In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
#import matplotlib.pyplot as plt
import scipy.stats as stats
#import statsmodels.api as sm
#import statsmodels.stats.api as sms
#from statsmodels.stats.proportion import proportion_confint
#import pylab
#import warnings
#warnings.simplefilter(action='ignore', category=FutureWarning)
```

In [2]:
```
temp = open('videodata-wgxmwm-x521b2.txt').read().splitlines()
templist = []
for i in temp:
    templist += [i.split(' ')]
data_1_raw = []
for i in templist:
    lst = []
    for j in i:
        if j != '':
            lst += [j]
    data_1_raw += [lst]
    lst = []
```

In [3]:
```
data_1 = pd.DataFrame(data_1_raw[1:], columns = data_1_raw[0])
data_1 = data_1.applymap(lambda x: pd.to_numeric(x))
```

In [4]:
```
data_1.head()
```

Out[4]:

	time	like	where	freq	busy	educ	sex	age	home	math	work	own	cdrom	email	grad
0	2.0	3.0	3.0	2.0	0.0	1.0	0.0	19.0	1.0	0.0	10.0	1.0	0.0	1.0	4.
1	0.0	3.0	3.0	3.0	0.0	0.0	0.0	18.0	1.0	1.0	0.0	1.0	1.0	1.0	2.
2	0.0	3.0	1.0	3.0	0.0	0.0	1.0	19.0	1.0	0.0	0.0	1.0	0.0	1.0	3.
3	0.5	3.0	3.0	3.0	0.0	1.0	0.0	19.0	1.0	0.0	0.0	1.0	0.0	1.0	3.
4	0.0	3.0	3.0	4.0	0.0	1.0	0.0	19.0	1.0	1.0	0.0	0.0	0.0	1.0	3.

In [5]:
```
data_2 = pd.read_csv('data_2.csv')
```

In [6]:
```
data_2.T
```

master	1	0	0	0	1	1	0	0	0	0	...	0	0	0	0	0	0
bored	0	1	0	0	0	1	0	1	0	0	...	0	0	1	0	0	0
other							Brainless		like sports		...						competiveness
graphic	0	0	0	0	0	0	0	1	1	0	...	0	1	1	0	1	0
time	1	1	0	1	0	1	0	0	0	0	...	0	1	0	0	1	0
frust	0	1	0	0	0	1	0	0	0	0	...	1	0	0	1	0	0
lonely	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0
rules	0	0	0	0	1	0	1	0	0	1	...	0	0	0	0	0	1
cost	1	0	1	0	1	1	0	0	1	0	...	1	1	1	0	1	0
boring	0	0	0	0	0	0	0	0	1	0	...	0	0	0	1	0	0
friends	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0
point	1	0	0	0	0	0	0	0	0	0	...	0	1	0	1	0	1
other2							too		unproductive								

In [7]:
```
data_1.describe()
```

Out[7]:

	time	like	where	freq	busy	educ	sex	age
count	91.000000	91.000000	91.000000	91.000000	91.000000	91.000000	91.000000	91.000000
mean	1.242857	4.076923	21.967033	16.461538	12.153846	14.549451	0.582418	19.516484
std	3.777040	10.098659	38.476097	33.896020	32.384126	34.670918	0.495893	1.846093
min	0.000000	1.000000	1.000000	1.000000	0.000000	0.000000	0.000000	18.000000
25%	0.000000	2.000000	3.000000	2.000000	0.000000	0.000000	0.000000	19.000000
50%	0.000000	3.000000	3.000000	3.000000	0.000000	1.000000	1.000000	19.000000
75%	1.250000	3.000000	5.000000	4.000000	1.000000	1.000000	1.000000	20.000000
max	30.000000	99.000000	99.000000	99.000000	99.000000	99.000000	1.000000	33.000000

In [8]:
```
data_2.describe()
```

Out[8]:

	Unnamed: 0	action	adv	sim	sport	strategy	relax	coord
count	91.00000	87.000000	87.000000	87.000000	87.000000	87.000000	87.000000	87.000000
mean	46.00000	0.517241	0.287356	0.172414	0.390805	0.632184	0.666667	0.045977
std	26.41338	0.502599	0.455153	0.379930	0.490759	0.485006	0.474137	0.210649
min	1.00000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	23.50000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
50%	46.00000	1.000000	0.000000	0.000000	0.000000	1.000000	1.000000	0.000000
75%	68.50000	1.000000	1.000000	0.000000	1.000000	1.000000	1.000000	0.000000
max	91.00000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

In [9]:
```
#Bootstrap of data_1
sample_pool = []
copy = data_1.copy()
for i in range(1000):
    sample = copy.sample(replace=True, n = len(copy))
    sample_pool += [sample]
```

Scenario 1

In [10]:
```
# The fraction of students who played a video games in the week prior to
```

In [11]:
```
#Point estimation
```

In [12]:
```
data_1['time'] = data_1['time'].replace(99, 0).replace(np.NaN, 0)
```

In [13]:
```
num_notplayed = len(data_1[data_1['time']!=0])
num_total = len(data_1)
fraction_point_estimation = num_notplayed / num_total
fraction_point_estimation
```

Out[13]: 0.3695652173913043

In [14]:
```
# Interval estimation w/ calculation
# N = 314, n = 91
```

In [15]:
```
N,n = 314, 91
```

In [16]:
```
x_bar = fraction_point_estimation

se = np.sqrt((x_bar*(1-x_bar) / (n-1)) * ((N - n)/N))

x_bar - 2*se, x_bar + 2*se
```

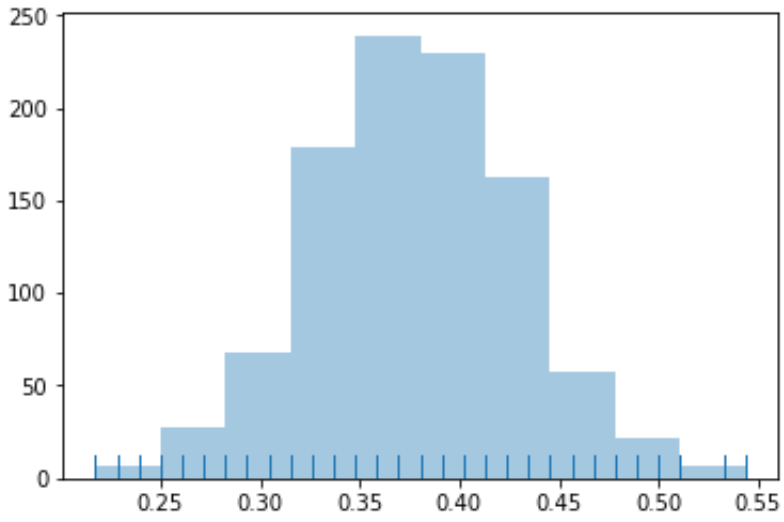
Out[16]: (0.283809718784465, 0.45532071599814367)

In [17]:
```
# Interval estimation w/ bootstrap simulation
```

In [18]:
```
fraction_pool = []
for i in sample_pool:
    i['time'] = i['time'].replace(99, 0).replace(np.NaN, 0)
    num_notplayed = len(i[i['time']!=0])
    num_total = len(i)
    fraction_point_estimation = num_notplayed / num_total
    fraction_pool += [fraction_point_estimation]
ci_low, ci_upp = np.percentile(fraction_pool, 2.5), np.percentile(fractio
```

In [19]:
```
sns.distplot(fraction_pool, bins=10, kde=False, rug=True )
```

Out[19]: <matplotlib.axes._subplots.AxesSubplot at 0x1a08c37518>



In [20]:
```
ci_low, ci_upp
```

Out[20]: (0.2717391304347826, 0.4782608695652174)

Scenario 2

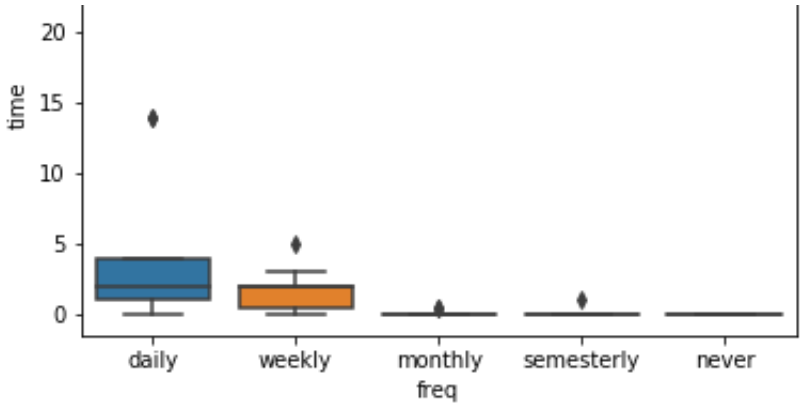
In [21]:
```
# Frequency of play & the fact of exam affect the frequency
```

In [22]:
```
data_1['freq'] = data_1['freq'].replace(np.NaN, 99)
```

In [23]:
```
grouped_df = pd.DataFrame(
             {
                 'time' : data_1[data_1['freq'] == 1]['time'],
                 'freq': ['daily' for i in data_1[data_1['freq'] == 1]['ti
             }
)
grouped_df1 = pd.DataFrame(
             {
                 'time' : data_1[data_1['freq'] == 2]['time'],
                 'freq': ['weekly' for i in data_1[data_1['freq'] == 2]['t
             }
)
grouped_df2 = pd.DataFrame(
             {
                 'time' : data_1[data_1['freq'] == 3]['time'],
                 'freq': ['monthly' for i in data_1[data_1['freq'] == 3]['
             }
)
grouped_df3 = pd.DataFrame(
             {
                 'time' : data_1[data_1['freq'] == 4]['time'],
                 'freq': ['semesterly' for i in data_1[data_1['freq'] == 4
             }
)
grouped_df4 = pd.DataFrame(
             {
                 'time' : data_1[data_1['freq'] == 99]['time'],
                 'freq': ['never' for i in data_1[data_1['freq'] == 99]['t
             }
)
cdf = pd.concat([grouped_df, grouped_df1, grouped_df2, grouped_df3, group
sns.boxplot(x ='freq', y ='time', data = cdf)

