```
import numpy as np # liner algebra
import pandas as pd #data processing, CSV file I/O (e.g. pd.read_csv)
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))
```

Load the file

```
# Replace 'your_file.csv' with the actual CSV file name in the folder
income_df = pd.read_csv(r'C:\Users\Bhavesh Govind Bhork\Downloads\all
repo\Inc_Exp_Data.csv')

income_df.head()
```

```
   Mthly_HH_Income  Mthly_HH_Expense  No_of_Fly_Members
Emi_or_Rent_Amt  \
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

   Annual_HH_Income Highest_Qualified_Member  No_of_Earning_Members
0             64200          Under-Graduate                      1
1             79920              Illiterate                      1
2            112800          Under-Graduate                      1
3             97200              Illiterate                      1
4            147000                Graduate                      1
```

Lets analyze the data

```
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
```

```
income_df.shape
```

```
(50, 7)
```

```
income_df.describe().T #int this code we we have used .T to transpose the dataframe, which shows mean mode etc
```

```
                        count       mean             std      min
25%  \
Mthly_HH_Income          50.0   41558.00   26097.908979   5000.0
23550.0
Mthly_HH_Expense         50.0   18818.00   12090.216824   2000.0
10000.0
No_of_Fly_Members        50.0       4.06       1.517382      1.0
3.0
Emi_or_Rent_Amt          50.0    3060.00    6241.434948      0.0
0.0
Annual_HH_Income         50.0  490019.04  320135.792123  64200.0
258750.0
No_of_Earning_Members    50.0       1.46       0.734291      1.0
1.0

                            50%       75%        max
Mthly_HH_Income         35000.0   50375.0   100000.0
Mthly_HH_Expense        15500.0   25000.0    50000.0
No_of_Fly_Members           4.0       5.0        7.0
Emi_or_Rent_Amt             0.0    3500.0    35000.0
Annual_HH_Income       447420.0  594720.0  1404000.0
No_of_Earning_Members       1.0       2.0        4.0
```

```
income_df.isna().any()
```

```
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
```

false bcz there is no null values in our dataset

Q. What is the mean expense of a household?

```
income_df.columns
```

```
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')
```

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

```
18818.0
```

Q. What is the median household expense?

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

```
15500.0
```

Q. What is monthly expense for most of households?

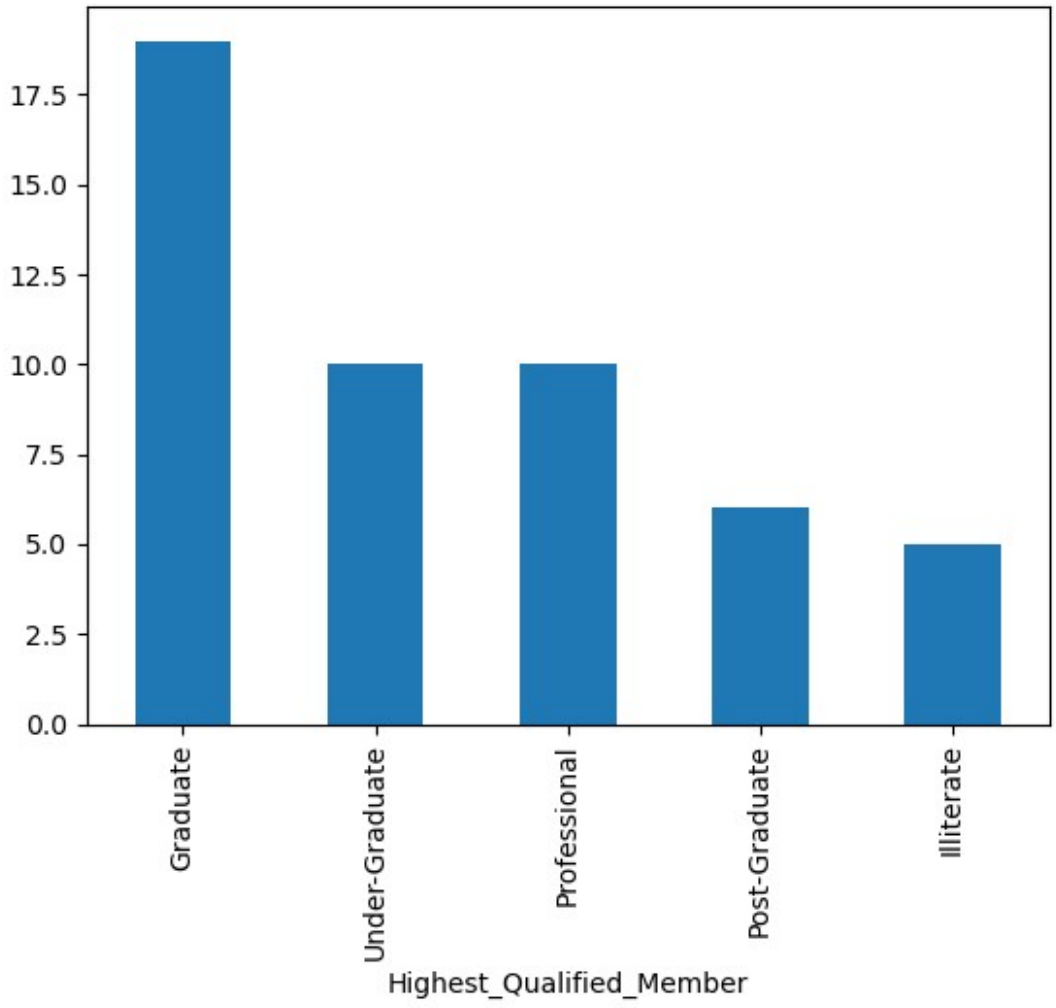
```
# Find the most common monthly household expense(s)
mth_exp_tmp =
income_df["Mthly_HH_Expense"].value_counts().reset_index()
mth_exp_tmp.columns = ["Mthly_HH_Expense", "count"]
mth_exp_tmp[mth_exp_tmp['count'] == mth_exp_tmp['count'].max()]
```

```
   Mthly_HH_Expense  count
0             25000      8
```

Q. plot histogram to count the highest qualified member

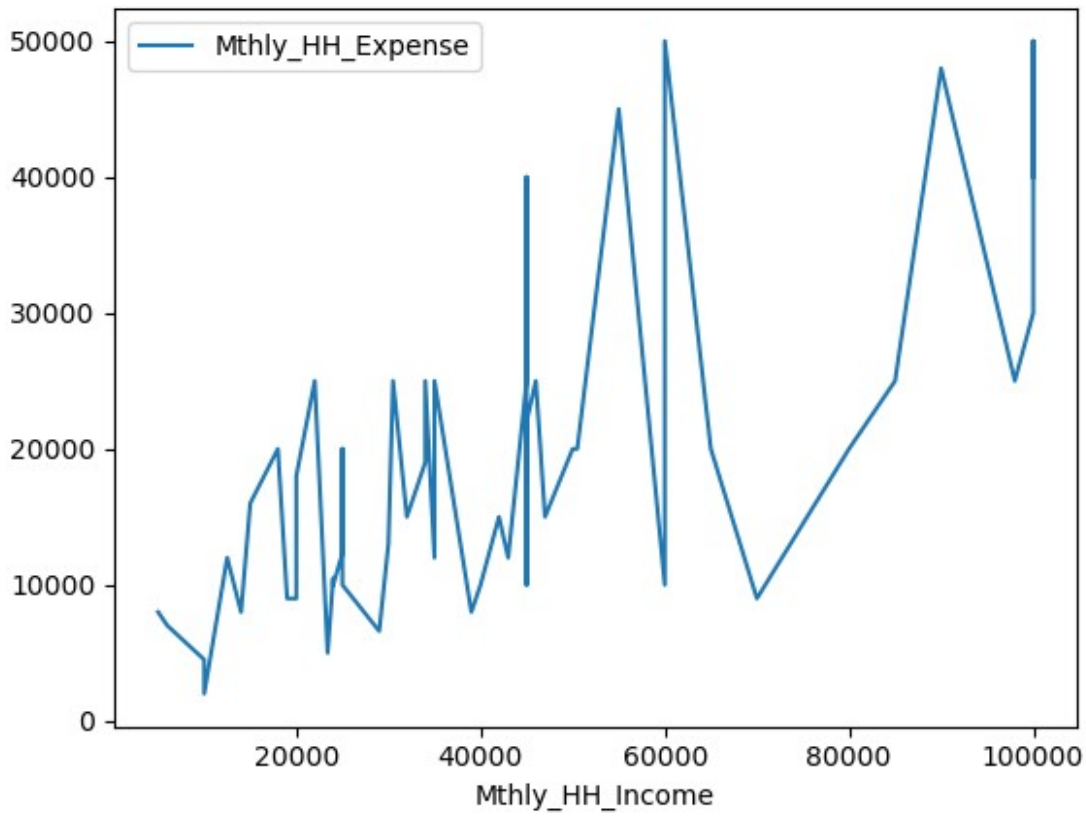
```
income_df["Highest_Qualified_Member"].value_counts().plot(kind="bar")
```

```
<Axes: xlabel='Highest_Qualified_Member'>
```



Q. Calculate IOR (difference btwn 75% nd 25% qurtile)

```
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"].quantile(0.25)
```



Q. Calculate standard dev for 1st 4 column

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

```
   Mthly_HH_Income  Mthly_HH_Expense  No_of_Fly_Members
Emi_or_Rent_Amt  \
0     26097.908979      12090.216824           1.517382
6241.434948

   Annual_HH_Income
0     320135.792123
```

Q, Calculate variance for 1st 3 columns

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

```
   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
```

Q. calculate the count of highest quilified member

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

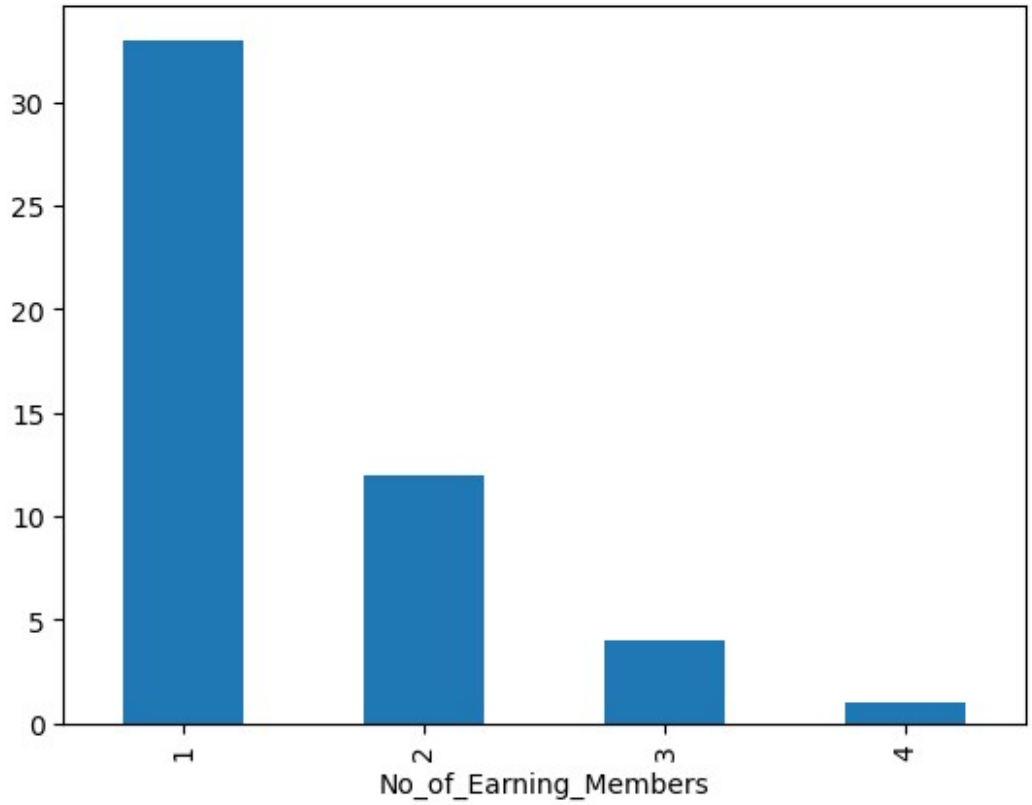
```
Highest_Qualified_Member  Graduate  Under-Graduate  Professional  \
count                           19              10            10

Highest_Qualified_Member  Post-Graduate  Illiterate
count                             6           5
```

Q. Plot the histogram to count the no of Earning Members

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

```
<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.

Q. Which Investment is better?

```
#Here we need to calculate the coeff of variation

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
```