In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

%matplotlib inline

import warnings
warnings.filterwarnings('ignore')
```

In [2]:

```
= pd.read_csv(r'C:\Users\LENOVO\Desktop\Monty Datascien\16th,17th\16th,17th\Descriptive stats code- practivle\Inc_Exp_Data.csv')
```

In [3]:

```
df
```

Out[3]:

	Mthly_HH_Income	Mthly_HH_Expense	No_of_Fly_Members	Emi_or_Rent_Amt	Annual_HH_Income	Highest_Qualified_Member	No_of_Earning
0	5000	8000	3	2000	64200	Under-Graduate	
1	6000	7000	2	3000	79920	Illiterate	
2	10000	4500	2	0	112800	Under-Graduate	
3	10000	2000	1	0	97200	Illiterate	
4	12500	12000	2	3000	147000	Graduate	
5	14000	8000	2	0	196560	Graduate	
6	15000	16000	3	35000	167400	Post-Graduate	
7	18000	20000	5	8000	216000	Graduate	
8	19000	9000	2	0	218880	Under-Graduate	
9	20000	9000	4	0	220800	Under-Graduate	
10	20000	18000	4	8000	278400	Under-Graduate	
11	22000	25000	6	12000	279840	Illiterate	
12	23400	5000	3	0	292032	Illiterate	
13	24000	10500	6	0	316800	Graduate	
14	24000	10000	4	0	244800	Graduate	
15	25000	12300	3	0	246000	Graduate	
16	25000	20000	3	3500	261000	Graduate	
17	25000	10000	6	0	258000	Under-Graduate	
18	29000	6600	2	2000	348000	Graduate	
19	30000	13000	4	0	385200	Graduate	
20	30500	25000	5	5000	351360	Under-Graduate	
21	32000	15000	4	0	445440	Professional	
22	34000	19000	6	0	330480	Professional	
23	34000	25000	3	4000	469200	Professional	
24	35000	12000	3	0	466200	Graduate	
25	35000	25000	4	0	449400	Professional	
26	39000	8000	4	0	556920	Under-Graduate	
27	40000	10000	4	0	412800	Under-Graduate	
28	42000	15000	4	0	488880	Graduate	
29	43000	12000	4	0	619200	Graduate	
30	45000	25000	6	0	523800	Graduate	
31	45000	40000	6	3500	507600	Professional	
32	45000	10000	2	1000	437400	Post-Graduate	
33	45000	22000	4	2500	610200	Post-Graduate	
34	46000	25000	5	3500	596160	Graduate	
35	47000	15000	7	0	456840	Professional	
36	50000	20000	4	0	570000	Professional	
37	50500	20000	3	0	581760	Professional	
38	55000	45000	6	12000	600600	Graduate	
39	60000	10000	3	0	590400	Post-Graduate	
40	60000	50000	6	10000	590400	Graduate	
41	65000	20000	4	5000	647400	Illiterate	
42	70000	9000	2	0	756000	Graduate	
43	80000	20000	4	0	1075200	Graduate	
44	85000	25000	5	0	1142400	Under-Graduate	
45	90000	48000	7	0	885600	Post-Graduate	
46	98000	25000	5	0	1152480	Professional	
47	100000	30000	6	0	1404000	Graduate	
48	100000	50000	4	20000	1032000	Professional	
49	100000	40000	6	10000	1320000	Post-Graduate	

Understanding About Dataset

In [5]:

```
df.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 [7]:

```
df.shape
```

Out[7]:

```
(50, 7)
```

In [8]:

```
df.size
```

Out[8]:

```
350
```

In [9]:

```
df.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 [10]:

```
df.describe()
```

Out[10]:

	Mthly_HH_Income	Mthly_HH_Expense	No_of_Fly_Members	Emi_or_Rent_Amt	Annual_HH_Income	No_of_Earning_Members
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 [11]:

