In [1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt



import os
for dirname, _,filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname,filename))
```

2.Load the file

In [3]:
```
income_df=pd.read_csv(r"D:\Data Science with AI\Data Science With AI\6th, 7th -a
```

In [4]:
```
income_df.head()
```

Out[4]:

	Mthly_HH_Income	Mthly_HH_Expense	No_of_Fly_Members	Emi_or_Rent_Amt	Annua
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	

3.Analyze the data

In [5]:
```
income_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 [6]:
```
income_df['Highest_Qualified_Member']=income_df['Highest_Qualified_Member'].asty
```

In [7]:
```
income_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     category
 6   No_of_Earning_Members   50 non-null     int64
dtypes: category(1), int64(6)
memory usage: 2.7 KB
```

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

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

In [9]:
```
income_df.describe().T
```

Out[9]:

	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 [10]:
```
income_df.isna.any()
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[10], line 1
----> 1 income_df.isna.any()

AttributeError: 'function' object has no attribute 'any'
```

In [11]:
```
income_df.isna().any()
```

Out[11]:
```
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 [12]:
```
income_df["Mthly_HH_Expense"].mean()
```

Out[12]:
```
np.float64(18818.0)
```

In [14]:
```
income_df['Mthly_HH_Expense'].median()
```

Out[14]:
```
15500.0
```

In [15]:
```
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().max(
```

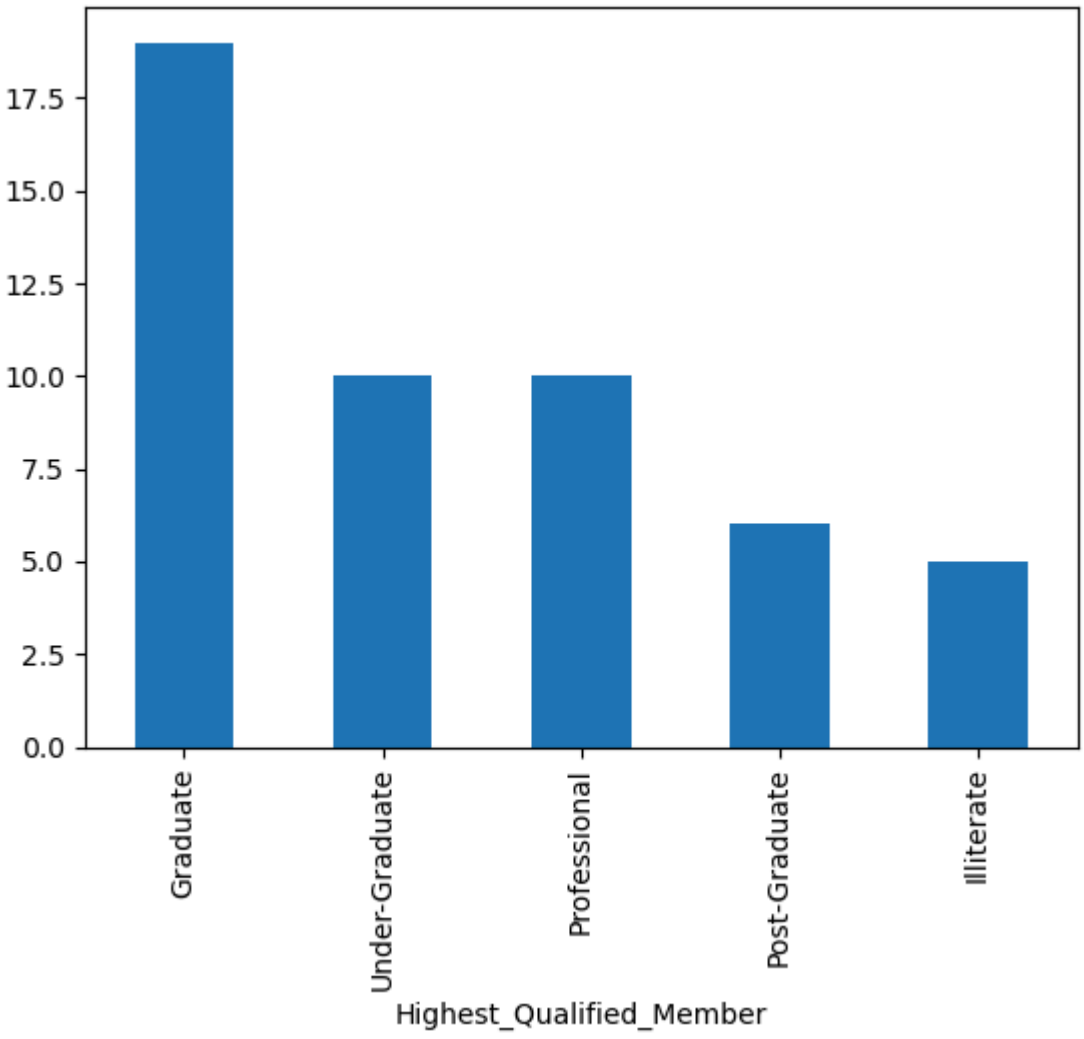
Out[15]:

	col_0	Mthly_HH_Expense	count
	18	25000	8

7.Plot the Histogram to count the Highest qualified member

In [17]:
```
income_df['Highest_Qualified_Member'].value_counts().plot(kind='bar')
```

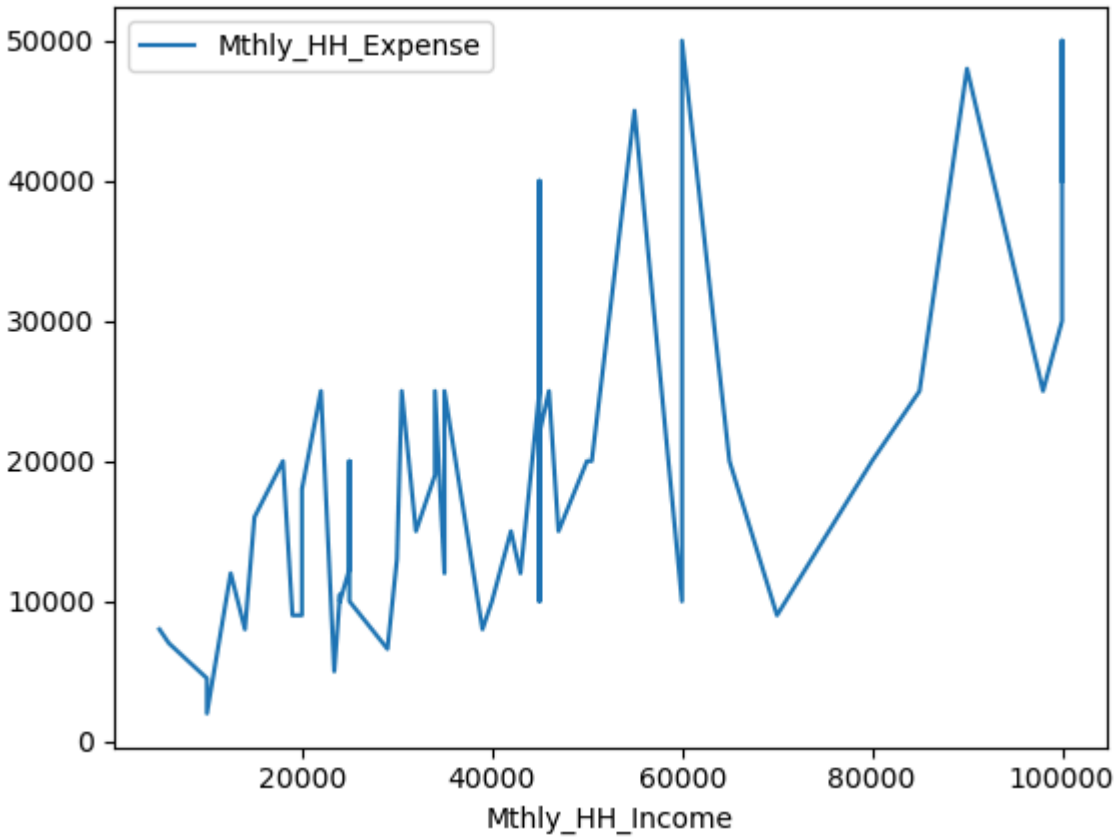
Out[17]:
```
<Axes: xlabel='Highest_Qualified_Member'>
```



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]: np.float64(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

In [21]:
```
pd.DataFrame(income_df.iloc[:,0:4].var().to_frame())
```

Out[21]:

	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

11.Calculate the count of Highest qualified member.

In [22]:
```
income_df['Highest_Qualified_Member'].value_counts().to_frame().T
```

Out[22]:

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

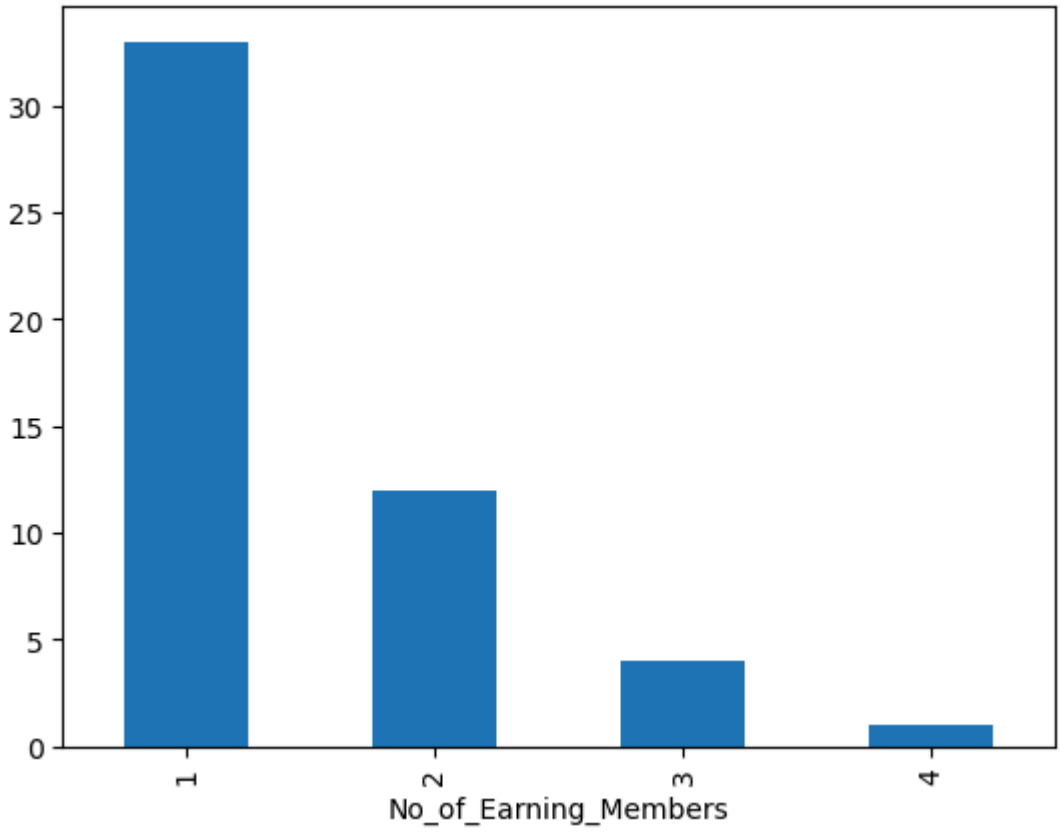
In [23]:
```
income_df['Highest_Qualified_Member']
```

Out[23]:
```
0     Under-Graduate
1         Illiterate
2     Under-Graduate
3         Illiterate
4           Graduate
5           Graduate
6      Post-Graduate
7           Graduate
8     Under-Graduate
9     Under-Graduate
10    Under-Graduate
11        Illiterate
12        Illiterate
13          Graduate
14          Graduate
15          Graduate
16          Graduate
17    Under-Graduate
18          Graduate
19          Graduate
20    Under-Graduate
21      Professional
22      Professional
23      Professional
24          Graduate
25      Professional
26    Under-Graduate
27    Under-Graduate
28          Graduate
29          Graduate
30          Graduate
31      Professional
32     Post-Graduate
33     Post-Graduate
34          Graduate
35      Professional
36      Professional
37      Professional
38          Graduate
39     Post-Graduate
40          Graduate
41        Illiterate
42          Graduate
43          Graduate
44    Under-Graduate
45     Post-Graduate
46      Professional
47          Graduate
48      Professional
49     Post-Graduate
Name: Highest_Qualified_Member, dtype: category
Categories (5, object): ['Graduate', 'Illiterate', 'Post-Graduate', 'Profession
al', 'Under-Graduate']
```

12.Plot the Histogram to count the No_of_Earning_Members

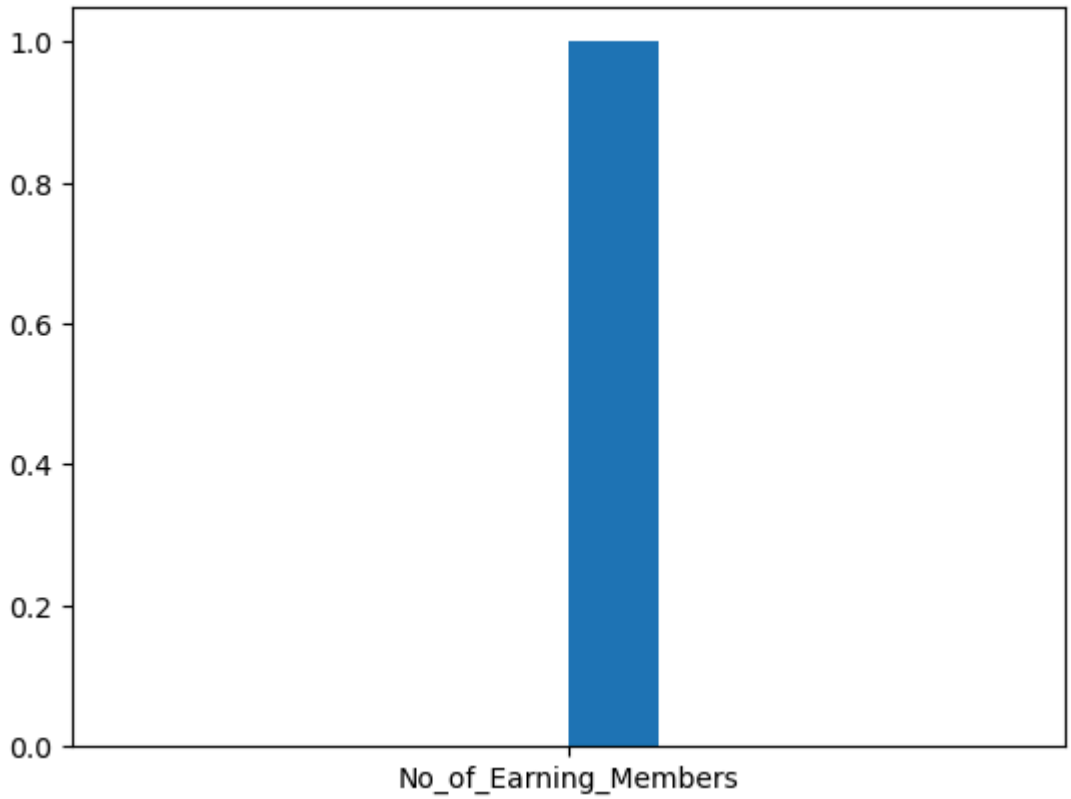
In [24]:
```
income_df['No_of_Earning_Members'].value_counts().plot(kind='bar')
```

Out[24]:
```
<Axes: xlabel='No_of_Earning_Members'>
```



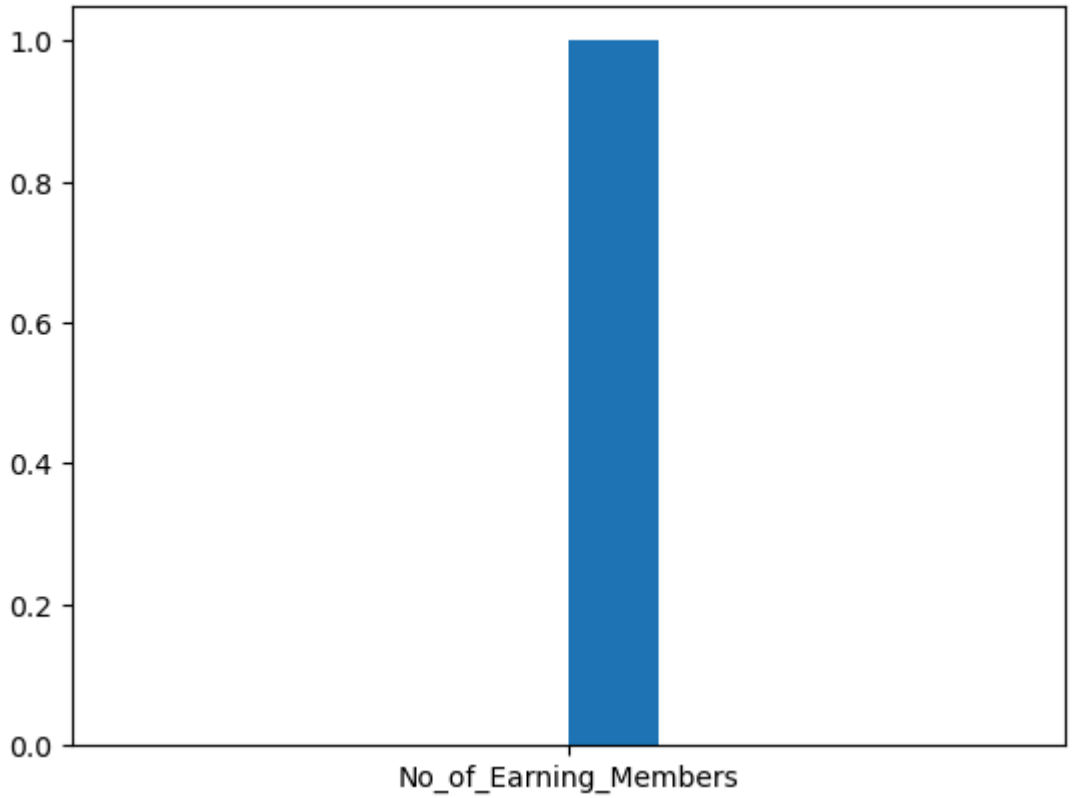
In [26]:
```
plt.hist('No_of_Earning_Members')
```

Out[26]:
```
(array([0., 0., 0., 0., 0., 1., 0., 0., 0., 0.]),
 array([-0.5, -0.4, -0.3, -0.2, -0.1,  0. ,  0.1,  0.2,  0.3,  0.4,  0.5]),
 <BarContainer object of 10 artists>)
```



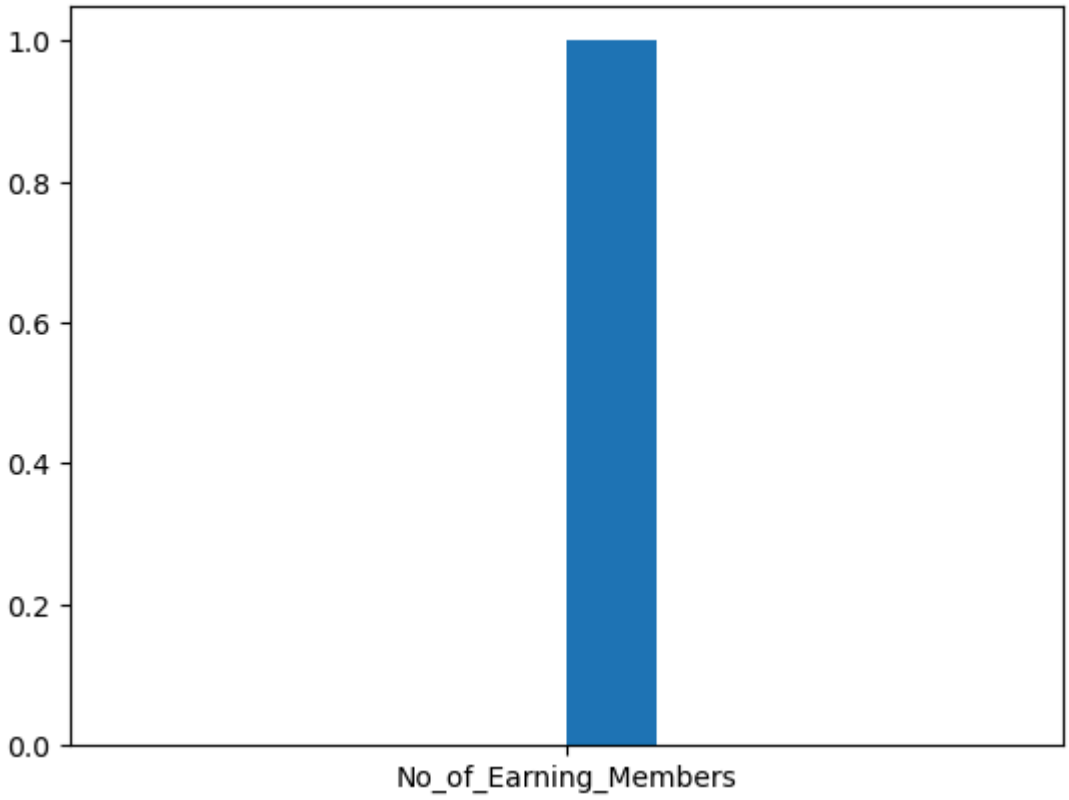
In [27]:
```
plt.hist('No_of_Earning_Members').value_counts()
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[27], line 1
----> 1 plt.hist(                        ).value_counts()

AttributeError: 'tuple' object has no attribute 'value_counts'
```



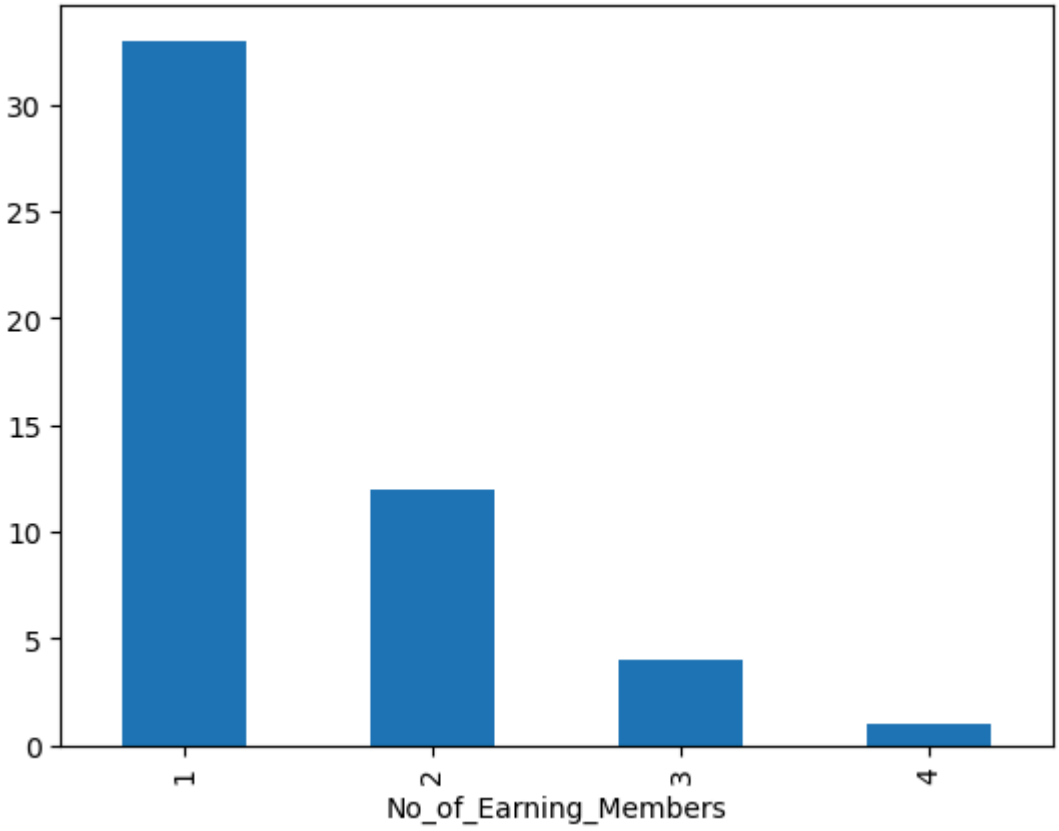
In [28]:
```
plt.hist('No_of_Earning_Members').value_counts().plot(kind='bar')
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[28], line 1
----> 1 plt.hist(                        ).value_counts().plot(kind='bar')

AttributeError: 'tuple' object has no attribute 'value_counts'
```



In [29]:
```
income_df['No_of_Earning_Members'].value_counts().plot(kind='bar')
```

Out[29]: <Axes: xlabel='No_of_Earning_Members'>



In [31]:
```
income_df['No_of_Earning_Members'].value_counts()
```

Out[31]:
```
No_of_Earning_Members
1    33
2    12
3     4
4     1
Name: count, dtype: int64
```

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 os 10%.Standard Deviation of the returns of these stocks is 10% and 5 % respectively

In [33]:
```
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
```

In [ ]:
```

```