In [1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
sns.set(style="whitegrid")
import matplotlib.pyplot as plt
from collections import Counter
%matplotlib inline
```

In [2]:
```
import warnings
import warnings
warnings.filterwarnings("ignore")
```

In [33]:
```
amir=pd.read_csv(r'E:\NIT(Data Science)by (prakash senapati sir (kodi))\Da
```

In [34]:
```
amir.head()
```

Out[34]:

	Mthly_HH_Income	Mthly_HH_Expense	No_of_Fly_Members	Emi_or_Rent_Amt	Annual_H
0	5000	8000	3	2000	
1	6000	7000	2	3000	
2	10000	4500	2	0	
3	10000	2000	1	0	
4	12500	12000	2	3000	

In [5]:
```
amir.tail()
```

Out[5]:

	Mthly_HH_Income	Mthly_HH_Expense	No_of_Fly_Members	Emi_or_Rent_Amt	Annual_I
45	90000	48000	7	0	
46	98000	25000	5	0	
47	100000	30000	6	0	
48	100000	50000	4	20000	
49	100000	40000	6	10000	

In [6]:
```
amir.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 [7]:
```
amir.describe().T
```

Out[7]:

	count	mean	std	min	25%	50%	
Mthly_HH_Income	50.0	41558.00	26097.908979	5000.0	23550.0	35000.0	5(
Mthly_HH_Expense	50.0	18818.00	12090.216824	2000.0	10000.0	15500.0	2:
No_of_Fly_Members	50.0	4.06	1.517382	1.0	3.0	4.0	
Emi_or_Rent_Amt	50.0	3060.00	6241.434948	0.0	0.0	0.0	:
Annual_HH_Income	50.0	490019.04	320135.792123	64200.0	258750.0	447420.0	59
No_of_Earning_Members	50.0	1.46	0.734291	1.0	1.0	1.0	

In [8]:
```
amir.shape
```

Out[8]:
```
(50, 7)
```

In [9]:
```
amir.isna().any()
```

Out[9]:
```
Mthly_HH_Income           False
Mthly_HH_Expense          False
No_of_Fly_Members         False
Emi_or_Rent_Amt           False
Annual_HH_Income          False
Highest_Qualified_Member  False
No_of_Earning_Members     False
dtype: bool
```

In [10]:
```
amir["Mthly_HH_Expense"].mean()
```

Out[10]:
```
18818.0
```

In [11]:
```
amir["Mthly_HH_Expense"].median()
```

Out[11]: 15500.0

In [12]:
```
monthly_expences = pd.crosstab(index=amir["Mthly_HH_Expense"], columns="co
monthly_expences.reset_index(inplace=True)
monthly_expences[monthly_expences["counts"]== amir.Mthly_HH_Expense.value_
```

Out[12]:

col_0	Mthly_HH_Expense	counts
18	25000	8

In [37]:
```
amir["Mthly_HH_Income"].quantile([0.25,0.75])
```

Out[37]:
```
0.25    23550.0
0.75    50375.0
Name: Mthly_HH_Income, dtype: float64
```

In [38]:
```
50375.0 - 23550.0
```

Out[38]: 26825.0

In [39]:
```
23550.0 -(1.5*26825.0)
```

Out[39]: -16687.5

In [40]:
```
 50375.0+(1.5*26825.0)
```

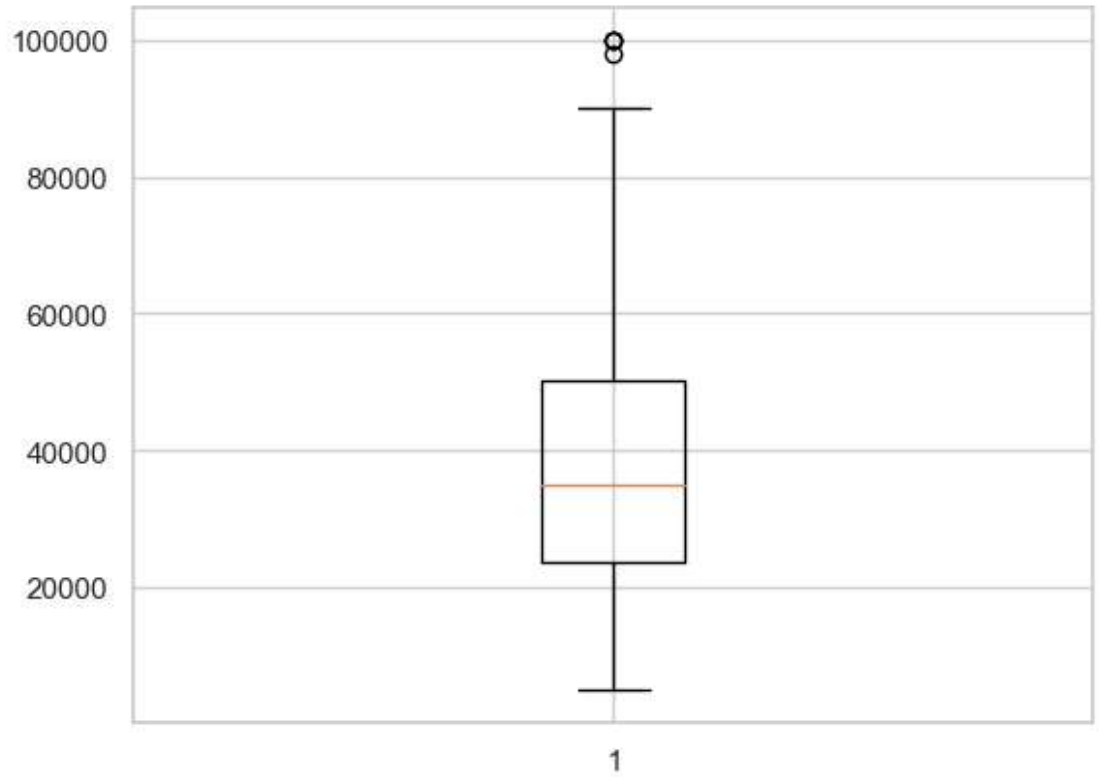
Out[40]: 90612.5

In [42]:
```
amir.describe(include="object")
```

Out[42]:

	Highest_Qualified_Member
count	50
unique	5
top	Graduate
freq	19

In [44]:
```
import matplotlib.pyplot as plt
plt.boxplot(amir["Mthly_HH_Income"])
plt.show()
```



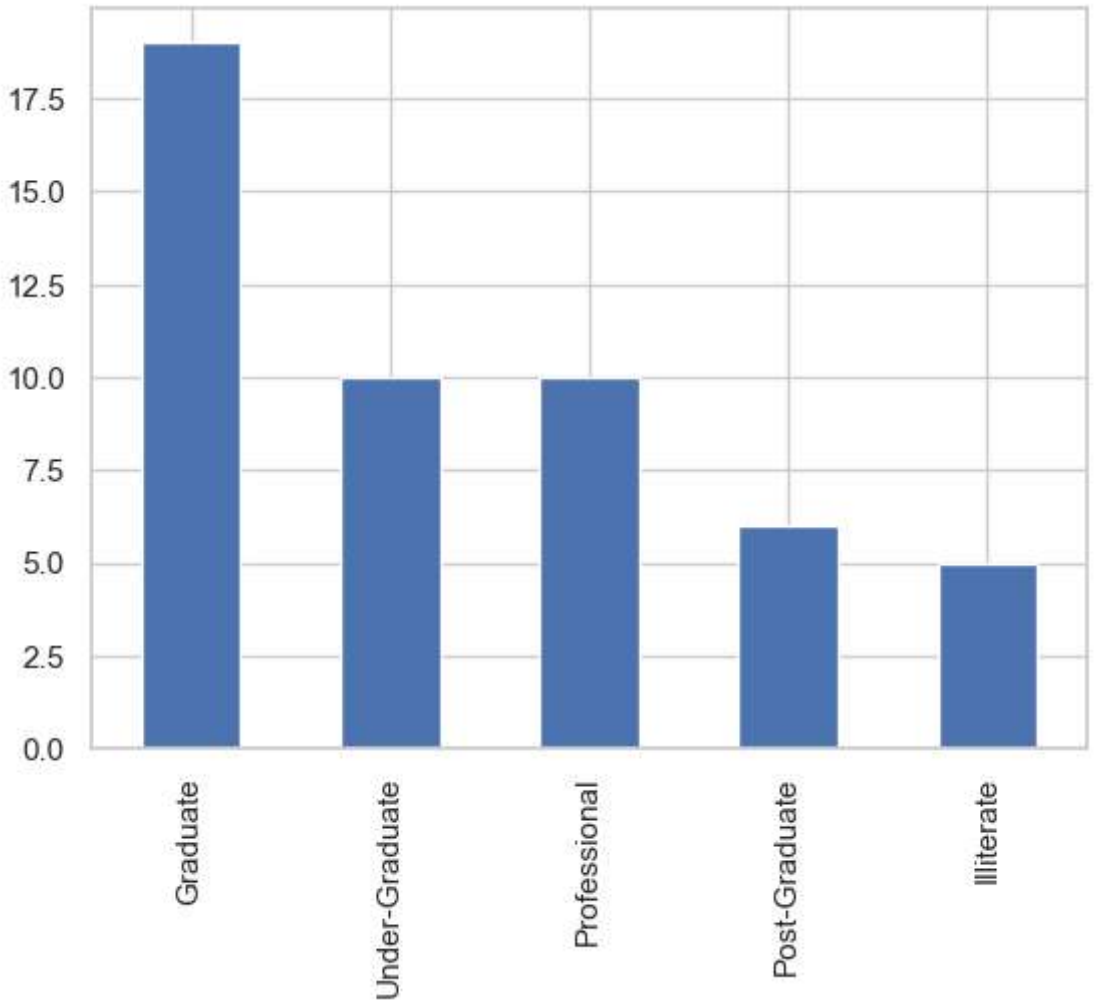
In [ ]:
```

```

visualisation

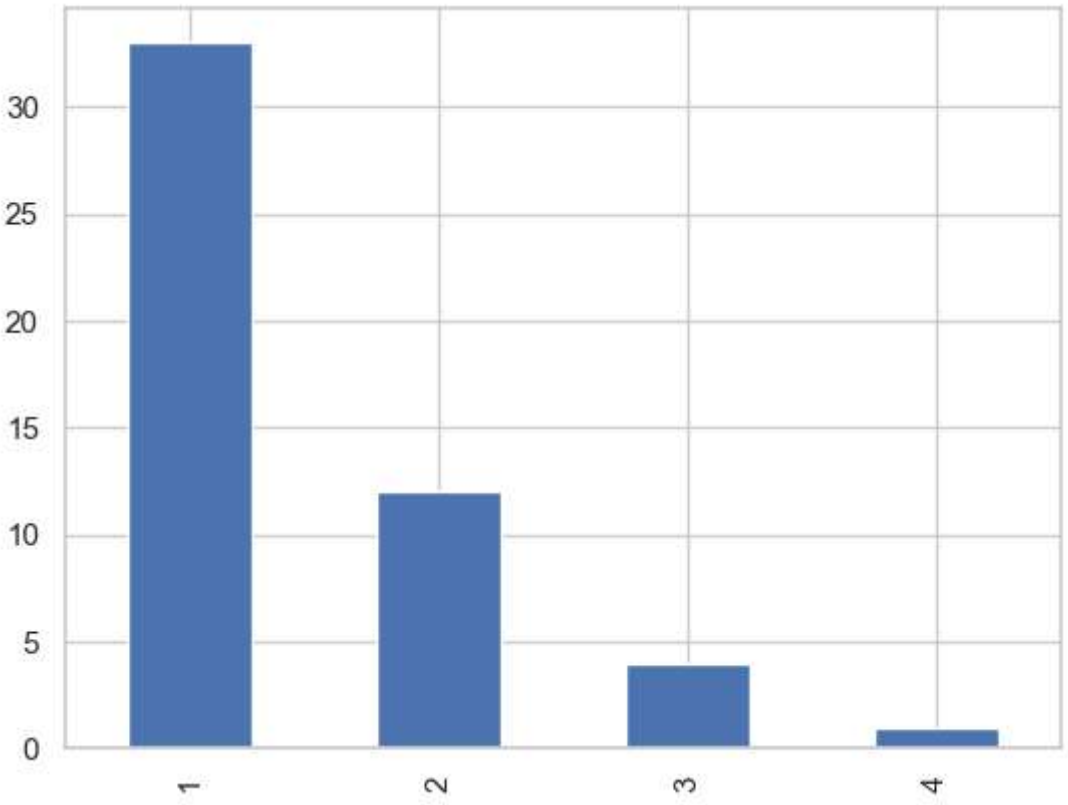
In [13]:
```
amir["Highest_Qualified_Member"].value_counts().plot(kind="bar")
```