#box plot with outliers.
```

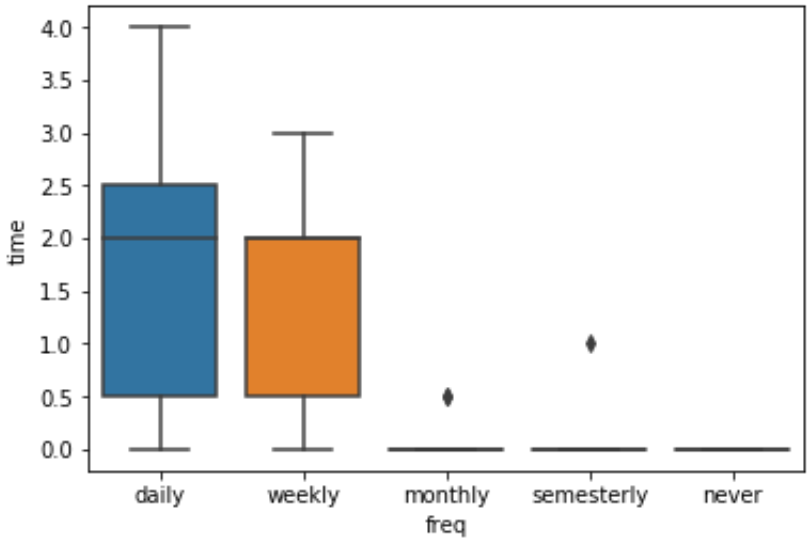
Out[23]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1a11b6dc50>
```



In [24]:
```
grouped_df = pd.DataFrame(
             {
                 'time' : data_1[(data_1['freq'] == 1) & (data_1['time'] <
                 'freq': ['daily' for i in data_1[(data_1['freq'] == 1) &
             }
)
grouped_df1 = pd.DataFrame(
             {
                 'time' : data_1[(data_1['freq'] == 2) & (data_1['time'] <
                 'freq': ['weekly' for i in data_1[(data_1['freq'] == 2) &
             }
)
grouped_df2 = pd.DataFrame(
             {
                 'time' : data_1[data_1['freq'] == 3]['time'],
                 'freq': ['monthly' for i in data_1[data_1['freq'] == 3]['
             }
)
grouped_df3 = pd.DataFrame(
             {
                 'time' : data_1[data_1['freq'] == 4]['time'],
                 'freq': ['semesterly' for i in data_1[data_1['freq'] == 4
             }
)
grouped_df4 = pd.DataFrame(
             {
                 'time' : data_1[data_1['freq'] == 99]['time'],
                 'freq': ['never' for i in data_1[data_1['freq'] == 99]['t
             }
)
cdf = pd.concat([grouped_df, grouped_df1, grouped_df2, grouped_df3, group
sns.boxplot(x ='freq', y ='time', data = cdf)

#box plot without outliers.
```

Out[24]: <matplotlib.axes._subplots.AxesSubplot at 0x1a11bbe0f0>

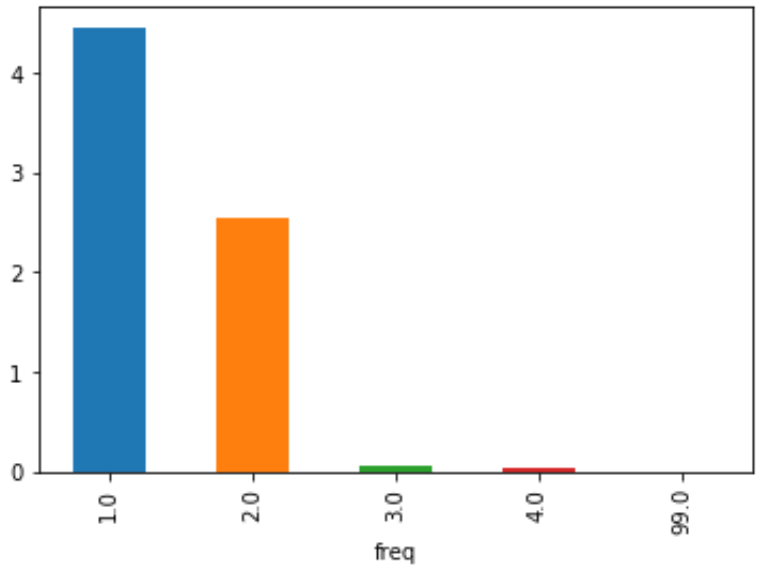


In [25]:
```
data_1.groupby('freq')['time'].mean()
```

Out[25]:
```
freq
1.0     4.444444
2.0     2.539286
3.0     0.055556
4.0     0.043478
99.0    0.000000
Name: time, dtype: float64
```

In [26]:
```
data_1.groupby('freq')['time'].mean().plot(kind = 'bar')
```

Out[26]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1a11b0a4e0>
```



In [27]:
```
#ANOVA Test
```

In [28]:
```
stats.f_oneway(data_1[data_1['freq'] == 1]['time'],data_1[data_1['freq']
               ,data_1[data_1['freq'] == 3]['time'], data_1[data_1['freq'
              data_1[data_1['freq'] == 99]['time'])
```

Out[28]:
```
F_onewayResult(statistic=4.475731155561718, pvalue=0.00245660061145134
68)
```

In [29]:
```
#Reject, different
```

In [ ]:
```

```

自己猜

In [30]:
```
#exam
```

In [ ]:
```

```

Scenario 3

In [31]:
```
# Interval average amount of time spent playing video games in the week p

# Overall shape of the distribution
```

In [32]:
```
#w/ formula in slide N = 314, n = 91
```

In [33]:
```
s = data_1['time'].std()
N = 314
n = 91
x_bar = data_1['time'].mean()
se = (s/np.sqrt(n)) * np.sqrt((N-n)/N)

x_bar-2*se, x_bar+ 2 * se
```

Out[33]:
```
(0.5652878396892033, 1.8934078124847096)
```

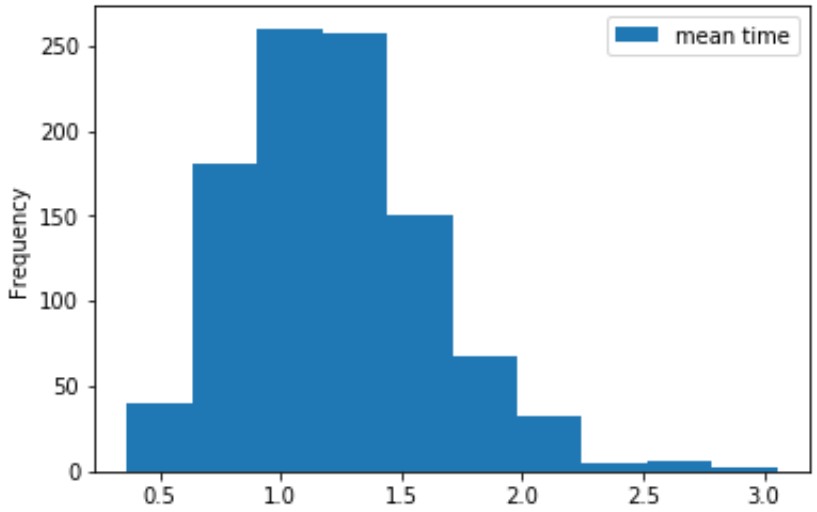
In [34]:
```
#w / bootstrap over 1000 Simulation
```

In [35]:
```
mean_time = []
for i in sample_pool:
    i['time'] = i['time'].replace(99, 0).replace(np.NaN, 0)
    mean_time += [i['time'].mean()]
np.percentile(mean_time, 2.5), np.percentile(mean_time, 97.5)
```

Out[35]:
```
(0.5880434782608696, 2.114266304347826)
```

In [36]:
```
pd.DataFrame(
    {'mean time':mean_time}
).plot(kind = 'hist')
```

Out[36]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1a11e682b0>
```



Scenario 4

不确定， 跳过

In [37]:
```
# Do you think students enjoy the game?

# list of the most important reason students like or dislike video games

# Nonrespondent who asked to skip the question
```

In [38]:
```
like_reason = data_2.iloc[0:,[i for i in range(1,12)]]
dislike_reason = data_2.iloc[0:, [i for i in range(12, 22)]]
```

In [39]:
```
like_reason.drop(columns = 'other').sum().sort_values(ascending = False)
```

Out[39]:
```
relax        58.0
strategy     55.0
action       45.0
sport        34.0
master       25.0
adv          25.0
bored        24.0
challenge    21.0
sim          15.0
coord         4.0
dtype: float64
```

In [40]:
```
dislike_reason.drop(columns = 'other2').sum().sort_values(ascending = Fal
```

Out[40]:
```
time       42.0
cost       35.0
point      29.0
frust      23.0
graphic    23.0
rules      17.0
boring     14.0
lonely      4.0
friends     2.0
dtype: float64
```

In [41]:
```
like_reason.other.value_counts()
```

Out[41]:
```
                83
competiveness    1
love it          1
lowers stress    1
fun              1
Brainless        1
excitement       1
like sports      1
addictive        1
Name: other, dtype: int64
```

In [42]:
```
dislike_reason.other2.value_counts()
```

Out[42]:
```
                        82
unproductive             4
the computer cheats      1
too realistic            1
hate losing              1
gives me blisters        1
do other things          1
Name: other2, dtype: int64
```

In [43]:
```
#other reasons are not dominant, so we omit it
```

In [44]:
```
#Education reason
```

In [45]:
```
len(data_1[data_1['educ'] == 1]) #37 students play for educational reason
```

Out[45]:
```
37
```

Scenario 5

In [46]:
```
#differences between those who like to play video games
#andthose who don't.

#use the questions in the last part of the survey,
#and make comparisons between male and female students,
#those who work for pay and those
#who don't, those who own a computer and those who don't
```

In [47]:
```
#Like to play: 1=never played, 2=very much, 3=somewhat,
# 4=not really, 5=not at all.

#Sex : 1=male, 0=female.
```

In [48]:
```
# like to play 2&3 vs 1&4&5&99&null
```

In [49]:
```
like = data_1[(data_1['like']==2) \
              |(data_1['like']==3)]
notlike = data_1[(data_1['like'] == 1) | \
                 (data_1['like'] == 4) |\
                 (data_1['like'] == 5) |\
                 (data_1['like'] == 99) |\
                 (data_1['like'].isnull())]
```

In [50]:
```
like.groupby('sex').size()
```

Out[50]:
```
sex
0.0    26
1.0    43
dtype: int64
```

In [51]:
```
sexpreftable = pd.DataFrame(
                {
                    'like': like.groupby('sex').size(),
                    'notlike': notlike.groupby('sex').size()
                }
)
sexpreftable.index =  ['female', 'male']
sexpreftable.index.name = 'sex'
sexpreftable.columns.name = 'pref'
ownpreftable = pd.DataFrame(
                {
                    'like': like.groupby('own').size(),
                    'notlike': notlike.groupby('own').size()
                }
)
ownpreftable.index = ['not_own', 'own']
ownpreftable.index.name = 'own'
ownpreftable.columns.name = 'pref'
```

In [52]:
```
#Graphical display and cross-tabulations
#are particularly helpful in making these kinds of comparisons.
```

In [53]:
```
sexpreftable
```

Out[53]:

pref	like	notlike
sex		
female	26	12
male	43	10

In [54]:
```
ownpreftable
```

Out[54]:

pref	like	notlike
own		
not_own	21	3
own	48	19

In [55]:
```
stats.chi2_contingency(sexpreftable.values)[1] #p_val of chi-square test
```

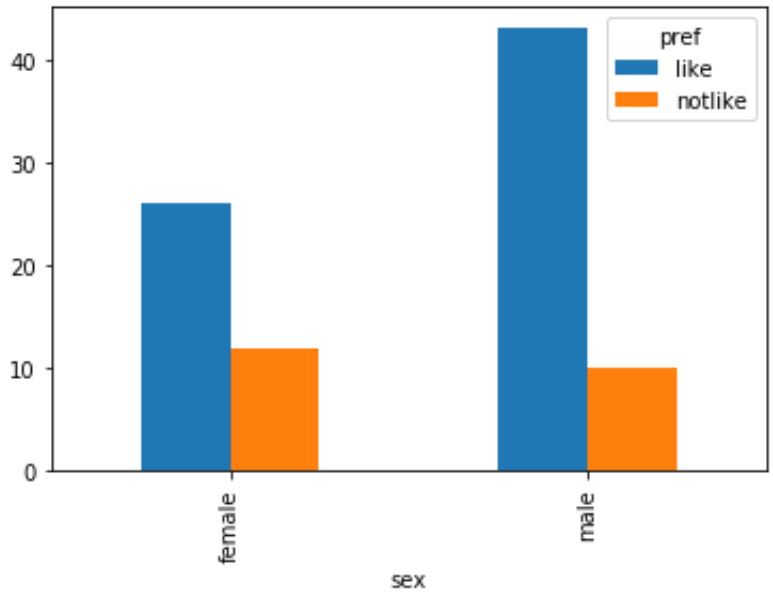
Out[55]:
```
0.250788205141197
```

In [56]:
```
stats.chi2_contingency(ownpreftable.values)[1] #p_val of chi-square test
```

Out[56]:
```
0.2008397383350607
```

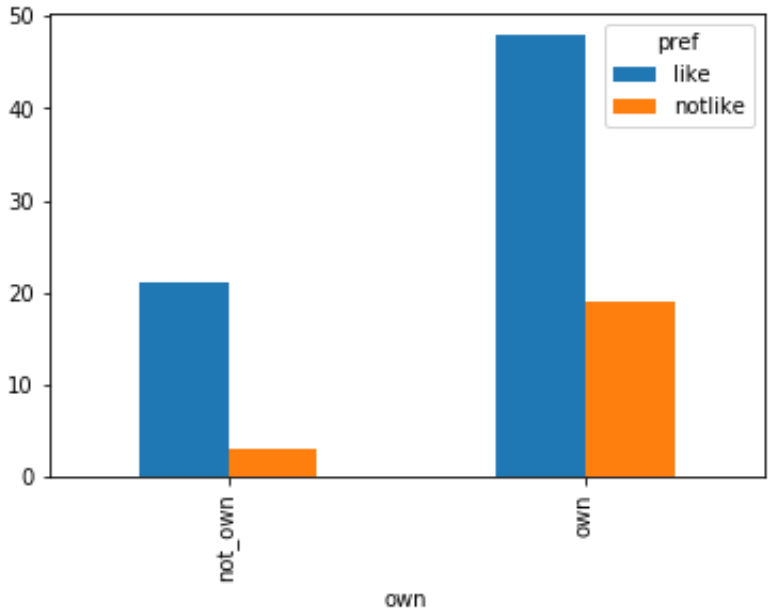
In [57]:
```
sexpreftable.plot(kind = 'bar')
```

Out[57]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1a11f91a90>
```



In [58]:
```
ownpreftable.plot(kind = 'bar')
```

Out[58]: <matplotlib.axes._subplots.AxesSubplot at 0x1a12088d68>



In [59]:
```
#Cross tab analysis
```

In [60]:
```
pd.crosstab(like.sex, like.own,margins=True)#like with own and sex attrib
```

Out[60]:

own	0.0	1.0	All
sex			
0.0	8	18	26
1.0	13	30	43
All	21	48	69

In [61]:
```
#chi-square test of independence
```

In [62]:
```
pd.crosstab(notlike.sex, notlike.own, margins = True)# not like with own
```

Out[62]:

own	0.0	1.0	All
sex			
0.0	3	9	12
1.0	0	10	10
All	3	19	22

In [63]:
```
#chi-square test of independence
```

Scenario 6

In [64]:
```
# Investigate the grade assignments of the gaming

#Need chi-square test
```

In [65]:
```
data_1.groupby('time')['grade'].mean()
```

Out[65]:
```
time
0.0     3.140351
0.1     4.000000
0.5     3.400000
1.0     3.400000
1.5     3.000000
2.0     3.642857
3.0     3.000000
4.0     4.000000
5.0     4.000000
14.0    2.500000
30.0    3.000000
Name: grade, dtype: float64
```

In [66]:
```
data_1.groupby('freq')['grade'].mean()
```

Out[66]:
```
freq
1.0     3.444444
2.0     3.500000
3.0     2.888889
4.0     3.217391
99.0    3.153846
Name: grade, dtype: float64
```

In [67]:
```
data_1.groupby('like')['grade'].mean()
```

Out[67]:
```
like
1.0     4.000000
2.0     3.260870
3.0     3.239130
4.0     3.461538
5.0     2.857143
99.0    3.000000
Name: grade, dtype: float64
```

In [68]:
```
#Bootstrap data to confirm
```

In [69]:
```
time_mean = []
freq_mean = []
like_mean = []
for i in sample_pool:
    time_mean += [data_1.groupby('time')['grade'].mean()]
    freq_mean += [data_1.groupby('freq')['grade'].mean()]
    like_mean += [data_1.groupby('like')['grade'].mean()]
print(sum(time_mean)/500, sum(freq_mean)/500, sum(like_mean)/500)
```

```
time
0.0     6.280702
0.1     8.000000
0.5     6.800000
1.0     6.800000
1.5     6.000000
2.0     7.285714
3.0     6.000000
4.0     8.000000
5.0     8.000000
14.0    5.000000
30.0    6.000000
Name: grade, dtype: float64 freq
1.0     6.888889
2.0     7.000000
3.0     5.777778
4.0     6.434783
99.0    6.307692
Name: grade, dtype: float64 like
1.0     8.000000
2.0     6.521739
3.0     6.478261
4.0     6.923077
5.0     5.714286
99.0    6.000000
Name: grade, dtype: float64
```

In [ ]: