1.Import necessary packages

In [2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)
```

2.Load the file

In [3]:
```
df=pd.read_csv("https://raw.githubusercontent.com/Mukund94/Datasets/main/Inc_Exp_Data%20(1).csv",sep=',')
```

In [4]:
```
df.head(20)
```

Out[4]:

	Mthly_HH_Income	Mthly_HH_Expense	No_of_Fly_Members	Emi_or_Rent_Amt	Annual_HH_Income	Highest_Qualified_Member	No_of_Earning_Members
0	5000	8000	3	2000	64200	Under-Graduate	1
1	6000	7000	2	3000	79920	Illiterate	1
2	10000	4500	2	0	112800	Under-Graduate	1
3	10000	2000	1	0	97200	Illiterate	1
4	12500	12000	2	3000	147000	Graduate	1
5	14000	8000	2	0	196560	Graduate	1
6	15000	16000	3	35000	167400	Post-Graduate	1
7	18000	20000	5	8000	216000	Graduate	1
8	19000	9000	2	0	218880	Under-Graduate	1
9	20000	9000	4	0	220800	Under-Graduate	2
10	20000	18000	4	8000	278400	Under-Graduate	2
11	22000	25000	6	12000	279840	Illiterate	1
12	23400	5000	3	0	292032	Illiterate	1
13	24000	10500	6	0	316800	Graduate	2
14	24000	10000	4	0	244800	Graduate	2
15	25000	12300	3	0	246000	Graduate	1
16	25000	20000	3	3500	261000	Graduate	1
17	25000	10000	6	0	258000	Under-Graduate	3
18	29000	6600	2	2000	348000	Graduate	1
19	30000	13000	4	0	385200	Graduate	1

3.Analyze the data

In [5]:
```
df.sample(15)
```

Out[5]:

	Mthly_HH_Income	Mthly_HH_Expense	No_of_Fly_Members	Emi_or_Rent_Amt	Annual_HH_Income	Highest_Qualified_Member	No_of_Earning_Members
44	85000	25000	5	0	1142400	Under-Graduate	2
24	35000	12000	3	0	466200	Graduate	1
9	20000	9000	4	0	220800	Under-Graduate	2
38	55000	45000	6	12000	600600	Graduate	2
43	80000	20000	4	0	1075200	Graduate	1
17	25000	10000	6	0	258000	Under-Graduate	3
14	24000	10000	4	0	244800	Graduate	2
19	30000	13000	4	0	385200	Graduate	1
37	50500	20000	3	0	581760	Professional	2
36	50000	20000	4	0	570000	Professional	1
46	98000	25000	5	0	1152480	Professional	2
39	60000	10000	3	0	590400	Post-Graduate	1
6	15000	16000	3	35000	167400	Post-Graduate	1
35	47000	15000	7	0	456840	Professional	4
48	100000	50000	4	20000	1032000	Professional	2

In [6]:
```
df.dtypes
```

Out[6]:
```
Mthly_HH_Income             int64
Mthly_HH_Expense            int64
No_of_Fly_Members           int64
Emi_or_Rent_Amt             int64
Annual_HH_Income            int64
Highest_Qualified_Member    object
No_of_Earning_Members       int64
dtype: object
```

In [7]:
```
df.columns
```

Out[7]:
```
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 [8]:
```
df.describe()
```

Out[8]:

	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

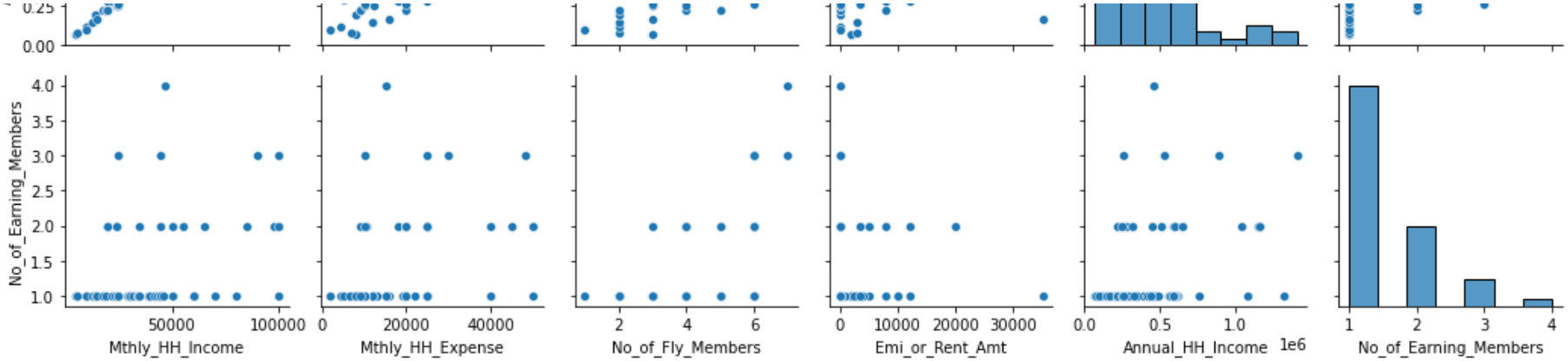
In [9]:
```
df.describe(include='all')
```

Out[9]:

	Mthly_HH_Income	Mthly_HH_Expense	No_of_Fly_Members	Emi_or_Rent_Amt	Annual_HH_Income	Highest_Qualified_Member	No_of_Earning_Men
count	50.000000	50.000000	50.000000	50.000000	5.000000e+01	50	50.00
unique	NaN	NaN	NaN	NaN	NaN	5	
top	NaN	NaN	NaN	NaN	NaN	Graduate	
freq	NaN	NaN	NaN	NaN	NaN	19	
mean	41558.000000	18818.000000	4.060000	3060.000000	4.900190e+05	NaN	1.46
std	26097.908979	12090.216824	1.517382	6241.434948	3.201358e+05	NaN	0.73
min	5000.000000	2000.000000	1.000000	0.000000	6.420000e+04	NaN	1.00
25%	23550.000000	10000.000000	3.000000	0.000000	2.587500e+05	NaN	1.00
50%	35000.000000	15500.000000	4.000000	0.000000	4.474200e+05	NaN	1.00
75%	50375.000000	25000.000000	5.000000	3500.000000	5.947200e+05	NaN	2.00
max	100000.000000	50000.000000	7.000000	35000.000000	1.404000e+06	NaN	4.00

In [10]:
```
sns.pairplot(df)
plt.show()
```


Mthly_HH_Income
Mthly_HH_Expense
No_of_Fly_Members
Emi_or_Rent_Amt
Annual_HH_Income
1e6



4.What is the Mean Expense of a Household?

In [11]:
```
df.Mthly_HH_Expense.mean()
```

Out[11]:
```
18818.0
```

In [12]:
```
df['Mthly_HH_Expense'].mean()
```

Out[12]:
```
18818.0
```

5.What is the Median Household Expense?

In [13]:
```
df.Mthly_HH_Expense.median()
```

Out[13]:
```
15500.0
```

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

In [14]:
```
df.Mthly_HH_Expense.mode()
```

Out[14]:
```
0    25000
Name: Mthly_HH_Expense, dtype: int64
```

In [15]:
```
df.Mthly_HH_Expense.value_counts()
```

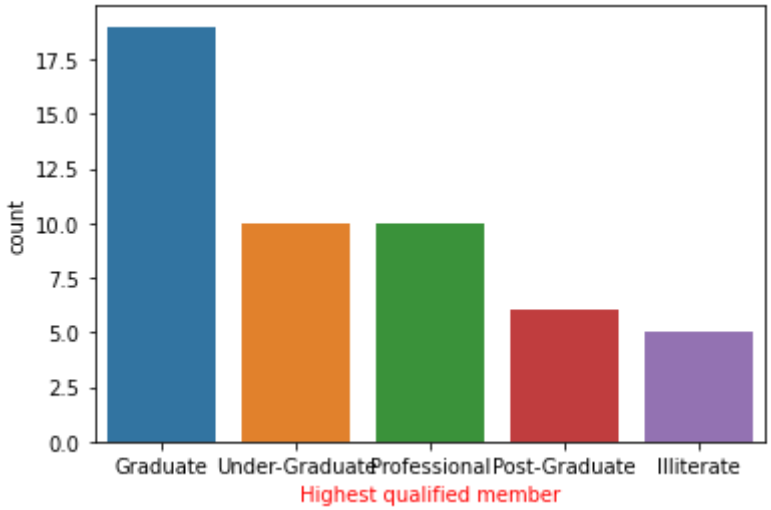
Out[15]:
```
25000    8
20000    6
10000    5
15000    3
9000     3
8000     3
12000    3
50000    2
40000    2
16000    1
48000    1
45000    1
22000    1
19000    1
4500     1
13000    1
6600     1
12300    1
7000     1
10500    1
5000     1
2000     1
18000    1
30000    1
Name: Mthly_HH_Expense, dtype: int64
```

7.Plot the Histogram to count the Highest qualified member

In [16]:
```
df.Highest_Qualified_Member.value_counts()
```

Out[16]:
```
Graduate          19
Under-Graduate    10
Professional      10
Post-Graduate      6
Illiterate         5
Name: Highest_Qualified_Member, dtype: int64
```

In [17]:
```
sns.countplot(x=df.Highest_Qualified_Member,order=df['Highest_Qualified_Member'].value_counts(ascending=False).index)
plt.xlabel("Highest qualified member",c='red')
plt.show()
```



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

In [18]:
```
df.describe(include='all')
```

Out[18]:

	Mthly_HH_Income	Mthly_HH_Expense	No_of_Fly_Members	Emi_or_Rent_Amt	Annual_HH_Income	Highest_Qualified_Member	No_of_Earning_Men
count	50.000000	50.000000	50.000000	50.000000	5.000000e+01	50	50.00
unique	NaN	NaN	NaN	NaN	NaN	5	
top	NaN	NaN	NaN	NaN	NaN	Graduate	
freq	NaN	NaN	NaN	NaN	NaN	19	
mean	41558.000000	18818.000000	4.060000	3060.000000	4.900190e+05	NaN	1.46
std	26097.908979	12090.216824	1.517382	6241.434948	3.201358e+05	NaN	0.73
min	5000.000000	2000.000000	1.000000	0.000000	6.420000e+04	NaN	1.00
25%	23550.000000	10000.000000	3.000000	0.000000	2.587500e+05	NaN	1.00
50%	35000.000000	15500.000000	4.000000	0.000000	4.474200e+05	NaN	1.00
75%	50375.000000	25000.000000	5.000000	3500.000000	5.947200e+05	NaN	2.00
max	100000.000000	50000.000000	7.000000	35000.000000	1.404000e+06	NaN	4.00

In [19]:
```
Q1=df.Mthly_HH_Income.quantile([.25])
Q1
```

Out[19]:
```
0.25    23550.0
Name: Mthly_HH_Income, dtype: float64
```

In [20]:
```
Q3=pd.DataFrame(df.Mthly_HH_Income.quantile([.75]))
Q3
```

Out[20]:

	Mthly_HH_Income
0.75	50375.0

In [21]:
```
#IQR Calculations for 'Mthly_HH_Income' column Only

quartiles = df['Mthly_HH_Income'].quantile([0.25, 0.75])
iqr = quartiles[0.75] - quartiles[0.25]
print(iqr)
```

```
26825.0
```

In [22]:
```
df.quantile(q=[0.25, 0.5, 0.75], axis=0, numeric_only=True)
```

Out[22]:

	Mthly_HH_Income	Mthly_HH_Expense	No_of_Fly_Members	Emi_or_Rent_Amt	Annual_HH_Income	No_of_Earning_Members
0.25	23550.0	10000.0	3.0	0.0	258750.0	1.0
0.50	35000.0	15500.0	4.0	0.0	447420.0	1.0
0.75	50375.0	25000.0	5.0	3500.0	594720.0	2.0

In [23]:
```
#IQR Calculations for all the numeric columns

quartiles=df.quantile(q=[0.25, 0.75], axis=0, numeric_only=True)
IQR_All_Numeric_Columns=quartiles.loc[0.75]-quartiles.loc[0.25]
IQR=pd.DataFrame(IQR_All_Numeric_Columns,columns=['IQR'])
IQR
```

Out[23]:

	IQR
Mthly_HH_Income	26825.0
Mthly_HH_Expense	15000.0
No_of_Fly_Members	2.0
Emi_or_Rent_Amt	3500.0
Annual_HH_Income	335970.0
No_of_Earning_Members	1.0

In [24]:
```
df.head()
```

Out[24]:

	Mthly_HH_Income	Mthly_HH_Expense	No_of_Fly_Members	Emi_or_Rent_Amt	Annual_HH_Income	Highest_Qualified_Member	No_of_Earning_Members
0	5000	8000	3	2000	64200	Under-Graduate	1
1	6000	7000	2	3000	79920	Illiterate	1
2	10000	4500	2	0	112800	Under-Graduate	1
3	10000	2000	1	0	97200	Illiterate	1
4	12500	12000	2	3000	147000	Graduate	1

In [25]:
```
df.Mthly_HH_Income.head()
```

Out[25]:
```
0     5000
1     6000
2    10000
3    10000
4    12500
Name: Mthly_HH_Income, dtype: int64
```

In [26]:
```
#Lower Outlier
Q1-1.5*IQR
```

Out[26]:

	IQR	0.25
Mthly_HH_Income	NaN	NaN
Mthly_HH_Expense	NaN	NaN
No_of_Fly_Members	NaN	NaN
Emi_or_Rent_Amt	NaN	NaN
Annual_HH_Income	NaN	NaN
No_of_Earning_Members	NaN	NaN

In [27]:
```
df[(df.Mthly_HH_Income<-16687.5)]['Mthly_HH_Income']
```

Out[27]:
```
Series([], Name: Mthly_HH_Income, dtype: int64)
```

In [28]:
```
#Upper Outlier
Q3+1.5*IQR
```

Out[28]:

	IQR	Mthly_HH_Income
0.75	NaN	NaN
Annual_HH_Income	NaN	NaN
Emi_or_Rent_Amt	NaN	NaN
Mthly_HH_Expense	NaN	NaN
Mthly_HH_Income	NaN	NaN
No_of_Earning_Members	NaN	NaN
No_of_Fly_Members	NaN	NaN

In [29]:
```
df[(df.Mthly_HH_Income>90612.5)]['Mthly_HH_Income']
```

Out[29]:
```
46     98000
47    100000
48    100000
49    100000
Name: Mthly_HH_Income, dtype: int64
```

9.Calculate Standard Deviation for first 4 columns.

In [30]:
```
df[['Mthly_HH_Income','Mthly_HH_Expense','No_of_Fly_Members','Emi_or_Rent_Amt']].std()
```

Out[30]:
```
Mthly_HH_Income      26097.908979
Mthly_HH_Expense     12090.216824
No_of_Fly_Members        1.517382
Emi_or_Rent_Amt       6241.434948
dtype: float64
```

10.Calculate Variance for first 3 columns.

In [31]:
```
df[['Mthly_HH_Income','Mthly_HH_Expense','No_of_Fly_Members']].var()
```

Out[31]:
```
Mthly_HH_Income      6.811009e+08
Mthly_HH_Expense     1.461733e+08
No_of_Fly_Members    2.302449e+00
dtype: float64
```

11.Calculate the count of Highest qualified member.

In [32]:
```
df.Highest_Qualified_Member.value_counts()
```

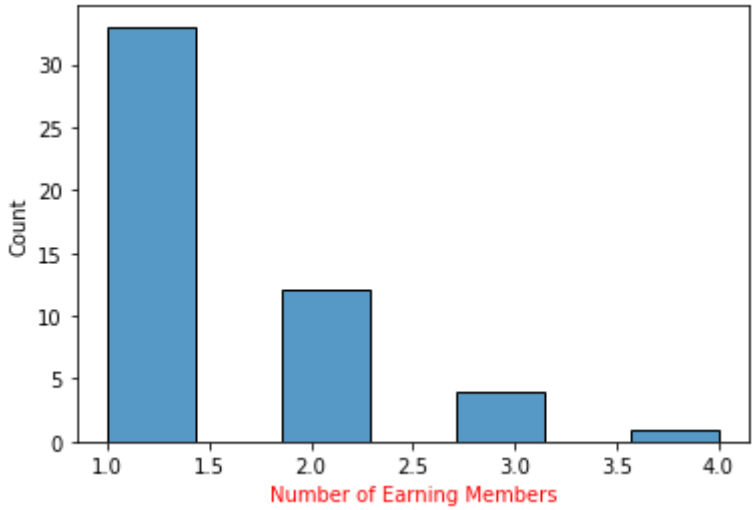
Out[32]:
```
Graduate          19
Under-Graduate    10
Professional      10
Post-Graduate      6
Illiterate         5
Name: Highest_Qualified_Member, dtype: int64
```

In [33]:
```
df.Highest_Qualified_Member.value_counts().head(1)
```

Out[33]:
```
Graduate    19
Name: Highest_Qualified_Member, dtype: int64
```

12.Plot the Histogram to count the No_of_Earning_Members

In [34]:
```
sns.histplot(x='No_of_Earning_Members',data=df)
plt.xlabel("Number of Earning Members",c='red')
plt.show()
```



In [35]:
```
df.No_of_Earning_Members.unique()
```

Out[35]:
```
array([1, 2, 3, 4], 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 is 10%. Standard Deviation of the returns of these stocks is 10% and 5% respectively.

Which is better investment?

I will invest in stock A because stock B is stable stock and might be less profit or loss though profilt is for sure in stock B, As it has less standard deviation.

B stock is Stable & A is volatile So A is a risky asset but it can be profitable as well if it goes to high So I will prefer A.