campaign-final

October 23, 2024

[4]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

[5]:
```
campaign = pd.read_csv('g:/CODES/EDA/campaign.csv')
```

[6]:
```
df = campaign.copy()
```

[7]:
```
df.head()
```

[7]:

	ID	Year_Birth	Education	Marital_Status	Income	Kidhome	\
0	1826	1970	Graduation	Divorced	$84,835.00	0	
1	1	1961	Graduation	Single	$57,091.00	0	
2	10476	1958	Graduation	Married	$67,267.00	0	
3	1386	1967	Graduation	Together	$32,474.00	1	
4	5371	1989	Graduation	Single	$21,474.00	1	

	Teenhome	Dt_Customer	Recency	MntWines	…	NumCatalogPurchases	\
0	0	6/16/14	0	189	…	4	
1	0	6/15/14	0	464	…	3	
2	1	5/13/14	0	134	…	2	
3	1	5/11/14	0	10	…	0	
4	0	4/8/14	0	6	…	1	

	NumStorePurchases	NumWebVisitsMonth	AcceptedCmp3	AcceptedCmp4	\
0	6	1	0	0	
1	7	5	0	0	
2	5	2	0	0	
3	2	7	0	0	
4	2	7	1	0	

	AcceptedCmp5	AcceptedCmp1	AcceptedCmp2	Complain	Country
0	0	0	0	0	SP
1	0	0	1	0	CA
2	0	0	0	0	US
3	0	0	0	0	AUS

```
4             0             0             0         0       SP

[5 rows x 27 columns]
```

[8]:
```
df.tail()
```

[8]:
```
         ID  Year_Birth   Education Marital_Status      Income  Kidhome  \
2234  10142        1976         PhD       Divorced  $66,476.00        0
2235   5263        1977    2n Cycle        Married  $31,056.00        1
2236     22        1976  Graduation       Divorced  $46,310.00        1
2237    528        1978  Graduation        Married  $65,819.00        0
2238   4070        1969         PhD        Married  $94,871.00        0

      Teenhome Dt_Customer  Recency  MntWines  …  NumCatalogPurchases  \
2234         1      3/7/13       99       372  …                    2
2235         0     1/22/13       99         5  …                    0
2236         0     12/3/12       99       185  …                    1
2237         0    11/29/12       99       267  …                    4
2238         2      9/1/12       99       169  …                    5

      NumStorePurchases  NumWebVisitsMonth  AcceptedCmp3  AcceptedCmp4  \
2234                 11                  4             0             0
2235                  3                  8             0             0
2236                  5                  8             0             0
2237                 10                  3             0             0
2238                  4                  7             0             1

      AcceptedCmp5  AcceptedCmp1  AcceptedCmp2  Complain  Country
2234             0             0             0         0       US
2235             0             0             0         0       SP
2236             0             0             0         0       SP
2237             0             0             0         0      IND
2238             1             0             0         0       CA

[5 rows x 27 columns]
```

[9]:
```
df.shape
```

[9]:
```
(2239, 27)
```

[10]:
```
df.size
```

[10]:
```
60453
```

[11]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
```

```
RangeIndex: 2239 entries, 0 to 2238
Data columns (total 27 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   ID                   2239 non-null   int64
 1   Year_Birth           2239 non-null   int64
 2   Education            2239 non-null   object
 3   Marital_Status       2239 non-null   object
 4   Income               2239 non-null   object
 5   Kidhome              2239 non-null   int64
 6   Teenhome             2239 non-null   int64
 7   Dt_Customer          2239 non-null   object
 8   Recency              2239 non-null   int64
 9   MntWines             2239 non-null   int64
 10  MntFruits            2239 non-null   int64
 11  MntMeatProducts      2239 non-null   int64
 12  MntFishProducts      2239 non-null   int64
 13  MntSweetProducts     2239 non-null   int64
 14  MntGoldProds         2239 non-null   int64
 15  NumDealsPurchases    2239 non-null   int64
 16  NumWebPurchases      2239 non-null   int64
 17  NumCatalogPurchases  2239 non-null   int64
 18  NumStorePurchases    2239 non-null   int64
 19  NumWebVisitsMonth    2239 non-null   int64
 20  AcceptedCmp3         2239 non-null   int64
 21  AcceptedCmp4         2239 non-null   int64
 22  AcceptedCmp5         2239 non-null   int64
 23  AcceptedCmp1         2239 non-null   int64
 24  AcceptedCmp2         2239 non-null   int64
 25  Complain             2239 non-null   int64
 26  Country              2239 non-null   object
dtypes: int64(22), object(5)
memory usage: 472.4+ KB
```

[12]:
```
df.isnull().sum()
```

[12]:
```
ID                   0
Year_Birth           0
Education            0
Marital_Status       0
Income               0
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
Country                0
dtype: int64
```

[13]:
```
df[df.duplicated()].sum()
```

[13]:
```
ID                     0
Year_Birth             0
Education              0
Marital_Status         0
Income                 0
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
Country                0
```

```
dtype: object
```

[14]:
```
df.describe().T
```

[14]:

	count	mean	std	min	25%	50%	75%	max
ID	2239.0	5590.444841	3246.372471	0.0	2827.5	5455.0	8423.5	11191.0
Year_Birth	2239.0	1968.802144	11.985494	1893.0	1959.0	1970.0	1977.0	1996.0
Kidhome	2239.0	0.443948	0.538390	0.0	0.0	0.0	1.0	2.0
Teenhome	2239.0	0.506476	0.544555	0.0	0.0	0.0	1.0	2.0
Recency	2239.0	49.121036	28.963662	0.0	24.0	49.0	74.0	99.0
MntWines	2239.0	304.067441	336.614830	0.0	24.0	174.0	504.5	1493.0
MntFruits	2239.0	26.307727	39.781468	0.0	1.0	8.0	33.0	199.0
MntMeatProducts	2239.0	167.016525	225.743829	0.0	16.0	67.0	232.0	1725.0
MntFishProducts	2239.0	37.538633	54.637617	0.0	3.0	12.0	50.0	259.0
MntSweetProducts	2239.0	27.074587	41.286043	0.0	1.0	8.0	33.0	263.0
MntGoldProds	2239.0	44.036177	52.174700	0.0	9.0	24.0	56.0	362.0
NumDealsPurchases	2239.0	2.324252	1.932345	0.0	1.0	2.0	3.0	15.0
NumWebPurchases	2239.0	4.085306	2.779240	0.0	2.0	4.0	6.0	27.0
NumCatalogPurchases	2239.0	2.662796	2.923542	0.0	0.0	2.0	4.0	28.0
NumStorePurchases	2239.0	5.791425	3.251149	0.0	3.0	5.0	8.0	13.0
NumWebVisitsMonth	2239.0	5.316213	2.427144	0.0	3.0	6.0	7.0	20.0
AcceptedCmp3	2239.0	0.072800	0.259867	0.0	0.0	0.0	0.0	1.0
AcceptedCmp4	2239.0	0.074587	0.262782	0.0	0.0	0.0	0.0	1.0
AcceptedCmp5	2239.0	0.072800	0.259867	0.0	0.0	0.0	0.0	1.0
AcceptedCmp1	2239.0	0.064314	0.245367	0.0	0.0	0.0	0.0	1.0
AcceptedCmp2	2239.0	0.013399	0.115001	0.0	0.0	0.0	0.0	1.0
Complain	2239.0	0.009379	0.096412	0.0	0.0	0.0	0.0	1.0

```
AcceptedCmp5            0.0      1.0
AcceptedCmp1            0.0      1.0
AcceptedCmp2            0.0      1.0
Complain                0.0      1.0
```

[15]:
```
# Data Cleaning
df['Income'] = df['Income'].replace({'\$': '', ',': ''}, regex=True).
 ↪astype(float)
```

```
<>:2: SyntaxWarning: invalid escape sequence '\$'
<>:2: SyntaxWarning: invalid escape sequence '\$'
C:\Users\Teju\AppData\Local\Temp\ipykernel_11484\3585520216.py:2: SyntaxWarning:
invalid escape sequence '\$'
  df['Income'] = df['Income'].replace({'\$': '', ',': ''},
regex=True).astype(float)
```

[16]:
```
# Convert 'Dt_Customer' to datetime
df['Dt_Customer'] = pd.to_datetime(df['Dt_Customer'], format='%Y-%m-%d')
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[16], line 2
      1 # Convert 'Dt_Customer' to datetime
----> 2 df['Dt_Customer'] = pd.to_datetime(df['Dt_Customer'], format='%Y-%m-%d'

File c:
 ↪\Users\Teju\AppData\Local\Programs\Python\Python312\Lib\site-packages\pandas\ :ore\tools\dat
 ↪py:1067, in to_datetime(arg, errors, dayfirst, yearfirst, utc, format, exact,␣
 ↪unit, infer_datetime_format, origin, cache)
   1065         result = arg.map(cache_array)
   1066     else:
-> 1067         values = convert_listlike(arg._values, format)
   1068         result = arg._constructor(values, index=arg.index, name=arg.nam )
   1069 elif isinstance(arg, (ABCDataFrame, abc.MutableMapping)):

File c:
 ↪\Users\Teju\AppData\Local\Programs\Python\Python312\Lib\site-packages\pandas\ :ore\tools\dat
 ↪py:433, in _convert_listlike_datetimes(arg, format, name, utc, unit, errors,␣
 ↪dayfirst, yearfirst, exact)
    431 # `format` could be inferred, or user didn't ask for mixed-format␣
 ↪parsing.
    432 if format is not None and format != "mixed":
--> 433     return␣
 ↪_array_strptime_with_fallback(arg, name, utc, format, exact, errors)
    435 result, tz_parsed = objects_to_datetime64(
    436     arg,
    437     dayfirst=dayfirst,
   (…)
```

```
    441     allow_object=True,
    442 )
    444 if tz_parsed is not None:
    445     # We can take a shortcut since the datetime64 numpy array
    446     # is in UTC

File c:
 ↪\Users\Teju\AppData\Local\Programs\Python\Python312\Lib\site-packages\pandas\:ore\tools\dat
 ↪py:467, in _array_strptime_with_fallback(arg, name, utc, fmt, exact, errors)
    456 def _array_strptime_with_fallback(
    457     arg,
    458     name,
   (…)
    462     errors: str,
    463 ) -> Index:
    464     """
    465     Call array_strptime, with fallback behavior depending on 'errors'.
    466     """
--> 467     result, tz_out =␣
 ↪array_strptime(arg, fmt, exact=exact, errors=errors, utc=utc)
    468     if tz_out is not None:
    469         unit = np.datetime_data(result.dtype)[0]

File strptime.pyx:501, in pandas._libs.tslibs.strptime.array_strptime()

File strptime.pyx:451, in pandas._libs.tslibs.strptime.array_strptime()

File strptime.pyx:583, in pandas._libs.tslibs.strptime._parse_with_format()

ValueError: time data "6/16/14" doesn't match format "%Y-%m-%d", at position 0.␣
 ↪You might want to try:
    - passing `format` if your strings have a consistent format;
    - passing `format='ISO8601'` if your strings are all ISO8601 but not␣
 ↪necessarily in exactly the same format;
    - passing `format='mixed'`, and the format will be inferred for each elemen␣
 ↪individually. You might want to use `dayfirst` alongside this.
```

[23]:
```
df.describe().T
```

[23]:

	count	mean	\
ID	2239.0	5590.444841	
Year_Birth	2239.0	1968.802144	
Income	2215.0	51969.8614	
Kidhome	2239.0	0.443948	
Teenhome	2239.0	0.506476	
Dt_Customer	2239	2013-07-10 10:26:25.350603008	
Recency	2239.0	49.121036	

MntWines	2239.0	304.067441	
MntFruits	2239.0	26.307727	
MntMeatProducts	2239.0	167.016525	
MntFishProducts	2239.0	37.538633	
MntSweetProducts	2239.0	27.074587	
MntGoldProds	2239.0	44.036177	
NumDealsPurchases	2239.0	2.324252	
NumWebPurchases	2239.0	4.085306	
NumCatalogPurchases	2239.0	2.662796	
NumStorePurchases	2239.0	5.791425	
NumWebVisitsMonth	2239.0	5.316213	
AcceptedCmp3	2239.0	0.0728	
AcceptedCmp4	2239.0	0.074587	
AcceptedCmp5	2239.0	0.0728	
AcceptedCmp1	2239.0	0.064314	
AcceptedCmp2	2239.0	0.013399	
Complain	2239.0	0.009379	

	min	25%	\
ID	0.0	2827.5	
Year_Birth	1893.0	1959.0	
Income	1730.0	35284.0	
Kidhome	0.0	0.0	
Teenhome	0.0	0.0	
Dt_Customer	2012-07-30 00:00:00	2013-01-16 00:00:00	
Recency	0.0	24.0	
MntWines	0.0	24.0	
MntFruits	0.0	1.0	
MntMeatProducts	0.0	16.0	
MntFishProducts	0.0	3.0	
MntSweetProducts	0.0	1.0	
MntGoldProds	0.0	9.0	
NumDealsPurchases	0.0	1.0	
NumWebPurchases	0.0	2.0	
NumCatalogPurchases	0.0	0.0	
NumStorePurchases	0.0	3.0	
NumWebVisitsMonth	0.0	3.0	
AcceptedCmp3	0.0	0.0	
AcceptedCmp4	0.0	0.0	
AcceptedCmp5	0.0	0.0	
AcceptedCmp1	0.0	0.0	
AcceptedCmp2	0.0	0.0	
Complain	0.0	0.0	

	50%	75%	\
ID	5455.0	8423.5	
Year_Birth	1970.0	1977.0	

Income	51373.0	68487.0
Kidhome	0.0	1.0
Teenhome	0.0	1.0
Dt_Customer	2013-07-09 00:00:00	2013-12-30 12:00:00
Recency	49.0	74.0
MntWines	174.0	504.5
MntFruits	8.0	33.0
MntMeatProducts	67.0	232.0
MntFishProducts	12.0	50.0
MntSweetProducts	8.0	33.0
MntGoldProds	24.0	56.0
NumDealsPurchases	2.0	3.0
NumWebPurchases	4.0	6.0
NumCatalogPurchases	2.0	4.0
NumStorePurchases	5.0	8.0
NumWebVisitsMonth	6.0	7.0
AcceptedCmp3	0.0	0.0
AcceptedCmp4	0.0	0.0
AcceptedCmp5	0.0	0.0
AcceptedCmp1	0.0	0.0
AcceptedCmp2	0.0	0.0
Complain	0.0	0.0

	max	std
ID	11191.0	3246.372471
Year_Birth	1996.0	11.985494
Income	162397.0	21526.320095
Kidhome	2.0	0.53839
Teenhome	2.0	0.544555
Dt_Customer	2014-06-29 00:00:00	NaN
Recency	99.0	28.963662
MntWines	1493.0	336.61483
MntFruits	199.0	39.781468
MntMeatProducts	1725.0	225.743829
MntFishProducts	259.0	54.637617
MntSweetProducts	263.0	41.286043
MntGoldProds	362.0	52.1747
NumDealsPurchases	15.0	1.932345
NumWebPurchases	27.0	2.77924
NumCatalogPurchases	28.0	2.923542
NumStorePurchases	13.0	3.251149
NumWebVisitsMonth	20.0	2.427144
AcceptedCmp3	1.0	0.259867
AcceptedCmp4	1.0	0.262782
AcceptedCmp5	1.0	0.259867
AcceptedCmp1	1.0	0.245367
AcceptedCmp2	1.0	0.115001

```
Complain                              1.0      0.096412
```

[24]:
```
df['Year_Customer'] = df['Dt_Customer'].dt.year
df['Month_Customer'] = df['Dt_Customer'].dt.month
```

[25]:
```
df.sample(5)
```

[25]:

	ID	Year_Birth	Education	Marital_Status	Income	Kidhome	Teenhome	\
1160	5989	1959	2n Cycle	Divorced	78353.0	0	1	
1871	7326	1971	Master	Married	56850.0	0	1	
1127	10380	1972	Master	Married	37787.0	1	0	
510	4971	1962	PhD	Together	31497.0	0	1	
2005	6974	1972	PhD	Together	83443.0	0	0	

	Dt_Customer	Recency	MntWines	…	NumWebVisitsMonth	AcceptedCmp3	\
1160	2013-04-16	51	752	…	8	0	
1871	2014-03-23	83	34	…	2	0	
1127	2013-09-20	50	40	…	8	0	
510	2012-12-06	22	108	…	8	0	
2005	2013-12-31	89	518	…	2	0	

	AcceptedCmp4	AcceptedCmp5	AcceptedCmp1	AcceptedCmp2	Complain	\
1160	0	0	0	0	0	
1871	0	0	0	0	0	
1127	0	0	0	0	0	
510	0	0	0	0	0	
2005	0	0	0	0	0	

	Country	Year_Customer	Month_Customer
1160	CA	2013	4
1871	CA	2014	3
1127	IND	2013	9
510	SA	2012	12
2005	GER	2013	12

[5 rows x 29 columns]

[26]:
```
df['Education'].value_counts()
```

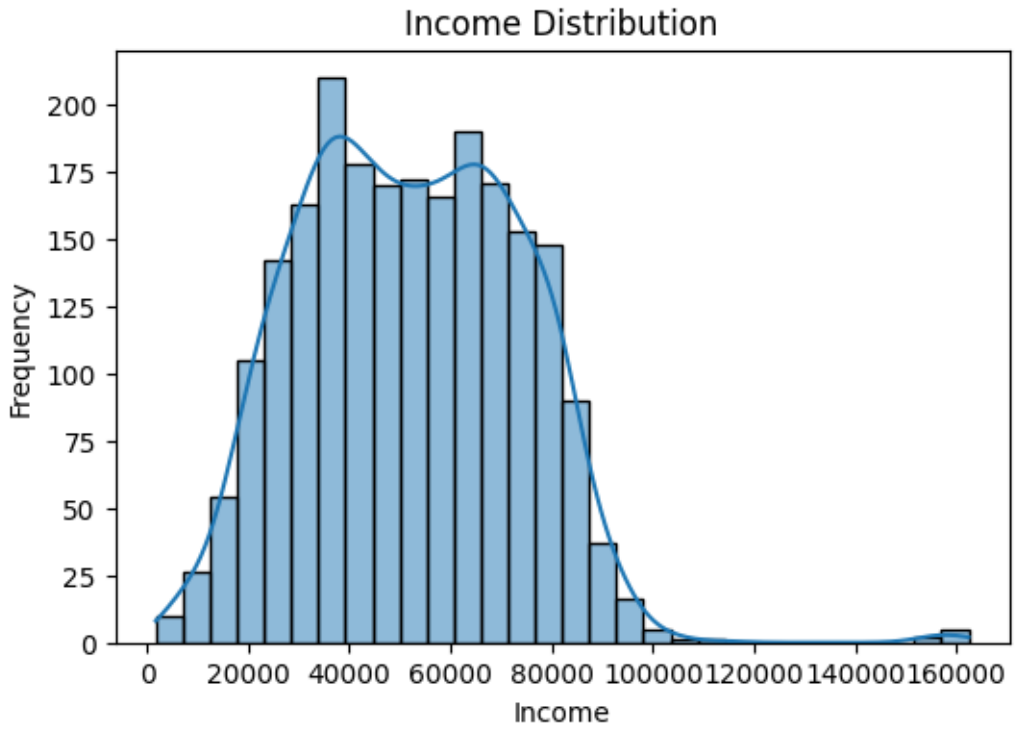
[26]:
```
Education
Graduation    1126
PhD            486
Master         370
2n Cycle       203
Basic           54
Name: count, dtype: int64
```

[27]:
```
df['Marital_Status'].value_counts()
```

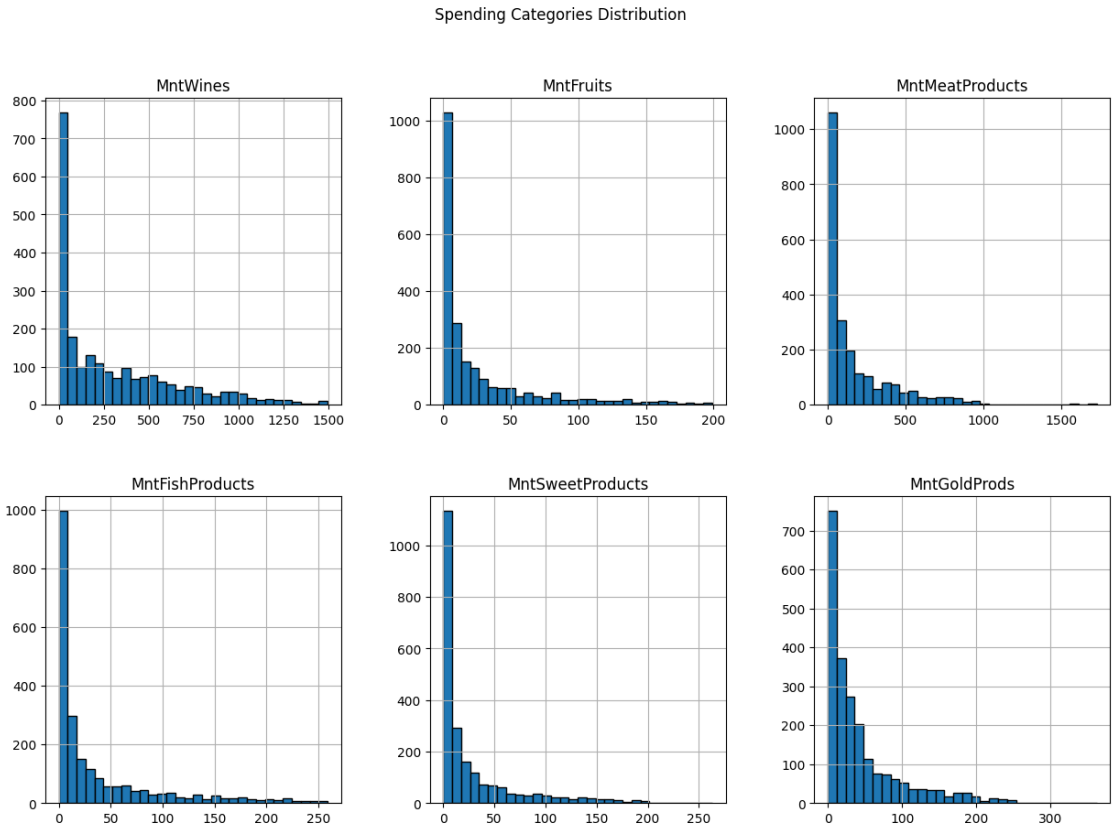
[27]:
```
Marital_Status
Married     864
Together    579
Single      480
Divorced    232
Widow        77
Alone         3
YOLO          2
Absurd        2
Name: count, dtype: int64
```

1 Visualizations

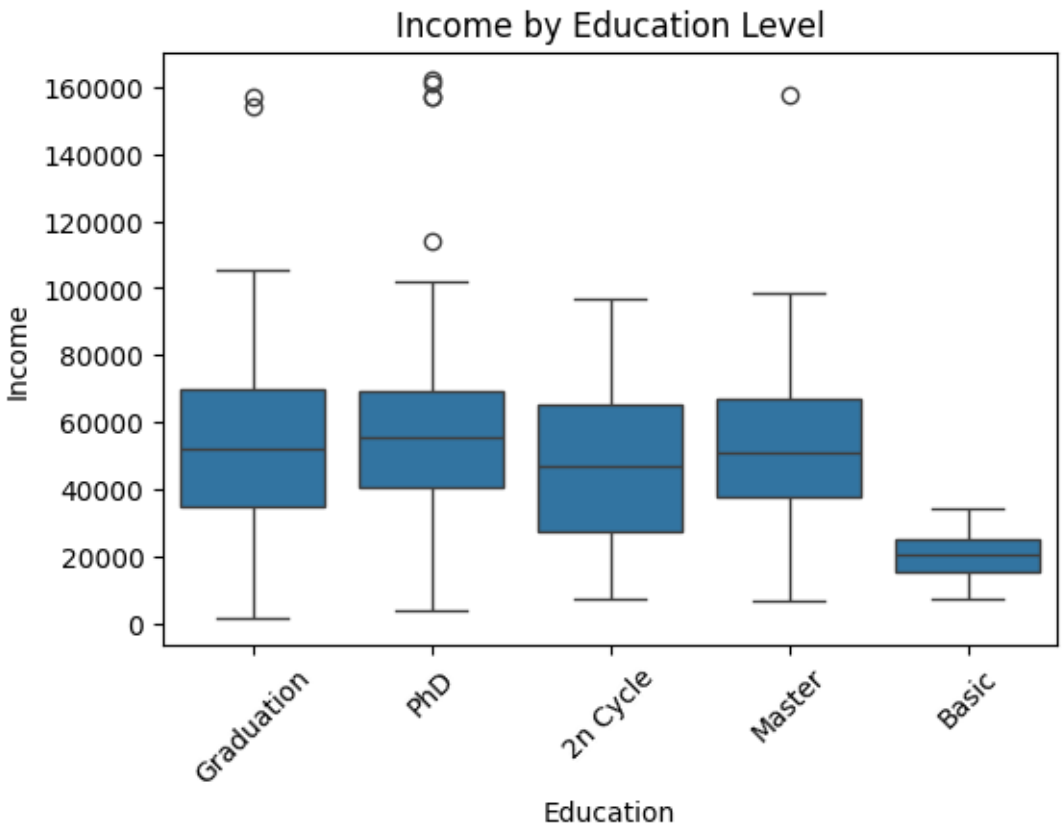
[30]:
```
## 1. Income Distribution
plt.figure(figsize=(6, 4))
sns.histplot(df['Income'], bins=30, kde=True)
plt.title('Income Distribution')
plt.xlabel('Income')
plt.ylabel('Frequency')
plt.show()
```



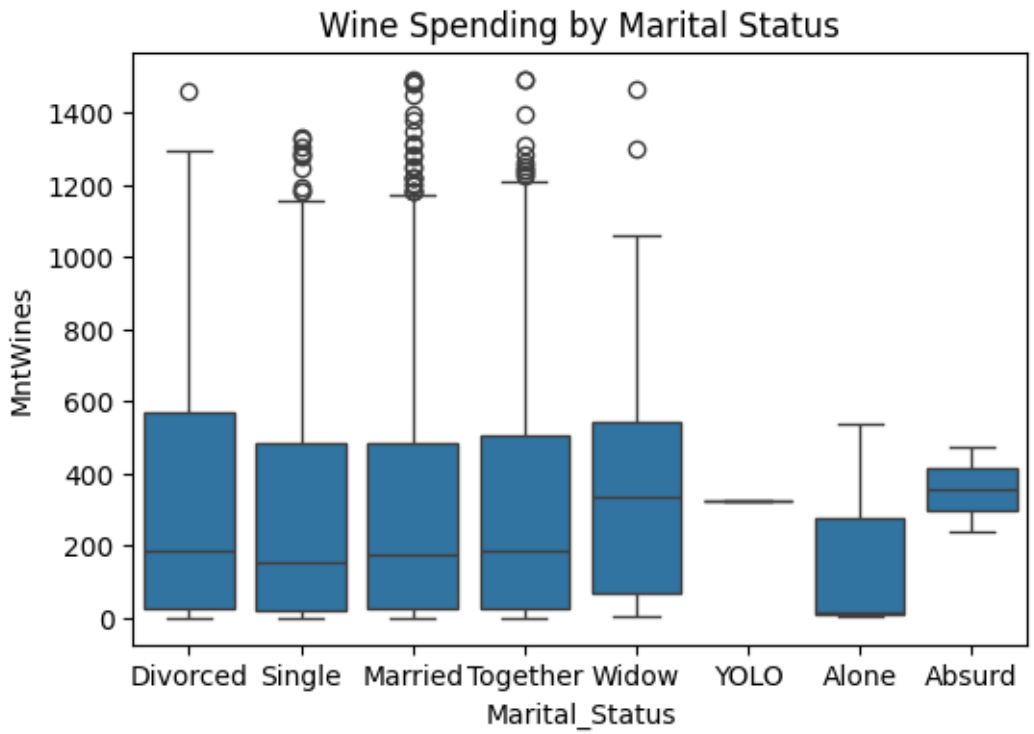
[31]:
```
## 2. Spending Categories Distribution
spending_cols = ['MntWines', 'MntFruits', 'MntMeatProducts', 'MntFishProducts',␣
 ↪'MntSweetProducts', 'MntGoldProds']
df[spending_cols].hist(bins=30, figsize=(15, 10), layout=(2, 3),␣
 ↪edgecolor='black')
plt.suptitle('Spending Categories Distribution')
plt.show()
```



[35]:
```
## 3. Box Plot: Income vs. Education
plt.figure(figsize=(6, 4))
sns.boxplot(x='Education', y='Income', data=df)
plt.title('Income by Education Level')
plt.xticks(rotation=45)
plt.show()
```

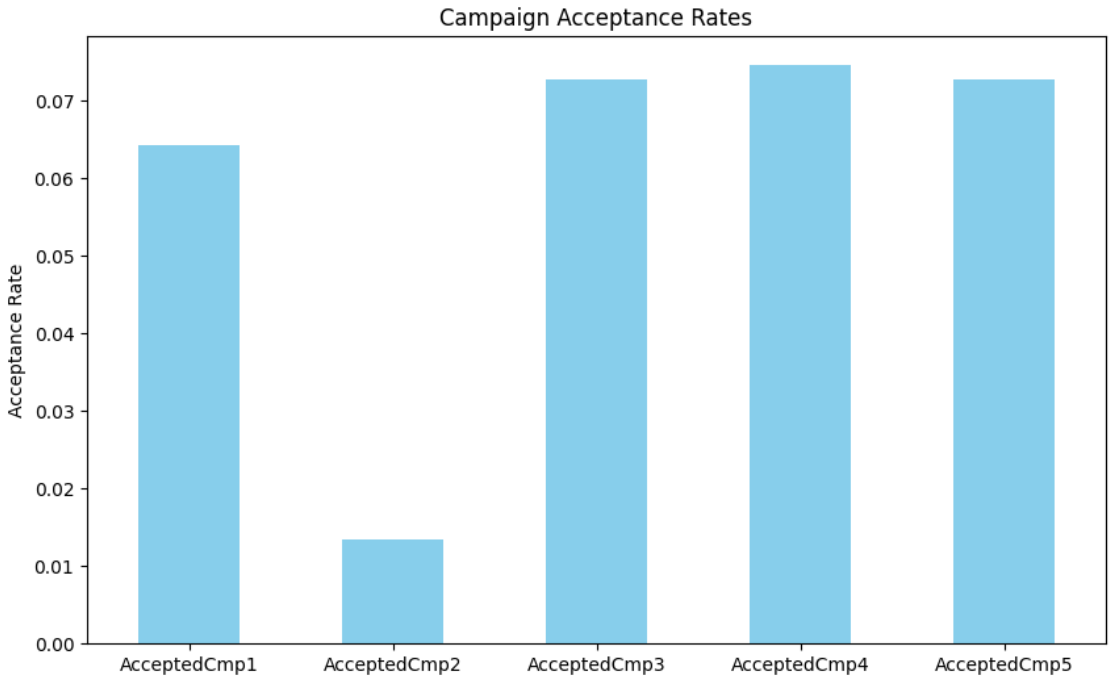


[38]:
```
## 4. Box Plot: Spending vs. Marital Status
plt.figure(figsize=(6, 4))
sns.boxplot(x='Marital_Status', y='MntWines', data=df)
plt.title('Wine Spending by Marital Status')
plt.show()
```



[ ]:

[17]:
```
## 5. Campaign Acceptance Rates
campaign_cols = ['AcceptedCmp1', 'AcceptedCmp2', 'AcceptedCmp3',␣
 ↪'AcceptedCmp4', 'AcceptedCmp5']
acceptance_rates = df[campaign_cols].mean()
acceptance_rates.plot(kind='bar', figsize=(10, 6), color='skyblue')
plt.title('Campaign Acceptance Rates')
plt.ylabel('Acceptance Rate')
plt.xticks(rotation=0)
plt.show()
```



[18]:
```
# Customer Segmentation based on Campaign Acceptance and Spending
high_spenders = df[df[spending_cols].sum(axis=1) > df[spending_cols].
 ↪sum(axis=1).mean()]
print("High Spenders who accepted campaigns:")
high_spender_accept = high_spenders[high_spenders[campaign_cols].sum(axis=1) >␣
 ↪0]
high_spender_accept.shape
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[18], line 2
      1 # Customer Segmentation based on Campaign Acceptance and Spending
----> 2 high_spenders = df[df[spending_cols].sum(axis=1) > df[spending_cols].
 ↪sum(axis=1).mean()]
      3 print("High Spenders who accepted campaigns:")
      4 high_spender_accept = high_spenders[high_spenders[campaign_cols].
 ↪sum(axis=1) > 0]

NameError: name 'spending_cols' is not defined
```

[49]:
```
high_spender_accept.sample(20)
```

[49]:

	ID	Year_Birth	Education	Marital_Status	Income	Kidhome	\
1758	5538	1975	Graduation	Divorced	83829.0	0	

```
968    4394         1965         PhD        Married   81051.0         0
796    3759         1958  Graduation       Together   65196.0         0
2181   8439         1964  Graduation       Together   63404.0         0
1012   2561         1966  Graduation         Single   63810.0         0
301    7366         1982      Master         Single   75777.0         0
402    7999         1955         PhD       Together   75261.0         0
177    1212         1973  Graduation        Married   52845.0         1
623   10140         1983         PhD       Together   70123.0         0
492    1685         1967         PhD       Together   62981.0         0
536    4261         1946         PhD         Single   82800.0         0
809   10489         1973  Graduation        Married   92955.0         0
574    4310         1944  Graduation        Married   80589.0         0
1243   2798         1977         PhD       Together  102160.0         0
219   10909         1948  Graduation        Married   92344.0         0
1979   6977         1974  Graduation       Together   75702.0         0
1907   1627         1957    2n Cycle       Divorced   77297.0         0
1384   3174         1959  Graduation       Together   87771.0         0
2023   5558         1954         PhD         Single   90933.0         0
253   10240         1949  Graduation       Together   69372.0         0

      Teenhome Dt_Customer  Recency  MntWines  …  NumWebVisitsMonth  \
1758         0  2013-10-08       78       897  …                  1
968          0  2014-05-23       43      1142  …                  2
796          2  2013-07-25       34       743  …                  5
2181         2  2014-06-06       97       734  …                  4
1012         1  2012-11-11       45       977  …                  8
301          0  2013-07-04       12       712  …                  1
402          0  2013-04-23       17      1239  …                  2
177          0  2013-08-13        7       384  …                  6
623          0  2013-09-28       27      1308  …                  3
492          0  2013-03-17       21       796  …                  3
536          0  2012-11-24       23      1006  …                  3
809          0  2013-08-19       35       693  …                  2
574          0  2014-01-22       25       507  …                  1
1243         0  2012-11-02       54       763  …                  4
219          0  2014-01-15        9       992  …                  1
1979         1  2012-10-14       87      1073  …                  6
1907         0  2013-01-26       84       408  …                  4
1384         1  2013-05-22       61      1492  …                  6
2023         0  2014-03-31       90      1020  …                  1
253          0  2013-02-19       10       997  …                  4

      AcceptedCmp3  AcceptedCmp4  AcceptedCmp5  AcceptedCmp1  AcceptedCmp2  \
1758             1             0             1             1             0
968              0             1             1             0             0
796              1             0             0             0             0
2181             0             0             0             1             0
```

1012	0	1	0	0	0
301	0	1	1	0	0
402	0	1	1	0	0
177	1	0	0	0	0
623	0	1	0	0	1
492	0	1	0	0	0
536	0	0	1	1	0
809	0	0	1	1	0
574	0	0	0	1	0
1243	0	1	1	1	0
219	1	0	1	0	0
1979	0	0	1	0	0
1907	0	0	0	1	0
1384	0	1	1	1	1
2023	0	0	1	0	0
253	0	1	1	0	0

	Complain	Country	Year_Customer	Month_Customer
1758	0	SP	2013	10
968	0	SP	2014	5
796	0	SP	2013	7
2181	0	SP	2014	6
1012	0	GER	2012	11
301	0	IND	2013	7
402	0	SP	2013	4
177	0	SP	2013	8
623	0	IND	2013	9
492	0	CA	2013	3
536	0	SA	2012	11
809	0	SA	2013	8
574	0	AUS	2014	1
1243	0	SA	2012	11
219	0	AUS	2014	1
1979	0	SP	2012	10
1907	0	SP	2013	1
1384	0	SP	2013	5
2023	0	SP	2014	3
253	0	CA	2013	2

[20 rows x 29 columns]

1.1 Feature Enginnering

[19]:
```
current_year = pd.to_datetime("now").year
df['Age'] = current_year - df['Year_Birth']
df.head()
```

[19]:
```
      ID  Year_Birth   Education Marital_Status   Income  Kidhome  Teenhome  \
0   1826        1970  Graduation       Divorced  84835.0        0         0
1      1        1961  Graduation         Single  57091.0        0         0
2  10476        1958  Graduation        Married  67267.0        0         1
3   1386        1967  Graduation       Together  32474.0        1         1
4   5371        1989  Graduation         Single  21474.0        1         0

  Dt_Customer  Recency  MntWines  …  NumStorePurchases  NumWebVisitsMonth  \
0     6/16/14        0       189  …                  6                  1
1     6/15/14        0       464  …                  7                  5
2     5/13/14        0       134  …                  5                  2
3     5/11/14        0        10  …                  2                  7
4      4/8/14        0         6  …                  2                  7

   AcceptedCmp3  AcceptedCmp4  AcceptedCmp5  AcceptedCmp1  AcceptedCmp2  \
0             0             0             0             0             0
1             0             0             0             0             1
2             0             0             0             0             0
3             0             0             0             0             0
4             1             0             0             0             0

   Complain  Country  Age
0         0       SP   54
1         0       CA   63
2         0       US   66
3         0      AUS   57
4         0       SP   35

[5 rows x 28 columns]
```

[20]:
```
# Income Binning
bins = [0, 30000, 60000, 90000, float('inf')]
labels = ['Low', 'Medium', 'High', 'Very High']
df['Income_Category'] = pd.cut(df['Income'], bins=bins, labels=labels)
df.head()
```

[20]:
```
      ID  Year_Birth   Education Marital_Status   Income  Kidhome  Teenhome  \
0   1826        1970  Graduation       Divorced  84835.0        0         0
1      1        1961  Graduation         Single  57091.0        0         0
2  10476        1958  Graduation        Married  67267.0        0         1
3   1386        1967  Graduation       Together  32474.0        1         1
4   5371        1989  Graduation         Single  21474.0        1         0

  Dt_Customer  Recency  MntWines  …  NumWebVisitsMonth  AcceptedCmp3  \
0     6/16/14        0       189  …                  1             0
1     6/15/14        0       464  …                  5             0
2     5/13/14        0       134  …                  2             0
```

```
3     5/11/14        0        10  …                 7             0
4      4/8/14        0         6  …                 7             1

   AcceptedCmp4  AcceptedCmp5  AcceptedCmp1  AcceptedCmp2  Complain  Country  \
0             0             0             0             0         0       SP
1             0             0             0             1         0       CA
2             0             0             0             0         0       US
3             0             0             0             0         0      AUS
4             0             0             0             0         0       SP

   Age  Income_Category
0   54             High
1   63           Medium
2   66             High
3   57           Medium
4   35              Low

[5 rows x 29 columns]
```

[21]:
```
# Family Size
df['Family_Size'] = df['Kidhome'] + df['Teenhome']
df.head()
```

[21]:
```
      ID  Year_Birth   Education Marital_Status   Income  Kidhome  Teenhome  \
0   1826        1970  Graduation       Divorced  84835.0        0         0
1      1        1961  Graduation         Single  57091.0        0         0
2  10476        1958  Graduation        Married  67267.0        0         1
3   1386        1967  Graduation       Together  32474.0        1         1
4   5371        1989  Graduation         Single  21474.0        1         0

  Dt_Customer  Recency  MntWines  …  AcceptedCmp3  AcceptedCmp4  \
0     6/16/14        0       189  …             0             0
1     6/15/14        0       464  …             0             0
2     5/13/14        0       134  …             0             0
3     5/11/14        0        10  …             0             0
4      4/8/14        0         6  …             1             0

   AcceptedCmp5  AcceptedCmp1  AcceptedCmp2  Complain  Country  Age  \
0             0             0             0         0       SP   54
1             0             0             1         0       CA   63
2             0             0             0         0       US   66
3             0             0             0         0      AUS   57
4             0             0             0         0       SP   35

   Income_Category  Family_Size
0             High            0
1           Medium            0
```

```
2              High            1
3            Medium            2
4               Low            1

[5 rows x 30 columns]
```

[22]:
```
# Total Spending
spending_cols = ['MntWines', 'MntFruits', 'MntMeatProducts', 'MntFishProducts',␣
 ↪'MntSweetProducts', 'MntGoldProds']
df['Total_Spending'] = df[spending_cols].sum(axis=1)
df.head()
```

[22]:
```
      ID  Year_Birth   Education Marital_Status   Income  Kidhome  Teenhome  \
0   1826        1970  Graduation       Divorced  84835.0        0         0
1      1        1961  Graduation         Single  57091.0        0         0
2  10476        1958  Graduation        Married  67267.0        0         1
3   1386        1967  Graduation       Together  32474.0        1         1
4   5371        1989  Graduation         Single  21474.0        1         0

  Dt_Customer  Recency  MntWines  …  AcceptedCmp4  AcceptedCmp5  \
0     6/16/14        0       189  …             0             0
1     6/15/14        0       464  …             0             0
2     5/13/14        0       134  …             0             0
3     5/11/14        0        10  …             0             0
4      4/8/14        0         6  …             0             0

   AcceptedCmp1  AcceptedCmp2  Complain  Country  Age  Income_Category  \
0             0             0         0       SP   54            High
1             0             1         0       CA   63          Medium
2             0             0         0       US   66            High
3             0             0         0      AUS   57          Medium
4             0             0         0       SP   35             Low

   Family_Size  Total_Spending
0            0            1190
1            0             577
2            1             251
3            2              11
4            1              91

[5 rows x 31 columns]
```

[23]:
```
# Campaign Acceptance Count
df['Campaign_Acceptance_Count'] = df[['AcceptedCmp1', 'AcceptedCmp2',
                                      'AcceptedCmp3', 'AcceptedCmp4',␣
 ↪'AcceptedCmp5']].sum(axis=1)
df.head()
```

[23]:
```
      ID  Year_Birth   Education Marital_Status   Income  Kidhome  Teenhome  \
0   1826        1970  Graduation       Divorced  84835.0        0         0
1      1        1961  Graduation         Single  57091.0        0         0
2  10476        1958  Graduation        Married  67267.0        0         1
3   1386        1967  Graduation       Together  32474.0        1         1
4   5371        1989  Graduation         Single  21474.0        1         0

  Dt_Customer  Recency  MntWines  …  AcceptedCmp5  AcceptedCmp1  \
0     6/16/14        0       189  …             0             0
1     6/15/14        0       464  …             0             0
2     5/13/14        0       134  …             0             0
3     5/11/14        0        10  …             0             0
4      4/8/14        0         6  …             0             0

   AcceptedCmp2  Complain  Country  Age  Income_Category  Family_Size  \
0             0         0       SP   54            High            0
1             1         0       CA   63          Medium            0
2             0         0       US   66            High            1
3             0         0      AUS   57          Medium            2
4             0         0       SP   35             Low            1

   Total_Spending  Campaign_Acceptance_Count
0            1190                          0
1             577                          1
2             251                          0
3              11                          0
4              91                          1

[5 rows x 32 columns]
```

[24]:
```
df.columns
```

[24]:
```
Index(['ID', 'Year_Birth', 'Education', 'Marital_Status', 'Income', 'Kidhome',
       'Teenhome', 'Dt_Customer', 'Recency', 'MntWines', 'MntFruits',
       'MntMeatProducts', 'MntFishProducts', 'MntSweetProducts',
       'MntGoldProds', 'NumDealsPurchases', 'NumWebPurchases',
       'NumCatalogPurchases', 'NumStorePurchases', 'NumWebVisitsMonth',
       'AcceptedCmp3', 'AcceptedCmp4', 'AcceptedCmp5', 'AcceptedCmp1',
       'AcceptedCmp2', 'Complain', 'Country', 'Age', 'Income_Category',
       'Family_Size', 'Total_Spending', 'Campaign_Acceptance_Count'],
      dtype='object')
```

[25]:
```
df.drop(['ID', 'Dt_Customer', 'NumDealsPurchases', 'NumWebPurchases',␣
 ↪'NumCatalogPurchases',
        'NumStorePurchases', 'NumWebVisitsMonth'], axis=1, inplace=True)
```

[26]:
```
df.drop(['Complain'], axis=1, inplace=True)
```

[27]:
```
df.head()
```

[27]:

	Year_Birth	Education	Marital_Status	Income	Kidhome	Teenhome	Recency	\
0	1970	Graduation	Divorced	84835.0	0	0	0	
1	1961	Graduation	Single	57091.0	0	0	0	
2	1958	Graduation	Married	67267.0	0	1	0	
3	1967	Graduation	Together	32474.0	1	1	0	
4	1989	Graduation	Single	21474.0	1	0	0	

	MntWines	MntFruits	MntMeatProducts	…	AcceptedCmp4	AcceptedCmp5	\
0	189	104	379	…	0	0	
1	464	5	64	…	0	0	
2	134	11	59	…	0	0	
3	10	0	1	…	0	0	
4	6	16	24	…	0	0	

	AcceptedCmp1	AcceptedCmp2	Country	Age	Income_Category	Family_Size	\
0	0	0	SP	54	High	0	
1	0	1	CA	63	Medium	0	
2	0	0	US	66	High	1	
3	0	0	AUS	57	Medium	2	
4	0	0	SP	35	Low	1	

	Total_Spending	Campaign_Acceptance_Count
0	1190	0
1	577	1
2	251	0
3	11	0
4	91	1

[5 rows x 24 columns]

[28]:
```
# Define numerical columns for outlier detection
spending_cols = ['MntWines', 'MntFruits', 'MntMeatProducts', 'MntFishProducts',␣
 ↪'MntSweetProducts', 'MntGoldProds', 'Income']
```

[29]:
```
def remove_outliers_iqr(df, columns):
    for col in columns:
        Q1 = df[col].quantile(0.25)
        Q3 = df[col].quantile(0.75)
        IQR = Q3 - Q1

        # Define outlier bounds
        lower_bound = Q1 - 1.5 * IQR
        upper_bound = Q3 + 1.5 * IQR

        # Filter out outliers
```

```
        df = df[(df[col] >= lower_bound) & (df[col] <= upper_bound)]
    return df
```

[30]:
```
# Remove outliers from the dataset
cleaned_data = remove_outliers_iqr(df, spending_cols)
```

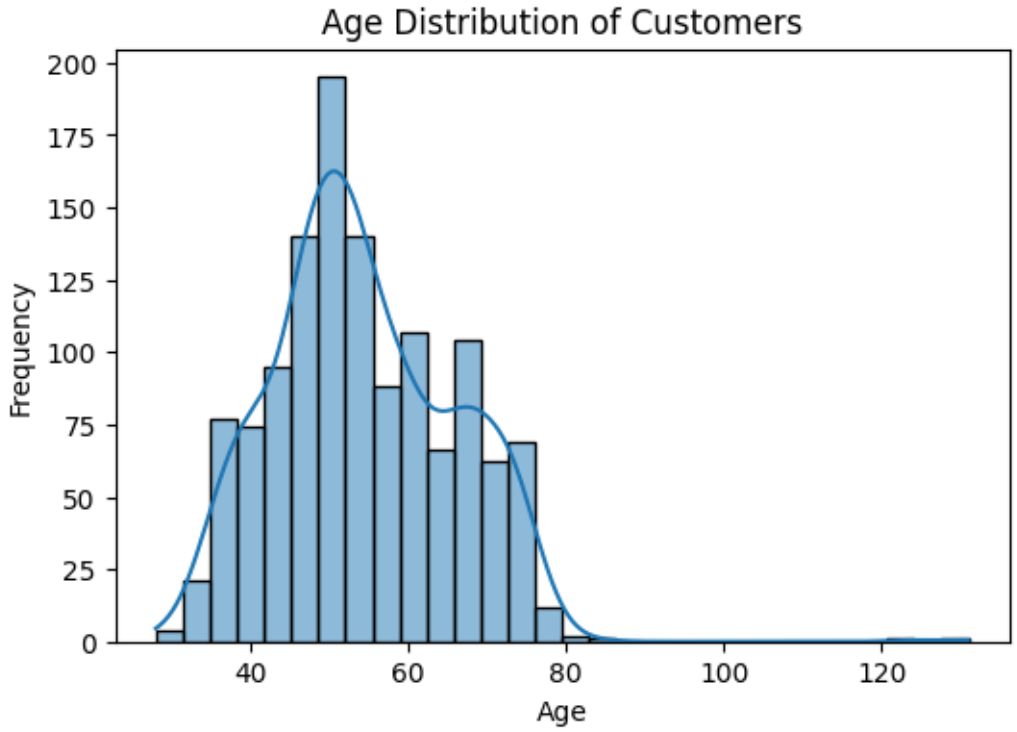
[31]:
```
cleaned_data.describe().T
```

[31]:

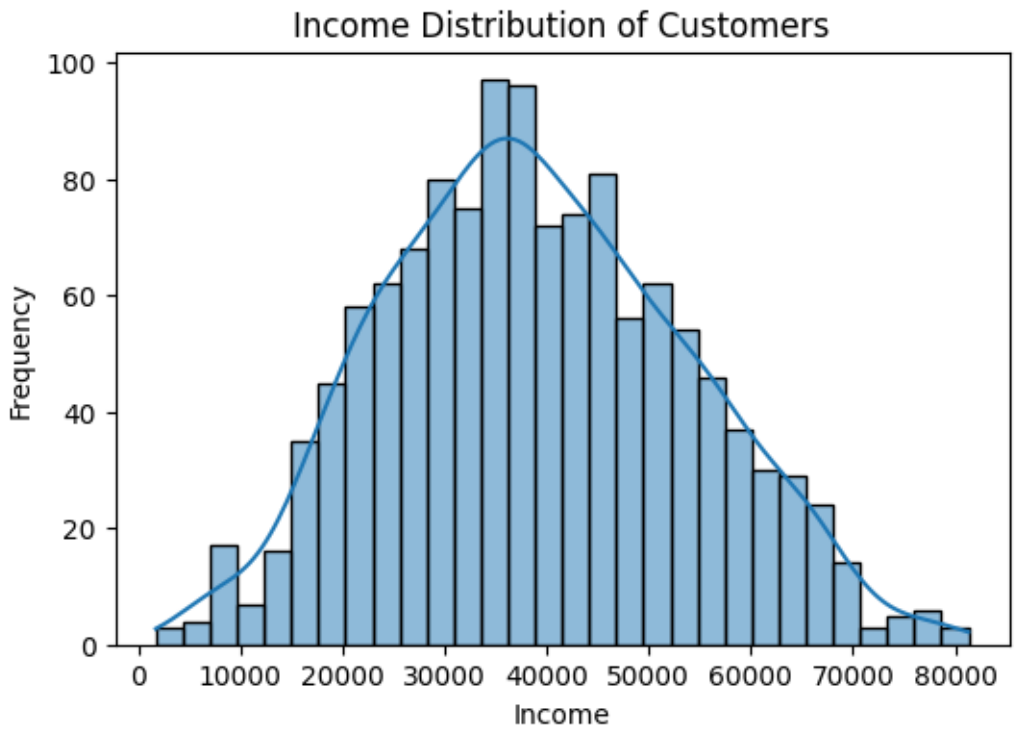
	count	mean	std	min	25%	50%	75%	max
Year_Birth	1259.0	1969.674345	11.508291	1893.0	1961.0	1971.0	1978.0	1996.0
Income	1259.0	38983.922160	14775.086193	1730.0	28284.5	38175.0	49401.0	81300.0
Kidhome	1259.0	0.679110	0.535215	0.0	0.0	1.0	1.0	2.0
Teenhome	1259.0	0.565528	0.547682	0.0	0.0	1.0	1.0	2.0
Recency	1259.0	48.982526	29.154100	0.0	24.0	49.0	75.0	99.0
MntWines	1259.0	133.340747	222.095825	0.0	10.0	31.0	154.0	1181.0
MntFruits	1259.0	5.889595	8.857001	0.0	0.0	3.0	7.0	71.0
MntMeatProducts	1259.0	38.357427	48.669150	0.0	9.0	19.0	49.5	375.0
MntFishProducts	1259.0	7.255759	8.776760	0.0	0.0	4.0	11.0	49.0
MntSweetProducts	1259.0	5.016680	6.101334	0.0	0.0	3.0	7.0	26.0
MntGoldProds	1259.0	16.756156	15.406215	0.0	5.0	12.0	24.0	70.0
AcceptedCmp3	1259.0	0.064337	0.245449	0.0	0.0	0.0	0.0	1.0
AcceptedCmp4	1259.0	0.057188	0.232294	0.0	0.0	0.0	0.0	1.0
AcceptedCmp5	1259.0	0.006354	0.079491	0.0	0.0	0.0	0.0	1.0
AcceptedCmp1	1259.0	0.011120	0.104905	0.0	0.0	0.0	0.0	1.0
AcceptedCmp2	1259.0	0.006354	0.079491	0.0	0.0	0.0	0.0	1.0
Age	1259.0	54.325655	11.508291	28.0				
Family_Size	1259.0	1.244639	0.684051	0.0				
Total_Spending	1259.0	206.616362	271.897223	5.0				
Campaign_Acceptance_Count	1259.0	0.145353	0.405056	0.0				

Age	46.0	53.0	63.0	131.0
Family_Size	1.0	1.0	2.0	3.0
Total_Spending	44.0	81.0	264.0	1513.0
Campaign_Acceptance_Count	0.0	0.0	0.0	4.0

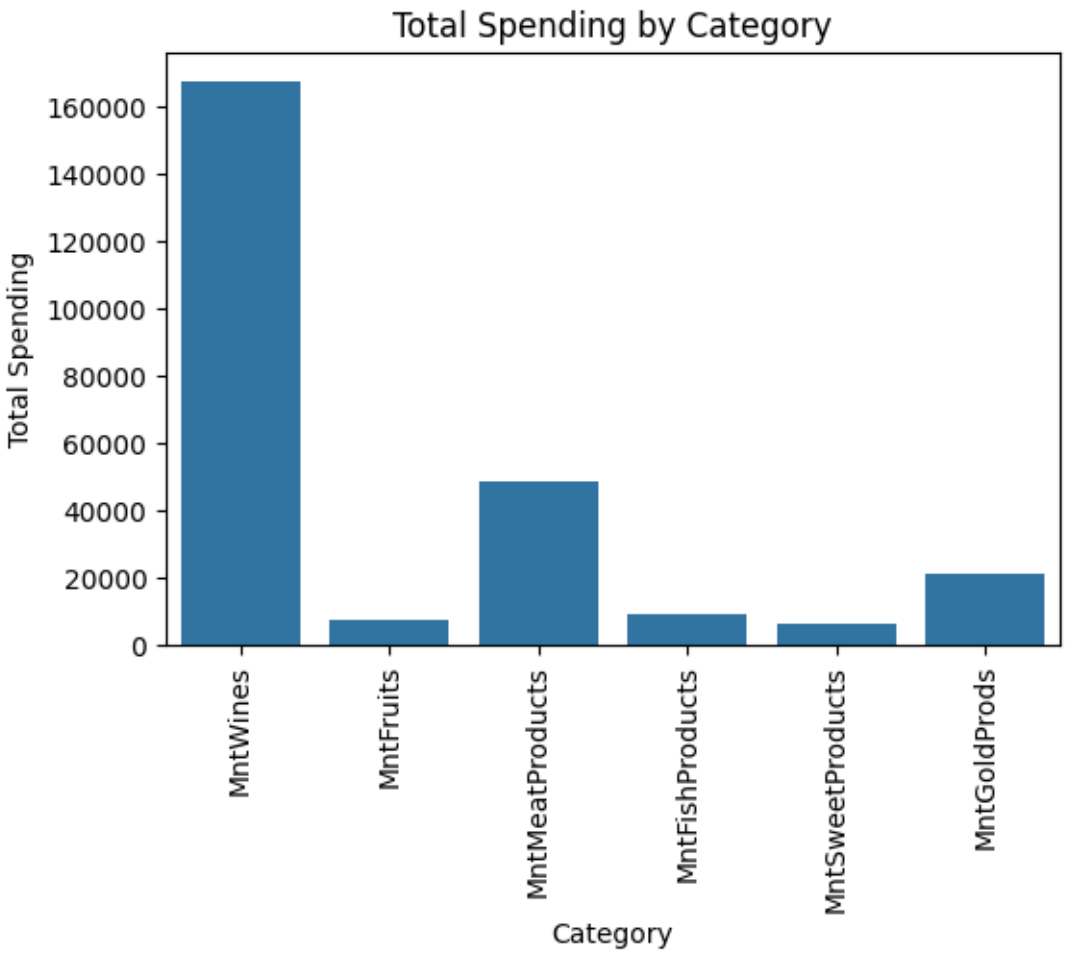
[32]:
```
## 1. Distribution of Age
plt.figure(figsize=(6, 4))
sns.histplot(current_year - cleaned_data['Year_Birth'], bins=30, kde=True)
plt.title('Age Distribution of Customers')
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.show()
```



[33]:
```
## 2. Income Distribution
plt.figure(figsize=(6, 4))
sns.histplot(cleaned_data['Income'], bins=30, kde=True)
plt.title('Income Distribution of Customers')
plt.xlabel('Income')
plt.ylabel('Frequency')
plt.show()
```

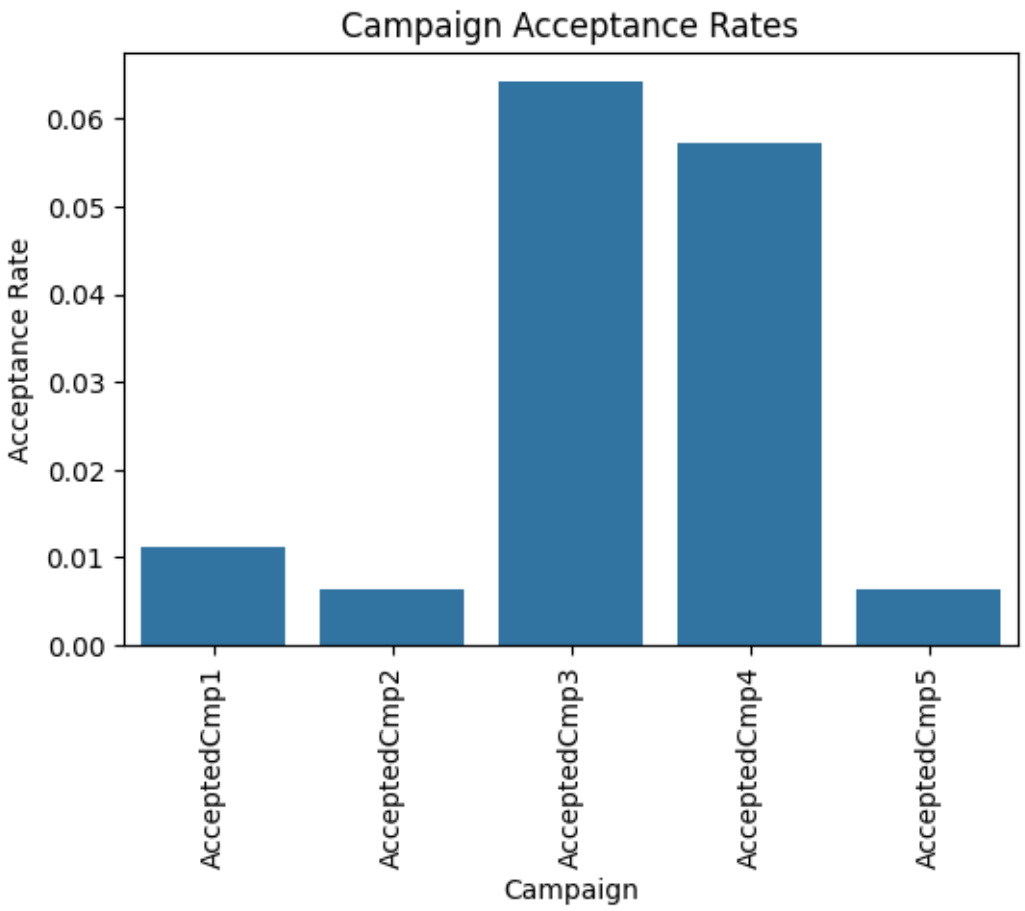


[34]:
```
## 3. Spending by Category
spending_cols = ['MntWines', 'MntFruits', 'MntMeatProducts', 'MntFishProducts',␣
↪'MntSweetProducts', 'MntGoldProds']
spending_data = cleaned_data[spending_cols].sum().reset_index()
spending_data.columns = ['Category', 'Total Spending']
plt.figure(figsize=(6, 4))
sns.barplot(x='Category', y='Total Spending', data=spending_data)
plt.title('Total Spending by Category')
plt.xticks(rotation=90)
plt.show()
```

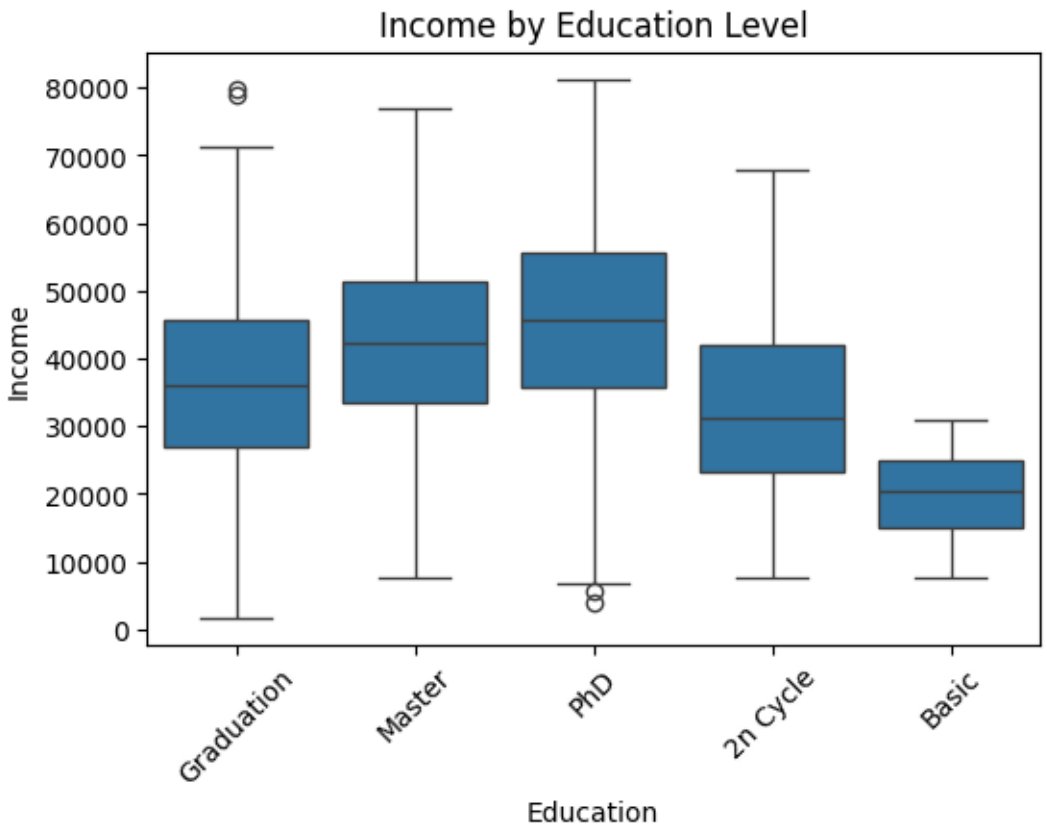


[35]:
```
## 4. Campaign Acceptance Rates
campaign_cols = ['AcceptedCmp1', 'AcceptedCmp2', 'AcceptedCmp3',␣
 ↪'AcceptedCmp4', 'AcceptedCmp5']
acceptance_rates = cleaned_data[campaign_cols].mean().reset_index()
acceptance_rates.columns = ['Campaign', 'Acceptance Rate']

plt.figure(figsize=(6, 4))
sns.barplot(x='Campaign', y='Acceptance Rate', data=acceptance_rates)
plt.title('Campaign Acceptance Rates')
plt.xticks(rotation=90)
plt.show()
```

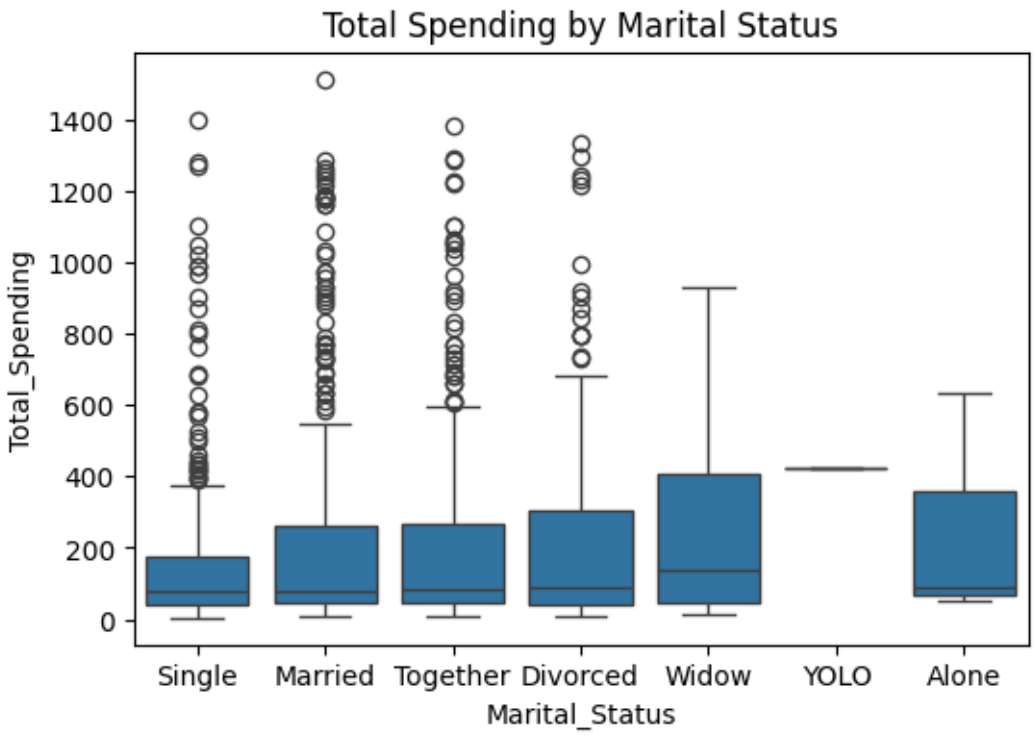


[36]:
```
## 5. Box Plot: Income vs. Education
plt.figure(figsize=(6, 4))
sns.boxplot(x='Education', y='Income', data=cleaned_data)
plt.title('Income by Education Level')
plt.xticks(rotation=45)
plt.show()
```



[37]:
```
## 6. Box Plot: Total Spending vs. Marital Status
cleaned_data['Total_Spending'] = cleaned_data[['MntWines', 'MntFruits',␣
 ↪'MntMeatProducts',
                                                'MntFishProducts',␣
 ↪'MntSweetProducts',
                                                'MntGoldProds']].sum(axis=1)

plt.figure(figsize=(6, 4))
sns.boxplot(x='Marital_Status', y='Total_Spending', data=cleaned_data)
plt.title('Total Spending by Marital Status')
plt.show()
```



[ ]:

Hypothesis testing:

Is income of customers dependent on their education

[38]:
```
from scipy import stats as st
```

[39]:
```
# Null Hypothesis (H0) : There is no relationship between income and education level.
# Alternative Hypothesis (Ha) : There is a relationship between income and education level.
# check for missing values in income column
df['Income'].isnull().sum()
df['Income'].dropna()
```

[39]:
```
0       84835.0
1       57091.0
2       67267.0
3       32474.0
4       21474.0
         ...
2234    66476.0
```

```
2235    31056.0
2236    46310.0
2237    65819.0
2238    94871.0
Name: Income, Length: 2215, dtype: float64
```

[40]:
```
# Group data by Education and create lists of incomes for each education level
income_by_education = [group['Income'].values for name, group in df.
 ↪groupby('Education')]
f_statistic, p_value = st.f_oneway(*income_by_education)
# Print results
print(f"F-statistic: {f_statistic}")
print(f"P-value: {p_value}")
```

```
F-statistic: nan
P-value: nan
```

[41]:
```
# Interpret results
alpha = 0.05  # significance level

if p_value < alpha:
    print("Reject the null hypothesis: There is a significant relationship␣
 ↪between income and education level.")
else:
    print("Fail to reject the null hypothesis: There is no significant␣
 ↪relationship between income and education level.")
```

```
Fail to reject the null hypothesis: There is no significant relationship between
income and education level.
```

[42]:
```
df.sample(10)
```

[42]:

	Year_Birth	Education	Marital_Status	Income	Kidhome	Teenhome	\
677	1963	Master	Divorced	49476.0	0	1	
1884	1971	PhD	Together	78642.0	0	1	
553	1959	Graduation	Together	71367.0	0	0	
67	1960	2n Cycle	Married	82504.0	0	0	
209	1945	PhD	Single	45576.0	0	0	
335	1983	Graduation	Together	78687.0	0	0	
396	1974	2n Cycle	Married	65463.0	1	0	
896	1978	2n Cycle	Married	26224.0	1	0	
230	1965	Graduation	Together	56046.0	0	0	
1816	1973	Graduation	Divorced	71128.0	1	0	

	Recency	MntWines	MntFruits	MntMeatProducts	…	AcceptedCmp4	\
677	29	386	23	95	…	0	
1884	83	1396	0	322	…	0	

```
553        24       227         23              389  …             0
67          2       362         50              431  …             0
209         9        56         19               29  …             0
335        13       817        185              687  …             0
396        17       391         32               70  …             1
896        39         4          7               15  …             0
230         9       577          0               64  …             0
1816       80       958        159              447  …             0

      AcceptedCmp5  AcceptedCmp1  AcceptedCmp2  Country  Age  Income_Category  \
677              0             0             0       CA   61          Medium
1884             0             0             0       SP   53            High
553              0             0             0       SP   65            High
67               0             0             0      IND   64            High
209              0             0             0       SP   79          Medium
335              1             0             0       SP   41            High
396              0             0             0       SP   50            High
896              0             0             0       SP   46             Low
230              0             0             0      GER   59          Medium
1816             0             0             0       US   51            High

      Family_Size Total_Spending  Campaign_Acceptance_Count
677             1            795                          0
1884            1           1816                          0
553             0            777                          0
67              0           1066                          0
209             0            145                          0
335             0           2130                          1
396             1            562                          1
896             1             63                          0
230             0            692                          1
1816            1           1615                          0

[10 rows x 24 columns]
```

1.1.1 Do higher income people spend more (take in account spending in all categories together)

Null Hypothesis : There is no relationship between income and total spending across all categories

Alternate Hypothesis: Higher income is associated with higher total spending across all categories.

[43]:
```
# Calculate average total spending by income category
average_spending = df.groupby('Income_Category')['Total_Spending'].mean()
average_spending
```

```
C:\Users\Teju\AppData\Local\Temp\ipykernel_11484\1405525553.py:2: FutureWarning:
The default of observed=False is deprecated and will be changed to True in a
future version of pandas. Pass observed=False to retain current behavior or
observed=True to adopt the future default and silence this warning.
  average_spending = df.groupby('Income_Category')['Total_Spending'].mean()
```

[43]:
```
Income_Category
Low            72.183784
Medium        299.216915
High         1185.615482
Very High    1606.365385
Name: Total_Spending, dtype: float64
```

[44]:
```
# Perform ANOVA to see if there's a significant difference in spending across␣
 ↪income categories
anova_results = st.f_oneway(
    df[df['Income_Category'] == 'Low']['Total_Spending'],
    df[df['Income_Category'] == 'Medium']['Total_Spending'],
    df[df['Income_Category'] == 'High']['Total_Spending'],
    df[df['Income_Category'] == 'Very High']['Total_Spending']
)
# Print ANOVA results
print(f"F-statistic: {anova_results.statistic:.2f}")
print(f"P-value: {anova_results.pvalue:.4f}")

# Interpret results
alpha = 0.05  # significance level

if anova_results.pvalue < alpha:
    print("Reject the null hypothesis: There is a significant relationship␣
 ↪between income and total spending.")
else:
    print("Fail to reject the null hypothesis: There is no significant␣
 ↪relationship between income and total spending.")
```

```
F-statistic: 1322.27
P-value: 0.0000
Reject the null hypothesis: There is a significant relationship between income
and total spending.
```

[ ]:

1.1.2 Do couples spend more or less money on wine than people living alone (set 'Married','Together':'In couple' and 'Divorced','Single','Absurd','Widow','YOLO':'Alone')

Null Hypothesis: There is no difference in wine spending between couples and individuals living alone.

Alternate Hypothesis: Couples spend more or less on wine than individuals living alone.

[45]:
```
# Categorize Marital Status
def categorize_marital_status(status):
    if status in ['Married', 'Together']:
        return 'In couple'
    else:
        return 'Alone'

df['Marital_Category'] = df['Marital_Status'].apply(categorize_marital_status)

# Separate spending based on marital category
couples_spending = df[df['Marital_Category'] == 'In couple']['MntWines']
alone_spending = df[df['Marital_Category'] == 'Alone']['MntWines']
```

[46]:
```
# Perform t-test
t_statistic, p_value = st.ttest_ind(couples_spending, alone_spending,␣
 ↪equal_var=False)

# Print results
print(f"T-statistic: {t_statistic:.2f}")
print(f"P-value: {p_value:.4f}")

# Interpret results
alpha = 0.05  # significance level

if p_value < alpha:
    print("Reject the null hypothesis: There is a significant difference in␣
 ↪wine spending between couples and individuals living alone.")
else:
    print("Fail to reject the null hypothesis: There is no significant␣
 ↪difference in wine spending between couples and individuals living alone.")
```

```
T-statistic: -0.27
P-value: 0.7863
Fail to reject the null hypothesis: There is no significant difference in wine
spending between couples and individuals living alone.
```

1.1.3 Are people with lower income are more attracted towards campaign or simply put accept more campaigns. (create two income brackets one below median , other above median income and create a column which tells if they have ever accepted any campaign)

Null Hypothesis: There is no difference in campaign acceptance between low-income and high-income individuals.

Alternate Hypothesis: Lower-income individuals accept more campaigns than higher-income individuals.

[47]:
```
# Calculate median income
median_income = df['Income'].median()
median_income
# Create income brackets
df['Median_Income_Category'] = df['Income'].apply(lambda x: 'Below Median' if x␣
 ↪< median_income else 'Above Median')
# Create a column indicating if any campaign was accepted
df['Accepted_Any_Campaign'] = df[['AcceptedCmp1', 'AcceptedCmp2',␣
 ↪'AcceptedCmp3',
                                  'AcceptedCmp4', 'AcceptedCmp5']].
 ↪sum(axis=1) > 0
# Calculate the average number of accepted campaigns for each income category
df['Accepted_Campaigns'] = df[['AcceptedCmp1', 'AcceptedCmp2', 'AcceptedCmp3',
                                'AcceptedCmp4', 'AcceptedCmp5']].sum(axis=1)
# Separate accepted campaigns based on income category
below_median_accepted = df[df['Income_Category'] == 'Below␣
 ↪Median']['Accepted_Campaigns']
above_median_accepted = df[df['Income_Category'] == 'Above␣
 ↪Median']['Accepted_Campaigns']
```

[47]:
```
np.float64(51373.0)
```

[52]:
```
# Perform t-test
t_statistic, p_value = st.ttest_ind(below_median_accepted,␣
 ↪above_median_accepted, equal_var=False)

# Print results
print(f"T-statistic: {t_statistic}")
print(f"P-value: {p_value}")

# Interpret results
alpha = 0.05  # significance level

if p_value < alpha:
    print("Reject the null hypothesis: Lower-income individuals accept more␣
 ↪campaigns than higher-income individuals.")
else:
    print("Fail to reject the null hypothesis: There is no significant␣
 ↪difference in campaign acceptance between lower and higher-income␣
 ↪individuals.")
```

```
T-statistic: nan
P-value: nan
Fail to reject the null hypothesis: There is no significant difference in
campaign acceptance between lower and higher-income individuals.
```

[ ]:
```

```

1.1.4 Insights

1. Income Impact on Campaign Acceptance: Lower-income individuals tend to accept more campaigns compared to higher-income individuals, indicating a potential sensitivity to price promotions.

2. Spending Behavior by Education Level: Customers with higher education levels often exhibit different spending patterns across product categories, suggesting targeted marketing could yield better results.

3. Marital Status Influence: Couples tend to spend differently than singles; campaigns could be designed to appeal specifically to family-oriented customers or single individuals based on their spending habits.

4. Geographic Variations: Different countries show varied responses to campaigns; localized marketing strategies may improve acceptance rates in specific regions.

5. Recency of Purchase Matters: Customers with recent purchases are more likely to accept new campaigns, highlighting the importance of timely marketing efforts following a purchase.

[ ]:

1.1.5 Recommedation

1. Targeted Campaigns for Low-Income Groups: Develop specific marketing campaigns aimed at lower-income individuals, as they may be more responsive to promotions and discounts.

2. Personalized Offers Based on Spending Patterns: Utilize customer spending data to create personalized offers that cater to individual preferences, enhancing engagement and acceptance rates.

3. Increase Awareness of Campaigns: Implement strategies to increase awareness of campaigns among customers, particularly those who have not accepted previous offers.

4. Leverage Social Proof: Use testimonials or case studies from satisfied customers in similar income brackets to encourage others to participate in campaigns.

5. Optimize Communication Channels: Analyze which communication channels (email, SMS, social media) yield the highest campaign acceptance rates and focus efforts there.

[ ]: