In [1]:

```
%pwd
```

Out[1]:

```
'/Users/sobia/Applied_Data'
```

In [2]:

```
import numpy as np
import pandas as pd
from  pandas import Series, DataFrame
from pandas.plotting import scatter_matrix
from pylab import rcParams
import matplotlib.pyplot as plt
import seaborn as sb
```

In [3]:

```
%matplotlib inline
rcParams['figure.figsize']=5,4
sb.set_style=['whitegrid']
```

In [4]:

```
#Firstly I tried to handle missing values, sometimes missing values
#couldn't be identified as NAN. I wrote down all possibilities to resolve this issue.
missing_values = ["n/a", "na", "--"]
marketing_df= pd.read_csv("marketing_campaign.csv", na_values = missing_values, sep=';')
marketing_df.head()
```

Out[4]:

	ID	Year_Birth	Education	Marital_Status	Income	Kidhome	Teenhome	Dt_Customer	Recency	MntWines	...	NumWel
0	5524	1957	Graduation	Single	58138.0	0	0	04.09.2012	58	635	...	
1	2174	1954	Graduation	Single	46344.0	1	1	08.03.2014	38	11	...	
2	4141	1965	Graduation	Together	71613.0	0	0	21.08.2013	26	426	...	
3	6182	1984	Graduation	Together	26646.0	1	0	10.02.2014	26	11	...	
4	5324	1981	PhD	Married	58293.0	1	0	19.01.2014	94	173	...	

5 rows × 29 columns

In [5]:

```
#create a subset ,can see the data structure here,both quantitaive and categorical are pr
esent
dfM=marketing_df.iloc[:, [2,3,4,9,10,11,12,13]]
dfM.head(),dfM.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2240 entries, 0 to 2239
Data columns (total 8 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Education         2240 non-null   object
 1   Marital_Status    2240 non-null   object
 2   Income            2216 non-null   float64
 3   MntWines          2240 non-null   int64
 4   MntFruits         2240 non-null   int64
 5   MntMeatProducts   2240 non-null   int64
 6   MntFishProducts   2240 non-null   int64
 7   MntSweetProducts  2240 non-null   int64
dtypes: float64(1), int64(5), object(2)
```

```
memory usage: 140.1+ KB
```

Out[5]:

```
(    Education Marital_Status   Income  MntWines  MntFruits  MntMeatProducts  \
 0  Graduation         Single  58138.0       635         88              546
 1  Graduation         Single  46344.0        11          1                6
 2  Graduation       Together  71613.0       426         49              127
 3  Graduation       Together  26646.0        11          4               20
 4         PhD        Married  58293.0       173         43              118

    MntFishProducts  MntSweetProducts
 0              172                88
 1                2                 1
 2              111                21
 3               10                 3
 4               46                27  ,
 None)
```

In [6]:

```
# no of missing values in each column.
missing_values = ["n/a", "na", "--"]
print(" \nCount total missing_values at each column in a DataFrame : \n\n",
      dfM.isnull().sum())
#One option is to remove the rows with missing values, as the number of outliers in the w
hole data is too little.
```

```
Count total missing_values at each column in a DataFrame :

 Education            0
Marital_Status        0
Income               24
MntWines              0
MntFruits             0
MntMeatProducts       0
MntFishProducts       0
MntSweetProducts      0
dtype: int64
```

In [9]:

```
#decided to replace it with median insted of mean becasue
#mean doesn't preserve the relationship among variables.
median=dfM['Income'].median()
median
```

Out[9]:

```
51381.5
```

In [11]:

```
dfM_=dfM.fillna(median)
```

In [12]:

```
#check again for missing values?
print(dfM_['Income'].isnull().values.any())
```

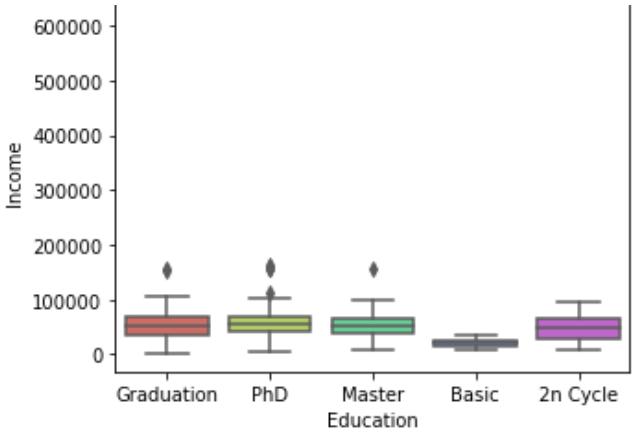
```
False
```

In [15]:

```
#we can see outliers here , they are not too much
sb.boxplot(y='Income',x='Education',data=dfM_,palette='hls')
```

Out[15]:

```
<AxesSubplot:xlabel='Education', ylabel='Income'>
```

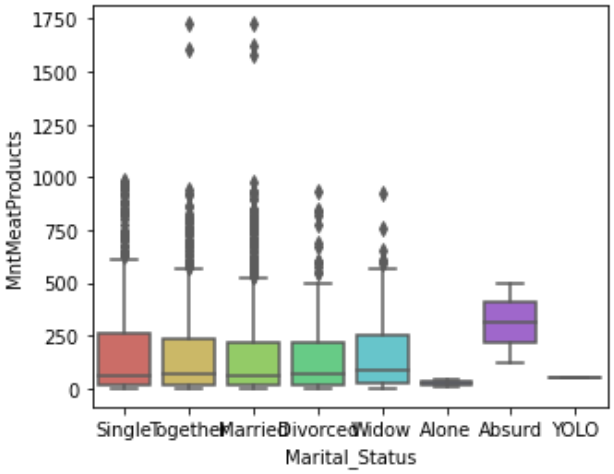


In [16]:

```
#As compare to Income there are a lot of outliers
sb.boxplot(y='MntMeatProducts',x='Marital_Status',data=dfM_,palette='hls')
```

Out[16]:

```
<AxesSubplot:xlabel='Marital_Status', ylabel='MntMeatProducts'>
```



In [18]:

```
#in order to get more information about the data set , used describe function to get a su
mmary
# as we can see here more than 75% people have income less than 68289.7500 same as 75% pp
l spend less than 232 on meat products
dfM_.describe()
```

Out[18]:

	Income	MntWines	MntFruits	MntMeatProducts	MntFishProducts	MntSweetProducts
count	2240.000000	2240.000000	2240.000000	2240.000000	2240.000000	2240.000000
mean	52237.975446	303.935714	26.302232	166.950000	37.525446	27.062946
std	25037.955891	336.597393	39.773434	225.715373	54.628979	41.280498
min	1730.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	35538.750000	23.750000	1.000000	16.000000	3.000000	1.000000
50%	51381.500000	173.500000	8.000000	67.000000	12.000000	8.000000
75%	68289.750000	504.250000	33.000000	232.000000	50.000000	33.000000
max	666666.000000	1493.000000	199.000000	1725.000000	259.000000	263.000000

In [19]:

```
#To remove outliers , find out the quantiles of income
IQ1=dfM_.Income.quantile(0.25)
IQ3=dfM_.Income.quantile(0.75)
```

```
IQ1,IQ3
```

Out[19]:

```
(35538.75, 68289.75)
```

In [20]:

```
Income_IQR=IQ3-IQ1
Income_lower_limit=IQ1-1.5*Income_IQR
Income_Upper_Limit=IQ3+1.5*Income_IQR
Income_IQR,Income_lower_limit,Income_Upper_Limit
```

Out[20]:

```
(32751.0, -13587.75, 117416.25)
```

In [21]:

```
#we obtained only 8 outliers but in above boxplot we were able to see less that 8 because
#there we extreme values
Incomedetecting_outliers=dfM_[(dfM_.Income<Income_lower_limit)|(dfM_.Income>Income_Upper_
Limit)]
Incomedetecting_outliers
```

Out[21]:

	Education	Marital_Status	Income	MntWines	MntFruits	MntMeatProducts	MntFishProducts	MntSweetProducts
164	PhD	Married	157243.0	20	2	1582	1	2
617	PhD	Together	162397.0	85	1	16	2	1
655	Graduation	Divorced	153924.0	1	1	1	1	1
687	PhD	Married	160803.0	55	16	1622	17	3
1300	Master	Together	157733.0	39	1	9	2	0
1653	Graduation	Together	157146.0	1	0	1725	2	1
2132	PhD	Married	156924.0	2	1	2	1	1
2233	Graduation	Together	666666.0	9	14	18	8	1

In [28]:

```
dfM_Nout=dfM_[(dfM_.Income>Income_lower_limit)&(dfM_.Income<Income_Upper_Limit)]
dfM_Nout.head()
```

Out[28]:

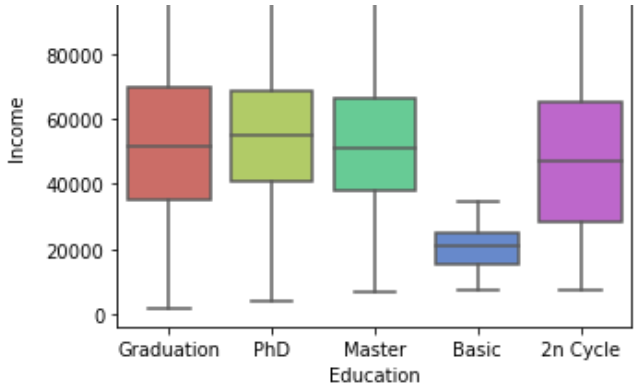
	Education	Marital_Status	Income	MntWines	MntFruits	MntMeatProducts	MntFishProducts	MntSweetProducts
0	Graduation	Single	58138.0	635	88	546	172	88
1	Graduation	Single	46344.0	11	1	6	2	1
2	Graduation	Together	71613.0	426	49	127	111	21
3	Graduation	Together	26646.0	11	4	20	10	3
4	PhD	Married	58293.0	173	43	118	46	27

In [23]:

```
#after removing outliers , made boxplot again to check outliers as we can see
#still there is outlier,it is to note that some observation has high income
sb.boxplot(y='Income',x='Education',data=dfM_Nout,palette='hls')
```

Out[23]:

```
<AxesSubplot:xlabel='Education', ylabel='Income'>
```



In [79]:

```
#So,I decided to find out extreme values , as we deducted total 15 outliers
#which is more than previous calculation.
min_thresold,max_thresold=dfM_.Income.quantile([0.001,0.995])
min_thresold,max_thresold
```

Out[79]:

```
(3626.519, 102122.94999999997)
```

In [80]:

```
dfM_[dfM_.Income > max_thresold]
```

Out[80]:

	Education	Marital_Status	Income	MntWines	MntFruits	MntMeatProducts	MntFishProducts	MntSweetProducts
164	PhD	Married	157243.0	20	2	1582	1	2
203	PhD	Together	102160.0	763	29	138	76	176
252	Graduation	Divorced	102692.0	168	148	444	32	172
617	PhD	Together	162397.0	85	1	16	2	1
646	Graduation	Together	105471.0	1009	181	104	202	21
655	Graduation	Divorced	153924.0	1	1	1	1	1
687	PhD	Married	160803.0	55	16	1622	17	3
1300	Master	Together	157733.0	39	1	9	2	0
1653	Graduation	Together	157146.0	1	0	1725	2	1
1898	PhD	Single	113734.0	6	2	3	1	262
2132	PhD	Married	156924.0	2	1	2	1	1
2233	Graduation	Together	666666.0	9	14	18	8	1

In [81]:

```
dfM_[dfM_.Income < min_thresold]
```

Out[81]:

	Education	Marital_Status	Income	MntWines	MntFruits	MntMeatProducts	MntFishProducts	MntSweetProducts
21	Graduation	Married	2447.0	1	1	1725	1	1
1245	Graduation	Divorced	1730.0	1	1	3	1	1
1524	Graduation	Single	3502.0	2	1	1	0	0

