1.Import necessary packages

In [14]:
```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import matplotlib.pyplot as plt
import seaborn as sns
```

2.Load the file

In [15]:
```
income_df=pd.read_csv(r'C:\Users\arati\Downloads\10th, 11th- Intro to Stats, Des
```

In [16]:
```
income_df
```

Out[16]:

	Mthly_HH_Income	Mthly_HH_Expense	No_of_Fly_Members	Emi_or_Rent_Amt	Annu
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	
5	14000	8000	2	0	
6	15000	16000	3	35000	
7	18000	20000	5	8000	
8	19000	9000	2	0	
9	20000	9000	4	0	
10	20000	18000	4	8000	
11	22000	25000	6	12000	
12	23400	5000	3	0	
13	24000	10500	6	0	
14	24000	10000	4	0	
15	25000	12300	3	0	
16	25000	20000	3	3500	
17	25000	10000	6	0	
18	29000	6600	2	2000	
19	30000	13000	4	0	
20	30500	25000	5	5000	
21	32000	15000	4	0	
22	34000	19000	6	0	
23	34000	25000	3	4000	
24	35000	12000	3	0	
25	35000	25000	4	0	
26	39000	8000	4	0	
27	40000	10000	4	0	
28	42000	15000	4	0	
29	43000	12000	4	0	
30	45000	25000	6	0	
31	45000	40000	6	3500	
32	45000	10000	2	1000	

	Mthly_HH_Income	Mthly_HH_Expense	No_of_Fly_Members	Emi_or_Rent_Amt	Annu
33	45000	22000	4	2500	
34	46000	25000	5	3500	
35	47000	15000	7	0	
36	50000	20000	4	0	
37	50500	20000	3	0	
38	55000	45000	6	12000	
39	60000	10000	3	0	
40	60000	50000	6	10000	
41	65000	20000	4	5000	
42	70000	9000	2	0	
43	80000	20000	4	0	
44	85000	25000	5	0	
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	

3.Analyze the data

In [4]:
```
income_df.info() # see the informations
```

```
<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 [6]:
```
income_df.shape # see number of rows and columns
```

Out[6]: (50, 7)

In [5]:
```
income_df.describe() #Descriptive Statistics
```

Out[5]:

	Mthly_HH_Income	Mthly_HH_Expense	No_of_Fly_Members	Emi_or_Rent_Amt	Ar
count	50.000000	50.000000	50.000000	50.000000	
mean	41558.000000	18818.000000	4.060000	3060.000000	
std	26097.908979	12090.216824	1.517382	6241.434948	
min	5000.000000	2000.000000	1.000000	0.000000	
25%	23550.000000	10000.000000	3.000000	0.000000	
50%	35000.000000	15500.000000	4.000000	0.000000	
75%	50375.000000	25000.000000	5.000000	3500.000000	
max	100000.000000	50000.000000	7.000000	35000.000000	

In [7]:
```
income_df.describe().T #Transpose row to column and column to row
```

Out[7]:

	count	mean	std	min	25%	50%
Mthly_HH_Income	50.0	41558.00	26097.908979	5000.0	23550.0	35000.0
Mthly_HH_Expense	50.0	18818.00	12090.216824	2000.0	10000.0	15500.0
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
No_of_Earning_Members	50.0	1.46	0.734291	1.0	1.0	1.0

In [8]:
```
income_df.isna().any()
```

Out[8]:
```
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
```

4.What is the Mean expense of a household ?

In [9]:
```
income_df["Mthly_HH_Expense"].mean()
```

Out[9]:
```
18818.0
```

5.What is the Median household Expense ?

In [10]:
```
income_df["Mthly_HH_Expense"].median()
```

Out[10]:
```
15500.0
```

6.What is the Monthly Expense for the most of the Households ?

In [11]:
```
mth_exp_tmp = pd.crosstab(index=income_df["Mthly_HH_Expense"], columns="count")
mth_exp_tmp.reset_index(inplace=True)
mth_exp_tmp[mth_exp_tmp['count'] == income_df.Mthly_HH_Expense.value_counts().ma
```

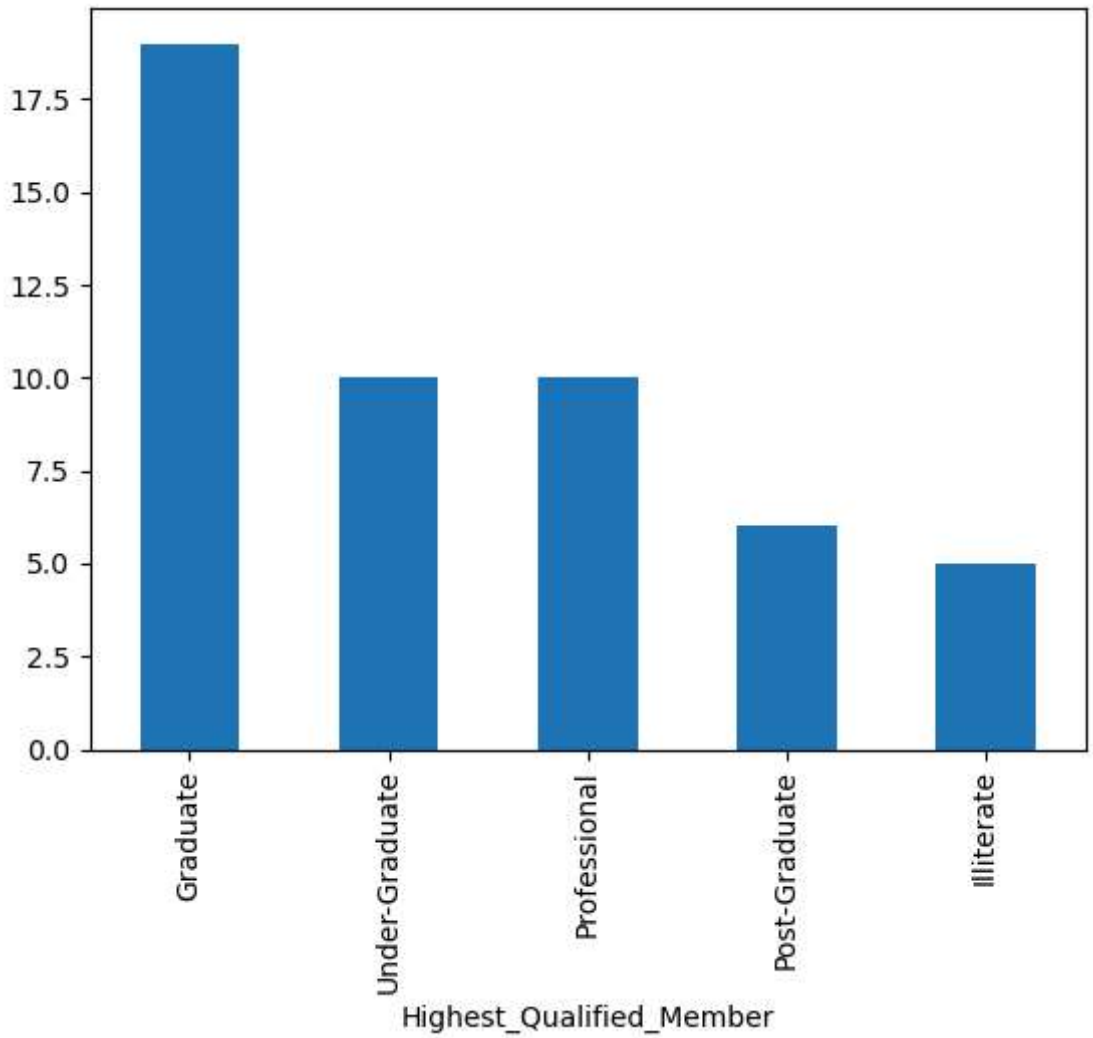
Out[11]:

col_0	Mthly_HH_Expense	count
18	25000	8

7.Plot the histogram to count the highest qualified member

In [12]:
```
income_df["Highest_Qualified_Member"].value_counts().plot(kind="bar")
```

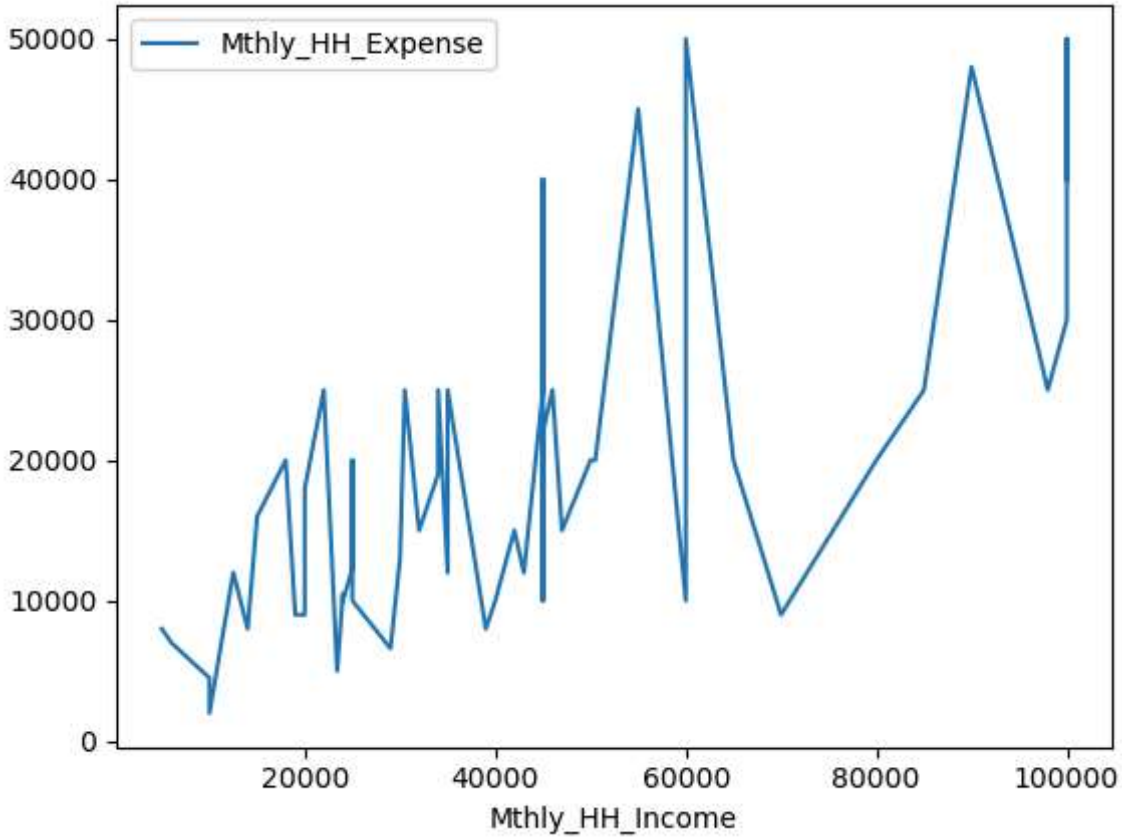
Out[12]:
```
<Axes: xlabel='Highest_Qualified_Member'>
```



8.Calculate IQR(difference between 75% and 25% quartile)

In [18]:
```
income_df.plot(x="Mthly_HH_Income", y="Mthly_HH_Expense")
IQR=income_df["Mthly_HH_Expense"].quantile(0.75)-income_df["Mthly_HH_Expense"].q
IQR
```

Out[18]: 15000.0



9.Calculate Standard Deviation for first 4 columns.¶

In [19]:
```
pd.DataFrame(income_df.iloc[:,0:5].std().to_frame()).T
```

Out[19]:

	Mthly_HH_Income	Mthly_HH_Expense	No_of_Fly_Members	Emi_or_Rent_Amt	Annua
0	26097.908979	12090.216824	1.517382	6241.434948	3

10.Calculate Variance for first 3 columns.

In [20]:
```
pd.DataFrame(income_df.iloc[:,0:4].var().to_frame()).T
```

Out[20]:

	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

11.Calculate the count of Highest qualified member.

In [21]:
```
income_df["Highest_Qualified_Member"].value_counts().to_frame().T
```

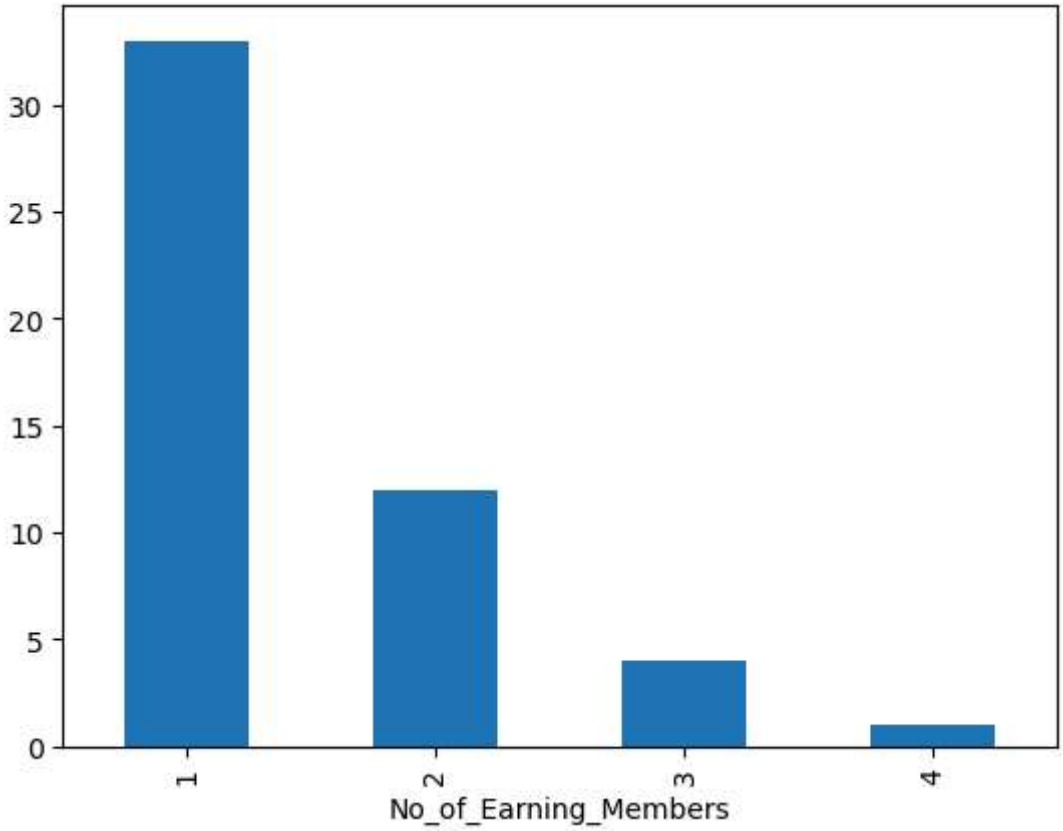
Out[21]:

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

12.Plot the Histogram to count the No_of_Earning_Members

In [22]:
```
income_df["No_of_Earning_Members"].value_counts().plot(kind="bar")
```

Out[22]: <Axes: xlabel='No_of_Earning_Members'>



13.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 [23]:
```
#Here we need to calculate the coeff of variation
```

```python
Coeff_of_var_StockA=10/15
print(Coeff_of_var_StockA)
Coeff_of_var_StockB=5/10
print(Coeff_of_var_StockB)
```

```
0.6666666666666666
0.5
```