In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

Import the Excel Package

In [2]:
```
pip install xlrd
```

```
Requirement already satisfied: xlrd in c:\users\captr\anaconda3\lib\site-packages
(2.0.1)
Note: you may need to restart the kernel to use updated packages.
```

Load The Data Set

In [3]:
```
data = pd.read_excel("C:\\Users\\captr\\OneDrive\\Desktop\\marketing data.xls")
data.head()
```

Out[3]:

	ID	Year_Birth	Education	Marital_Status	Income	Kidhome	Teenhome	Dt_Customer	R
0	1826	1970	Graduation	Divorced	$84,835.00	0	0	6/16/14	
1	1	1961	Graduation	Single	$57,091.00	0	0	6/15/14	
2	10476	1958	Graduation	Married	$67,267.00	0	1	5/13/14	
3	1386	1967	Graduation	Together	$32,474.00	1	1	2014-11-05 00:00:00	
4	5371	1989	Graduation	Single	$21,474.00	1	0	2014-08-04 00:00:00	

5 rows × 28 columns

Data Inspection

In [4]:
```
data.Dt_Customer.head()
```

Out[4]:
```
0               6/16/14
1               6/15/14
2               5/13/14
3    2014-11-05 00:00:00
4    2014-08-04 00:00:00
Name: Dt_Customer, dtype: object
```

In [5]:
```
print(data.columns)
```

```
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', 'Response', 'Complain', 'Country'],
      dtype='object')
```

In [6]:
```
data[' Income '].head()
```

Out[6]:
```
0    $84,835.00
1    $57,091.00
2    $67,267.00
3    $32,474.00
4    $21,474.00
Name:  Income , dtype: object
```

In [7]:
```
data.isnull().sum()
```

Out[7]:
```
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
Response                0
Complain                0
Country                 0
dtype: int64
```

As we can see that the income column has 24 null values so we will fix it

In [8]:
```
data.columns = data.columns.str.strip()
data['Income'] = data['Income'].replace('[\$,]', '', regex=True).astype(float)
income_means = data.groupby(['Education', 'Marital_Status'])['Income'].transform('m
data['Income'] = data['Income'].fillna(income_means)
```

In [9]:
```
data.head(30)
```

Out[9]:

	ID	Year_Birth	Education	Marital_Status	Income	Kidhome	Teenhome	Dt_Customer	Re
0	1826	1970	Graduation	Divorced	84835.0	0	0	6/16/14	
1	1	1961	Graduation	Single	57091.0	0	0	6/15/14	
2	10476	1958	Graduation	Married	67267.0	0	1	5/13/14	
3	1386	1967	Graduation	Together	32474.0	1	1	2014-11-05 00:00:00	
4	5371	1989	Graduation	Single	21474.0	1	0	2014-08-04 00:00:00	
5	7348	1958	PhD	Single	71691.0	0	0	3/17/14	
6	4073	1954	2n Cycle	Married	63564.0	0	0	1/29/14	
7	1991	1967	Graduation	Together	44931.0	0	1	1/18/14	
8	4047	1954	PhD	Married	65324.0	0	1	2014-11-01 00:00:00	
9	9477	1954	PhD	Married	65324.0	0	1	2014-11-01 00:00:00	
10	2079	1947	2n Cycle	Married	81044.0	0	0	12/27/13	
11	5642	1979	Master	Together	62499.0	1	0	2013-09-12 00:00:00	
12	10530	1959	PhD	Widow	67786.0	0	0	2013-07-12 00:00:00	
13	2964	1981	Graduation	Married	26872.0	0	0	10/16/13	
14	10311	1969	Graduation	Married	4428.0	0	1	2013-05-10 00:00:00	
15	837	1977	Graduation	Married	54809.0	1	1	2013-11-09 00:00:00	
16	10521	1977	Graduation	Married	54809.0	1	1	2013-11-09 00:00:00	
17	10175	1958	PhD	Divorced	32173.0	0	1	2013-01-08 00:00:00	
18	1473	1960	2n Cycle	Single	47823.0	0	1	7/23/13	
19	2795	1958	Master	Single	30523.0	2	1	2013-01-07 00:00:00	
20	2285	1954	Master	Together	36634.0	0	1	5/28/13	
21	115	1966	Master	Single	43456.0	0	1	3/26/13	
22	10470	1979	Master	Married	40662.0	1	0	3/15/13	
23	4065	1976	PhD	Married	49544.0	1	0	2013-12-02 00:00:00	
24	10968	1969	Graduation	Single	57731.0	0	1	11/23/12	
25	5985	1965	Master	Single	33168.0	0	1	10/13/12	
26	5430	1956	Graduation	Together	54450.0	1	1	9/14/12	
27	8432	1956	Graduation	Together	54450.0	1	1	9/14/12	
28	453	1956	PhD	Widow	35340.0	1	1	6/29/14	

	ID	Year_Birth	Education	Marital_Status	Income	Kidhome	Teenhome	Dt_Customer	Re
29	9687	1975	Graduation	Single	73170.0	0	0	5/31/14	

In [10]:
```
data.isnull().sum()
```

Out[10]:
```
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
Response               0
Complain               0
Country                0
dtype: int64
```

Woah !! Problem solved as you can see from above

In [11]:
```
type(data.Kidhome)
```

Out[11]:
```
pandas.core.series.Series
```

Creating variables to represent the total number of children

In [12]:
```
data["total_Children"] = data.Kidhome+data.Teenhome
data.head()
```

Out[12]:

	ID	Year_Birth	Education	Marital_Status	Income	Kidhome	Teenhome	Dt_Customer	Rece
0	1826	1970	Graduation	Divorced	84835.0	0	0	6/16/14	
1	1	1961	Graduation	Single	57091.0	0	0	6/15/14	
2	10476	1958	Graduation	Married	67267.0	0	1	5/13/14	
3	1386	1967	Graduation	Together	32474.0	1	1	2014-11-05 00:00:00	
4	5371	1989	Graduation	Single	21474.0	1	0	2014-08-04 00:00:00	

5 rows × 29 columns

In [13]:
```
from datetime import datetime
```

Creating variables to represent the age of the customer

In [14]:
```
current_age = datetime.now().year
data['Age'] = current_age - data['Year_Birth']
data.head()
```

Out[14]:

	ID	Year_Birth	Education	Marital_Status	Income	Kidhome	Teenhome	Dt_Customer	Rece
0	1826	1970	Graduation	Divorced	84835.0	0	0	6/16/14	
1	1	1961	Graduation	Single	57091.0	0	0	6/15/14	
2	10476	1958	Graduation	Married	67267.0	0	1	5/13/14	
3	1386	1967	Graduation	Together	32474.0	1	1	2014-11-05 00:00:00	
4	5371	1989	Graduation	Single	21474.0	1	0	2014-08-04 00:00:00	

5 rows × 30 columns

Creating variables to represent the total spending of the customer

In [15]:
```
spending_cols=["MntWines","MntFruits","MntMeatProducts","MntFishProducts","MntSweet
```

In [16]:
```
data["Total_Spending"]=data[spending_cols].sum(axis=1)
data.head(5)
```

Out[16]:

	ID	Year_Birth	Education	Marital_Status	Income	Kidhome	Teenhome	Dt_Customer	Rece
0	1826	1970	Graduation	Divorced	84835.0	0	0	6/16/14	
1	1	1961	Graduation	Single	57091.0	0	0	6/15/14	
2	10476	1958	Graduation	Married	67267.0	0	1	5/13/14	
3	1386	1967	Graduation	Together	32474.0	1	1	2014-11-05 00:00:00	
4	5371	1989	Graduation	Single	21474.0	1	0	2014-08-04 00:00:00	

5 rows × 31 columns

In [17]:
```
data["Total_Spending"]=data.MntWines + data.MntFruits + data.MntMeatProducts + data
```

In [18]:
```
data.head()
```

Out[18]:

	ID	Year_Birth	Education	Marital_Status	Income	Kidhome	Teenhome	Dt_Customer	Rece
0	1826	1970	Graduation	Divorced	84835.0	0	0	6/16/14	
1	1	1961	Graduation	Single	57091.0	0	0	6/15/14	
2	10476	1958	Graduation	Married	67267.0	0	1	5/13/14	
3	1386	1967	Graduation	Together	32474.0	1	1	2014-11-05 00:00:00	
4	5371	1989	Graduation	Single	21474.0	1	0	2014-08-04 00:00:00	

5 rows × 31 columns

Creating variables to represent the total Purchasing of the customer

In [19]:
```
data["Total_Purchases"]=data.NumDealsPurchases + data.NumWebPurchases + data.NumCat
data.head()
```

Out[19]:

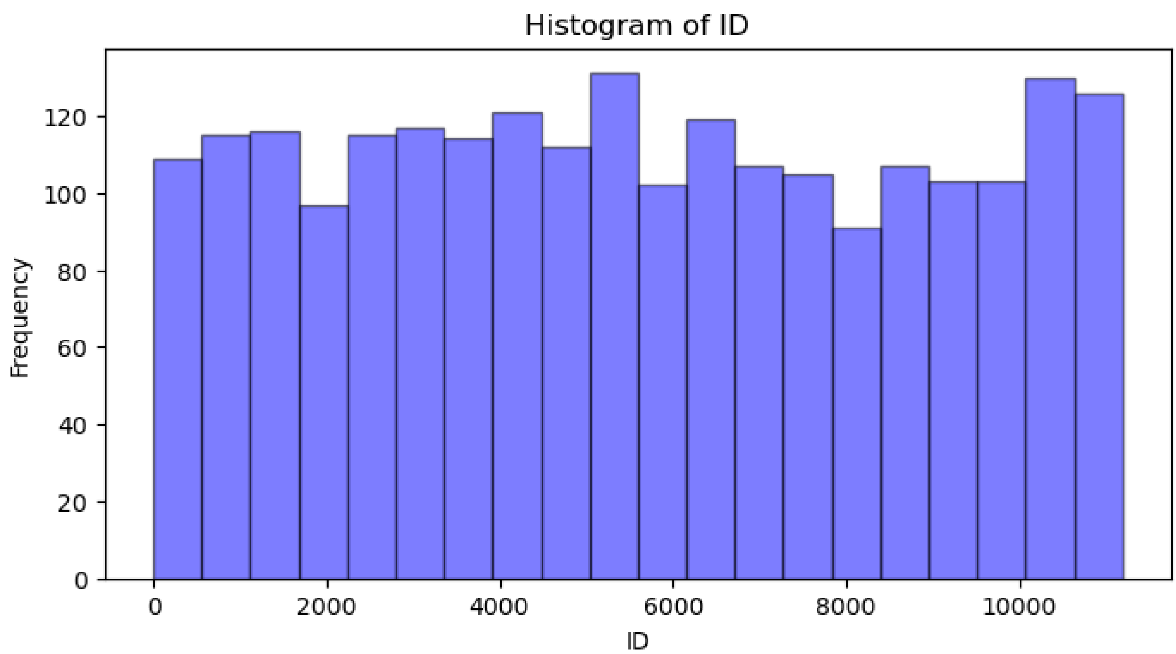
	ID	Year_Birth	Education	Marital_Status	Income	Kidhome	Teenhome	Dt_Customer	Rece
0	1826	1970	Graduation	Divorced	84835.0	0	0	6/16/14	
1	1	1961	Graduation	Single	57091.0	0	0	6/15/14	
2	10476	1958	Graduation	Married	67267.0	0	1	5/13/14	
3	1386	1967	Graduation	Together	32474.0	1	1	2014-11-05 00:00:00	
4	5371	1989	Graduation	Single	21474.0	1	0	2014-08-04 00:00:00	

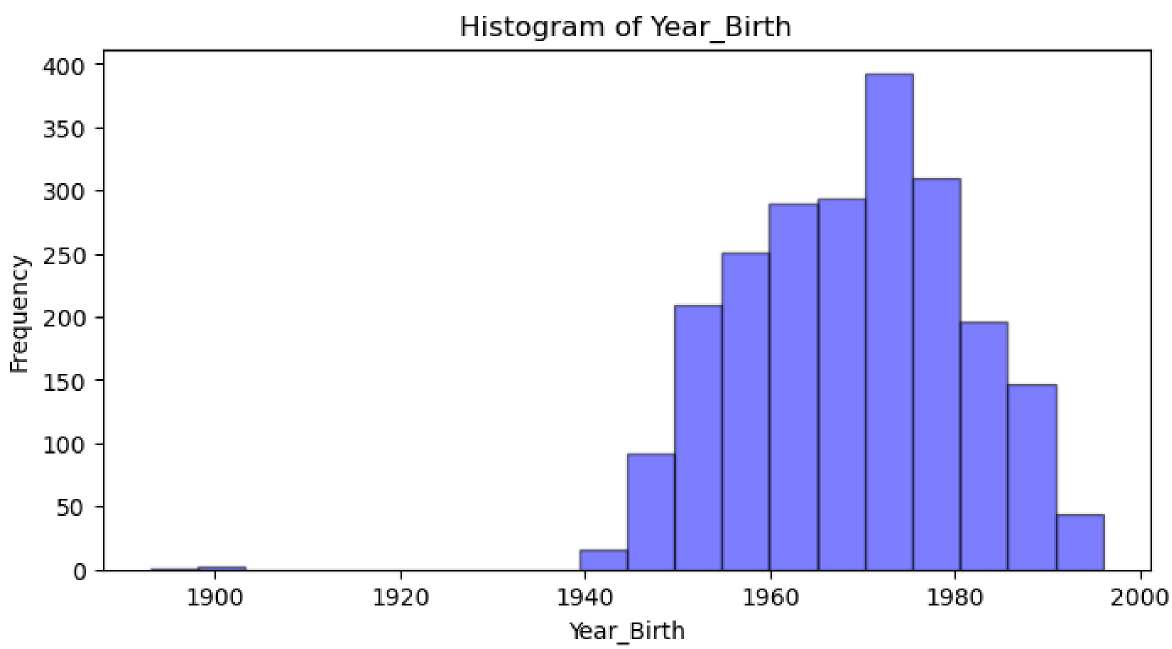
5 rows × 32 columns

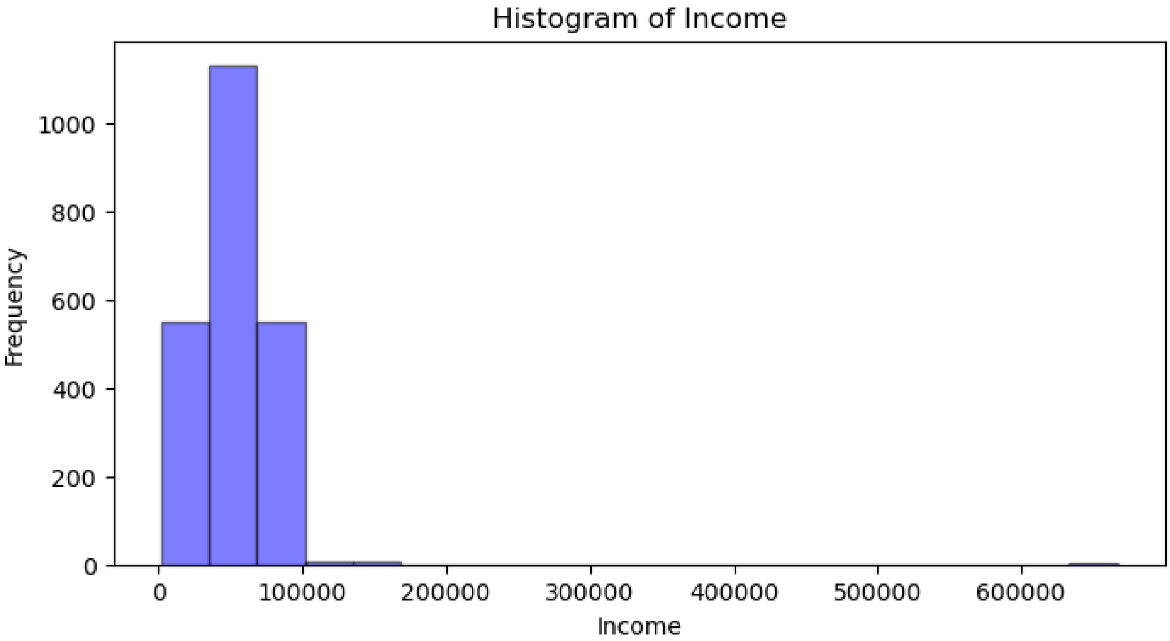
Generated box plots and histograms to gain insights into the distributions and identify outliers.

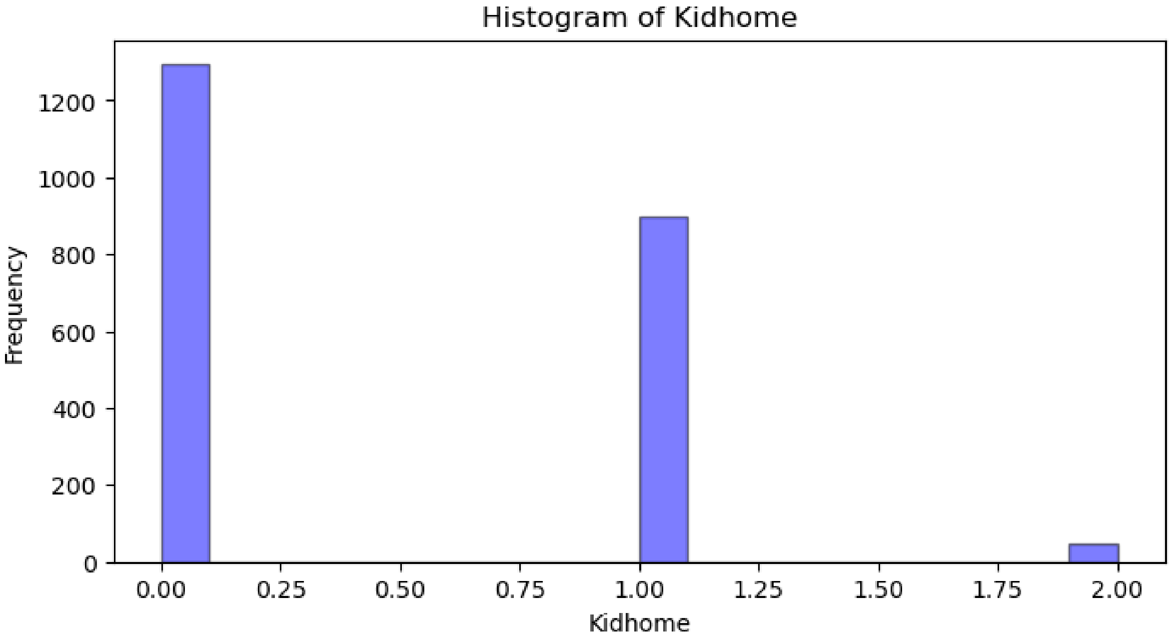
In [20]:
```
%matplotlib inline
```

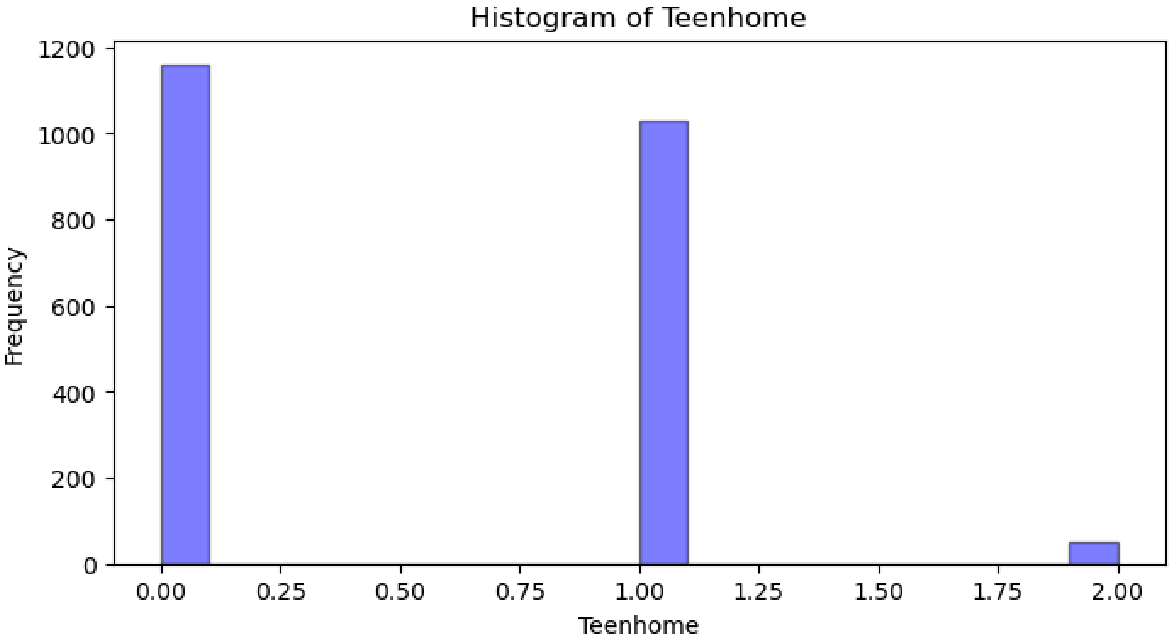
In [21]:
```
import matplotlib.pyplot as plt

# Selecting only the numeric columns
numeric_columns = data.select_dtypes(include=['float64', 'int64']).columns


for col in numeric_columns:
    plt.figure(figsize=(8, 4))
    plt.hist(data[col], bins=20, alpha=0.5, color='blue', edgecolor='black')
    plt.title(f'Histogram of {col}')
    plt.xlabel(col)
    plt.ylabel('Frequency')
    plt.show()
```

