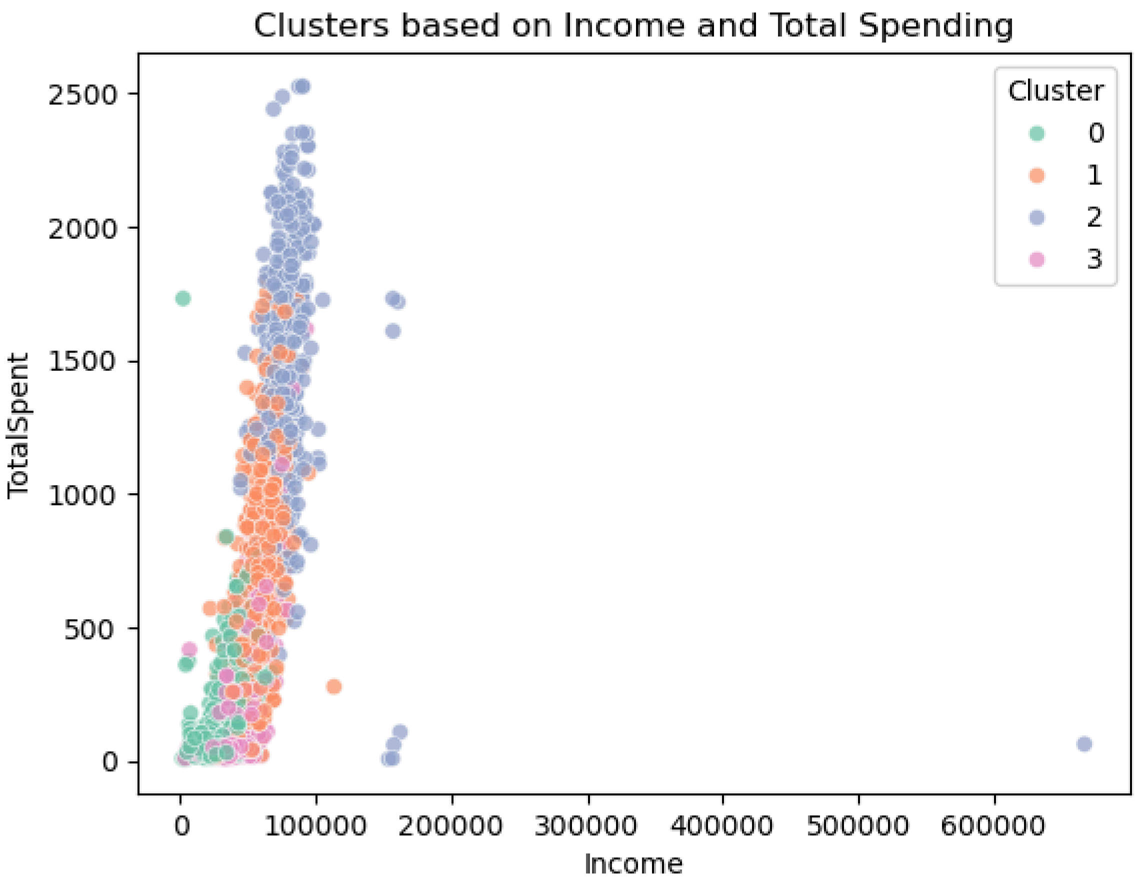
Cluster	Age	Income	Children	Recency	TotalSpent
0	47.63	31699.32	0.83	48.90	140.95
1	64.57	59476.98	0.89	47.01	741.61
2	54.74	79703.03	0.12	51.02	1447.72
3	59.49	43616.56	2.11	49.86	212.91

```
To describe each cluster, I calculated the mean of numerical
features, which represents the typical value for a customer in
that cluster.

Cluster 3: High-income, high-spending customers (Age: 54,
Income: 79K, TotalSpent: 1,443).
These are the most valuable clients in terms of revenue per
customer, with very few children. They are a top priority for
loyalty programs and premium offers.

Cluster 1: Low-income, low-spending customers (Age: 47, Income:
32K, TotalSpent: 147).
Young and less engaged clients. They may respond well to
targeted promotions to increase engagement.

Cluster 0: Older, medium-income customers (Age: 66, Income: 59K,
```

TotalSpent: 712).
Stable, moderately spending clients. Important to retain these customers.

Cluster 2: Families with children (Age: 60, Income: 44K, Children: 2.1, TotalSpent: 220).
Medium-income families, likely price-sensitive, with more discount purchases. They benefit from family-oriented offers.

In [44]:
```
sns.scatterplot(data=df, x='Income', y='TotalSpent', hue='Cluster', palette='Set
plt.title('Clusters based on Income and Total Spending')
plt.show()
```



The scatterplot shows customers segmented into clusters based on their income and total spending.

In [45]:
```
marital_distribution = pd.DataFrame()

for c in sorted(df['Cluster'].unique()):
    counts = df[df['Cluster']==c]['Marital_Status'].value_counts()
    marital_distribution[c] = counts

marital_distribution = marital_distribution.fillna(0).astype(int)

marital_distribution.columns = [f'Cluster {c}' for c in marital_distribution.col

marital_distribution
```

Out[45]:

Marital_Status	Cluster 0	Cluster 1	Cluster 2	Cluster 3
Married	278	226	176	177
Single	173	96	122	83
Together	168	157	128	120
Divorced	50	85	46	51
Widow	8	36	16	16

```
The table shows that all marital statuses are distributed across
the clusters without any cluster being dominated by a single
category.
This indicates that marital status alone is not a strong
discriminating feature for clustering customers, and other
features such as income, spending, and purchasing behavior are
more informative for segmenting the customer base.
```

In [46]:

```
df.groupby('Cluster')[['NumWebPurchases','NumCatalogPurchases','NumStorePurchase
```

Out[46]:

Cluster	NumWebPurchases	NumCatalogPurchases	NumStorePurchases	NumDealsPurch
0	2.56	0.72	3.49	
1	5.76	3.39	7.64	
2	5.02	6.03	8.30	
3	3.13	0.98	4.11	

```
Analyzing purchase behavior across clusters revealed distinct
shopping preferences.

* Cluster 0: customers buy through multiple channels with
moderate web and catalog purchases and higher in-store
purchases, showing balanced engagement and moderate
responsiveness to promotions.
* Cluster 1: members have low spending across all channels but
visit the website frequently, indicating low conversion despite
high online activity.
Cluster 2: consists of deal-sensitive families who make moderate
purchases in-store and via catalog, showing high responsiveness
to promotions.
* Cluster 3: represents high-spending premium customers who buy
across all channels but are less influenced by deals, purchasing
efficiently with fewer visits.
These insights complement the income and total spending
segmentation, providing a detailed understanding of customer
behavior for targeted marketing strategies.
```

```
Business Insights:

Cluster 3 are top-value customers worth prioritizing for loyalty
programs.
Cluster 1 are low-engagement clients who may respond well to
promotions.
Cluster 2 could benefit from family-oriented offers or bundle
discounts.
Cluster 0 likely includes older, stable customers — focus on
retention.
```

Conclusion:

```
This project demonstrates how unsupervised learning (K-Means)
can reveal meaningful customer segments.
The identified clusters highlight differences in income,
spending behavior, and engagement level, providing actionable
insights for marketing and customer relationship management.
```