In [1]:

```
import numpy as np
import pandas as pd
import warnings
warnings.filterwarnings('ignore')
import os
os.getcwd
import statistics as st
```

In [2]:
```
stats = pd.read_csv(r'income (descriptive stats).csv')
stats
```

Out[2]:

	Mthly_HH_Income	Mthly_HH_Expense	No_of_Fly_Members	Emi_or_Rent_Amt	Annual_HH_Income	Highest_Qualified
0	5000	8000	3	2000	64200	Under
1	6000	7000	2	3000	79920	
2	10000	4500	2	0	112800	Under
3	10000	2000	1	0	97200	
4	12500	12000	2	3000	147000	
5	14000	8000	2	0	196560	
6	15000	16000	3	35000	167400	Post
7	18000	20000	5	8000	216000	
8	19000	9000	2	0	218880	Under
9	20000	9000	4	0	220800	Under
10	20000	18000	4	8000	278400	Under
11	22000	25000	6	12000	279840	
12	23400	5000	3	0	292032	
13	24000	10500	6	0	316800	
14	24000	10000	4	0	244800	
15	25000	12300	3	0	246000	
16	25000	20000	3	3500	261000	
17	25000	10000	6	0	258000	Under
18	29000	6600	2	2000	348000	
19	30000	13000	4	0	385200	
20	30500	25000	5	5000	351360	Under
21	32000	15000	4	0	445440	Pr
22	34000	19000	6	0	330480	Pr
23	34000	25000	3	4000	469200	Pr
24	35000	12000	3	0	466200	
25	35000	25000	4	0	449400	Pr
26	39000	8000	4	0	556920	Under
27	40000	10000	4	0	412800	Under
28	42000	15000	4	0	488880	
29	43000	12000	4	0	619200	
30	45000	25000	6	0	523800	
31	45000	40000	6	3500	507600	Pr
32	45000	10000	2	1000	437400	Post
33	45000	22000	4	2500	610200	Post
34	46000	25000	5	3500	596160	
35	47000	15000	7	0	456840	Pr
36	50000	20000	4	0	570000	Pr
37	50500	20000	3	0	581760	Pr
38	55000	45000	6	12000	600600	
39	60000	10000	3	0	590400	Post
40	60000	50000	6	10000	590400	
41	65000	20000	4	5000	647400	

	Mthly_HH_Income	Mthly_HH_Expense	No_of_Fly_Members	Emi_or_Rent_Amt	Annual_HH_Income	Highest_Qualified
42	70000	9000	2	0	756000	
43	80000	20000	4	0	1075200	
44	85000	25000	5	0	1142400	Under
45	90000	48000	7	0	885600	Post
46	98000	25000	5	0	1152480	Pr
47	100000	30000	6	0	1404000	
48	100000	50000	4	20000	1032000	Pr
49	100000	40000	6	10000	1320000	Post

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50 entries, 0 to 49
Data columns (total 7 columns):
 #   Column                  Non-Null Count  Dtype 
---  ------                  --------------  ----- 
 0   Mthly_HH_Income         50 non-null     int64 
 1   Mthly_HH_Expense        50 non-null     int64 
 2   No_of_Fly_Members       50 non-null     int64 
 3   Emi_or_Rent_Amt         50 non-null     int64 
 4   Annual_HH_Income        50 non-null     int64 
 5   Highest_Qualified_Member  50 non-null     object
 6   No_of_Earning_Members   50 non-null     int64 
dtypes: int64(6), object(1)
memory usage: 2.9+ KB
```

In [4]:
```
stats.shape
```

Out[4]:
```
(50, 7)
```

In [5]:
```
stats.columns
```

Out[5]:
```
Index(['Mthly_HH_Income', 'Mthly_HH_Expense', 'No_of_Fly_Members',
       'Emi_or_Rent_Amt', 'Annual_HH_Income', 'Highest_Qualified_Member',
       'No_of_Earning_Members'],
      dtype='object')
```

In [6]:
```
stats.columns = ['monthly.income','monthly.expenses','family.count','emi.amount',\
                                    'annual.income','high.qualification','members.earnings']
```

In [7]:

```
stats
```

Out[7]:

	monthly.income	monthly.expenses	family.count	emi.amount	annual.income	high.qualification	members.earnings
0	5000	8000	3	2000	64200	Under-Graduate	1
1	6000	7000	2	3000	79920	Illiterate	1
2	10000	4500	2	0	112800	Under-Graduate	1
3	10000	2000	1	0	97200	Illiterate	1
4	12500	12000	2	3000	147000	Graduate	1
5	14000	8000	2	0	196560	Graduate	1
6	15000	16000	3	35000	167400	Post-Graduate	1
7	18000	20000	5	8000	216000	Graduate	1
8	19000	9000	2	0	218880	Under-Graduate	1
9	20000	9000	4	0	220800	Under-Graduate	2
10	20000	18000	4	8000	278400	Under-Graduate	2
11	22000	25000	6	12000	279840	Illiterate	1
12	23400	5000	3	0	292032	Illiterate	1
13	24000	10500	6	0	316800	Graduate	2
14	24000	10000	4	0	244800	Graduate	2
15	25000	12300	3	0	246000	Graduate	1
16	25000	20000	3	3500	261000	Graduate	1
17	25000	10000	6	0	258000	Under-Graduate	3
18	29000	6600	2	2000	348000	Graduate	1
19	30000	13000	4	0	385200	Graduate	1
20	30500	25000	5	5000	351360	Under-Graduate	1
21	32000	15000	4	0	445440	Professional	1
22	34000	19000	6	0	330480	Professional	1
23	34000	25000	3	4000	469200	Professional	1
24	35000	12000	3	0	466200	Graduate	1
25	35000	25000	4	0	449400	Professional	2
26	39000	8000	4	0	556920	Under-Graduate	1
27	40000	10000	4	0	412800	Under-Graduate	1
28	42000	15000	4	0	488880	Graduate	1
29	43000	12000	4	0	619200	Graduate	1
30	45000	25000	6	0	523800	Graduate	3
31	45000	40000	6	3500	507600	Professional	2
32	45000	10000	2	1000	437400	Post-Graduate	1
33	45000	22000	4	2500	610200	Post-Graduate	1
34	46000	25000	5	3500	596160	Graduate	1
35	47000	15000	7	0	456840	Professional	4
36	50000	20000	4	0	570000	Professional	1
37	50500	20000	3	0	581760	Professional	2
38	55000	45000	6	12000	600600	Graduate	2
39	60000	10000	3	0	590400	Post-Graduate	1
40	60000	50000	6	10000	590400	Graduate	1
41	65000	20000	4	5000	647400	Illiterate	2

	monthly.income	monthly.expenses	family.count	emi.amount	annual.income	high.qualification	members.earnings
42	70000	9000	2	0	756000	Graduate	1
43	80000	20000	4	0	1075200	Graduate	1
44	85000	25000	5	0	1142400	Under-Graduate	2
45	90000	48000	7	0	885600	Post-Graduate	3
46	98000	25000	5	0	1152480	Professional	2
47	100000	30000	6	0	1404000	Graduate	3
48	100000	50000	4	20000	1032000	Professional	2
49	100000	40000	6	10000	1320000	Post-Graduate	1

In [8]:

```
stats.columns
```

Out[8]:

```
Index(['monthly.income', 'monthly.expenses', 'family.count', 'emi.amount',
       'annual.income', 'high.qualification', 'members.earnings'],
      dtype='object')
```

In [9]:

```
stats.head()
```

Out[9]:

	monthly.income	monthly.expenses	family.count	emi.amount	annual.income	high.qualification	members.earnings
0	5000	8000	3	2000	64200	Under-Graduate	1
1	6000	7000	2	3000	79920	Illiterate	1
2	10000	4500	2	0	112800	Under-Graduate	1
3	10000	2000	1	0	97200	Illiterate	1
4	12500	12000	2	3000	147000	Graduate	1

In [10]:

```
stats.tail()
```

Out[10]:

	monthly.income	monthly.expenses	family.count	emi.amount	annual.income	high.qualification	members.earnings
45	90000	48000	7	0	885600	Post-Graduate	3
46	98000	25000	5	0	1152480	Professional	2
47	100000	30000	6	0	1404000	Graduate	3
48	100000	50000	4	20000	1032000	Professional	2
49	100000	40000	6	10000	1320000	Post-Graduate	1

In [11]:
```
stats[0:49:5]
```

Out[11]:

	monthly.income	monthly.expenses	family.count	emi.amount	annual.income	high.qualification	members.earnings
0	5000	8000	3	2000	64200	Under-Graduate	1
5	14000	8000	2	0	196560	Graduate	1
10	20000	18000	4	8000	278400	Under-Graduate	2
15	25000	12300	3	0	246000	Graduate	1
20	30500	25000	5	5000	351360	Under-Graduate	1
25	35000	25000	4	0	449400	Professional	2
30	45000	25000	6	0	523800	Graduate	3
35	47000	15000	7	0	456840	Professional	4
40	60000	50000	6	10000	590400	Graduate	1
45	90000	48000	7	0	885600	Post-Graduate	3

In [12]:
```
stats.describe()
```

Out[12]:

	monthly.income	monthly.expenses	family.count	emi.amount	annual.income	members.earnings
count	50.000000	50.000000	50.000000	50.000000	5.000000e+01	50.000000
mean	41558.000000	18818.000000	4.060000	3060.000000	4.900190e+05	1.460000
std	26097.908979	12090.216824	1.517382	6241.434948	3.201358e+05	0.734291
min	5000.000000	2000.000000	1.000000	0.000000	6.420000e+04	1.000000
25%	23550.000000	10000.000000	3.000000	0.000000	2.587500e+05	1.000000
50%	35000.000000	15500.000000	4.000000	0.000000	4.474200e+05	1.000000
75%	50375.000000	25000.000000	5.000000	3500.000000	5.947200e+05	2.000000
max	100000.000000	50000.000000	7.000000	35000.000000	1.404000e+06	4.000000

In [13]:
```
stats.transpose().describe()
```

Out[13]:

	0	1	2	3	4	5	6	7	8	9	...	40	41	42	43	44	4
count	7	7	7	7	7	7	7	7	7	7	...	7	7	7	7	7	
unique	7	7	7	6	7	7	7	7	7	7	...	7	7	7	7	7	
top	5000	6000	10000	1	12500	14000	15000	18000	19000	20000	...	60000	65000	70000	80000	85000	900
freq	1	1	1	2	1	1	1	1	1	1	...	1	1	1	1	1	

4 rows × 50 columns

In [14]:
```
stats.isnull().any()
```

Out[14]:
```
monthly.income         False
monthly.expenses       False
family.count           False
emi.amount             False
annual.income          False
high.qualification     False
members.earnings       False
dtype: bool
```

```
In [15]: stats.isnull().sum()
```

```
Out[15]: monthly.income        0
         monthly.expenses      0
         family.count          0
         emi.amount            0
         annual.income         0
         high.qualification    0
         members.earnings      0
         dtype: int64
```

Mean

x1+x2+x3------+xN/N

In [16]:

```
stats
```

Out[16]:

	monthly.income	monthly.expenses	family.count	emi.amount	annual.income	high.qualification	members.earnings
0	5000	8000	3	2000	64200	Under-Graduate	1
1	6000	7000	2	3000	79920	Illiterate	1
2	10000	4500	2	0	112800	Under-Graduate	1
3	10000	2000	1	0	97200	Illiterate	1
4	12500	12000	2	3000	147000	Graduate	1
5	14000	8000	2	0	196560	Graduate	1
6	15000	16000	3	35000	167400	Post-Graduate	1
7	18000	20000	5	8000	216000	Graduate	1
8	19000	9000	2	0	218880	Under-Graduate	1
9	20000	9000	4	0	220800	Under-Graduate	2
10	20000	18000	4	8000	278400	Under-Graduate	2
11	22000	25000	6	12000	279840	Illiterate	1
12	23400	5000	3	0	292032	Illiterate	1
13	24000	10500	6	0	316800	Graduate	2
14	24000	10000	4	0	244800	Graduate	2
15	25000	12300	3	0	246000	Graduate	1
16	25000	20000	3	3500	261000	Graduate	1
17	25000	10000	6	0	258000	Under-Graduate	3
18	29000	6600	2	2000	348000	Graduate	1
19	30000	13000	4	0	385200	Graduate	1
20	30500	25000	5	5000	351360	Under-Graduate	1
21	32000	15000	4	0	445440	Professional	1
22	34000	19000	6	0	330480	Professional	1
23	34000	25000	3	4000	469200	Professional	1
24	35000	12000	3	0	466200	Graduate	1
25	35000	25000	4	0	449400	Professional	2
26	39000	8000	4	0	556920	Under-Graduate	1
27	40000	10000	4	0	412800	Under-Graduate	1
28	42000	15000	4	0	488880	Graduate	1
29	43000	12000	4	0	619200	Graduate	1
30	45000	25000	6	0	523800	Graduate	3
31	45000	40000	6	3500	507600	Professional	2
32	45000	10000	2	1000	437400	Post-Graduate	1
33	45000	22000	4	2500	610200	Post-Graduate	1
34	46000	25000	5	3500	596160	Graduate	1
35	47000	15000	7	0	456840	Professional	4
36	50000	20000	4	0	570000	Professional	1
37	50500	20000	3	0	581760	Professional	2
38	55000	45000	6	12000	600600	Graduate	2
39	60000	10000	3	0	590400	Post-Graduate	1
40	60000	50000	6	10000	590400	Graduate	1
41	65000	20000	4	5000	647400	Illiterate	2

	monthly.income	monthly.expenses	family.count	emi.amount	annual.income	high.qualification	members.earnings
42	70000	9000	2	0	756000	Graduate	1
43	80000	20000	4	0	1075200	Graduate	1
44	85000	25000	5	0	1142400	Under-Graduate	2
45	90000	48000	7	0	885600	Post-Graduate	3
46	98000	25000	5	0	1152480	Professional	2
47	100000	30000	6	0	1404000	Graduate	3
48	100000	50000	4	20000	1032000	Professional	2
49	100000	40000	6	10000	1320000	Post-Graduate	1

```
In [17]: stats['monthly.income'].mean()

Out[17]: 41558.0
```

```
In [18]: stats['monthly.expenses'].mean()

Out[18]: 18818.0
```

```
In [19]: stats['emi.amount'].mean()

Out[19]: 3060.0
```

```
In [20]: stats['annual.income'].mean()

Out[20]: 490019.04
```

```
In [21]: stats['members.earnings'].mean()

Out[21]: 1.46
```

```
In [22]: stats['family.count'].mean()

Out[22]: 4.06
```

Median

n+1/n or middle number (1,2,3,4,5 = 3 is median)

```
In [23]: stats['monthly.income'].median()

Out[23]: 35000.0
```

```
In [24]: stats['monthly.expenses'].median()

Out[24]: 15500.0
```

```
In [25]: stats['emi.amount'].median()

Out[25]: 0.0
```

```
In [26]: stats['annual.income'].median()

Out[26]: 447420.0
```

In [27]:
```
stats['members.earnings'].median()
```

Out[27]:
```
1.0
```

In [28]:
```
stats['family.count'].median()
```

Out[28]:
```
4.0
```

Monthly expenses

In [29]:
```
monthlyexpenses = pd.crosstab(index=stats['monthly.expenses'],columns='count')
#pd is an alias for the pandas library, assumed to be imported at the beginning of your code.
#crosstab() is a pandas function used to compute a cross-tabulation table.
#index=stats['monthly.expenses'] specifies the column in the 'stats' DataFrame that willbe used
#It selects the 'monthly.expenses' column from the 'stats' DataFrame.
#columns='count' specifies that the count of occurrences will be displayed as a single column in
```

In [30]:
```
monthlyexpenses
```

Out[30]:

col_0	count
monthly.expenses	
2000	1
4500	1
5000	1
6600	1
7000	1
8000	3
9000	3
10000	5
10500	1
12000	3
12300	1
13000	1
15000	3
16000	1
18000	1
19000	1
20000	6
22000	1
25000	8
30000	1
40000	2
45000	1
48000	1
50000	2

In [31]:
```
monthlyexpenses.reset_index(inplace=True)
```

In [32]:
```
monthlyexpenses
```

Out[32]:

col_0	monthly.expenses	count
0	2000	1
1	4500	1
2	5000	1
3	6600	1
4	7000	1
5	8000	3
6	9000	3
7	10000	5
8	10500	1
9	12000	3
10	12300	1
11	13000	1
12	15000	3
13	16000	1
14	18000	1
15	19000	1
16	20000	6
17	22000	1
18	25000	8
19	30000	1
20	40000	2
21	45000	1
22	48000	1
23	50000	2

In [33]:
```
# highest expenses in a month by most families
monthlyexpenses[monthlyexpenses['count'] == stats['monthly.expenses'].value_counts().max()]
```

Out[33]:

col_0	monthly.expenses	count
18	25000	8

In [34]:
```
monthlyexpenses = pd.crosstab(index=stats['monthly.expenses'],columns='count')
monthlyexpenses.reset_index(inplace=True)
monthlyexpenses[monthlyexpenses['count'] == stats['monthly.expenses'].value_counts().max()]
```

Out[34]:

col_0	monthly.expenses	count
18	25000	8

In [35]:

```
monthlyexpenses = pd.crosstab(index=stats['monthly.expenses'],columns='count')
monthlyexpenses.reset_index(inplace=True)
monthlyexpenses[monthlyexpenses['count'] == stats['monthly.expenses'].value_counts().min()]
```

Out[35]:

col_0	monthly.expenses	count
0	2000	1
1	4500	1
2	5000	1
3	6600	1
4	7000	1
8	10500	1
10	12300	1
11	13000	1
13	16000	1
14	18000	1
15	19000	1
17	22000	1
19	30000	1
21	45000	1
22	48000	1

In [36]:

```
monthlyexpenses = pd.crosstab(index=stats['monthly.expenses'],columns='count')
monthlyexpenses.reset_index(inplace=True)
monthlyexpenses[monthlyexpenses['count'] == stats['monthly.expenses'].value_counts().mean()]
```

Out[36]:

col_0	monthly.expenses	count

In [37]:
```
monthlyexpenses = pd.crosstab(index=stats['monthly.expenses'],columns='count')
monthlyexpenses.reset_index(inplace=True)
monthlyexpenses[monthlyexpenses['count'] == stats['monthly.expenses'].value_counts().median()]
```

Out[37]:

col_0	monthly.expenses	count
0	2000	1
1	4500	1
2	5000	1
3	6600	1
4	7000	1
8	10500	1
10	12300	1
11	13000	1
13	16000	1
14	18000	1
15	19000	1
17	22000	1
19	30000	1
21	45000	1
22	48000	1

high qualificatin

In [38]:
```
highqualification = pd.crosstab(index=stats['high.qualification'],columns='count')
highqualification.reset_index(inplace=True)
highqualification
```

Out[38]:

col_0	high.qualification	count
0	Graduate	19
1	Illiterate	5
2	Post-Graduate	6
3	Professional	10
4	Under-Graduate	10

In [39]:
```
highqualification = pd.crosstab(index=stats['high.qualification'],columns='count')
highqualification.reset_index(inplace=True)
highqualification[highqualification['count'] == stats['high.qualification'].value_counts().min()]
```

Out[39]:

col_0	high.qualification	count
1	Illiterate	5

In [40]:

```
highqualification = pd.crosstab(index=stats['high.qualification'],columns='count')
highqualification.reset_index(inplace=True)
highqualification[highqualification['count'] == stats['high.qualification'].value_counts().max()
```

Out[40]:

col_0	high.qualification	count
0	Graduate	19

In [41]:

```
highqualification = pd.crosstab(index=stats['high.qualification'],columns='count')
highqualification.reset_index(inplace=True)
highqualification[highqualification['count'] == stats['high.qualification'].value_counts().mean(
```

Out[41]:

col_0	high.qualification	count
3	Professional	10
4	Under-Graduate	10

In [42]:

```
highqualification = pd.crosstab(index=stats['high.qualification'],columns='count')
highqualification.reset_index(inplace=True)
highqualification[highqualification['count'] == stats['high.qualification'].value_counts().median
```

Out[42]:

col_0	high.qualification	count
3	Professional	10
4	Under-Graduate	10

In [43]:

```
familycount = pd.crosstab(index=stats['family.count'],columns='count')
familycount.reset_index(inplace=True)
familycount
```

Out[43]:

col_0	family.count	count
0	1	1
1	2	8
2	3	9
3	4	15
4	5	5
5	6	10
6	7	2

Visualization

In [44]:

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

In [45]:

```
stats
```

Out[45]:

	monthly.income	monthly.expenses	family.count	emi.amount	annual.income	high.qualification	members.earnings
0	5000	8000	3	2000	64200	Under-Graduate	1
1	6000	7000	2	3000	79920	Illiterate	1
2	10000	4500	2	0	112800	Under-Graduate	1
3	10000	2000	1	0	97200	Illiterate	1
4	12500	12000	2	3000	147000	Graduate	1
5	14000	8000	2	0	196560	Graduate	1
6	15000	16000	3	35000	167400	Post-Graduate	1
7	18000	20000	5	8000	216000	Graduate	1
8	19000	9000	2	0	218880	Under-Graduate	1
9	20000	9000	4	0	220800	Under-Graduate	2
10	20000	18000	4	8000	278400	Under-Graduate	2
11	22000	25000	6	12000	279840	Illiterate	1
12	23400	5000	3	0	292032	Illiterate	1
13	24000	10500	6	0	316800	Graduate	2
14	24000	10000	4	0	244800	Graduate	2
15	25000	12300	3	0	246000	Graduate	1
16	25000	20000	3	3500	261000	Graduate	1
17	25000	10000	6	0	258000	Under-Graduate	3
18	29000	6600	2	2000	348000	Graduate	1
19	30000	13000	4	0	385200	Graduate	1
20	30500	25000	5	5000	351360	Under-Graduate	1
21	32000	15000	4	0	445440	Professional	1
22	34000	19000	6	0	330480	Professional	1
23	34000	25000	3	4000	469200	Professional	1
24	35000	12000	3	0	466200	Graduate	1
25	35000	25000	4	0	449400	Professional	2
26	39000	8000	4	0	556920	Under-Graduate	1
27	40000	10000	4	0	412800	Under-Graduate	1
28	42000	15000	4	0	488880	Graduate	1
29	43000	12000	4	0	619200	Graduate	1
30	45000	25000	6	0	523800	Graduate	3
31	45000	40000	6	3500	507600	Professional	2
32	45000	10000	2	1000	437400	Post-Graduate	1
33	45000	22000	4	2500	610200	Post-Graduate	1
34	46000	25000	5	3500	596160	Graduate	1
35	47000	15000	7	0	456840	Professional	4
36	50000	20000	4	0	570000	Professional	1
37	50500	20000	3	0	581760	Professional	2
38	55000	45000	6	12000	600600	Graduate	2
39	60000	10000	3	0	590400	Post-Graduate	1
40	60000	50000	6	10000	590400	Graduate	1
41	65000	20000	4	5000	647400	Illiterate	2

	monthly.income	monthly.expenses	family.count	emi.amount	annual.income	high.qualification	members.earnings
42	70000	9000	2	0	756000	Graduate	1
43	80000	20000	4	0	1075200	Graduate	1
44	85000	25000	5	0	1142400	Under-Graduate	2
45	90000	48000	7	0	885600	Post-Graduate	3
46	98000	25000	5	0	1152480	Professional	2
47	100000	30000	6	0	1404000	Graduate	3
48	100000	50000	4	20000	1032000	Professional	2
49	100000	40000	6	10000	1320000	Post-Graduate	1

In [46]:

```
highqualification = pd.crosstab(index=stats['high.qualification'],columns='count')
highqualification.reset_index(inplace=True)
highqualification
```

Out[46]:

col_0	high.qualification	count
0	Graduate	19
1	Illiterate	5
2	Post-Graduate	6
3	Professional	10
4	Under-Graduate	10

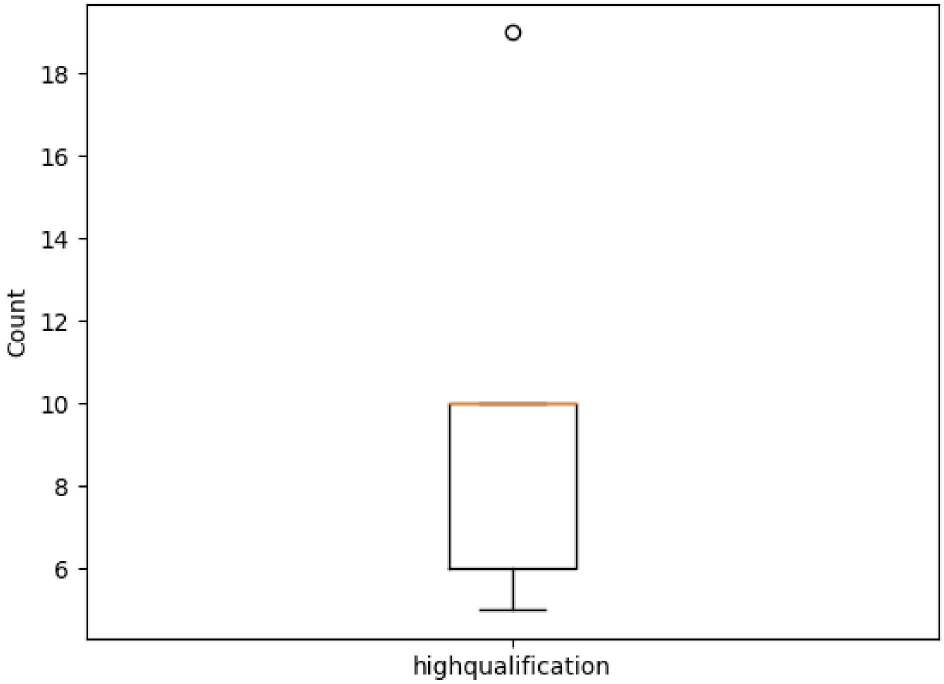
In [47]:

```
highqualification = pd.crosstab(index=stats['high.qualification'], columns='count')
highqualification.reset_index(inplace=True)

plt.boxplot(highqualification['count'])
plt.xticks([1], ['highqualification'])
plt.ylabel('Count')
```

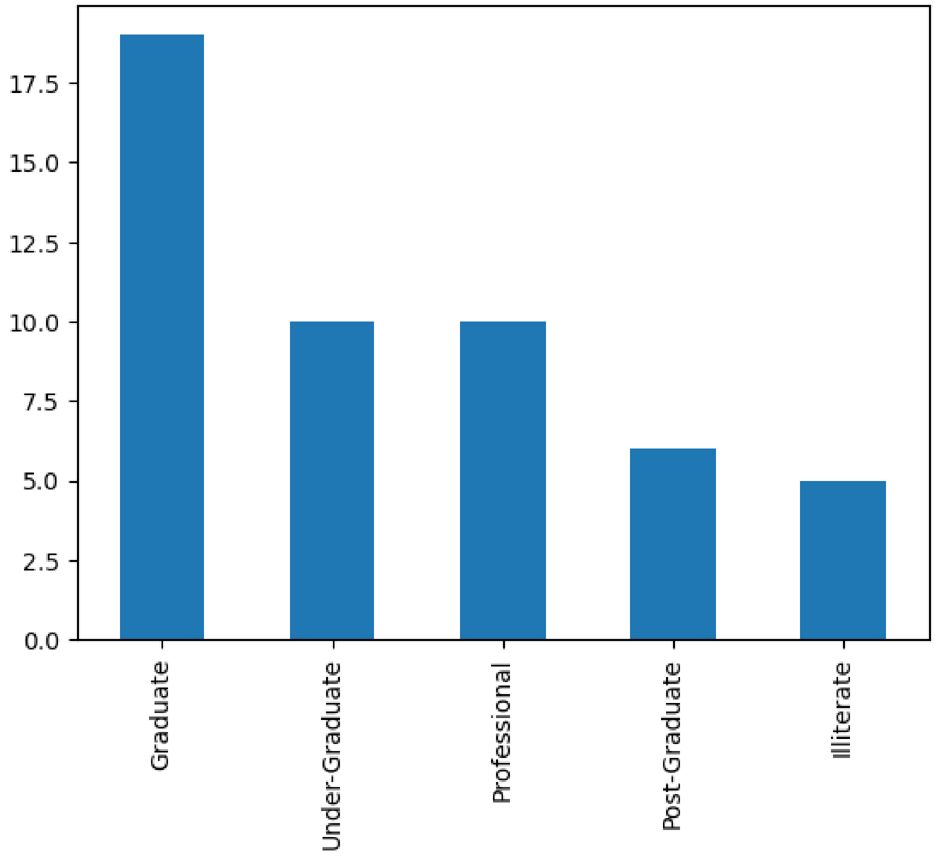
Out[47]:

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

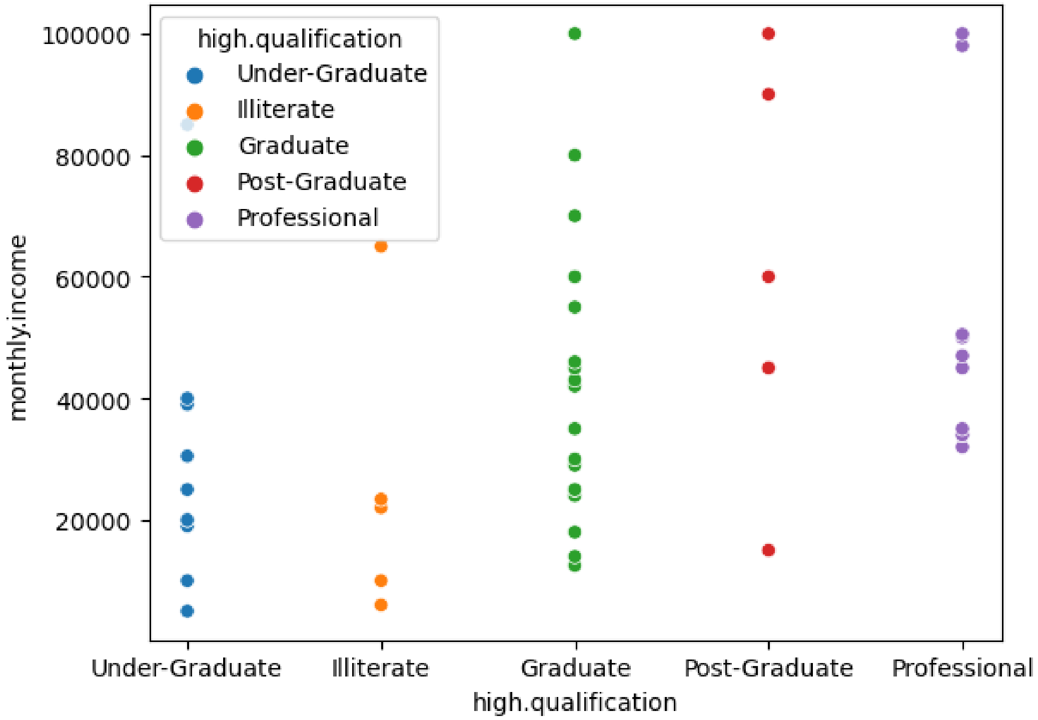


In [48]:
```
stats["high.qualification"].value_counts().plot(kind="bar")
```

Out[48]:
```
<Axes: >
```

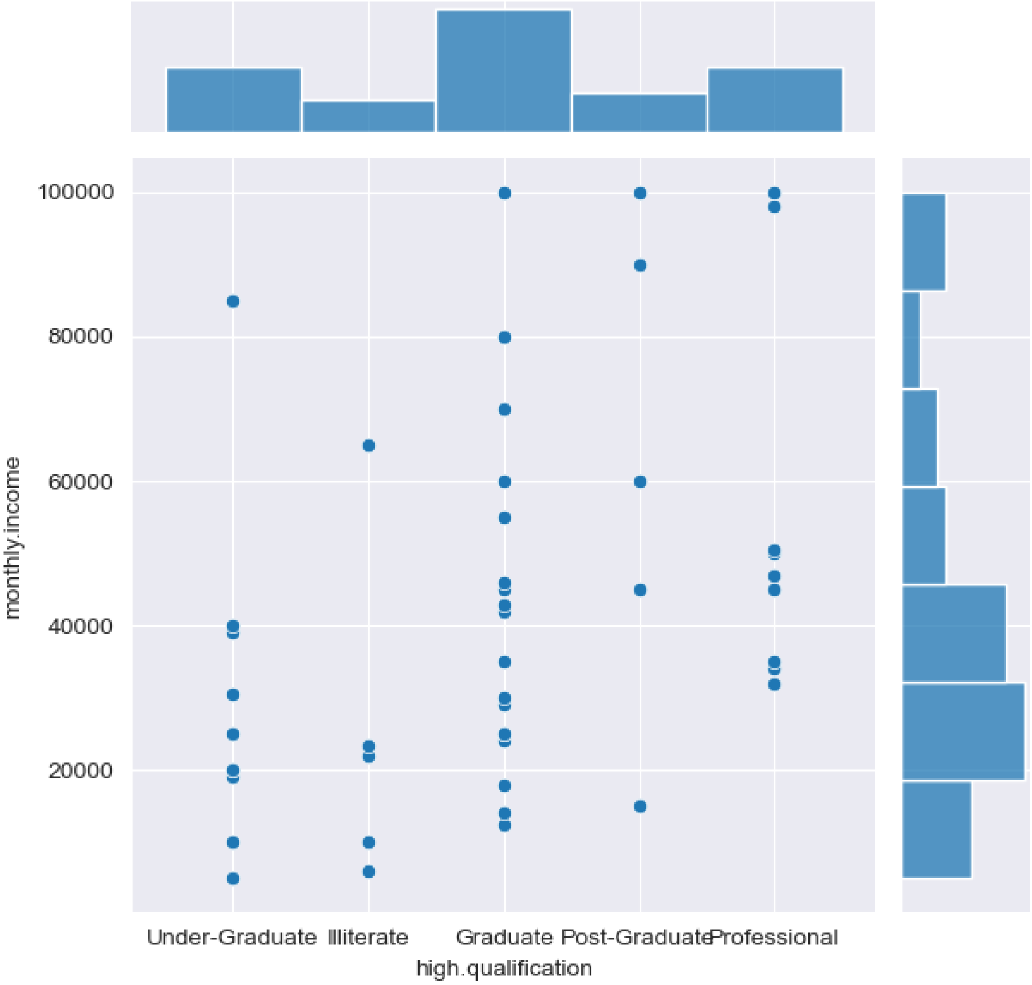


In [49]:
```
a=sns.scatterplot(data=stats,x='high.qualification',y='monthly.income', hue='high.qualification'
```



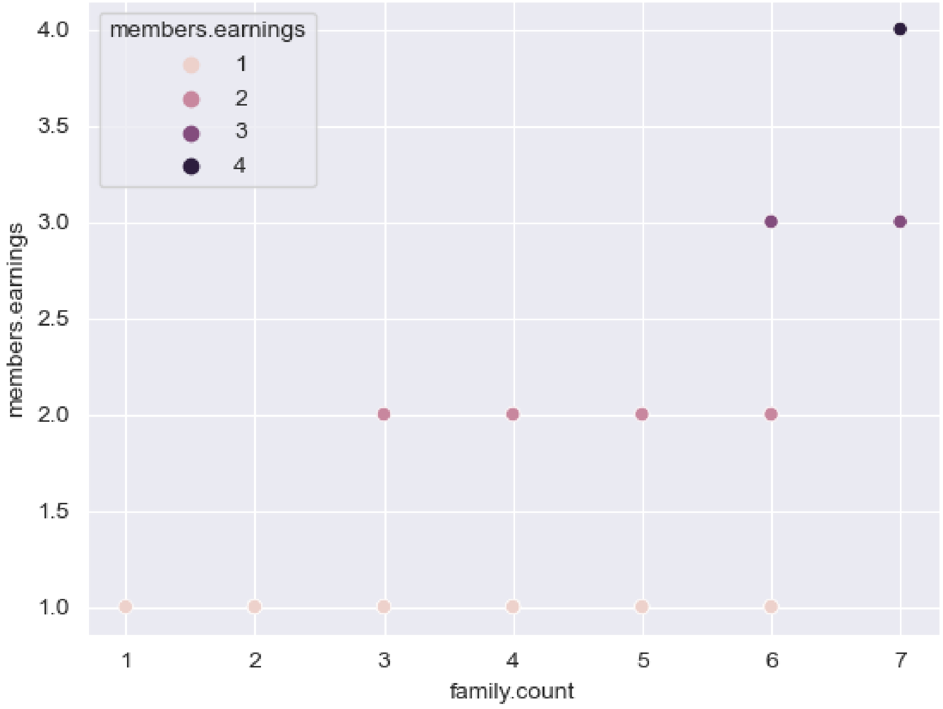
```
In [50]: sns.set_style('darkgrid')
```

```
In [51]: a=sns.jointplot(data=stats,x='high.qualification',y='monthly.income')
```



In [52]:

```
c = sns.scatterplot(data=stats, x='family.count', y='members.earnings', hue='members.earnings')
```



In [53]:

```
familycount = pd.crosstab(index=stats['family.count'],columns='count')
familycount.reset_index(inplace=True)
familycount
```

Out[53]:

col_0	family.count	count
0	1	1
1	2	8
2	3	9
3	4	15
4	5	5
5	6	10
6	7	2

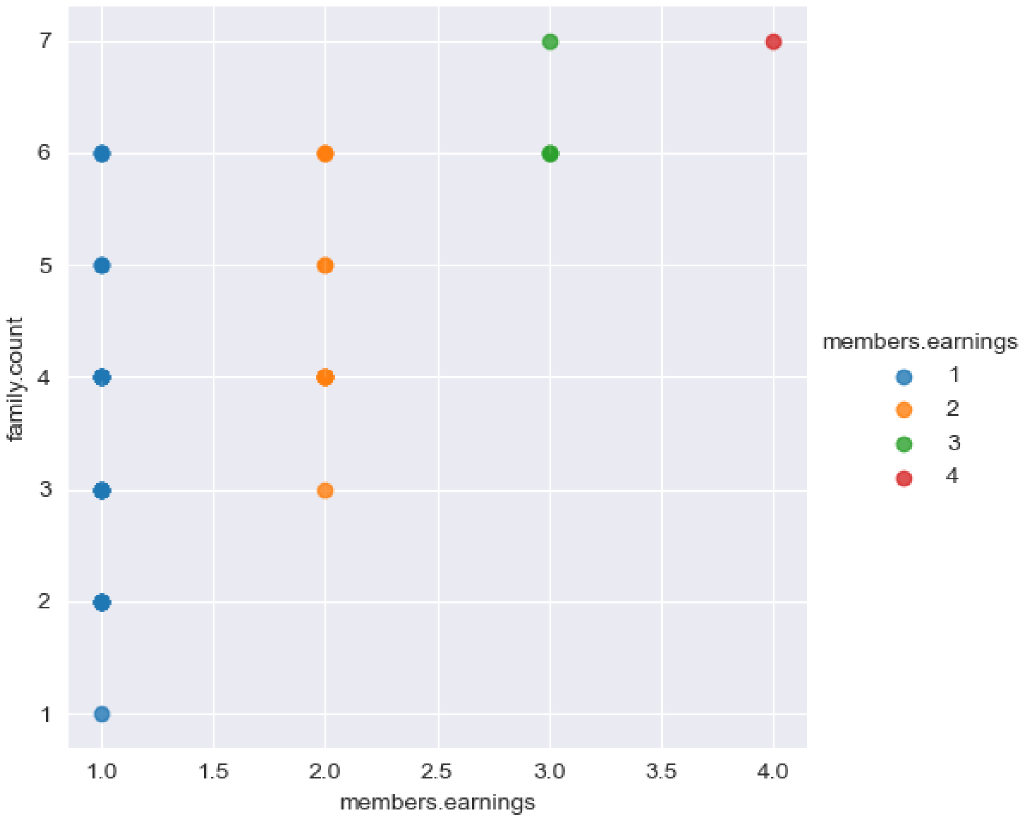
In [54]:

```
membersearnings = pd.crosstab(index=stats['members.earnings'],columns='count')
membersearnings.reset_index(inplace=True)
membersearnings
```

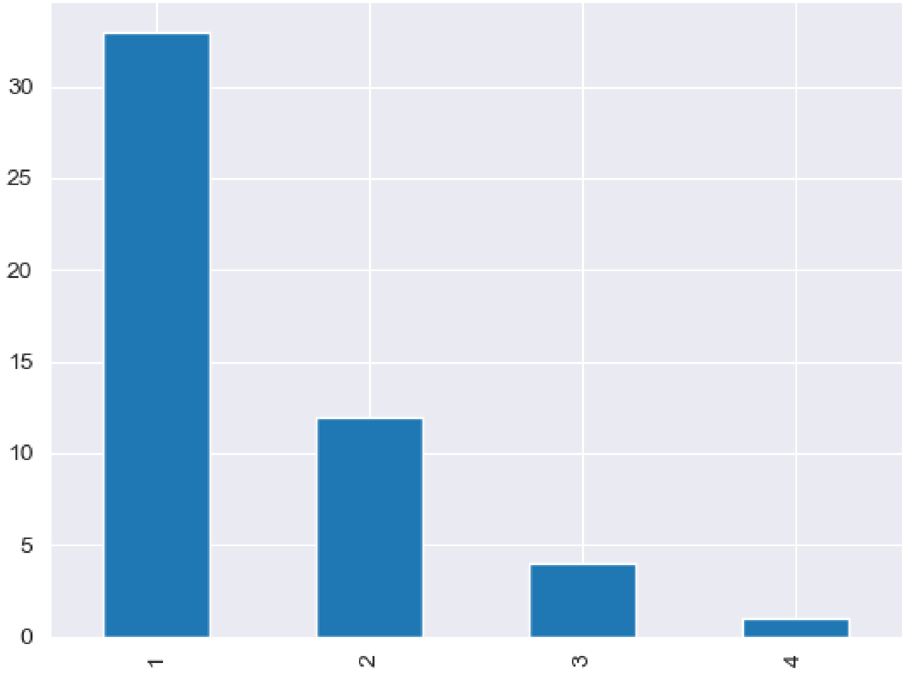
Out[54]:

col_0	members.earnings	count
0	1	33
1	2	12
2	3	4
3	4	1

In [55]:
```
c = sns.lmplot(data=stats, x='members.earnings', y='family.count', hue='members.earnings',aspect
```

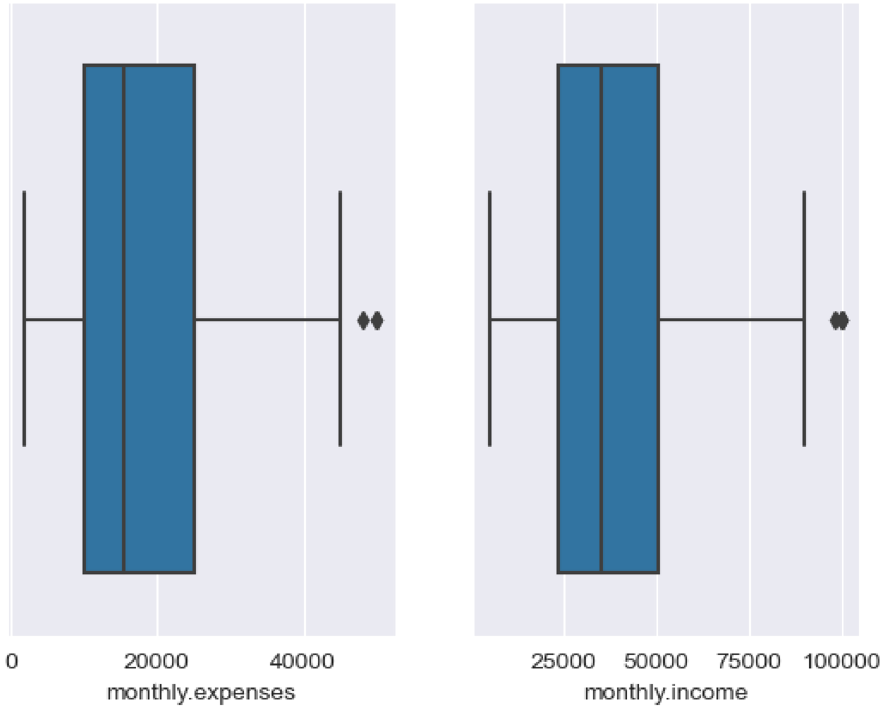


In [56]:
```
stats["members.earnings"].value_counts().plot(kind="bar").cmap='Reds_r'
```



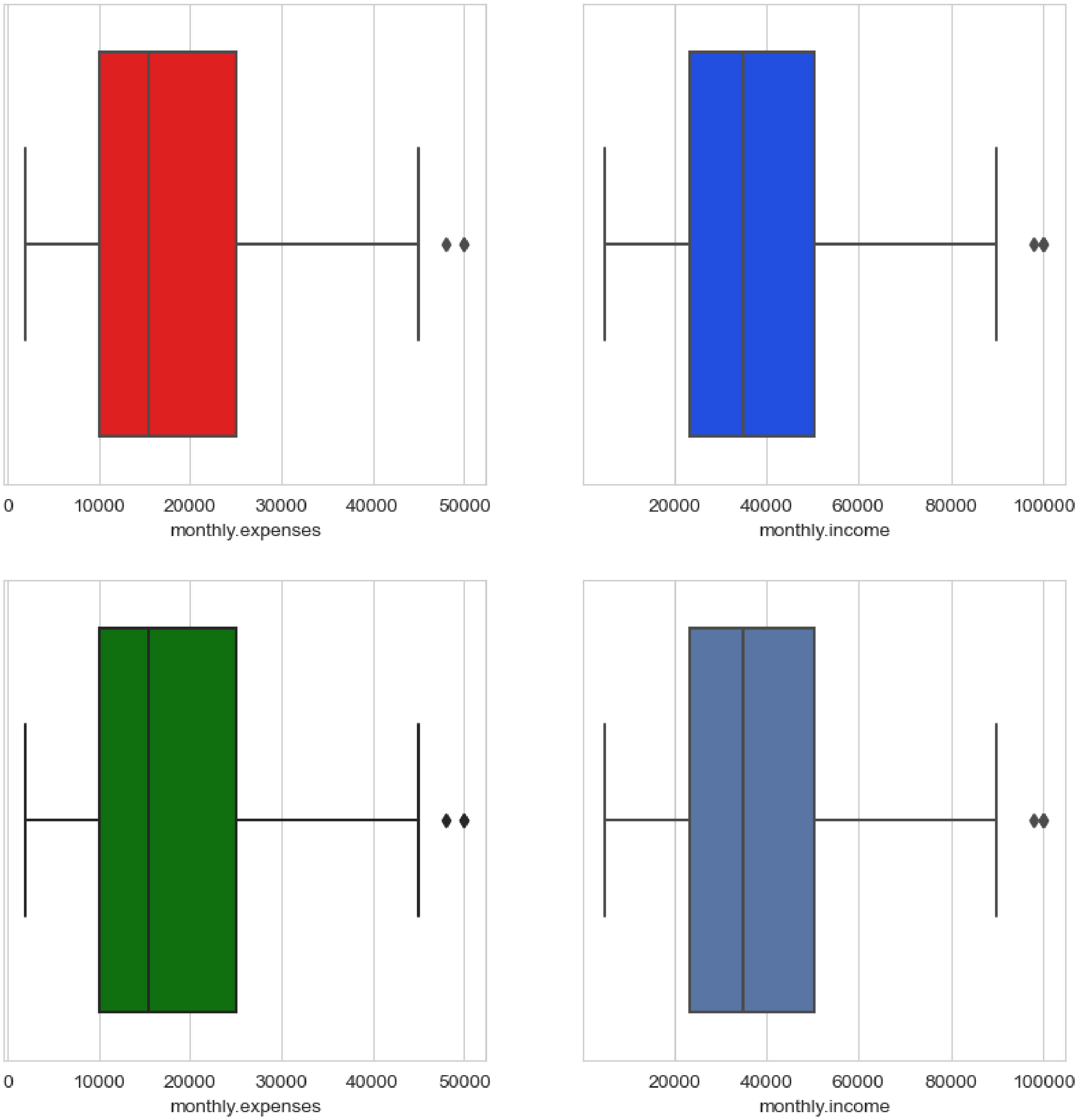
```
In [57]: fig , axes =plt.subplots(1,2)
         sns.boxplot(data=stats,x ='monthly.expenses', ax=axes[0] )
         sns.boxplot(data=stats,x='monthly.income', ax=axes[1] )

Out[57]: <Axes: xlabel='monthly.income'>
```

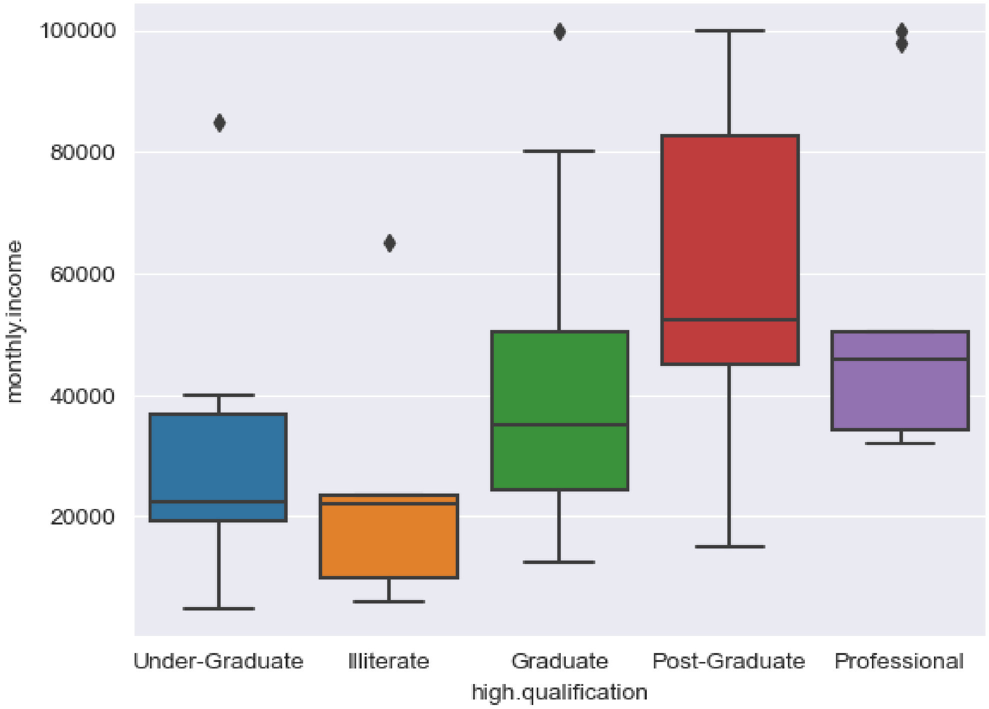


In [83]:
```
fig , axes =plt.subplots(2,2,figsize = (10,10))
sns.boxplot(data=stats,x ='monthly.expenses', ax=axes[0,0],color='r')
sns.boxplot(data=stats,x='monthly.income', ax=axes[0,1], palette = 'bright')
sns.boxplot(data=stats,x ='monthly.expenses', ax=axes[1,0], color = 'g')
sns.boxplot(data=stats,x='monthly.income', ax=axes[1,1],palette = 'deep')
```

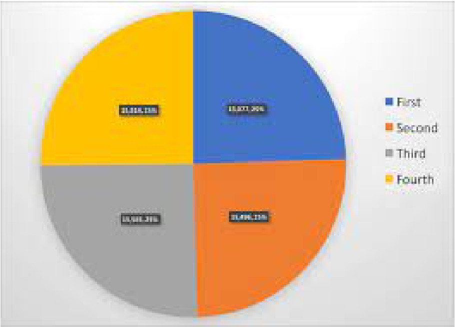
Out[83]:
```
<Axes: xlabel='monthly.income'>
```



```
In [59]: a = sns.boxplot(data=stats, x='high.qualification', y='monthly.income')
```



quartile = 25%,50%,75%,100%



```
-in pandas quartile will write as 0.25=25%, 0.50 = 50%, 0.75=75%
- in numpy 25=25,50=50,75=75
```

In [60]:

```
q1 = stats.quantile(0.25)
q2 = stats.quantile(0.50)  # Also the median
q3 = stats.quantile(0.75)
print("Q1:", q1)
print("Q2 (Median):", q2)
print("Q3:", q3)
```

```
Q1: monthly.income       23550.0
monthly.expenses     10000.0
family.count             3.0
emi.amount               0.0
annual.income       258750.0
members.earnings         1.0
Name: 0.25, dtype: float64
Q2 (Median): monthly.income       35000.0
monthly.expenses     15500.0
family.count             4.0
emi.amount               0.0
annual.income       447420.0
members.earnings         1.0
Name: 0.5, dtype: float64
Q3: monthly.income       50375.0
monthly.expenses     25000.0
family.count             5.0
emi.amount            3500.0
annual.income       594720.0
members.earnings         2.0
Name: 0.75, dtype: float64
```

In [61]:

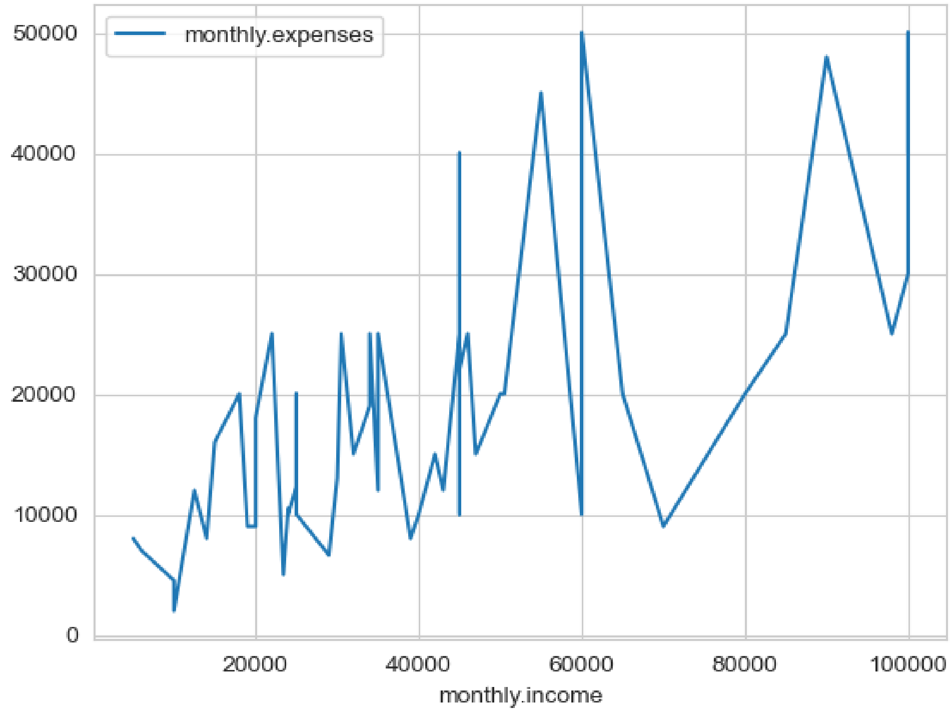
```
sns.set_style('whitegrid')
```

In [62]:

```
stats.plot(x="monthly.income", y="monthly.expenses")
quartile=stats["monthly.expenses"].quantile(0.75)-stats["monthly.expenses"].quantile(0.25)
quartile
```

Out[62]:

```
15000.0
```



Standard Deviation

In [63]:
```
pd.DataFrame(stats[:].std().to_frame())
```

Out[63]:

	0
monthly.income	26097.908979
monthly.expenses	12090.216824
family.count	1.517382
emi.amount	6241.434948
annual.income	320135.792123
members.earnings	0.734291

In [64]:
```
pd.DataFrame(stats[:10].std().to_frame()).transpose()
```

Out[64]:

	monthly.income	monthly.expenses	family.count	emi.amount	annual.income	members.earnings
0	5230.519413	5283.359621	1.173788	10805.862606	60547.609036	0.316228

In [65]:
```
pd.DataFrame(stats[11:30].std().to_frame()).transpose()
```

Out[65]:

	monthly.income	monthly.expenses	family.count	emi.amount	annual.income	members.earnings
0	6710.552918	6594.641631	1.196975	3021.124457	109874.150799	0.561951

In [66]:
```
pd.DataFrame(stats[46:49].std().to_frame()).transpose()
```

Out[66]:

	monthly.income	monthly.expenses	family.count	emi.amount	annual.income	members.earnings
0	1154.700538	13228.756555	1.0	11547.005384	189807.683722	0.57735

In [67]:
```
pd.DataFrame(stats.iloc[:,9:45].std().to_frame()).transpose()
```

Out[67]:

0

In [68]:
```
pd.DataFrame(stats.iloc[9:45].std().to_frame()).transpose()
```

Out[68]:

	monthly.income	monthly.expenses	family.count	emi.amount	annual.income	members.earnings
0	16581.161619	10289.67891	1.317465	3472.339681	207327.161112	0.73625

Variance

In [69]:
```
pd.DataFrame(stats[9:45].var().to_frame()).transpose()
```

Out[69]:

	monthly.income	monthly.expenses	family.count	emi.amount	annual.income	members.earnings
0	2.749349e+08	1.058775e+08	1.735714	1.205714e+07	4.298455e+10	0.542063

In [70]:

```
pd.DataFrame(stats[0:49].var().to_frame()).transpose()
```

Out[70]:

	monthly.income	monthly.expenses	family.count	emi.amount	annual.income	members.earnings
0	6.226827e+08	1.396804e+08	2.270408	3.874320e+07	8.997776e+10	0.545918

In [71]:

```
pd.DataFrame(stats[0:2].var().to_frame()).transpose()
```

Out[71]:

	monthly.income	monthly.expenses	family.count	emi.amount	annual.income	members.earnings
0	500000.0	500000.0	0.5	500000.0	123559200.0	0.0

```
suppose you have option to invest in Stock A or Stock B. The stocks have different expected
returns and standard deviations. The expected return of Stock A is 15% and Stock B is 10%.
Standard Deviation of the returns of these stocks is 10% and 5% respectively.
Which is better investment?
```

In [72]:

```
#stock A expected return = 15%
#stock B expected return = 10%
#standard deviation of returns of these stocks is 10% and 5%
coeff_of_var_stockA = 10/15
print('coeff_of_var_stockA',coeff_of_var_stockA)
coeff_of_var_stockB = 5/10
print('coeff_of_var_stockB',coeff_of_var_stockB)
```

```
coeff_of_var_stockA 0.6666666666666666
coeff_of_var_stockB 0.5
```

In [ ]: