Importing the Libraries

In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
```

Import the dataset

In [2]:

```
dataset=pd.read_csv("C:/Users/navna/Downloads/Data_lab_exam.csv")
```

In [3]:
```
dataset
```

	features	observ_features	price_per_square_foot
0	0.44	0.68	511.14
1	0.99	0.23	717.10
2	0.84	0.29	607.91
3	0.28	0.45	270.40
4	0.07	0.83	289.88
...	...	...	...
95	0.99	0.13	636.22
96	0.28	0.46	272.12
97	0.87	0.36	696.65
98	0.23	0.87	434.53
99	0.77	0.36	593.86

100 rows × 3 columns

In [4]:
```
dataset.shape
```

```
(100, 3)
```

In [5]:

```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 3 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   features              100 non-null    float64
 1   observ_features       100 non-null    float64
 2   price_per_square_foot 100 non-null    float64
dtypes: float64(3)
memory usage: 2.5 KB
```

In [6]:

```
dataset.isna()
```

	features	observ_features	price_per_square_foot
0	False	False	False
1	False	False	False
2	False	False	False
3	False	False	False
4	False	False	False
...	...	...	...
95	False	False	False
96	False	False	False
97	False	False	False
98	False	False	False
99	False	False	False

100 rows × 3 columns

In [7]:
```
dataset.isna().sum()
```

```
features                0
observ_features         0
price_per_square_foot   0
dtype: int64
```

In [8]:
```
print(type(dataset))
```

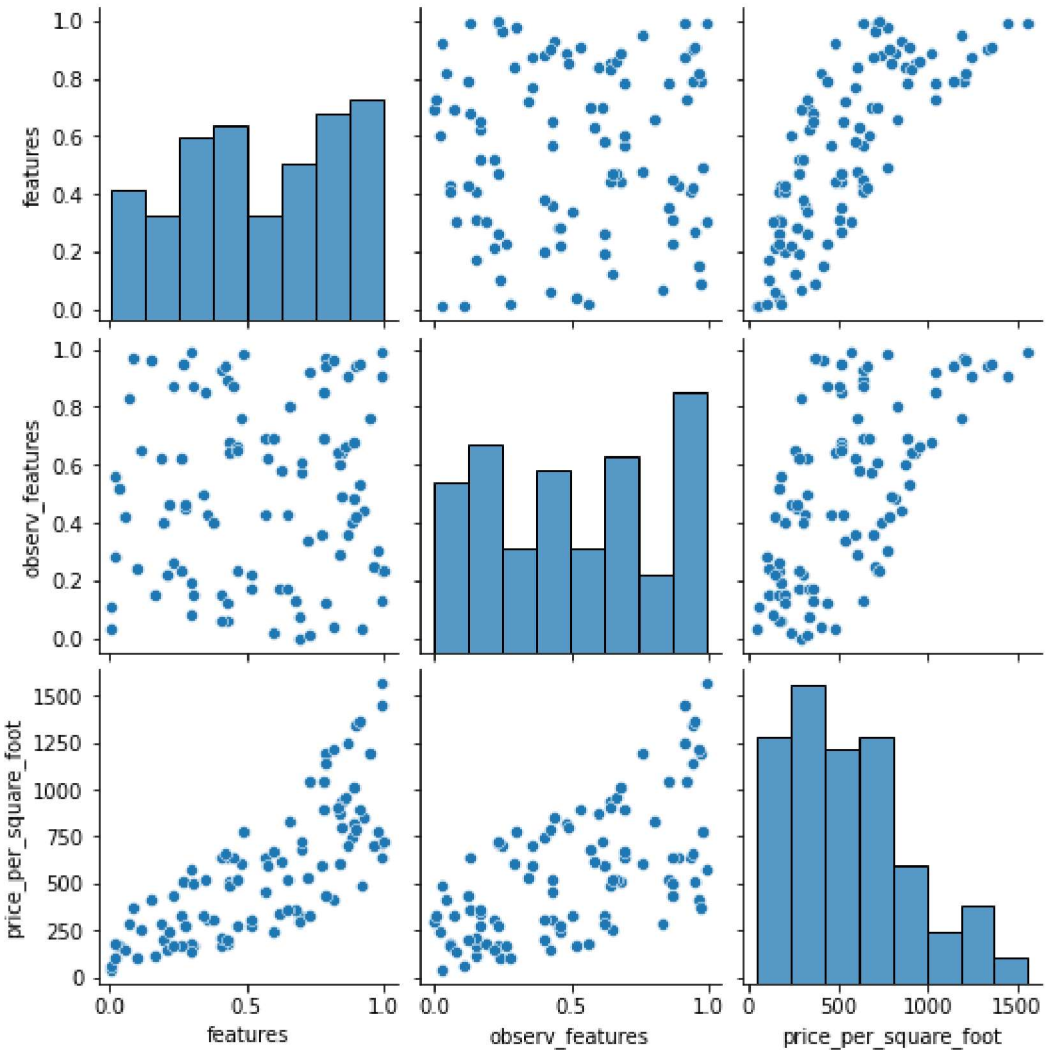
```
<class 'pandas.core.frame.DataFrame'>
```

In [52]:
```
dataset.describe()
```

	features	observ_features	price_per_square_foot
count	100.000000	100.000000	100.000000
mean	0.550300	0.501700	554.214600
std	0.293841	0.307124	347.312796
min	0.010000	0.000000	42.080000
25%	0.300000	0.230000	278.172500
50%	0.570000	0.485000	514.285000
75%	0.822500	0.760000	751.752500
max	1.000000	0.990000	1563.820000

In [9]:
```
sns.pairplot(dataset)
```

<seaborn.axisgrid.PairGrid at 0x2516e6b9370>



Conclusion:-

As the number of features and its respective observed_features are incresed then the price is also increases respectively.

In [10]:

```
# Independent Variable
X=dataset.iloc[:,:-1].values

# Dependent Variable
y=dataset.iloc[:,-1].values
```

In [11]:

```
#Check the values of Independent Variable
print(X)
```

```
[[0.44 0.68]
 [0.99 0.23]
 [0.84 0.29]
 [0.28 0.45]
 [0.07 0.83]
 [0.66 0.8 ]
 [0.73 0.92]
 [0.57 0.43]
 [0.43 0.89]
 [0.27 0.95]
 [0.43 0.06]
 [0.87 0.91]
 [0.78 0.69]
 [0.9  0.94]
 [0.41 0.06]
 [0.52 0.17]
 [0.47 0.66]
 [0.65 0.43]
 [0.85 0.64]
 [0.93 0.44]
 [0.41 0.93]
 [0.36 0.43]
 [0.78 0.85]
 [0.69 0.07]
 [0.04 0.52]
 [0.17 0.15]
 [0.68 0.13]
 [0.84 0.6 ]
 [0.38 0.4 ]
 [0.12 0.65]
 [0.62 0.17]
 [0.79 0.97]
 [0.82 0.04]
 [0.91 0.53]
 [0.35 0.85]
 [0.57 0.69]
```

```
[0.52 0.22]
[0.31 0.15]
[0.6  0.02]
[0.99 0.91]
[0.48 0.76]
[0.3  0.19]
[0.58 0.62]
[0.65 0.17]
[0.6  0.69]
[0.95 0.76]
[0.47 0.23]
[0.15 0.96]
[0.01 0.03]
[0.26 0.23]
[0.01 0.11]
[0.45 0.87]
[0.09 0.97]
[0.96 0.25]
[0.63 0.58]
[0.06 0.42]
[0.1  0.24]
[0.26 0.62]
[0.41 0.15]
[0.91 0.95]
[0.83 0.64]
[0.44 0.64]
[0.2  0.4 ]
[0.43 0.12]
[0.21 0.22]
[0.88 0.4 ]
[0.31 0.87]
[0.99 0.99]
[0.23 0.26]
[0.79 0.12]
[0.02 0.28]
[0.89 0.48]
[0.02 0.56]
[0.92 0.03]
[0.72 0.34]
[0.3  0.99]
[0.86 0.66]
```

```
 [0.47 0.65]
 [0.79 0.94]
 [0.82 0.96]
 [0.9  0.42]
 [0.19 0.62]
 [0.7  0.57]
 [0.7  0.61]
 [0.69 0.  ]
 [0.98 0.3 ]
 [0.3  0.08]
 [0.85 0.49]
 [0.73 0.01]
 [1.   0.23]
 [0.42 0.94]
 [0.49 0.98]
 [0.89 0.68]
 [0.22 0.46]
 [0.34 0.5 ]
 [0.99 0.13]
 [0.28 0.46]
 [0.87 0.36]
 [0.23 0.87]
 [0.77 0.36]]
```

In [12]:

```
#Check the values of dependent Variable
print(y)
```

```
[ 511.14  717.1   607.91  270.4   289.88  830.85 1038.09  455.19  640.17
  511.06  177.03 1242.52  891.37 1339.72  169.88  276.05  517.43  522.25
  932.21  851.25  640.11  308.68 1046.05  332.4   171.85  109.55  361.97
  872.21  303.7   256.38  341.2  1194.63  408.6   895.54  518.25  638.75
  301.9   163.38  240.77 1449.05  609.    174.59  593.45  355.96  671.46
 1193.7   278.88  411.4    42.08  166.19   58.62  642.45  368.14  702.78
  615.74  143.79  109.    328.28  205.16 1360.49  905.83  487.33  202.76
  202.01  148.87  745.3   503.04 1563.82  165.21  438.4    98.47  819.63
  174.44  483.13  534.24  572.31  957.61  518.29 1143.49 1211.31  784.74
  283.7   684.38  719.46  292.23  775.68  130.77  801.6   323.55  726.9
  661.12  771.11 1016.14  237.69  325.89  636.22  272.12  696.65  434.53
  593.86]
```

Splitting the dataset(Training and Testing)

In [13]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.25, random_state=28)
```

In [14]:
```
# Check the values of X_train data
X_train
```

```
array([[0.3 , 0.99],
       [0.47, 0.65],
       [0.42, 0.94],
       [0.27, 0.95],
       [0.31, 0.87],
       [0.19, 0.62],
       [0.41, 0.93],
       [0.89, 0.68],
       [0.01, 0.03],
       [0.86, 0.66],
       [0.06, 0.42],
       [0.26, 0.23],
       [0.87, 0.91],
       [0.93, 0.44],
       [0.34, 0.5 ],
       [0.63, 0.58],
       [0.88, 0.4 ],
       [0.96, 0.25],
       [0.3 , 0.19],
       [0.98, 0.3 ],
       [0.23, 0.87],
       [0.79, 0.97],
       [0.7 , 0.57],
       [0.84, 0.6 ],
       [0.6 , 0.02],
       [0.47, 0.66],
       [0.1 , 0.24],
       [0.87, 0.36],
       [0.89, 0.48],
       [0.15, 0.96],
       [0.17, 0.15],
       [0.84, 0.29],
       [0.91, 0.95],
       [0.9 , 0.42],
       [0.44, 0.64],
       [0.65, 0.43],
```

```
       [0.43, 0.06],
       [0.43, 0.12],
       [0.7 , 0.61],
       [0.41, 0.15],
       [0.26, 0.62],
       [0.73, 0.01],
       [0.28, 0.46],
       [0.6 , 0.69],
       [1.  , 0.23],
       [0.69, 0.  ],
       [0.07, 0.83],
       [0.68, 0.13],
       [0.52, 0.17],
       [0.38, 0.4 ],
       [0.23, 0.26],
       [0.02, 0.28],
       [0.31, 0.15],
       [0.49, 0.98],
       [0.35, 0.85],
       [0.83, 0.64],
       [0.95, 0.76],
       [0.79, 0.94],
       [0.77, 0.36],
       [0.57, 0.43],
       [0.65, 0.17],
       [0.04, 0.52],
       [0.43, 0.89],
       [0.79, 0.12],
       [0.22, 0.46],
       [0.69, 0.07],
       [0.45, 0.87],
       [0.21, 0.22],
       [0.78, 0.69],
       [0.85, 0.49],
       [0.28, 0.45],
       [0.82, 0.04],
       [0.78, 0.85],
       [0.66, 0.8 ],
       [0.99, 0.23]])
```

In [15]:

```
# Check the values of X_train data
y_train
```

```
array([ 572.31,  518.29,  661.12,  511.06,  503.04,  283.7 ,  640.11,
       1016.14,   42.08,  957.61,  143.79,  166.19, 1242.52,  851.25,
        325.89,  615.74,  745.3 ,  702.78,  174.59,  775.68,  434.53,
       1194.63,  684.38,  872.21,  240.77,  517.43,  109.  ,  696.65,
        819.63,  411.4 ,  109.55,  607.91, 1360.49,  784.74,  487.33,
        522.25,  177.03,  202.01,  719.46,  205.16,  328.28,  323.55,
        272.12,  671.46,  726.9 ,  292.23,  289.88,  361.97,  276.05,
        303.7 ,  165.21,   98.47,  163.38,  771.11,  518.25,  905.83,
       1193.7 , 1143.49,  593.86,  455.19,  355.96,  171.85,  640.17,
        438.4 ,  237.69,  332.4 ,  642.45,  148.87,  891.37,  801.6 ,
        270.4 ,  408.6 , 1046.05,  830.85,  717.1 ])
```

In [16]:

```
y_train
```


```
array([ 572.31,  518.29,  661.12,  511.06,  503.04,  283.7 ,  640.11,
       1016.14,   42.08,  957.61,  143.79,  166.19, 1242.52,  851.25,
        325.89,  615.74,  745.3 ,  702.78,  174.59,  775.68,  434.53,
       1194.63,  684.38,  872.21,  240.77,  517.43,  109.  ,  696.65,
        819.63,  411.4 ,  109.55,  607.91, 1360.49,  784.74,  487.33,
        522.25,  177.03,  202.01,  719.46,  205.16,  328.28,  323.55,
        272.12,  671.46,  726.9 ,  292.23,  289.88,  361.97,  276.05,
        303.7 ,  165.21,   98.47,  163.38,  771.11,  518.25,  905.83,
       1193.7 , 1143.49,  593.86,  455.19,  355.96,  171.85,  640.17,
        438.4 ,  237.69,  332.4 ,  642.45,  148.87,  891.37,  801.6 ,
        270.4 ,  408.6 , 1046.05,  830.85,  717.1 ])
```

In [17]:

```
y_test
```

```
array([1563.82,  169.88,  511.14,  130.77, 1339.72,  932.21,  341.2 ,
        301.9 ,  368.14,  636.22, 1449.05,  483.13,  895.54,  609.  ,
        534.24,  638.75,  174.44,  256.38, 1038.09,  308.68,  593.45,
       1211.31,   58.62,  278.88,  202.76])
```

Splitting the dataset

In [18]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.25, random_state=28)
```

In [19]:

```
from sklearn.naive_bayes import GaussianNB
from sklearn import metrics
from sklearn.model_selection import train_test_split
```

In [54]:

```
### Training the Simple Regression Model
from sklearn.linear_model import LinearRegression
reg=LinearRegression
reg.fit(X_train, y_train)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Input In [54], in <cell line: 4>()
      2 from sklearn.linear_model import LinearRegression
      3 reg=LinearRegression
----> 4 reg.fit(X_train, y_train)

TypeError: fit() missing 1 required positional argument: 'y'
```

Problem Statement 2

In [21]:

```
data=pd.read_csv("C:/Users/navna/Downloads/Data_cust_problem.csv")
```

In [22]:
```
data
```

	cust_id	age	income	gender	marital_status	buys
0	1	18	High	male	single	no
1	2	19	High	male	marrid	no
2	3	28	High	male	single	yes
3	4	38	Medium	male	single	yes
4	5	36	Low	female	single	yes
5	6	40	Low	female	marrid	no
6	7	28	Low	female	marrid	yes
7	8	18	Medium	male	single	no
8	9	19	Low	female	marrid	yes
9	10	37	Medium	female	single	yes
10	11	18	Medium	female	marrid	yes
11	12	28	Medium	male	marrid	yes
12	13	28	High	female	single	yes
13	14	39	Medium	male	marrid	no

In [23]:
```
data.shape
```

```
(14, 6)
```

In [24]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14 entries, 0 to 13
Data columns (total 6 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   cust_id         14 non-null     int64
 1   age             14 non-null     int64
 2   income          14 non-null     object
 3   gender          14 non-null     object
 4   marital_status  14 non-null     object
 5   buys            14 non-null     object
dtypes: int64(2), object(4)
memory usage: 800.0+ bytes
```

In [25]:

```
data.isna()
```

	cust_id	age	income	gender	marital_status	buys
0	False	False	False	False	False	False
1	False	False	False	False	False	False
2	False	False	False	False	False	False
3	False	False	False	False	False	False
4	False	False	False	False	False	False
5	False	False	False	False	False	False
6	False	False	False	False	False	False
7	False	False	False	False	False	False
8	False	False	False	False	False	False
9	False	False	False	False	False	False
10	False	False	False	False	False	False
11	False	False	False	False	False	False
12	False	False	False	False	False	False
13	False	False	False	False	False	False

In [26]:

```
data.isna().sum()
```

```
cust_id           0
age               0
income            0
gender            0
marital_status    0
buys              0
dtype: int64
```

In [27]:

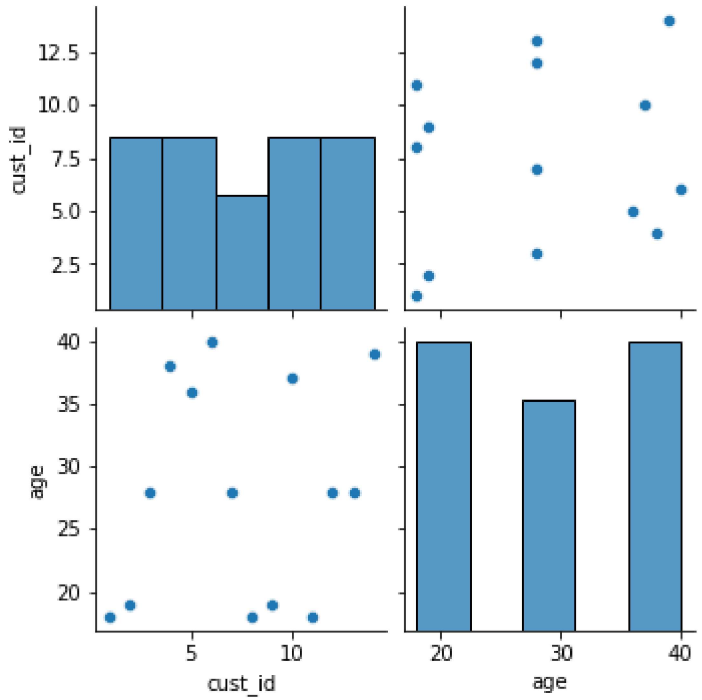
```
print(type(data))
```

```
<class 'pandas.core.frame.DataFrame'>
```

In [28]:

```
sns.pairplot(data)
```

```
<seaborn.axisgrid.PairGrid at 0x2516967abe0>
```



In [29]:
```
data.corr(method = 'pearson')
```

	cust_id	age
cust_id	1.000000	0.174403
age	0.174403	1.000000

In [30]:
```
data.income.count()
```

```
14
```

In [31]:
```
data
```

	cust_id	age	income	gender	marital_status	buys
0	1	18	High	male	single	no
1	2	19	High	male	marrid	no
2	3	28	High	male	single	yes
3	4	38	Medium	male	single	yes
4	5	36	Low	female	single	yes
5	6	40	Low	female	marrid	no
6	7	28	Low	female	marrid	yes
7	8	18	Medium	male	single	no
8	9	19	Low	female	marrid	yes
9	10	37	Medium	female	single	yes
10	11	18	Medium	female	marrid	yes
11	12	28	Medium	male	marrid	yes
12	13	28	High	female	single	yes
13	14	39	Medium	male	marrid	no

In [32]:
```
data.columns
```

```
Index(['cust_id', 'age', 'income', 'gender ', 'marital_status', 'buys'], dtype='object')
```

In [33]:
```
data.isna().any()
```

```
cust_id           False
age               False
income            False
gender            False
marital_status    False
buys              False
dtype: bool
```

In [34]:
```
data['income'].unique()
```

```
array(['High', 'Medium', 'Low'], dtype=object)
```

In [37]:
```
data['marital_status'].unique()
```

```
array(['single', 'marrid'], dtype=object)
```

In [38]:
```
data['buys'].unique()
```

```
array(['no ', 'yes'], dtype=object)
```

In [39]:

```
data['income'].value_counts()
```

```
Medium    6
High      4
Low       4
Name: income, dtype: int64
```

In [40]:

```
data['marital_status'].value_counts()
```

```
single    7
marrid    7
Name: marital_status, dtype: int64
```

In [41]:

```
data['buys'].value_counts()
```

```
yes    9
no     5
Name: buys, dtype: int64
```

In [43]:
```
data['gender'].value_counts()
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
File C:\Anaconda3\lib\site-packages\pandas\core\indexes\base.py:3621, in Index.get_loc(self, key, method, tolerance)
   3620 try:
-> 3621     return self._engine.get_loc(casted_key)
   3622 except KeyError as err:

File C:\Anaconda3\lib\site-packages\pandas\_libs\index.pyx:136, in pandas._libs.index.IndexEngine.get_loc()

File C:\Anaconda3\lib\site-packages\pandas\_libs\index.pyx:163, in pandas._libs.index.IndexEngine.get_loc()

File pandas\_libs\hashtable_class_helper.pxi:5198, in pandas._libs.hashtable.PyObjectHashTable.get_item()

File pandas\_libs\hashtable_class_helper.pxi:5206, in pandas._libs.hashtable.PyObjectHashTable.get_item()

KeyError: 'gender'

The above exception was the direct cause of the following exception:

KeyError                                  Traceback (most recent call last)
Input In [43], in <cell line: 1>()
----> 1 data['gender'].value_counts()

File C:\Anaconda3\lib\site-packages\pandas\core\frame.py:3505, in DataFrame.__getitem__(self, key)
   3503 if self.columns.nlevels > 1:
   3504     return self._getitem_multilevel(key)
-> 3505 indexer = self.columns.get_loc(key)
   3506 if is_integer(indexer):
   3507     indexer = [indexer]

File C:\Anaconda3\lib\site-packages\pandas\core\indexes\base.py:3623, in Index.get_loc(self, key, method, tolerance)
   3621     return self._engine.get_loc(casted_key)
   3622 except KeyError as err:
-> 3623     raise KeyError(key) from err
   3624 except TypeError:
   3625     # If we have a listlike key, _check_indexing_error will raise
   3626     #  InvalidIndexError. Otherwise we fall through and re-raise
   3627     #  the TypeError.
```

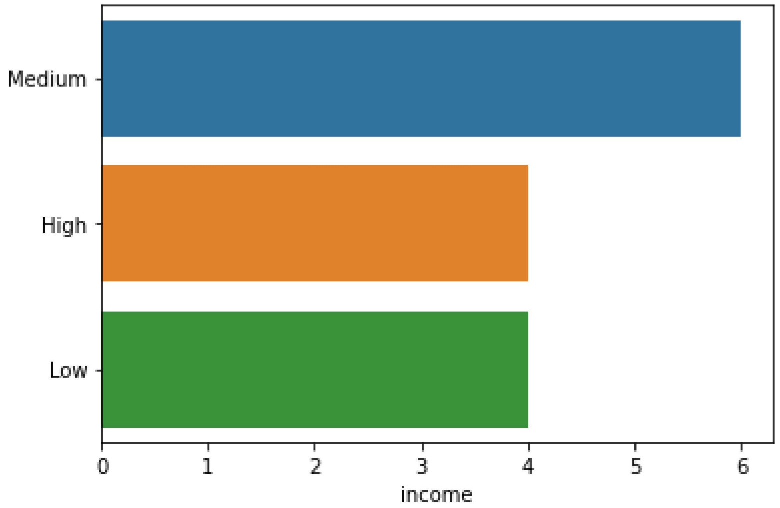
```
   3628     self._check_indexing_error(key)

KeyError: 'gender'
```

In [46]:

```
income_class=data.income.value_counts()
sns.barplot(y=income_class.index, x=income_class)
```

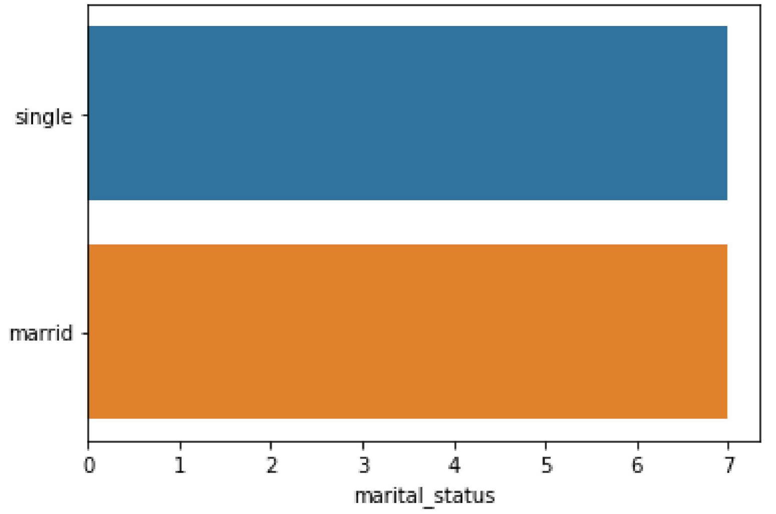
```
<AxesSubplot:xlabel='income'>
```



In [47]:

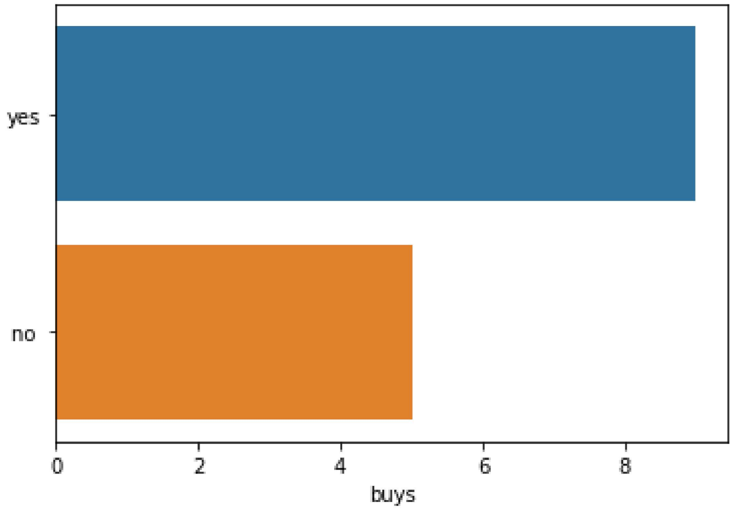
```
marital_status=data.marital_status.value_counts()
sns.barplot(y=marital_status.index, x=marital_status)
```

```
<AxesSubplot:xlabel='marital_status'>
```



In [48]:
```
buys=data.buys.value_counts()
sns.barplot(y=buys.index, x=buys)
```

```
<AxesSubplot:xlabel='buys'>
```



In [55]:
```
### What is the probability of buy
probability_of_buy = data['buys']
probability_of_buy.groupby(probability_of_buy).size()
```

```
buys
no     5
yes    9
Name: buys, dtype: int64
```

In [63]:
```
total_cust=data.buys.count()
total_cust
```

```
14
```

In [64]:
```
total_buy=9
total_buy
```

```
9
```

In [66]:
```
percent_buy=(total_buy/total_cust)*100
percent_buy
```

```
64.28571428571429
```

In [ ]: