Q: I decided to treat this as a classification problem by creating a new binary variable affair (did the woman have at least one affair?) and trying to predict the classification for each woman.

In [68]:
```
# Import Packages
import numpy as np
import pandas as pd
import statsmodels.api as sm
import matplotlib.pyplot as plt
from patsy import dmatrices
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.cross_validation import cross_val_score
```

In [69]:
```
dta = sm.datasets.fair.load_pandas().data
dta['affair'] = (dta.affairs > 0).astype(int)
```

In [70]:
```
dta.head()
```

Out[70]:

	rate_marriage	age	yrs_married	children	religious	educ	occupation	occupation_husb	affa
0	3.0	32.0	9.0	3.0	3.0	17.0	2.0	5.0	0.111
1	3.0	27.0	13.0	3.0	1.0	14.0	3.0	4.0	3.2307
2	4.0	22.0	2.5	0.0	1.0	16.0	3.0	5.0	1.4000
3	4.0	37.0	16.5	4.0	3.0	16.0	5.0	5.0	0.7272
4	5.0	27.0	9.0	1.0	1.0	14.0	3.0	4.0	4.6666

In [71]:
```
# Exploration of data

dta.groupby('affair').mean()
```

Out[71]:

	rate_marriage	age	yrs_married	children	religious	educ	occupation	occupatio
affair								
0	4.329701	28.390679	7.989335	1.238813	2.504521	14.322977	3.405286	3.
1	3.647345	30.537019	11.152460	1.728933	2.261568	13.972236	3.463712	3.

In [72]:
```
dta.groupby('rate_marriage').mean()
```

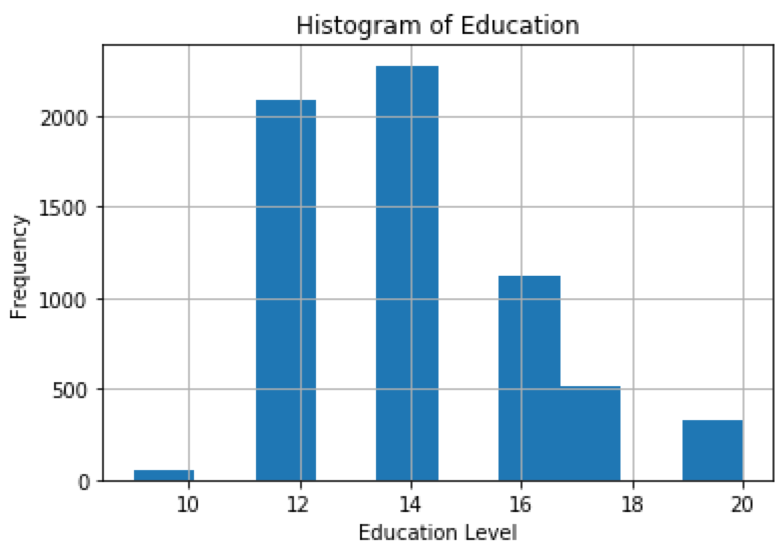
Out[72]:

	age	yrs_married	children	religious	educ	occupation	occupation_husb
rate_marriage							
1.0	33.823232	13.914141	2.308081	2.343434	13.848485	3.232323	3.838384
2.0	30.471264	10.727011	1.735632	2.330460	13.864943	3.327586	3.764368
3.0	30.008056	10.239174	1.638469	2.308157	14.001007	3.402820	3.798590
4.0	28.856601	8.816905	1.369536	2.400981	14.144514	3.420161	3.835861
5.0	28.574702	8.311662	1.252794	2.506334	14.399776	3.454918	3.892697

In [73]:
```
%matplotlib inline
```

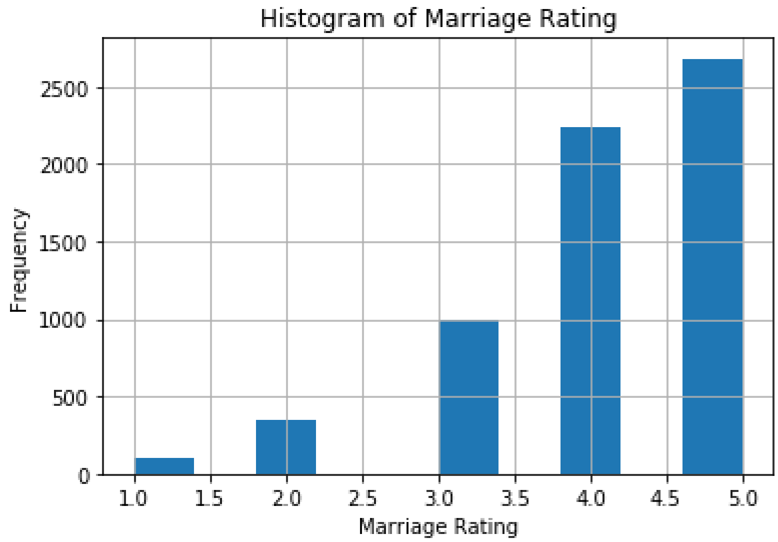
In [74]:
```
# Histograms of education and marriage rating.
dta.educ.hist()
plt.title('Histogram of Education')
plt.xlabel('Education Level')
plt.ylabel('Frequency')
```

Out[74]: Text(0,0.5,'Frequency')



In [75]:
```
# histogram of marriage rating
dta.rate_marriage.hist()
plt.title('Histogram of Marriage Rating')
plt.xlabel('Marriage Rating')
plt.ylabel('Frequency')
```

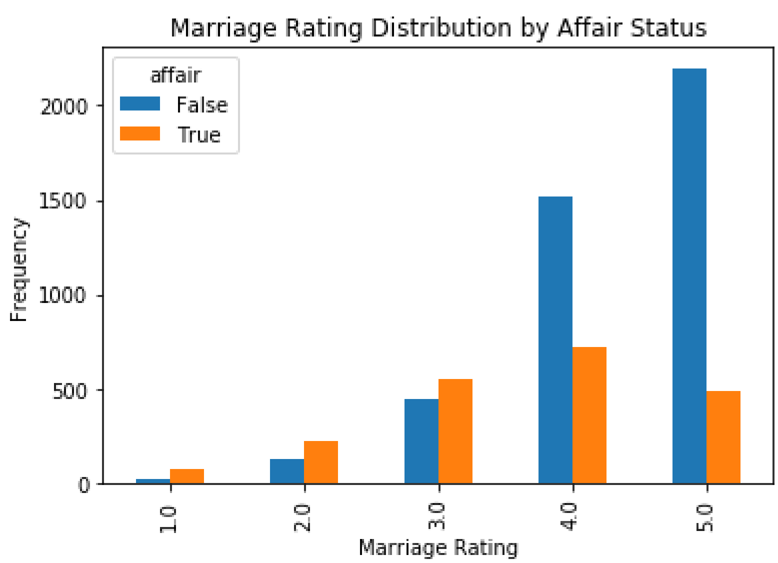
Out[75]: Text(0,0.5,'Frequency')



In [76]:
```
#affairs versus NO affairs Distribution

# barplot of marriage rating grouped by affair (True or False)
pd.crosstab(dta.rate_marriage, dta.affair.astype(bool)).plot(kind='bar')
plt.title('Marriage Rating Distribution by Affair Status')
plt.xlabel('Marriage Rating')
plt.ylabel('Frequency')
```

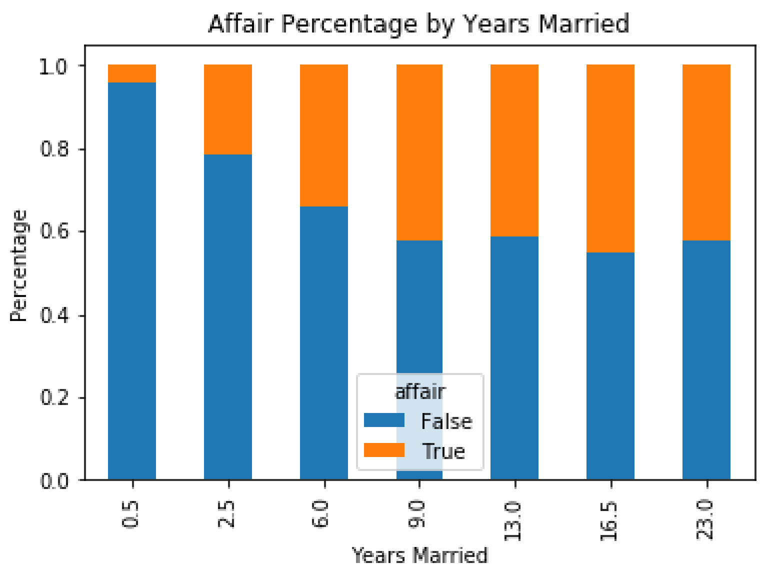
Out[76]: Text(0,0.5,'Frequency')



In [77]:
```
#Affair Percentage by Years Married

affair_yrs_married = pd.crosstab(dta.yrs_married, dta.affair.astype(bool))
affair_yrs_married.div(affair_yrs_married.sum(1).astype(float), axis=0).plot(kind
plt.title('Affair Percentage by Years Married')
plt.xlabel('Years Married')
plt.ylabel('Percentage')
```

Out[77]: Text(0,0.5,'Percentage')



In [78]:
```
#dummy variables for occupation and occupation_husb
y, X = dmatrices('affair ~ rate_marriage + age + yrs_married + children + \
                  religious + educ + C(occupation) + C(occupation_husb)',
                  dta, return_type="dataframe")
print( X.columns)
```

```
Index(['Intercept', 'C(occupation)[T.2.0]', 'C(occupation)[T.3.0]',
       'C(occupation)[T.4.0]', 'C(occupation)[T.5.0]', 'C(occupation)[T.6.0]',
       'C(occupation_husb)[T.2.0]', 'C(occupation_husb)[T.3.0]',
       'C(occupation_husb)[T.4.0]', 'C(occupation_husb)[T.5.0]',
       'C(occupation_husb)[T.6.0]', 'rate_marriage', 'age', 'yrs_married',
       'children', 'religious', 'educ'],
      dtype='object')
```

In [79]:
```
# rename column names
X = X.rename(columns = {'C(occupation)[T.2.0]':'occ_2',
                        'C(occupation)[T.3.0]':'occ_3',
                        'C(occupation)[T.4.0]':'occ_4',
                        'C(occupation)[T.5.0]':'occ_5',
                        'C(occupation)[T.6.0]':'occ_6',
                        'C(occupation_husb)[T.2.0]':'occ_husb_2',
                        'C(occupation_husb)[T.3.0]':'occ_husb_3',
                        'C(occupation_husb)[T.4.0]':'occ_husb_4',
                        'C(occupation_husb)[T.5.0]':'occ_husb_5',
                        'C(occupation_husb)[T.6.0]':'occ_husb_6'})
```

In [80]:
```
y = np.ravel(y)
y
```

Out[80]:
```
array([1., 1., 1., ..., 0., 0., 0.])
```

In [81]:
```
# logistic regression model to fit X and y
model = LogisticRegression()
model = model.fit(X, y)
# accuracy on the training set
model.score(X, y)
```

Out[81]:
```
0.7258875274897895
```

73% accuracy

In [82]:
```
y.mean()
```

Out[82]:
```
0.3224945020420987
```

32% of women had affairs.

In [83]:
```
# coefficients
pd.DataFrame(list(zip(X.columns, np.transpose(model.coef_))))
```

Out[83]:

	0	1
0	Intercept	[1.489835891324933]
1	occ_2	[0.18806639024440983]
2	occ_3	[0.4989478668156914]
3	occ_4	[0.25066856498524825]
4	occ_5	[0.8390080648117001]
5	occ_6	[0.8339084337443315]
6	occ_husb_2	[0.1906359445867889]
7	occ_husb_3	[0.2978327129263421]
8	occ_husb_4	[0.161408854076061 6]
9	occ_husb_5	[0.18777091388972483]
10	occ_husb_6	[0.19401637225511495]
11	rate_marriage	[-0.7031233597323255]
12	age	[-0.058417774481 68919]
13	yrs_married	[0.10567653799735635]
14	children	[0.016919266970905608]
15	religious	[-0.3711362653137546]
16	educ	[0.00401650319563816]

In [84]:
```
#Model Evaluation

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_st
model2 = LogisticRegression()
model2.fit(X_train, y_train)
```

Out[84]:
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=1,
          penalty='l2', random_state=None, solver='liblinear', tol=0.0001,
          verbose=0, warm_start=False)
```

In [85]:
```
# class labels for the test set predicition
predicted = model2.predict(X_test)
print( predicted)
```

```
[1. 0. 0. ... 0. 0. 0.]
```

In [86]:
```
# class probabilities generation
probs = model2.predict_proba(X_test)
print (probs)
```

```
[[0.3514634  0.6485366 ]
 [0.90955084 0.09044916]
 [0.72567333 0.27432667]
 ...
 [0.55727385 0.44272615]
 [0.81207043 0.18792957]
 [0.74734601 0.25265399]]
```

In [87]:
```
#  evaluation metrics generation
print(metrics.accuracy_score(y_test, predicted))
print(metrics.roc_auc_score(y_test, predicted))
```

```
0.7298429319371728
0.6339179260634122
```

73% is the same as we experienced when training

In [88]:
```
print (metrics.confusion_matrix(y_test, predicted))
print (metrics.classification_report(y_test, predicted))
```

```
[[1169  134]
 [ 382  225]]
             precision    recall  f1-score   support

        0.0       0.75      0.90      0.82      1303
        1.0       0.63      0.37      0.47       607

avg / total       0.71      0.73      0.71      1910
```

In [89]:
```
# model evaluation using 10-fold cross-validation
scores = cross_val_score(LogisticRegression(), X, y, scoring='accuracy', cv=10)
print(scores)
print(scores.mean())
```

```
[0.72100313 0.70219436 0.73824451 0.70597484 0.70597484 0.72955975
 0.7327044  0.70440252 0.75157233 0.75      ]
0.7241630685514876
```

Looks good. It's still performing at 73% accuracy.

In [ ]: