Importing Libraries

In [2]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

Loading The Dataset

In [6]:
```
df = pd.read_csv(r"C:\Users\AKSHAY\OneDrive\Desktop\Code\Projects\Project Codes\
```

In [8]:
```
df
```

Out[8]:

	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	

Analyze The Dataset

In [18]:
```
# We are checing if there are any missing values, the data type for each column
```

In [12]:
```
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 [20]:
```
# We are checking the no. of rows and columns ( Dimensions )
```

In [16]:
```
df.shape
```

Out[16]: (50, 7)

In [26]:
```
# We are finding the descriptive statistics of the dataset
```

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

Out[24]:

	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 [28]:
```
# We are interchanging the rows and columns of the above output
```

In [32]:
```
df.describe().transpose()
```

Out[32]:

	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 [34]:
```
# We are checking if there are any null values in the dataset
```

In [36]:
```
df.isna().any()
```

Out[36]:
```
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
```

Mean Household Expense

In [45]:
```
df['Mthly_HH_Expense'].mean()
```

Out[45]: 18818.0

Median Household Expense

In [53]:
```
df['Mthly_HH_Expense'].median()
```

Out[53]: 15500.0

Maximum Monthly Household Expense

-> We are first creating a frequency table, where we are grouping the unique values in the "Mthly_HH_Expense" column,

-> Then we are creating another column called 'count', in this we are storing how many times the values are repeating in the above column

-> Then we are checking the maximum number of times a value has repeated, and we are priniting that value

In [75]:
```
mth_exp_max = pd.crosstab(index = df['Mthly_HH_Expense'], columns = 'count')
mth_exp_max.reset_index(inplace = True)
mth_exp_max[mth_exp_max['count'] == df.Mthly_HH_Expense.value_counts().max()]
```

Out[75]:

	col_0	Mthly_HH_Expense	count
18		25000	8

Minimum Monthly Household Expense

-> We are first creating a frequency table, where we are grouping the unique values in the "Mthly_HH_Expense" column,

-> Then we are creating another column called 'count', in this we are storing how many times the values are repeating in the above column

-> Then we are checking the minimum number of times a value has repeated, and we are priniting that value

In [72]:
```
mth_exp_min = pd.crosstab(index = df['Mthly_HH_Expense'], columns = 'count')
mth_exp_min.reset_index(inplace = True)
mth_exp_min[mth_exp_min['count'] == df.Mthly_HH_Expense.value_counts().min()]
```

Out[72]:

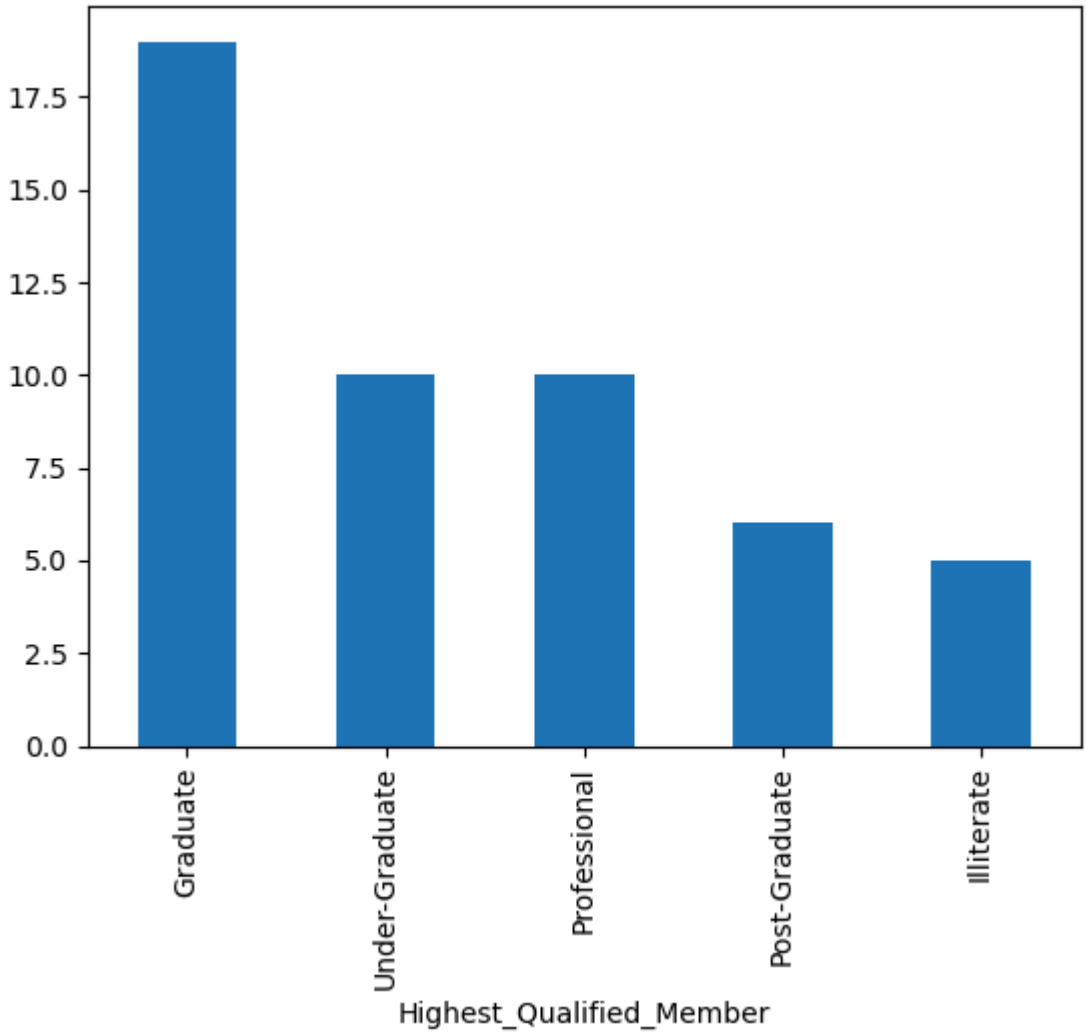
col_0	Mthly_HH_Expense	count
0	2000	1
1	4500	1
2	5000	1
3	6600	1
4	7000	1
8	10500	1
10	12300	1
11	13000	1
13	16000	1
14	18000	1
15	19000	1
17	22000	1
19	30000	1
21	45000	1
22	48000	1

Histogram Plot to Count the Highest Qualified Employee

In [78]:
```
# we are going to count the number of times a value has repeated, and accordning
```

In [81]:
```
df['Highest_Qualified_Member'].value_counts().plot(kind = 'bar')
```

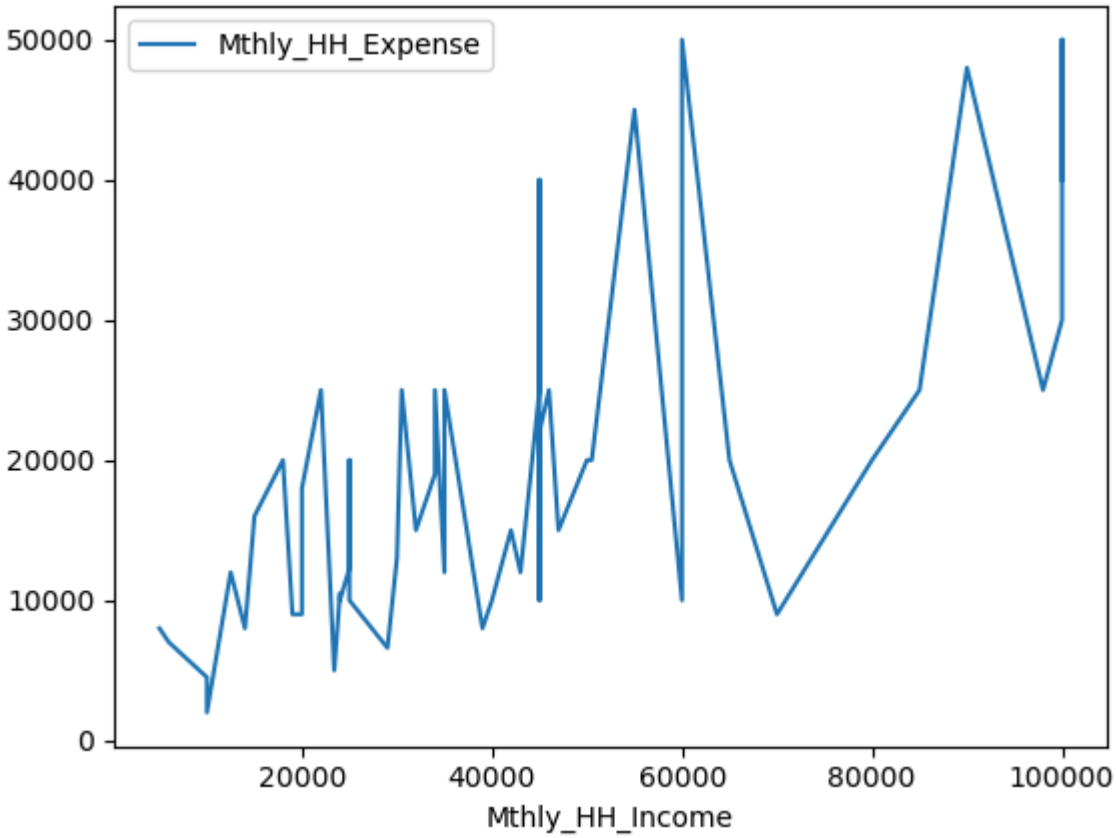
Out[81]:
```
<Axes: xlabel='Highest_Qualified_Member'>
```



Calculating the IQR

In [98]:
```
df.plot(x = 'Mthly_HH_Income', y = 'Mthly_HH_Expense')
IQR = df['Mthly_HH_Expense'].quantile(0.75) - df['Mthly_HH_Expense'].quantile(0.
print("The 50% IQR is:",IQR)
```

```
The 50% IQR is: 15000.0
```



Calculating the Standard Deviation (for first 4 columns)

-> We are first creating a data frame

-> We are then indexing the values using iloc[] function, this is used for integer-based indexing

-> Then we are finding the Standard Deviation of the columns

In [151…
```
pd.DataFrame(df.iloc[:,0:5].std().to_frame())
```

Out[151…

	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 [111…
```
# Transposing the above output
```

In [113…
```
pd.DataFrame(df.iloc[:,0:5].std().to_frame()).transpose()
```

Out[113...

	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

Calculating the Variance (for first 3 columns)

-> We are first creating a data frame

-> We are then indexing the values using iloc[] function, this is used for integer-based indexing

-> Then we are finding the Variance of the columns

In [122...
```
pd.DataFrame(df.iloc[:, 0:4].var().to_frame())
```

Out[122...

	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 [124...
```
# Transposing the above output
```

In [126...
```
pd.DataFrame(df.iloc[:, 0:4].var().to_frame()).transpose()
```

Out[126...

	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

Calculating the Count of Highest Qualified Member (column)

In [139...
```
# We are using the value_counts() function, but just using the to_frame() functi
# to add it to a dataset form
```

In [154...
```
df['Highest_Qualified_Member'].value_counts().to_frame().transpose()
```

Out[154...

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

Histogram Plot to Count the No_of_Earnings_Members (column)

In [148…
```
# we are going to count the number of times a value has repeated, and accordning
```

In [145…
```
df['No_of_Earning_Members'].value_counts().plot(kind = 'bar')
```

Out[145… `<Axes: xlabel='No_of_Earning_Members'>`