Marketing Campaign Analysis

Objective: The objective of this analysis is to understand customer purchasing behavior, apply clustering techniques to segment customers into distinct groups, and analyze the customer profiles of each segment to develop tailored marketing strategies.

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib as plt
from sklearn.cluster import KMeans
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
df=pd.read_excel("/content/drive/MyDrive/marketing_campaign.xlsx")
df.head()
```

	ID	Year_Birth	Education	Marital_Status	Income	Kidhome	Teenhome	Dt_Customer	Recency	MntWines	...	NumWebVisitsMonth	A
0	5524	1957	Graduation	Single	58138.0	0	0	2012-09-04	58	635	...	7	
1	2174	1954	Graduation	Single	46344.0	1	1	2014-03-08	38	11	...	5	
2	4141	1965	Graduation	Together	71613.0	0	0	2013-08-21	26	426	...	4	
3	6182	1984	Graduation	Together	26646.0	1	0	2014-02-10	26	11	...	6	
4	5324	1981	PhD	Married	58293.0	1	0	2014-01-19	94	173	...	5	

5 rows × 29 columns

```
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
```

```
# Clearing duplicate values
df.drop_duplicates(inplace=True)
```

```
#Count null value
df.isnull().sum()
```

	0
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

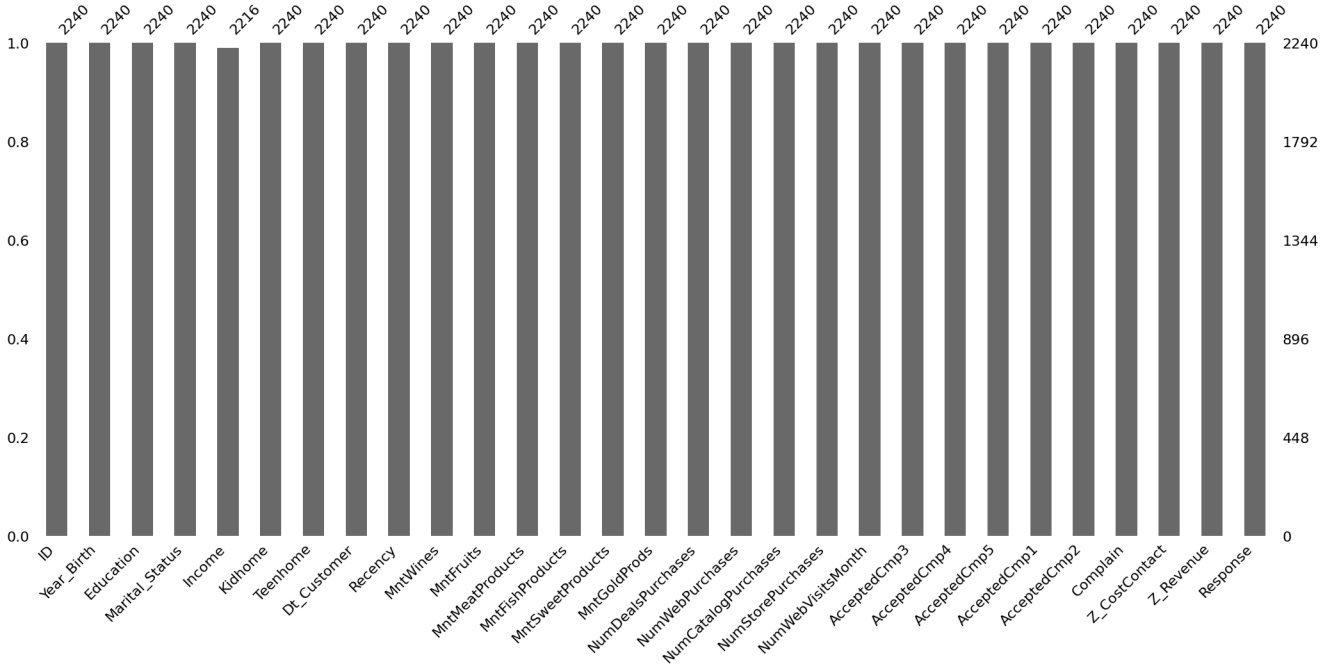
```
#Look up for the row where the Income is null
df[df['Income'].isnull()]
```

	ID	Year_Birth	Education	Marital_Status	Income	Kidhome	Teenhome	Dt_Customer	Recency	MntWines	...	NumWebVisitsMontl
10	1994	1983	Graduation	Married	NaN	1	0	2013-11-15	11	5	...	7
27	5255	1986	Graduation	Single	NaN	1	0	2013-02-20	19	5	...	7
43	7281	1959	PhD	Single	NaN	0	0	2013-11-05	80	81	...	2
48	7244	1951	Graduation	Single	NaN	2	1	2014-01-01	96	48	...	6
58	8557	1982	Graduation	Single	NaN	1	0	2013-06-17	57	11	...	6
71	10629	1973	2n Cycle	Married	NaN	1	0	2012-09-14	25	25	...	8
90	8996	1957	PhD	Married	NaN	2	1	2012-11-19	4	230	...	9
91	9235	1957	Graduation	Single	NaN	1	1	2014-05-27	45	7	...	7
92	5798	1973	Master	Together	NaN	0	0	2013-11-23	87	445	...	7
128	8268	1961	PhD	Married	NaN	0	1	2013-07-11	23	352	...	6
133	1295	1963	Graduation	Married	NaN	0	1	2013-08-11	96	231	...	4
312	2437	1989	Graduation	Married	NaN	0	0	2013-06-03	69	861	...	3
319	2863	1970	Graduation	Single	NaN	1	2	2013-08-23	67	738	...	7
1379	10475	1970	Master	Together	NaN	0	1	2013-04-01	39	187	...	5
1382	2902	1958	Graduation	Together	NaN	1	1	2012-09-03	87	19	...	5
1383	4345	1964	2n Cycle	Single	NaN	1	1	2014-01-12	49	5	...	7
1386	3769	1972	PhD	Together	NaN	1	0	2014-03-02	17	25	...	7
2059	7187	1969	Master	Together	NaN	1	1	2013-05-18	52	375	...	3
2061	1612	1981	PhD	Single	NaN	1	0	2013-05-31	82	23	...	6
2078	5079	1971	Graduation	Married	NaN	1	1	2013-03-03	82	71	...	8
2079	10339	1954	Master	Together	NaN	0	1	2013-06-23	83	161	...	6
2081	3117	1955	Graduation	Single	NaN	0	1	2013-10-18	95	264	...	7
2084	5250	1943	Master	Widow	NaN	0	0	2013-10-30	75	532	...	7
2228	8720	1978	2n Cycle	Together	NaN	0	0	2012-08-12	53	32	...	6

24 rows × 29 columns

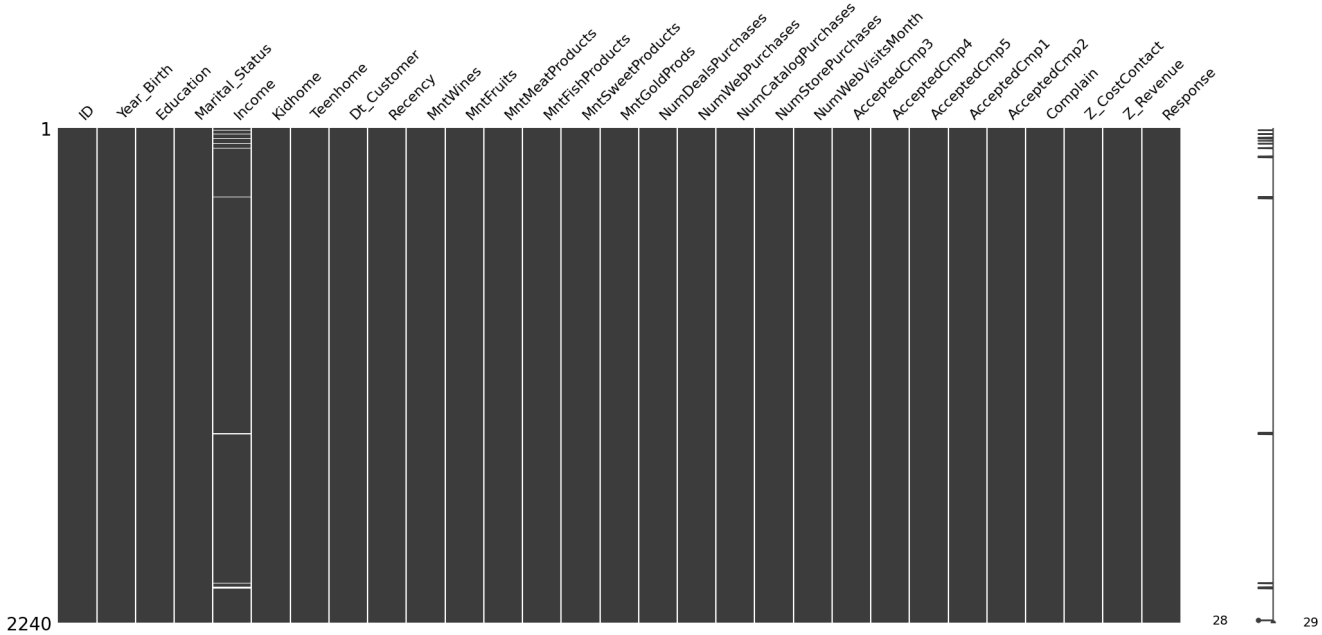
```
import missingno as msno
# Visualize missing data
msno.bar(df)
```

```
<Axes: >
```




```
# Visualize missing data
msno.matrix(df)
```

```
<Axes: >
```



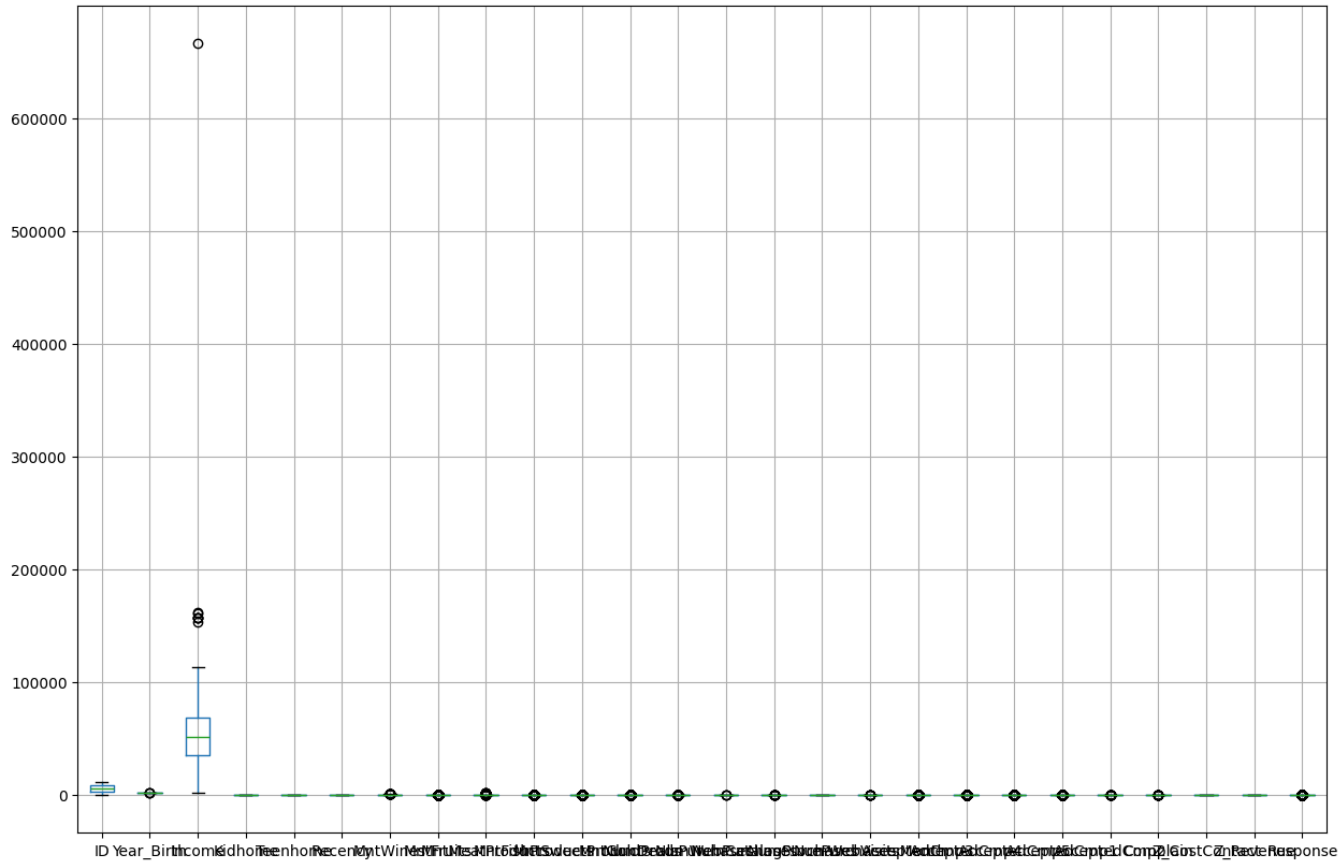
Since the missing data only appear in one column and only take a very small amount portion of the entire dataset, I've decided to drop the rows contain the missing values

```
# Drop missing value
df.dropna(inplace=True)
```

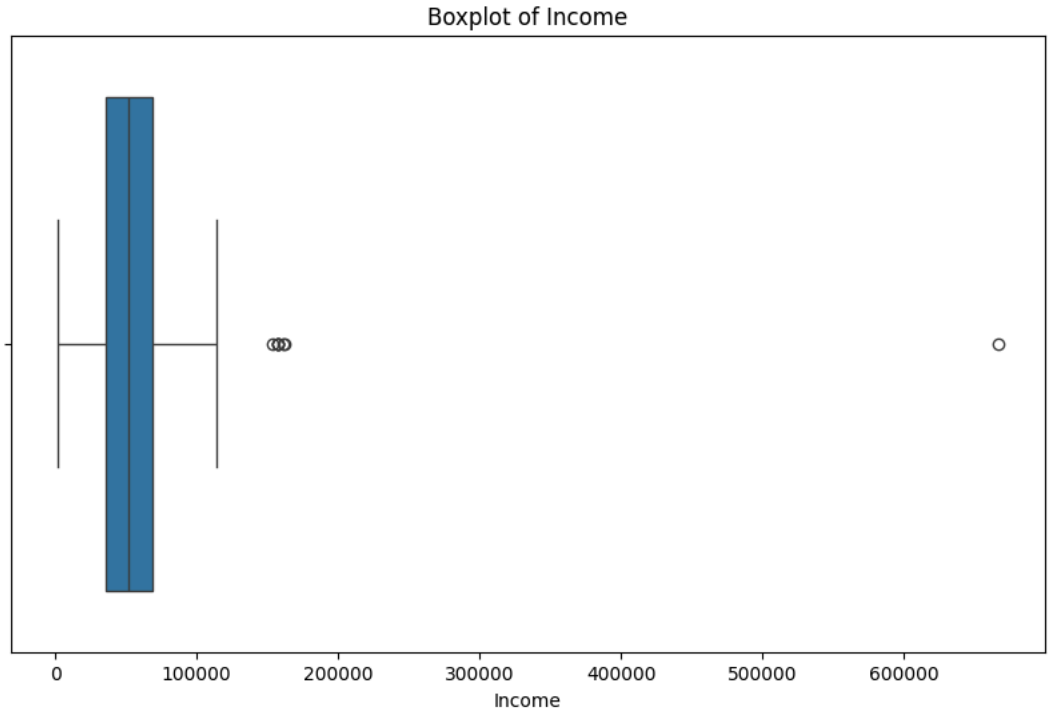
Outliers

```
import matplotlib.pyplot as plt
```

```
# Create a figure and axis
fig,ax = plt.subplots(figsize=(15, 10))
# Plot the box plots for all columns
df.boxplot(ax=ax)
# Show the plot
plt.show()
```




```
# Plot the distribution of a column to visualize the outliers
plt.figure(figsize=(10, 6))
sns.boxplot(x=df['Income'])
plt.title('Boxplot of Income')
plt.show()
```



```
# Calculate the IQR for the Income column
Q1 = df['Income'].quantile(0.25)
Q3 = df['Income'].quantile(0.75)
IQR = Q3 - Q1

# Identify the outliers in the Income column
outliers = df[(df['Income'] < (Q1 - 1.5 * IQR)) | (df['Income'] > (Q3 + 1.5 * IQR))]

# Print the number of outliers
print("Number of outliers in the Income column:", len(outliers))
```

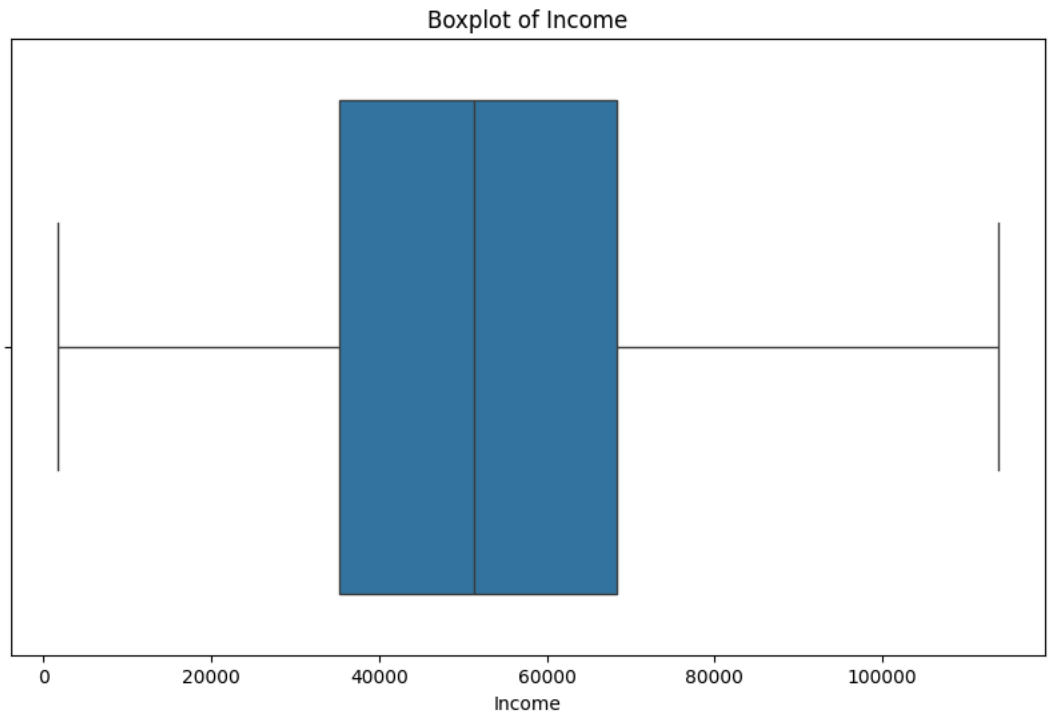
```
Number of outliers in the Income column: 8
```

```
# Remove the outliers in the Income column
df = df[~((df['Income'] < (Q1 - 1.5 * IQR)) | (df['Income'] > (Q3 + 1.5 * IQR)))]
```

```
# Print the updated shape of the dataframe
print("Updated shape of the dataframe:", df.shape)
```

```
Updated shape of the dataframe: (2208, 29)
```

```
# Plot the distribution of a column to visualize the outliers
plt.figure(figsize=(10, 6))
sns.boxplot(x=df['Income'])
plt.title('Boxplot of Income')
plt.show()
```



Feature Engineering

```
# Review the dataset
df.head()
```

	ID	Year_Birth	Education	Marital_Status	Income	Kidhome	Teenhome	Dt_Customer	Recency	MntWines	...	NumWebVisitsMonth	A
0	5524	1957	Graduation	Single	58138.0	0	0	2012-09-04	58	635	...	7	
1	2174	1954	Graduation	Single	46344.0	1	1	2014-03-08	38	11	...	5	
2	4141	1965	Graduation	Together	71613.0	0	0	2013-08-21	26	426	...	4	
3	6182	1984	Graduation	Together	26646.0	1	0	2014-02-10	26	11	...	6	
4	5324	1981	PhD	Married	58293.0	1	0	2014-01-19	94	173	...	5	

5 rows × 29 columns

```
print("Unique values in Education column:", df['Education'].unique())
print("Unique values in Marital_Status column:", df['Marital_Status'].unique())
```

```
Unique values in Education column: ['Graduation' 'PhD' 'Master' 'Basic' '2n Cycle']
Unique values in Marital_Status column: ['Single' 'Together' 'Married' 'Divorced' 'Widow' 'Alone' 'Absurd' 'YOLO']
```

```
# Classify education levels
def education_level(education):
    if education in ['Graduation', 'PhD', 'Master']:
        return 'High'
    elif education in ['Basic']:
        return 'Middle'
    else:
        return 'Low'

df['Education_Level'] = df['Education'].apply(education_level)
```

```
# Classify martial status
def living_status(marital_status):
    if marital_status in ['Alone', 'Absurd', 'YOLO']:
        return 'Alone'
    else:
        return 'In a relationship'

df['Living_Status'] = df['Marital_Status'].apply(living_status)
```

```
# Creating Age
df['Age'] = 2024 - df['Year_Birth']
```

```
# Creating number of campaings accepted
df['Total_Campaigns_Accepted'] = df[['AcceptedCmp1', 'AcceptedCmp2', 'AcceptedCmp3', 'AcceptedCmp4', 'AcceptedCmp5']].sum(axis=1)
```

```
# Creating average spent per purchase
df['Average_Spend'] = (df[['MntWines', 'MntFruits', 'MntMeatProducts', 'MntFishProducts', 'MntSweetProducts', 'MntGoldProds']].sum(axis=
```

```
# Creating spent per purchase
df['Spent'] = df['MntWines']+df["MntWines"] +df['MntFruits']+ df['MntMeatProducts'] +df['MntFishProducts']+df['MntSweetProducts']+ df['M
```

```
# Creating parenthood status
df['Is_Parent'] = (df['Kidhome'] + df['Teenhome'] > 0).astype(int)
```

```
# Creating average monthly web visit for the company website
df['avg_web_visits'] = df['NumWebVisitsMonth'] / 12
```

```
# Creating online purchase ratio
df['online_purchase_ratio'] = df['NumWebPurchases'] / (df['NumWebPurchases'] + df['NumCatalogPurchases'] + df['NumStorePurchases'])
```

```
# Drop irrelevant column
to_drop = ['Dt_Customer', 'Z_CostContact', 'Z_Revenue', 'Year_Birth', 'ID']
df = df.drop(to_drop, axis=1)
```

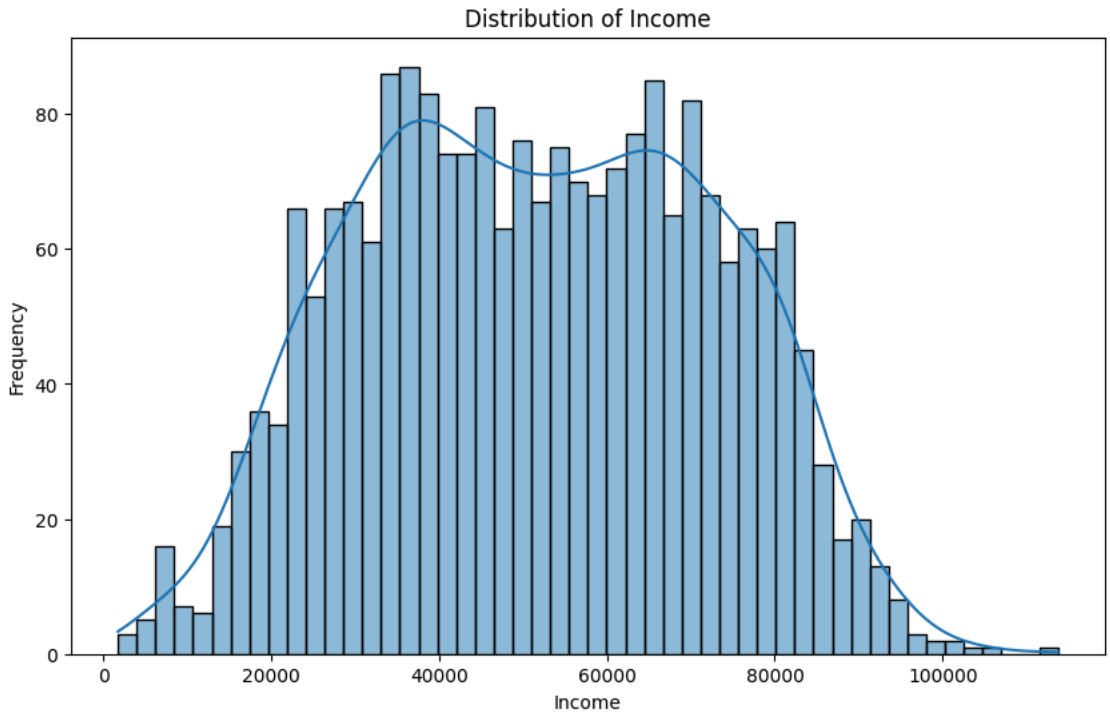
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 2208 entries, 0 to 2239
Data columns (total 33 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   Education                2208 non-null   object
 1   Marital_Status           2208 non-null   object
 2   Income                   2208 non-null   float64
 3   Kidhome                  2208 non-null   int64
 4   Teenhome                 2208 non-null   int64
 5   Recency                  2208 non-null   int64
 6   MntWines                 2208 non-null   int64
 7   MntFruits                2208 non-null   int64
 8   MntMeatProducts          2208 non-null   int64
 9   MntFishProducts          2208 non-null   int64
 10  MntSweetProducts         2208 non-null   int64
 11  MntGoldProds             2208 non-null   int64
 12  NumDealsPurchases        2208 non-null   int64
 13  NumWebPurchases          2208 non-null   int64
 14  NumCatalogPurchases      2208 non-null   int64
 15  NumStorePurchases        2208 non-null   int64
 16  NumWebVisitsMonth        2208 non-null   int64
 17  AcceptedCmp3             2208 non-null   int64
 18  AcceptedCmp4             2208 non-null   int64
 19  AcceptedCmp5             2208 non-null   int64
 20  AcceptedCmp1             2208 non-null   int64
```

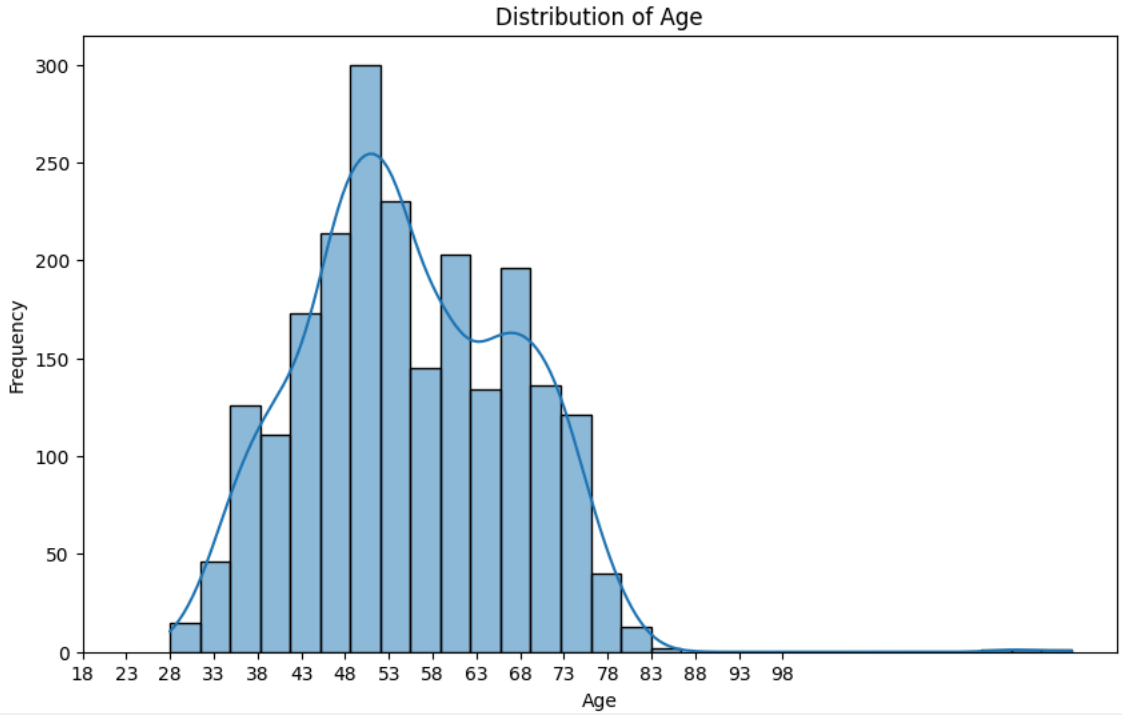
```
 21  AcceptedCmp2              2208 non-null   int64
 22  Complain                  2208 non-null   int64
 23  Response                  2208 non-null   int64
 24  Education_Level           2208 non-null   object
 25  Living_Status             2208 non-null   object
 26  Age                       2208 non-null   int64
 27  Total_Campaigns_Accepted  2208 non-null   int64
 28  Average_Spend             2208 non-null   float64
 29  Spent                     2208 non-null   int64
 30  Is_Parent                 2208 non-null   int64
 31  avg_web_visits            2208 non-null   float64
 32  online_purchase_ratio     2204 non-null   float64
dtypes: float64(4), int64(25), object(4)
memory usage: 586.5+ KB
```

EDA

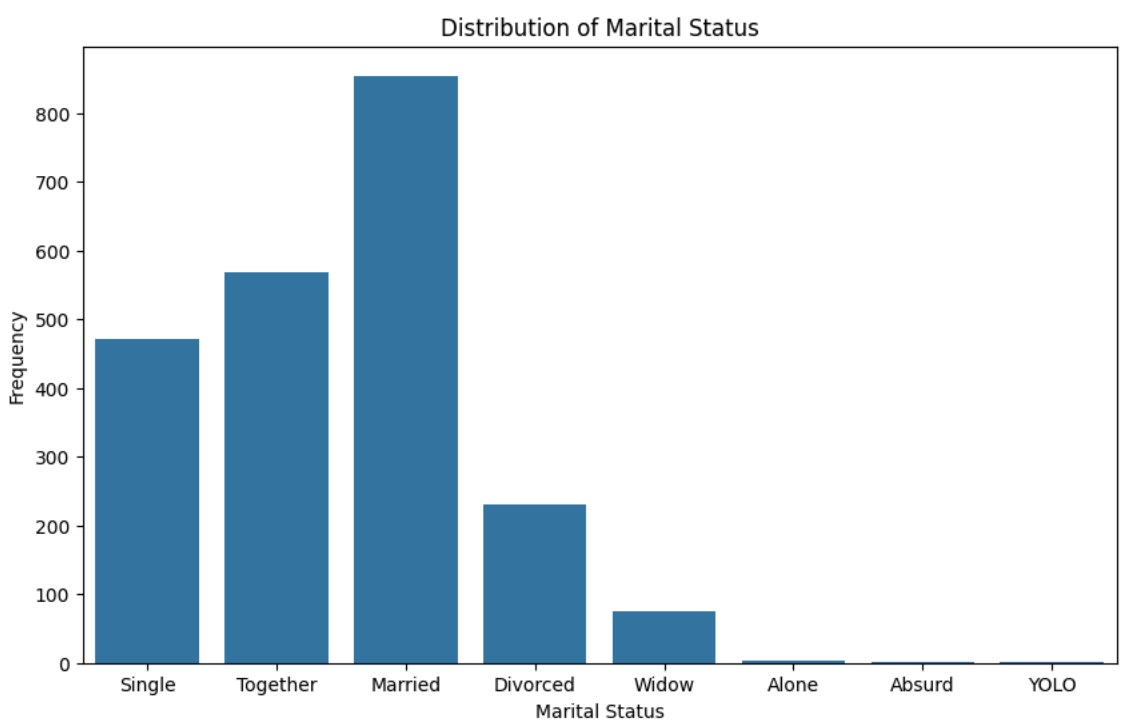
```
# Create distribution of income via histogram
plt.figure(figsize=(10, 6))
sns.histplot(df['Income'], bins=50, kde=True)
plt.title('Distribution of Income')
plt.xlabel('Income')
plt.ylabel('Frequency')
plt.show()
```



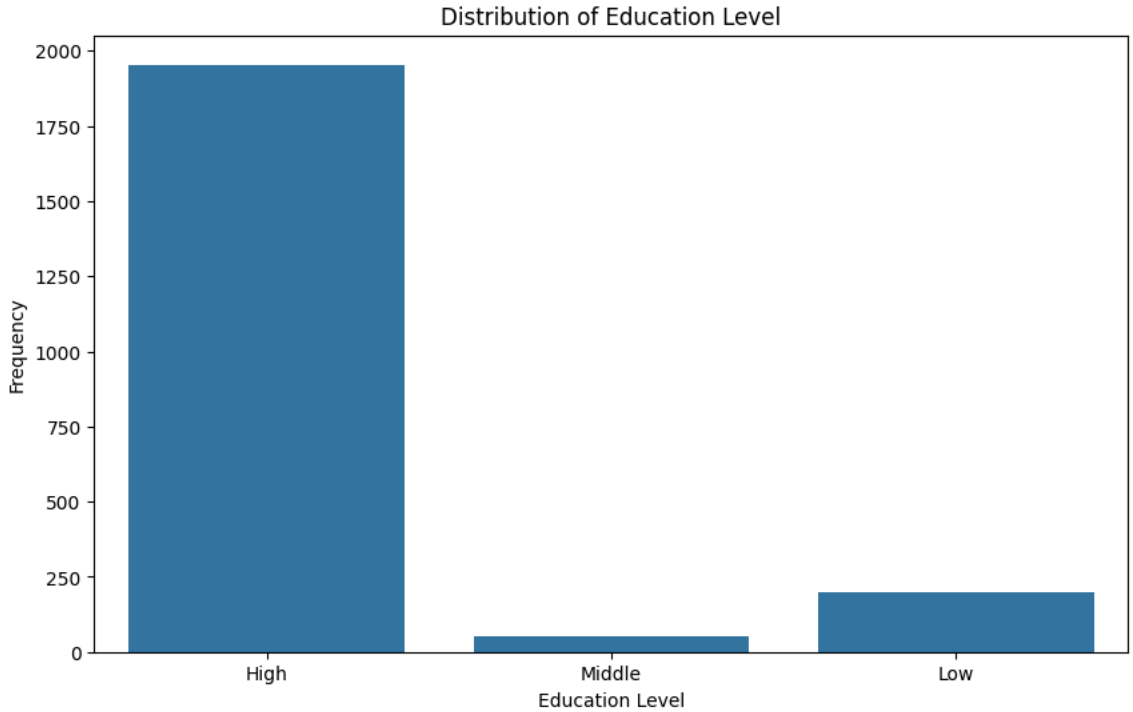

```
# Create distribution of age via histogram
plt.figure(figsize=(10, 6))
sns.histplot(df['Age'], bins=30, kde=True)
plt.title('Distribution of Age')
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.xticks(range(18, 100, 5)) # Customize x-axis ticks for better readability
plt.show()
```



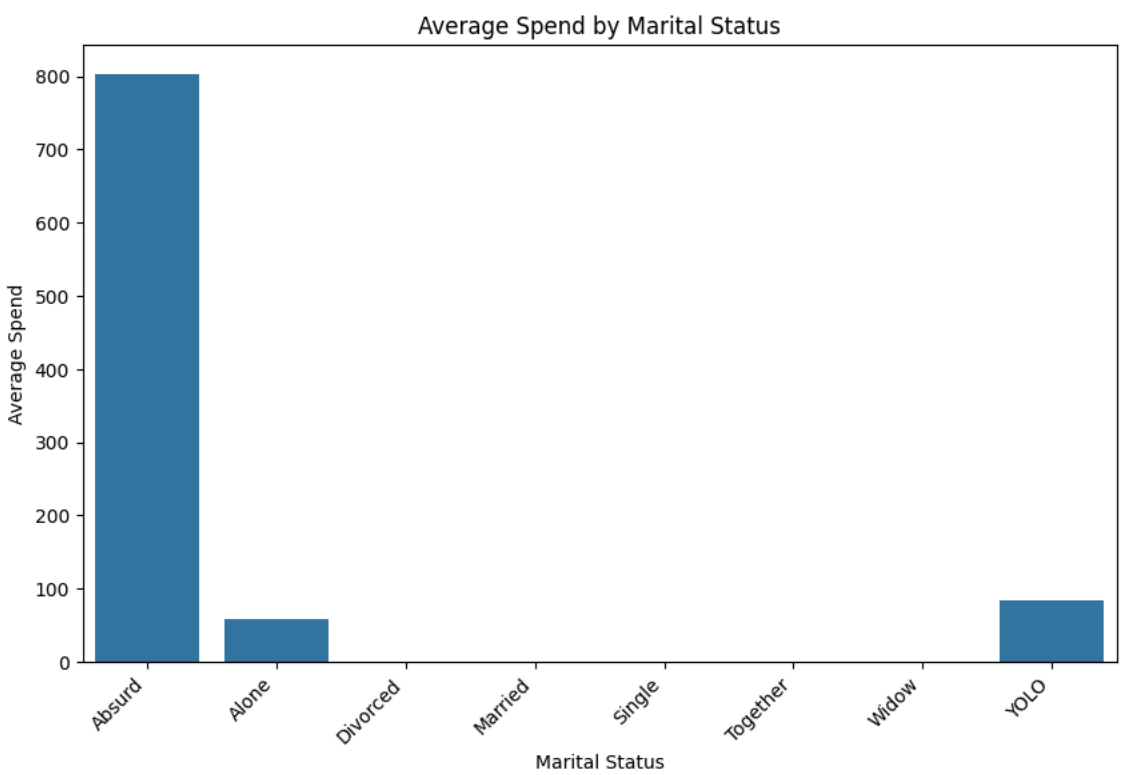

```
# Create distribution of martial status via histogram
plt.figure(figsize=(10, 6))
sns.countplot(x=df['Marital_Status'])
plt.title('Distribution of Marital Status')
plt.xlabel('Marital Status')
plt.ylabel('Frequency')
plt.show()
```



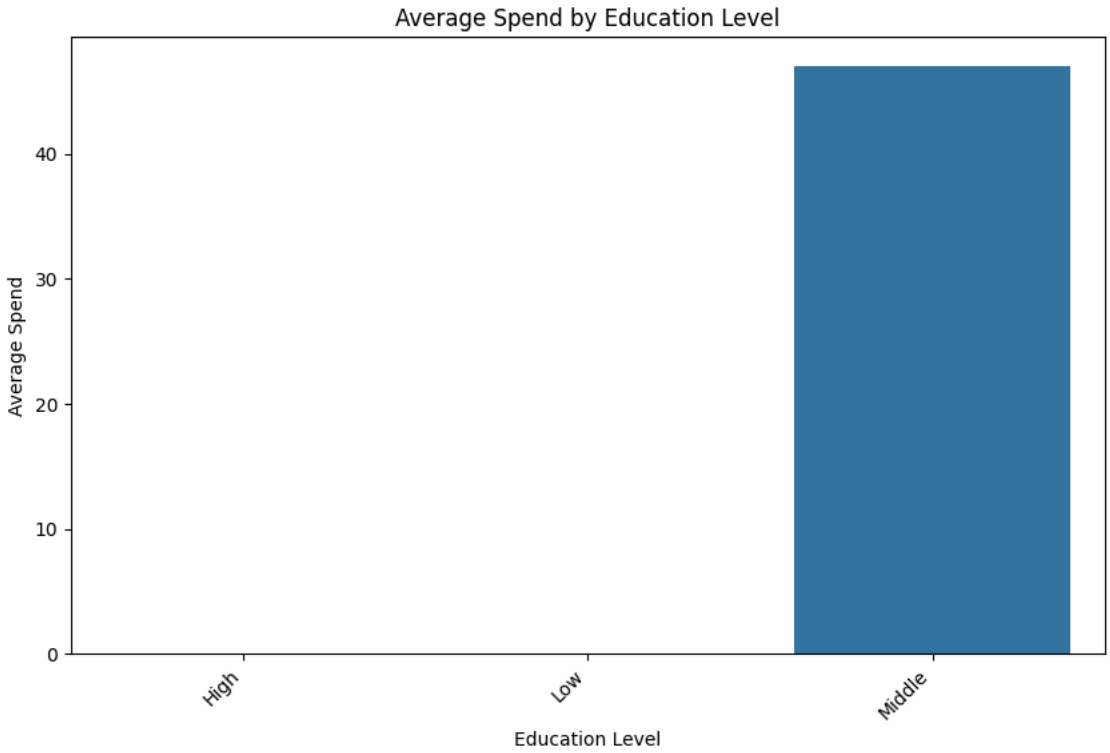

```
# Create distribution of educational level through histogram
plt.figure(figsize=(10, 6))
sns.countplot(x=df['Education_Level'])
plt.title('Distribution of Education Level')
plt.xlabel('Education Level')
plt.ylabel('Frequency')
plt.show()
```



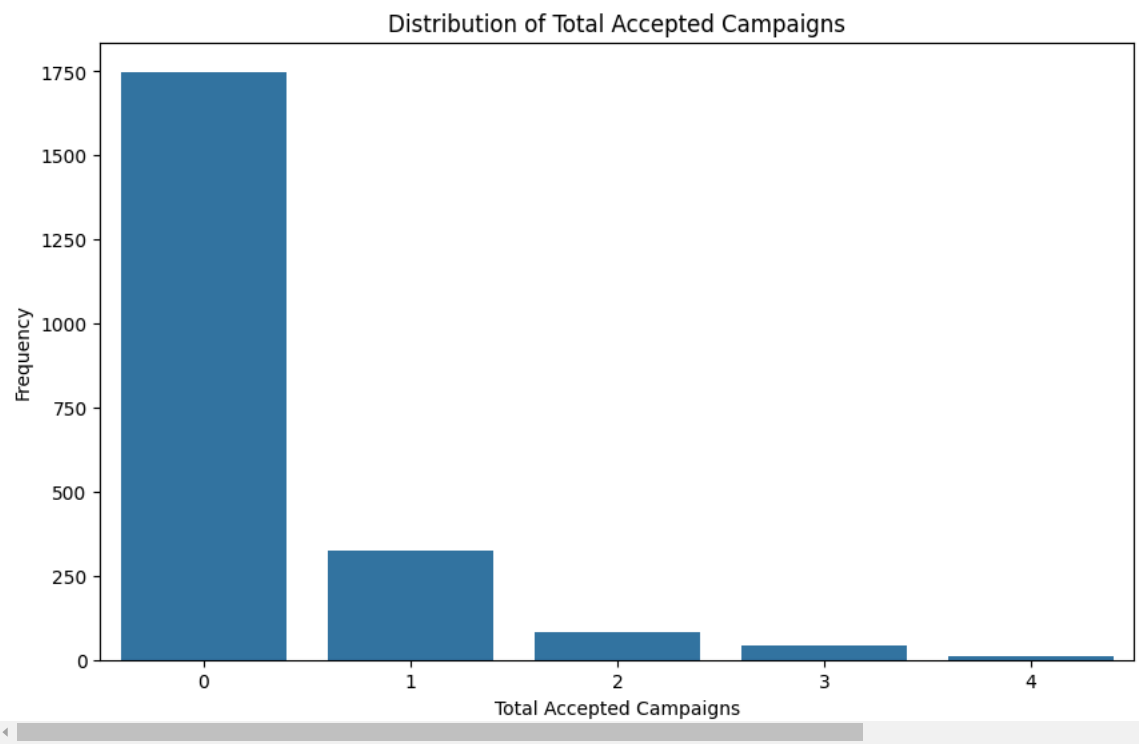

```
# Create average spend by martial status
average_spend_by_marital_status = df.groupby('Marital_Status')['Average_Spend'].mean().reset_index()
plt.figure(figsize=(10, 6))
sns.barplot(x='Marital_Status', y='Average_Spend', data=average_spend_by_marital_status)
plt.title('Average Spend by Marital Status')
plt.xlabel('Marital Status')
plt.ylabel('Average Spend')
plt.xticks(rotation=45, ha='right')
plt.show()
```



```
# Create average spend by education level
average_spend_by_education_level = df.groupby('Education_Level')['Average_Spend'].mean().reset_index()
plt.figure(figsize=(10, 6))
sns.barplot(x='Education_Level', y='Average_Spend', data=average_spend_by_education_level)
plt.title('Average Spend by Education Level')
plt.xlabel('Education Level')
plt.ylabel('Average Spend')
plt.xticks(rotation=45, ha='right')
plt.show()
```

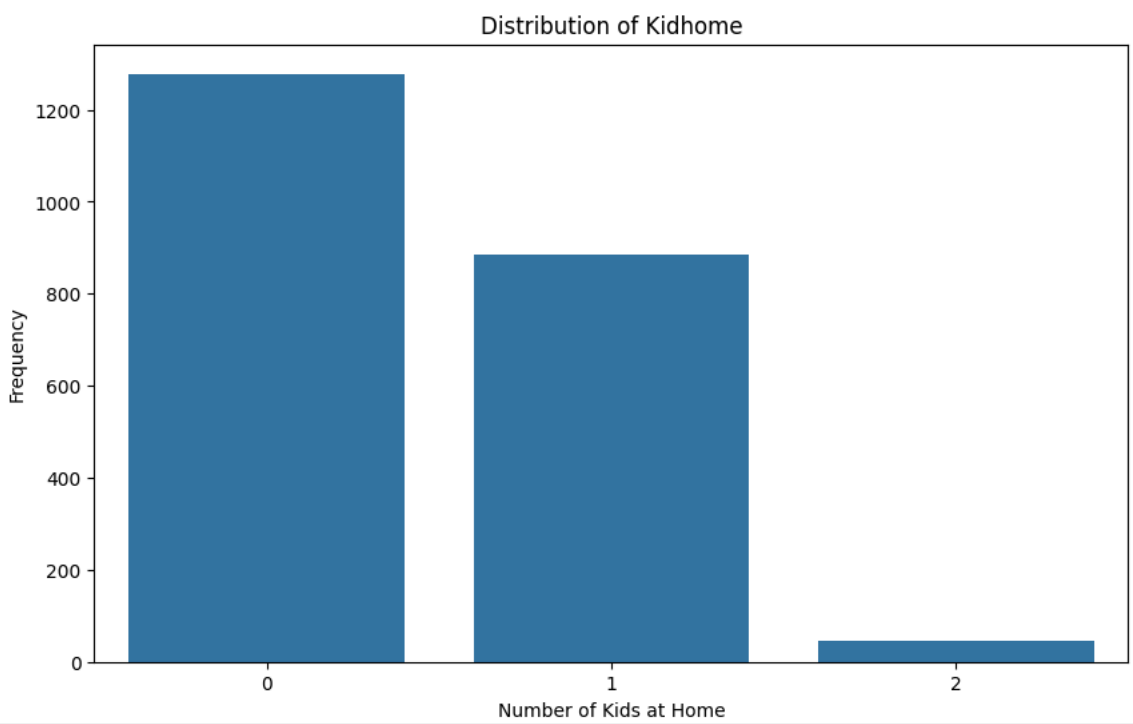



```
# Create total accepted campaign distribution
plt.figure(figsize=(10, 6))
sns.countplot(x=df['Total_Campaigns_Accepted'])
plt.title('Distribution of Total Accepted Campaigns')
plt.xlabel('Total Accepted Campaigns')
plt.ylabel('Frequency')
plt.show()
```

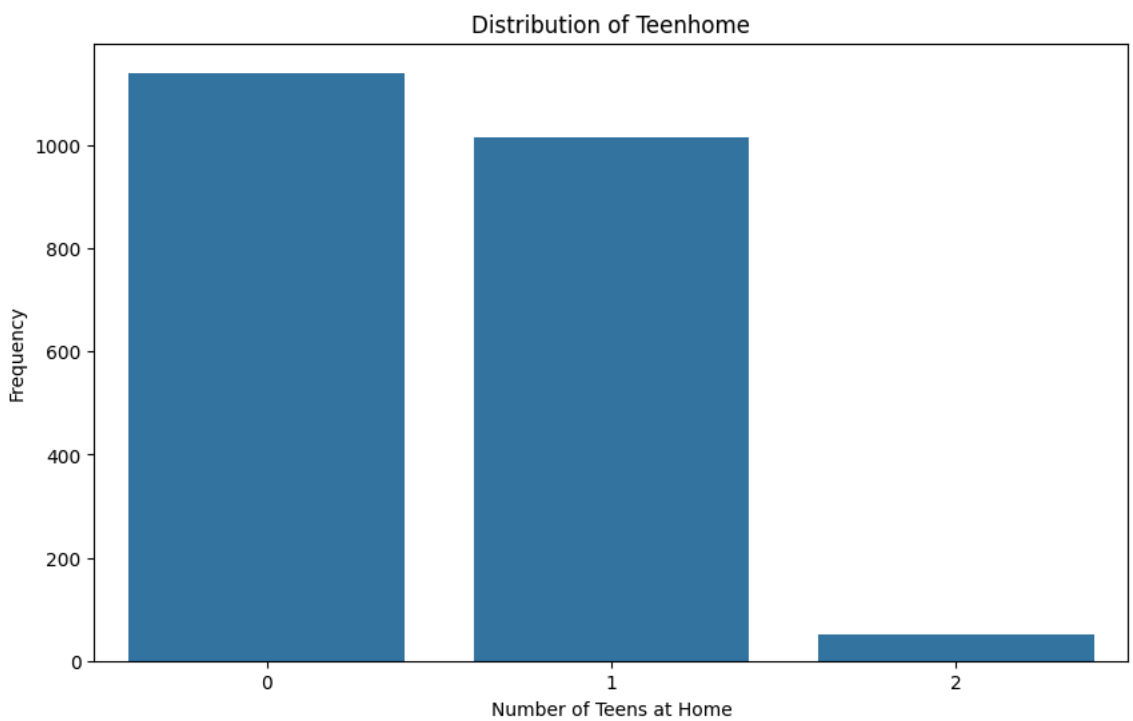



```
# Create distribution of kid home
plt.figure(figsize=(10, 6))
```

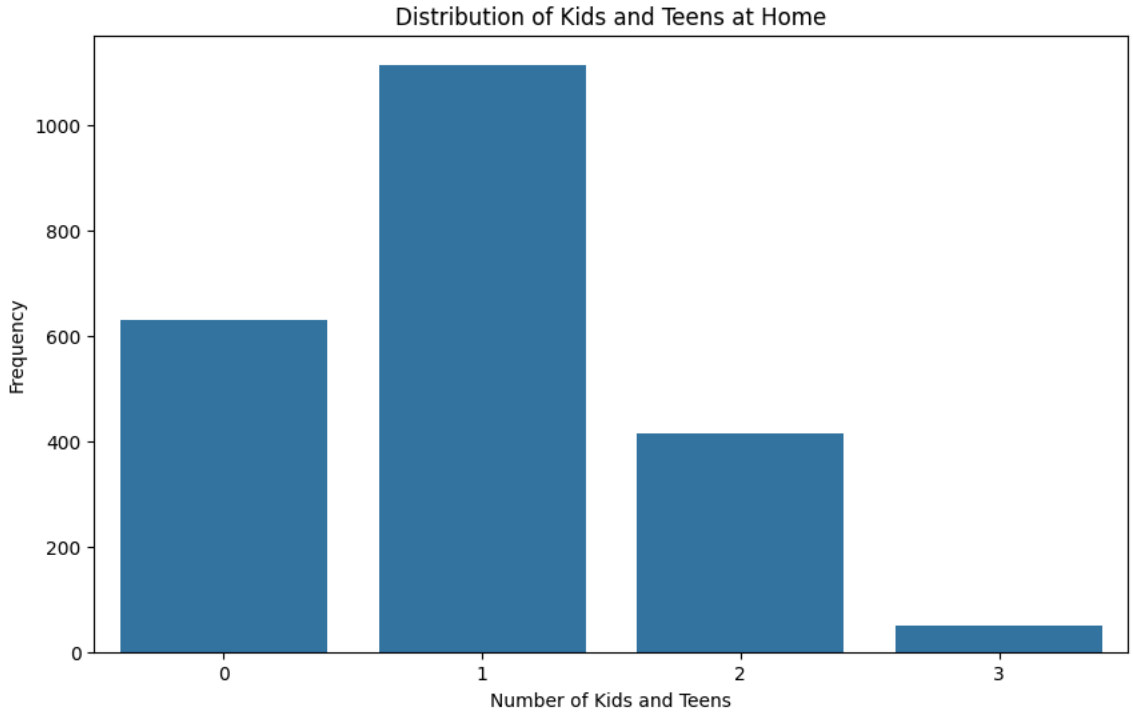
```
sns.countplot(x=df['Kidhome'])
plt.title('Distribution of Kidhome')
plt.xlabel('Number of Kids at Home')
plt.ylabel('Frequency')
plt.show()
```



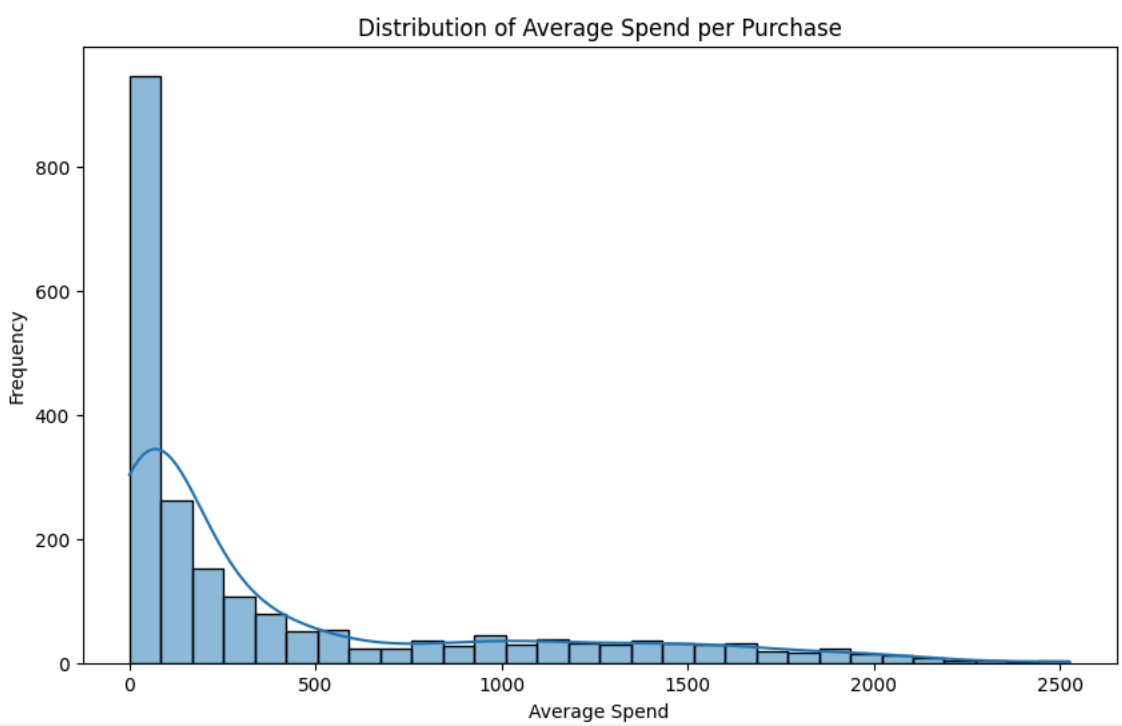

```
# Create distribution of teen home
plt.figure(figsize=(10, 6))
sns.countplot(x=df['Teenhome'])
plt.title('Distribution of Teenhome')
plt.xlabel('Number of Teens at Home')
plt.ylabel('Frequency')
plt.show()
```



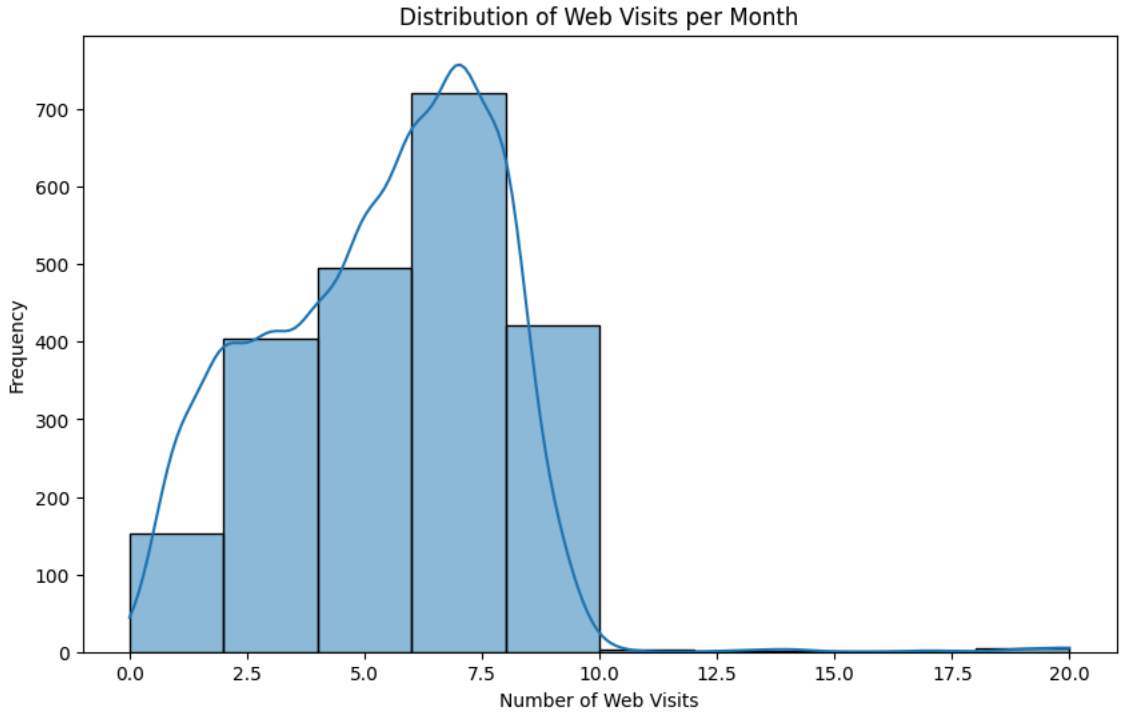

```
# Create distribution of kid and teen home
plt.figure(figsize=(10, 6))
sns.countplot(x=df['Kidhome'] + df['Teenhome'])
plt.title('Distribution of Kids and Teens at Home')
plt.xlabel('Number of Kids and Teens')
plt.ylabel('Frequency')
plt.show()
```



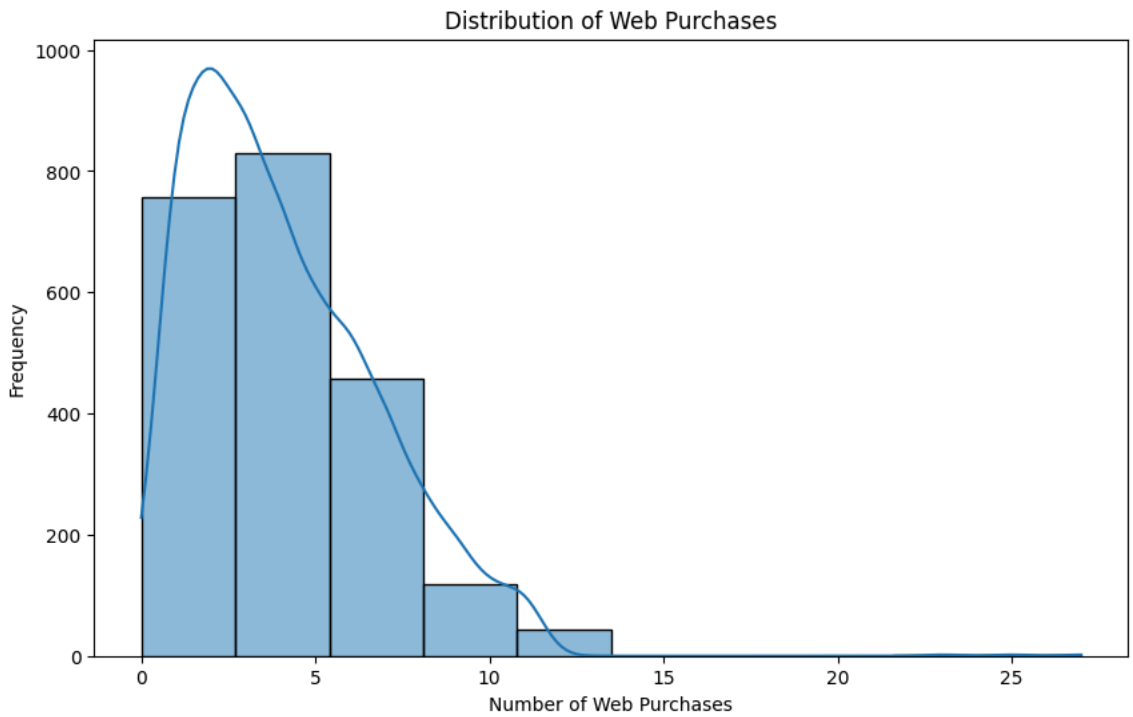

```
# prompt: Create average spend per purchase distribution
plt.figure(figsize=(10, 6))
sns.histplot(df['Average_Spend'], bins=30, kde=True)  # Adjust the number of bins as needed
plt.title('Distribution of Average Spend per Purchase')
plt.xlabel('Average Spend')
plt.ylabel('Frequency')
plt.show()
```



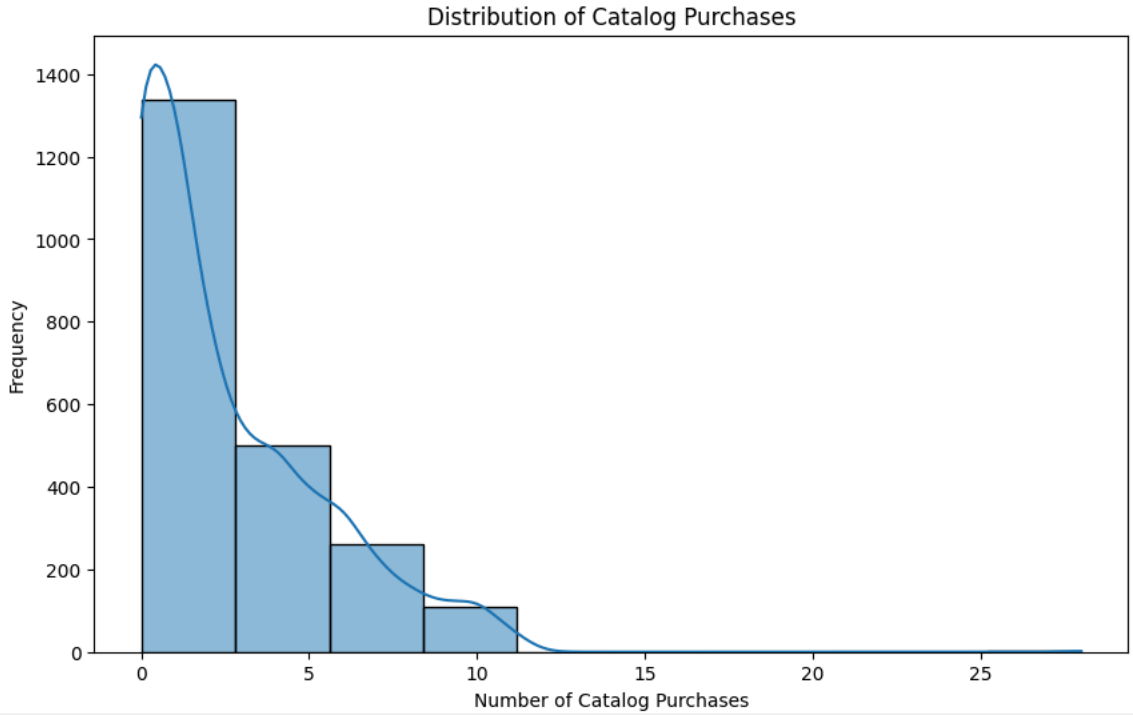

```
# Create distribution of web visits per month
plt.figure(figsize=(10, 6))
sns.histplot(df['NumWebVisitsMonth'], bins=10, kde=True)
plt.title('Distribution of Web Visits per Month')
plt.xlabel('Number of Web Visits')
plt.ylabel('Frequency')
plt.show()
```



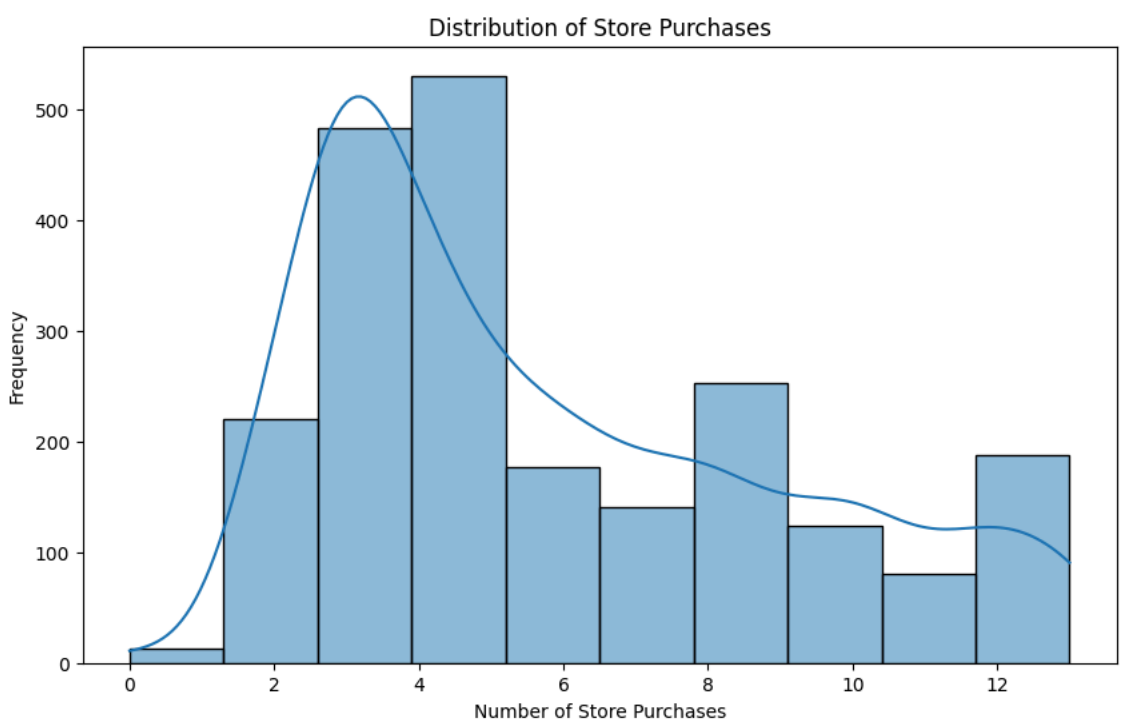

```
# Create distribution of web purchases
plt.figure(figsize=(10, 6))
sns.histplot(df['NumWebPurchases'], bins=10, kde=True)
plt.title('Distribution of Web Purchases')
plt.xlabel('Number of Web Purchases')
plt.ylabel('Frequency')
plt.show()
```




```
# Create distribution of catalog purchases
plt.figure(figsize=(10, 6))
sns.histplot(df['NumCatalogPurchases'], bins=10, kde=True)
plt.title('Distribution of Catalog Purchases')
plt.xlabel('Number of Catalog Purchases')
plt.ylabel('Frequency')
plt.show()
```




```
# Create distribution of store purchases
plt.figure(figsize=(10, 6))
sns.histplot(df['NumStorePurchases'], bins=10, kde=True)
plt.title('Distribution of Store Purchases')
plt.xlabel('Number of Store Purchases')
plt.ylabel('Frequency')
plt.show()
```



K-Mean Clustering

One-hot encode

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 2208 entries, 0 to 2239
```

```
Data columns (total 33 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Education                 2208 non-null   object
 1   Marital_Status            2208 non-null   object
 2   Income                    2208 non-null   float64
 3   Kidhome                   2208 non-null   int64
 4   Teenhome                  2208 non-null   int64
 5   Recency                   2208 non-null   int64
 6   MntWines                  2208 non-null   int64
 7   MntFruits                 2208 non-null   int64
 8   MntMeatProducts           2208 non-null   int64
 9   MntFishProducts           2208 non-null   int64
 10  MntSweetProducts          2208 non-null   int64
 11  MntGoldProds              2208 non-null   int64
 12  NumDealsPurchases         2208 non-null   int64
 13  NumWebPurchases           2208 non-null   int64
 14  NumCatalogPurchases       2208 non-null   int64
 15  NumStorePurchases         2208 non-null   int64
 16  NumWebVisitsMonth         2208 non-null   int64
 17  AcceptedCmp3              2208 non-null   int64
 18  AcceptedCmp4              2208 non-null   int64
 19  AcceptedCmp5              2208 non-null   int64
 20  AcceptedCmp1              2208 non-null   int64
 21  AcceptedCmp2              2208 non-null   int64
 22  Complain                  2208 non-null   int64
 23  Response                  2208 non-null   int64
 24  Education_Level           2208 non-null   object
 25  Living_Status             2208 non-null   object
 26  Age                       2208 non-null   int64
 27  Total_Campaigns_Accepted  2208 non-null   int64
 28  Average_Spend             2208 non-null   float64
 29  Spent                     2208 non-null   int64
 30  Is_Parent                 2208 non-null   int64
 31  avg_web_visits            2208 non-null   float64
 32  online_purchase_ratio     2204 non-null   float64
dtypes: float64(4), int64(25), object(4)
memory usage: 586.5+ KB
```

```
# Generate new table for clustering
data=df.drop(['Education','Marital_Status','Kidhome','Teenhome','AcceptedCmp3','AcceptedCmp4','AcceptedCmp5','AcceptedCmp2','AcceptedCm
```

```
# One hot encode
data_encoded = pd.get_dummies(data, columns=['Education_Level', 'Living_Status'])
```

Data Scaling

```
data_encoded.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 2208 entries, 0 to 2239
Data columns (total 27 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   Income                          2208 non-null   float64
 1   Recency                         2208 non-null   int64
 2   MntWines                        2208 non-null   int64
 3   MntFruits                       2208 non-null   int64
 4   MntMeatProducts                 2208 non-null   int64
 5   MntFishProducts                 2208 non-null   int64
 6   MntSweetProducts                2208 non-null   int64
 7   MntGoldProds                    2208 non-null   int64
 8   NumDealsPurchases               2208 non-null   int64
 9   NumWebPurchases                 2208 non-null   int64
 10  NumCatalogPurchases             2208 non-null   int64
 11  NumStorePurchases               2208 non-null   int64
 12  NumWebVisitsMonth               2208 non-null   int64
 13  Complain                        2208 non-null   int64
 14  Response                        2208 non-null   int64
 15  Age                             2208 non-null   int64
 16  Total_Campaigns_Accepted        2208 non-null   int64
 17  Average_Spend                   2208 non-null   float64
 18  Spent                           2208 non-null   int64
 19  Is_Parent                       2208 non-null   int64
 20  avg_web_visits                  2208 non-null   float64
 21  online_purchase_ratio           2204 non-null   float64
 22  Education_Level_High            2208 non-null   bool
 23  Education_Level_Low             2208 non-null   bool
 24  Education_Level_Middle          2208 non-null   bool
 25  Living_Status_Alone             2208 non-null   bool
 26  Living_Status_In a relationship 2208 non-null   bool
dtypes: bool(5), float64(4), int64(18)
memory usage: 407.5 KB
```

```
data_encoded['Education_Level_Low'] = data_encoded['Education_Level_Low'].astype(int)
data_encoded['Education_Level_Middle'] = data_encoded['Education_Level_Middle'].astype(int)
data_encoded['Education_Level_High'] = data_encoded['Education_Level_High'].astype(int)
data_encoded['Living_Status_Alone'] = data_encoded['Living_Status_Alone'].astype(int)
data_encoded['Living_Status_In a relationship'] = data_encoded['Living_Status_In a relationship'].astype(int)
```

```
data_encoded.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 2208 entries, 0 to 2239
Data columns (total 27 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   Income                          2208 non-null   float64
 1   Recency                         2208 non-null   int64
 2   MntWines                        2208 non-null   int64
 3   MntFruits                       2208 non-null   int64
 4   MntMeatProducts                 2208 non-null   int64
 5   MntFishProducts                 2208 non-null   int64
 6   MntSweetProducts                2208 non-null   int64
 7   MntGoldProds                    2208 non-null   int64
 8   NumDealsPurchases               2208 non-null   int64
 9   NumWebPurchases                 2208 non-null   int64
 10  NumCatalogPurchases             2208 non-null   int64
 11  NumStorePurchases               2208 non-null   int64
 12  NumWebVisitsMonth               2208 non-null   int64
 13  Complain                        2208 non-null   int64
 14  Response                        2208 non-null   int64
 15  Age                             2208 non-null   int64
 16  Total_Campaigns_Accepted        2208 non-null   int64
 17  Average_Spend                   2208 non-null   float64
 18  Spent                           2208 non-null   int64
 19  Is_Parent                       2208 non-null   int64
 20  avg_web_visits                  2208 non-null   float64
 21  online_purchase_ratio           2204 non-null   float64
 22  Education_Level_High            2208 non-null   int64
 23  Education_Level_Low             2208 non-null   int64
 24  Education_Level_Middle          2208 non-null   int64
 25  Living_Status_Alone             2208 non-null   int64
 26  Living_Status_In a relationship 2208 non-null   int64
dtypes: float64(4), int64(23)
memory usage: 483.0 KB
```

```
# Scaling data using standard scaler
from sklearn.preprocessing import StandardScaler
columns = data_encoded.columns
data_encoded = np.nan_to_num(data_encoded, nan=0.0, posinf=1e10, neginf=-1e10)
scaler = StandardScaler()
data_scaled = scaler.fit_transform(data_encoded)
```

```
# Transform array into dataframe
data_scaled_df = pd.DataFrame(data_scaled, columns=columns)
data_scaled_df.head()
```

	Income	Recency	MntWines	MntFruits	MntMeatProducts	MntFishProducts	MntSweetProducts	MntGoldProds	NumDealsPurchases	Num
0	0.314089	0.310588	0.974689	1.545554	1.747688	2.449620	1.480933	0.846621	0.362555	
1	-0.255431	-0.380686	-0.874529	-0.638540	-0.731613	-0.652518	-0.635460	-0.735161	-0.167943	
2	0.964782	-0.795450	0.355320	0.566478	-0.176066	1.336500	-0.148933	-0.040720	-0.698440	
3	-1.206626	-0.795450	-0.874529	-0.563226	-0.667335	-0.506535	-0.586807	-0.754451	-0.167943	
4	0.321573	1.554881	-0.394444	0.415851	-0.217388	0.150388	-0.002975	-0.561551	1.423550	

5 rows × 27 columns

Finding number of clusters using elbow method

```
pip install kneed
```

```
Collecting kneed
  Downloading kneed-0.8.5-py3-none-any.whl.metadata (5.5 kB)
Requirement already satisfied: numpy>=1.14.2 in /usr/local/lib/python3.10/dist-packages (from kneed) (1.26.4)
Requirement already satisfied: scipy>=1.0.0 in /usr/local/lib/python3.10/dist-packages (from kneed) (1.13.1)
Downloading kneed-0.8.5-py3-none-any.whl (10 kB)
Installing collected packages: kneed
Successfully installed kneed-0.8.5
```

```
from kneed import KneeLocator
import matplotlib.pyplot as plt
inertia_values = []  # Danh sách lưu inertia cho từng k

for k in range(1, 11):
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(data_scaled_df)
    inertia_values.append(kmeans.inertia_)

# Using kneed to find the cluster
kneedle = KneeLocator(range(1, 11), inertia_values, curve='convex', direction='decreasing')

# Optimal number of cluster
print(f"Optimal number of clusters (k): {kneedle.elbow}")
```

```
Optimal number of clusters (k): 3
```

Fit the model

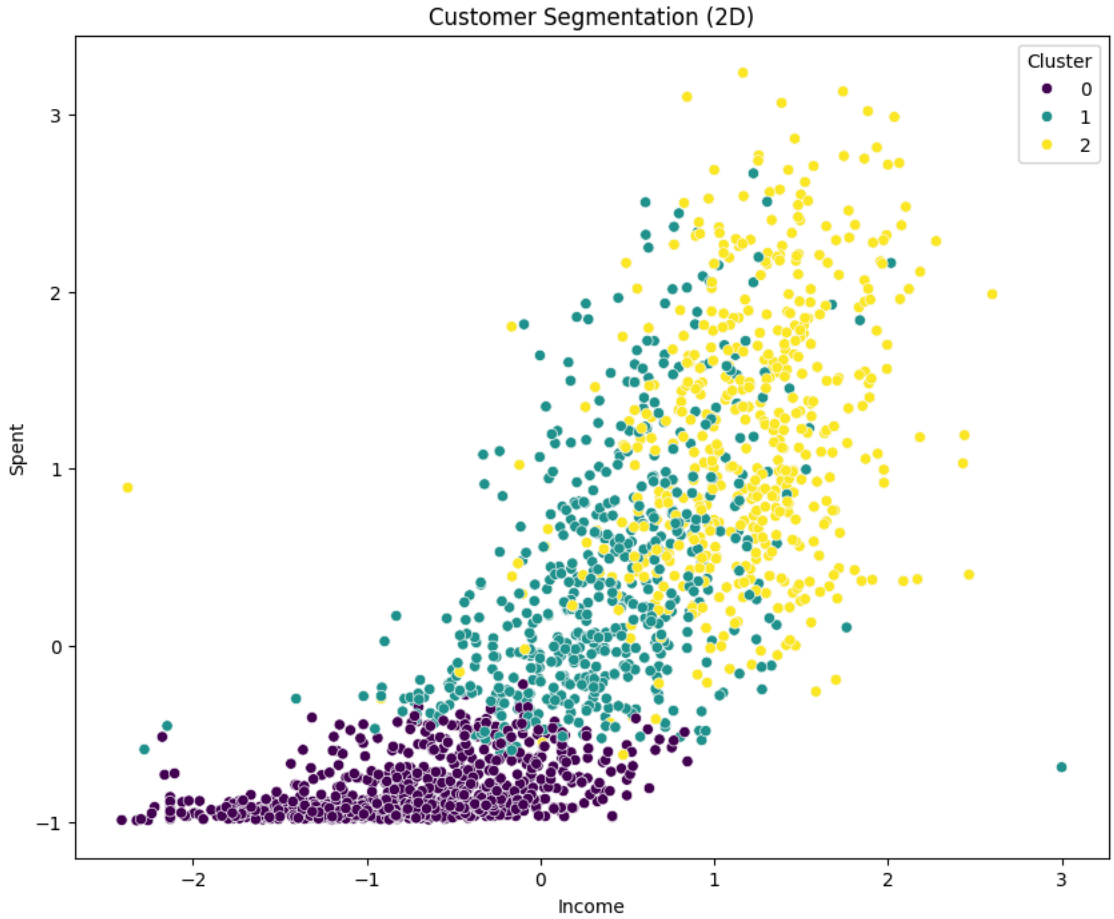
```
# Fit the model
from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters=3, random_state=42)
kmeans.fit(data_scaled_df)
```

```
KMeans
KMeans(n_clusters=3, random_state=42)
```

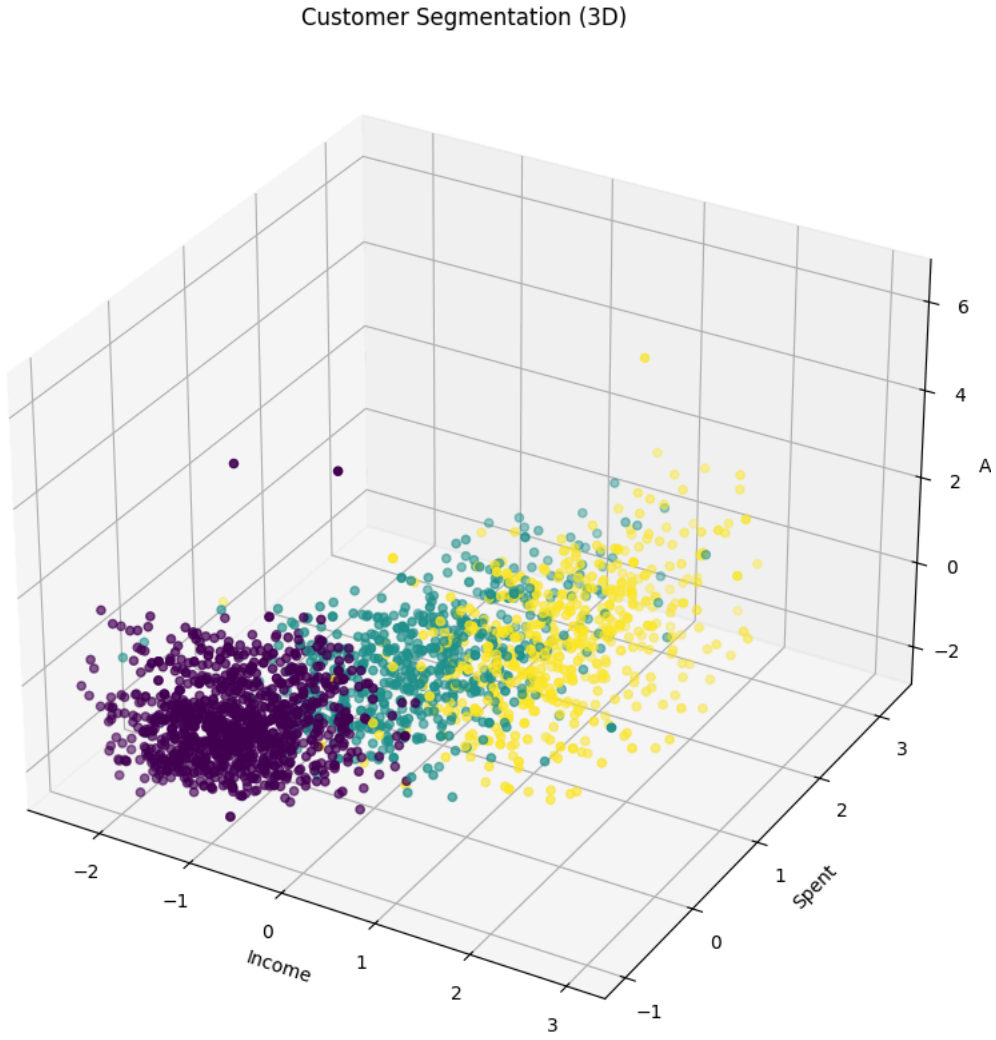
```
## Evaluate clusters quality
from sklearn.metrics import silhouette_score
# Silhouette Score
sil_score = silhouette_score(data_scaled_df, kmeans.labels_)
print(f"Silhouette Score: {sil_score}")
from sklearn.metrics import calinski_harabasz_score
# Calinski-Harabasz Index
ch_index = calinski_harabasz_score(data_scaled_df, kmeans.labels_)
print(f"Calinski-Harabasz Index: {ch_index}")
from sklearn.metrics import davies_bouldin_score

# Davies-Bouldin Index
db_index = davies_bouldin_score(data_scaled_df, kmeans.labels_)
print(f"Davies-Bouldin Index: {db_index}")
```

```
Silhouette Score: 0.17818597655377874
Calinski-Harabasz Index: 459.5989591933623
Davies-Bouldin Index: 1.9749627756451993
```

```
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import pandas as pd
# Add cluster labels to the DataFrame
data_scaled_df['Cluster'] = kmeans.labels_
# 2D Visualization
plt.figure(figsize=(10, 8))
sns.scatterplot(x='Income', y='Spent', hue='Cluster', data=data_scaled_df, palette='viridis')
plt.title('Customer Segmentation (2D)')
plt.show()
```

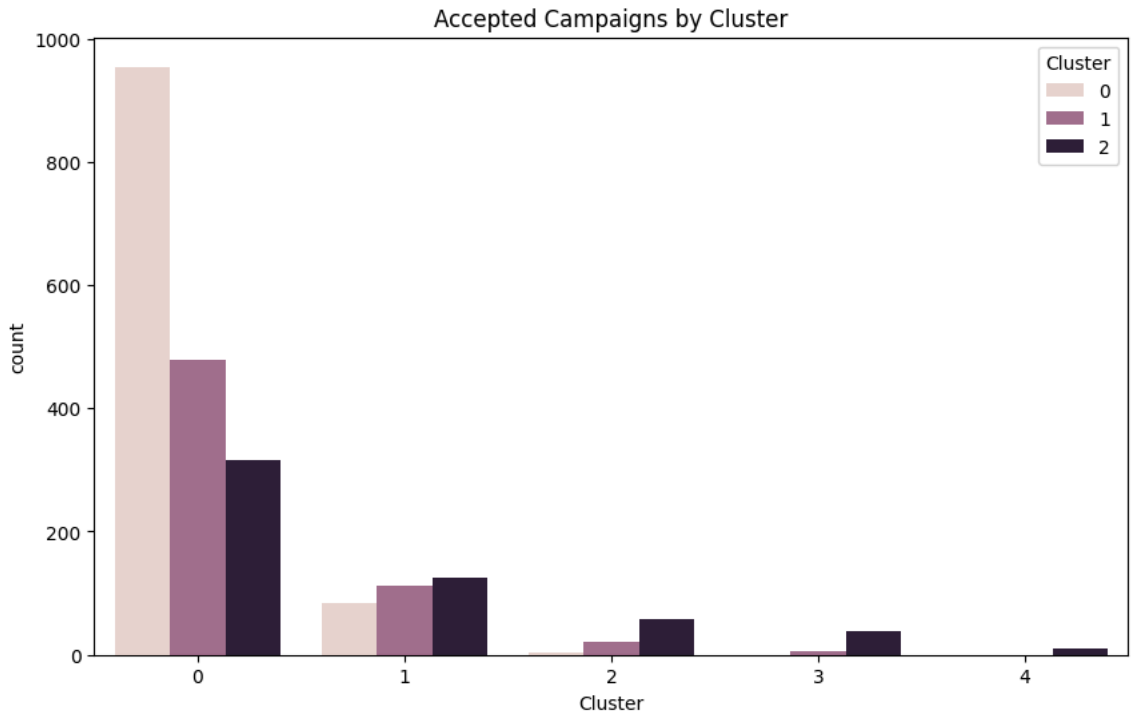



```
# 3D Visualization
fig = plt.figure(figsize=(12, 10))
ax = fig.add_subplot(111, projection='3d')

# Choose three features for 3D visualization
features = ['Income', 'Spent', 'Age']

scatter = ax.scatter(data_scaled_df[features[0]], data_scaled_df[features[1]], data_scaled_df[features[2]], c=data_scaled_df['Cluster'],

ax.set_xlabel(features[0])
ax.set_ylabel(features[1])
ax.set_zlabel(features[2])
ax.set_title('Customer Segmentation (3D)')

plt.show()
```




```
# Accepted campaign by clusters
data['Cluster']=kmeans.labels_
plt.figure(figsize=(10, 6))
sns.countplot(x=data['Total_Campaigns_Accepted'], hue=data['Cluster'])
plt.title('Accepted Campaigns by Cluster')
plt.xlabel('Cluster')
```

```
Text(0.5, 0, 'Cluster')
```



```
# Average spend by cluster
plt.figure(figsize=(10, 6))
sns.boxplot(x=data['Cluster'], y=data['Average_Spend'])
plt.title('Average Spend by Cluster')
plt.xlabel('Cluster')
```

```
Text(0.5, 0, 'Cluster')
```