Out[13]: <Axes: >

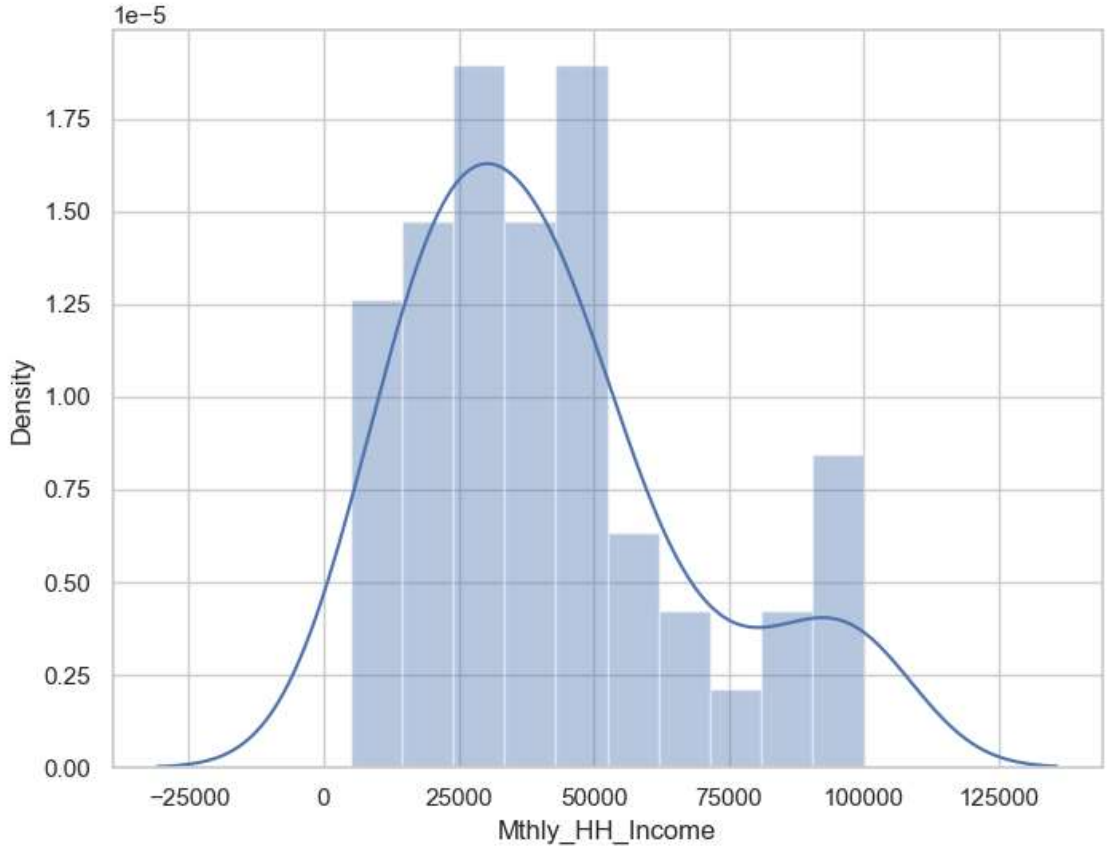


In [14]:
```
amir["No_of_Earning_Members"].value_counts().plot(kind="bar")
```

Out[14]: <Axes: >



In [15]:
```
f, ax = plt.subplots(figsize=(8, 6))
x = amir["Mthly_HH_Income"]
x = pd.Series(x, name="Mthly_HH_Income")
ax = sns.distplot(x, bins=10)
plt.show()
```

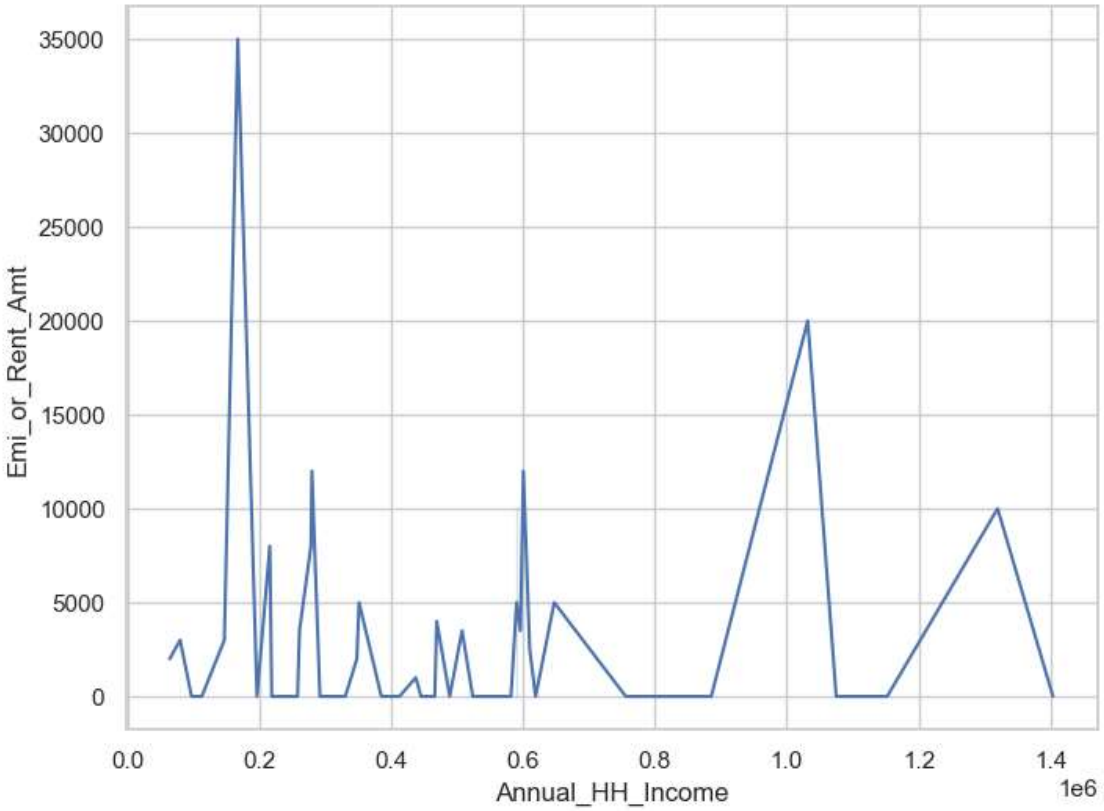


In [19]:
```
amir.head(1)
```

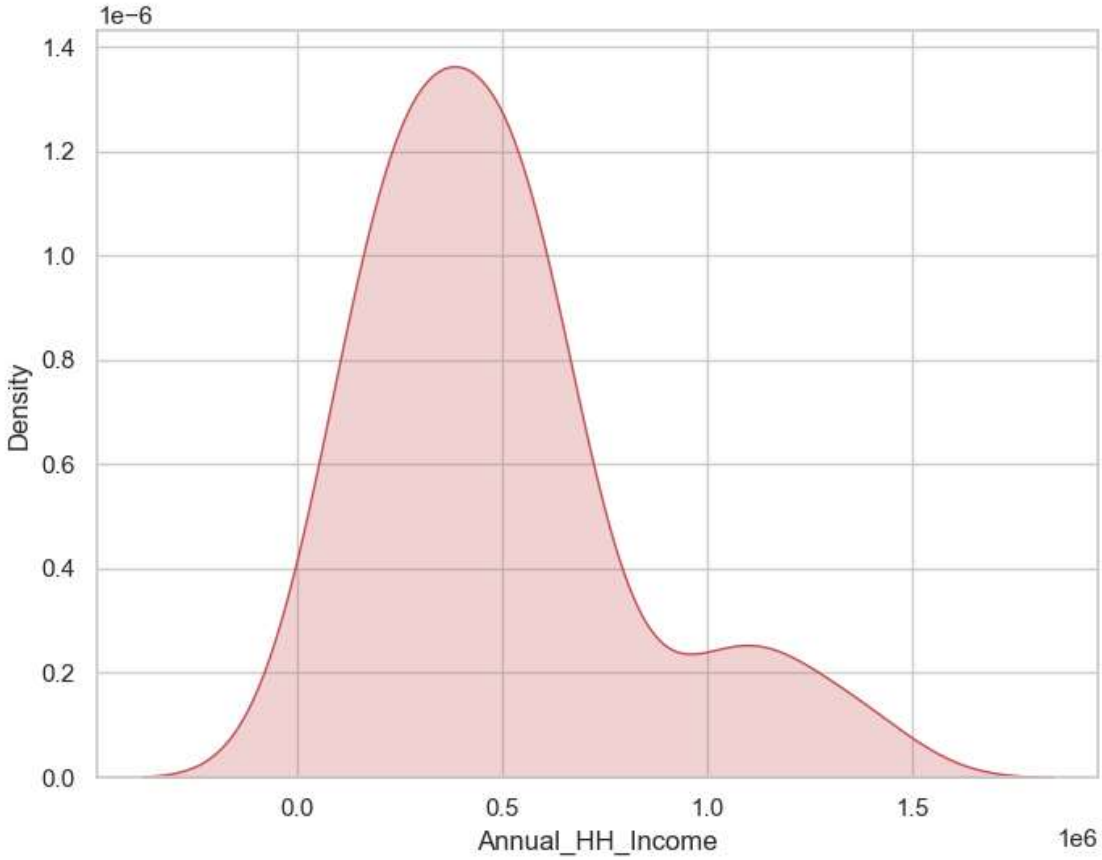
Out[19]:

	Mthly_HH_Income	Mthly_HH_Expense	No_of_Fly_Members	Emi_or_Rent_Amt	Annual_H
0	5000	8000	3	2000	

In [23]:
```
f, ax = plt.subplots(figsize=(8, 6))
ax = sns.lineplot(x="Annual_HH_Income", y="Emi_or_Rent_Amt", data=amir)
plt.show()
```



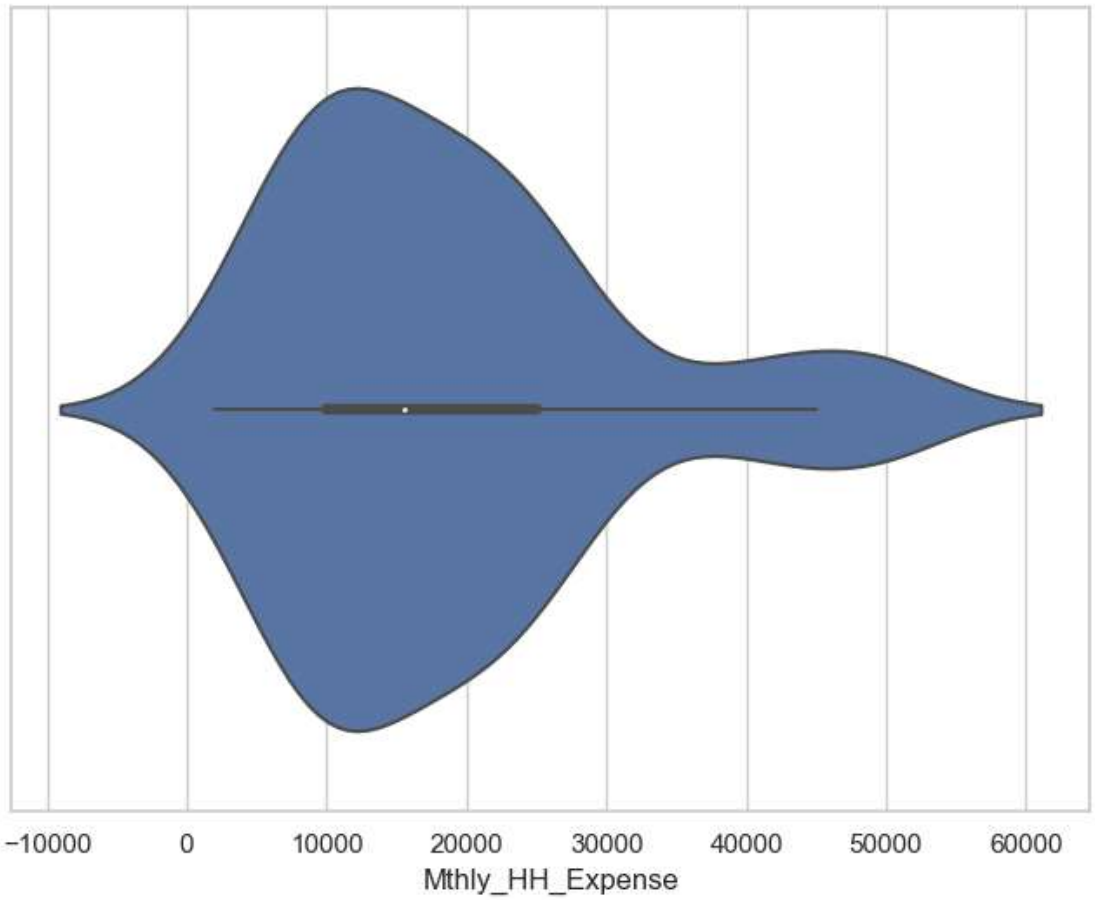
In [24]:
```
f, ax = plt.subplots(figsize=(8, 6))
x = amir["Annual_HH_Income"]
x = pd.Series(x, name="Annual_HH_Income")
ax = sns.kdeplot(x, shade=True, color="r")
plt.show()
```



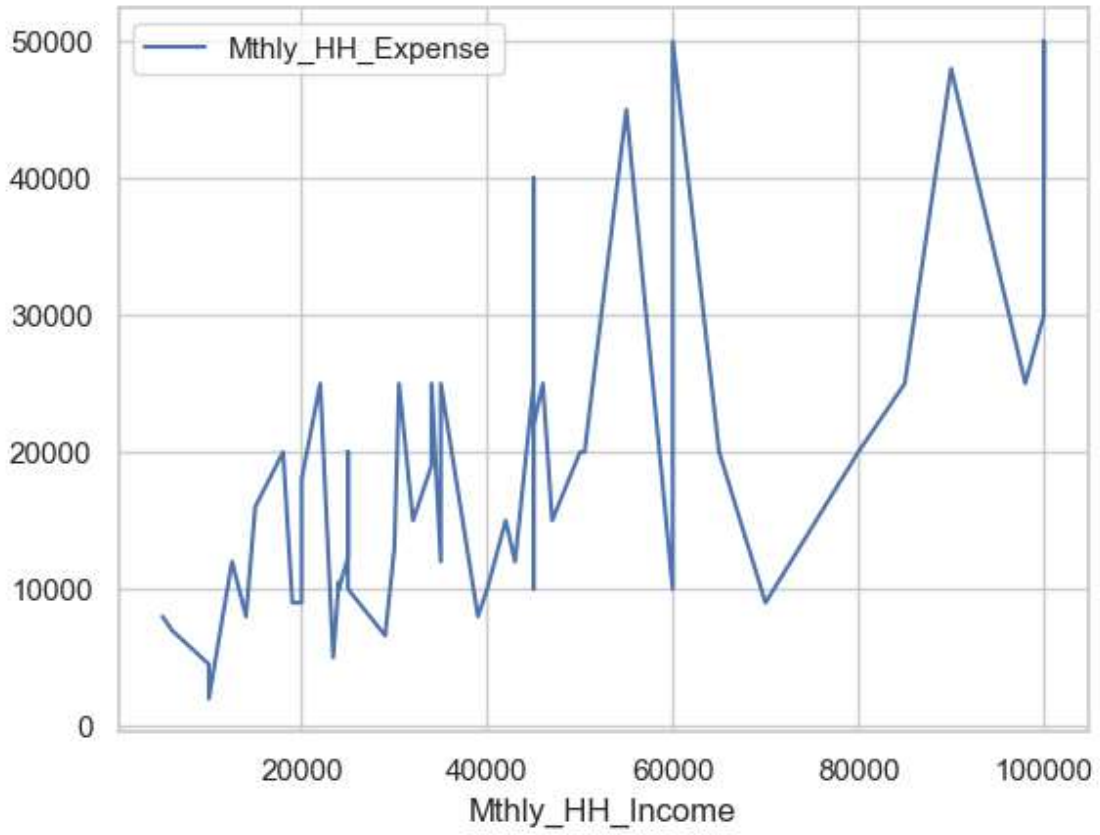
In [ ]:
```
amir.head(1)
```

seaborn violinplot() function

In [25]:
```
f, ax = plt.subplots(figsize=(8, 6))
sns.violinplot(x=amir["Mthly_HH_Expense"])
plt.show()
```



In [26]:
```
amir.plot("Mthly_HH_Income", y="Mthly_HH_Expense")
Q1 = amir["Mthly_HH_Expense"].quantile(0.75)
Q3 = amir["Mthly_HH_Expense"].quantile(0.25)
IQR = Q3 - Q1
```



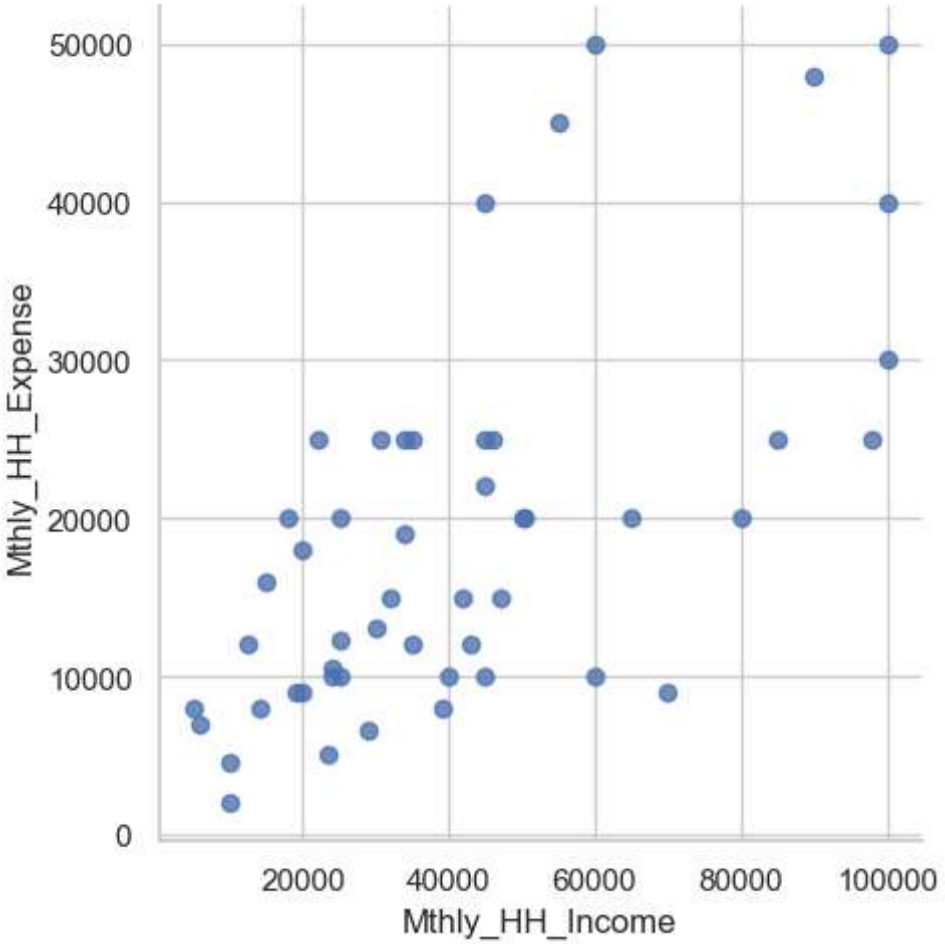
Advance visualisation

In [27]:
```
amir.head(1)
```

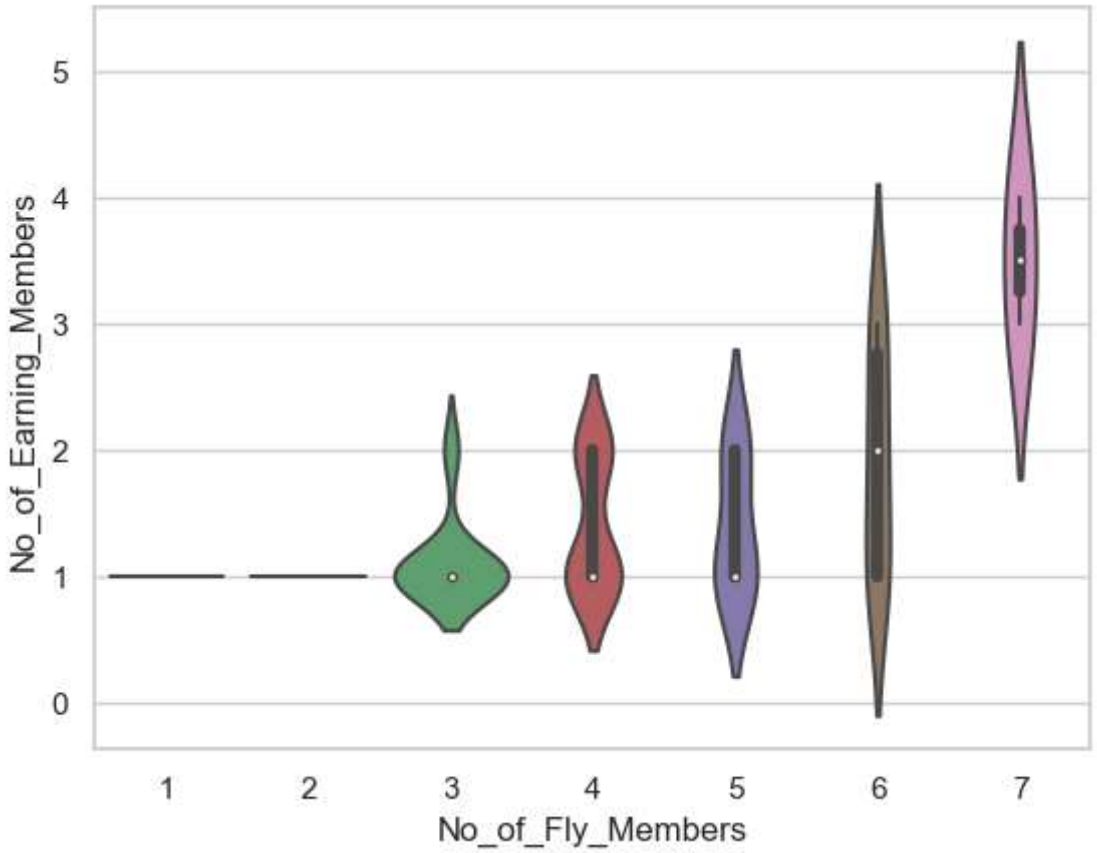
Out[27]:

	Mthly_HH_Income	Mthly_HH_Expense	No_of_Fly_Members	Emi_or_Rent_Amt	Annual_H
0	5000	8000	3	2000	

In [30]:
```
vis1 = sns.lmplot(data=amir, x="Mthly_HH_Income", y="Mthly_HH_Expense",\
                  fit_reg=False)
```



In [32]:
```
z = sns.violinplot(data=amir, x="No_of_Fly_Members", y = "No_of_Earning_Me
```



STANDARD DEVIATION

In [ ]:
```
pd.DataFrame(amir.iloc[:,0:5].std().to_frame()).T
```

First three column variance

In [ ]:
```
pd.DataFrame(amir.iloc[:,0:4].var().to_frame()).T
```

In [ ]:
```
amir["Highest_Qualified_Member"].value_counts().to_frame().T
```

In [ ]:
```

```