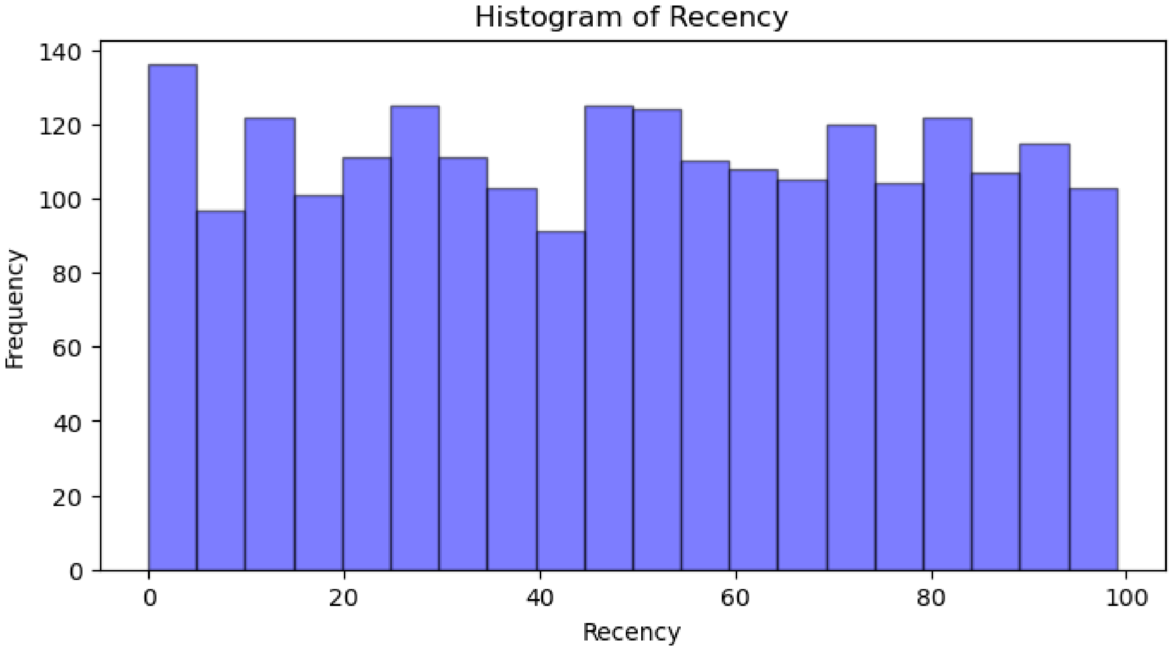


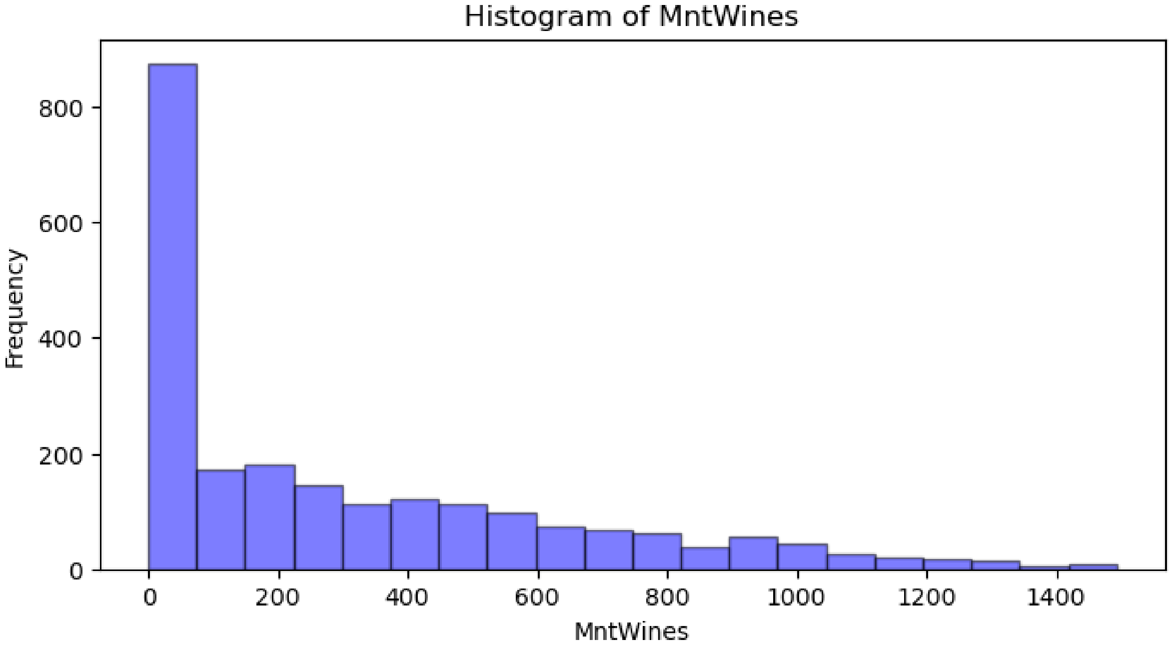


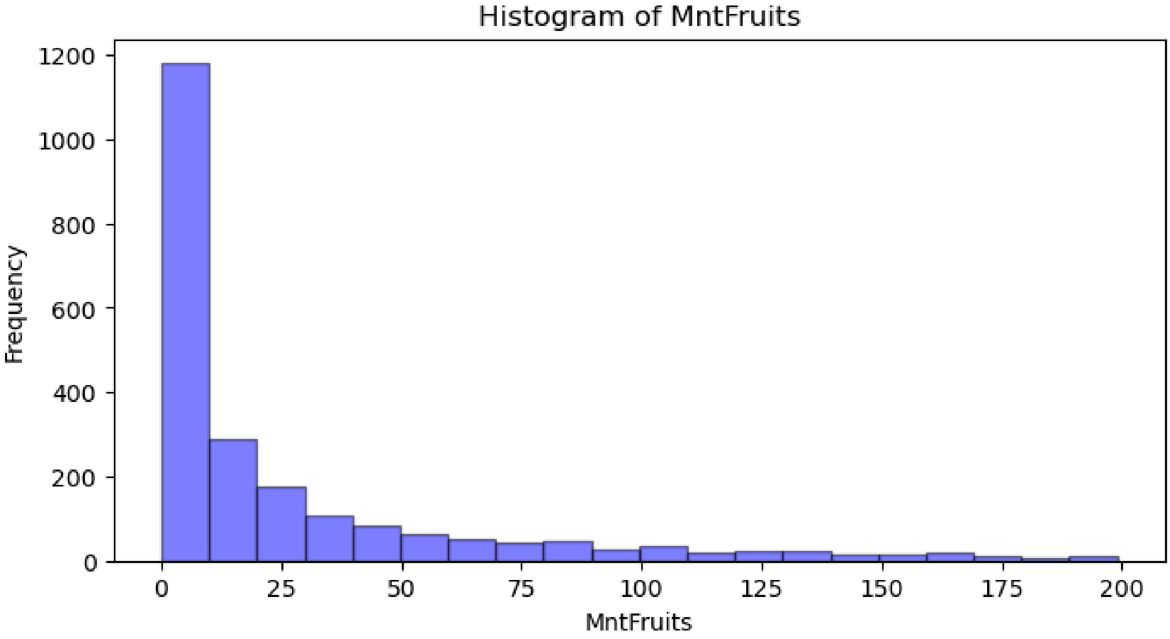

Histogram of Income
Frequency
0
200
400
600
800
1000
0
100000
200000
300000
400000
500000
600000
Income

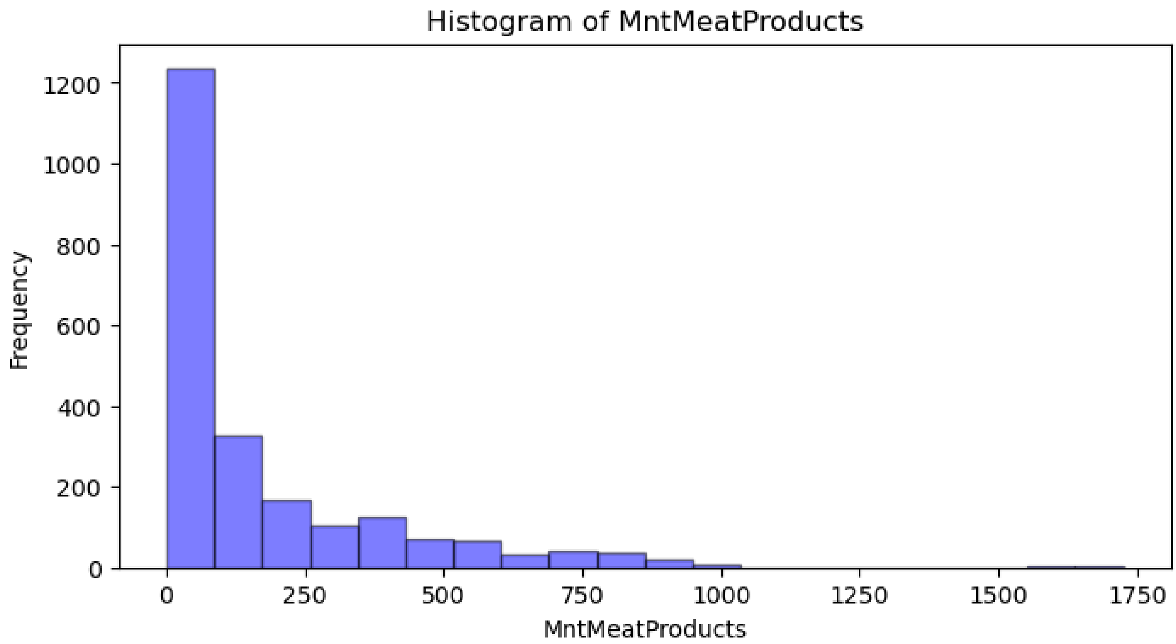

Histogram of Kidhome
Frequency
0
200
400
600
800
1000
1200
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
Kidhome

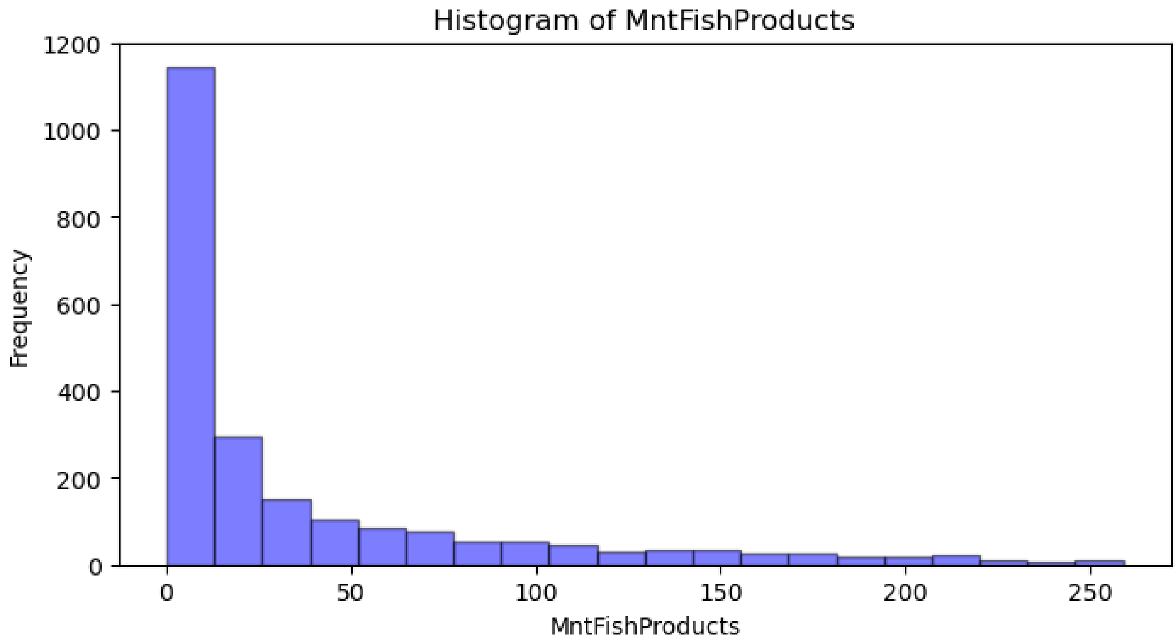

Histogram of Teenhome
Frequency
0
200
400
600
800
1000
1200
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
Teenhome

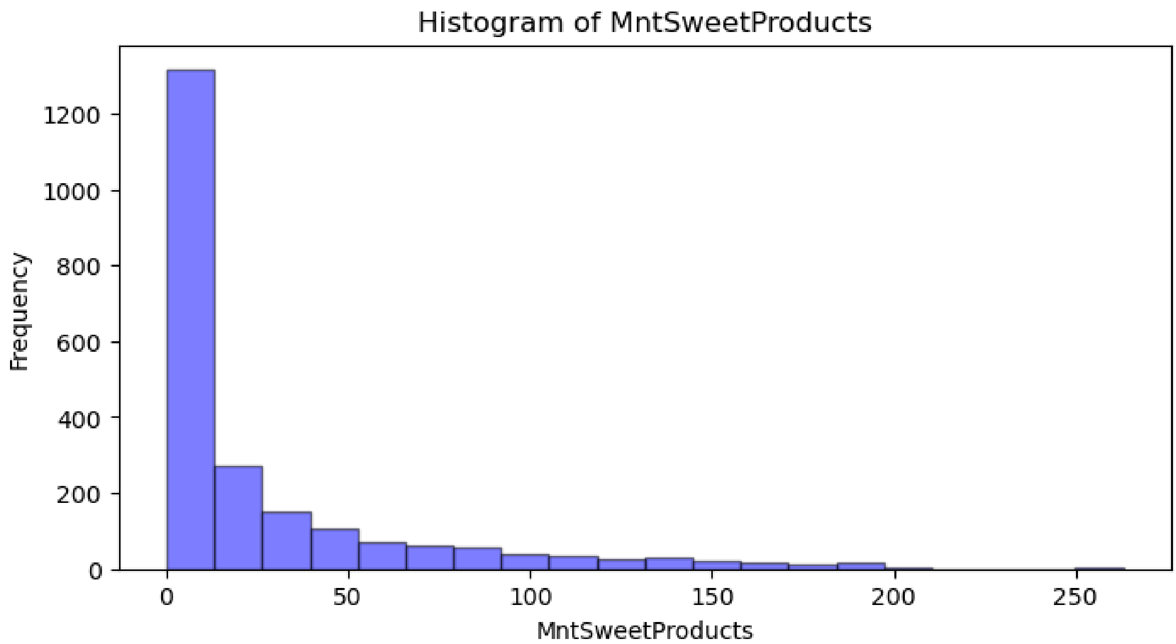

Histogram of Recency
Frequency
Recency
0
20
40
60
80
100
120
140

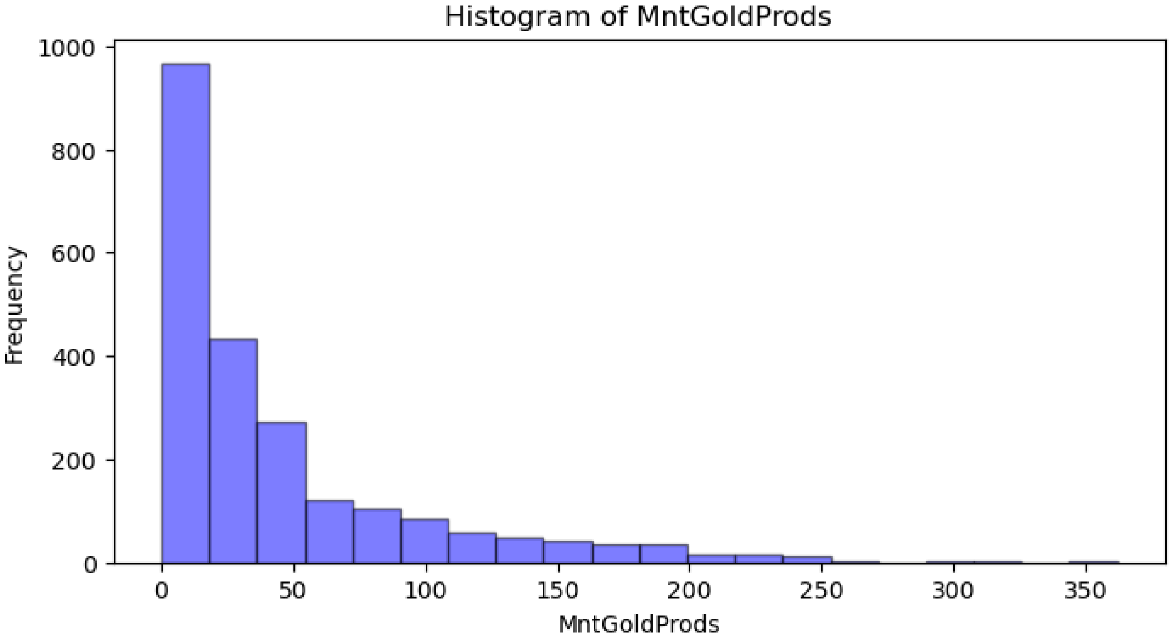

Histogram of MntWines
Frequency
MntWines
0
200
400
600
800
1000
1200
1400

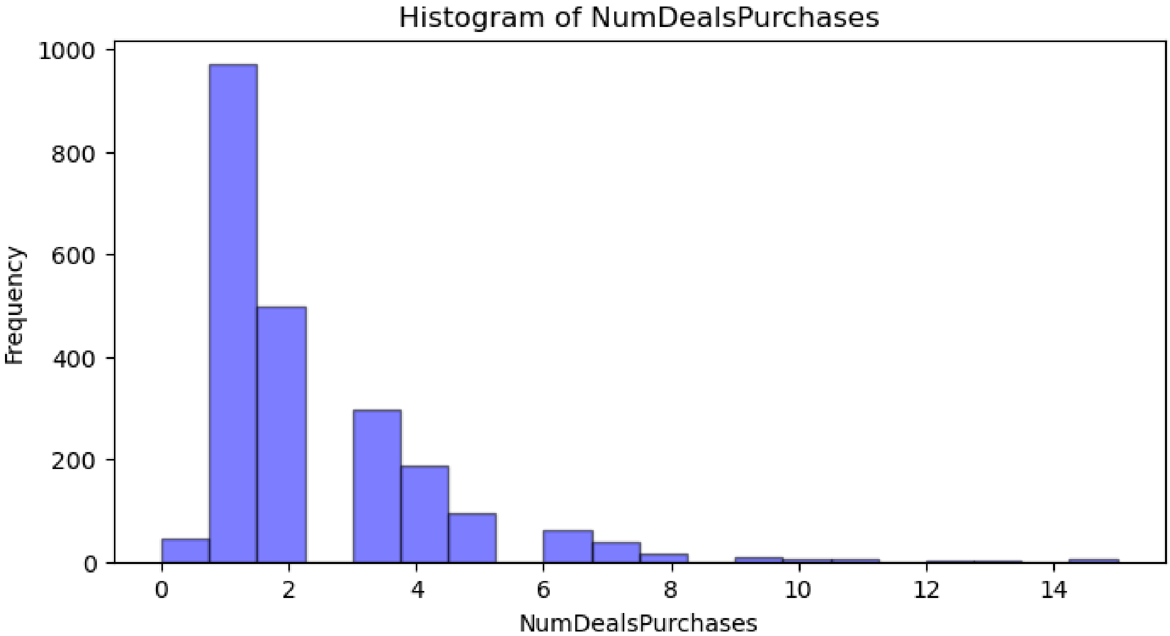

Histogram of MntFruits
Frequency
MntFruits
0
25
50
75
100
125
150
175
200
400
600
800
1000
1200

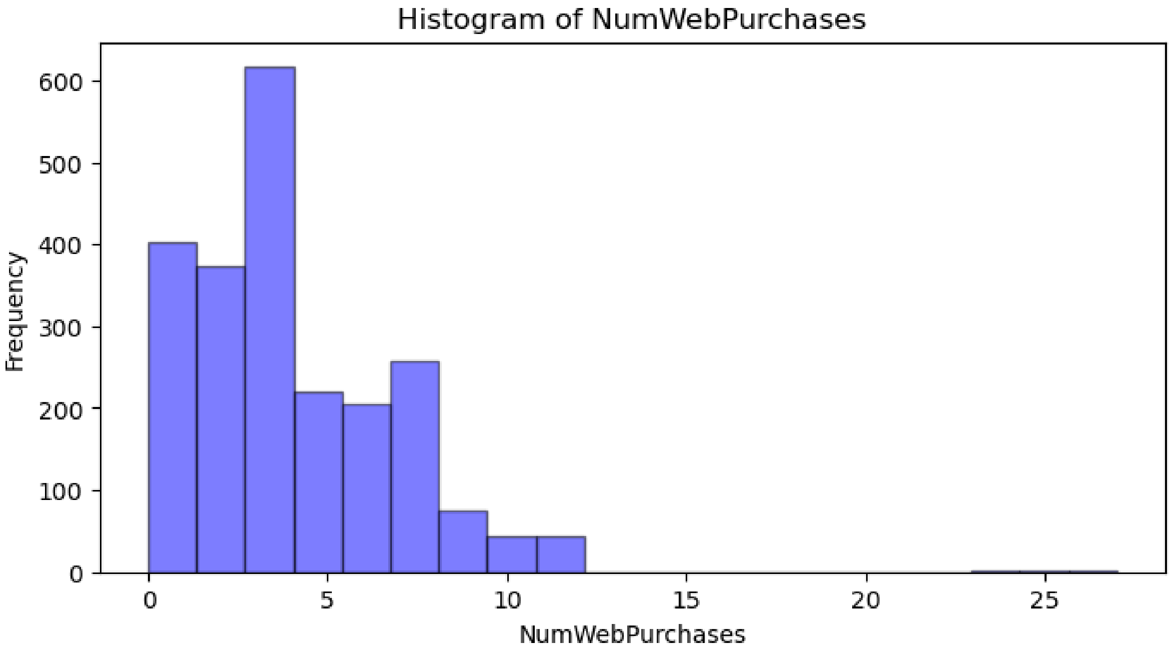

Histogram of MntMeatProducts
Frequency
0
200
400
600
800
1000
1200
0
250
500
750
1000
1250
1500
1750
MntMeatProducts

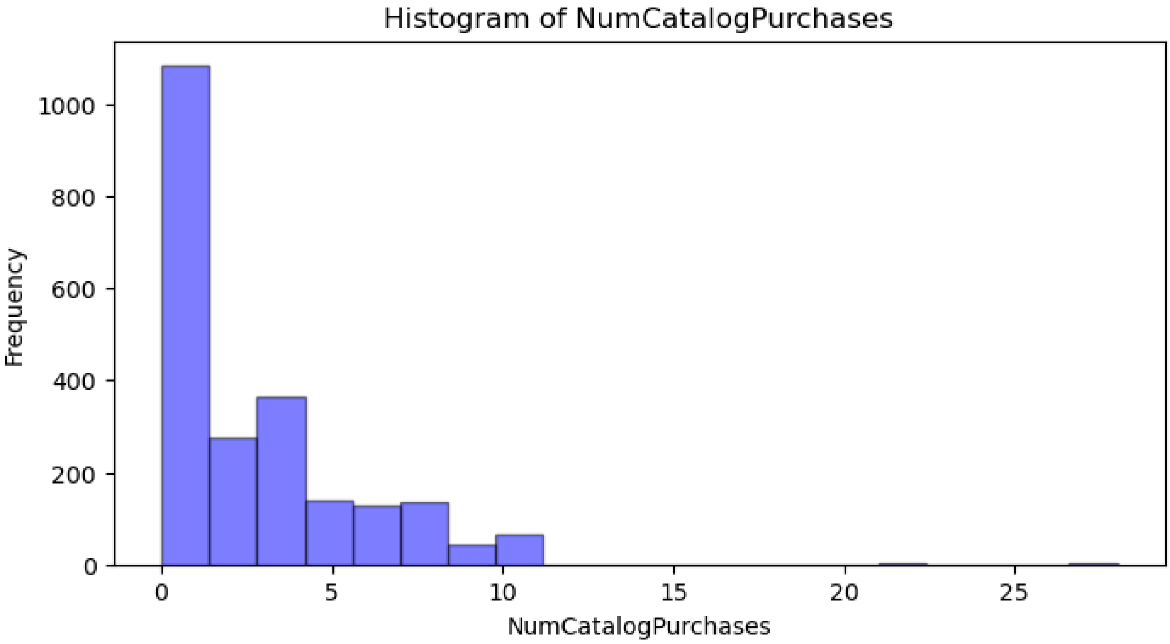

Histogram of MntFishProducts
Frequency
0
200
400
600
800
1000
1200
0
50
100
150
200
250
MntFishProducts


Histogram of MntSweetProducts
Frequency
0
200
400
600
800
1000
1200
0
50
100
150
200
250
MntSweetProducts

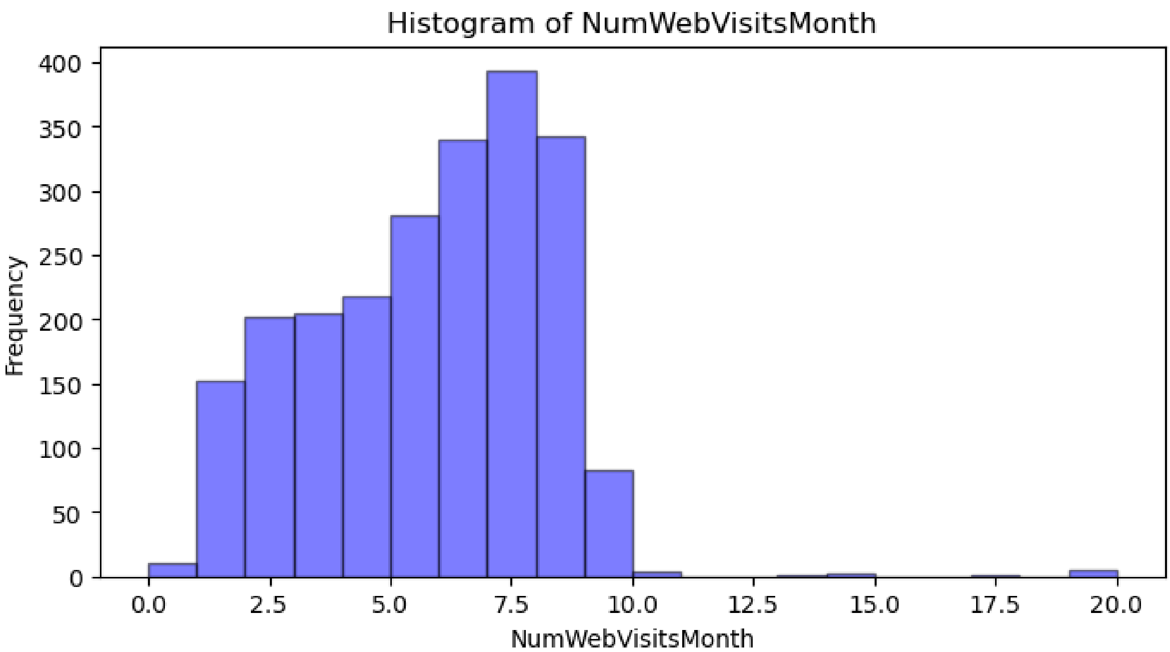

Histogram of MntGoldProds
Frequency
MntGoldProds
0
200
400
600
800
1000
0
50
100
150
200
250
300
350

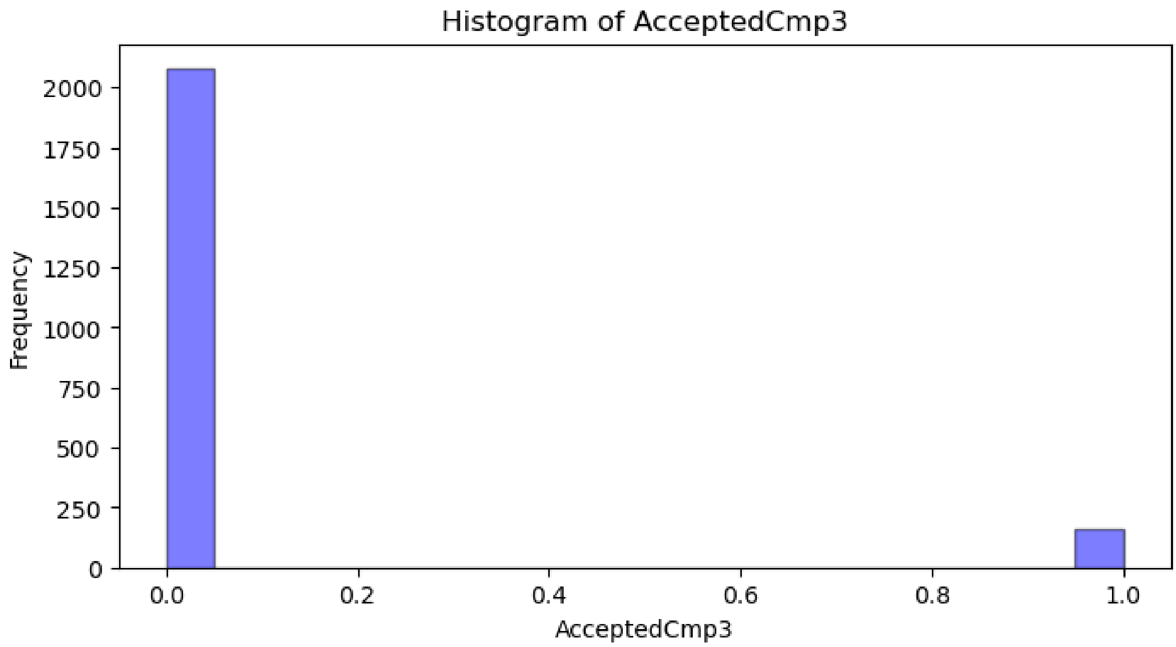

Histogram of NumDealsPurchases
Frequency
NumDealsPurchases
0
200
400
600
800
1000
0
2
4
6
8
10
12
14

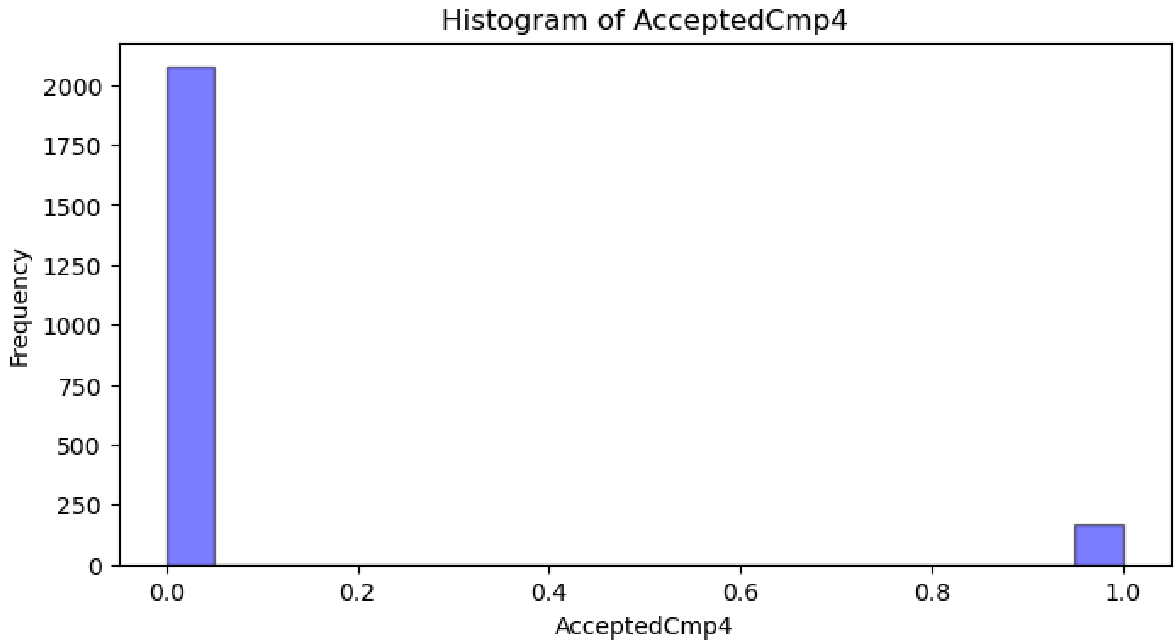

Histogram of NumWebPurchases
Frequency
NumWebPurchases
0
100
200
300
400
500
600
0
5
10
15
20
25

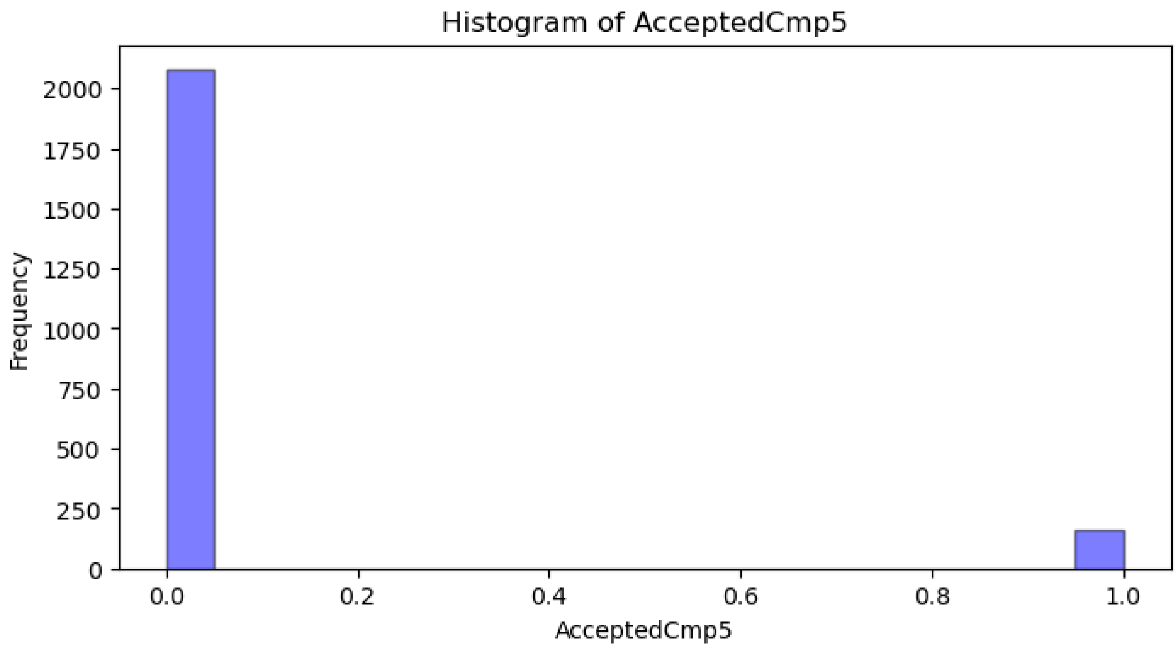

Histogram of NumCatalogPurchases
Frequency
NumCatalogPurchases
0
200
400
600
800
1000
0
5
10
15
20
25

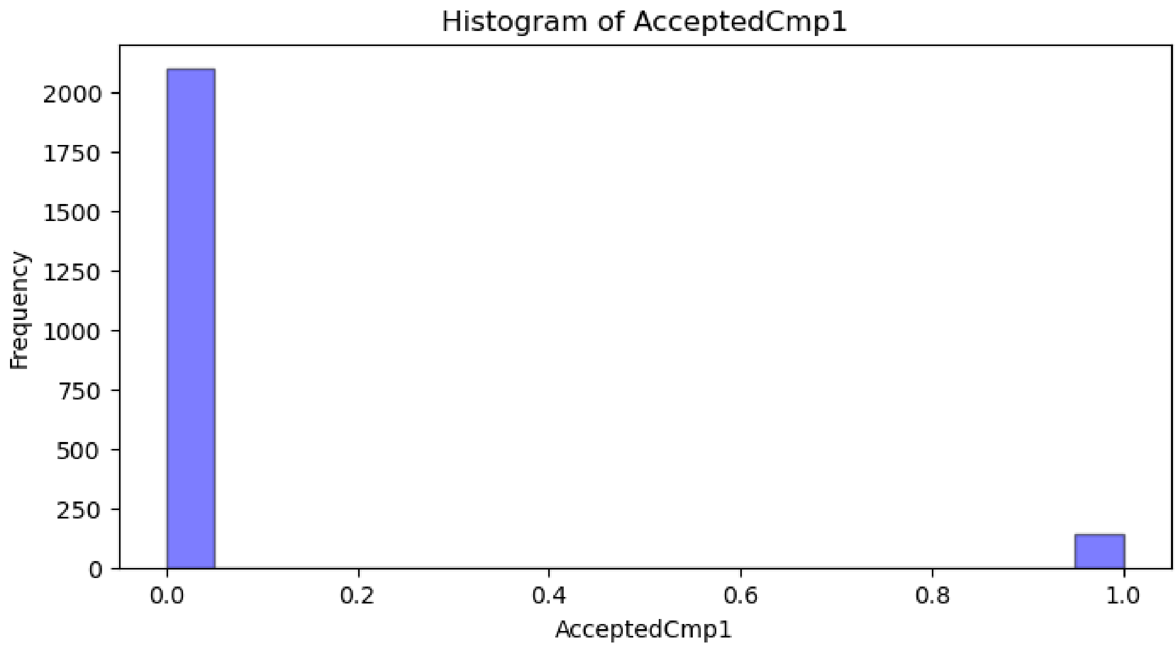

Histogram of NumStorePurchases
Frequency
NumStorePurchases
0
100
200
300
400
500
0
2
4
6
8
10
12

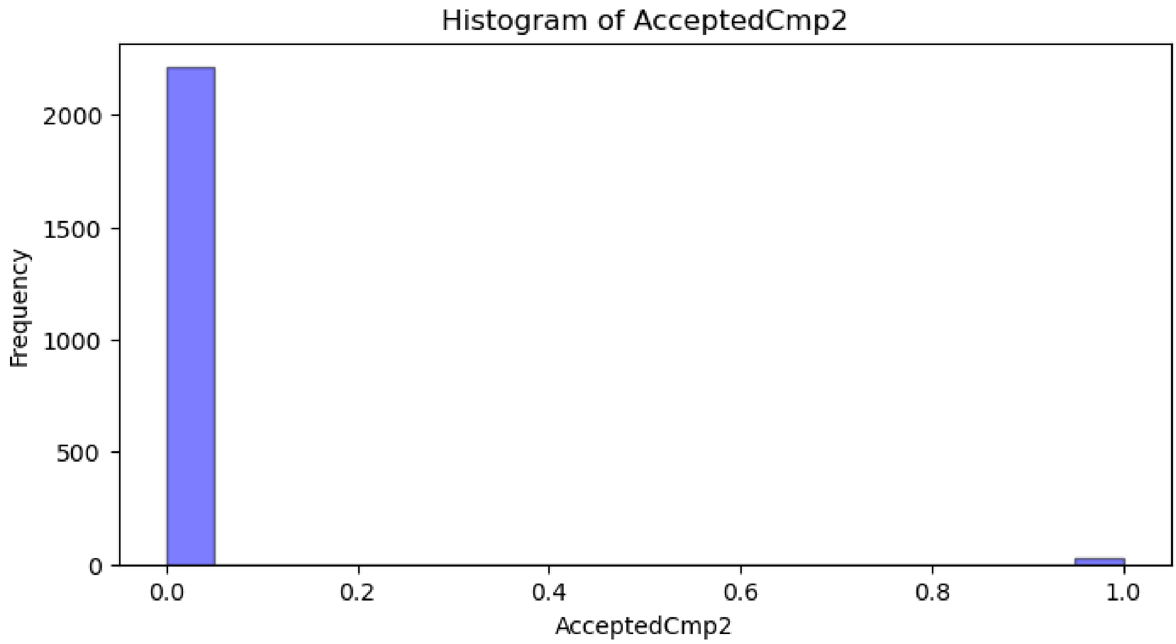

Histogram of NumWebVisitsMonth
Frequency
NumWebVisitsMonth
0
50
100
150
200
250
300
350
400
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0

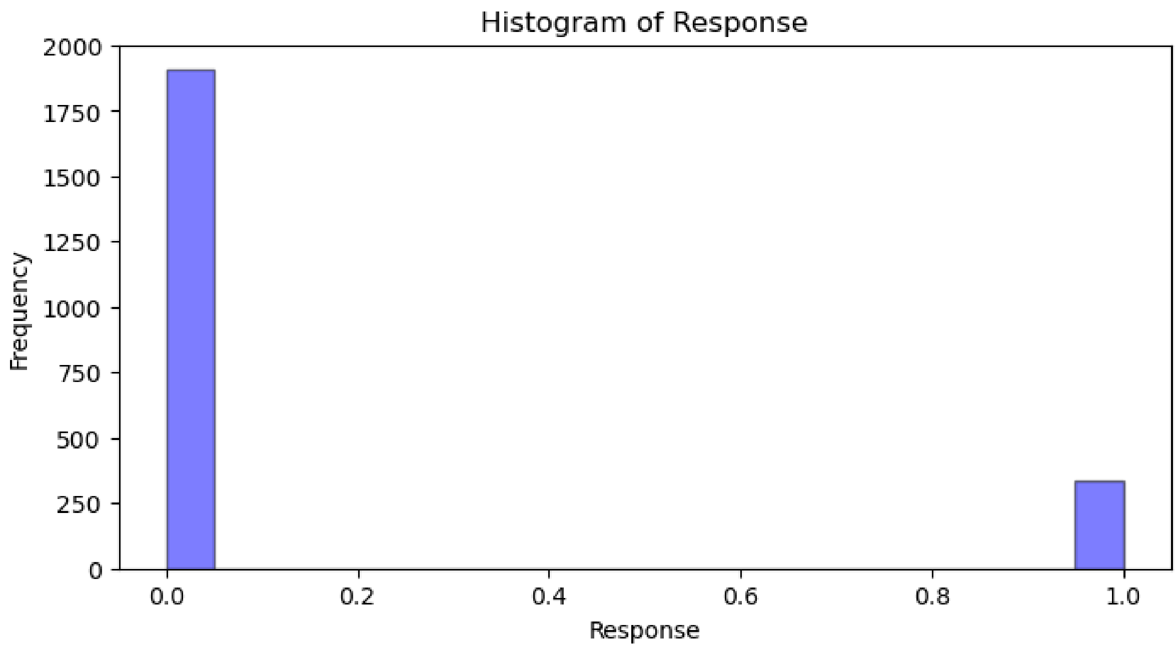

Histogram of AcceptedCmp3
Frequency
2000
1750
1500
1250
1000
750
500
250
0
0.0
0.2
0.4
0.6
0.8
1.0
AcceptedCmp3

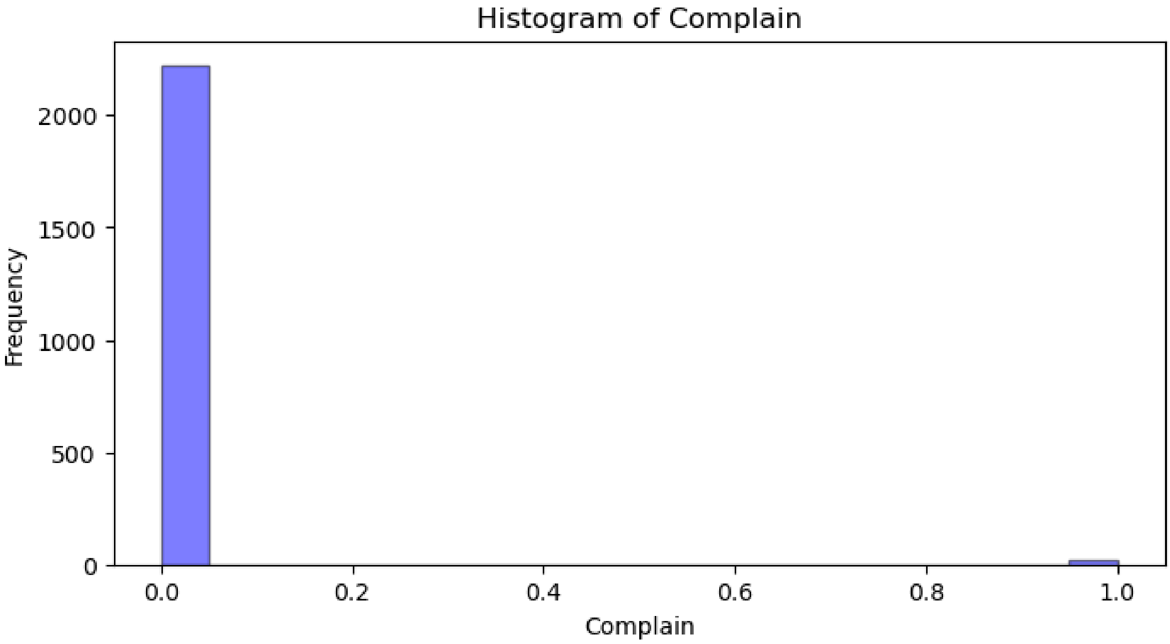

Histogram of AcceptedCmp4
Frequency
2000
1750
1500
1250
1000
750
500
250
0
0.0
0.2
0.4
0.6
0.8
1.0
AcceptedCmp4

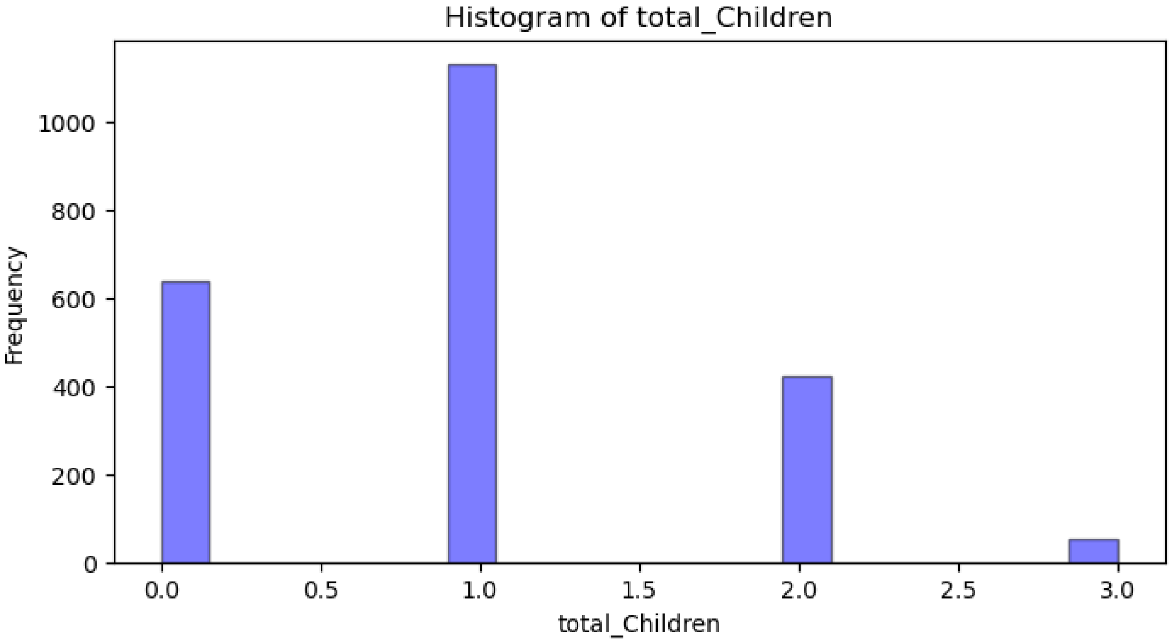

Histogram of AcceptedCmp5
Frequency
2000
1750
1500
1250
1000
750
500
250
0
0.0
0.2
0.4
0.6
0.8
1.0
AcceptedCmp5

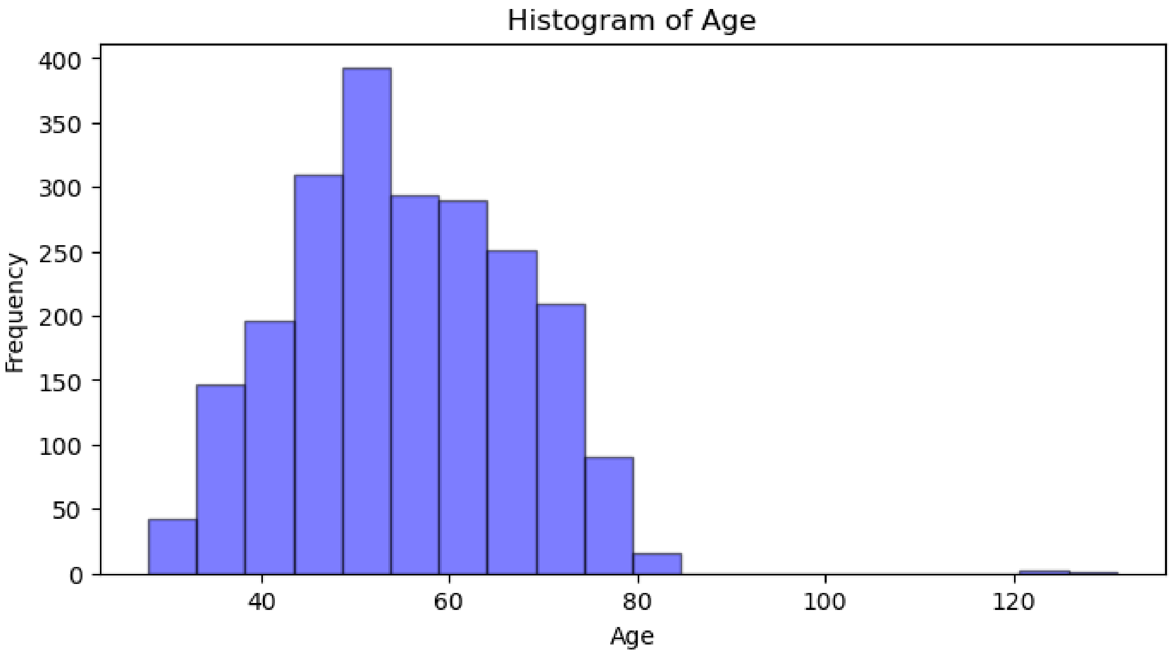

Histogram of AcceptedCmp1
Frequency
2000
1750
1500
1250
1000
750
500
250
0
0.0
0.2
0.4
0.6
0.8
1.0
AcceptedCmp1

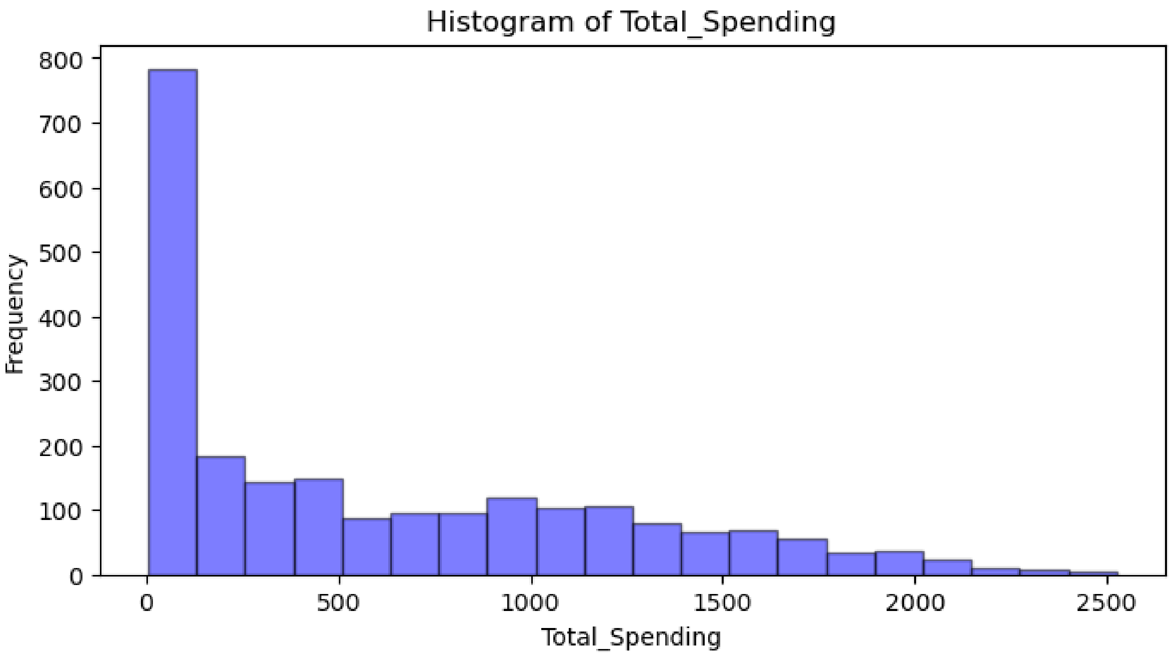

Histogram of AcceptedCmp2
Frequency
2000
1500
1000
500
0
0.0
0.2
0.4
0.6
0.8
1.0
AcceptedCmp2

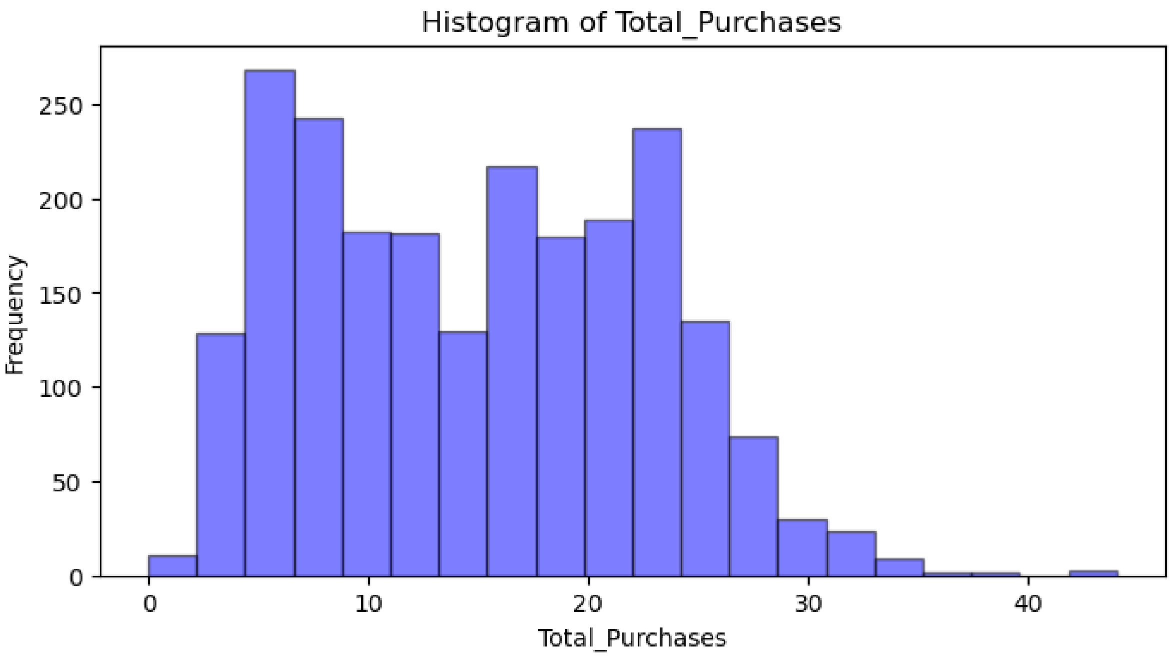

Histogram of Response
Frequency
2000
1750
1500
1250
1000
750
500
250
0
0.0
0.2
0.4
0.6
0.8
1.0
Response


Histogram of Complain
Frequency
0
500
1000
1500
2000
0.0
0.2
0.4
0.6
0.8
1.0
Complain


Histogram of total_Children
Frequency
0
200
400
600
800
1000
0.0
0.5
1.0
1.5
2.0
2.5
3.0
total_Children

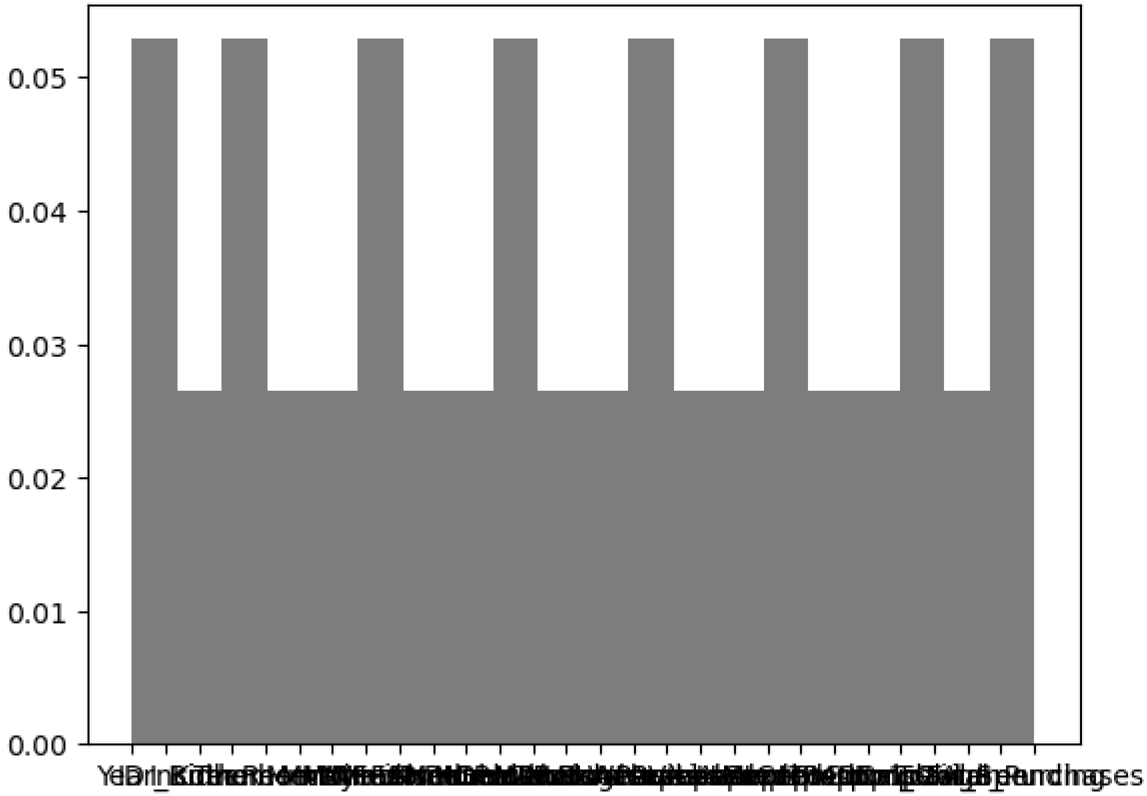

Histogram of Age
Frequency
0
50
100
150
200
250
300
350
400
40
60
80
100
120
Age





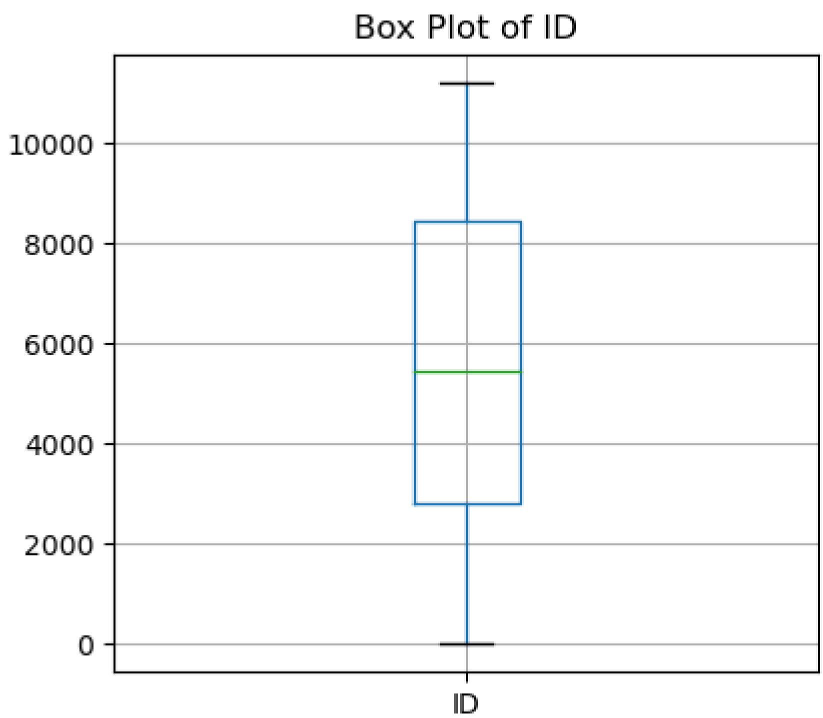
In [22]:
```
numeric_columns = data.select_dtypes(include=['float64', 'int64']).columns
plt.hist(numeric_columns,bins=20,alpha=0.5,density=True,histtype="stepfilled",color
```

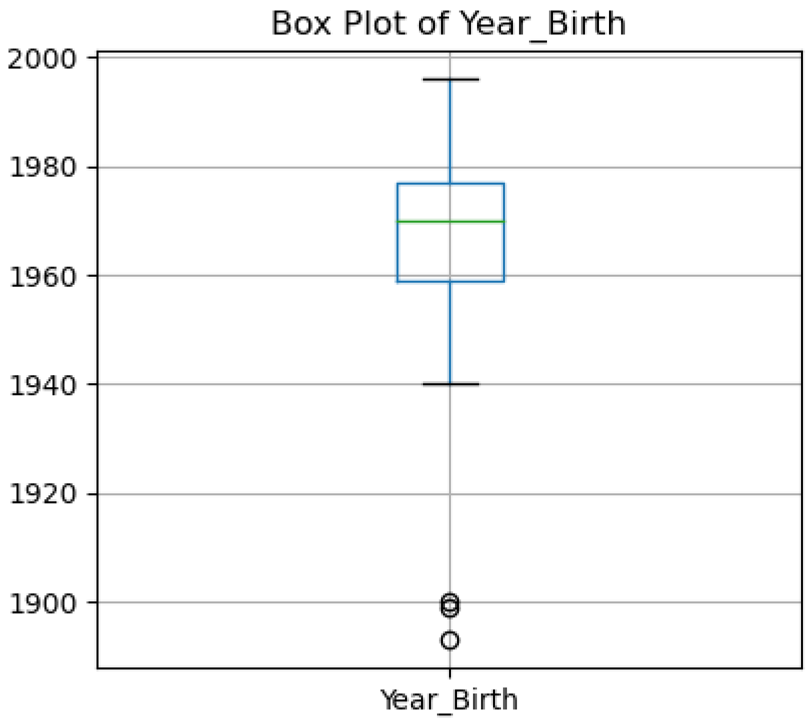
Out[22]:
```
(array([0.05291005, 0.02645503, 0.05291005, 0.02645503, 0.02645503,
        0.05291005, 0.02645503, 0.02645503, 0.05291005, 0.02645503,
        0.02645503, 0.05291005, 0.02645503, 0.02645503, 0.05291005,
        0.02645503, 0.02645503, 0.05291005, 0.02645503, 0.05291005]),
 array([ 0.  ,  1.35,  2.7 ,  4.05,  5.4 ,  6.75,  8.1 ,  9.45, 10.8 ,
        12.15, 13.5 , 14.85, 16.2 , 17.55, 18.9 , 20.25, 21.6 , 22.95,
        24.3 , 25.65, 27.  ]),
 [<matplotlib.patches.Polygon at 0x2c085d3ab10>])
```

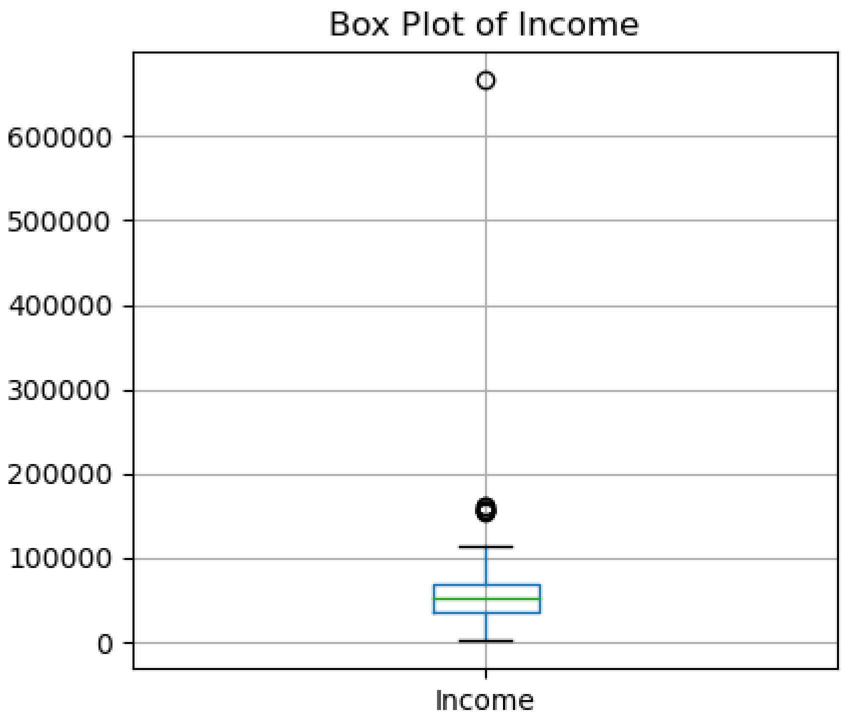


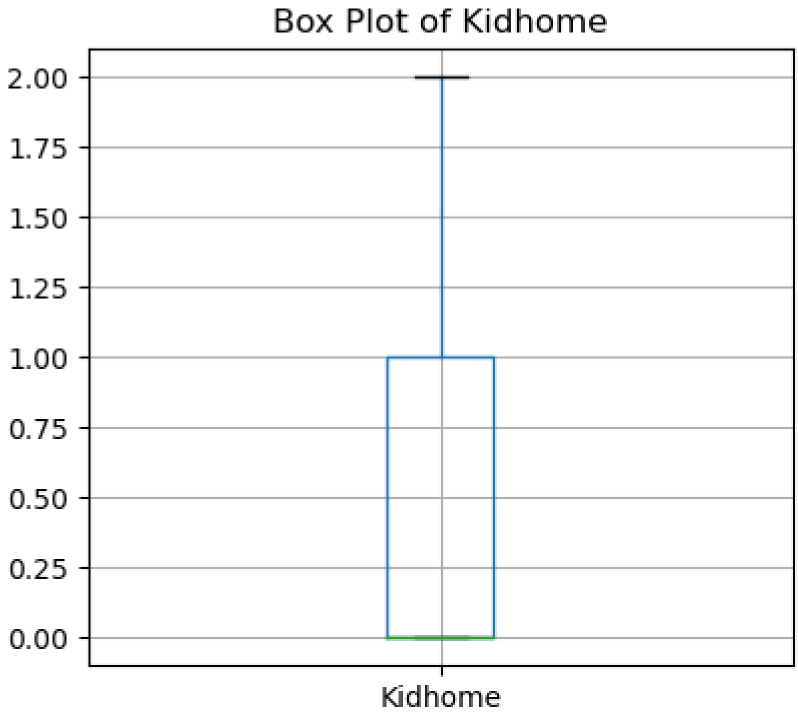
In [23]:
```
numeric_columns = data.select_dtypes(include=['float64', 'int64']).columns
for col in numeric_columns:
    plt.figure(figsize=(10, 4))
    plt.subplot(1, 2, 1)
    data.boxplot(column=col)
    plt.title(f'Box Plot of {col}')
```

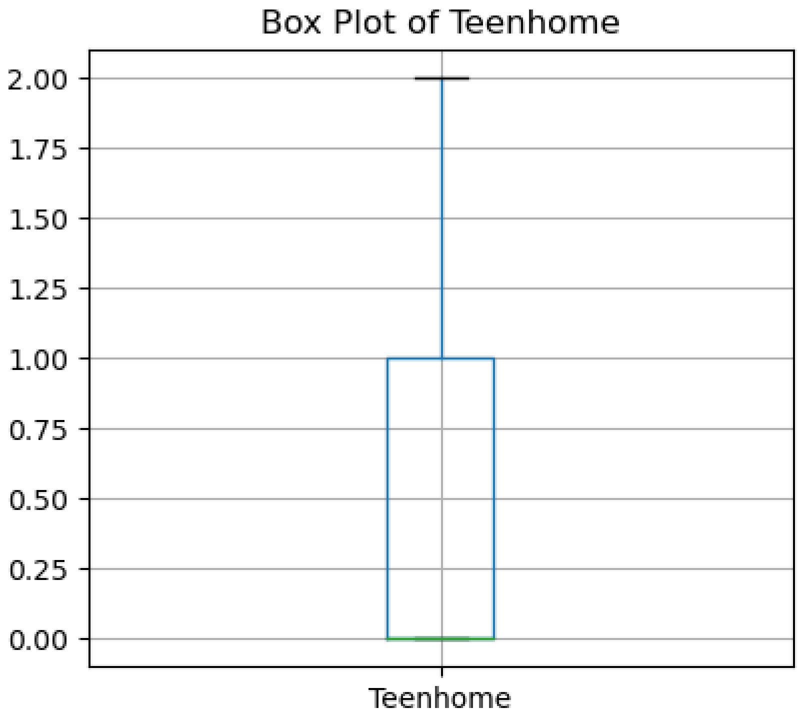
```
C:\Users\captr\AppData\Local\Temp\ipykernel_13208\2962160218.py:3: RuntimeWarning: 
More than 20 figures have been opened. Figures created through the pyplot interfac
e (`matplotlib.pyplot.figure`) are retained until explicitly closed and may consum
e too much memory. (To control this warning, see the rcParam `figure.max_open_warn
ing`). Consider using `matplotlib.pyplot.close()`.
  plt.figure(figsize=(10, 4))
```

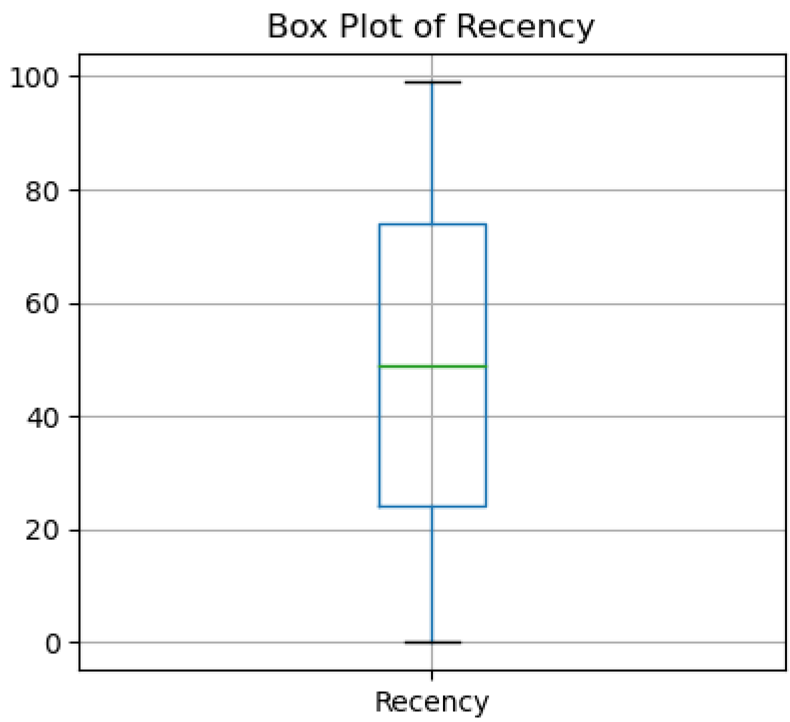


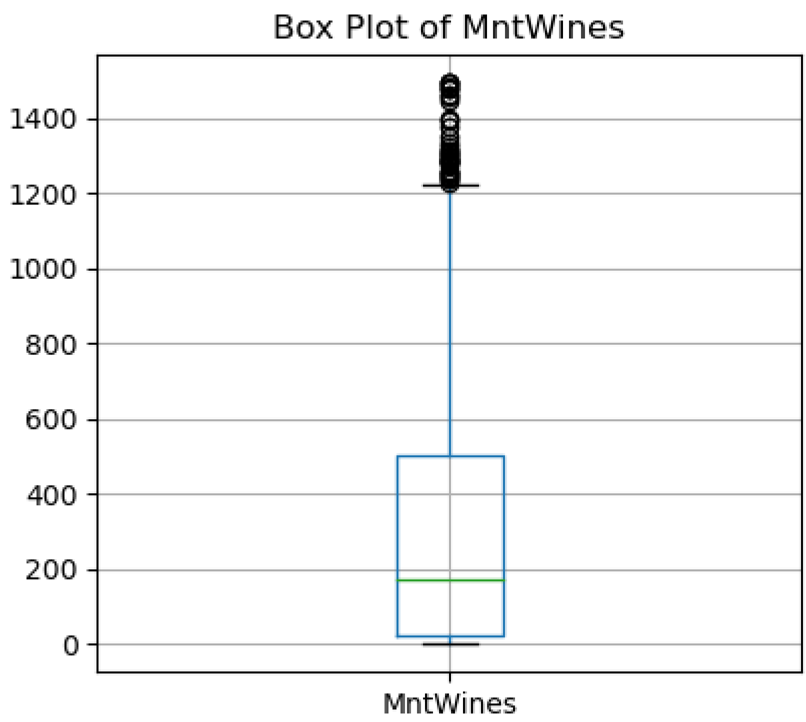

Box Plot of Year_Birth
2000
1980
1960
1940
1920
1900
Year_Birth

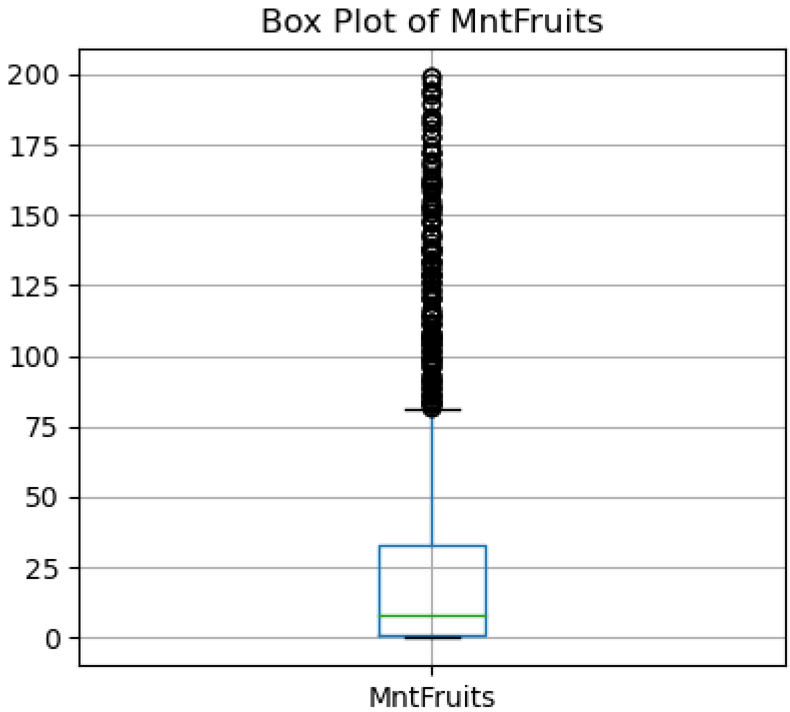

Box Plot of Income
600000
500000
400000
300000
200000
100000
0
Income

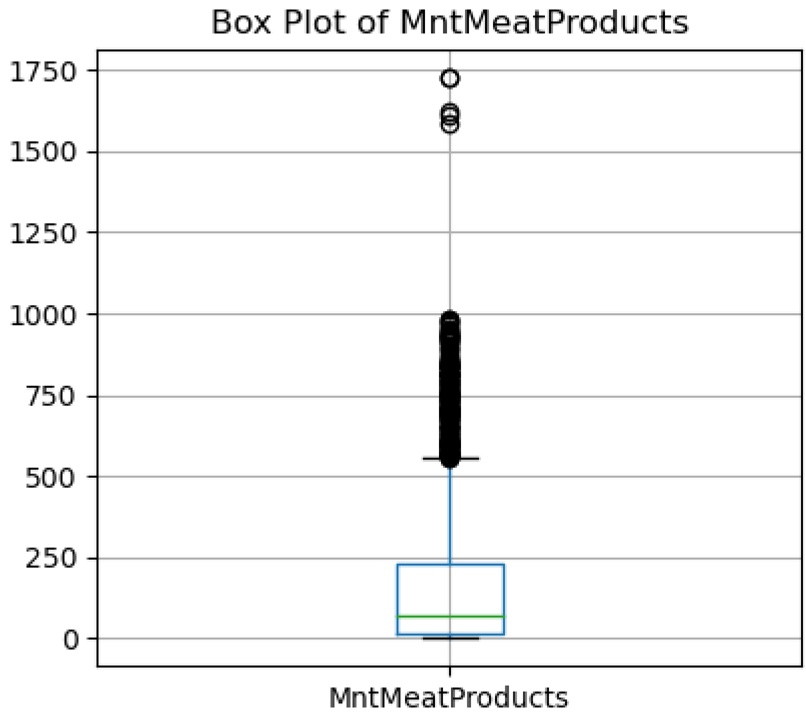

Box Plot of Kidhome
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
Kidhome

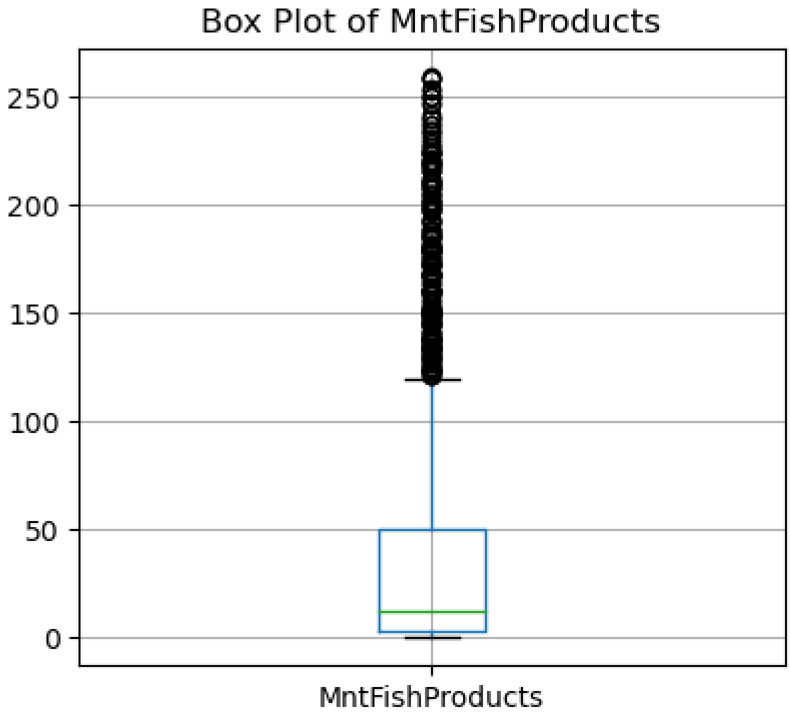

Box Plot of Teenhome
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
Teenhome

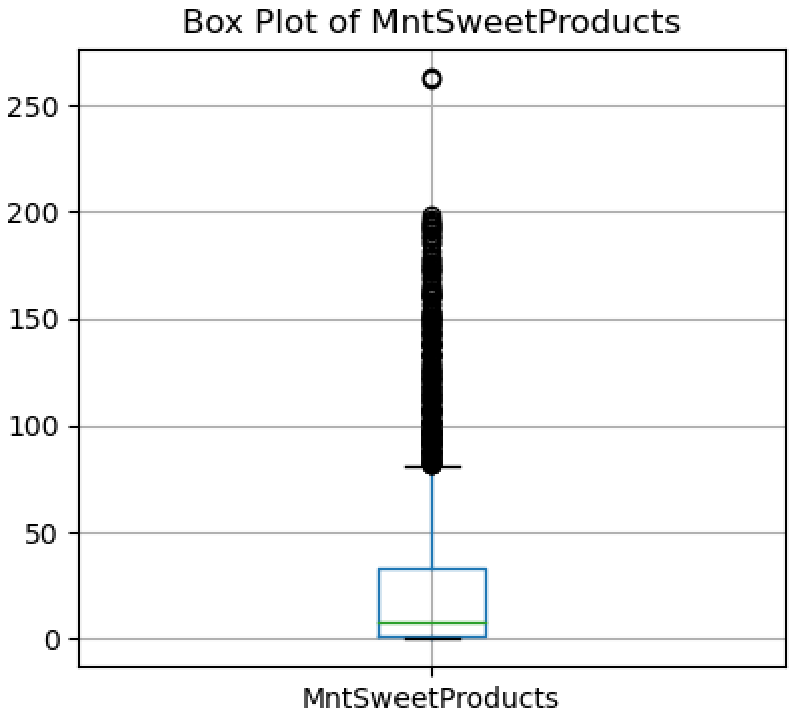

Box Plot of Recency
100
80
60
40
20
0
Recency

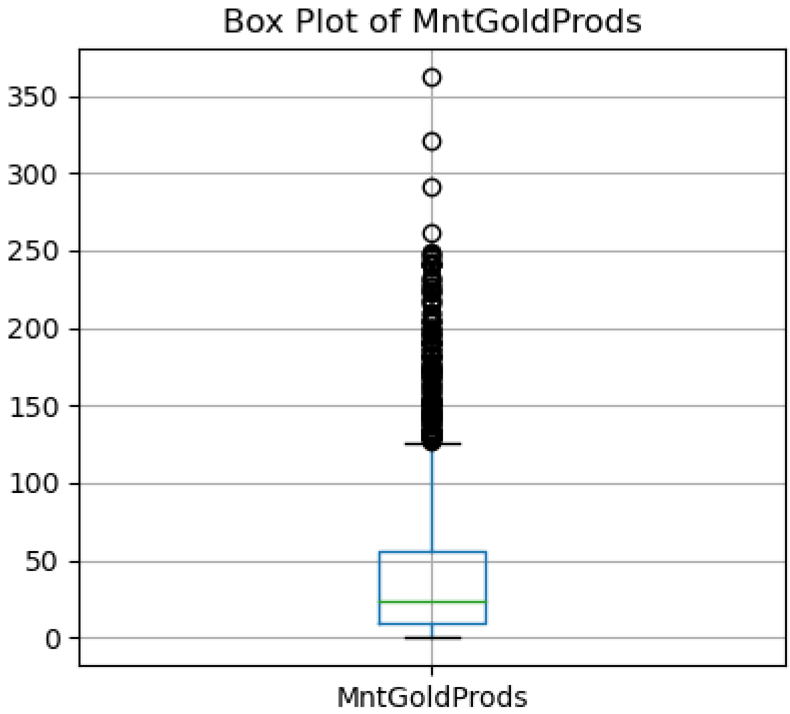

Box Plot of MntWines
1400
1200
1000
800
600
400
200
0
MntWines

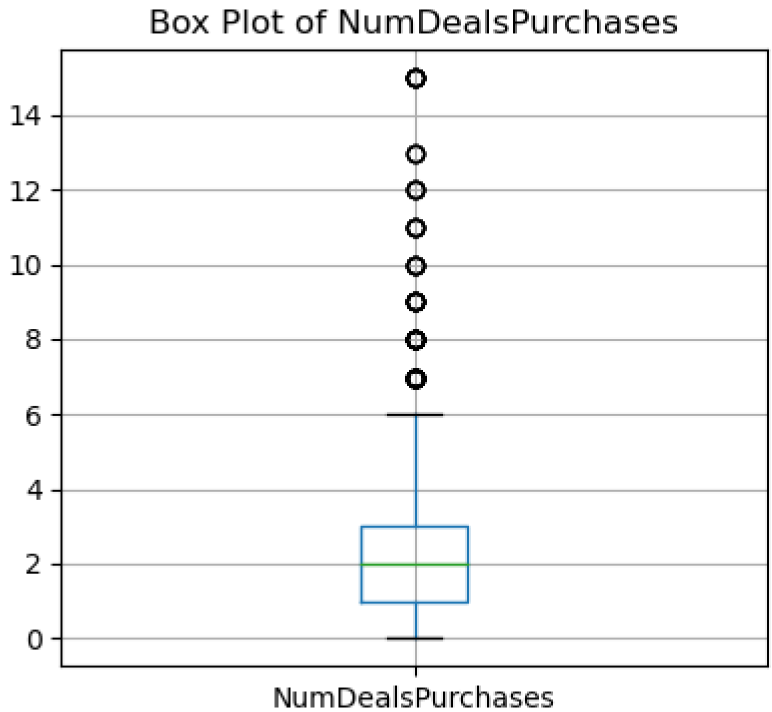

Box Plot of MntFruits
200
175
150
125
100
75
50
25
0
MntFruits

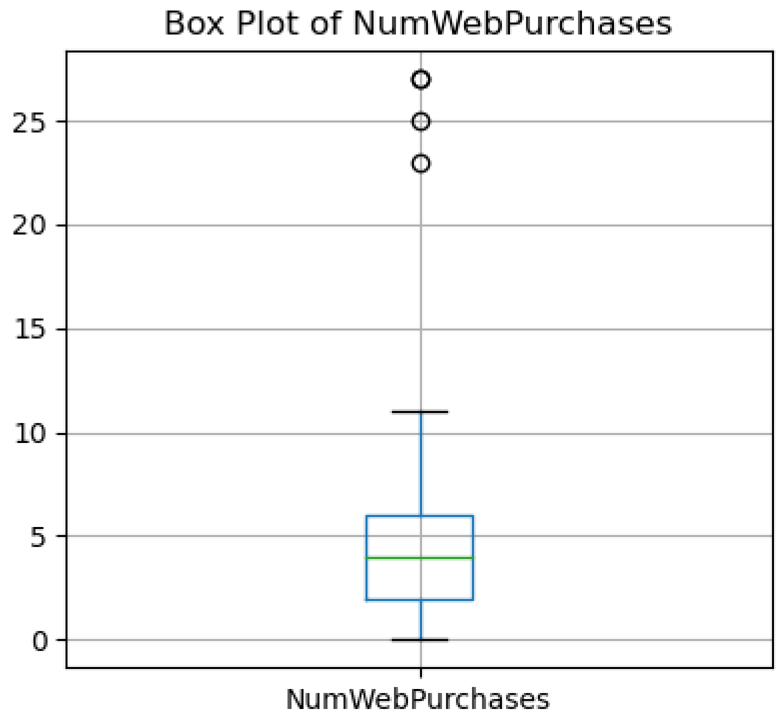

Box Plot of MntMeatProducts
1750
1500
1250
1000
750
500
250
0
MntMeatProducts

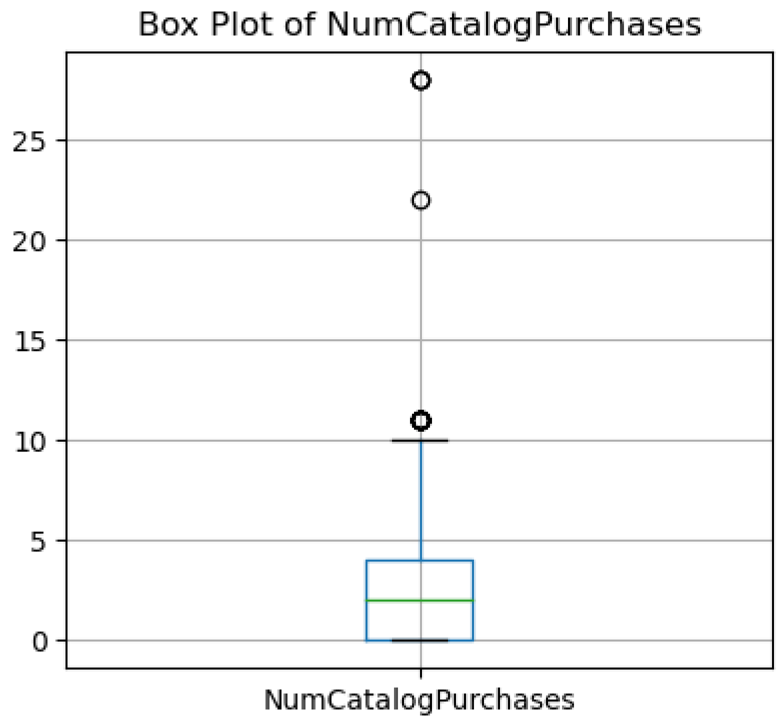

Box Plot of MntFishProducts
250
200
150
100
50
0
MntFishProducts


Box Plot of MntSweetProducts
250
200
150
100
50
0
MntSweetProducts

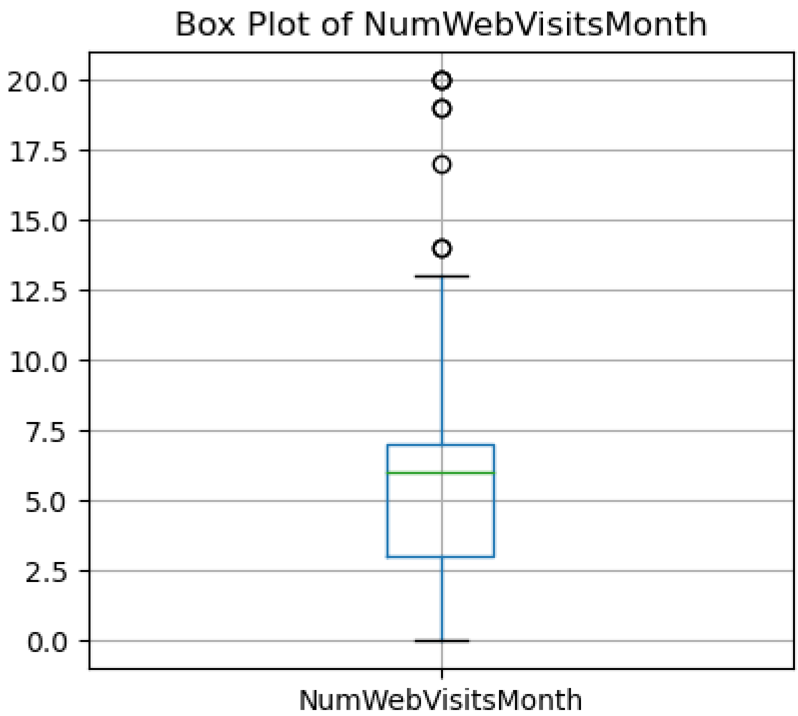

Box Plot of MntGoldProds
350
300
250
200
150
100
50
0
MntGoldProds

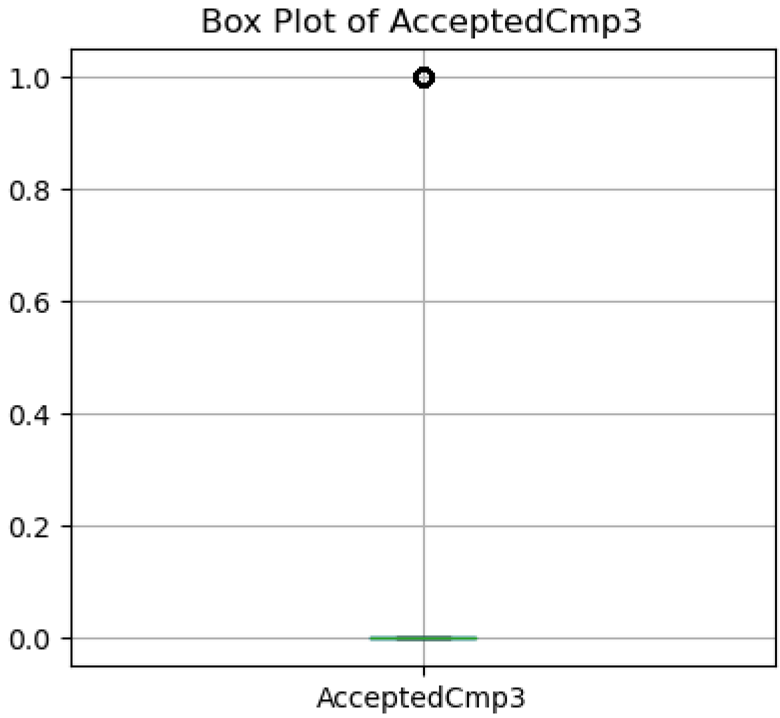

Box Plot of NumDealsPurchases
14
12
10
8
6
4
2
0
NumDealsPurchases

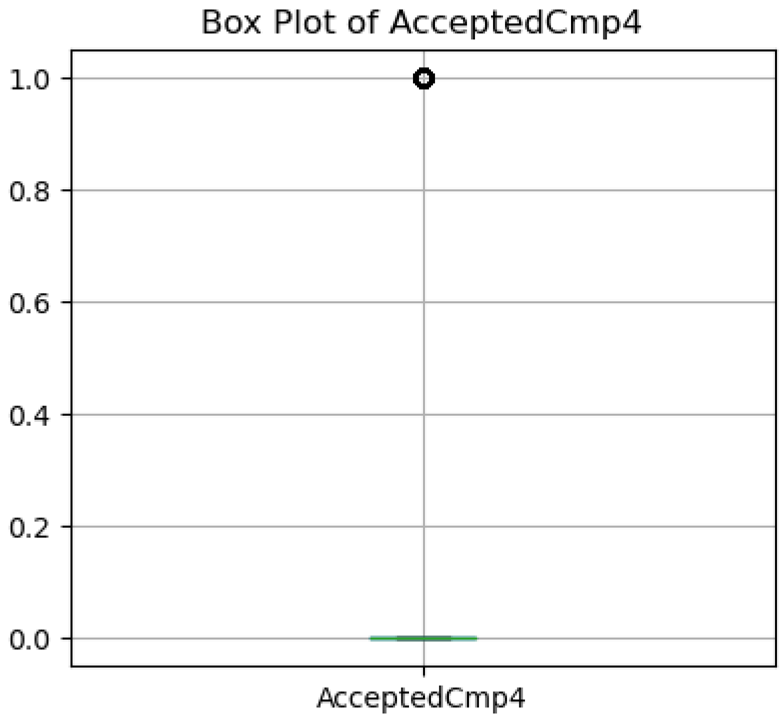

Box Plot of NumWebPurchases
25
20
15
10
5
0
NumWebPurchases

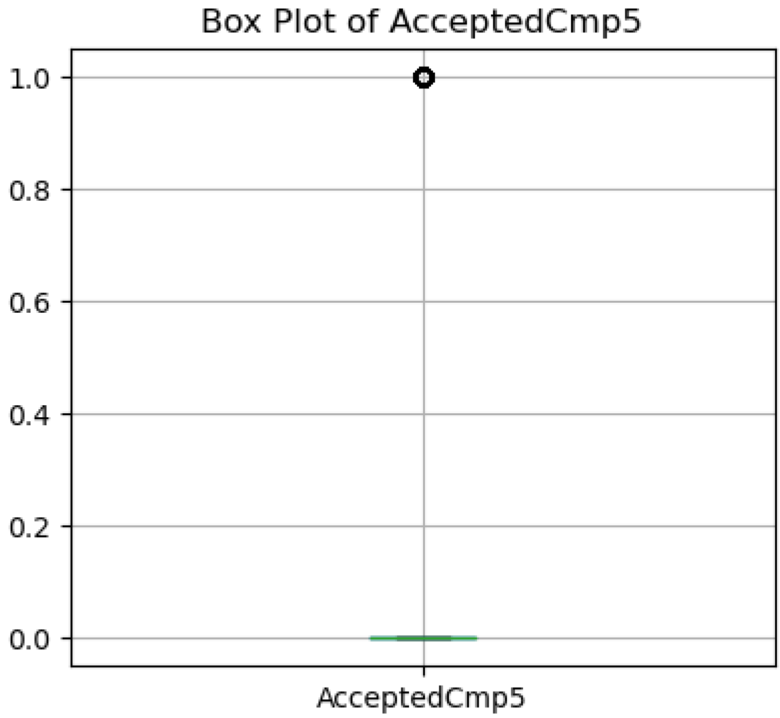

Box Plot of NumCatalogPurchases
25
20
15
10
5
0
NumCatalogPurchases

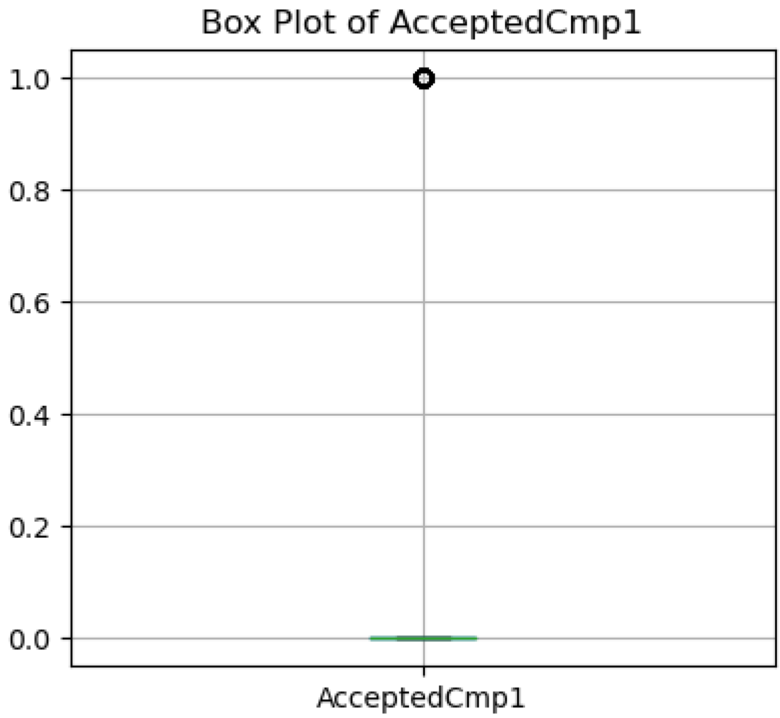

Box Plot of NumStorePurchases
12
10
8
6
4
2
0
NumStorePurchases

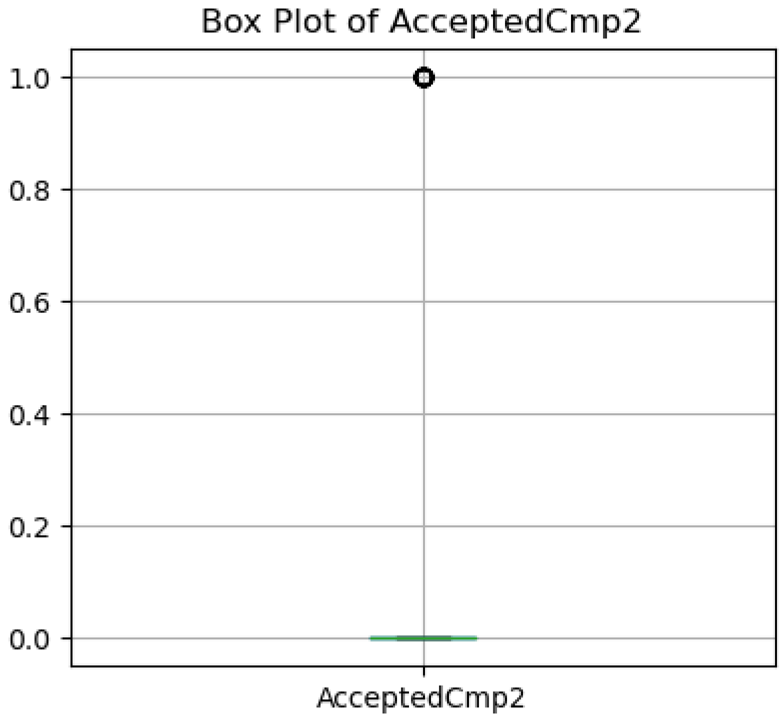

Box Plot of NumWebVisitsMonth
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
NumWebVisitsMonth

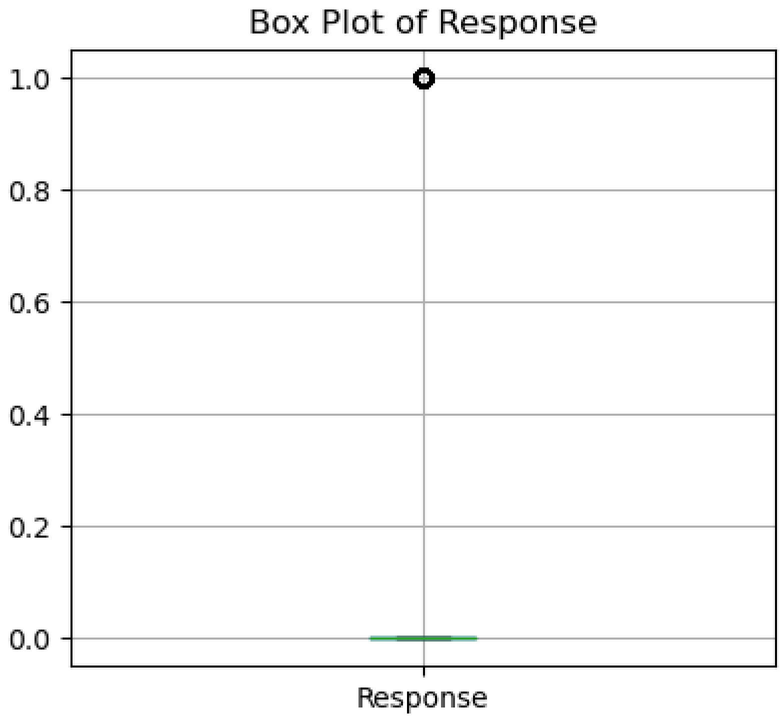

Box Plot of AcceptedCmp3
1.0
0.8
0.6
0.4
0.2
0.0
AcceptedCmp3

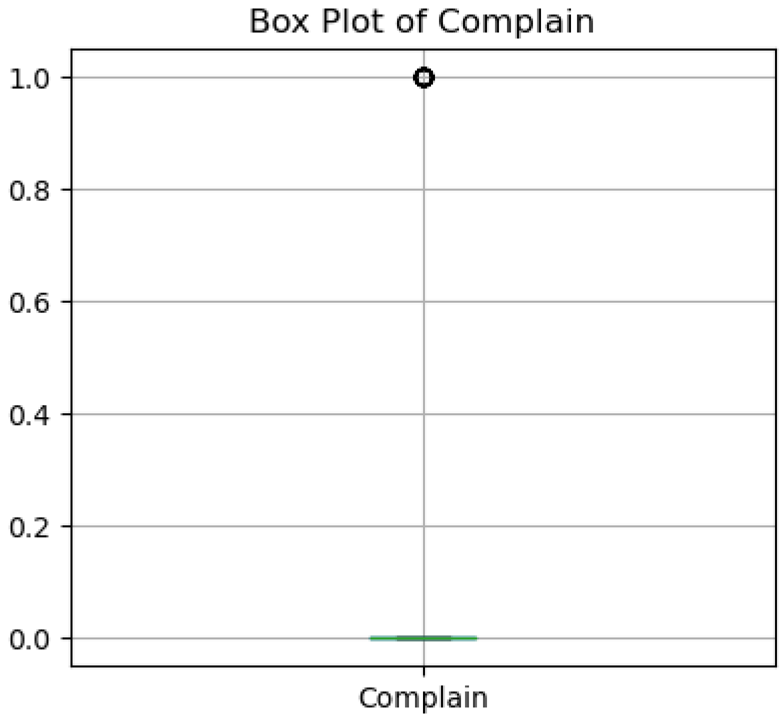

Box Plot of AcceptedCmp4
1.0
0.8
0.6
0.4
0.2
0.0
AcceptedCmp4

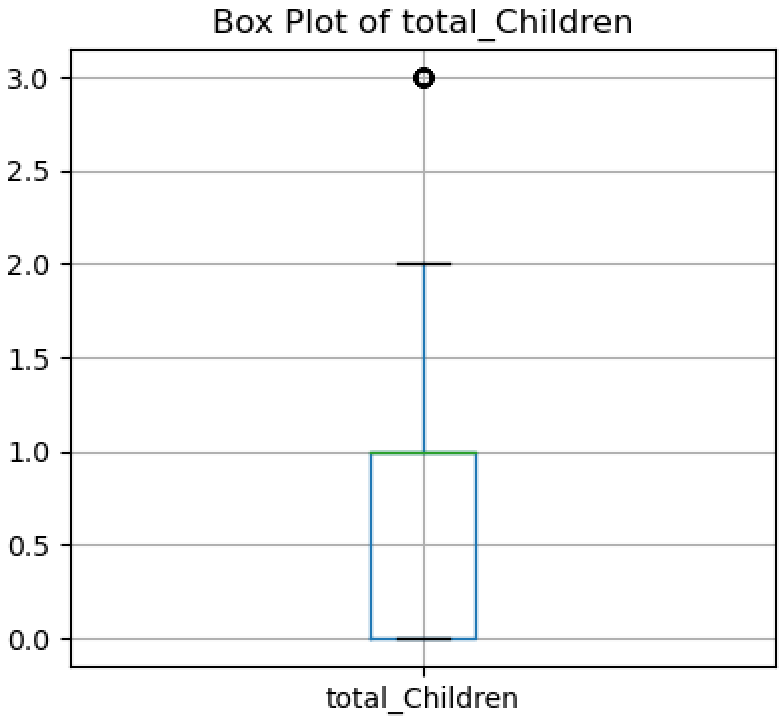

Box Plot of AcceptedCmp5
1.0
0.8
0.6
0.4
0.2
0.0
AcceptedCmp5

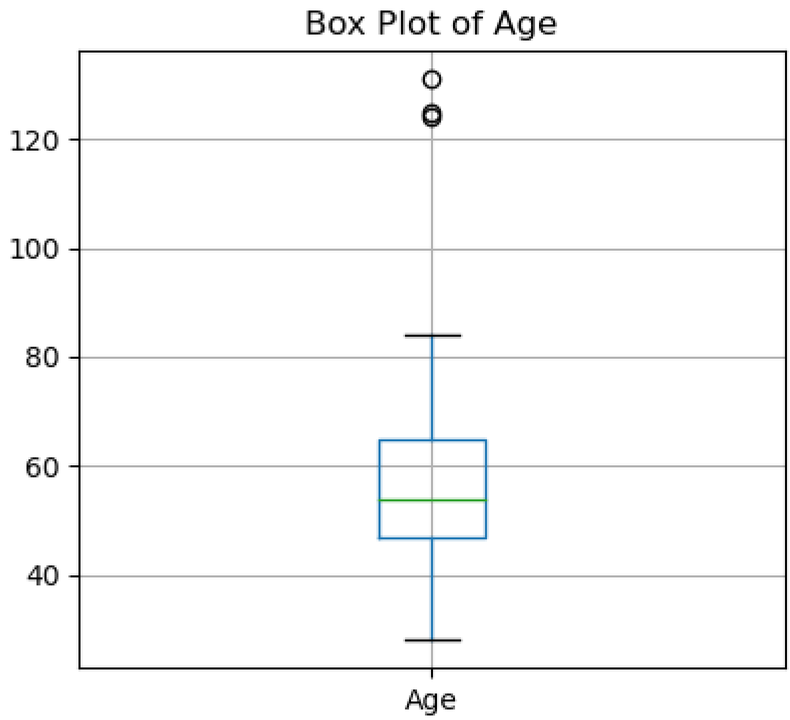

Box Plot of AcceptedCmp1
1.0
0.8
0.6
0.4
0.2
0.0
AcceptedCmp1

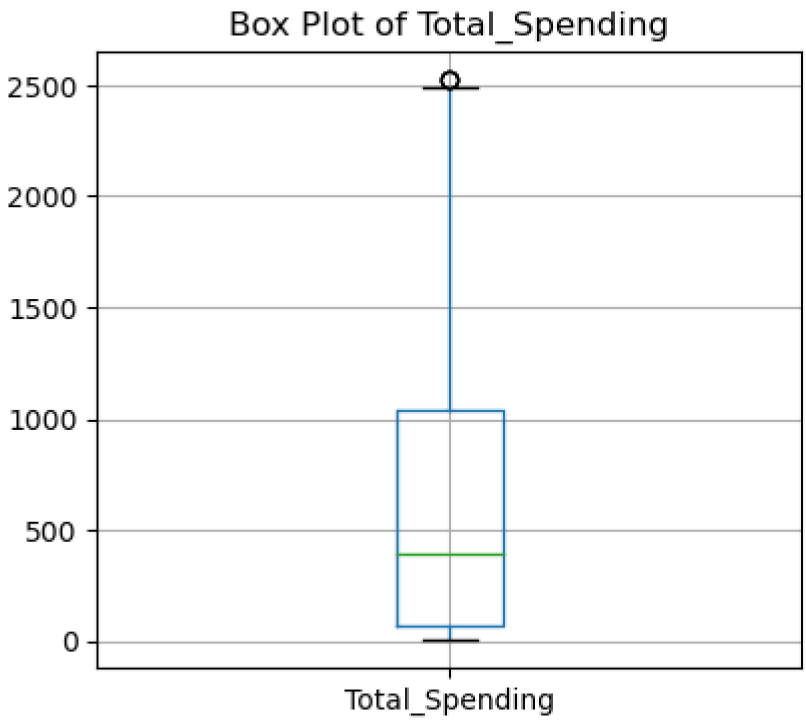

Box Plot of AcceptedCmp2
1.0
0.8
0.6
0.4
0.2
0.0
AcceptedCmp2

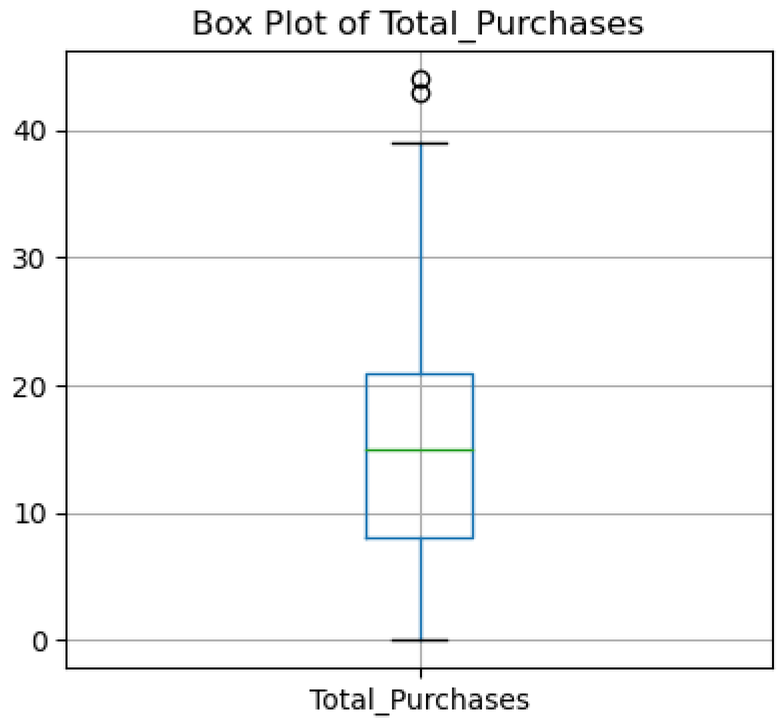

Box Plot of Response
1.0
0.8
0.6
0.4
0.2
0.0
Response


Box Plot of Complain
1.0
0.8
0.6
0.4
0.2
0.0
Complain


Box Plot of total_Children
3.0
2.5
2.0
1.5
1.0
0.5
0.0
total_Children


Box Plot of Age
120
100
80
60
40
Age


Box Plot of Total_Spending
2500
2000
1500
1000
500
0
Total_Spending



Applied ordinal and one-hot encoding based on the various types of categorical variables

In [24]:
```
import pandas as pd
from sklearn.preprocessing import OneHotEncoder, OrdinalEncoder
```

In [25]:
```
ordinal_columns = ['Education']
nominal_columns = ['Marital_Status','Country']
```

In [26]:
```
ordinal_encoder = OrdinalEncoder()
data[ordinal_columns] = ordinal_encoder.fit_transform(data[ordinal_columns])
```

In [27]:
```
onehot_encoder = OneHotEncoder(sparse=False, drop='first')  # drop='first' to avoid
nominal_encoded = onehot_encoder.fit_transform(data[nominal_columns])
```

```
C:\Users\captr\anaconda3\Lib\site-packages\sklearn\preprocessing\_encoders.py:972:
FutureWarning: `sparse` was renamed to `sparse_output` in version 1.2 and will be
removed in 1.4. `sparse_output` is ignored unless you leave `sparse` to its defaul
t value.
  warnings.warn(
```

In [28]:
```
nominal_encoded_df = pd.DataFrame(nominal_encoded, columns=onehot_encoder.get_featu
```

In [29]:
```
data = pd.concat([data, nominal_encoded_df], axis=1)
data.drop(columns=nominal_columns, inplace=True)
```

In [30]:
```
data.head()
```

Out[30]:

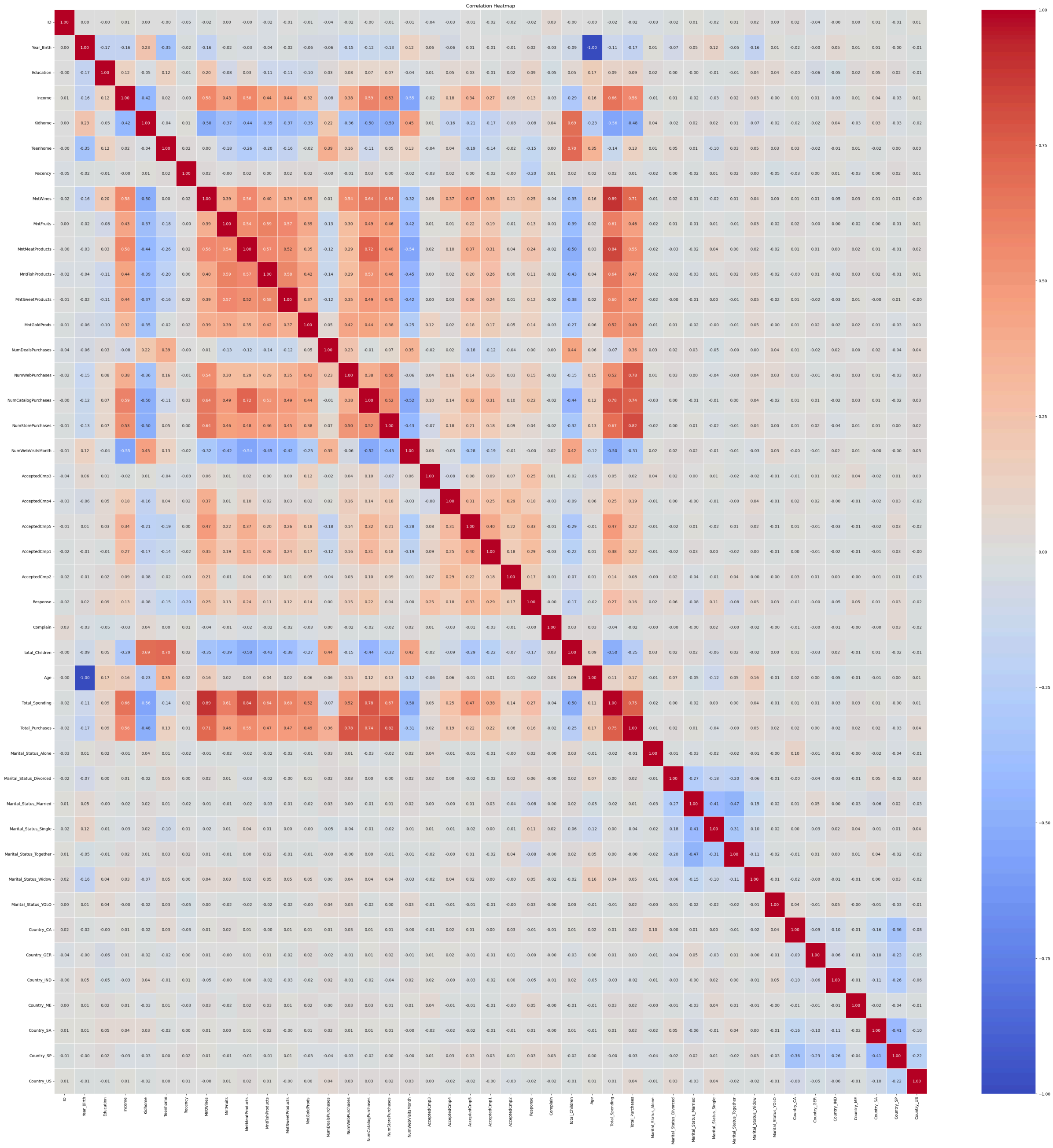
	ID	Year_Birth	Education	Income	Kidhome	Teenhome	Dt_Customer	Recency	MntWines
0	1826	1970	2.0	84835.0	0	0	6/16/14	0	189
1	1	1961	2.0	57091.0	0	0	6/15/14	0	464
2	10476	1958	2.0	67267.0	0	1	5/13/14	0	134
3	1386	1967	2.0	32474.0	1	1	2014-11-05 00:00:00	0	10
4	5371	1989	2.0	21474.0	1	0	2014-08-04 00:00:00	0	6

5 rows × 44 columns

Generated a heatmap to illustrate the correlation between different pairs of variables

In [32]:

```
import seaborn as sns
import matplotlib.pyplot as plt

# Select only numeric columns for the correlation matrix
numeric_data = data.select_dtypes(include=['float64', 'int64'])

# correlation matrix
correlation_matrix = numeric_data.corr()

# heatmap
plt.figure(figsize=(50, 50))
sns.heatmap(correlation_matrix, annot=True, fmt=".2f", cmap="coolwarm", linewidths=
plt.title("Correlation Heatmap")
plt.show()
```



Hypothesis Testing

i) Older individuals may not possess the same level of technological proficiency and may, therefore, lean toward traditional in-store shopping preferences

In [33]:
```python
import scipy.stats as stats
import statsmodels.api as sm
import statsmodels.formula.api as smf


age_instore_corr, age_instore_pval = stats.spearmanr(data['Age'], data['NumStorePur
age_online_corr, age_online_pval = stats.spearmanr(data['Age'], data['NumWebPurchas

print(f"Spearman correlation between Age and In-store Purchases: {age_instore_corr:
print(f"Spearman correlation between Age and Online Purchases: {age_online_corr:.2f

# Regression analysis for age and shopping preferences
instore_model = smf.ols('NumStorePurchases ~ Age', data=data).fit()
```

```
online_model = smf.ols('NumWebPurchases ~ Age', data=data).fit()

print("\nIn-store Purchases Regression Summary:\n", instore_model.summary())
print("\nOnline Purchases Regression Summary:\n", online_model.summary())
```

```
Spearman correlation between Age and In-store Purchases: 0.17, p-value: 0.0000
Spearman correlation between Age and Online Purchases: 0.16, p-value: 0.0000

In-store Purchases Regression Summary:
                             OLS Regression Results
==============================================================================
Dep. Variable:      NumStorePurchases   R-squared:                       0.016
Model:                            OLS   Adj. R-squared:                  0.016
Method:                 Least Squares   F-statistic:                     37.44
Date:                Fri, 01 Nov 2024   Prob (F-statistic):           1.11e-09
Time:                        13:21:28   Log-Likelihood:                -5800.2
No. Observations:                2240   AIC:                         1.160e+04
Df Residuals:                    2238   BIC:                         1.162e+04
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept      3.8696      0.321     12.048      0.000       3.240       4.499
Age            0.0348      0.006      6.119      0.000       0.024       0.046
==============================================================================
Omnibus:                      215.972   Durbin-Watson:                   1.701
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              221.533
Skew:                           0.718   Prob(JB):                     7.85e-49
Kurtosis:                       2.439   Cond. No.                         266.
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly s
pecified.

Online Purchases Regression Summary:
                             OLS Regression Results
==============================================================================
Dep. Variable:        NumWebPurchases   R-squared:                       0.021
Model:                            OLS   Adj. R-squared:                  0.021
Method:                 Least Squares   F-statistic:                     48.09
Date:                Fri, 01 Nov 2024   Prob (F-statistic):           5.30e-12
Time:                        13:21:28   Log-Likelihood:                -5443.4
No. Observations:                2240   AIC:                         1.089e+04
Df Residuals:                    2238   BIC:                         1.090e+04
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept      2.2286      0.274      8.137      0.000       1.692       2.766
Age            0.0336      0.005      6.935      0.000       0.024       0.043
==============================================================================
Omnibus:                      714.580   Durbin-Watson:                   1.793
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             4011.906
Skew:                           1.392   Prob(JB):                         0.00
Kurtosis:                       8.936   Cond. No.                         266.
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly s
pecified.
```

ii) Customers with children likely experience time constraints, making online shopping a more convenient option.

In [35]:

```
import scipy.stats as stats
import statsmodels.formula.api as smf


# Step 1: Perform Spearman's correlation test
children_online_corr, children_online_pval = stats.spearmanr(data["total_Children"]

print(f"Spearman correlation between Total Children and Online Purchases: {childrer

# Step 2: Regression analysis for Total Children and Online Purchases
online_model = smf.ols('NumWebPurchases ~ total_Children', data=data).fit()

print("\nOnline Purchases Regression Summary:\n", online_model.summary())
```

```
Spearman correlation between Total Children and Online Purchases: -0.19, p-value:
0.0000

Online Purchases Regression Summary:
                             OLS Regression Results
==============================================================================
Dep. Variable:        NumWebPurchases   R-squared:                       0.021
Model:                            OLS   Adj. R-squared:                  0.021
Method:                 Least Squares   F-statistic:                     48.99
Date:                Fri, 01 Nov 2024   Prob (F-statistic):           3.39e-12
Time:                        13:28:41   Log-Likelihood:                -5442.9
No. Observations:                2240   AIC:                         1.089e+04
Df Residuals:                    2238   BIC:                         1.090e+04
Df Model:                           1
Covariance Type:            nonrobust
==================================================================================
                     coef    std err          t      P>|t|      [0.025      0.975]
----------------------------------------------------------------------------------
Intercept          4.5990      0.094     49.107      0.000       4.415       4.783
total_Children    -0.5410      0.077     -6.999      0.000      -0.693      -0.389
==============================================================================
Omnibus:                      739.227   Durbin-Watson:                   1.781
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             4161.022
Skew:                           1.446   Prob(JB):                         0.00
Kurtosis:                       9.019   Cond. No.                         2.94
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly s
pecified.
```

iii) Sales at physical stores may face the risk of cannibalization by alternative distribution channels.

In [36]:

```
import scipy.stats as stats
import statsmodels.formula.api as smf

# Step 1: Perform Spearman's correlation between store and alternative channels
store_online_corr, store_online_pval = stats.spearmanr(data['NumStorePurchases'], c
store_catalog_corr, store_catalog_pval = stats.spearmanr(data['NumStorePurchases'],

print(f"Spearman correlation between Store and Online Purchases: {store_online_corr
print(f"Spearman correlation between Store and Catalog Purchases: {store_catalog_cc
```

```
# Step 2: Multiple regression analysis to evaluate impact on store sales
cannibalization_model = smf.ols('NumStorePurchases ~ NumWebPurchases + NumCatalogPu

print("\nMultiple Regression Summary for Store Purchases:\n", cannibalization_model
```

```
Spearman correlation between Store and Online Purchases: 0.67, p-value: 0.0000
Spearman correlation between Store and Catalog Purchases: 0.71, p-value: 0.0000

Multiple Regression Summary for Store Purchases:
                             OLS Regression Results
==============================================================================
Dep. Variable:      NumStorePurchases   R-squared:                       0.379
Model:                            OLS   Adj. R-squared:                  0.378
Method:                 Least Squares   F-statistic:                     681.7
Date:                Fri, 01 Nov 2024   Prob (F-statistic):          6.63e-232
Time:                        13:30:55   Log-Likelihood:                -5285.7
No. Observations:                2240   AIC:                         1.058e+04
Df Residuals:                    2237   BIC:                         1.059e+04
Df Model:                           2
Covariance Type:            nonrobust
==================================================================================
=====
                          coef    std err          t      P>|t|      [0.025
0.975]
----------------------------------------------------------------------------------
-----
Intercept               2.9459      0.099     29.898      0.000       2.753
3.139
NumWebPurchases         0.4184      0.021     19.864      0.000       0.377
0.460
NumCatalogPurchases     0.4264      0.020     21.296      0.000       0.387
0.466
==============================================================================
Omnibus:                      157.310   Durbin-Watson:                   1.770
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              769.666
Skew:                           0.059   Prob(JB):                    7.40e-168
Kurtosis:                       5.869   Cond. No.                         10.9
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly s
pecified.
```

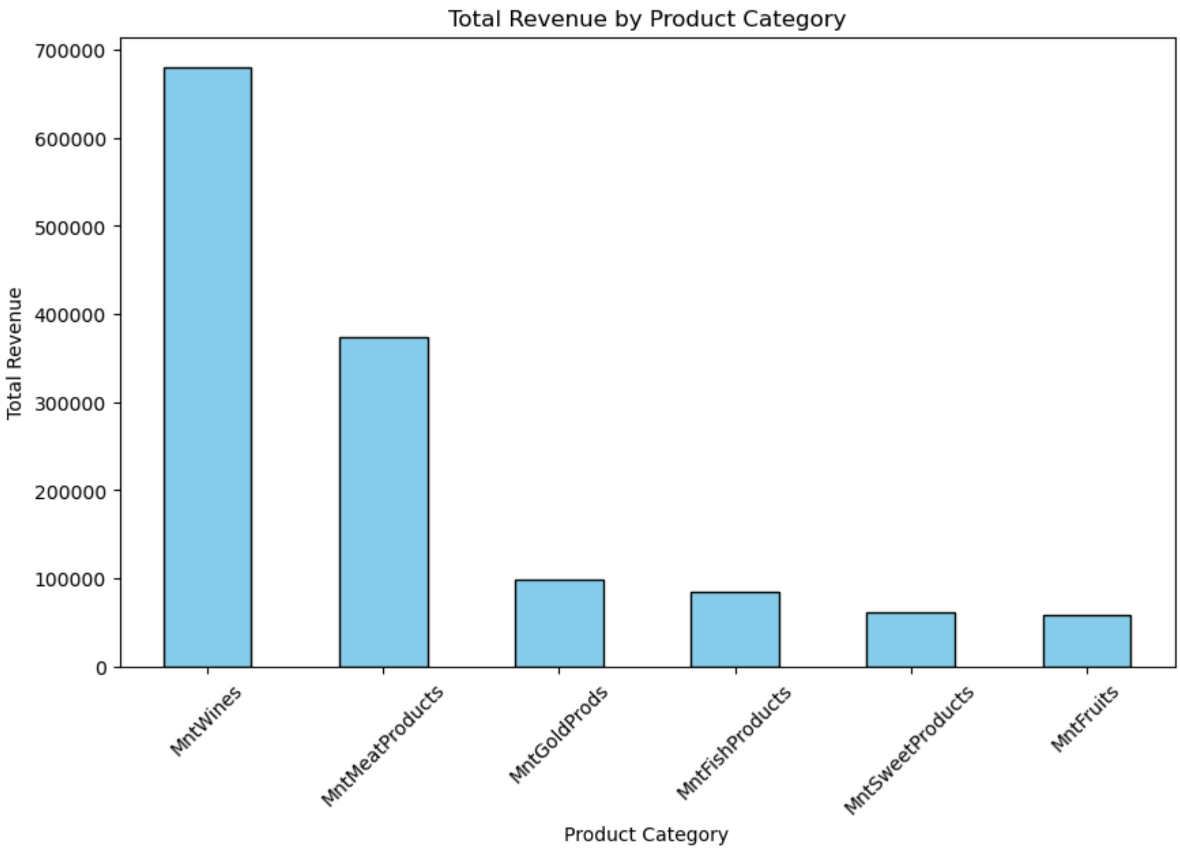
Data Visualisation

Identifying the top-performing products and those with the lowest revenue.

In [45]:

```
import matplotlib.pyplot as plt

# Calculate total revenue for each product
product_revenue = data[['MntWines', 'MntFruits', 'MntMeatProducts', 'MntFishProduct

# Sort revenues in descending order for clear visualization
product_revenue = product_revenue.sort_values(ascending=False)

# Plot the revenue for each product
plt.figure(figsize=(10, 6))
product_revenue.plot(kind='bar', color='skyblue', edgecolor='black')
plt.title("Total Revenue by Product Category")
plt.xlabel("Product Category")
```

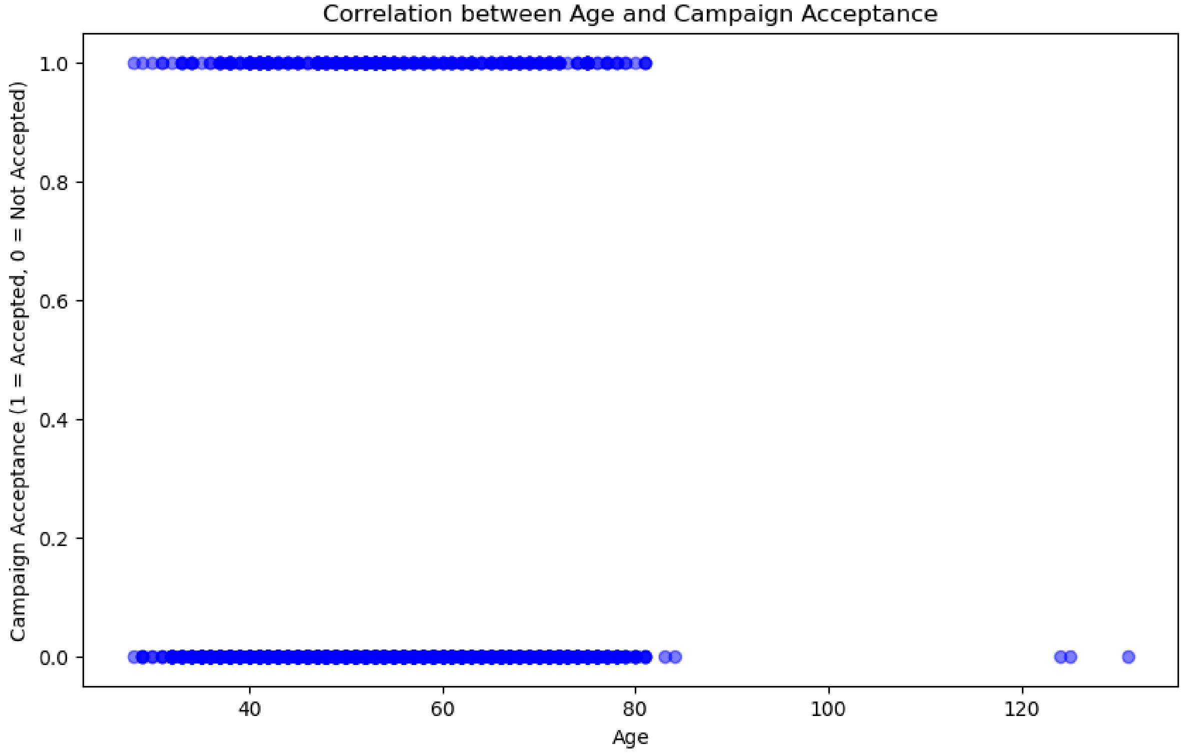
```
plt.ylabel("Total Revenue")
plt.xticks(rotation=45)
plt.show()
```



Examining if there is a correlation between customers' age and the acceptance rate of the last campaign

In [47]:

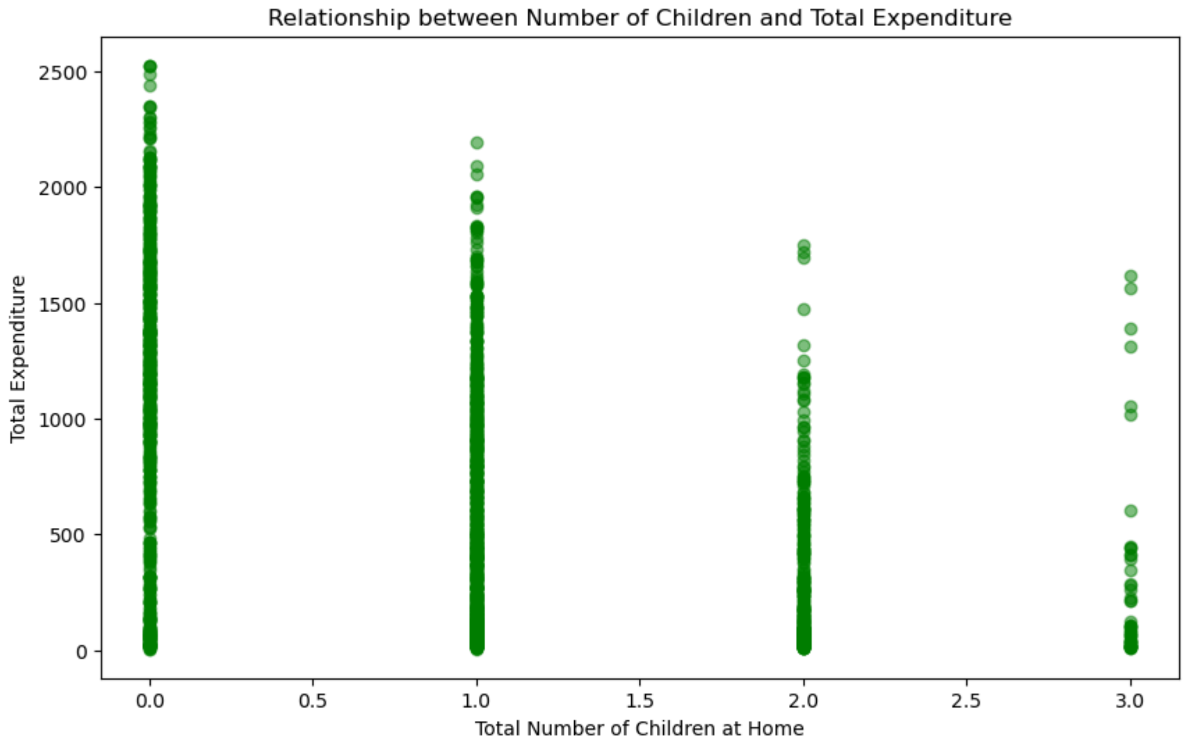
```
import matplotlib.pyplot as plt
import scipy.stats as stats


from datetime import datetime
data['Age'] = datetime.now().year - data['Year_Birth']

# Step 1: Calculated acceptance rate by age group
age_response = data.groupby('Age')['Response'].mean()

# Step 2: Scatter plotted for visual analysis
plt.figure(figsize=(10, 6))
plt.scatter(data['Age'], data['Response'], alpha=0.5, color='blue')
plt.title("Correlation between Age and Campaign Acceptance")
plt.xlabel("Age")
plt.ylabel("Campaign Acceptance (1 = Accepted, 0 = Not Accepted)")
plt.show()

# Step 4: Calculate Spearman correlation to measure the relationship
correlation, p_value = stats.spearmanr(data['Age'], data['Response'])
print(f"Spearman Correlation between Age and Campaign Acceptance: {correlation:.2f}

# Interpretation
if p_value < 0.05:
    print("There is a statistically significant correlation between age and campaig
else:
    print("There is no statistically significant correlation between age and campai
```




```
Spearman Correlation between Age and Campaign Acceptance: -0.02, p-value: 0.3268
There is no statistically significant correlation between age and campaign accepta
nce.
```

Determining the country with the highest number of customers who accepted the last campaign

In [52]:
```
plt.figure(figsize=(10, 6))
plt.scatter(data['total_Children'], data['Total_Spending'], alpha=0.5, color='greer
plt.title("Relationship between Number of Children and Total Expenditure")
plt.xlabel("Total Number of Children at Home")
plt.ylabel("Total Expenditure")
plt.show()
```



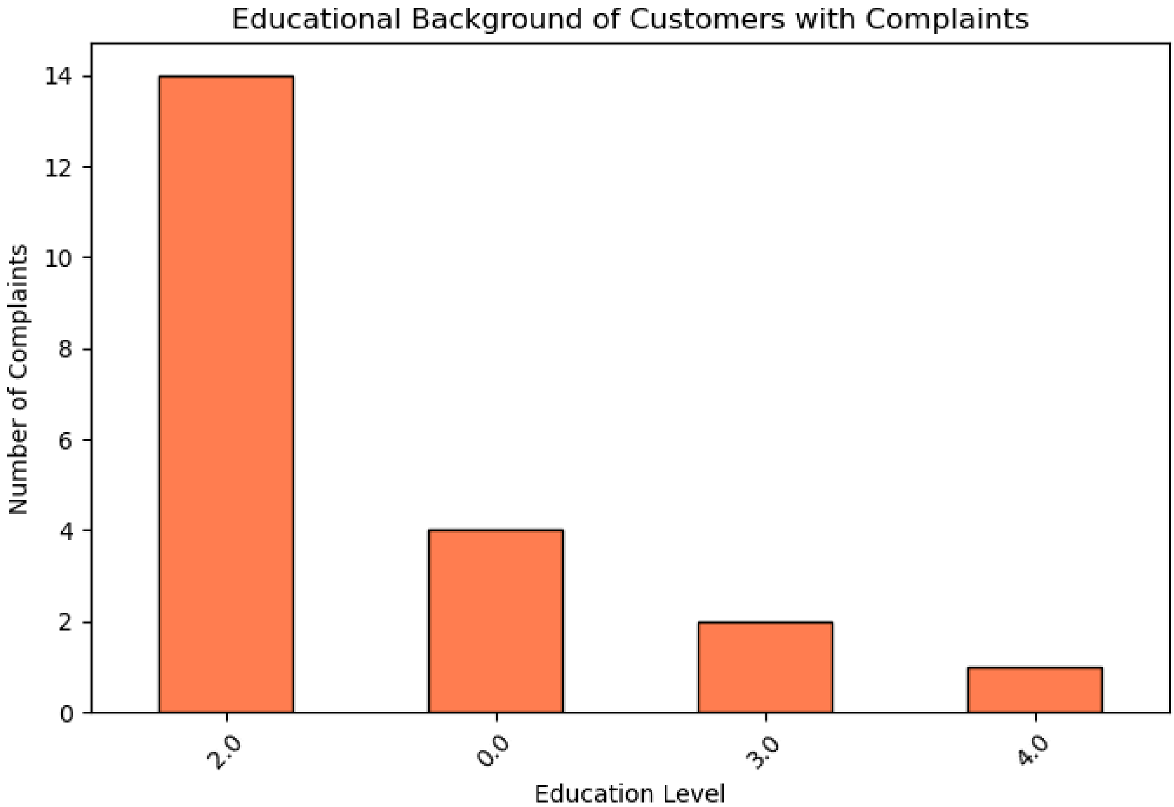
Investigating if there is a discernible pattern in the number of children at home and the total expenditure

In [55]:
```
correlation, p_value = stats.spearmanr(data['total_Children'], data['Total_Spending
print(f"Spearman Correlation between Total Children and Total Expenditure: {correla
if p_value < 0.05:
    print("There is a statistically significant relationship between the number of
else:
    print("There is no statistically significant relationship between the number of
```

```
Spearman Correlation between Total Children and Total Expenditure: -0.48, p-value:
0.0000
There is a statistically significant relationship between the number of children a
t home and total expenditure.
```

Analyzed the educational background of customers who lodged complaints in the last two years

In [56]:
```
import matplotlib.pyplot as plt


complaint_data = data[data['Complain'] == 1]


education_complaints = complaint_data['Education'].value_counts()


plt.figure(figsize=(8, 5))
education_complaints.plot(kind='bar', color='coral', edgecolor='black')
plt.title("Educational Background of Customers with Complaints")
plt.xlabel("Education Level")
plt.ylabel("Number of Complaints")
plt.xticks(rotation=45)
plt.show()


print("Education Background Counts for Complaints:\n", education_complaints)
```




```
Education Background Counts for Complaints:
 Education
2.0    14
0.0     4
3.0     2
4.0     1
Name: count, dtype: int64
```