```
df.describe().T   # by using .T it will transpose rows into column and vice versa.
```

Out[11]:

	count	mean	std	min	25%	50%	75%	max
Mthly_HH_Income	50.0	41558.00	26097.908979	5000.0	23550.0	35000.0	50375.0	100000.0
Mthly_HH_Expense	50.0	18818.00	12090.216824	2000.0	10000.0	15500.0	25000.0	50000.0
No_of_Fly_Members	50.0	4.06	1.517382	1.0	3.0	4.0	5.0	7.0
Emi_or_Rent_Amt	50.0	3060.00	6241.434948	0.0	0.0	0.0	3500.0	35000.0
Annual_HH_Income	50.0	490019.04	320135.792123	64200.0	258750.0	447420.0	594720.0	1404000.0
No_of_Earning_Members	50.0	1.46	0.734291	1.0	1.0	1.0	2.0	4.0

Checking For Missing values or NAN

In [13]:

```
df.isna().any()
```

Out[13]:

```
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 [14]:

```
df.isnull().sum()
```

Out[14]:

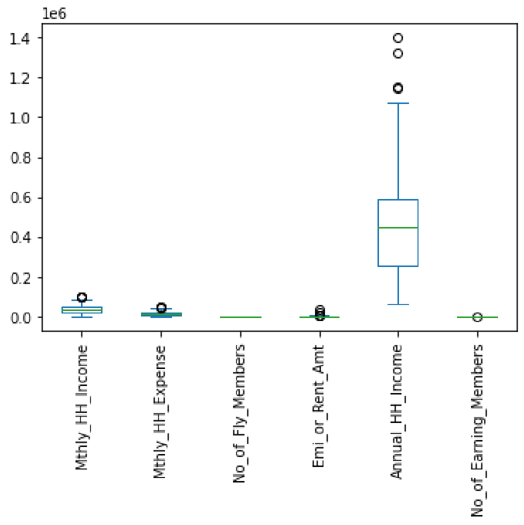
```
Mthly_HH_Income            0
Mthly_HH_Expense           0
No_of_Fly_Members          0
Emi_or_Rent_Amt            0
Annual_HH_Income           0
Highest_Qualified_Member   0
No_of_Earning_Members      0
dtype: int64
```

There are no missing values or nan in our dataset

Checkings For Outliers

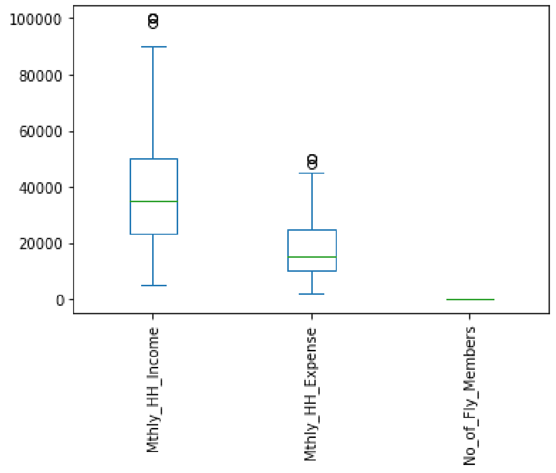
In [30]:

```
thly_HH_Income', 'Mthly_HH_Expense', 'No_of_Fly_Members','Emi_or_Rent_Amt', 'Annual_HH_Income', 'Highest_Qualified_Member','No_of_
icks(rotation =90)
ow()
```



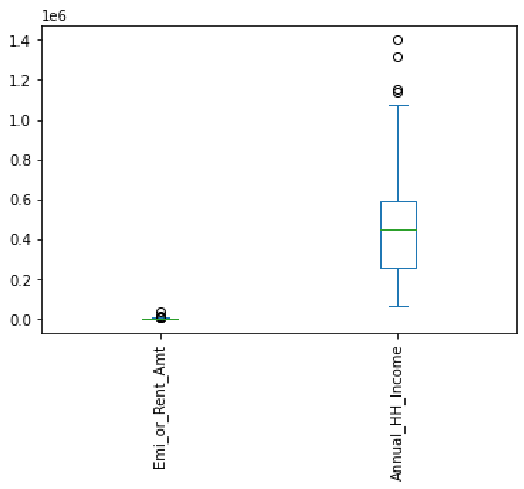
In [31]:

```
df[['Mthly_HH_Income', 'Mthly_HH_Expense', 'No_of_Fly_Members']].plot(kind='box')
plt.xticks(rotation = 90)
plt.show()
```



In [32]:

```
df[['Emi_or_Rent_Amt', 'Annual_HH_Income', 'Highest_Qualified_Member']].plot(kind='box')
plt.xticks(rotation = 90)
plt.show()
```



From the above graphs it is clear that we have outliers in data

Finding Mean Expense of a Household

In [33]:

```
df["Mthly_HH_Expense"].mean()
```

Out[33]:

```
18818.0
```

Finding Median Expense of a Household

In [34]:

```
df["Mthly_HH_Expense"].median()
```

Out[34]:

```
15500.0
```

In [36]:

```
df_exp = pd.crosstab(index=df["Mthly_HH_Expense"], columns="count")
df_exp.reset_index(inplace=True)
df_exp[df_exp['count'] == df.Mthly_HH_Expense.value_counts().max()]
```

Out[36]:

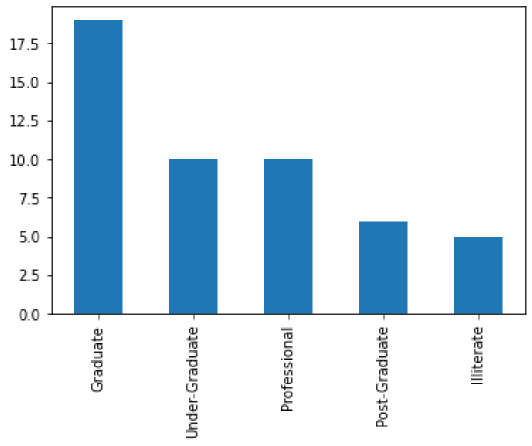
col_0	Mthly_HH_Expense	count
18	25000	8

In [37]:

```
df["Highest_Qualified_Member"].value_counts().plot(kind="bar")
```

Out[37]:

```
<AxesSubplot:>
```

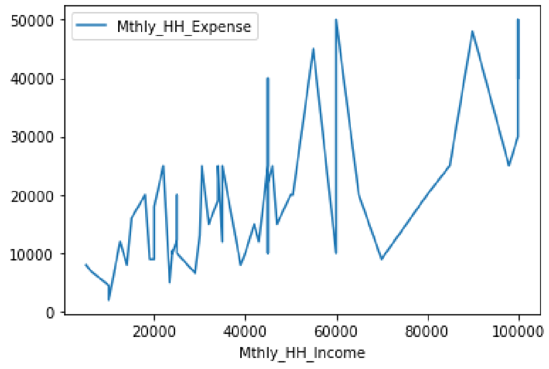


In [38]:

```
df.plot(x="Mthly_HH_Income", y="Mthly_HH_Expense")
```

Out[38]:

```
<AxesSubplot:xlabel='Mthly_HH_Income'>
```



Higher the income Higher is the expense

In [39]:

```
IQR=df["Mthly_HH_Expense"].quantile(0.75)- df["Mthly_HH_Expense"].quantile(0.25)
IQR
```

Out[39]:

```
15000.0
```

Finding Standard Deviation(std) for first 4 columns

In [40]:

```
pd.DataFrame(df.iloc[:,0:5].std().to_frame())
```

Out[40]:

	0
Mthly_HH_Income	26097.908979
Mthly_HH_Expense	12090.216824
No_of_Fly_Members	1.517382
Emi_or_Rent_Amt	6241.434948
Annual_HH_Income	320135.792123

In [41]:

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

Out[41]:

	Mthly_HH_Income	Mthly_HH_Expense	No_of_Fly_Members	Emi_or_Rent_Amt	Annual_HH_Income
0	26097.908979	12090.216824	1.517382	6241.434948	320135.792123

Finding Variance(var) for first 3 columns

In [42]:

```
pd.DataFrame(df.iloc[:,0:4].var().to_frame())
```

Out[42]:

	0
Mthly_HH_Income	6.811009e+08
Mthly_HH_Expense	1.461733e+08
No_of_Fly_Members	2.302449e+00
Emi_or_Rent_Amt	3.895551e+07

In [43]:

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

Out[43]:

	Mthly_HH_Income	Mthly_HH_Expense	No_of_Fly_Members	Emi_or_Rent_Amt
0	6.811009e+08	1.461733e+08	2.302449	3.895551e+07

Finding the values count of Highest qualified member

In [44]:

```
df["Highest_Qualified_Member"].value_counts().to_frame()
```

Out[44]:

	Highest_Qualified_Member
Graduate	19
Under-Graduate	10
Professional	10
Post-Graduate	6
Illiterate	5

In [45]:

```
df["Highest_Qualified_Member"].value_counts().to_frame().T
```

Out[45]:

	Graduate	Under-Graduate	Professional	Post-Graduate	Illiterate
Highest_Qualified_Member	19	10	10	6	5

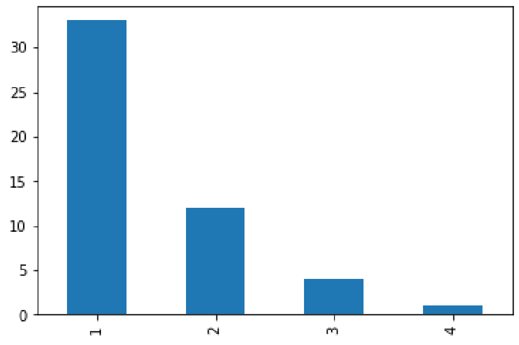
Plot of No_of_Earning_Members

In [53]:

```
df["No_of_Earning_Members"].value_counts().plot(kind="bar")
```

Out[53]:

```
<AxesSubplot:>
```

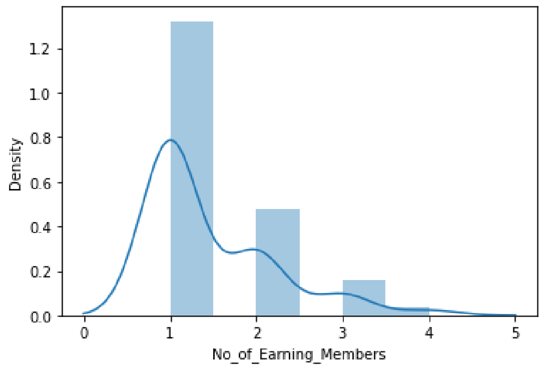


In [47]:

```
sns.distplot(df['No_of_Earning_Members'])
```

Out[47]:

```
<AxesSubplot:xlabel='No_of_Earning_Members', ylabel='Density'>
```



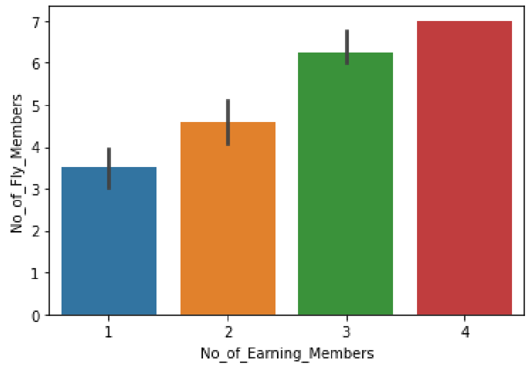
From these 2 graphs we understood that most of the family only have 1 person who is earing

In [59]:

```
sns.barplot('No_of_Earning_Members','No_of_Fly_Members',data=df)
```

Out[59]:

```
<AxesSubplot:xlabel='No_of_Earning_Members', ylabel='No_of_Fly_Members'>
```



From here we understand that families having more than 6 members have more earning members(Almost half the size of familymembers)

In [ ]:

```
```