In [82]:

```
#create a new data frame with no outlier.
df2=dfM_[(dfM_.Income < max_thresold)&(dfM_.Income >min_thresold)]
df2
```

Out[82]:

	Education	Marital_Status	Income	MntWines	MntFruits	MntMeatProducts	MntFishProducts	MntSweetProducts
0	Graduation	Single	58138.0	635	88	546	172	88
1	Graduation	Single	46344.0	11	1	6	2	1
2	Graduation	Together	71613.0	426	49	127	111	21
3	Graduation	Together	26646.0	11	4	20	10	3
4	PhD	Married	58293.0	173	43	118	46	27
...	...	...	...	...	...	...	...	...
2235	Graduation	Married	61223.0	709	43	182	42	118
2236	PhD	Together	64014.0	406	0	30	0	0
2237	Graduation	Divorced	56981.0	908	48	217	32	12
2238	Master	Together	69245.0	428	30	214	80	30
2239	PhD	Married	52869.0	84	3	61	2	1

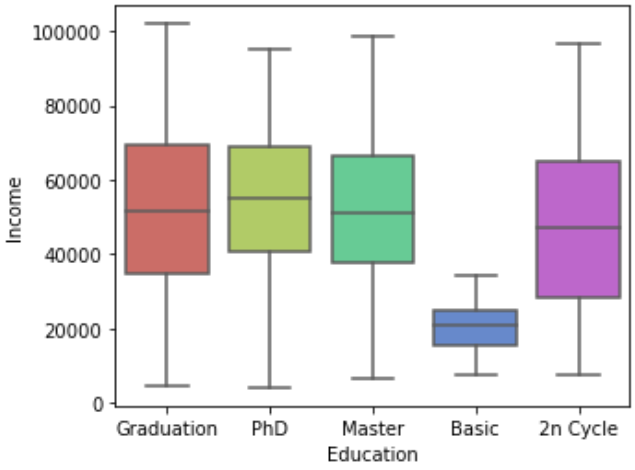
2225 rows × 8 columns

In [83]:

```
#Now we can see here ther is no outlier
sb.boxplot(y='Income',x='Education',data=df2,palette='hls')
```

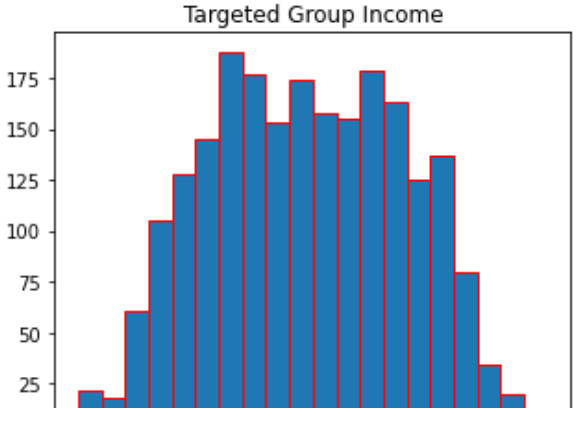
Out[83]:

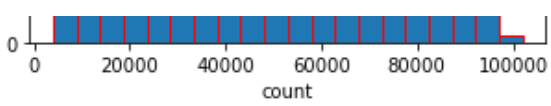
```
<AxesSubplot:xlabel='Education', ylabel='Income'>
```



In [92]:

```
#check normality by histogram
plt.hist(df2['Income'],"auto",edgecolor='red')
plt.xlabel('Income')
plt.xlabel('count')
plt.title("Targeted Group Income")
plt.show()
```





df2.describe() as we can see here most of the people income is between 20k to 100k which isn't enough.

In [93]:

```
from scipy.stats import shapiro
```

In [99]:

```
stat, p = shapiro(df2['Income'])
print('Statistics=%.3f, p=%.3f' % (stat, p))
alpha = 0.05
if p > alpha:
    print('Sample looks Gaussian (fail to reject H0)')
else:
    print('Sample does not look Gaussian (reject H0)')
```

```
Statistics=0.983, p=0.000
Sample does not look Gaussian (reject H0)
```

In [ ]:

```

```